COFFEE RUST DETECTION FOR CATURRA VARIETY USING DECISION TREES

Santiago Cartagena Agudelo
Paulina Pérez Garcés

Medellín, October 29th 2019.

Decision Tree and Random Forest

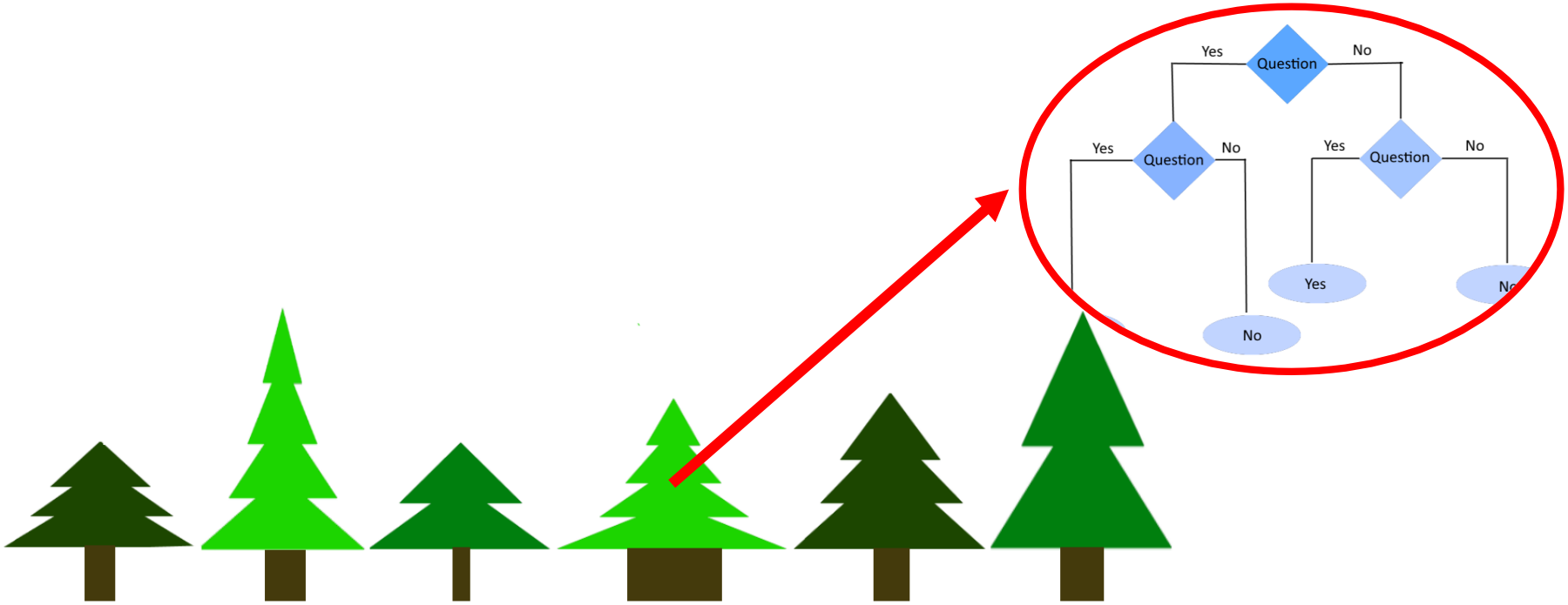

Figure 1: Random Forest filled with decision trees

Data Structure Operations

Taking into account the following notation, take a look of the chart about the complexity analysis in the algorithm:

- N represents the labels.
- M represents the rows.
- K represents the columns.
- L represents the values.

METHOD	COMPLEXITY
GINI INDEX	O(n*m) + O(n) = O(n*m)
INFORMATION GAIN	O (1) + O(n*m) = O (n*m)
BEST DATA SPLIT	O(k*l) + O (1) +O(m*n) + O(m) = O(k*l) + O(m*l) + O(m)
PARTITIONING	O(m) + O (1) = O(m)
BUILDING THE TREE	O(m) + O(k*l) + O (1) +O(m*n) + O(m) = O(k*l) + O(m*l) + O(m)

Table 1: Complexity analysis for the different methods implemented.

