SABER PRO SCORE. USAGE OF DATA STRUCTURES THAT WILL TELL IF YOUR PAST COULD DEFINE YOUR FUTURE

Manuela Zapata Mesa
Universidad Eafit
Colombia
mzapatam1@eafit.edu.co

María Alejandra Moncada Agudelo
Universidad Eafit
Colombia
mamoncadaa@eafit.edu.co

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

ABSTRACT
The objective of this study is creating an algorithm based on decision trees, which predicts the Saber Pro test score, based on different social, economic, and academic variables. Finding the solution to this problem would be very useful for universities as they could change their approach in order to improve their average score and by using the information provided by the algorithm the adjustments will be much easier and efficient. The four similar problems used in the report are algorithms frequently used in artificial intelligence with a very similar function among them.

We constructed a decision tree based in the Cart algorithm because we consider that this provides the most accurate solution. We made a prediction of student's success and we conclude that this kind of algorithms are very useful to make predictions and also for society.

Keywords

Saber pro test, decision trees, algorithm, analysis.

ACM CLASSIFICATION Keywords

Computing methodologies → Machine learning → Machine Learning algorithms → Dynamic programming for Markov decision processes → Value iteration1.

1. INTRODUCTION

Currently, the world's universities have an amount of information about their students never seen before, this information in most cases is not properly used by the universities. With the data that is counted, there are endless things that could be done, which might be very useful for a university and even for a country; one of the most important things is the prediction of success in academic tests, information that would allow institutions to greatly improve its results and helps its students; Therefore, it is pertinent that each of the holders of this information have a system that can take the data they have and return predictions based on that material

2. PROBLEM

The problem is based on creating an algorithm that uses the decision trees to predict the students score in the Saber Pro tests, for this there is a series of social, academic and economic variables, which will be used to perform the prediction.

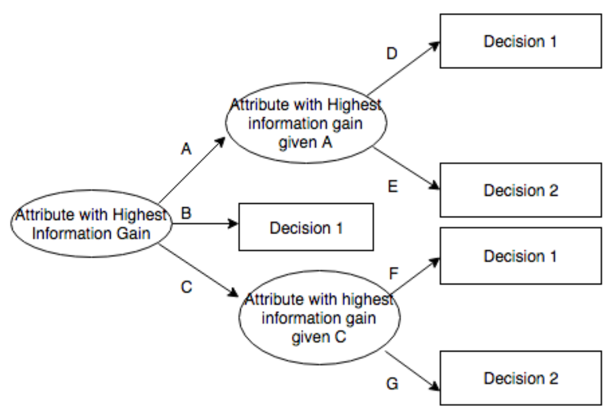
3. RELATED WORK

3.1 ID3 algorithm

Iterative Dichotomiser 3 is an algorithm invented by Ross Quinlan that is typically used in the machine learning and natural language processing domains.

This algorithm is used in artificial intelligence. This algorithm works based in a set of examples, each example has attributes; the attribute is a binary objective, from these examples the algorithm tries to obtain a hypothesis that classified new instances. ID3 does that constructing a decision tree.

This algorithm does not guarantee an optimal solution and is harder to use on continuous data that on factored data.



[1]

3.2 C4.5

Algorithm C4.5 is an algorithm invented by Ross Quilan, this algorithm is an extension of the ID3 algorithm. The decision trees created by C4.5 can be used for classification. C4,5 construct the tree in the same way that ID3 does, using examples; C4.5 choose an attribute that divides the example set in subsets. A few of the things that C4.5 has and ID3 does not:

-manage of continuous and factored data.

-manage of the example set with missing values.

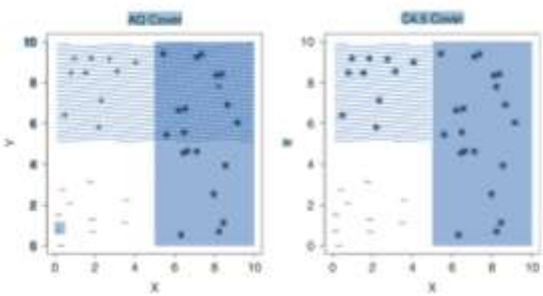
-manage of attributes with different cost.

