ALGORITHM FOR THE PREDICTION ON THE ICFES SABER PRO EXAM RESULTS

Samuel Ceballos Posada
Universidad EAFIT
Colombia
sceballosp@eafit.edu.co

Pedro Botero Aristizábal
Universidad EAFIT
Colombia
pboteroa@eafit.edu.co

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

ABSTRACT

The Icfes Saber Pro Exam is a test that every student who is part of a university has to do in order to graduate. This exam, although not helping much for the professional future of the student, is a way for universities to understand their weaknesses and look for the points I which they should focus their academic programs.

The algorithm uses previous information and exams to predict how each student will do on the Saber Pro, by analyzing their result on the Icfes exam made at the end of High School and other information collected including family wealthness, time using technology devices, city of living, etc. and putting the students on certain group using classification that determine how good he/she will do on the exam. This algorithm has previously been used by Universities in other countries to look for further ways of improvement on their curriculums. The solution proposed for this problem was a binary decision tree implementing the CART algorithm.

Keywords

Exams; Predictions; Data Structures; CART Algorithm; Complexity

ACM CLASSIFICATION Keywords

•Applied computing~ Law, social and behavioral sciences ~ Sociology

•Software and its engineering ~ Software organization and properties ~ Contextual software domains ~ Operating systems ~ File systems management

1. INTRODUCTION

Universities operate in a very competitive environment, they are constantly confronting other institutions in order to attract the best students and making sure they stay. Also, the Saber Pro exam results play an important factor in university rankings, so they care a lot about student performance.

It is a fact that universities collect a lot of data from each student when they are applying. More specifically, universities in Colombia ask for the results of the ICFES exam, which is never used again after the student is accepted in the institution. With the implementation of a decision tree algorithm, universities could use the data from the ICFES exam to determine how a student will perform in the Saber Pro exam. This way universities could see what are the areas in which students struggle the most, thus make adjustments to their curriculum in order to improve performance in the exam.

2. PROBLEM

The main objective of this project is to establish an algorithm that is able to predict how a person will do on the ICFES Saber Pro exam based on information that includes habits, results on previous exams and basic information.

The solution of this project will help universities to discover where their weaknesses are at and try to implement new techniques to help their students grow better on those aspects.

3. RELATED WORK

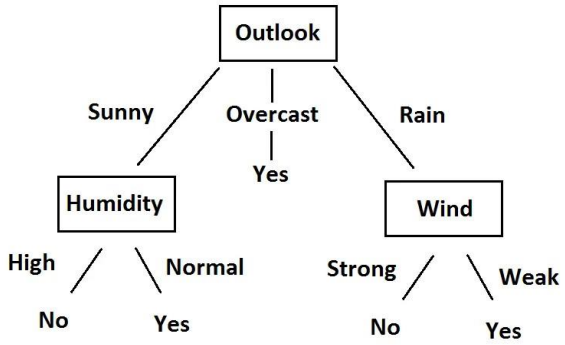
3.1 ID3

The ID3 algorithm, also known as Iterative Dichotomiser 3 is a recursive algorithm invented by Ross Quinlan in 1975, used to generate a decision tree from a fixed set of examples attempting to create the smallest tree possible. The resulting tree is used to classify future samples. It is most commonly used in machine learning and natural language processing.

ID3 works with the following steps:

1. Calculate the entropy of the attributes.
2. Calculate the Information Gain for the attributes, then split the set into subsets using the attribute with the minimum entropy.
3. Find the attribute with the maximum information gain.
4. Recourse until the desired tree is found.

For example, the table is a data set of factors to play tennis outside, which produces the following tree:



Even though ID3 is the most common decision tree algorithm, it has some disadvantages. For example, all

attributes must be nominal values, the data set cannot have missing data and the algorithm tends to fall into overfitting.

3.2 C4.5

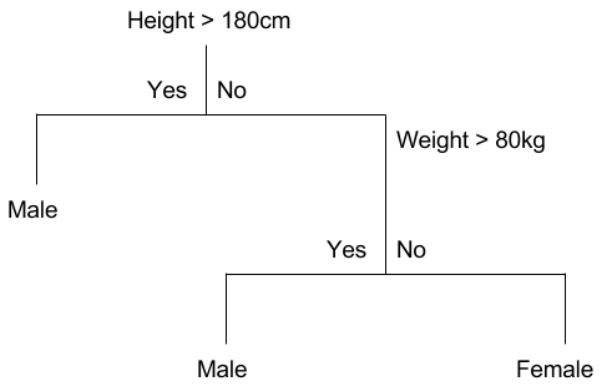
C4.5 is another algorithm created by Ross Quinlan based on ID3. It generates a decision tree where each node splits the classes based on the Information Gain. The attribute with the highest Information gain is used as the splitting criteria, just like ID3.

