Prediction and analysis of state exams via the use of decision trees

Simón Cárdenas Villada Universidad Eafit Colombia scardenasv@eafit.edu.co	Juan Pablo Yepes Universidad Eafit Colombia jpyepesg@eafit.edu.co	Miguel Correa Universidad Eafit Colombia macorream@eafit.edu.co	Mauricio Toro Universidad Eafit Colombia mtorobe@eafit.edu.co

Summary:

The objective of this project is that, through decision trees, a prediction is generated, which, it helps a student (who has previously submitted the tests saber 11o) to know whether in the future will get a higher than average grade in state tests or test saber pro.
This situation has a large number of variables, which can positively or negatively affect the result in the tests, so it is of the utmost importance to study all these variables thoroughly, in order to obtain a reliable prediction. This problem is of paramount importance as this allows us to measure the level of education in a country in one way or another and study how external factors such as sociodemographic factors may or may not affect test results. This project has different related problems, for example, the sociodemographic situation of the student. These will be resolved in the next few verbatim.

Keywords

Decision trees, machine learning, academic success,
prediction of test results

1 Introduction:

Today a good education is sought for all people, however, this is affected by external factors such as corruption, social situation, among others. These factors often affect the student's academic success as they have a great influence on the learning method, and as such the pace of learning that the student can carry.
This problem is evident in large numbers in Latin American or Central American countries as corruption and poverty abound, so students do not have the right and necessary materials for adequate learning, thus creating a direct impact on the results of state tests, both in the tests saber 11th and in the tests saber pro. This project will create a decision tree capable of predicting whether a student will get an above-average score on the test to saber pro, taking into account different factors.

1.1 Problem:

The problem for this project is to predict whether a student will get a higher-than-average grade on the test to saber pro. As explained above, this situation has a big impact on society as it is essential for human beings to have a good education, however, this is affected by many external factors. Creating a solution to this problem would be very useful in society, because, based on the results of the algorithm, the state will be able to keep quality control in middle education easily and more efficiently.

1.2 Solución

In this work, we focused on decision trees because they provide great explainability [5]. We avoid black-box methods such as neural networks, support-vector machines and random forests because they lack explainability. [5]

The solution to implement for this problem is a decision tree algorithm based on the CART model. We chose this path because it is less complicated than other models and the math and algorithms behind it are more simple to understand and implement.

1.3 Structure of the article

In the following, in section 2, we present the work related to the problem. Later, in section 3, we present the datasets and methods used in this research. In section 4, we present the design of the algorithm. Then, in section 5, we present the results. Finally, in section 6, we discussed the results and proposed some future working directions.

2.1 Predictive and incremental validity of self-regulation skills on academic success at the university

"This paper analyzes the predictive and incremental validity of self-regulatory skills on academic success at university through the follow-up of 218 students over four academic courses. " (García & Perez Gonzalez, 2011)
This paper selected 218 students as an experimental publication and took into account variables such as: pre-entry education at the university and sociodemographic variables.
In this work they presented two models, one called RAP-1 and one RAP-3.
It can be said that the accuracy achieved was good as the models showed a result according to the variables they took into account.

2.2 Predicting academic performance: linear regression versus logistic regression

In this report they seek to evaluate the capacity of linear regression and logistic regression in predicting performance and academic success/failure, based on variables, such as attendance and class participation, which is very useful in the project to be developed as they show analysis by logistic regression, giving good accuracy in the results of the report. The project "Predicting Academic Performance: Linear Regression versus Logistic Regression" states that the best predictor of future academic performance is the previous performance "

2.3 PREDICTING STUDENTS' ACADEMIC PERFORMANCE: COMPARING ARTIFICIAL NEURAL NETWORK, DECISION TREE AND LINEAR REGRESSION

Predicting students' academic performance is critical for educational institutions because strategic programs can be planned in improving or maintaining students' performance during their period of studies in the institutions. The result of this study shows that all of the three models produce more than 80% accuracy. It also shows that artificial neural network outperforms the other two models.
206 students' data were obtained for this study

2.4 Student Academic Performance Prediction Model Using Decision Tree and Fuzzy Genetic Algorithm

This work aims to develop students academic performance Prediction model using two selected classification methods; Decision three and fuzzy genetic algorithm.
The algorithm used to implement decision tree is the C4.5 algorithm.

