Informatics & Systems Department

Academic period 2020-2

Success prediction model in the Higher Education Quaity Exam (Saber Pro) using decision tree-based learning

Juan Felipe Agudelo Velez - jfagudelov@eafit.edu.co
Jose Manuel Ramírez Gomez - jmramirezg@eafit.edu.co
Steven Oviedo Aristizabal - soviedoa@eafit.edu.co
Diego Velásquez Varela - dvelasquev@eafit.edu.co

EAFIT University
Informatics & Systems Department
Colombia
Medellín
050022
T: +57 (4) 261 9500

Abstract

The purpose of this report is to specify the factors that intervene in the success rates of colombian students who take the Higher Education Quality Exam (Saber Pro) by analyzing their score in the State Examination of Middle Education (Saber 11) and socioeconomic factors.

To identify trends in student achievement, we developed a CART decision tree-based algorithm that predicts the probability that a student will score above average. Taking this into account, risks and opportunities can be identified that help state institutions (and higher education) to make decisions for the improvement of education in Colombia, such as finding investment sectors, awarding scholarships, or redesigning educational curricula.

Keywords

- Machine Learning
- Artificial Intelligence
- Applied mathematics
- Operations research
- Higher education

Acknowledgements

We thank our EAFIT University for the quality with which its courses are delivered, as well as the professor Gloria Stella Sepulveda Cossio and the academic assistant Luisa Toro Villegas for their excellent commitment to learning.

Contents

Chapter 1

Introduction

In Colombia, the coverage rate to access higher education has been constantly increasing during the last 15 years, in 2018, as a result of the efforts of the country's public and private entities to democratize education, the coverage was located at 52.01% according to the figures published by the National Information System on Youth and Adolescents of Colombia (JUACO).
Although it is undeniable that increased access to university is a positive result, academic dropout continues to be one of the obstacles that must be overcome; According to figures from the World Bank, 42% of Colombian students drop out of their career. The present work is determined through a decision tree-based learning algorithm (CART) which are the main factors that influence the success of a student.

1.1 Aims and Objectives

1. Undertake a relevant background study to identify existing work in the area, and to identify appropriate techniques which can be adopted to produce a solution in this project.

2. Create a decision tree that identifies what influences the success of a student in Colombia in the Saber Pro higher education quality exam according to socioeconomic data and their score in the Saber 11 high school state exam.

3. Evaluate results using an appropriate framework or set of success criteria that are clearly related to the problem we set.

1.2 Project Approach

Our research is framed within the axis of the prediction of student success and its processing through an intelligent system. In this sense, we carry out an analysis of various sources of information to first, provide a solution to the problem investigated through the selection, design, and development of a Machine learning model to achieve a correct classification and prediction, and second, to contribute to our learning experience as first-year mathematical engineering students.

Chapter 2

Related Work

Decision trees are predictive modeling tools capable of performing classification and regression tasks, they are widely popular for being simple and for producing results that are easy to explain with a small amount of data. There are different classes of trees that fit a type of problem, below we explain the best known.

2.1 Key terminology

Gini impurity: The Gini impurity measure is one of the methods used in decision tree algorithms to decide the optimal split from a root node, and subsequent splits. To put it into context, a decision tree is trying to create sequential questions such that it partitions the data into smaller groups. Once the partition is complete a predictive decision is made at this terminal node (based on a frequency).
In other words Gini Impurity tells us what is the probability of misclassifying an observation. Note that the lower the Gini the better the split. In other words the lower the likelihood of misclassification.[2]

Entropy: As it relates to machine learning, is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. [3]

Information gain: Is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees, is calculated by comparing the entropy of the dataset before and after a transformation.[4]

2.2 ID3

ID3 algorithm creates a hypothesis about a specific topic, starting from a sample set. This set is made by tuples of values, which ones we'll call attributes, and between these tuples, there's one that we're gonna sort out, this specific attribute is binary, Which means that it only has two possible values. In this way, the algorithm tries to get a statute that can sort out the event for future instances.
It achieves this goal by creating a decision tree, this tree has 3 elements:

- Node: these ones contain the attributes.
- Archbow: these contain the possible values from the main node.

- Leaf: these are nodes that sort out the sample as a positive value or a negative value.

To calculate the entropy we use the next formula:

$$Entropy = \frac{-p}{p+n} log_2(\frac{p}{p+n}) - \frac{n}{p+n} log_2(\frac{n}{p+n})$$

where p is how many positive samples there are and n how many negative there are. It should be taken into account if logarithm is positive or negative.