The main difference between the two is that C4.5 can use discrete and continuous attributes and it can handle missing attribute values.

3.3 CART

First introduced by Leo Breiman during the 1980's. The CART algorithm, also known as Classification And Regression Algorithm, is used to predict results based on different types of modeling problems. Due to its simplicity, it is commonly known as a decision tree, but recently many platforms have started to use the term CART when referring to it. This algorithm is commonly used as a foundation for more advanced algorithms, like bagged decision trees, random forest and boosted decision trees.

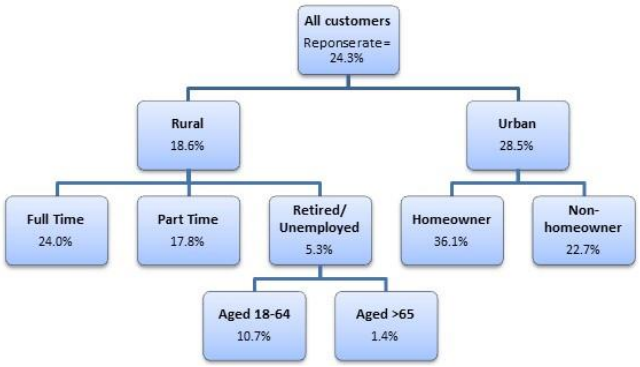
The CART model is represented by a binary tree. Where each root node holds an input variable which splits into boolean values (True or False). Then, the leaf node contains output variables which are the ones used to make a prediction on the data. The purity of the model can be tested by using the Gini Index Function, which uses the probabilities to determine how effective the model is. The lower the index, the purest the model and vice versa.



3.4 CHAID

The CHAID algorithm (Chi-square Automatic Interaction Detector) was first brought into attention by Gordon V. Kass in 1980. This method is used to determine and discover relationships between categorical response and predictor variables, in order to discover patterns that may be difficult to visualize in an unorganized manner. It then uses the chi-square test to measure and analyze how different the model data is compared to the expectations it had.

This method is commonly used on direct marketing to determine how a certain group of customers may respond to a campaign or advertisement made by a specific enterprise. complicated and follow a logic that is easy to understand. Also, they permit to append information at any memory space inside the array making it easier to control.



4. List of Lists

A list of lists is a very simple way to organize large quantities of information, its operations are not very complicated and follow a logic that is easy to understand. Also, they permit to append information at any memory space inside the list making it easier to control.

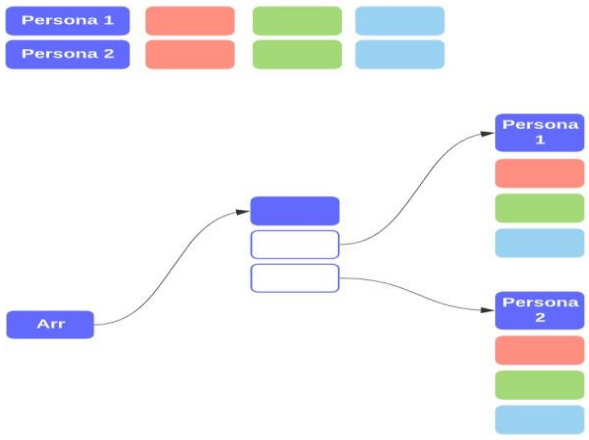


Figure 1: List of lists. Each inner lists contains all the info needed for one person. Each color represents a different variable.

4.1 Operations of the data structure

4.1.1. Data Append

By appending data into a list, you can insert information at the end of the list and the structure will change according to its new size and elements.

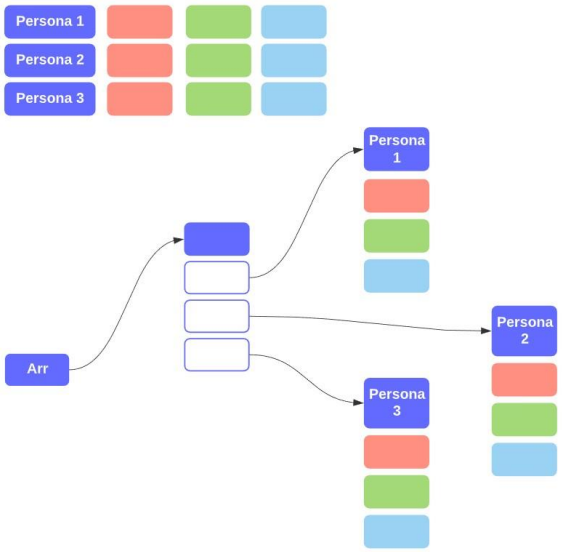


Figure 2: Data Append operation on the previous list.

