# **ALGORITHM TO PREDICT ACADEMIC SUCCESS IN HIGHER EDUCATION**

| Manuela Moreno Cordoba  Universidad Eafit  Colombia  [mmorenoc2@eafit.edu.co](mailto:ssaldarris@eafit.edu.co) | Yhilmar Andres Chaverra Castaño  Universidad Eafit  Colombia  yachaverrc@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |
| --- | --- | --- |

# **ABSTRACT**

## The purpose of this project is to analyze and solve the problem of predicting the success of a higher education student in Colombia. One of the key factors to execute this project is to be able to find and use an algorithm based on decision trees, which allows me to collect data and that in turn provides the tendency that a student can have for success through a certain amount of data.

## This problem is of the utmost importance since the solution allows us to know which Colombian population could obtain success in higher education, making a better analysis of the country's growth and progress knowing this data. There are many cases like this one, which will be explained as this document is read

## **Keywords**

|  |
| --- |
| Prediction; Academic success; Data structures; Algorithms;  Decision trees |

## **ACM CLASSIFICATION Keywords**

* Information systems > Storage and retrieval of information> Search and retrieval of information > Grouping
* Consultation Formulation > Recovery models > Search process > Selection process

# **1. INTRODUCTION**

Education is one of the factors that most influences the progress of people and societies around the world, the quality of education can determine important areas in a society such as economic growth, access to better employment opportunities, improve the condition of life, the advancement of science, technology, and innovation.

Education can influence the success of a person in an influential way and Colombia, the digital era will revolutionize the area of ​​education, so it is important to consider the Colombian population that may or may not succeed in education higher. However, there are very few tests and solutions on how to predict the success of a student in the academic field, and since success can be defined in multiple ways, we will define academic success as a student's tendency to obtain a Total score, above the average of your cohort, in the Saber Protests. The Saber Protests are the standardized tests that the Colombian government performs at the end of a university degree.

# **2. PROBLEM**

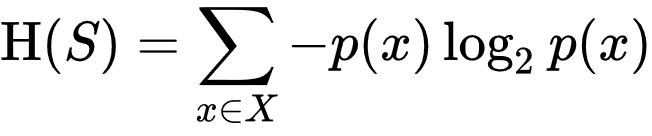
## The problem is that we must design an algorithm, based on decision trees and ICFES data, to predict whether a student will have a total score, on the Saber Protests, above average or not. The variables required for the algorithm must also be considered since it must have the flexibility to interpret data even if it does not respond to all the necessary data.

## **3. RELATED WORK**

## **3.1 ID3:**

In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. It builds a decision tree for the given data in a top-down fashion, starting from a set of objects and a specification of properties Resources and Information. each node of the tree, one property is tested based on maximizing information gain and minimizing entropy, and the results are used to split the object set. This process is recursively done until the set-in a given sub-tree is homogeneous (i.e. it contains objects belonging to the same category). The ID3 algorithm uses a greedy search. It selects a test using the information gain criterion, and then never explores the possibility of alternate choices You should mention the first algorithmic problem and its solution.

In ID3, entropy is calculated for each remaining attribute. The attribute with the smallest entropy is used to split the set S on this iteration.



ID3 does not guarantee an optimal solution. It can converge upon local optima. It uses a greedy strategy by selecting the locally best attribute to split the dataset on each iteration. The algorithm's optimality can be improved by using backtracking during the search for the optimal decision tree at the cost of possibly taking longer.

## **3.2 C4.5**

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the partitioned sub lists.

C4.5 made a few improvements to ID3. Some of these are:

* Handling both continuous and discrete attributes - In order to handle continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it.[[5]](https://en.wikipedia.org/wiki/C4.5_algorithm#cite_note-5)
* Handling training data with missing attribute values - C4.5 allows attribute values to be marked as ? for missing. Missing attribute values are simply not used in gain and entropy calculations.
* Handling attributes with differing costs.
* Pruning trees after creation - C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes.

## **3.3 C5:**

You should mention the third algorithmic problem and its solution.

## This node uses the C5.0 algorithm to build either a decision tree or a rule set. A C5.0 model works by splitting the sample based on the field that provides the maximum information gain. Each subsample defined by the first split is then split again, usually based on a different field, and the process repeats until the subsamples cannot be split any further. Finally, the lowest-level splits are reexamined, and those that do not contribute significantly to the value of the model are removed or pruned.

## C5.0 can produce two kinds of models. A decision tree is a straightforward description of the splits found by the algorithm. Each terminal (or "leaf") node describes a subset of the training data, and each case in the training data belongs to exactly one terminal node in the tree. In other words, exactly one prediction is possible for any data record presented to a decision tree.

