PREDICCIÓN DEL ÉXITO ACADÉMICO EN EDUCACIÓN SUPERIOR UTILIZANDO ÁRBOLES DE DECISIÓN

Miguel Ángel Sarmiento Aguiar
Universidad Eafit
Colombia
msarmie4@eafit.edu.co

Marlon Pérez Ríos
Universidad Eafit
Colombia
mperezr@eafit.edu.co

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

RESUMEN
El objetivo de este informe es mostrar una predicción al puntaje de los estudiantes de pregrado en las universidades colombianas en las pruebas saber pro y mejorar lo que se podría sacar en ellas, ya que esto servirá para aconsejar a los jóvenes antes de tomar la decisión de escoger una carrera, y así salgan más preparados para presentarla y ayuden al desarrollo del país aumentando el nivel académico de este a nivel mundial.

Se busca poder predecir el posible éxito de la población estudiantil en algún programa de educación superior, y con esta predicción, aconsejar respecto a una decisión posiblemente más acertada.

Para la solución de esto se realiza un algoritmo basado en árboles de decisión que reciba una base de datos de entrenamiento y cree el árbol según la impureza de los subconjuntos de datos creados para cada una de las variables y valores que el algoritmo decida tomar. Con dicho árbol creado, se le ingresa otro conjunto de datos para que el algoritmo determine cuantos estudiantes tendrían éxito y cuantos no en las pruebas de educación superior.

Finalmente, se comparan los resultados del algoritmo con los resultados reales del conjunto de datos, obteniendo para cada uno de ellos siempre errores por debajo del 10%.

Palabras clave
- Estructuras de datos.
- Árboles de decisión.
- Predicción éxito académico.

Palabras clave de la clasificación de la ACM
- Modelos de computación.
- Diseño y análisis de algoritmos.
- Sistemas de gestión de datos.
- Aprendizaje automático.
- Gráficos de computadora.
  [1]

1. INTRODUCCIÓN
En la actualidad el nivel de las pruebas saber pro en la población de jóvenes estudiantes de pregrado es baja, con respecto a otras universidades del mundo, debido a que la mayoría al finalizar la educación secundaria no sabe en qué área poseen mejores competencias, por esto toman una decisión con mayor probabilidad de ser errónea respecto a la probabilidad de éxito. Casi en la totalidad de los casos dicha decisión se toma basándose en el gusto por alguna profesión o por el deseo de una gran remuneración monetaria.

Por esto se pretende mostrar la metodología por la cual se va a desarrollar un algoritmo basado en árboles de decisión, con el fin de predecir el posible éxito de los jóvenes en las diferentes carreras universitarias, reflejándose en las pruebas saber pro, todo esto en base a la información de variables sociodemográficas y de las pruebas saber 11 suministrada por el gobierno colombiano en los últimos años.

2. PROBLEMA
El problema por resolver es la creación de un algoritmo basado en árboles de decisión y en los datos suministrados por el ICFES, mencionados anteriormente. La solución a este problema podrá predecir si el puntaje en las pruebas Saber Pro estará o no por encima del promedio, teniendo en cuenta que los datos del ICFES no solo incluyen variables académicas como el puntaje sino también variables sociodemográficas como género, edad, información de los padres, estrato, entre muchas otras.

Así, se pretende disminuir el nivel de deserción de los estudiantes de pregrado y aumentar los puntajes en la prueba saber pro, guiándolos por la opción en la que tienen mayor probabilidad de éxito, así se puede ahorrar una gran cantidad de recursos tanto para ellos como para el gobierno, añadiendo que con dicha predicción se puede aumentar el interés y las ganas por el estudio, incrementando el número de posibles profesionales en el país.

3. TRABAJOS RELACIONADOS

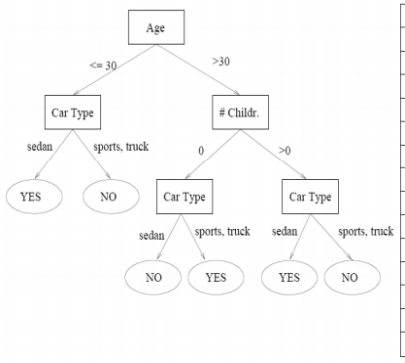
3.1 Algoritmo CART
Es una técnica de aprendizaje de árbol de decisión no paramétrica que produce árboles de clasificación o regresión, dependiendo de si la variable dependiente es categórica o numérica, respectivamente. La palabra binario implica que un nodo en un árbol de decisión solo puede dividirse en dos grupos. CART utiliza el índice de Gini como medida de impureza para seleccionar el atributo. El atributo con la mayor reducción de impurezas se utiliza para dividir los registros del nodo. CART acepta datos con valores numéricos o categóricos y también maneja valores

de atributos faltantes. Utiliza la poda de complejidad de costos y también genera árboles de regresión.