Gini Impurity Index

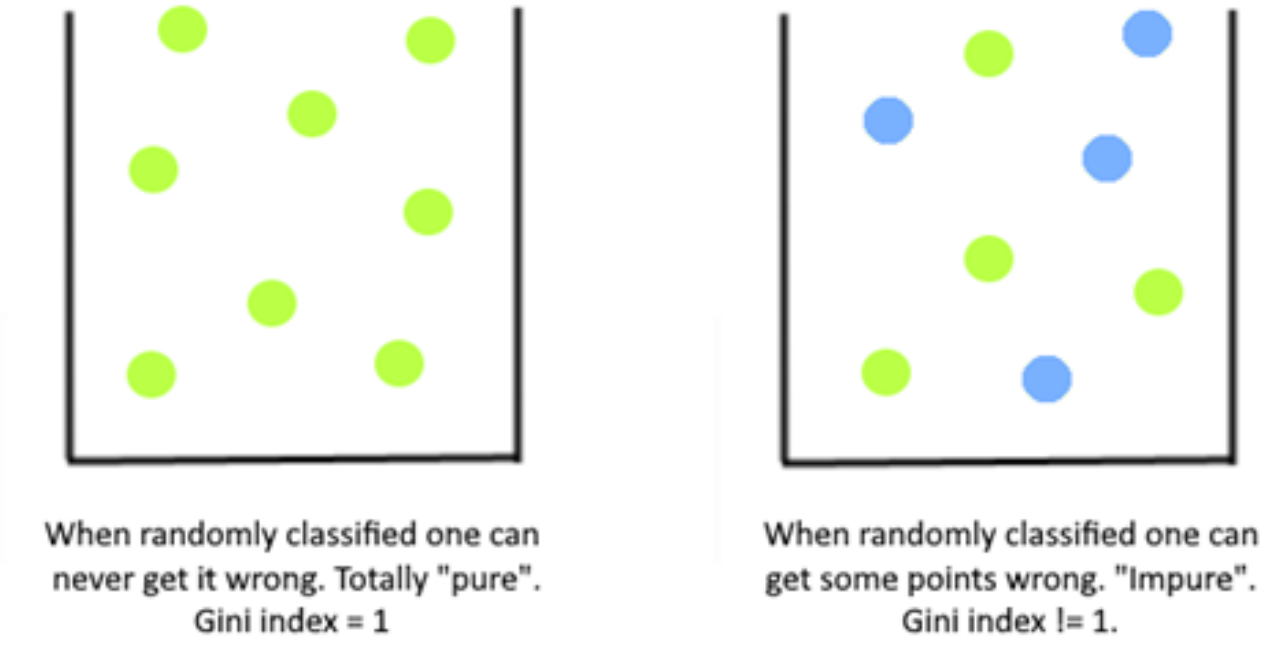

Figure 2: Gini calculation method explained.

