ANÁLISIS PREDICTICO PARA LAS PRUEBAS SABER PRO

Luis Fernando Vargas Agudelo
Universidad Eafit
Colombia
lvarga12@eafit.edu.co

Tomás Bedoya Henao
Universidad Eafit
Colombia
Tbedoyah@eafit.edu.co

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

RESUMEN
En el presente informe se presentan diferentes metodologías para el análisis predictivo que puedan dar respuesta a la problemática de poder conocer, analizar y predecir si un estudiante según sus resultados en la prueba de conocimiento saber 11 si será exitoso en los resultados de la prueba Saber Pro, esto con el fin de comprender cuales pueden ser las falencias y las variables que influyen directamente en estos resultados.

Específicamente se explica, a partir de una lista de listas como estructura de datos, cómo se puede predecir el éxito de un estudiante dado, mediante un modelo de machine learning basado en el algoritmo para árboles de decisión CART (De Clasificación y regresión), haciendo un recorrido a través de sus métodos y las clases necesarias para su implementación; así mismo, los resultados obtenidos, que son el objetivo principal del desarrollo del modelo. Es de vital importancia el empleo de modelos de Machine Learning para solucionar problemas de la vida real y de esa forma reducir la incertidumbre en muchos ámbitos, como el de inversiones o mantenimiento de maquinaria; resulta entonces fundamental e imperante el conocimiento de este tipo de modelos para aportar en gran medida a la sociedad, en términos más generales; en el caso particular de las pruebas saber PRO, es importante desarrollar modelos de gran ajuste predictivo, pues sirven para la toma de decisiones que repercute directamente a la calidad del capital humano o recursos humanos que, como es bien sabido, son un factor fundamental a la hora de determinar variables económicas importantes como el pib per-cápita u otras medidas que buscan cuantificar la calidad de vida de un Estado o nación.

1. INTRODUCCIÓN

El considerable aumento de datos que se generan día a día con la interacción de las personas y los instrumentos tecnológicos, el análisis masivo de los datos se ha convertido en una herramienta cada vez más importante para las empresas y los países.

Gobiernos como China han comenzado a utilizar el análisis de datos para obtener puntajes de sus habitantes y de esta forma saber quiénes son aquellos de los cuales pueden tener más beneficios sociales, económicos, legales entre otros. Esta situación ha generado grandes escándalos éticos en los que se pone en duda la libertad de expresión y social.

En Colombia el análisis de los datos está comenzando a ser algo muy importante para las entidades académicas, estas mediante un análisis predictivo desean conocer cuál será el comportamiento de sus estudiantes en el transcurso del tiempo y de alguna manera predecir si sus resultados en pruebas futuras serán exitosos o no. Soluciones así impulsarían fuertemente a Colombia en la región, ya que actualmente los índices de educación media y superior no son los mejores, además estudios demuestran que el 45% de los estudiantes no están seguros de que la carrera seleccionada es realmente lo que les gusta, lo cual lleva a un alza en la deserción de la educación superior la cual se encuentra cerca del 43%.

Poder crear soluciones que contribuyan con estas situaciones tienen gran importancia para lo sociedad, de esta forma diferentes entidades podrán conocer en que tienen falencias las personas y más importante aún, en que son realmente buenas, permitiendo así que con herramientas de aprendizaje se pulan sus puntos fuertes y obtener el mayor beneficio para ellos mismos y la sociedad.

El objetivo de este informe es conocer, entender y analizar cuáles métodos existen para ayudar a darle una solución a esta necesidad y darle al lector una idea clara de su funcionamiento.

2. PROBLEMA

La necesidad que se desea solucionar es poder conocer mediante un análisis predictivo si un estudiante de educación superior tendrá éxito o no en las pruebas Saber Pro. para calificar los resultados de la prueba como exitosos, el puntaje global obtenido debe estar por encima del promedio general.

Para poder predecir el éxito o no se utilizarán variables predictoras como el estado socioeconómico cuando el individuo presentó las pruebas saber 11 y el desempeño obtenido en las mismas.

Resolver esta problemática tiene diferentes objetivos, algunos pueden ser conocer las capacidades y falencias que tienen los estudiantes, identificar cuáles pueden ser gustos profesionales y permitir crear una educación un poco más personalizada lo que traería grandes beneficios para los diferentes entes interesados.

3. METODOLOGÍAS

Antes de conocer cuáles son las diferentes soluciones que existen para este tipo de problemáticas es conveniente

conocer cuál es el funcionamiento general y entender el flujo del análisis predictivo.

Como su nombre lo indica el análisis predictivo se basa en realizar predicciones basadas en los datos, este proceso se realiza con técnicas estadísticas y aprendizaje automático con el fin de crear un modelo predictivo. Con esto modelos es posible predecir cosas como ¿Cuál es la posibilidad de que un cliente regrese por un nuevo producto financiero? o ¿En cuánto tiempo una máquina podría necesitar un cambio de pieza?

Los diferentes modelos predictivos cumplen un simple flujo general: recolección de los datos, preprocesamiento de los datos, desarrollo del modelo predictivo y finalmente la salida de estos modelos es integrada con sistemas analíticos para la lectura.