2.3 C4.0

he C4.5 algorithm is from the family of top-down decision trees (TDIDT). They belong to the inductive methods of machine learning, that is, they learn through previously classified examples.
In data mining, they are used to model the classifications in the data employing decision trees.
The mechanics of the algorithm can be explained as follows:
Let K be a training data set, with classes A1, A2, A3, ..., An. We choose the attribute that divides from more efficiently its set of samples in subsets in each node of the tree, and then, the process is repeated within each of the partitioned sublists.

This algorithm has some base cases:

- All the samples in the list belong to the same class. When this happens, it just creates a leaf node for the decision tree that tells you to choose that class.
- None of the functions provide information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- An instance of a class never seen before has been found. Again, C4.5 creates a decision node higher up the tree using the expected value.

2.4 CART

The CART algorithm consists of a binary tree that classifies both categorical (ordinal or nominal) and continuous or numeric variables. This is achieved through the creation of branches and nodes for each of the daughter leaves that represent a prediction on the data that is evaluated based on predictions given by previous nodes. Each node represents a question and its leaves represent a prediction.

However, since each classification is the product of a prediction. These can produce incorrect results such as classifying a case in the wrong class. That is known as impurity and represents the number of misclassified cases concerning the expected results. the impurity, despite having several methods for its measurement, we always seek to minimize it to obtain the most reliable predictions in the way that we

will show below.

Mathematics of the algorithm: In the CART algorithm, there are three decision criteria for the division and selection of the best option, are the Gini index, Towing criterion, and Towing ordering criterion. Simply, the Gini index can be measured as:

1 - (The square of the probability of choosing the right side of the leaf) - (The square of the probability of choosing the left side of the leaf)

The Towing criterion can be calculated with the following formula:

$$\Delta i(s,t) = p_L p_R \left[\sum_j | p(j \mid t_L) - p(j \mid t_R) | \right]^2 .$$

Where Delta i (s, t) is the probability of the best choice of a solution for a node t given a separation s.

The Towing sort order can be calculated with the following formula:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) = p_L p_R \left[\sum_{j \in C_1} \{ p(j \mid t_L) - p(j \mid t_R) \} \right]^2$$

Stop criteria

- If all the possible cases that a node has are identical, it is said that this is pure and the execution of the algorithm must end.
- If all cases in a node have the same probability for each predictor, the node does not separate and the algorithm must end.
- If the size of the tree exceeds a maximum limit, given by the capacity of the machine on which you are working, the tree growing process will stop as will the algorithm.
- If the size of a node is less than the minimum node size that has been specified by the user, the node will not be split.

Chapter 3

Methodology (Or Approach)

We have a total of 11 files that contain data related to the Saber Pro exams for the execution of the algorithm, 5 of them will be used for tests while the remaining 6 will be used to train the model. The data sets contain the following dimensions.

Name	Dimensions (Columns*Rows)
Test0	5000*78
Test1	15000*78
Test2	25000*78
Test3	35000*78
Test4	45000*78
Test_Lite	16*78
Train0	15000*78
Train1	45000*78
Train2	75000*78
Train3	105000*78
Train4	135000*78

