**PROYECTO FINAL DE ESTRUCTURA DE DATOS Y ALGORITMOS**

Con este proyecto basado en la estrategia Machine Learning se busca predecir las probabilidades de llegar al éxito en las pruebas Saber Pro, para esto se tuvieron en cuenta datasets que proporcionan información personal de los estudiantes colombianos que presentaron las pruebas Saber 11, y Saber Pro del 2014 al 2019, dando a conocer los resultados obtenidos.

La realización de este proyecto permitirá a la comunidad colombiana conocer aquellos factores que influyen en el éxito de sus estudiantes en las pruebas Saber Pro. Principalmente, podrá identificar aquellas áreas que más dificultades o que peor desempeño demuestran tener los estudiantes, para así crear estrategias que les permitan mejorar.

**ESTRATEGIA CON LOS DATASETS:**

Consideramos que los datos que nos permitirían evaluar el éxito de los estudiantes en las pruebas saber Pro dependería de los puntajes obtenidos en las pruebas saber 11 y saber Pro, por lo que realizamos un filtro de datos, dejando en uso las columnas que proporcionan los puntajes obtenidos en cada una de las sesiones en las que se divide la prueba, es decir, lenguaje, matemáticas, biología, química, física, ciencias sociales, filosofía, inglés, puntaje profesional y el éxito.

Para realizar la lectura y el filtro de los datos hicimos uso de la librería pandas, numpy, y el método filter, exclusivamente para obtener el filtro de los datos.

**LIBRERÍAS:**

Hicimos uso del algoritmo **CART** para realizar nuestro árbol de decisión, para facilitarnos el proceso usamos la librería Sklearn que es una de las librerías más importantes a la hora de trabajar con modelos Machine Learning.

Claramente la librería Sklearn tiene varias sublibrerias, las utilizadas para este proyecto fueron:

* Metrics
* Ensemble
* Model\_Selection
* Tree

**Metrics:** Permitió realizar la matriz de confusión y calcular la precisión del algoritmo.

**Emsemble:** De acá pudimos importar el bosque aleatorio y su implementación.

**Model\_Selection:** Nos permitió realizar gran parte del proyecto, puesto que con esta entrenamos el modelo.

**Tree:** Facilitó la creación del árbol de decisión y la exportación para su visualización con la librería GraphViz.

También utilizamos otras librerías como por ejemplo Pandas y Numpy para la lectura de los datasets y la manipulación de DataFrames y GraphViz para la renderización de los árboles de decisión generados.

**IMPLEMENTACIÓN DEL CÓDIGO**

Para empezar, decidimos hacer una lectura a los datos usando el comando read\_csv de la librería pandas para guardar los archivos de datos en DataFrames y así trabajar más fácilmente con ellos. Luego hacemos una limpieza de datos borrando algunas columnas que ni nos aportaban ni nos quitaban eficiencia en el modelo, quedándonos un DataFrame con menos columnas y más flexible para operaciones.

Ahora dividimos el DataFrame con los datos, almacenamos en un dataframe llamado X todos los datos que nos servirán para hacer la predicción, es decir, todas las columnas que funcionan como variables predictoras, y en Y guardaremos las variables a predecir, que en este caso sería únicamente la columna de éxito en las pruebas para cada estudiante.

A continuación, usamos el método train\_test\_split para dividir los datos y crear subconjuntos de datos de entrenamiento (train) y de prueba (test) para los dataframe X y Y, quedándonos unas nuevas variables llamadas x\_train, y\_train, x\_test y y\_test.Estas variables serán clave para la implementación del árbol.

Ahora usaremos la librería sklearn para crear un árbol de clasificación, así que creamos el árbol clf y lo ajustamos con los datos de x\_train y del y\_test. Así pues, podemos usar nuestro árbol clf, que ya está entrenado para predecir los resultados del y\_test, los cuales son los resultados propios y reales de éxito en las pruebas de los estudiantes. Las predicciones se guardan en la variable y\_pred.

De esta forma nos quedan dos resultados para cada fila de datos: por un lado, tenemos una columna con los resultados reales de las pruebas, en el otro lado tenemos una columna con los resultados predictos por el árbol. En ambos casos se clasifica con 1 o 0, para indicar respectivamente si el estudiante logró el éxito en la prueba o si en el otro caso no le fue también

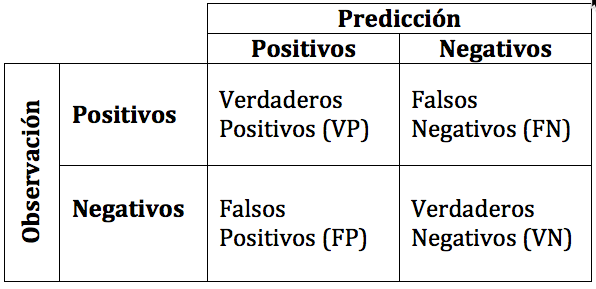
**EFICIENCIA DEL ALGORITMO**

Con estos datos podemos encontrar que tan acertado fue nuestro algoritmo y así saber si tenemos que mejorarlo o si nos da una buena estimación del resultado. Para esto usaremos la sublibrería metrics de sklearn, que nos permite usar el accuracy\_score, que es justamente la medida que estamos buscando, este comando retornará un numero entre 0 y 1 que nos indicará la eficiencia del algoritmo.