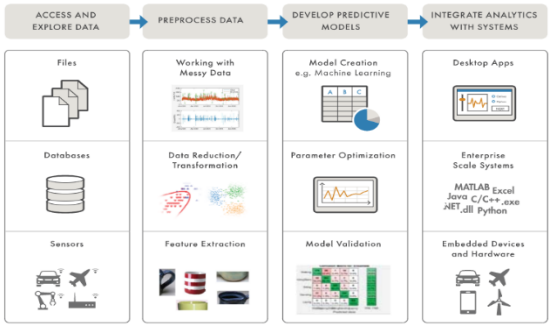

Ilustración 1 Flujo de trabajo de análisis predictivo. Tomado de:MathWorks, predictive-analytics

La manera más común y simple de realizar pronósticos para variables aleatorias, es la **regresión lineal**, que permite investigar las relaciones entre una variable dependiente y otras independientes mediante la formulación de ecuaciones matemáticas y procesos de optimización. De esta manera, se puede encontrar una línea (Función) que se ajuste lo mejor posible a los datos que se tienen o sobre los cuales se pretende trabajar en una predicción.

Los modelos **autorregresivos** o AR buscan generar pronósticos del proceso modelado basándose en el valor que haya tomado en momentos anteriores, ya que usualmente los datos están ordenados cronológicamente. En otras palabras, la variable dependiente y la variable explicativa son la misma, pero en diferentes momentos.

Por otro lado, **las redes neuronales** artificiales buscan imitar el proceso de aprendizaje neuronal biológico. Dichas neuronas artificiales se conectan entre sí y 'aprenden' con la ejecución del proceso, en esencia, dados unos parámetros o estimadores, se busca llegar a un resultado determinado mediante la combinación de estos (Condicionada por pesos o funciones), saber qué combinación produce x o y resultado deseado, es el problema que pretenden solucionar las redes neuronales.

Por su parte, los **árboles de decisión** permiten seleccionar matemáticamente la opción más beneficiosa, pues en general un árbol de decisión es un mapa que contiene las diferentes opciones junto con los posibles resultados generados al escoger entre las opciones mapeadas, una tras otra. Por lo general comienza con un único nodo que se bifurca o se ramifica, generando otros nodos adicionales que se vuelven a ramificar, todos atados a probabilidades, costos o beneficios y de esta manera, la metodología permite encontrar el camino óptimo. Existen tres tipos de nodos: de decisión, de probabilidades y nodos de finalización.

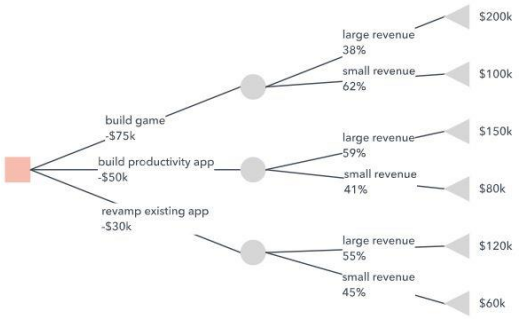

Ilustración 2 Mapa árbol de decisión. Tomado de: Árbol de decisión, lucidchart

3. TRABAJOS RELACIONADOS

Para desarrollar la solución al problema planteado en el inicio del informe se realizará un árbol de decisión, debido a esto es importante conocer cuáles son las diferentes formas que existen para desarrollarlos y los problemas similares que se han solucionado. A continuación, se profundizará en esto. Es importante recalcar que la diferencia entre los diferentes algoritmos radica, principalmente son las estrategias para 'podar' el árbol o reglas empleadas para dividir los nodos.

3.1 Algoritmo ID3

Las soluciones que se realizan con este algoritmo están basadas en nodos de decisión que a su vez están asociados a uno de los atributos, estos nodos cuentan con dos o más ramas que representan posibles valores del atributo. También está conformado por nodos-hojas, que representan al atributo objetivo que se quiere clasificar, pero todos al mismo tiempo, por lo que representa la decisión final del árbol. Este algoritmo utiliza el concepto de ganancia de información para seleccionar los atributos más útiles en cada iteración, mediante la entropía, pues a mayor homogeneidad de una muestra, más tiende a cero ésta. Si la muestra está igualmente distribuida, la entropía es igual a cero.

De esta manera, el algoritmo basa la ganancia e información en el decremento de la entropía. El atributo que crea las ramas más homogéneas (O con entropía más cercana a cero), se calcula de la siguiente manera:

i. Se calcula la entropía total.

ii. Se dividen los datos en función de los atributos.
iii. Se calcula la entropía de cada rama y se suma cada una de estas para obtener el total.
iv. Se resta el resultado del cálculo inicial (La entropía total).
v. El resultado es la ganancia de información o decremento de la entropía.
vi. El atributo que tenga mayor ganancia de información es un nodo de decisión del árbol planteado.

La fórmula para la entropía es:

Usar el algoritmo ID3 tiene algunas ventajas: se puede construir un árbol pequeño con reglas comprensibles; es un algoritmo rápido; dado que es recursivo no continúa si encuentra nodos hoja, por lo que se reduce el número de iteraciones o comprobaciones; utiliza todo el conjunto de datos que se pone a servicio.