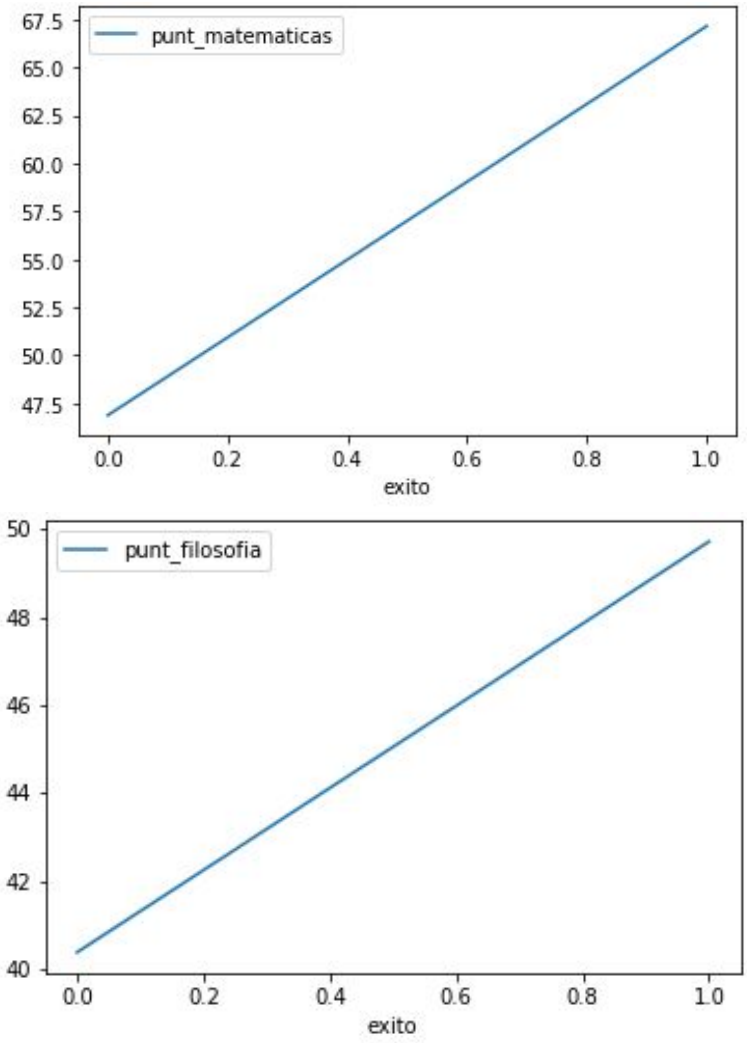
Where the rows are the labels of the academic, social, and economic data, and the columns the information corresponding to each of the students.

However, since the goal of the project is to determine a student's chances of success on the exam, many of the tags must be filtered to avoid causing problems within the execution of the program. In other words, knowing if Student A's code is "AB1450122011530" is not valuable information in determining whether they will be successful on the test. Which brings us to the next point.

3.0.1 Exporatory data analysis

The exploratory analysis of data is a critical process in which, supported by computational and statistical tools, we discover patterns, anomalies, and verify assumptions for better handling of a data set.
Within our data, we have a Boolean label called "success" where we will find a 1 if the student passed, and a 0 otherwise.

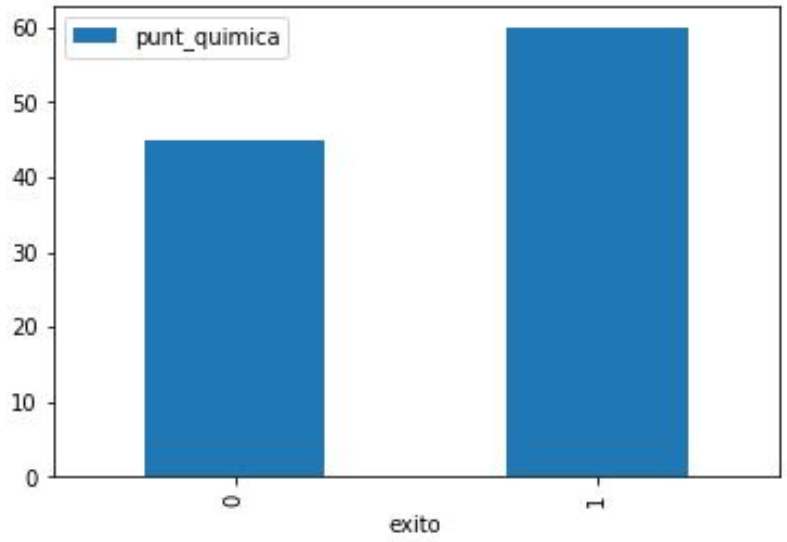
In this order of ideas, we began to relate the labels of the dataset with the success column, to clearly determine which were the factors that had an influence when determining whether or not a student passed their exam.



As can be seen in the previous figures, a high score in mathematics has a greater incidence of success than a good score in philosophy, it can be observed in the data while in philosophy with only 50 points there is a tendency to succeed, in mathematics 67.5 are needed.

Under the same methodology, it was observed that gender does not affect whether or not the exam is passed, women have a majority in both successes and failures, which is directly related to the demographics of the country (there are more women than men).

A good score in chemistry was also a determining factor in winning the exam, it can be seen that the students who failed obtained a maximum of 45 points.



Under this criterion, the rest of useful columns were obtained to determine the success of the student in the Saber Pro, which are:

- English level
- Socioeconomic level
- SISBEN score (survey by the Colombian government that takes into account
- social and economic factors to detect investment sectors)

With the labels chosen, we move on to the next point of implementation of our predictive model.