-delete the parts of the tree that does not help

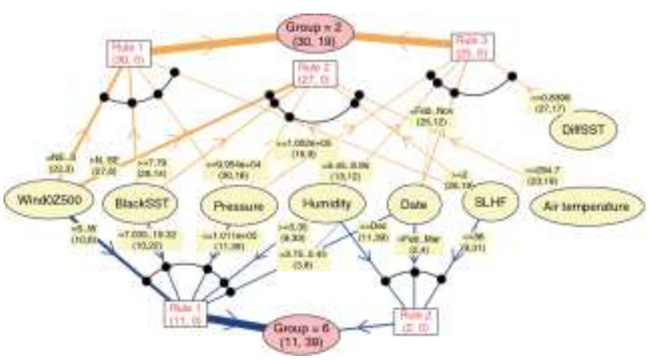
[2]

3.3 AQ algorithm

The algorithm quasi-optimal (AQ) is a powerful machine learning methodology aimed at learning symbolic decision rules from a set of examples and counterexamples. It has been applied to solve several problems from different domains, including the generation of individuals within an evolutionary computation framework. [3]



[3]



[3]

3.4 CART algorithm

The CART algorithm is structured as a sequence of questions, the answers to which determine what 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.

The basic idea of tree growing is to choose a split among all the possible splits at each node so that the resulting child nodes are the "purest". In this algorithm, only univariate splits are considered. That is, each split depends on the value of only one predictor variable. All possible splits consist of possible splits of each predictor.

A tree is grown starting from the root node by repeatedly using the following steps on each node.

1. **Find each predictor's best split**.

   For each continuous and ordinal predictor, sort its values from the smallest to the largest. For the sorted predictor, go through each value from top to examine each candidate split point to determine the best. The best split point is the one that maximize the splitting criterion the most when the node is split according to it.

2. **Find the node's best split.**

   Among the best splits found in step 1, choose the one that maximizes the splitting criterion.

3. **Split the node using its best split found in step 2 if the stopping rules are not satisfied.[4]**

[4]

4. Matrix

estu_consecutivo.1	estu_exterior	periodo
SB11201320218705	NO	20183
SB11201320132366		20152
SB11201220495594	NO	20173

Figure 1: Matrix n*m, where the number in each line is an individual and each column is the information, we have about them.

4.1 read_Dataset

Figure 2: The first thing method read data set does is open the document with Open() where it converts the document to cvs.reader that converts the document into an iterator that is delimited by ;

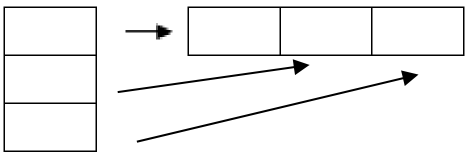


Figure 3: Then in a for, you take each line of the iterator, that converts into an array when it goes through the for. And it fills again a new arrangement called data, which will be an array of arrays, therefore it will be a matrix

4.2 Design criteria of the data structure

We decided to choose a matrix as the structure to store our data, due to it´s low complexity to get access to the data (O(1)), which we think can reduce significantly the complexity of the entire program, because having access to the data will be one of the most used functions in our algorithm. Also, its columns and rows division it´s going to make the needed data separation easier in order to reduce the code´s methods complexity.

4.3 Complexity analysis

Method	Complexity
Read_Dataset	O(n*m)

Table 1: Complexity**,** Being the number of rows n and m the number of columns

4.4 Execution time

	5000	15000	45000	75000	135000
Read_Dataset	0.0838s	0.2453s	0.8513s	1.2659s	2.7016s

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

4.5 Memory used

	5000	15000	45000	75000	135000
Read_Dataset	114,835,456 bytes	115,580,928 bytes	116,199,424 bytes	117,252,096 bytes	118,202,388 bytes

Table 3: Memory used for each operation of the data structure and for each data set data sets.