Algunas de las principales desventajas son: el sobre entrenamiento o sobre clasificación que puede generar el algoritmo, sólo se comprueba o evalúa un atributo en cada paso; clasificar datos continuos es muy costoso y requiere una gran capacidad computacional, pues se deben crear varios árboles para conocer dónde se rompe esa continuidad.

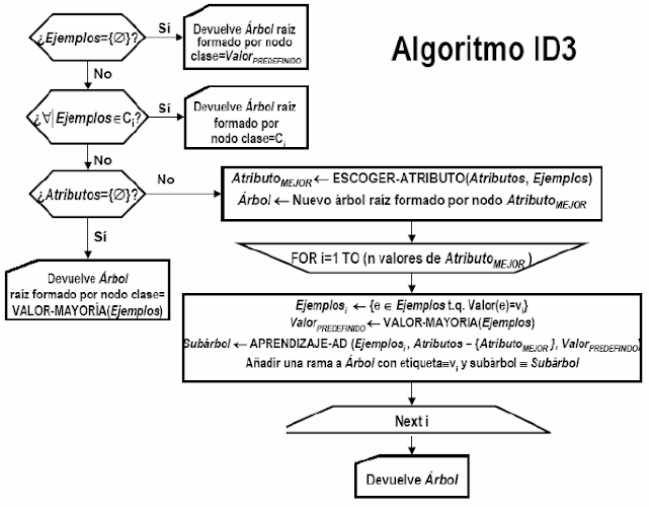

Ilustración 3 Pseudocódigo ID3. Tomado de: López, Bruno. Algoritmo ID3.

EL algoritmo ID3 es frecuentemente utilizado por diferentes empresas destinadas al marketing, ya que con estos pueden tomar decisiones de gran importancia en aspectos de lanzamiento de un nuevo producto al mercado, una problemática muy común es saber si un producto debe ser lanzado a escala nacional o regional y si se debe distribuir a nivel nacional o solo en algunas regiones. Problemáticas como estas son muy comunes y una gran herramienta para su solución son los árboles de decisión, para darle una solución a esta problemática se aconseja realizar el dibujo desarrolla del árbol de decisión como el siguiente:

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

$$E(S) = -P \log_2 P - N \log_2 N$$

Donde P son los ejemplos positivos y N los negativos

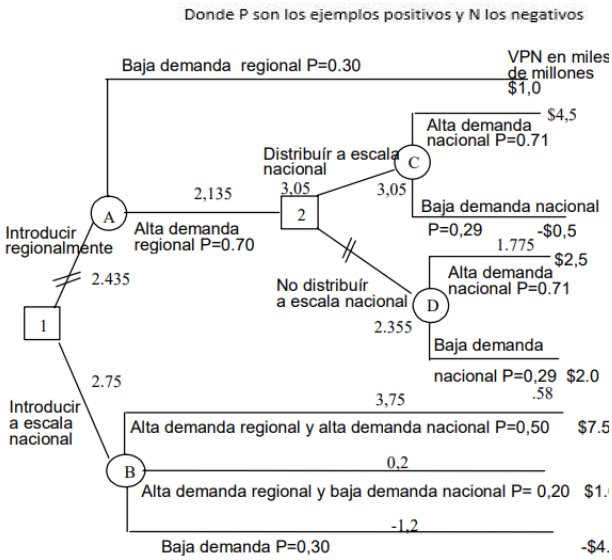

Ilustración 4 Árbol de decisión desarrollado. Tomado de: cashflow88.com/decisiones/riesgo/capitulo8

En el ejemplo se puede observar que la probabilidad por una demanda alta a nivel regional es del 70% Cada combinación puede generar una decisión diferente y así tomar alternativas.

3.2 Algoritmo CART

Esta metodología básicamente construye un árbol máximo que sobreajuste la información suministrada y posteriormente hace el árbol más sencillo, es decir, deja los nodos más importantes para finalmente seleccionar el árbol óptimo.

En primera instancia, se tiene el nodo raíz, por lo que el algoritmo busca partirlo en dos nodos hijos tomando como criterio la variable más adecuada para ello. Para elegir la mejor variable se implementa una medida de 'pureza' en la valoración de los dos nodos hijos. Adicionalmente, se asegura que la pureza de ambos nodos sea la máxima, y para ellos se utiliza regularmente la función Gini. También se puede evaluar la pureza de forma conjunta para todo el árbol.

Desde la raíz del árbol (Nodo inicial) hasta los nodos hojas se deben asignar una clase o una etiqueta a los datos. Esta asignación se realiza mediante una función que tiene en cuenta el número de apariciones de esta, las probabilidades y la pureza de la partición.

El proceso se repite hasta que se llega al tope máximo de niveles del árbol (Si se ha fijado); sólo hay una observación en cada nodo-hoja (por lo que estarían ya clasificados); o todas las observaciones tienen la misma probabilidad asignada en los nodos hoja, debido a esto el criterio de máxima pureza es imposible de determinar.

El árbol complejo debe simplificarse, para esto es usado un método para 'podar' el árbol. El procedimiento debe retirar los nodos que aportan muy poca precisión al modelo; se usa entonces una medida de costo-complejidad y se busca el árbol que obtiene menos valor en el parámetro.

De esta manera, los árboles sencillos pueden ser generados con menor dificultad.

Para seleccionar el árbol óptimo el algoritmo compara los resultados obtenidos con los del registro real (Con los que aprendió) para ver cuál es el que más se ajusta.