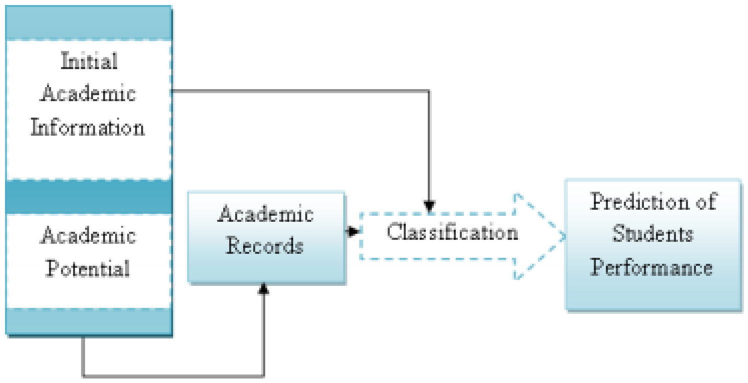

Initial Academic Information
Academic Potential
Academic Records
Classification
Prediction of Students Performance

Fig.1.Proposed Data Mining Model

The results returned by both the decision tree and the fuzzy genetic algorithm were different, but it is not speceficed if a method was more accurate than the other.

3.1 Data collection and processing

We obtained data from *the Colombian Institute for the Promotion of Higher Education* (ICFES), which are available online at ftp.icfes.gov.co. This data includes anonymized results from Saber 11 and Saber Pro. Saber 11 results were obtained from all Colombian high school graduates, from 2008 to 2014, and Saber Pro results from all Colombian undergraduate graduates, from 2012 to 2018. There were 864,000 records for Saber 11 and 430,000 for Saber Pro. Both Saber 11 and Saber Pro included not only the scores but also the socioeconomic data of the students, collected by the ICFES, before the test.

In the next step, both datasets were merged using the unique identifier assigned to each student. Therefore, a new dataset was created that included students who took both standardized tests. The size of this new dataset is 212,010 students. The binary predictor variable was then defined as follows: Is the student's score in The Saber Pro higher than the national average of the period in which he took the test?

It was discovered that the datasets were not balanced. There were 95,741 students above average and 101,332 below average. We perform a subsampling to balance the dataset by a ratio of 50%-50%. After subsampling, the final data set had 191,412 students.

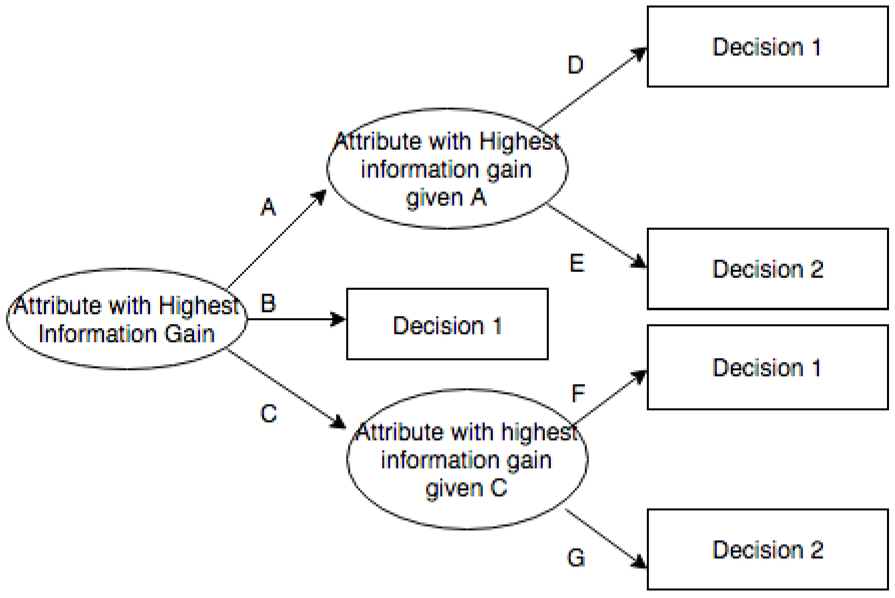
Finally, to analyze the efficiency and learning rates of our implementation, we randomly create subsets of the primary dataset, as shown in Table 1. Each dataset was divided into 70% for training and 30% for validation. Datasets are available in https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets. .