4.6 Result analysis

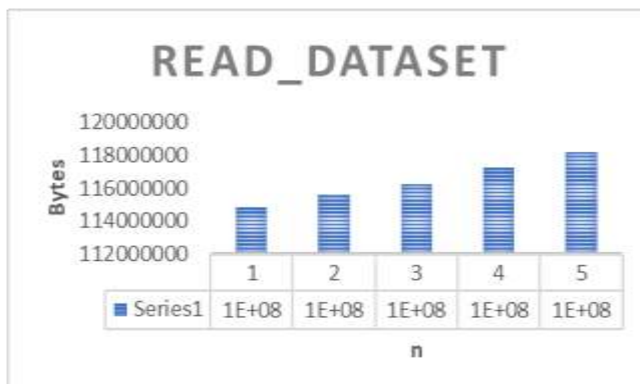


Table 4: Analysis of time results

4. CART

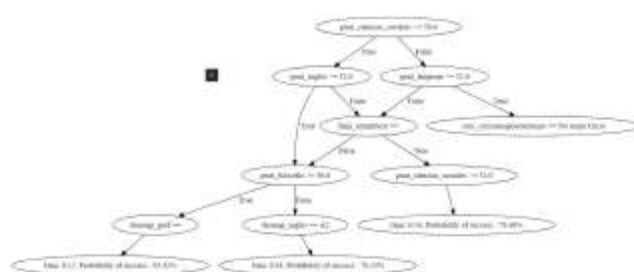


Figure 1: Tree created with the CART algorithm wrote from scratch

5.1 Operations of the data structure

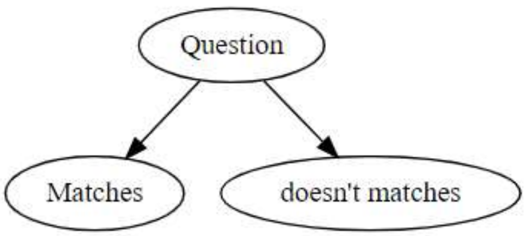


Figure 2: Partition method; separates the rows in two list, the true list and false list; based on a question.

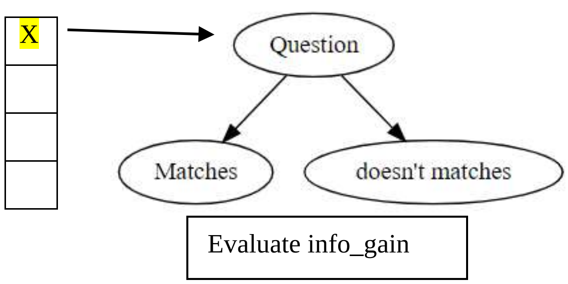


Figure 3: Find_best_split method, based in the values of the columns chooses the best question to split the data using information gain

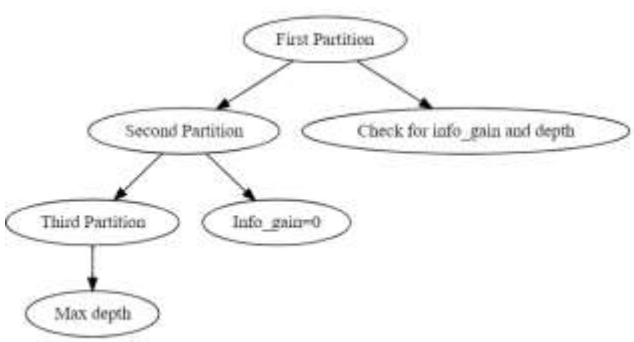


Figure 4: build_tree method, uses Find_best_split and recursion to create a tree, checking for the information gain and the max depth in every node

5.2 Design criteria of the data structure

Besides being the structure studied in the course, decision trees are one of the most used structures to generate predictions. We decided to construct ours based in the CART algorithm because it was more convenient for its easiness to be understood and to be programmed and since this is our first project using decision trees, this will work as an advantage to understand more complex algorithms in the future.

5.3 Complexity analysis

Method	**Complexity**
Class_counts	O(n)
Is_numeric	O(1)
Question_init_	O(1)
Question match	O(1)
Partitions	O(n)
Gini	O(n)
Info_gain	O(n)
Find_best_split	O(m*n)
Leaf_init_	O(n)
Build_tree	O(2^(m*n))
Classify	O(n^2)
Print_leaf	O(m)
Covert	O(m*n)
Predicted	O(m)

Being the number of rows n and m the number of columns

5.4 Execution time

	lite	Data 0	Data 1
Complete Algorithm	0.348s	27 m	188m

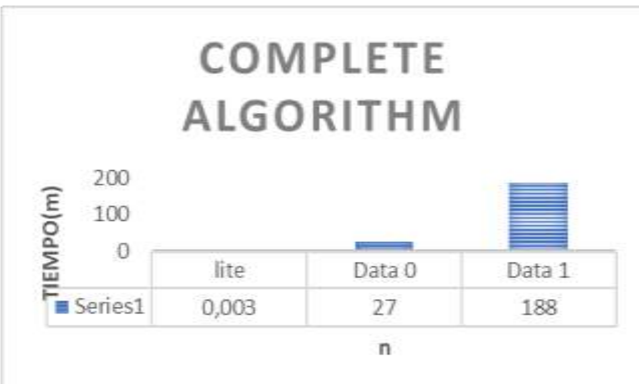
Table 6: Execution time of the operations of the data structure for each data set.