Un “accuracy” cercano a uno indica una buena eficiencia en el algoritmo, y un “accuracy” más cercano a 0 significará que el algoritmo no arroja una buena predicción de los datos reales

Después de calcular la eficiencia, podremos conocer también la matriz de confusión. Esta es una matriz 2x2 que nos indica:

* La cantidad de predicciones que arrojaron positivo y acertaron la predicción
* La cantidad de predicciones que arrojaron negativo y acertaron la predicción
* La cantidad de predicciones que arrojaron positivo y equivocaron en la predicción
* La cantidad de predicciones que arrojaron negativo y equivocaron en la predicción



Tomado de: <https://rpubs.com/chzelada/275494>

Vemos en la imagen, que las predicciones acertadas, tanto de positivos como de negativos las encontramos en la diagonal principal de la matriz, mientras que las predicciones fallidas estarán en la diagonal secundaria. Siendo la idea construir un modelo que sea un buen predictor, es de esperar que, si nuestro modelo sale bien, en la diagonal principal estarán casi todos los datos, mientras que en la secundaria debería haber pocos datos. El caso ideal es donde la diagonal secundaria está llena de ceros

Para construir esta matriz la sublibreria metrics nos será muy útil, ya que acá podremos encontrar el comando confusion\_matrix, que nos generará automáticamente la matriz. Lo único que necesitaremos será darle como parámetros a y\_test (los datos reales de éxito) y a y\_pred (los datos de éxito predichos) y este comparará ambas columnas ubicando los casos de éxito y de fracaso en la matriz en su lugar correspondiente

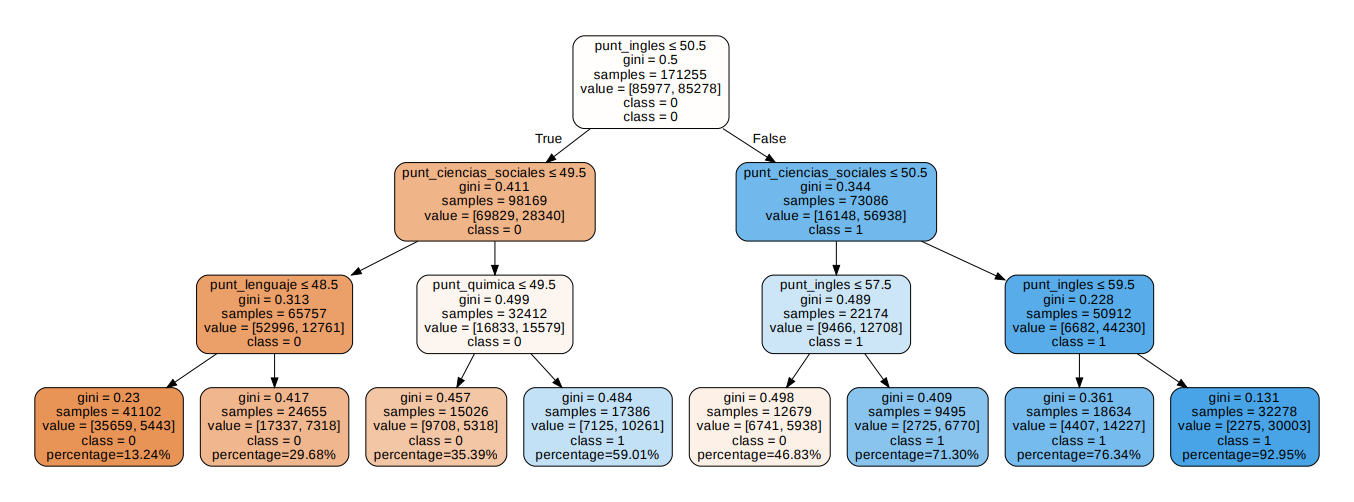
**EXPORTAR Y VISUALIZAR EL ÁRBOL**

También sklearn nos permite visualizar el árbol de forma que nos quede más claro el algoritmo y su implementación sobre el conjunto de datos. Para eso usaremos la sublibreria trees e importaremos el comando export\_graphviz (más tarde usaremos también la librería graphviz).

Si miramos la documentación de este comando, veremos que necesita tener como parámetros: el árbol que vamos a graficar, las variables predictoras en forma de lista, los nombres de las clases (es decir, los posibles resultados que arrojará la variable a predecir, en este caso sería 0 y 1, representando fracaso y éxito en las pruebas) y otros parámetros estéticos sobre la representación gráfica del árbol (como el redondeo en las esquinas de cada nodo, el coloreado, etc.) que ya son de gusto personal

Sin embargo, el comando export\_graphviz no nos mostrará aún el dibujo del árbol, sino que este retorna un archivo DOT con los datos para la representación gráfica, aquí es donde entra la librería graphviz. Esta librería tiene un método llamado Source, el cual nos servirá para renderizar los datos del archivo .DOT que nos arrojó el método anterior, para ya poder verlo ahora si en su forma grafica

También para acabar podemos guardar la gráfica del árbol con el método render, el cual crea un pdf con el dibujo del árbol en la misma dirección donde se encuentra el proyecto.