## **3.4 CART:**

## The CART algorithm is structured as a sequence of questions, the answers to which determine the next question if any should be. The result of these questions is a tree-like structure where the ends are terminal nodes at which point there are no more questions. A simple example of a decision tree is as follows

## The main elements of CART (and any decision tree algorithm) are:

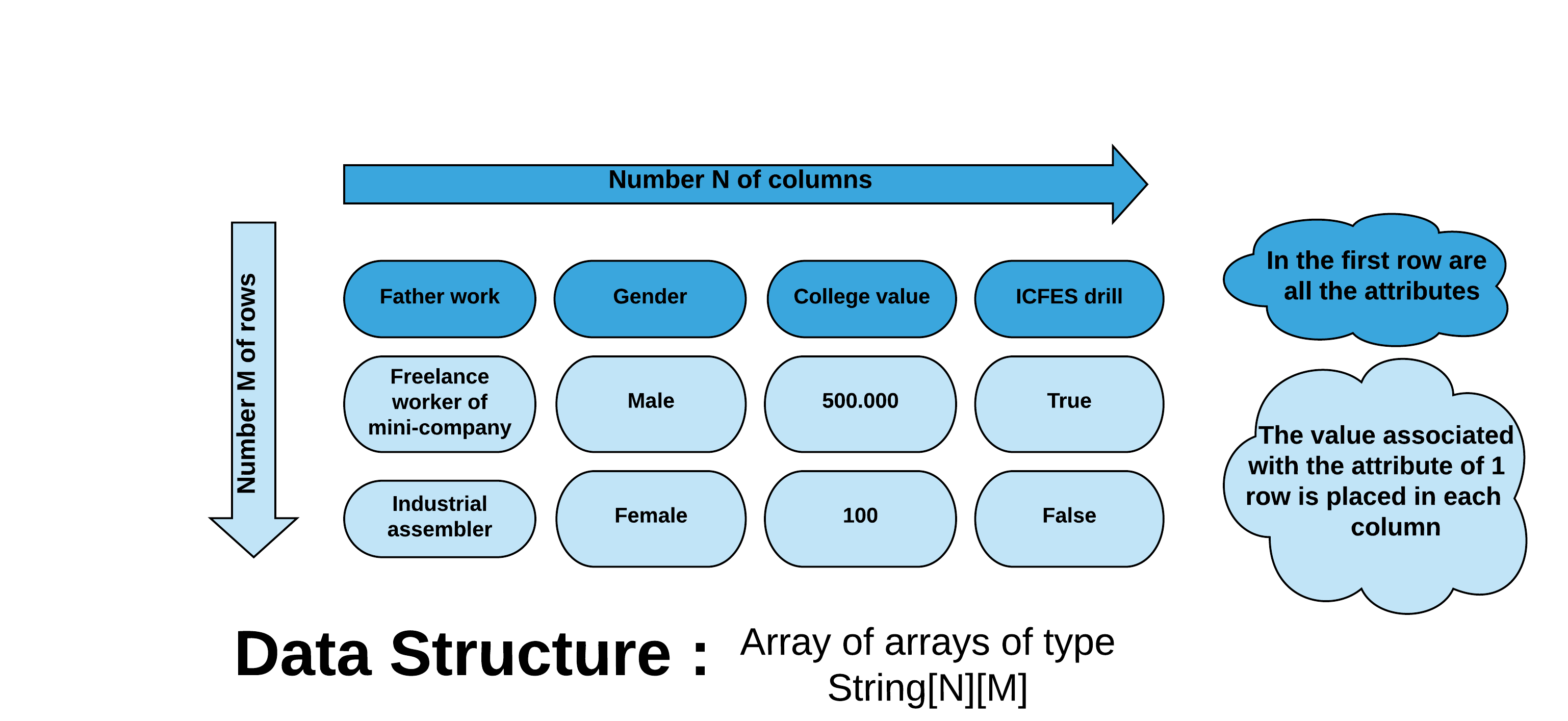
Rules for splitting data at a node based on the value of one variable.

Stopping rules for deciding when a branch is terminal and can be split no more; and

Finally, a prediction for the target variable in each terminal node.

