[DS-07] Decision trees

Miguel-Angel Canela
Associate Professor, IESE Business School

What is a decision tree?

A **decision tree** is a collection of **decision nodes**, connected by branches, extending downward from the **root node**, until terminating in the **leaf nodes**. The usual graphical representation of a decision tree puts the root on top and the leaves at the bottom, as in Figure 1 (built with `Python` and exported to PDF format with `Graphviz`), which shows a tree with 8 decision nodes and 9 leaf nodes.

Decision trees can be used for classification and regression purposes. This note has a classification perspective. A decision tree creates a partition of the data set into a collection of subsets, one for each leaf. When the tree is used for prediction purposes, the predicted value is the same for all the instances in the same leaf.

Binary trees contain two branches per decision node. Any tree can be redesigned as a binary tree. Note that the tree can be binary without the predicted attribute being binary, that is, without the classification being binary (Figure 1 is one example). The discussion of this lecture is restricted to binary trees and binary classification. I also assume that the positive and negative class are coded as 1/0. In a decision tree, the same score is assigned to all the instances of the same leaf. The score is equal to the positive rate in that leaf.

¶ The decision trees discussed here are not the same as those used in decision analysis, in which there are two types of nodes, event nodes and decision nodes.

Decision tree algorithms

The top popular methods for developing decision trees are those based on the CART and C4.5 algorithms. **CART** (Classification And Regression Trees), is used in the R package `rpart` and in the Python package `scikit-learn`. For C4.5, you can look at Witten *et al* (2011).

In every decision node there is a **split**, defined by a binary variable. The splitting variable can be one of the columns of the data set or can be a dummy created from a numeric variable by means of a cutoff. The idea of the CART algorithm for selecting the splitting variable is simple: the difference between the actual class and the score is taken as a residual. The **optimal split** is the one for which the sum of squared residuals is minimum. This is equivalent to say that the correlation between the dummy that codes the classes and the splitting variable is maximum.

Pruning

Overfitting is a typical problem of predictive models. Overfitting occurs when a predictive model fits satisfactorily the data used to obtain it but it fails with data which have not been used in the obtention of the model. This typically happens with very complex models if the data set is not big enough.

Decision trees are usually pruned. **Pruning** reduces the size of the tree by removing branches that provide little predictive power. The dual goal of pruning is to reduce the complexity of the

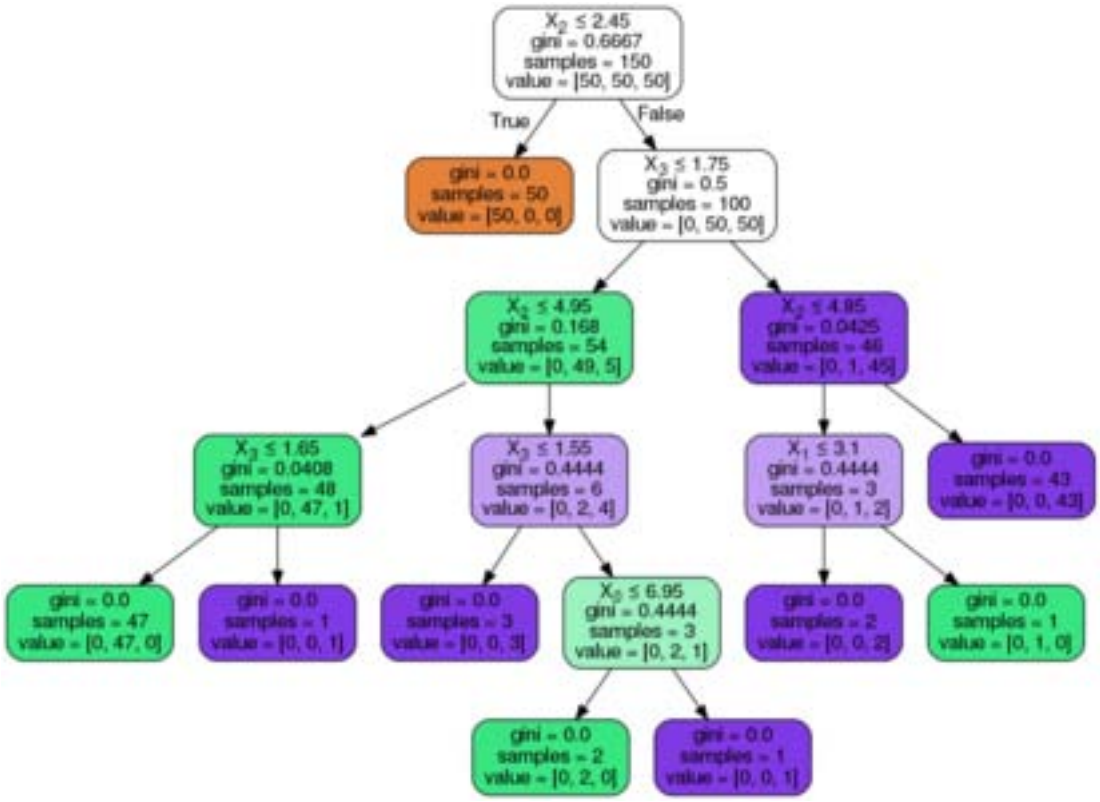

Figure 1. Decision tree

final classifier as well as to prevent overfitting. There are many techniques for tree pruning, which differ in the measurement used to optimize the performance. For instance, the CART algorithm uses a **cost-complexity model**. By tampering with the **complexity parameter**, we can get more or less pruning. The complexity parameter is the minimum increase in R-squared required for additional splits.

Nevertheless, in any implementation of these algorithms, there is a **default pruning rule**, which is applied when nothing is specified. If you are not happy with the default rule, you can use an option for specifying how aggressively you want to prune.

References

1. F Provost & T Fawcett (2013), *Data Science for Business — What You Need to Know About Data Mining and Data-Analytic Thinking*, O'Reilly.
2. IH Witten, E Frank & MA Hall (2011), *Data Mining — Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.