Information Gain

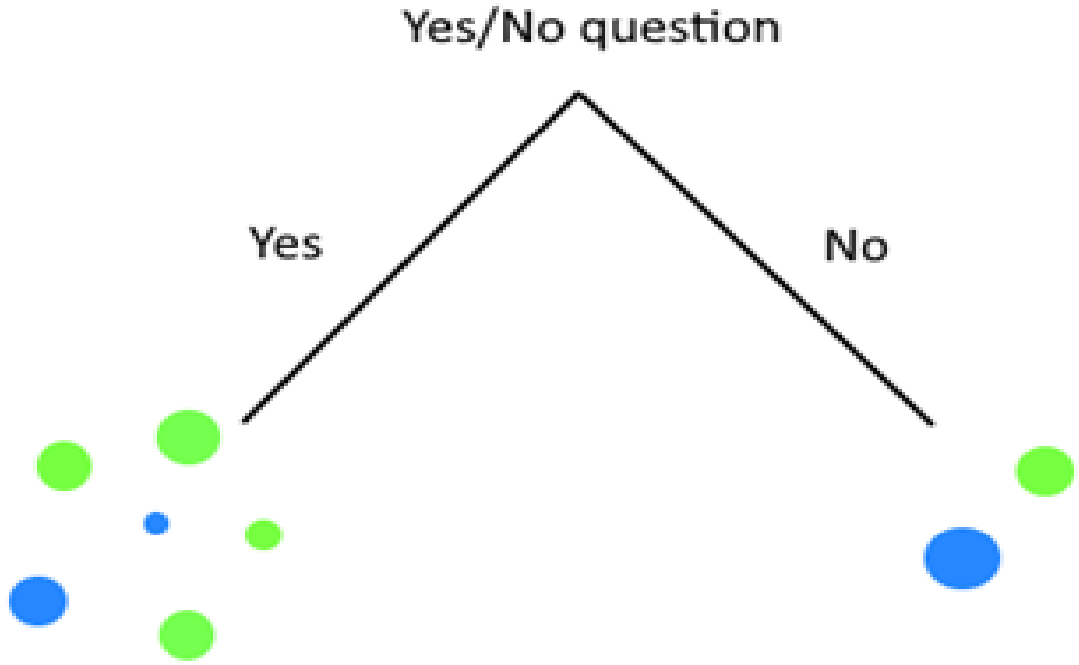

Figure 3: Information Gain explained.