	Data set 1	Data set 2	Data set 3	Data set 4	Data set 5
Training	15,000	45,000	75,000	105,000	135,000
Validation	5,000	15,000	25,000	35,000	45,000

ID3

ID3 is an algorithm invented by Ross Quinlan in 1986. It is used to generate a decision tree from a specified dataset. The algorithm starts with the dataset as the root and with each iteration it cycles trough every unused attribute of the set calculating the information gain of said attribute. After this the attribute with the highest information gain is chosen and the set is split to generate subsets of data until all data has been processed into a node.

Time complexity: $O(h)$, where h is the height of the tree.


Attribute with Highest Information Gain
A
B
C
Attribute with Highest information gain given A
Decision 1
Attribute with highest information gain given C
D
E
F
G
Decision 1
Decision 2
Decision 1
Decision 2

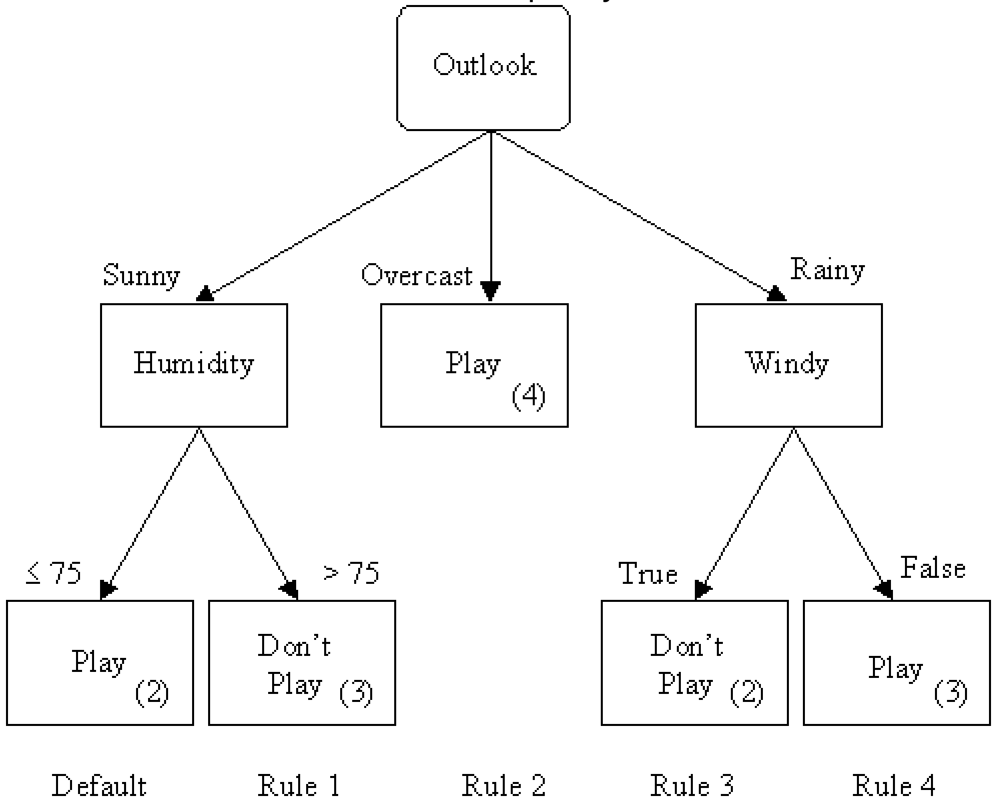
https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/ID3_algorithm_decision_tree.png/220px-ID3_algorithm_decision_tree.png

C4.5

C4.5 is an algorithm invented by Ross Quinlan as an extension to ID3. The main use of C4.5 is to generate decision trees to classify data and it is utilized extensively as a statistical sorting tool. It uses a training dataset in the same way that ID3 does by using information entropy. In each node of the tree the algorithm determines an attribute to divide the set into subsets based on the information gain the attribute has, after this it recursively divides everything into smaller subsets.

C4.5 has some improvements over ID3, such as being able to manage continuous and discrete attributes and allowing to set attribute values as missing so the algorithm ignores this in the entropy and gain calculations.