3.3 Algoritmo C 4.5

Este algoritmo, como el ID3, genera árboles de decisión con arreglos de datos de entrenamiento mediante el concepto de entropía (De la teoría de información), usándolo como criterio para la división eficaz; esto ya se explicó en el algoritmo ID3.

El algoritmo tiene algunos casos base: todas las muestras pertenecen a una misma clase o tienen una misma etiqueta, por lo que genera un nodo hoja eligiendo esa clase; ninguna de las características genera una ganancia de información, por lo que crea un nodo de decisión con el valor esperado de la clase; instancia de una clase antes no vista es encontrada, creando un nodo con el valor esperado de la clase.

Sin embargo, tiene mejoras sustanciales con respecto al algoritmo ID3:

El algoritmo no sólo maneja atributos discretos sino también continuos, mediante el establecimiento de un umbral y la comparación de valores del atributo con éste para dividir la lista.

Permite marcar valores de atributos con (?) para indicar falta de los mismos y no utilizarlos posteriormente en los cálculos de ganancia y entropía.

Poda los árboles después de su creación, sustituyendo a aquellos que no aportan precisión por nodos hoja.

3.4 Algoritmo CHAID

Dado que las principales diferencias radican en la manera como el algoritmo clasifica o divide los datos, podemos decir que CHAID (Chi-are automatic interaction detector) usa como base la distribución x^2. El índice de probabilidad Chi-cuadrado es un método estadístico que pretende establecer la independencia entre la distribución de los datos observados u obtenidos empíricamente y una distribución teórica. Testea la hipótesis nula de que dos variables son independientes la una de la otra o, en otras palabras, que el comportamiento o valor de una de ellas no induce o condiciona de ninguna manera el comportamiento de la otra. Para el caso particular de árboles de decisión, se postula la hipótesis nula de que la división y la variable de la clase son independientes.

Para terminar el proceso, se usa un umbral para comparar los valores del atributo. Adicionalmente, usa la corrección de Bonferroni para solucionar el problema de comparaciones múltiples. Esta corrección, aplicada a árboles de decisión, lo que hace básicamente es mitigar el sesgo que pueda haber cuando se tienen entradas con muchos valores, es decir, esta corrección o ajuste al número de valores categóricos de la variable de entrada.

Finalmente, este algoritmo, como el C 4.5 (Y el c5, su evolución), también utiliza una metodología para marcar datos o valores faltantes en los atributos y así no incluirlos en los cálculos de división. Puede reconocer valores de clase que sesguen su funcionamiento o precisión marcándolos como faltantes y así seguir trabajando con normalidad.

4. Lista de listas (Matriz)

Para la realización de la estructura de datos se tuvieron en cuenta muchos aspectos, como la eficiencia, velocidad para trabajar con los datos, el lenguaje de programación, entre otras. Debido a estas variables se llegó a la conclusión que la mejor forma de guardar los datos en memoria sería una lista de listas, donde cada la lista principal es un vector unidimensional en el que cada elemento es un estudiante (objeto) con sus atributos y las listas interiores son todos los atributos de los estudiantes.

A continuación, un ejemplo gráfico de la estructura seleccionada.

La lista principal se encuentra compuestas de listas:

[

E1 1	E1 2	E1 3	E1 4	E1 5	E1…n

E2 1	E2 2	E2 3	E2 4	E2 5	E2…n

E3 1	…	…	…	…	E3 n
.	…	…	…	…	…
.					
.					

Em 1	Em 2	Em 3	Em 4	Em 5	Em n

]

Donde:

E: Estudiante
m: filas
n: columnas
m x n = matriz (Arreglo filas, columnas)
Em n: Estudiante número m (Fila), atributo número n del estudiante (columna).

4.1 Operaciones de la estructura de datos

Acceder: la operación o función Acceder, recibe dos argumentos como números: val_estu y val_data, siendo el segundo opcional. Estos números, son las coordenadas de acceso a la matriz general, la función primero evalúa si se le ingresó o no la coordenada de columna (Val_data, que sería el atributo de cada estudiante) y, en caso de que no, retorna al estudiante número val_estu, es decir, un vector fila que contiene todos los atributos de un estudiante en particular (Partiendo de que se conoce su posición en la matriz); si se le ingresa val_data, la función retorna el atributo en la posición val_data del estudiante número val_estu. En resumen, la función retorna un elemento (O vector), partiendo de dos índices, o sea, partiendo de que se conoce su posición.