Chapter 4

Design

As we explained before, in the CART tree algorithm we need a Gini index and an information gain to divide our tree using the best questions to get the most precise predictions from the tree.
In this order of ideas, our code is entirely based on finding the best options for each new node created by the algorithm. As we are using huge amounts of data packed in CSV extension files, we decided pandas data frames to be the structure of the program, so we can easily manage them and also give the program the capacity of operating with other data frames.
The algorithm will be explained below:

- **Data filtering:** Given the amount of data recollected in this kind of exam, we must select the columns from the database that might provide the most useful information to the tree, even despite the fact that the tree itself chooses it. The reason for this is essentially reducing computation time. The program will need to go through every column and row to calculate the Gini index and information gain, increasing exponentially for large amounts of data, and therefore, reducing the number of places to visit will help to get faster to the answer. This selection was done as explained before in the Methodology section.

- **Filter_df:** This method catches all the data sets and then returns a new data frame without null values, and just using the columns defined on columns_to_use.

- **Is_value:** The job of this method is to verify if a value is an int or a float.

- **Class_counts:** This function counts the number of successes and failures that a determinate data frame has.

After the design of the auxiliary methods, we proceed to the construction of the tree that is explained in the next chapter.

Chapter 5

Implementation

5.0.1 Building of the decision tree

The tree will recursively create new nodes that ask the best question finding the Gini index for the data frame received from the last node, then finds this question testing all possible questions to be asked with the information in the data received, seeking for the one with the highest information gain and the best Gini. At the end of this process, creates two child nodes for the node. One containing the values of the node that are true to the question generated and the other one for the ones that not. This whole process repeats for both child nodes till it finds a node that has 0 gain. This one will be a leaf and will contain the tree predictions. This is done with the next methods:

- **Question class:** It creates a question using the name of a column and a value. To match the values of the data frame that satisfies the question or not we implemented the match(self, value) method, which takes a value and compares it with the value in the question. If it is a numerical value, uses the operator "is less than or equal to" in another case, uses the operator "Is equal to".

- **Gini impurity method:** Given a data frame, the Gini method calls the class_counts method to get the information about what are the number of repetitions of 0 and 1, where the 0 represents failure and the 1 represents success. Then, with this information, the method finds the Gini impurity finding the probability of 0 and the probability of 1. This is possible by dividing the number of repetitions by the size of the data frame.

- **Information gain method:** This will let us know how much useful information the tree will get from a question. It is calculated using the formula shown (before). It finds the Gini index for both true and false branches using the Gini method explained before and then replaces it in the formula.

- **Partition method:** Given a data frame and a question, the partition method, divide the tree into two new nodes that contain the values of the data frame that meet the condition and the ones that don't. Besides this, the method saves on the data frame if the value was successful or not.

- **Find best partition method:** This method finds the best question based on the information gain. This function takes each value on each column and

creates a question and verifies the obtained gain. Based on this, the function compares which question with which value generates more gain and the highest, and then return it as the select question.

To optimize our implementation, we carry out a method that applies the random forests. which are explained below.

5.0.2 Random Forest

Random Forests are a set of decision trees that use bagging (the combination of different machine learning strategies) in this method, different trees see different portions of the data. No tree sees the training data. What makes each tree is trained with different data samples for the same problem. Thus, when combining the results, some are compensated with others and we obtain a prediction that generalizes better.

Chapter 6

Testing and Evaluation

The complexities of the methods of our model are shown below.

Columns_to_use	O (1)
Filter_Df	O (n)
Class_counts	O (n)
Class Question ()	O (1)
Match	O (1)
Partition	O (n)
__repr__	O (1)
Gini	O(n)
Info gain	O (1)
Find best partition	O(n^2)
Leaf	O (1)
Decision Node	O (1)
Build Tree	O(n^2)
Class Tree	O (n^2)
Classify	O(n)
Print Tree	O(n)
Random Forest	O(n^2)

In total, the RAM memory expenditure was 800MB.