Time complexity: *O(|E|log|V|)*



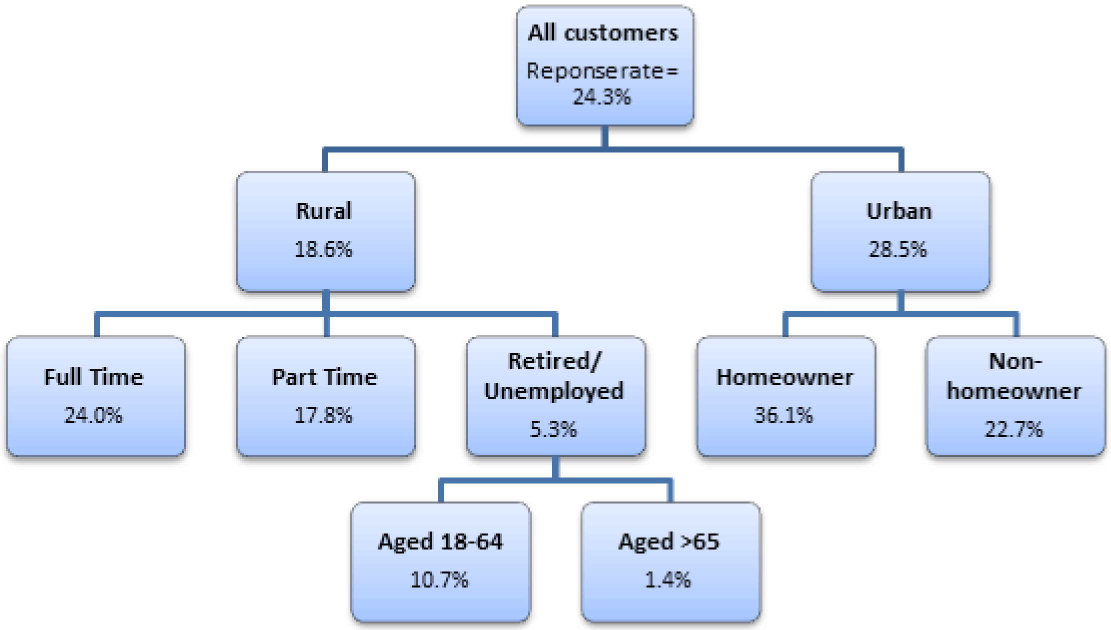
http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/golftree.gif

CHAID

The CHAID algorithm was first introduced by Kass in 1980 and is based primarily on chi-squared statistics, this means that the test yields a value between 0 and 1 indicating the difference between two classes.

The algorithm by default Uses Bonferroni adjustment control and determine the size of the final tree. Its main differentiating factor is that it uses multiway splits at each node, therefore, it can produce multiple branches with only a single parent node. Depending on the statistical significance of each category CHAID will determine an alpha-to-split and an alpha-to-merge to process all categories. The algorithm will continue iteratively until it finds that there are no more splits possible given the alpha values.

Time complexity: O(mn log n)



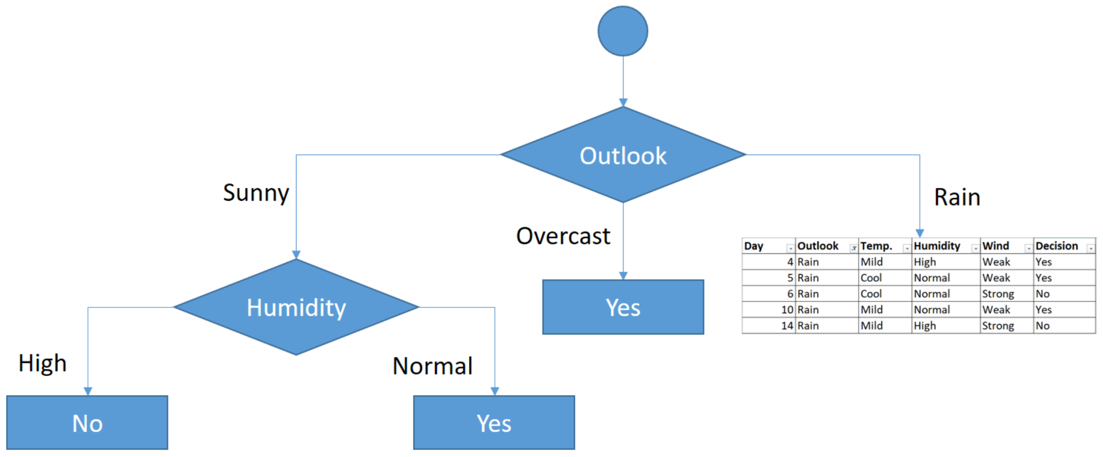
https://select-statistics.co.uk/wp-content/uploads/2015/11/chaid-tree-diagram.png