El análisis de árboles de clasificación y regresión (CART) generalmente consiste en tres pasos:

1. Construcción del árbol máximo.
2. Poda del árbol.
3. Selección del árbol óptimo mediante un procedimiento de validación cruzada.

Como ejemplo se supone el árbol y los datos en la Figura 6, donde se quiere determinar un conjunto de reglas que indiquen si un conductor vive o no en los suburbios.



Car type	Age	Chd	Sub
sedan	23	0	yes
sports	31	1	no
sedan	36	1	no
truck	25	2	no
sports	30	0	no
sedan	36	0	no
sedan	25	0	yes
truck	36	1	no
sedan	30	2	yes
sedan	31	1	yes
sports	25	0	no
sedan	45	1	yes
sports	23	2	no
truck	45	0	yes

Figura 1 - Ejemplo CART

Se concluye:

- Si Age ≤ 30 y CarType = Sedan entonces Si
- Si Age ≤ 30 y CarType = truck/Sports entonces No
- Si Age > 30, Children = 0 y CarType = Sedan entonces No
- Si Age > 30, Children = 0 y CarType = truck/Sports entonces Si
- Si Age > 30, Children > 0 y CarType = Sedan entonces Si
- Si Age > 30, Children > 0 y CarType = truck/Sports entonces No

[2]

4. Estructura de datos arreglo de arreglos

Se decide crear una estructura de datos que sea un arreglo de arreglos, donde cada estudiante es un arreglo que contiene las variables de estudio como trabajo de la madre, tipo de piso, colegio, estrato, entre otros.

	estu_consecutivo.1	estu_exterior	periodo	...	exito
Estudiante 1	1-1	1-2	1-3	...	1-m
Estudiante 2	2-1	2-2	2-3	...	2-m
Estudiante 3	3-1	3-2	3-3	...	3-m
...	...	...	...	...	...
Estudiante n	n-1	n-2	n-3	...	n-m

Figura 2 - Estructura de datos

4.1 Operaciones de la estructura de datos

Para solucionar el problema eficientemente, primero se decide crear una estructura de datos que contenga toda la información de la base de datos como texto, después se divide dicha estructura en arreglos, cada uno de los cuales es una fila de la base, es decir, cada fila es un estudiante. Luego se procede a dividir cada una de estas filas en arreglos, cada uno de los cuales es uno de los atributos de cada estudiante.

La operación de llenado de la estructura de datos se puede resumir en la siguiente figura:

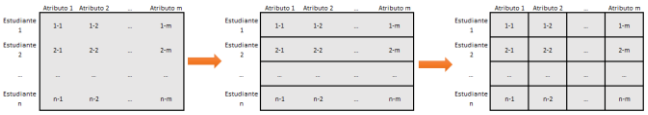


Figura 3 - Llenar estructura de datos

La otra operación de la estructura de datos es acceder a un elemento de ella, lo cual, por ser un arreglo de arreglos, no implica recorrer el número de filas o el número de columnas completamente, como se muestra en la siguiente figura:

