
Decision Trees
With
Python
By Giulia Osti
25.05.2023 / #WAIpractice

Hello There!

Position
PhD candidate @ UCD

Research Topic
AI & digital curation/heritage

Education
former archaeologist (!)

Linkedin
godigitalcorner

Twitter
@semanticnoodles

Email
giulia.osti@ucdconnect.ie

Today's Takeaways

- Understand the limitations and potential of decision trees
- Become familiar with key concepts and mechanisms
- Learn how to build and evaluate decision trees in Python using scikit-learn
- Know alternative tools for tree building and visualisation

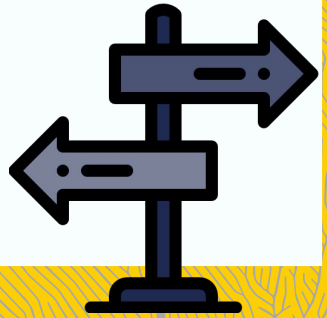
What are ‘Decision Trees’?

A group of ML algorithms often referred to as CART (classification and regression trees).

They allow to solve decision problems by categorising object according to their decision features.

Use Cases:

- Product analysis
- Botany (!)
- Exploratory analysis



Potentials & Limitations

Advantages:

- Highly visual & explainable
- Almost zero data processing
- Reliable (you can validate them)
- Can handle mixed data (categorical & ordinal)

Caveats:

- Sensitive to variance
- Careful reasoning on categorical variables
- One tree does not make a forest

Tools of the Trade

Python

Scikit-learn suite

Main portal

Documentation

Google Colab

Our Dataset

Alternatives To Python

WEKA

Fully interfaced ML free software
(no coding required) for ‘small’ datasets

Main portal | Documentation

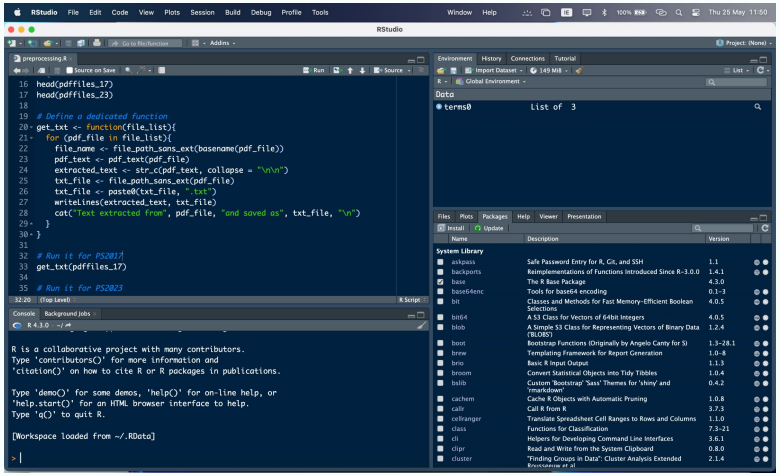
WEKA manual

R Studio

Integrated Development Environment
free, multiplatform (supports python!)