CART

CART was introduced by Breiman in 1984. This algorithm produces only binary splits unlike CHAID and its based on the principles of purity and balance, this means that a CART decision tree should have equal population on each side of the root. Forcing this balance can lead to some sacrifices in the purity of the nodes and can also increase demand in computational resources but the trees are taller and less wide than the ones made with CHAID.

The recursive splitting procedure of the algorithm stops when it detects no further gain can be made, or some pre-defined stopping rules are met.

Time complexity: O(v·nlogn)



Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

https://i0.wp.com/sefiks.com/wp-content/uploads/2018/08/cart-step-4.png?ssl=1

4. ALGORITHM DESIGN AND IMPLEMENTATION

In what follows, we explain the data structure and the algorithms used in this work. The implementation of the data structure and algorithm is available at Github[1].

4.1 Data Structure

The data estructure that will be used for this project is CART decision tree, with the objective of predicting the results of a given group of students based on several conditions.

The tree splits the information based on the answer given and the purity of the nodes that the split produces. This is why a binary tree is an excellent estructure to determine decisions based on previous information.

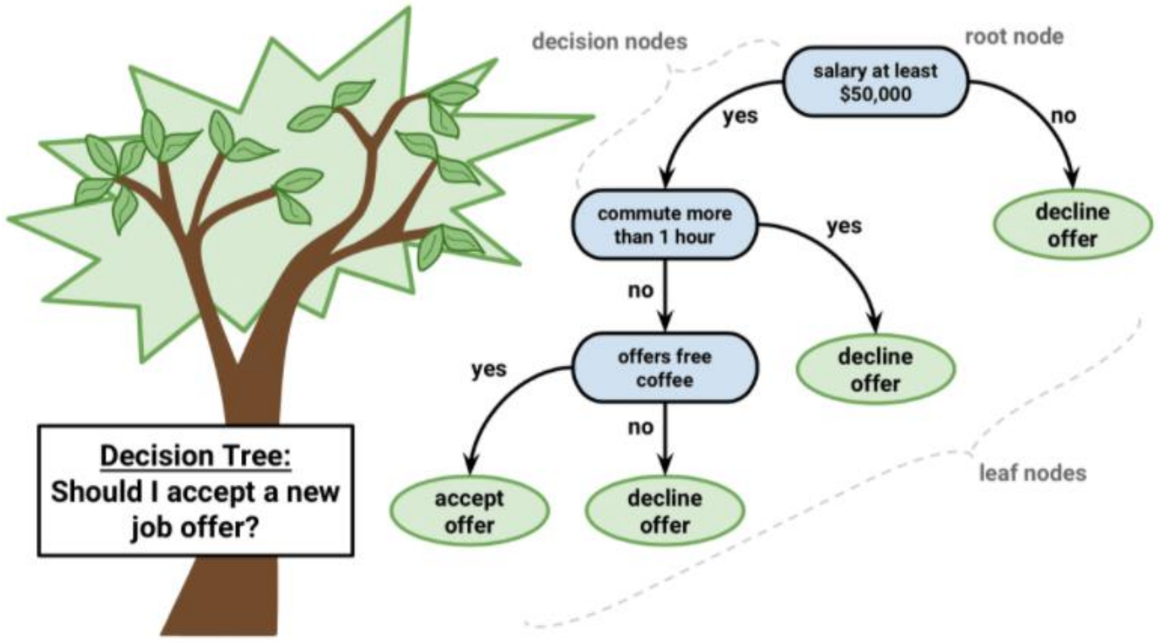


Figure 1: A binary decision tree to predict whether you should take a job or not, taking into account time, salary and some amenities such as coffee.