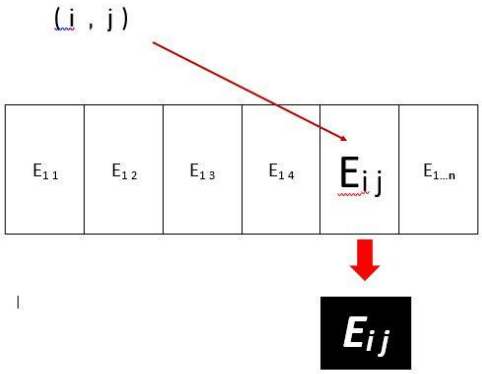

Ilustración 5 Operación acceder

Buscar: este método recibe como argumentos el arreglo de datos o matriz de datos (Donde va a buscar el elemento que se le indique), las cabeceras de esos datos, el código de estudiante y de manera opcional, una cadena de caracteres que simula un atributo Para el caso en el que sólo se le especifica un código de estudiante (Que es lógico buscar estudiantes en una base de datos por su código o documento de identidad), la función retorna un vector fila o, para el caso, un estudiante completo, con todos sus atributos, en formato de lista; para el segundo caso, la función retorna un atributo en específico de este estudiante, que se le indica en los argumentos de la función, resultando de esta manera un elemento del vector estudiante, no toda la lista. ¿Cómo lo hace? Primero, testea que si se recibió el argumento ID_estudiante como una cadena, si es así, empieza a buscar en la columna 0 (Código del estudiante) una cadena coincidente a la especificada, esto mediante un ciclo for; al encontrar esta mediante una comparación lógica, se sobrescribe el argumento ID_Estudiante por la posición que tiene éste en el arreglo de datos; luego, verifica si se le asignó o no el argumento opcional (atributo), para retornar, en caso de que no, el vector estudiante completo; en caso de que el atributo no sea nulo o se le haya declarado a la función por el usuario, ésta retorna un elemento del arreglo basado en las posiciones obtenidas mediante la comparación basada en ciclos for. A diferencia de la búsqueda del ID_Estudiante, para el atributo, se compara con el vector donde se almacenaron las cabeceras, no con el arreglo de datos general.

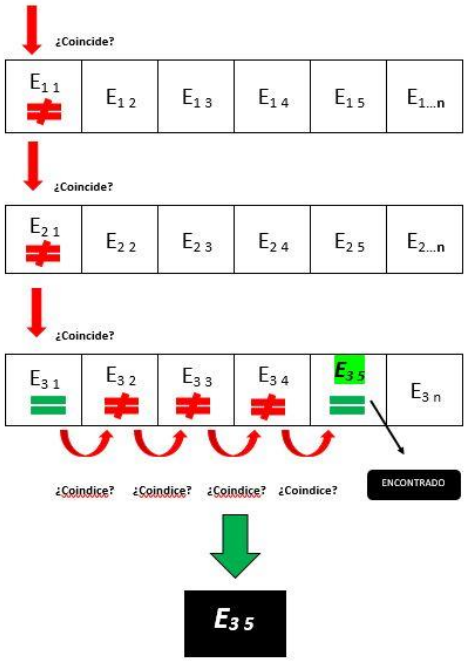

Ilustración 6 Operación buscar

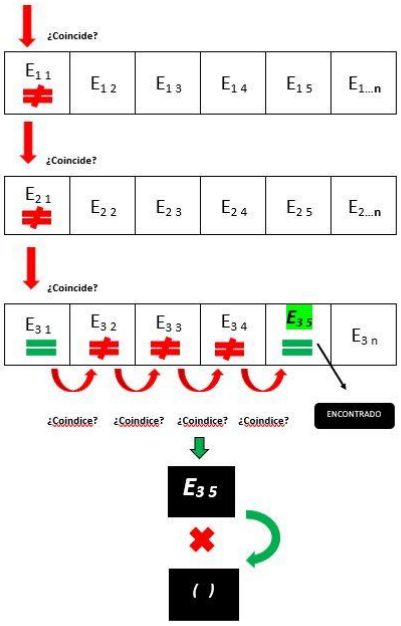

Ilustración 7 Operación borrar

Borrar: este método trabaja igual que el Buscar, es decir, mediante la especificación de argumentos opcionales u obligatorios y el primer testeo de lo que se ingresó (Que sea coherente o no), inicia una búsqueda apoyada en el ciclo for, primero por filas (Con la ID) y luego en columnas (Con el atributo), para encontrar la posición deseada que se desea eliminar. En pocas palabras, el método funciona igual que Buscar, pero tiene una pequeña diferencia: en vez de retornar el elemento, dato, valor o vector que el usuario especifica en los argumentos (Como cadenas), lo elimina, que específicamente se reemplaza por None.

Insertar: la función insertar recibe como argumentos el arreglo principal de datos, el vector de las cabeceras, y opcionales Estudiante (Que es un vector fila, representando a un estudiante con todos sus atributos), ID_Estudiante, Atributo, y Nuevo valor. En primer lugar, verifica que, si no se le ingreso el vector estudiante, se le hayan ingresado los otros parámetros o argumentos, es decir, el ID, el atributo y el nuevo valor. Si no puede operar, imprime mensaje diciendo que no están todos los argumentos necesarios. Si no se le ingresa ningún parámetro además del arreglo y el vector cabeceras, también imprime que no puede hacer la inserción. Seguido, la función testea, si se le ingresó el vector estudiante, mediante el método de listas en python append, se agrega ese vector fila en la última fila del arreglo de datos o matriz general. Si el vector estudiante es vacío, entonces procede a buscar en la matriz general el ID_estudiante ingresado, s decir, el estudiante, y tras haberlo encontrado, se mueve a través de las columnas buscando el atributo ingresado como carácter, que se cuantifica mediante la búsqueda de la posición en el vector cabecera, para así, reemplazar esa coordenada (fila, columna o ID_Estudiante, Atributo) con el nuevo valor ingresado por el usuario. De esta manera, se agrega todo un estudiante como vector fila, o se inserta un atributo de un estudiante en específico (Ya existente), modificando su antiguo valor. Como es de esperarse, es una extensión del método BUSCAR.

