**Data structures and algorithms final project**

With this project based on the strategy of Machine Learning the objective is to predict the chances of achieving success on the test “Saber Pro”. For this, we took into consideration datasets which gave us personal information of the Colombian students who attended the test “Saber 11” and “Saber Pro” from 2014 to 2019, Giving us the results.

The process of this project will allow the Colombian community to know what are those factors that influence the success of their students in the test “Saber Pro”. Principally this will help identifying those areas that suffer more difficulties or with worse academic results. Thus, creating strategies that allow them to get better results.

**Strategies with the Dataset**

We considered that the data which will allow us to evaluate the success from the students on the test “Saber Pro” would depend from the scores gotten on the test “Saber 11” and “Saber Pro”, so we filtered the data, using the columns that provide the scores gotten in every section from which divides the test, for example: language, mathematics, biology, chemistry, physics, social sciences, philosophy, English, professional score and “success.”

To make the lecture and filter the data we used the libraries “pandas”, “NumPy” and the method filter exclusively to filter the data.

**Libraries:**

We used the algorithm **CART** to make our decision tree. To make the process easier we used the library “Sklearn” which is one of the most important libraries for working with Machine Learning models.

Clearly the library Sklearn has many sub-libraries, for this project we used:

* Metrics
* Ensemble
* Model Selection
* Tree

**Metrics:** It allowed us to make a confusion matrix and calculate the precision of the algorithm

**Ensemble:** It allowed us to import a random forest and his implementation.

**Model\_Selection:** It helped us for most of the project because it was used to train the model.

**Tree:** It allowed us the creation of the decision tree and to export its visualization with the library “GraphViz”

We also used other libraries like Pandas and NumPy for the reading of the datasets and the manipulation of DataFrames and GraphViz for rendering the generated decision trees.

**Code Implementation:**

To begin we decided reading the data using the command read\_csv from the library Pandas to save the files of data on the DataFrame and thus work easier with them. Then we did the cleaning of data, erasing some columns that did not provide nor remove efficiency on the model. Leaving the DataFrame less columns and more flexible operations.

Then, we divided the DataFrame with the data. Stored in a DataFrame called “X” with all the data that will be helpful for making the prediction, so that, all the columns that worked as predictive variables. And on “Y” we saved the variables that will be predicted. In this case it would be the only column of success on the test for each student.

Next, we used the function train\_test\_split to divide the data and create subsets of data from training and testing for the dataframe “X” and “Y”. leaving us with new called x\_train, x\_test and y\_test. These variables will be key for the implementation of the tree.

Now we will use the sklearn library to create a classification tree, so we create the clf tree and adjust it with the data from x\_train and y\_test. We can use our clf-tree, which is already trained to predict the results of the y\_test, which are the actual and own results of successful student testing. The predictions are stored in the variable y\_pred.

This leaves us with two results for each row of data: on one side we have a column with the actual test results, on the other side we have a column with the results predicted by the tree. In both cases it is classified with 1 or 0, to indicate respectively if the student was successful in the test or if in the other case, he was not successful as well

**Efficiency of The Algorithm**

With these data we can find out how accurate our algorithm was and thus know if we have to improve it or if it gives us a good estimate of the result. For this we will use the sklearn metrics sublibrary, which allows us to use the accuracy\_score, which is exactly the measure we are looking for, this command will return a number between 0 and 1 that will indicate the efficiency of the algorithm.

An "accuracy" close to one indicates a good efficiency in the algorithm, and an "accuracy" closer to 0 will mean that the algorithm does not give a good prediction of the real data

After calculating the efficiency, we will also be able to know the confusion matrix. This is a 2x2 matrix that tells us:

* The number of predictions that were positive and correct
* The number of predictions that were negative and correct
* The number of predictions that were positive and incorrect
* The number of predictions that were negative and incorrect

|  |  |  |
| --- | --- | --- |
| **Prediction** | | |
| **Positive** | | **Negative** |
| **observation** | **Positive** | Correct Positives | | Incorrect Negatives | |
| **Negative** | Incorrect Positives | | Correct Negatives | |

We see in the image that the correct predictions, both positive and negative, are found in the main diagonal of the matrix, while the failed predictions will be in the secondary diagonal. Since the idea is to build a model that is a good predictor, it is to be expected that if our model comes out well, in the main diagonal there will be almost all the data, while in the secondary diagonal there should be few data. The ideal case is where the secondary diagonal is full of zeros.

To build this matrix the metrics sublibrary will be very useful, since here we can find the confusion\_matrix command, which will automatically generate the matrix. The only thing we will need is to give as parameters to y\_test (the real success data) and to y\_pred (the predicted success data) and this will compare both columns placing the success and failure cases in the matrix in their corresponding place.