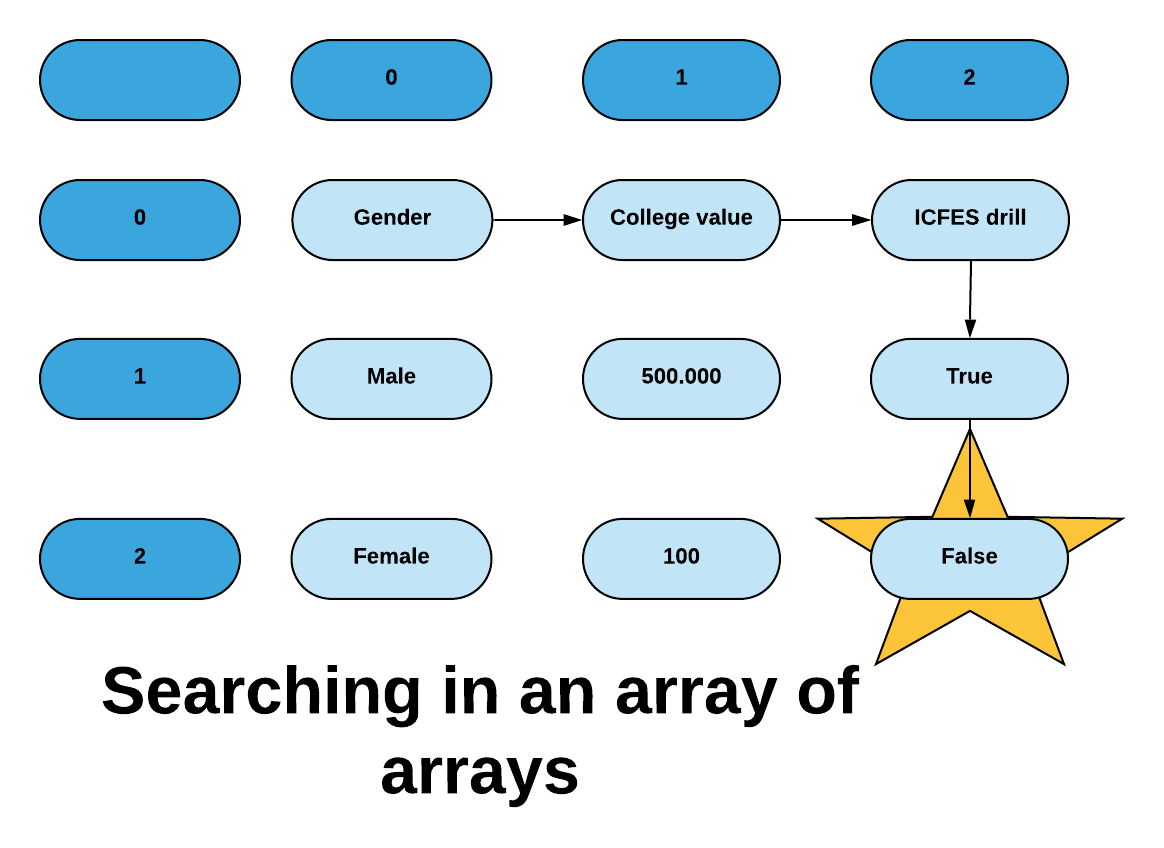
## **4. Array of arrays:**

## The main data structure used to store the data is an array of floating number arrays, where the rows are the students and the columns are the scores obtained in each of the Icfes tests.

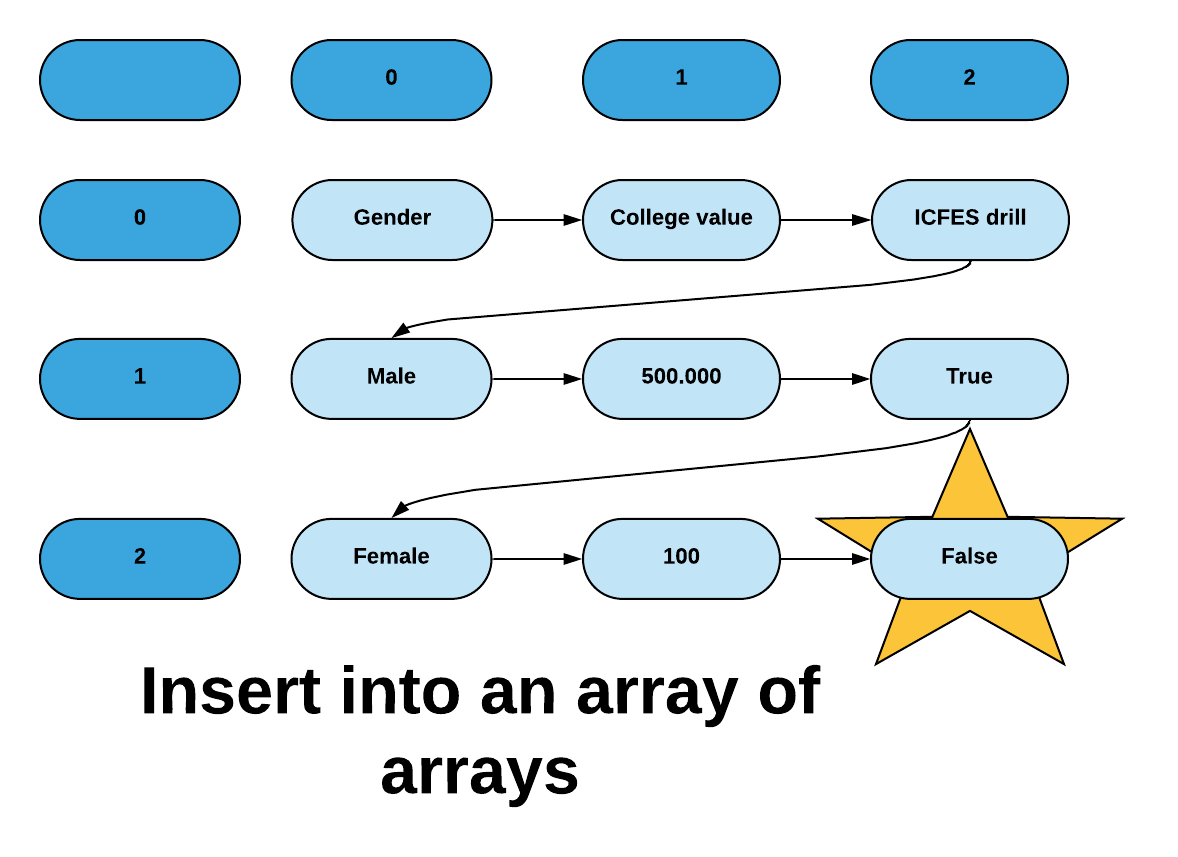


**Figure 1:** Array of arrays of type String with a number N of columns and number M of rows. In the first row are all the attributes and the value associated with the attribute of 1 row is placed in each column.

