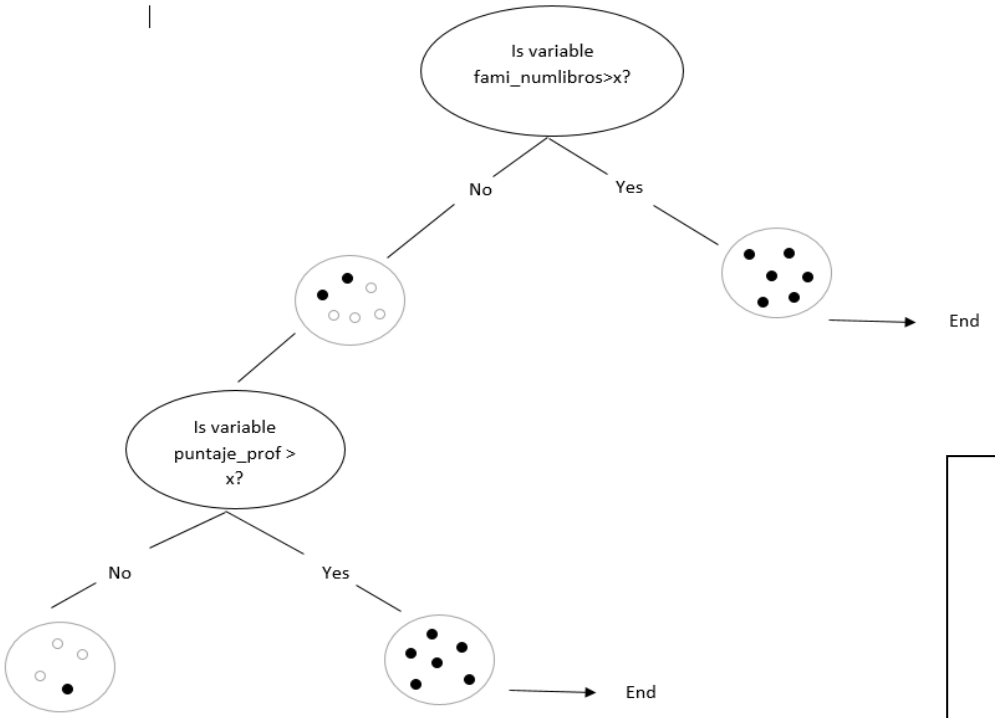
PREDICTION ALGORITHM FOR THE ACHIEVEMENT OF TEST "SABERPRO"

David Restrepo Ramírez

Juan Felipe López Gutiérrez

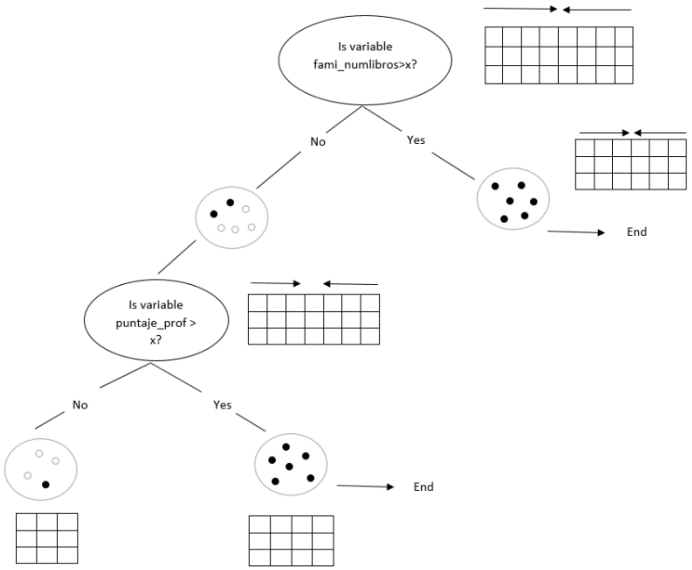
Medellin, May 21st, 2020.

Designed Data Structure



Picture 1: CART data structure evaluates and creates new child nodes based on the results it obtains.

Data Structure Operations



Picture 2: CART tree algorithm is going through the dataset. as it finds the lowest Gini impurity which divides the data, it creates nodes with conditions and takes the variable processed out

Operation	Complexity
leerArchivo(String data)	$O(n^2)$
seleccionarDataset()	O(n)
llenarMatriz(double [][] matriz)	$O(n^2)$
llenarImpureza(double [][] matriz)	$O(n^2)$
menores(double [][] matriz)	O(n)
add Child(Node child Node, int position)	O(1)
addNewNode(Node u, Object info, int i)	O(1)
numberOfNodesInTree(Node rootNode)	O(n)

Table 1: Table to report complexity analysis of the matrix to proceed with the next variables.

Design Criteria of the Data Structure

Decision trees are good for sorting, searching and storing big volumes of data with a low Big-O complexity.

CART (Classification and Regression Tree) algorithm is a good decision tree implementation which can be easily managed recursively. It provides a clear and simple comprehension of the structures that are being created which filter and classify the data.

Time and Memory

Table 2: Execution time of the operations

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

Operation	Execution time
leerArchivo(String data)	0.0ms
seleccionarDataset()	0.0ms
llenarMatriz(double [][] matriz)	0.0ms
llenarImpureza(double [][] matriz)	0.0ms
menores(double [][] matriz)	0.0ms
addChild(Node childNode, int position)	0.0ms
addNewNode(Node u, Object info, int i)	0.0ms
numberOfNodesInTree(Node rootNode)	0.0ms

Data set	Memory used
data_set_train.csv	11.0MB
data_set_test.csv	3.7MB