The tree construction matrix, with the filtered data, follows the following structure for each of the questions generated by the model.

```
0.49743589743589745 Is desemp_ingles == B1?
   estu_depto_reside.1 estu_mcpio_reside.1  ... desemp_ingles exito
0         CUNDINAMARCA          FACATATIVÁ  ...            A1     0
1              CORDOBA              LORICA  ...            A-     0
2            ANTIOQUIA            MEDELLÍN  ...            A-     0
3            ANTIOQUIA            MEDELLÍN  ...            B1     0
4                VALLE                CALI  ...            A1     0
5                VALLE                CALI  ...            B+     1
6               BOGOTA         BOGOTÁ D.C.  ...            A1     1
7               BOGOTA         BOGOTÁ D.C.  ...            A-     0
8            MAGDALENA         SANTA MARTA  ...            A1     0
9               BOYACA               TUNJA  ...            A1     1
10           SANTANDER         BUCARAMANGA  ...            B+     1
11              BOGOTA         BOGOTÁ D.C.  ...            B1     1
12              BOGOTA         BOGOTÁ D.C.  ...            A2     1
13              BOGOTA         BOGOTÁ D.C.  ...            A-     0
14              BOGOTA         BOGOTÁ D.C.  ...            A-     1
```

However, when printing our CART, we found a problem (which has not yet been detected) that causes the tree to not print completely, with the TEST LITE, we obtain:

```
Is desemp_ingles == B1?
--> True:
--> False:
  Is punt_filosofia >= 30.0?
  --> True:
    Is cole_depto_ubicacion == VALLE?
    --> True:
    --> False:
      Is cole_depto_ubicacion == BOGOTA?
      --> True:
        Is punt_fisica >= 53.0?
        --> True:
        --> False:
      --> False:
        Is desemp_ingles == A1?
        --> True:
        --> False:
          Is punt_ingles >= 40.0?
          --> True:
          --> False:
  --> False:
```

Chapter 7

Conclusion

During the last decades, both state and private institutions have worked together to improve the Colombian educational system. For this, it is crucial to have knowledge of the academic areas in which they must work to achieve educational quality in primary, secondary, and higher education.

Technology is one of the tools of the century to address this type of problem, in our case, we developed (partially) a CART model that helps to identify the probabilities of success of a student who takes the Saber Pro higher education exam, from which, we can conclude that:

1. Gender is not a determining factor in a student's success on state tests.
2. The level of English, and a good score in mathematics and chemistry, are special enhancers of success for a student who will take the state test.
3. Academic success is directly related to the socioeconomic gap that exists in the country. Students who belong to higher social strata have a greater probability of success, which is in particular one of the determining decisions that must be made within the government so that resources are allocated for public education (to which it is common to access by vulnerable populations) to achieve quality education that closes the gap.
4. The code still needs to be refined to give clear solutions, so that the tree can be fully generated.

7.1 Future Work

In each project there are pending things to do, in our case, in order of priorities, the first thing is to refine the code so that it can return a CART tree in its entirety, the second, to implement in it the existing methods of random forests to obtain better predictions. At the same time, as outgoing first-year mathematical engineering students, we trust that in the course of our learning we will obtain the necessary tools to give continuity to the project.

Bibliography

[1] Guzmán Ruiz, C., Durán Muriel, D., Franco Gallego, J., Castaño Vélez, E., Gallón Gomez, S., Gómez Portilla, K., Vásquez Velásquez, J. (2009). Deserción estudiantil en la educación superior colombiana, Metodología de seguimiento, diagnóstico y elementos para su prevención (1.ª ed.). https://www.mineducacion.gov.co/sistemasdeinformacion/1735/articles-254702_libro_desercion.pdf

[2] Loaiza, S. (2020). Gini Impurity Measure – a simple explanation using python. https://towardsdatascience.com/gini-impurity-measure-dbd3878ead33

[3] Entropy in Machine Learning. (2017, December 12). Retrieved from https://study.com/academy/lesson/entropy-in-machine-learning.html.

[4] Brownlee, J. (2019, october). Information Gain and Mutual Information for Machine Learning. https://machinelearningmastery.com/information-gain-and-mutual-information/

[5] Gupta, P. (2017). Decision Trees in Machine Learning. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

[6] Geek For Geeks. Decision Tree. (2019) https://www.geeksforgeeks.org/decision-tree/

[7] Keno, L. (2020). Making Data Trees in Python. https://medium.com/swlh/making-data-trees-in-python-3a3ceb050cfd

[8] Gadige, M. (2020). Binary Trees in Python. 2020. https://www.educative.io/edpresso/binary-trees-in-python

[9] TK. Everything you need to know about tree data structures. 2017. https://www.freecodecamp.org/news/all-you-need-to-know-about-tree-data-structures-bceacb85490c/

[10] Valente, J. (s. f) Decision Tree from Scratch in Python. 2019. https://towardsdatascience.com/decision-tree-from-scratch-in-python-46e99dfea775