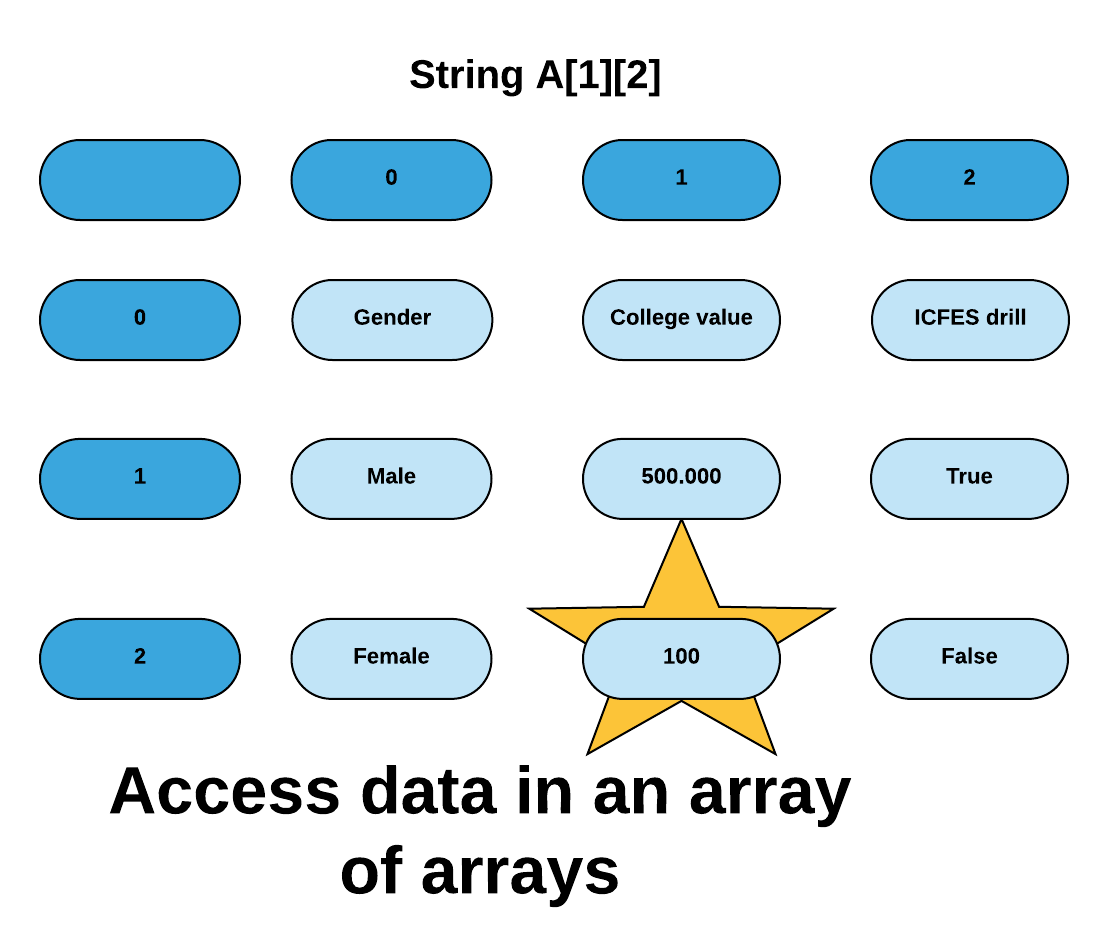
## **4.** **4.1 Operations of an ArrayList of ArrayList of Strings**



**Figure 2:** Search for data in an array of arrays



**Figure 3:** Insert data into an array of arrays



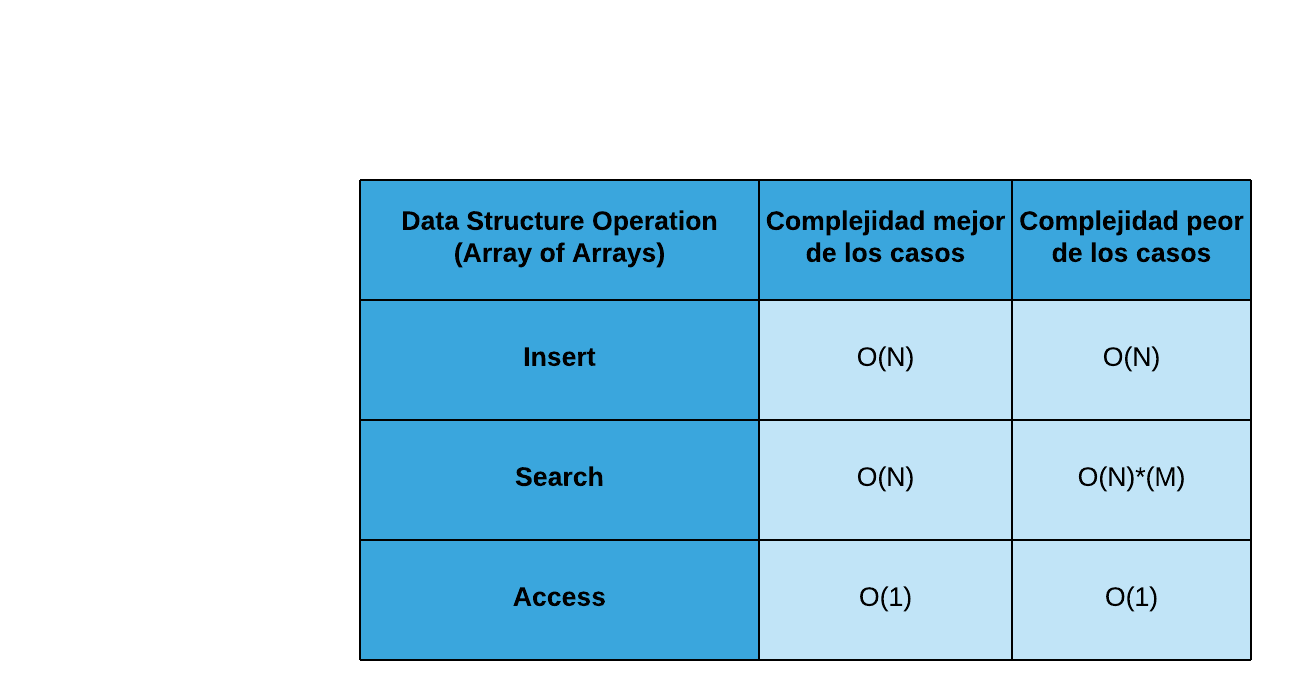
**Figure 4:** Access data in an array of arrays