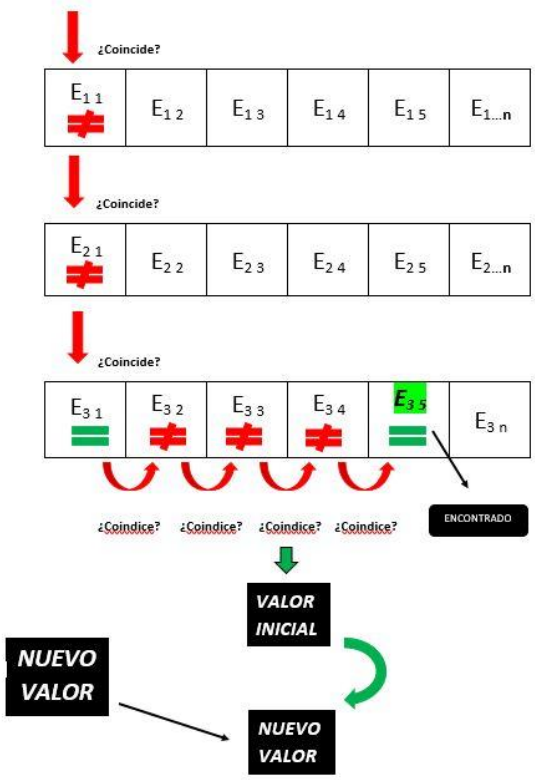

Ilustración 8 Operación insertar

4.2 Criterios de diseño de la estructura de datos

Dado que las matrices, son arreglos de arreglos (Lista de listas en este caso), resultan indicadas a la hora de acceder a datos, de eliminar y de agregar, pues a diferencia de una pila, por ejemplo, para buscar un dato no hay que iterar sobre cada uno de ellos hasta llegar al requerido con la finalidad de conocerlo, eliminarlo o modificarlo.

En una matriz, 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.
Podemos afirmar entonces, que almacenar los datos en una matriz, resulta eficiente no sólo por el número de datos, sino porque se tiene varias columnas o categorías que hay que diferenciar de cada fila o estudiante, haciendo esta estructura más eficiente (Sólo evaluando el caso de una pila, donde hay que agregar un paso adicional, y es el iterar sobre cada uno de los elementos para encontrar el buscado). Esto se puede ver más claramente:

4.3 Análisis de Complejidad

Para calcular la complejidad de las diferentes operaciones se tuvieron en cuenta dos situaciones. Promedio hace referencia cuando se desea realizar la operación sobre la lista principal, es decir, si se desea buscar, borrar, insertar o acceder a una lista estudiante con todos sus atributos.

Por otra parte, el peor de los casos hace referencia a cuando se desea buscar, borrar, insertar o acceder a un atributo en específico de una lista estudiante, este puede ser considerado un plus que tienen las operaciones ya lo esperado es no realizar cambios en los atributos de los estudiantes, por tal motivo si estas añadiduras se eliminan o no se utilizan la complejidad de las operaciones serían las que aparecen en la columna Promedio.

	Complejidad	
Operación	Promedio	Peor de los casos
Buscar	O(n)	O(n*m)
Borrar	O(n)	O(n*m)
Insertar	O(1)	O(n*m)
Acceder	O(1)	O(1)

Ilustración 9 Complejidad de las operaciones

4.4 Tiempos de Ejecución

Para calcular el tiempo de ejecución de que se toma el programa en cargar los diferentes datasets de los estudiantes y las 4 operaciones que se pueden realizar a los mismos: acceder, buscar, insertar y eliminar. Se utilizó la librería time, esta permite saber cuánto tarda desde que comienza un algoritmo hasta que ejecuta le última línea en segundos. Esta metodología se usó tanto para el tiempo de carga como el de las operaciones en el peor de los casos, a continuación, se muestran los resultados obtenidos:

Nota, los valores de tiempo se encuentran multiplicados por 100.

	Conjunto 1		Conjunto 2		Conjunto 3		Conjunto 4		Conjunto 5		Conjunto 6	
	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train
Cantidad de datos	5000	15000	15000	45000	25000	75000	35000	105000	45000	135000	19255	57765
Tamaño en Bytes	21516	62456	62456	180640	100156	325672	142688	463780	180640	587020	79096	257280
Carga de datos (segundos)	6.0838	43.6831	22.1406	81.6844	42.2901	109.0117	54.8534	131.051	78.5918	174.0369	53.2591	70.8108

Operaciones (segundos)												
Buscar	0.0999	0.4992	0.5016	1.3937	0.9008	3.4906	1.2967	3.4907	1.4926	5.0843	0.8975	1.7986
Borrar	0.0997	0.399	0.399	1.3989	0.9007	2.5936	1.1006	3.4944	1.3927	6.5824	0.6981	1.7956
Insertar	0.0994	0.499	0.4984	1.3964	0.7977	2.7923	1.1966	3.2881	1.4926	4.3879	0.5984	2.0911
Acceder	0	0	0	0	0	0	0	0	0	0	0	0