Finding the Best Split

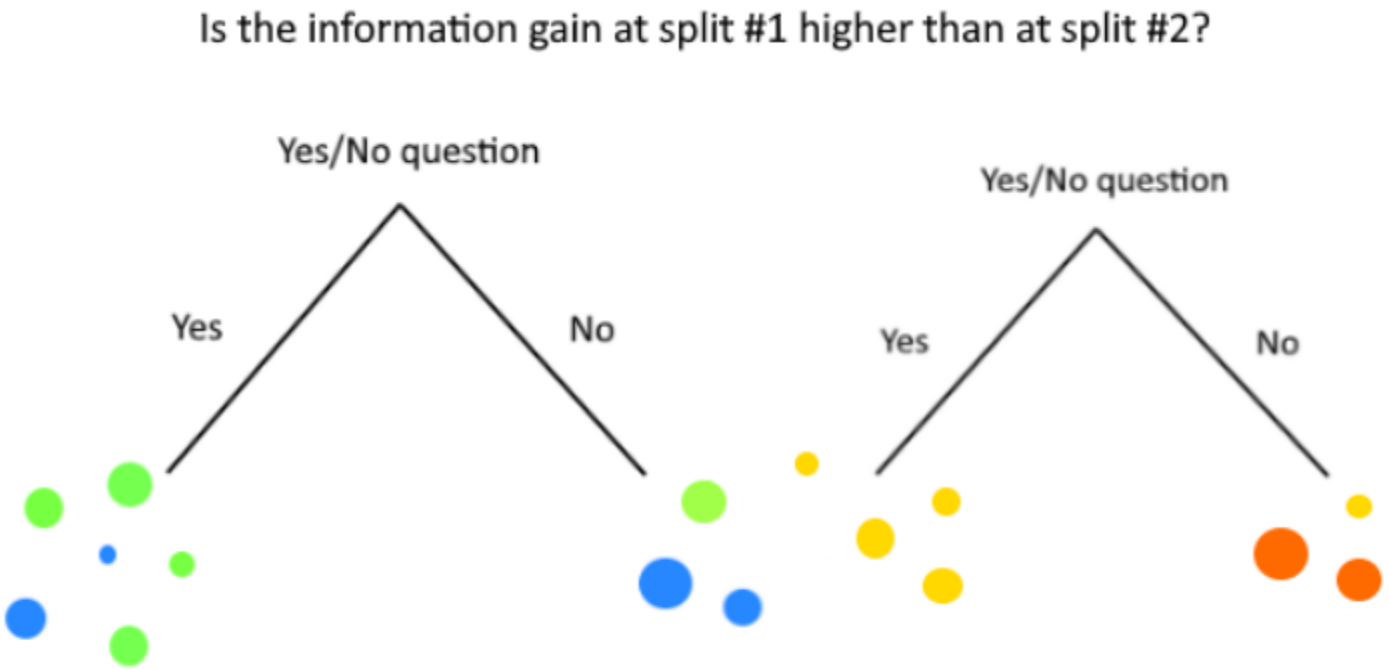

Figure 4: Representation of find_best_split

Partitions

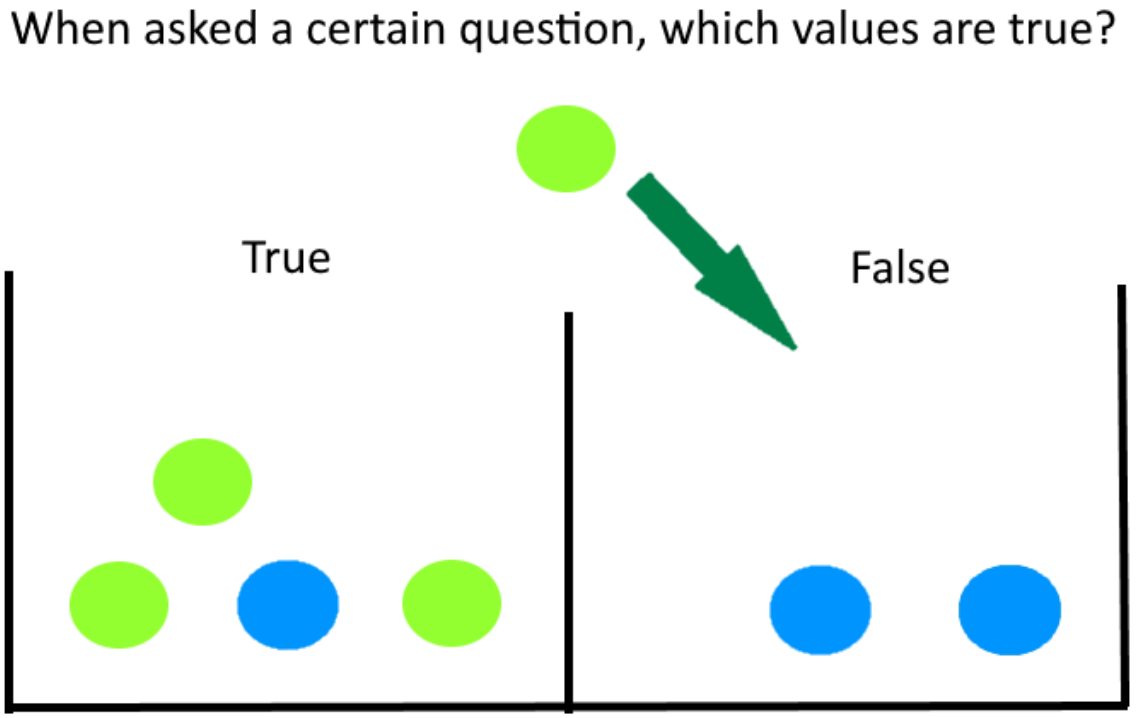

Figure 5: Graphic representation of partitioning the dataset.