5.5 Memory used

	lite	Data 0	Data 1
Complete Algorithm	128,221,184 bytes	222,990,336 bytes	333,176,832 bytes

Table 7: Memory used for each operation of the data structure and for each data set data sets.

5.6 Result analysis



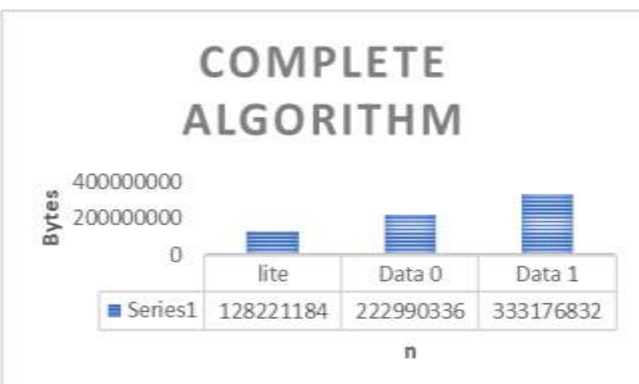


Table 8: Analysis of the results

6. CONCLUSIONS

Our prediction model it´s based in the Cart algorithm and counts with several methods that together help to build a decision tree that predicts the success of students based in a set of variables of different kinds (social, economic, familiar,…, etc). The results we obtained with this model are very fulfilling because our code predicts the percentage of possibility that some individual (a student) is successful or unsuccessful.

The most important result we obtained in this project is that our algorithm makes predictions, because we tried several types of implementation for the Cart algorithm but due to lack of understanding in some of the codes, we were not able to make predictions with the code. However, we managed to simplify the algorithm and obtained results. That was the most important moment in the realization of the code.

We believe that in the future we will be able to improve both complexity and space in memory, besides improving the Cart algorithm implementation so our results can be more accurate.

6.1 Future work

We would like to improve the algorithm´s complexity being able to use other structures with higher complexity and maybe more difficult to handle but much more efficient for this work. Also, being able to identify faster when a code has an error and be more aware about the inner functioning of the structures we are using.

ACKNOWLEDGEMENTS

We thank for assistance with printing the tree to the student Luisa Toro from Eafit and we also thank Juliana Restrepo from Eafit for the help with memory measure

REFERENCES

[1] Anon. 2020. ID3 algorithm. (May 2020). Retrieved June 1, 2020 from https://en.wikipedia.org/wiki/ID3_algorithm

[2] Sefik Serengil et al. 2020. A Step By Step C4.5 Decision Tree Example. (April 2020). Retrieved June 1, 2020 from https://sefiks.com/2018/05/13/a-step-by-step-c4-5-decision-tree-example/

[3] Guido Cervone, Pasquale Franzese, and Allen P.K. Keesee. 1970. Algorithm quasi-optimal AQ learning: Semantic Scholar. (January 1970). Retrieved June 1, 2020 from https://www.semanticscholar.org/paper/Algorithm-quasi-optimal-AQ-learning-Cervone-Franzese/2b3caed191cef2c27aa5e0cf81c72174722313b3

[4] Francesca Lucaroni, Domenico Cicciarella Modica, Mattia MacIno, Leonardo Palombi, Alessio Abbondanzieri, Giulia Agosti, Giorgia Biondi, Laura Morciano, and Antonio Vinci. 2019. Can risk be predicted? An umbrella systematic review of current risk prediction models for cardiovascular diseases, diabetes and hypertension. *BMJ Open* 9, 12 (2019), 1–6. DOI:https://doi.org/10.1136/bmjopen-2019-030234

[5] Javier Trujillano, Mariona Badia, Luis Servi, Jaume March, and Angel Rodriguez-Pozo. 2009. Stratification of the severity of critically ill patients with classification trees. *BMC Med. Res. Methodol.* 9, 1 (2009). DOI:https://doi.org/10.1186/1471-2288-9-83