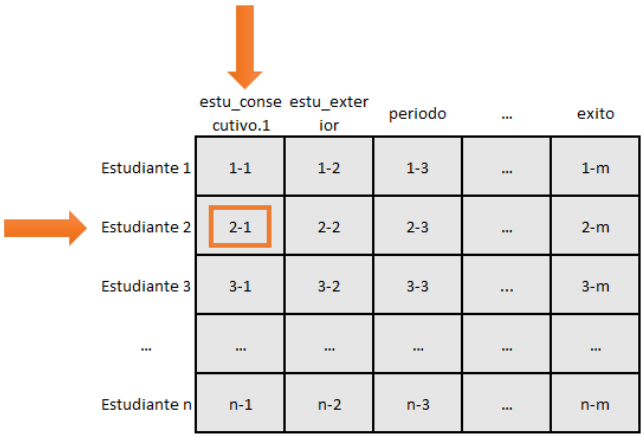


Figura 4 - Acceder a estructura de datos

El algoritmo primeramente crea la estructura de datos de entrenamiento (Train) y de prueba (Test) de la forma descrita anteriormente. Posteriormente se crea un árbol de decisión en base al conjunto de datos de entrenamiento, diciéndole al algoritmo cuál será la altura del árbol, ya que no sería correcto que el árbol tenga demasiados niveles, ya que se puede evidenciar que se llega a un punto en el cual, si se sigue aumentando el número de niveles, hay tan pocos datos en ellos y tantas preguntas que hacer que el error del algoritmo aumenta. Después, como agregado para el proyecto, se decide que el algoritmo cree un código para poder dibujar el árbol creado a través de la página web WebGraphviz. Luego, el algoritmo recorre el conjunto de datos de prueba y mira cuáles son los datos reales de éxito de los estudiantes, y se calcula el éxito de cada uno de los datos de este mismo conjunto en base a las probabilidades y a la manera en que el árbol de decisión esté ya construido. Posteriormente se comparan ambos resultados (éxito real y éxito pronosticado) para calcular el porcentaje de error del algoritmo, y posteriormente imprimir todos los resultados relevantes al usuario. o y cuantos no en las pruebas de educación superior.

Adicionalmente, se diseña el código para ser ejecutado bajo dos modalidades: La primera, que es la principal, carga una estructura de datos de prueba completa y determina cuantos de los estudiantes tendrían éxito, y los compara con los datos reales; la segunda, carga solo el estudiante que el usuario escoja, y predice si dicho estudiante tendrá éxito o no, y cuál será la probabilidad de que esto pase.

4.2 Criterios de diseño de la estructura de datos

La estructura de datos se diseñó para reducir el tiempo de ejecución y la memoria utilizada al ejecutar el código en Python, ya que se utilizaron operaciones evitando el aumento de la complejidad de la variable n, que corresponde al número de filas (estudiantes). Esto es importante porque dicha variable tiene un valor considerable entre 15000 (archivo más pequeño) y 135000 (archivo más grande), por lo cual puede aumentar significativamente la memoria y el tiempo de ejecución si se multiplica por ella misma.

De esta forma, se realiza el algoritmo de tal forma que reduzca al máximo tener que recorrer el número de filas del conjunto de datos, para evitar que la complejidad debida a este número (que es la variable más relevante) crezca.

4.3 Análisis de Complejidad

Según el algoritmo obtenido, el cálculo de la complejidad de todas sus operaciones queda de la siguiente manera, resaltando en negrilla la complejidad del algoritmo total (main), la complejidad del entrenamiento del árbol (Arbol_Binario) y la complejidad del cálculo del éxito (Calcular_Exito_Colectivo):

FUNCIÓN	COMPLEJIDAD
Crear_Estructura_Datos(Direccion_csv)	O(n*m)
Dividir_Matriz_En_Dos(Matriz, Pos_Variable, Valor)	O(n)
Calcular_Valores_Variable(Matriz, Pos_Variable)	O(n*log(n))
Calcular_Mejor_Variable(Matriz)	O(n*m*log(n))
Arbol_Binario(Matriz, Nivel_Maximo)	**O(n*m*log(n)*2^m)**
Calcular_Probabilidad_Exito(Matriz)	O(n)
Calcular_Pronostico_Exito(Fila, Arbol)	O(m)
Calcular_Exito_Colectivo(Matriz, Arbol)	**O(n*m)**
Calcular_Exito_Individual(Matriz, Arbol)	O(m)
main(Train, Test, Nivel_Arbol, Fila)	**O(n*m*log(n)*2^m)**

Tabla 1 - Complejidad del algoritmo

4.4 Tiempos de Ejecución

Se calcula el tiempo que se demora el algoritmo completo en ejecutarse para cada uno de los conjuntos de datos de entrenamiento y prueba.

CONJUNTO DE DATOS	NÚMERO DE FILAS	TIEMPO DE EJECUCIÓN [Segundos]
Train 0 - Test 0	20000	3,698
Train 1 - Test 1	60000	14,954
Train 5 - Test 5	77020	20,137
Train 2 - Test 2	100000	28,569
Train 3 - Test 3	140000	42,354
Train 4 - Test 4	180000	57,242

Tabla 2 - Tiempos de ejecución

4.5 Memoria

Se calcula la memoria utilizada por el ordenador para cada uno de los conjuntos de datos de entrenamiento y prueba.