Ilustración 10 Tiempos de ejecución para la carga y las diferentes operaciones

4.5 Memoria

Por otra parte, 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, a continuación, se muestran los resultados obtenidos:

	Conjunto 1		Conjunto 2		Conjunto 3		Conjunto 4		Conjunto 5		Conjunto 6	
	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train
Cantidad de datos	5000	15000	15000	45000	25000	75000	35000	105000	45000	135000	19255	57765
Tamaño en MB	21.516	62.456	62.456	180.64	100.156	325.672	142.688	463.78	180.64	587.02	79.096	257.28

Ilustración 11 Espacio utilizado para cada conjunto de datos

4.6 Análisis de los resultados

Si observamos los tiempos que se obtuvieron en el numeral 4.5 se puede tener una idea de cómo es el comportamiento de las operaciones en tiempo cuando se debe realizar sobre el peor de los casos, debido a esto es un poco complicado poder tener una visión clara de cómo sería el comportamiento de las operaciones si se realizan sobre cada uno de los elementos de la lista, este ejercicio ayuda a comprender de una forma mucho más clara la complejidad de las operaciones, a continuación se muestra el tiempo que tardan las operaciones en ser aplicadas a todos los elementos de la lista.

Tiempo tomado para hacer la operación a cada dato de la lista			
Datos	5000	15000	45000
Buscar	6,003	76,169	695,2
Acceder	0,0029	0,01	0,03
Insertar	0,0059	0,01	0,05
Borrar	4,234	57,986	570,68

Ilustración 12 Espacio utilizado para cada conjunto de datos

Como se evidencia en la imagen anterior, el ejercicio reafirma el tipo de complejidad de las operaciones como Buscar y Borrar O(n) y Acceder e Insertar como O(1). Esto es un resultado favorable ya que los resultados obtenidos son muy similares a los que brindan diferentes herramientas de estructuras de datos como lo son las listas enlazadas.

5. SOLUCIÓN FINAL DISEÑADA

Dada entonces nuestra estructura de datos, se procedió a implementar un árbol de decisión del tipo CART. En términos generales, el algoritmo del CART se inicia con un nodo raíz para el árbol, que recibe todo el set de datos para entrenamiento (Lista de lfilas-listas).Todos los nodos reciben una lista de filas como inputs.

Para el desrrollo del algoritmo se tomó como base un algoritmo propuesto por Google developers, (Gordon, 2017) este algoritmo, desde un pequeño dataset pretende predecir una fruta según sus características, fue necesario realizer un studio profundo del código para poder coomprender su comportamiento y evaluar las diferentes formas de poder adaptarlo y crear las funcionalidades faltantes para el caso deseado.

Cada nodo va a preguntar algo que retorna verdadero o falso sobre alguna de las características o atributos del objeto, en este caso, del estudiante; con base en esta pregunta el algoritmo parte el set de datos en dos, que se convierten en los inputs de los dos nodos hijos que se le agregan al árbol, con el objetivo de obtener atributos más homogéneos o con menos etiquetas. En otras palabras, se pregunta para lograr la mayor pureza posible en la distribución de los atributos en cada nodo (Un tipo de etiqueta en un nodo, no dos tipos o tres, eso es la pureza --> no incertidumbre acerca del tipo de atributo).

Para saber lo que debemos preguntar y cuándo (En qué nodo), se necesita cuantificar cuánto ayuda una pregunta a hacer más homogéneas las etiquetas o, en otras palabras, a reducir la incertidumbre; y cuantificar la cantidad de incertidumbre en un nodo en particular mediante la medida de Pureza de Gini. Se puede cuantificar cuánto reduce una pregunta a dicha incertidumbre mediante el concepto de ganancia de información. Estas dos medidas se van a usar para encontrar la pregunta correcta en un punto del árbol o nodo.

Para saber qué tipo de pregunta se puede hacer, hay que discriminar entre cadenas (Operador ==) o números (Operadores >= o < o <= o >); sabiendo esto, el algoritmo itera sobre las columnas (Atributos) del objeto, o sea del estudiante (fila), basándose en el tipo de datos de dicha columna para hacer la pregunta, determinando con el concepto de ganancia de información cuál de todas las posibles es la más adecuada y permite dividir mejor el set de datos, una 'rama' representada con una lista, que contiene las filas para las cuáles hubo un valor Verdadero retornado con respecto a la pregunta planteada, y otra para las filas que retornaron Falso; así hasta generar un nodo con la máxima pureza posible, una hoja, y poder predecir el écito o no (1 ó 0) de un estudiante con determinadas características.

Se contruye el árbol recursivamente (Haciendo llamados para agregar nodos en ramas izquierdas y derechas) hasta que no se pueda preguntar nada más, es decir, hasta llegar a una hoja del árbol, donde se puede dar la predección.

6. RESULTADOS:

A continuación se muestran los diferentes valores en términos de tiempo de ejecución y exactitud del algoritmo desarrollado.

Ilustración 13 Tiempo de ejecución del algoritmo

En la gráfica es posible notar el crecimiento exponencial que está sufriendo el tiempo, esto está fuertemente ligado al tipo de estructura de datos que se planteó para el desarrollo del mismo, es posible reducer estos tiempos de ejecución si se utlizan estructuras tipo matrices que permitan realizar operaciones en complejidad constante.