**4.2 Design criteria of the data structure**

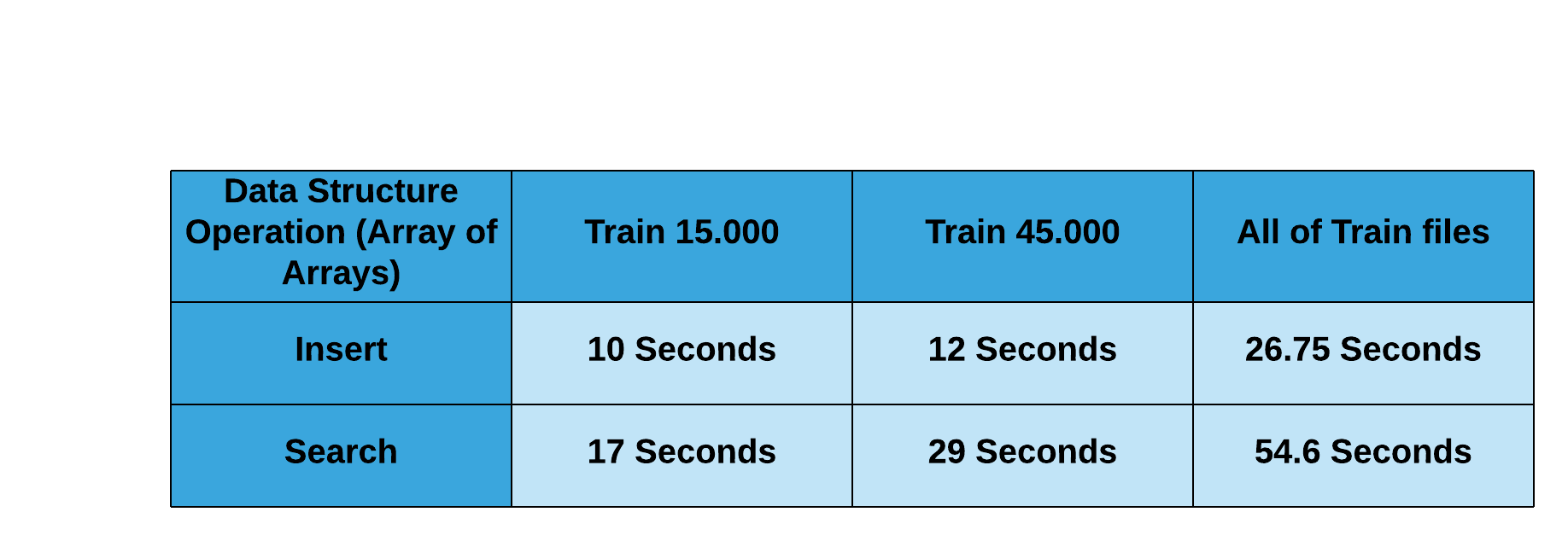
When choosing a structure to save the data, it was chosen to use an Array of Arrays of Float type, where the rows are the students and the columns are the scores that the students will obtain in each subject of the ICFES tests. In the files that are read, each student has 78 attributes, of which we consider that the scores of the subjects are the most important data. When we specify this arrangement with type Float, it can be seen that the arrangement can only have data of floating types , causing many data that is not necessary to be deleted instantly, such as the city or the gender of the school in which the student studied, therefore it is necessary to specify which are the columns that we are going to read. This structure was also chosen because knowing the number of students in a file (Number of rows), the matrix can be built much faster, making the code much more efficient.

When choosing this structure, we also consider the complexity of access. When starting to create the binary tree, comparisons are made between the data all the time, this means that the algorithm has to be constantly accessing the data in the array, and since the access complexity is o (1), this makes the decision tree much faster to complete.