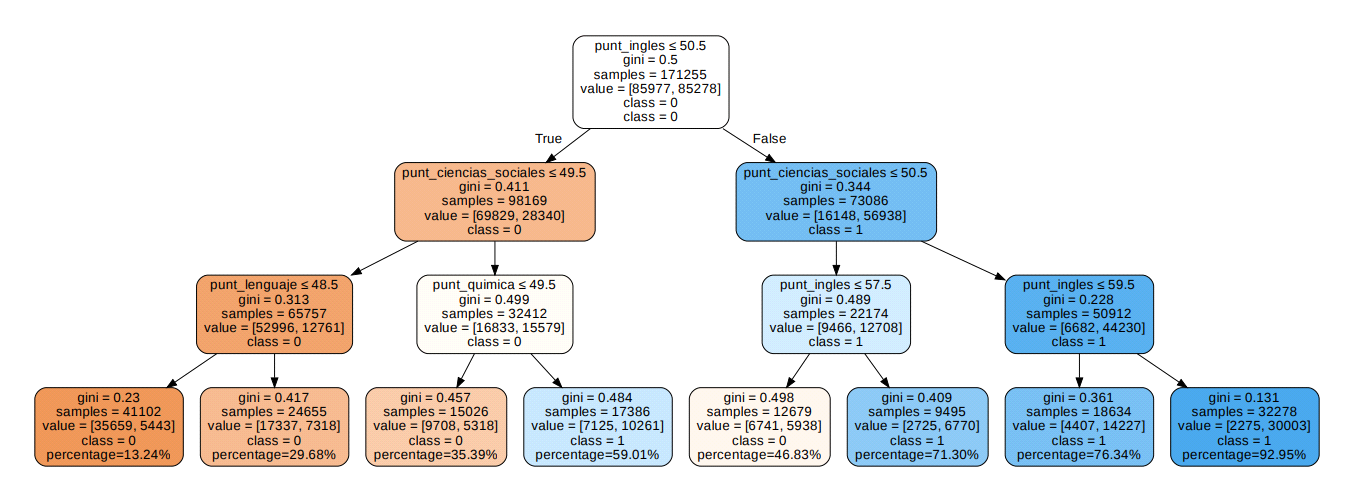
**Export and Visualize the Tree**

Sklearn also allows us to visualize the tree in a way that makes the algorithm and its implementation on the data set clearer. For that we will use the sublibrary trees and we will import the command export\_graphviz (later we will use also the graphviz library).

If we look at the documentation of this command, we will see that it needs to have as parameters: the tree that we are going to graph, the predictor variables in list form, the names of the classes (for example: the possible results that the variable to be predicted will give, in this case it would be 0 and 1, representing failure and success in the tests) and other aesthetic parameters on the graphic representation of the tree (such as the rounding at the corners of each node, the colouring, etc) which is a matter of personal taste.

Nevertheless, the export\_graphviz command will not yet show the tree drawing, but returns a DOT file with the data for the graphical representation, this is where the graphviz library comes in. This library has a function called Source, which will serve to render the data from the DOT file that the previous method gave us, so that we can see it now if in its graphic form

To finish we can save the tree graph with the render function, which creates a pdf with the drawing in the same path as the project.



On this graph we can appreciate that a series of variables appears on each node, here we will explain the meaning of each variable:

Firstly, we find the condition that evaluates the node to separate the data, in the first case it is punt\_ingles <= 50.5, this will be the test through which the student's data passes and then it will be directed to the next test in the left or right node. The leaves do not have this condition, since it is the end point of the tree where the success of the student in the test is evaluated

Then we find the gini impurity for each node, which indicates us the probability that a student is classified incorrectly, i.e. that he is in class 0 and has been assigned to class 1, or vice versa. How impure (i.e. how mixed up) is the data in the node. We see that, in some ways, as we go down the tree, this impurity decreases too, i.e. the gini impurity of the child node is less than that of the parent node, making the model more effective.

Samples: gives us the total number of students who meet a condition established in a node, in our case the condition is determined by the score obtained in one of the areas.

Value: is a list in which the total data of Samples is divided, this list has 2 values that represent the number of elements that belong to each class. The first number indicates the number of elements contained in the node that belong to class 0, i.e. "Not successful", and the second number indicates the number of elements that belong to class 1, which would be "Success".

Class: is the class to which the node belongs, that is, if the node tends to represent (or represents in itself) data belonging to the classes "Success" or "Non-success", by means of the values 1 and 0 respectively

**Success Probability**

The chances of a student success according to the scores in the test “Saber 11” and “Saber Pro” was evaluated considering the following:

According to the definition of Samples and Values, we made a rule of three considering that:

100% --------->Samples

  X   ---------> Values [1]

In such a way that the likelihood of each students would be X = (100\*Values [1])/Samples.

**Conclusions**

Despite that for this project we did not take in consideration aspects such as socioeconomic level, the study environment, and so on, these can influence the chances of success of students in the test “Saper Pro”, we wanted to make an evaluation in this regard through graphs, but we thought it was not the best choice, because we had to take into account the amount of data that was in the datasets with some qualities, for example, If we wanted to analyze in which area students are more likely to succeed on these tests, we would come up with a possible error since there are a greater number of students from urban areas than rural areas, so even though there are students from rural areas who succeed, this number could be exceeded by the number of people from urban areas who succeed, simply because they are a larger population.

From the tree graph it is inferred that students who scored higher than 50.5 in English and in social studies generally have a good probability of having won the test, as it would be a probability of at least 0.76, so these two factors are key to evaluating student success, especially the score in English. Evidence of this is the student 4289 from the TEST 0.csv file, since we can see how he obtained almost all the high tests, but in the English test he obtained a score of 20.0, which ended up qualifying him as "Not successful" on the test

We also see that if we read the leaf nodes from left to right, we will notice that the gini indexes are low at the sides of the ends, but if we go to the middle, we will see that these are considerably higher, which means that at these points the model may not be very effective. This may be due to the fact that the model we are using as an example has very few ramifications, in other words, very few criteria for evaluating such a large amount of information, which makes it end up falling short.

Thanks to this project, students can now take a strategy to determine which factors could lead to having success in the test “Saber Pro”, although, the best ides will always be to maximize our own skills in each of the subjects to be evaluated.