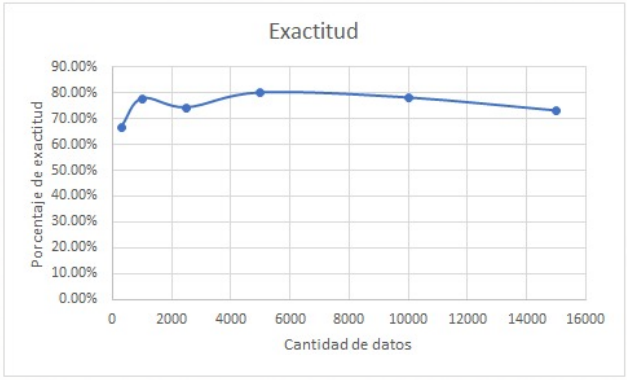

Ilustración 14 exactitud del algoritmo

En este caso es posible observar que el comportamiento de exactitud luego de los 1000 datos se comporta de forma similar, obteniendo valores entre 75% y 82%, estos valores para una predicción representan una baja confiabilidad, es por esto que muchas veces se pretende crear bosques aleatorios que permiten disminuir esta incertidumbre.

Se abordó el problema con una estructura de datos que, a nuestro criterio, era de fácil implementación y acceso en memoria, como lo fue la lista de listas o arreglo de arreglos, cuyos métodos principalmente buscaban agilizar la obtención de información en una base de datos tan grande. Con dicha estructura se buscó implementar un árbol de clasificación y regresión (CART), con el objetivo de predecir el éxito en las pruebas académicas de un estudiante en particular.

7.Trabajos futuros

El objetivo para un futuro cercano es modelar un bosque aleatorio, para que la prediccion tenga mejor ajuste o porcentaje de acierto, pues un solo árbol de decisión no es fiable a la hora de predecir formalmente. Adicional a ello, sería bueno modelar no sólo un CART sino diferentes tipos de metodologías para generar árboles de decisión como el CHAID.

Es importante mejorar u optimizar la estructura de datos, pues es muy lenta y siempre se debe buscar la eficiencia en tiempo y memoria. Quizá implementarlo con un dataframe de numpy (Que es prácticamente una matriz implementada eficientemente), sea una solución viable como acercamiento inicial.

Para implementar un árbol con eficiente, sería pertinente hacerlo con una librería ya creada en el lenguaje de programación de preferencia, pero no quedarse ahí, sino intentar también entender el código, cómo trabaja, cuánto tarda. Establecer relaciones entre lo que se crea y lo que es eficiente resulta sumamente importante a la hora del crecimiento profesional.

AGRADECIMIENTOS

Estamos profundamente agradecidos con la Universidad y a los docentes por su gran labor, por presionarnos y sacar lo mejor de nosotros, pero ofrecer bastante al mismo tiempo. Especialmente, a nuestro profesor y guía principal, así como al curso general de Estructura de datos y algoritmos uno, ya que aportó al entendimiento de los algoritmos en general (Pues busca que haya investigación e implementación), así como su complejidad y sobretodo, su empleabilidad o uso en la vida real.
Como característica adicional, resaltamos la vocación del docente al intentar prepararnos para entrevista en grandes –y sobretodo exigentes – empresas. Dado que no somos del pregrado Ingeniería en Sistemas, consideramos que es de vital importancia un curso como este para carreras afines a las matemáticas o el ámbito cuantitativo, computacional.

A las diferentes fuentes Open Source y especialmente a Google developers con sus cursos gratiutos, los cuales permiten aprender, experimentar y crear los proyectos tantos personales como profesionales

REFERENCIAS

Analicaweb. (julio de 30 de 2015). *La predicción del dato: Redes Neuronales Artificiales.* Recuperado el 7 de febrero de 2020, de https://www.analiticaweb.es/la-prediccion-del-dato-redes-neuronales-artificiales/

Caparrini Sancho, F. (enero de 5 de 2013). *Árboles de desición: Algoritmo ID3.* Recuperado el 7 de

febrero de 2020, de https://es.slideshare.net/FernandoCaparrini/arboles-decision-id3

Casas Mogollón, P. A. (6 de diciembre de 2018). *El problema no es solo plata: 42 % de los universitarios deserta.* Recuperado el 7 de febrero de 2020, de El Espectador: https://www.elespectador.com/noticias/educacion/el-problema-no-es-solo-plata-42-de-los-universitarios-deserta-articulo-827739

Gordon, J. (19 de 07 de 2017). *github.com*. Obtenido de Let's Write a Decision Tree Classifier from Scratch - Machine Learning Recipes #8: https://github.com/random-forests/tutorials/blob/master/decision_tree.ipynb

Helpes. (19 de febrero de 2019). *Algoritmo de C4.5.* Recuperado el 7 de febrero de 2020, de http://www3.helpes.eu/01096402/AlgoritmoDeC45

Numerentur. (29 de marzo de 2019). *Árboles de desición - DT.* Recuperado el 7 de febrero de 2020, de http://numerentur.org/arboles-de-decision/

Scielo. (febrero de 2018). *de decisión (CART). Mortalidad hospitalaria del infarto agudo de miocardio.* Recuperado el 7 de febrero de 2020, de http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0213-91112008000100013