En la gráfica podemos ver que aparece una serie de variables en cada nodo, a continuación, explicaremos lo que significa cada una:

En primer lugar, encontramos la condición que evalúa el nodo para separar los datos, en el primer caso es punt\_ingles <= 50.5, esta será la prueba por la que pasa el dato del estudiante para luego ser dirigido a la siguiente prueba en el nodo izquierdo o en el derecho. Las hojas no tienen esta condición, pues es el punto final del árbol donde se evalúa el éxito del estudiante en la prueba

Luego encontramos la impureza de gini para cada nodo, que nos indica la probabilidad de que un estudiante sea clasificado incorrectamente, es decir que este pertenezca a la clase 0 y haya sido asignado en la clase 1, o viceversa. que tan impuros (o sea, que tan mezclados) están los datos que están en el nodo. Vemos que, en algunos caminos, a medida que vamos bajando de nivel en el árbol, esta impureza decrece también, es decir, la impureza de gini del nodo hijo es menor que la del nodo padre, haciendo que el modelo sea más eficaz.

Samples, nos da a conocer el total de estudiantes que cumplen una condición establecida en un nodo, en nuestro caso la condición está determinada por el puntaje obtenido en alguna de las áreas.

Value es una lista en la que se divide el total de datos de Samples, esta lista tiene 2 valores que señalan la cantidad de elementos que pertenecen a cada clase. El primer número nos indica cuantos de los elementos totales del nodo pertenecen a la clase 0, es decir “No éxito”, y el segundo elemento de la lista nos indica la cantidad de elementos que pertenecen a la clase 1, que sería “Éxito”

Class es la clase a la que pertenece el nodo, es decir, si el nodo tiende a representar (o representa en sí mismo) datos pertenecientes a las clases “Éxito” o “No éxito”, mediante los valores 1 y 0 respectivamente

**PROBABILIDAD DE ÉXITO**

La probabilidad de que un estudiante tengan éxito según los puntajes obtenidos en las pruebas saber Icfes y las pruebas saber Pro, fue evaluada teniendo en cuenta lo siguiente:

Dada la definición de Samples y Values, realizamos una regla de 3 simple, consideramos que:

100% --------->Samples

X --------->Values[1]

De tal forma que el porcentaje o la probabilidad de éxito de un estudiante estaría dada por X=(100\*Values[1])/Samples.

**CONCLUSIONES**

Aunque para este proyecto no tuvimos en cuenta aspectos como el nivel socioeconómico, el ambiente de estudio, entre otros factores, estos pueden influir en las probabilidades de éxito de los estudiantes en las pruebas saber Pro, quisimos hacer una evaluación en cuanto a esto por medio de gráficos, pero no nos parecía algo muy correcto, ya que había que tener en cuenta la cantidad de datos que había en los datasets con algunas cualidades, por ejemplo, si quisiéramos analizar en que zona los estudiantes son más propensos a llegar al éxito en estas pruebas, llegaríamos a un posible error ya que hay un mayor número de estudiantes pertenecientes a zonas urbanas que rurales, por lo que a pesar de que hayan estudiantes de zonas rurales que llegaron al éxito, esta cantidad podría ser superada por el número de personas de zonas urbanas que llegan al éxito, simplemente porque son una mayor población.

Del grafico del árbol se deduce que los estudiantes que tuvieron un puntaje en inglés y en sociales ambos mayores a 50.5 generalmente tienen buena probabilidad de haber ganado la prueba, pues sería una probabilidad de al menos 0.76, por lo que estos dos factores son claves para evaluar el éxito de los estudiantes, especialmente el puntaje en inglés. Prueba de esto es el estudiante 4289 del archivo TEST 0.csv pues podemos notar como obtuvo casi todas las pruebas altas, pero en la de inglés este obtiene un puntaje de 20.0, lo que le termina llevando a calificarlo como “No éxito” en la prueba

También vemos que si leemos los nodos hojas de izquierda a derecha, notaremos que los indices de gini son bajos por lados de los extremos, pero si vamos a la mitad veremos que estos son considerablemente más altos, lo que significa que en estos puntos el modelo puede no ser muy eficaz. Esto puede deberse a que el modelo que estamos usando de ejemplo tiene muy pocas ramificaciones, es decir, muy pocos criterios para evaluar semejante cantidad de información, lo que hace que termine quedandose corto.

Gracias a este proyecto los estudiantes pueden tomar una estrategia para determinar qué factores podrían llevarlos a tener éxito en las pruebas saber Pro, aunque por supuesto lo más correcto será siempre intentar potenciar las habilidades en cada una de las áreas a evaluar.