CONJUNTO DE DATOS	NÚMERO DE FILAS	CONSUMO DE MEMORIA [MB]
Train 0 - Test 0	20000	20,48
Train 1 - Test 1	60000	20,48
Train 5 - Test 5	77020	30,72
Train 2 - Test 2	100000	40,96
Train 3 - Test 3	140000	51,20
Train 4 - Test 4	180000	61,44

Tabla 3 - Consumo de memoria

4.6 Análisis de los resultados

El resumen de datos calculados se puede observar en la siguiente tabla:

CONJUNTO DE DATOS	NÚMERO DE FILAS	TIEMPO DE EJECUCIÓN [Segundos]	CONSUMO DE MEMORIA [MB]
Train 0 - Test 0	20000	3,698	20,48
Train 1 - Test 1	60000	14,954	20,48
Train 5 - Test 5	77020	20,137	30,72
Train 2 - Test 2	100000	28,569	40,96
Train 3 - Test 3	140000	42,354	51,20
Train 4 - Test 4	180000	57,242	61,44

Tabla 4 - Resumen datos calculados

Adicionalmente, se puede observar el comportamiento de estas dos variables calculadas según el número de filas ingresadas, como se muestra en la siguiente figura, la cual indica un comportamiento lineal en esta variable:

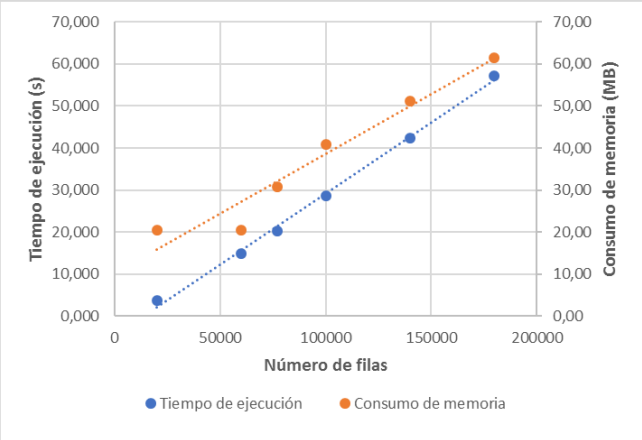


Figura 5 - Resumen comportamiento

5. CONCLUSIONES

A lo largo de la realización del proyecto se pudo evidenciar que la complejidad de los algoritmos juega un valor muy importante que comúnmente pasamos por alto, ya que, por lo general, al tratar con conjuntos de datos relativamente pequeños, no notamos un posible mal diseño en el programa. En cambio, si se trabaja con conjuntos de datos grandes, pequeños cambios en el algoritmo muestran grandes diferencias en tiempos de ejecución y consumo de memoria.

Como se puede observar es una buena solución ya que se obtienen buenos tiempos de ejecución (menores a 1 minuto) trabajando con arreglos de arreglos en Python, además se requiere de poca memoria RAM (61.44Mb para el archivo más grande). Por último, los errores calculados entre los datos reales y el pronóstico del algoritmo están en su mayoría por debajo del 10%, trabajando con árbol binario de búsqueda hasta nivel 6.

Adicionalmente, se puede evidenciar que, a partir del séptimo nivel del árbol de decisión, el error del algoritmo comienza a aumentar, por lo que para conjuntos de datos de tamaños similares a los utilizados anteriormente se recomienda que el árbol de decisión no supere esta cantidad de niveles.

5.1 Trabajos futuros

Como trabajo futuro se podría pensar en la implementación del algoritmo en una idea de negocio. Por ejemplo, se puede prestar el servicio a empresas o entidades que se dediquen a otorgar becas a posibles estudiantes de educación superior de predecir si dicho estudiante va a tener éxito o no en dicha educación, y ya la entidad encargada tomaría la decisión de darle o no la beca al estudiante. Además, también se podría ofrecer el servicio a las mismas universidades, para que estas realicen una predicción del éxito de sus estudiantes a recibir, y con esta predicción determinar a quienes aceptarán y a quienes no.

AGRADECIMIENTOS

Agradecemos a nuestros compañeros monitores del curso, quienes nos acompañaron desde el principio del curso y estuvieron atentos en todo momento ante cualquier eventualidad o duda que tuviéramos acerca del proyecto.

REFERENCIAS

1. ACM. (2012). ACM Computing Classification System. [online] Available at: https://dl.acm.org/ccs
2. Serna, S. (2009). Comparación de árboles de regresión y clasificación y regresión logística. Maestría. Universidad Nacional de Colombia.