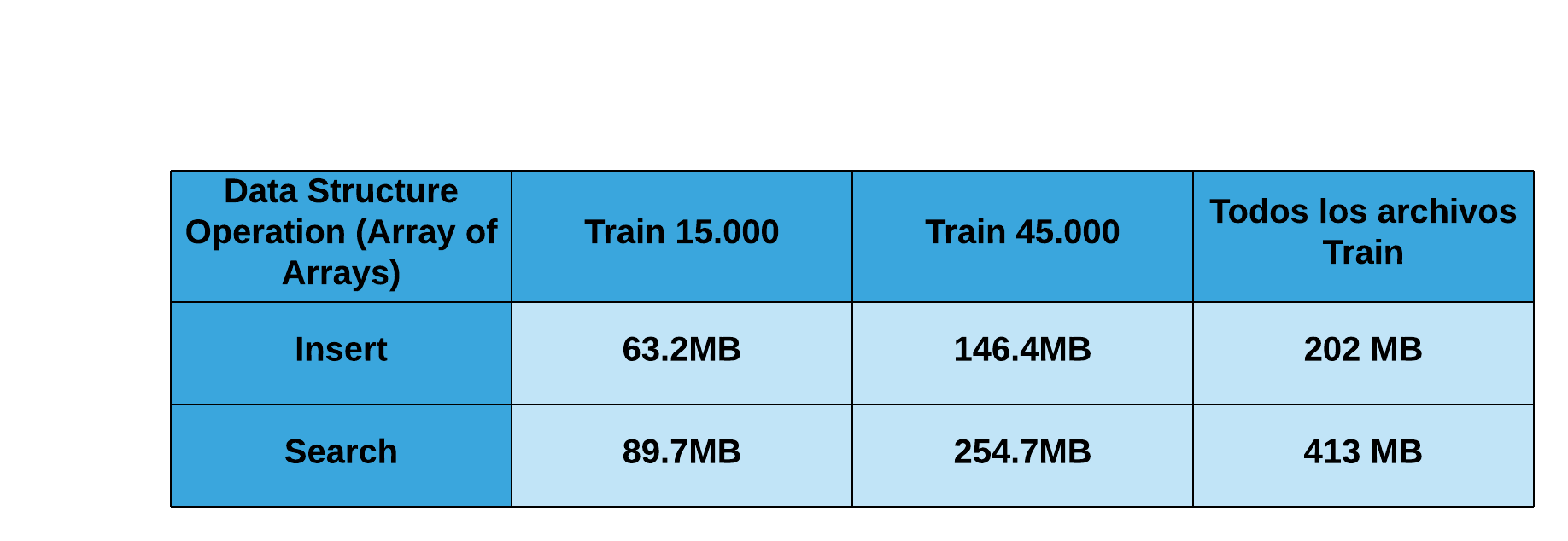
**4.3 Complexity analysis**



**Table 1:** Table to report complexity analysis

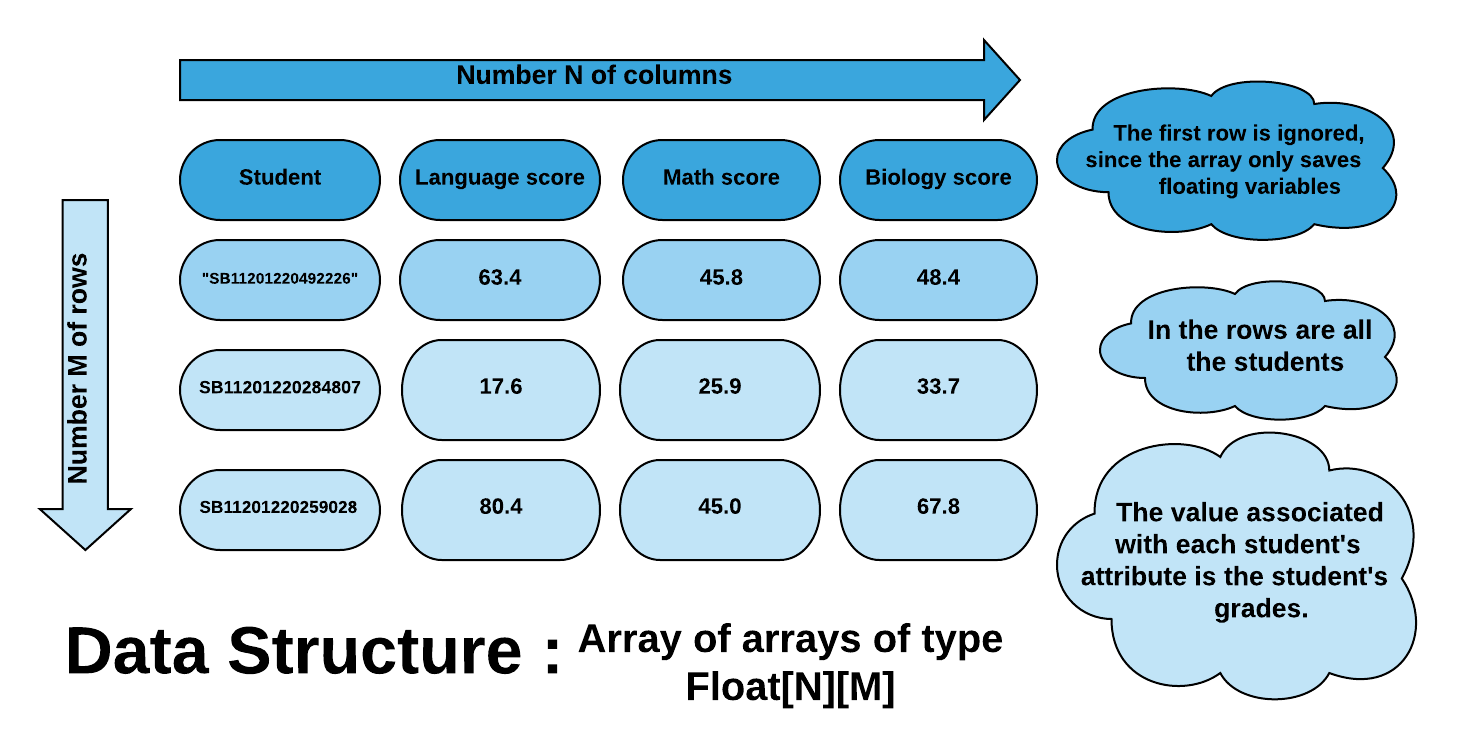
 **Table 2:** Execution time of the operations of the data structure for each data set.