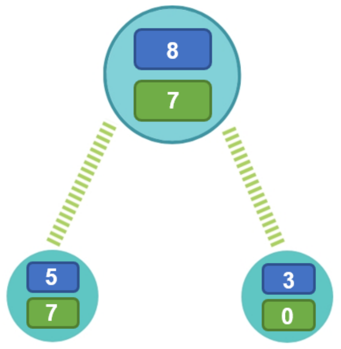
4.2 Algorithms

One of the main algorithms used is based on Gini's impurity, a statistical calculation used to determine that grade of impurity in a given node. This measure is very useful to choose a node that maintains the quality of the information, therefore improving the results.

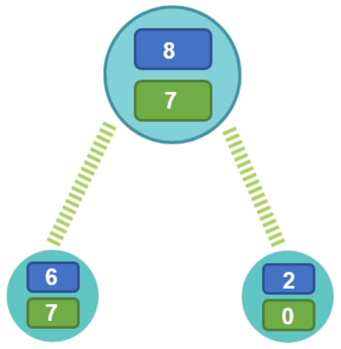
To calculate the best condition to split the information, we use the Gini impurity of the left and right node to calculate the weighted impurity. This returns a value between 0 and 0.5 inclusive, and with this information the tree can choose the value closest to 0 and make the correct split.

4.2.1

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



This node split is based on the condition “fami_nivelsisben == Nivel 2.” For this case, left Gini impurity is 0.474, right Gini impurity is 0.0, and weighted Gini impurity is 0.385.
Elements in blue boxes were marked as having success “0”, and elements in green boxes with success “1”.



This node split is based on the condition “fami_estratovivienda.1 == Estrato 1”
For this case, left Gini impurity is 0.90, right Gini impurity is 0, and weighted Gini impurity is 0.429.
Elements in blue boxes were marked as having success “0”, and elements in green boxes with success “1”.

The node above determines the condition which is split in two, a node on the left which determines the condition cases that have a success of 0 and a node on the right which determines the condition cases that succeed. This splitting occurs recursively for each side of the tree, meaning that on each node a function to determine the best categories on which to split the dataset is called (determined by Gini’s impurity), until the lowest possible impurity is reached, meaning that the tree has successfully created the tags that are most useful to predict the “success” tag of each student.

4.2.2 Testing algorithm

The model is tested by a function that uses a node or root that contains the tree previously mentioned and an array containing the information of the student that will be tested. This method then proceeds to enter said information into the root node and at the end it will return 0 or 1 depending on what the algorithm thinks will the success of the student.

The following step will be to create a two dimensional array that contains a testing dataset with a larger number of students and then submit each one to the prediction algorithm, storing both the prediction result and the actual success status. This will be important later, because when the loop finishes, meaning that each student in the testing dataset has been tested, we have to compare the results. By doing this we can quantify the rate of effectiveness of the prediction in terms of the percentage of predictions that it got right.

4.3 Complexity analysis of the algorithms

Algorithm	**Time Complexity**
Train the decision tree	$O(m*n*2^m)$
Test the decision tree	$O(m*n*\log n)$

Table 1: To train the decision tree we need to call a recursive method for the left and right side, so that would be the complexity of 2^m, but given the it has to check every column to see what is the best value to split the 2-D array, we have to multiply it by the number of columns and rows of the matrix that represents the dataset.

To test the decision tree there is a sorting step that takes time O(n log n) because we are using merge sort. Since we're doing that for each of the O(m) tags, the runtime ends up working out to O(m*n*log n) total work done per node.

Algorithm	**Memory Complexity**
Train the decision tree	O(m*n)
Test the decision tree	O(1)

Table 2: Time Complexity of the training and testing algorithms. Given that the information of the dataset is represented as a two dimensional array of strings, the complexity is dictated by the number of rows times the number of columns of the matrix. For testing is just accesing the array and adding the results, so it takes constant time.

4.4 Design criteria of the algorithm

The algorithm has a lot of parts and methods that help it in the task of creating the tree, we choose a binary tree to help reduce the memory consumed by the recursion, since having a lot of branches or nodes would have increased considerably the complexity of the functions. We also chose to store the possible tags of each column in a Hash table (HashSet in Java) because its fast access and insertion complexity, paired with the fact that in its implementation it ignores duplicate elements, will help to make operations smoother.

Another important decision was to use a sorting algorithm in the matrix that contains the students to organize the data before calling any other functions. We choose merge sort because it is very effective and its time complexity is not that high compared to other sorting algorithms like bubble sort.