4.2 Design criteria of the data structure

A list of lists is a very simple way to organize data where similar elements are clustered in bigger items. In this case, the elements would be each feature

4.3 Complexity analysis

Function	Complexity
Data Append	O(n)

Table 1: Report of the complexity analysis

4.4 Execution time

Create	
Dataset	**Runtime (s)**
0	0,56
1	1,72
2	2,96
3	3,95
4	5,56
5	2,11

Table 2: Time taken for each operation of the data structure

Dataset	Best time (s)	Worst time (s)	Average time (s)
0	0,54	0,58	0,56
1	1,57	2,48	1,72
2	2,74	4,00	2,96
3	3,78	4,82	3,95
4	5,47	5,65	5,56
5	1,96	2,99	2,11

Table 3: Best, worst and average runtime for each dataset.

4.5 Memory used

Create	
Dataset	**Memory Used**
0	126,58
1	247,03
2	367,47
3	487,65
4	607,94
5	296,82

Table 4: Memory used for each operation of the data structure and for each data set.

Dataset	Best memory (mb)	Worst memory (mb)	Average memory (mb)
0	126,5	126,68	126,58
1	246,96	247,16	247,03
2	367,3	367,69	367,47
3	487,48	487,73	487,65
4	607,74	608,21	607,94
5	296,63	297,08	296,82

Table 5: Best, worst and average memory use for each dataset.

4.6 Result analysis

In terms of efficiency, the data structure chosen for this project works on what would be called a good manner. It is very easy to understand and every aspect in terms of memory and runtime isn't bad. For this kind of project, the list of lists is a very good data structure.

Dataset	Memory Used (mb)	Runtime (s)
0	126,58	0,56
1	247,03	1,72
2	367,47	2,96
3	487,65	3,95
4	607,94	5,56
5	296,82	2,11

Table 6: Analysis of the memory used and the runtime

5. Binary Decision Tree

A binary decision tree is a very useful if the questions that are being used can be answered with yes/no or true/false. This permits the user to go through multiple lines of data and predict an outcome for each of them. The algorithm we used is called CART (Classification and Regression Tree).

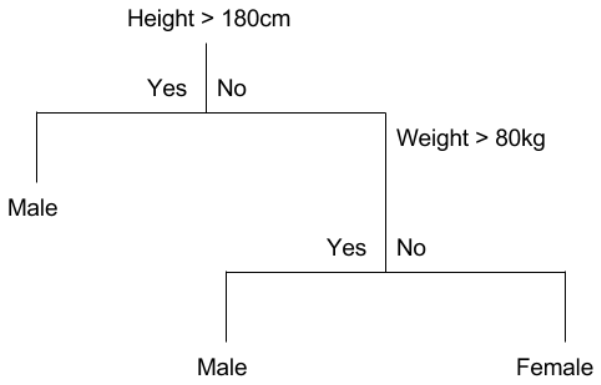


Figure 3: A graphical representation of a binary search tree, specifically a CART algorithm.

5.1 Operations of the data structure

There are two main operations when using a binary decision tree: creating the tree itself and using it to get information about new data.

5.1.1 Creating the Tree

To create the tree, the columns are separated in numerical and categorical, and we need to enter a data frame with all the nada needed in order to create and train the tree for the first time. The data needs to be separated into these two categories because a feature like "How much time does the student uses to surf the internet per day", the answer to this question is recorded with answers like '30 minutes or less' and 'More than 2 hours', answers that can be buffered with an "equals to" operation. On the other hand, a question like the score a certain student got in a subject like biology, can't be compared with the same operand, as this would create a very small data set that meets each specific grade, instead we use the operand 'greater than' so that the rows from the data frame that meet the question has a better proportion compared to the ones that don't meet it.