Main portal | User Guide

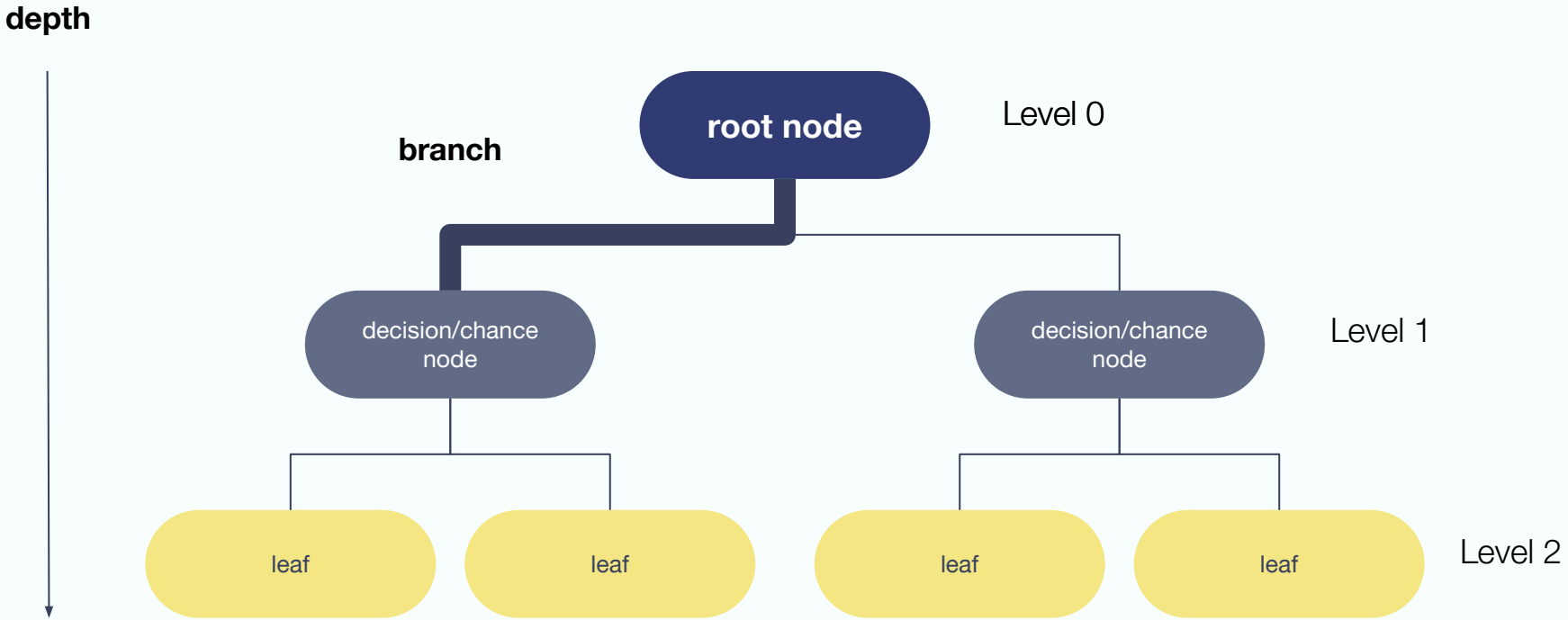
rpart.plot package on CRAN



A quick walk through some

Key Concepts

Anatomy of a Decision Tree



Essential Glossary

Root node

The topmost node in a decision tree. It represents the entire dataset and serves as the starting point for the tree's recursive partitioning process.

Chance node

A node in a decision tree that represents a situation where multiple outcomes are possible.

Leaf

A terminal node in a decision tree. It represents the final prediction or outcome.

Pruning

The process of reducing the complexity of a decision tree by removing nodes and branches.

Weighting

Strategy to handle imbalanced classes, or assigning different weights to samples or classes.

Depth

It represents the number of decisions or splits required to reach a prediction.

A Potential Use Case: Self-Tracking

About

A fictional dataset containing records of some behavioural habits related to productivity.

Task

Build a decision tree to get some insights about my focusing techniques of choice.

Spoiler: we are going to fail!

Focusing techniques

- Deep Work, almost hyperfocusing on a single task until it's done.
- Task Batching, completion of 1+ similar tasks jointly.
- Eat the Frog, run prioritised tasks depending on the difficulty.
- Zen to Done, just do it.

It’s Python time, so

Let’s Get Started

Conclusive remarks

- Decision trees work with ‘simple’ data. If you want to handle complexity you need to understand deeply how to weight and prune properly (doing some gardening).
- Make it simple for the machine to understand.
- Experiment with anything you are familiar with.
- Real world-like datasets bite, but do not give up!

Some extra resources

Here are some resources to get you started with:

- Data wrangling with Python
- Decision Trees in Python

- Data wrangling with R
- Decision trees in R

Datacamp and Udemy are actually offering nice courses on the topic - paid but for all the pockets!

Quick Recap

- Understand the limitations and potential of decision trees
- Become familiar with key concepts and mechanisms
- Learn how to build and evaluate decision trees in Python using scikit-learn
- Know alternative tools for tree building and visualisation


Thank You
For Joining!
Any Questions?