An important part of our design criteria is that we decided to focus only on the results of each module in the ICFES exam. We believe this to be the most objective route because in this way only the academic performance of the student is taken into consideration. In addition to this, by comparing only numerical data we can simplify the different functions because its easier to deal with numbers than with String values.

The construction of the tree and some of the classes that are used to build the nodes and the different labels that separate the columns were heavily based on the video guide posted by Google Developers on YouTube, because we believe this to be one the best representations of a classifying tree.

5. RESULTS

5.1 Model evaluation

In this section, we present some metrics to evaluate the model. Accuracy is the ratio of number of correct predictions to the total number of input samples. Precision. is the ratio of successful students identified correctly by the model to successful students identified by the model. Finally, Recall is the ratio of successful students identified correctly by the model to successful students in the dataset.

5.1.1 Evaluation on training datasets

In what follows, we present the evaluation metrics for the training datasets in Table 3.

	Dataset 1 ***N=800***	***Dataset 2*** ***N=5000***
Accuracy	75%	75%
Precision	50.1%	50.1%
Recall	50.2%	48.2%

Table 3. Model evaluation on the training datasets.

5.1.2 Evaluation on test datasets

	Dataset 1 ***N=800***	***Dataset 2*** ***N=5000***	***Dataset 3*** ***N=135000***
Accuracy	69%	69%	70%
Precision	83.55%	83.55%	84%
Recall	52.5%	52.5%	52.5%

5.2 Execution times

	Dataset 1 ***N=800***	***Dataset 2*** ***N=5000***	***Dataset 3*** ***N=135000***
Training time	7.8 s	6.022 min	Estimado: 90mins
Testing time	250ms	421ms	Estimado: 10s

5.3 Memory consumption

We present memory consumption of the binary decision tree, for different datasets, in Table 6.

	Dataset 1 ***N=800***	***Dataset 2*** ***N=5000***	***Dataset 3*** ***N=135000***
Memory consumption	240 MB	380 MB	2.1GB

Table 6: Memory consumption of the binary decision tree for different datasets.

6. DISCUSSION OF THE RESULTS

The results obtained are congruent with the level of design the algorithm has. Given that it's only a single tree the accuracy and other metrics of measurement are in the same range as similar solutions. The high memory consumption was expected given the heavy reliance on recursion to build the tree, but the building of the tree when using large datasets does take a considerable amount of time so that could be a possible improvement. We believe that it would need more refinement in order to be used to give scholarships because the precision could be better.

6.1 Future work

In the future researching about ways to build the tree that don't rely on recursion could be interesting, because maybe it would help the memory consumption when working with large datasets. Also simplifying the code for understandability and easier troubleshooting would be a great improvement. The idea of bluing a random forest could help with the accuracy of the predictions, but it could mean that some of the functions and methods would need an overhaul to work with more than one tree.

ACKNOWLEDGEMENTS

We want to extend our gratitude to the monitoring team and all of the students that helped solve or gave us feedback when designing, implementing, and troubleshooting this algorithm. Also to our parents, family and friends that in one way or another provided us with help and care while we spent stressful nights thinking about the challenging but interesting task this posed.

References

1. García Jiménez, María. And Alvarado Izquierdo, Jesús. And Jiménez Blanco, Amelia. La predicción del rendimiento académico: regresión lineal versus regresión logística. Psicothema, Oviedo, 2000
2. García, Rafael. And Pérez González, Francisco. Validez predictiva e incremental de las habilidades de autorregulación sobre el éxito académico en la universidad. Revista de psicodidactica, Vitoria-Gasteiz,2011
3. Ibrahim, Zaidah. And Rusli, Daliela. Predicting Students' Academic Performance: Comparing Artificial Neural Network, Decision Tree and Linear Regression. Research gate, Kuala Lumpur, 2007
4. Hamsa,Hashmia. And Indiradevi, Simi. Student academic performance Prediction model using decision tree and fuzzy genetic algorithm. Elsevier ltd, Pathanamthitta, 2016
5. Johanna Orellana Alvear. Decision Tree and Random forest. Bookdown, 2018

6. Gordon, Josh. Google Developers. 2017,09/13. “Let’s Write a Decision Tree Classifier from Scratch - Machine Learning Recipes #8”. Link: https://www.youtube.com/watch?v=LDRbO9a6XPU