After the first question is made, a recursive call is made with two new data frames that include the lines that meet and don't meet the requirement made by the question respectively. It goes like these until the information gain made by the question is 0 or the maximum depth the user wants the tree to have is reached

5.1.2 Testing the Tree

Now in order to test the tree, the function receives the tree that got preciously created and a new data frame that is different to the one used for the creation and training of the tree. This time the method makes a recursive call that ends when a node of the tree is considered a leaf, which means that it has no more nodes to either side of it. When a leaf is found, the method returns the prediction for that leaf. If the node is not a leaf, the function then calls itself back using the same tree but looking at the node that are at its sides. In other words, it looks for the true or false nodes taking into consideration its own question.

5.2 Design criteria of the data structure

The more difficult part on the design is the creation of the tree as it needs to be efficient enough to have small runtime and memory use but, have as well the accuracy needed in order for the tree to work properly and return values that are almost always equal to the ones it should theoretically return. After this, the algorithm should go itself through each student and predict whether he will do good or bad in this national exams for Bachelor's Degree's graduates.

5.3 Complexity analysis

	Complexity
Creating	$O(n^2*m)$
Testing	O(n*m)

Table 7: Report of the complexity analysis

5.4 Execution time

	Time for each data set (s)					
	0	**1**	**2**	**3**	**4**	**5**
Creation	22,08	73,30	130,95	181,91	231,90	130,31
Testing	0,05	0,16	0,28	0,41	0,53	0,23
Total	22,13	73,46	131,22	182,32	232,43	130,54

Table 8: Execution time of the operations of the data structure for each data set.

5.5 Memory used

	Data sets					
	0	1	2	3	4	5
Memory used (mb)	74,2	194,74	314,88	435,09	555,46	244,06

Table 9: Memory used for each operation of the data structure and for each data set data sets.

5.6 Result analysis

When it comes to accuracy, the data structure used wasn't as good as expected, although they are considered to have a good accuracy for a decision tree: On the other side, the time and memory used by the algorithm shows that it is efficient as it can run large amounts of data in what is considered to be a small time frame and memory use.

Dataset	Time Creating (s)	Time Testing (s)	Memory Used (mb)	Accuracy (%)
0	22,08	0,05	74,2	71,22
1	73,30	0,16	194,74	71,86
2	130,95	0,28	314,88	70,74
3	181,91	0,41	435,09	72,26
4	231,90	0,53	555,46	70,85
5	130,31	0,23	244,06	70,91

Table 8: Analysis of the results

6. CONCLUSIONS

Based on the fact that education is already one of the most important aspects on life and it is considered the be the most important in the future upon us. It is mandatory that everyone gets to understand and reach new levels to classify students based on one of the most used means of classification, the exams. Hence the importance of this project and many others that try to understand and reach new ideas that can be of help on the near future.

On this project, we managed to predict on a good accuracy, how good can a student do on a future exam based on an exam previously taken by that same person. This project can be of massive help for institution that are looking to classify students looking for a scholarship or a letter of acceptance. Also, based on the fact the first attempt of this structure gave a 0% accuracy, it is pretty clear that the report grew and got better, as it reached an accuracy of approximately 72%.

6.1 Future work

In the future, this project could be improving by making it more complex, having a full random forest instead of a single decision tree. Also, it could research in other exams to make it be an international approach to student's classification.

ACKNOWLEDGEMENTS

We would like to thank Universidad EAFIT for giving us the opportunity to enroll in the course in which this project was done. We would also like to thank fellow students, Luisa Toro Villegas, Gregorio Perez Bernal and Juliana Restrepo Tobar for giving us ideas that we could use to get our project running the best way possible.

REFERENCES

1. Jason Brownlee. 2019. Classification And Regression Trees for Machine Learning. (August 2019). Retrieved February 9, 2020 from https://machinelearningmastery.com/classification-andregression-trees-for-machine-learning/

2. T. Santhanam and Shyam Sundaram. Application of CART Algorithm in Blood Donors Classification. Journal of Computer Science 6, Tamil Nadu, 2010 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.8749&rep=rep1&type=pdf

3. Sarah Littler. 2018. CHAID (Chi-square Automatic Interaction Detector). (May 2018). Retrieved February 9, 2020 from https://select-statistics.co.uk/blog/chaid-chisquare-automatic-interaction-detector/

4. Sefik Serengil. 2017. A Step by Step ID3 Decision Tree Example. (November 2017). Retrieved February 9, 2020 from https://sefiks.com/2017/11/20/a-step-by-step-id3-decision-tree-example/

5. Sefik Serengil. 2018. A Step By Step C4.5 Decision Tree Example. (May 2018). Retrieved February 9, 2020 from https://sefiks.com/2018/05/13/a-step-by-step-c4-5-decisiontree-example/