## **4.5 Memory used**



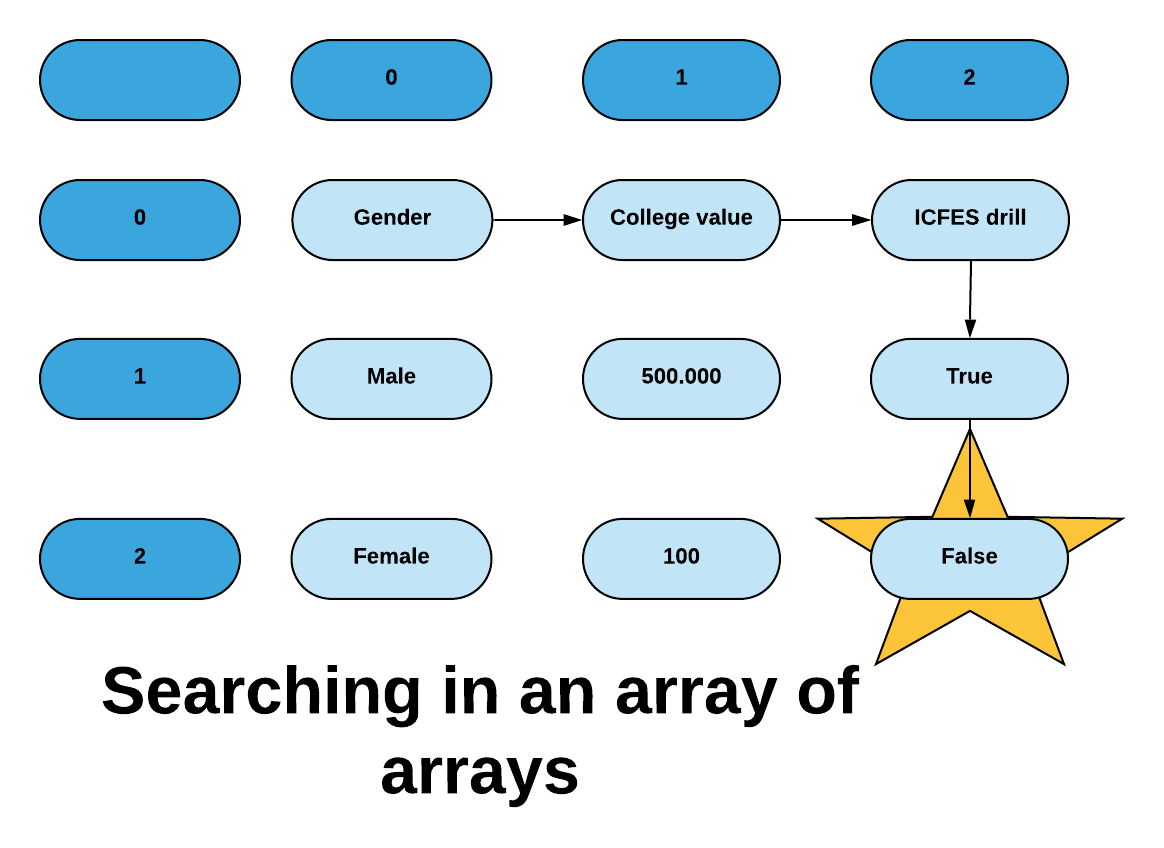
**5. ARRAYS OF ARRAYS OF FLOATS**

## The main data structure used to store the data is an array of floating number arrays, where the rows are the students and the columns are the scores obtained in each of the Icfes tests.