Building the Tree

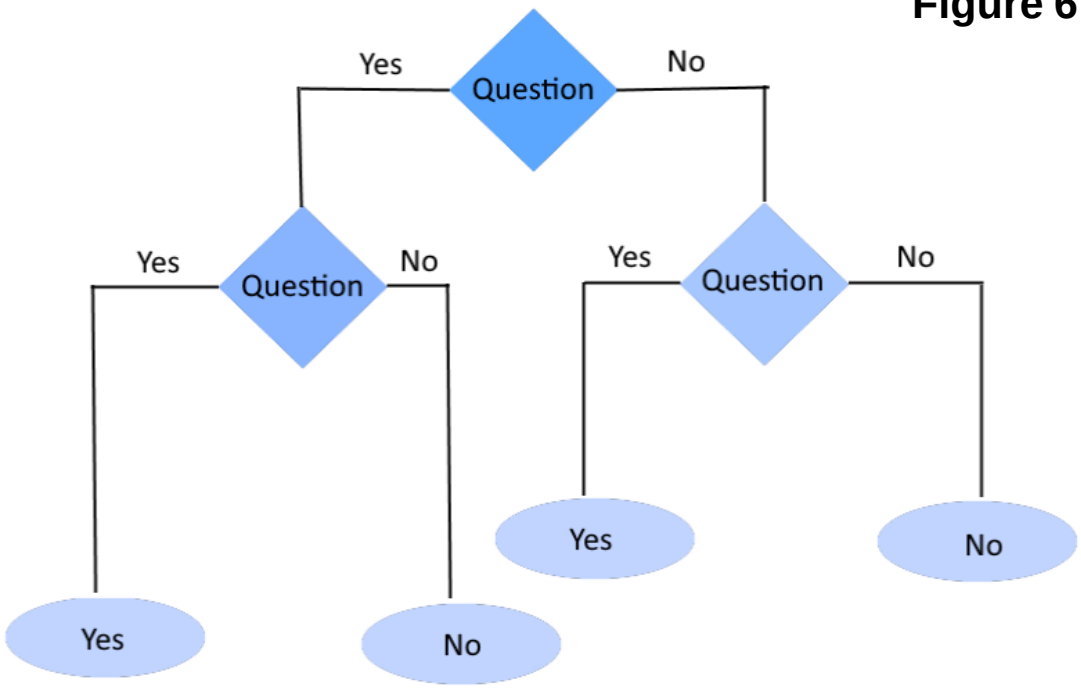

Figure 6: Decision Tree structure

Design Criteria of the Data Structure

- Decision Trees are considered “one of the best and mostly used supervised learning methods.” (Brid, R. S; 2018)
- Easy to understand, since the tree analogy is fairly simple.
- Fast and mostly accurate.
- Can work with classifications as well as regressions.
- One of the decision trees’ biggest flaws is overfitting, but with a random forest it stops being an overwhelming issue.
- Easy to adapt them to different circumstances.
- A single tree had an accuracy of almost 80%.

Time and Memory Consumption

METHOD	TRAINING DATA	TESTING DATA
GINI INDEX	0.2s	0.4s
INFORMATION GAIN	0.5s	0.7s
FIND BEST SPLIT	0.6s	0.8s
PARTITIONING	0.1s	0.1s
BUILDING THE TREE	1.5s	1.5s

TRAINING DATA	TESTING DATA
120 MB	160MB

Table 3 and 4: Time and memory consumption for the different methods.

Implementation

```
In [6]: import pandas as pd

trainingData = pd.read_csv('data_set_train.csv', delimiter = ',')
header = []

class Question:
    def __init__(self, column, value):
        self.column = column
        self.value = value

    def match(self, example):  #Can the chosen value 'pass the test'
        val = example[self.column]
        return val >= self.value

    def __repr__(self): #Prints the question into words and numbers
        condition = ">="
        return "Is %s %s %s?" % (
            header[self.column], condition, str(self.value))

class Leaf:
    def __init__(self, rows):
        totals = {}
        labels = set([row[-1] for row in rows])
        for label in labels:
            totals[label] = 0
        for row in rows:
            if row[-1]==label:
                totals[label]+=1
        self.predictions = totals

class Decision_Node:
    def __init__(self, question, true_branch, false_branch):
        self.question = question
        self.true_branch = true_branch
        self.false_branch = false_branch
```

```
            Is illuminance >= 948?
            --> True:
              Predict {'no': 1}
            --> False:
              Predict {'yes': 0, 'no': 1}
        --> False:
          Predict {'yes': 0, 'no': 1}
    --> False:
      Is env_temperature >= 19?
      --> True:
        Is illuminance >= 1852?
        --> True:
          Is illuminance >= 2185?
          --> True:
            Predict {'yes': 0, 'no': 1}
          --> False:
            Predict {'yes': 0, 'no': 1}
        --> False:
          Predict {'yes': 0, 'no': 1}
      --> False:
        Predict {'yes': 0, 'no': 1}
--> False:
  Is illuminance >= 2106?
  --> True:
    Predict {'yes': 0, 'no': 1}
  --> False:
    Predict {'yes': 0, 'no': 1}

79.0% de aciertos.
```

Figures 7 and 8: Part of the implemented code and results.

Report in arXiv

S. Cartagena-Agudelo, P. Pérez-Garcés, and M. Toro. Coffee Rust detection for Caturra variety using decision trees. ArXiv e-prints, Oct. 2018. Available at: https://arxiv.org/submit/2905373/view