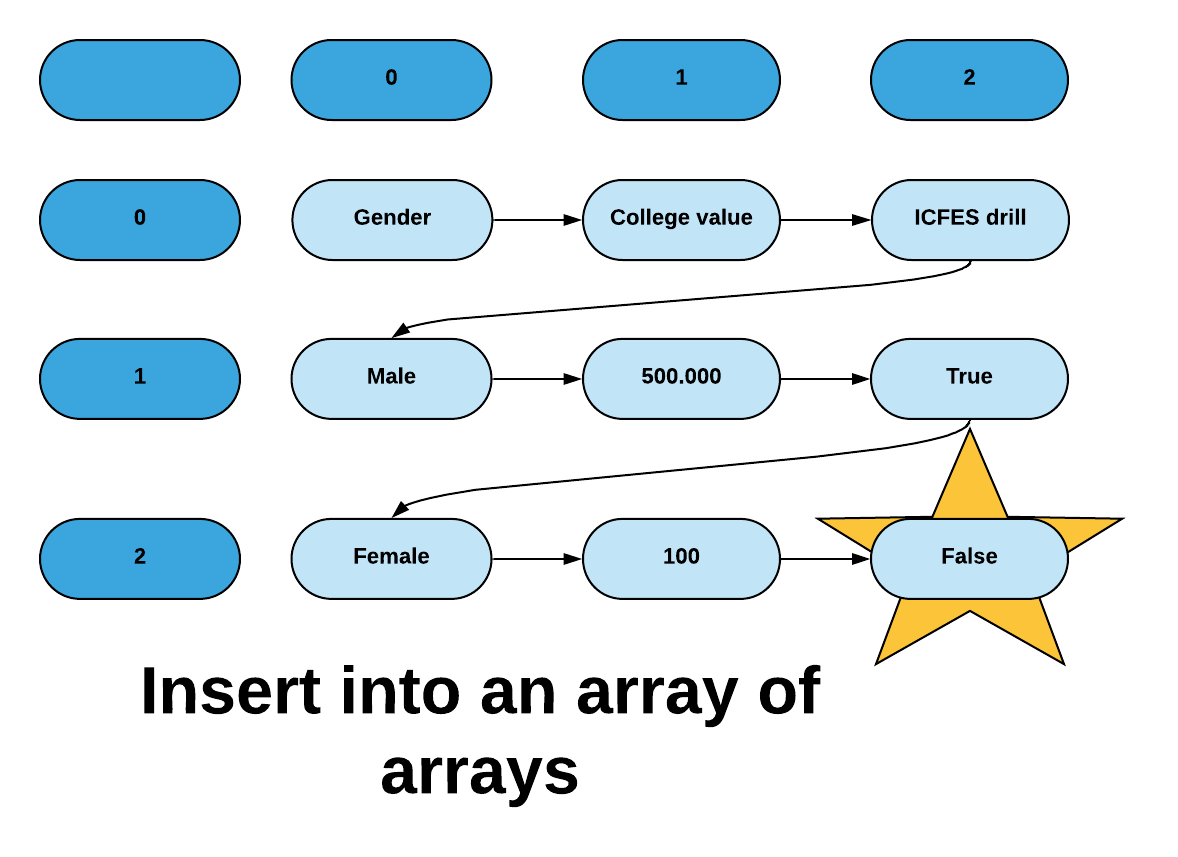


**Figure 1**: Floating type array arrangement. Where each row represents the students and each column the notes of each one

## **5.1 Operations of the data structure**



**Figure 2:** Search for data in an array of arrays



**Figure 3:** Insert data into an array of arrays

**5.2 Design criteria of the data structure**

When choosing a structure to save the data, it was chosen to use an Array of Arrays of Float type, where the rows are the students and the columns are the scores that the students will obtain in each subject of the ICFES tests. In the files that are read, each student has 78 attributes, of which we consider that the scores of the subjects are the most important data. When we specify this arrangement with type Float, it can be seen that the arrangement can only have data of floating types , causing many data that is not necessary to be deleted instantly, such as the city or the gender of the school in which the student studied, therefore it is necessary to specify which are the columns that we are going to read. This structure was also chosen because knowing the number of students in a file (Number of rows), the matrix can be built much faster, making the code much more efficient.

When choosing this structure, we also consider the complexity of access. When starting to create the binary tree, comparisons are made between the data all the time, this means that the algorithm has to be constantly accessing the data in the array, and since the access complexity is o (1), this makes the decision tree much faster to complete.

**5.3 Complexity analysis**

Derive the complexity of each operation of the data structure for the worst case and best case, as an example, this is a way to report the complexity analysis

**Table 5:** Table to report complexity analysis

## **5.4 Execution time**

|  |  |  |
| --- | --- | --- |
|  | **Tiempo contsrucción del árbol (milisegundos)** | **Espacio construcción del árbol (MiB)** |
| **Find the number of rows** | 15803 | 0.5 |
| **Insert data with defined row numbers** | 63055 | 1.2 |
| **create root** | 98341 | 2.2 |
| **create node** | 120624 | 3.4 |
| **predict success** | 185071 | 4.2 |
| **compare occurrence** | 206.13 | 5.3 |

## **Table 6:** Execution time of the operations of the data structure for each data set.

## **5.5 Memory used**

# Memory used

|  |  |
| --- | --- |
| **Students** | **Memory Consumption (mb)** |
| **5000** | 19 |
| **15000** | 28 |
| **25000** | 42 |
| **19255** | 38 |
| **35000** | 66 |
| **45000** | 79 |
| **57765** | 83 |
| **75000** | 91 |
| **105000** | 107 |
| **135000** | 122 |

**Table 3.** Memory consumption according to the total

## **5.6 Result analysis**

You can see how the runtime gradually increases for the amount of data analyzed, there is a value a bit far from the others at the end, but it is because they were plotted following the order of the name of the datasets,

**Table 8:** Analysis of the results

**6. CONCLUSIONS**

The main objective of the project was to create a decision tree that could predict the success of a student through the score they had obtained on the saber icfes tests.

The final data structure selected to store the student data was a float matrix, because it does not have much complexity in its operations, especially in the access operation, because many comparisons are made in the tree and it is constantly necessary to access the data. A floating type array was selected because the data used were those of the score, not to mention that these data allow the precision of the information collected and generated to be maintained. The data structure for the tree was CART, a binary type tree, this was needed since the separation of the data is done when deciding which is the best variable according to the Gini impurity calculated for each possible degree.

At the beginning of this practice, it had been decided to make a string type matrix, because the idea was to save all the possible data, however after analyzing, the conclusion was drawn that the most important data were the scores obtained by each student, since the more the data sets are divided, the lower the tree's precision. However, the idea of ​​the matrix continued in the same way. The most important thing is to make these predictions is not to take data that reach discrimination students

# **REFERENCES**

Reference sourced using ACM reference format. Read ACM guidelines in <http://bit.ly/2pZnE5g>

As an example, consider this two references:

1.Adobe Acrobat Reader 7, Be sure that the references sections text is Ragged Right, Not Justified. <http://www.adobe.com/products/acrobat/>.

2. Fischer, G. and Nakakoji, K. Amplifying designers’ creativity with domainoriented design environments. in Dartnall, T. ed. Artificial Intelligence and Creativity: An Interdisciplinary Approach, Kluwer Academic Publishers, Dordrecht, 1994, 343-364.

1. ID3 algorithm <https://en.wikipedia.org/wiki/ID3_algorithm>

2. C4.5 algoritm

<https://en.wikipedia.org/wiki/C4.5_algorithm>