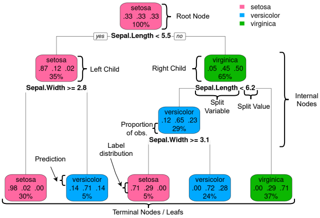
Introduction to Machine Learning

CART: In a nutshell



Learning goals

- Get a quick overview on CART models
- Understand basic concepts used to fit CART models
- You can use this slide deck either as a first overview or a final recap

WHAT IS A TREE?

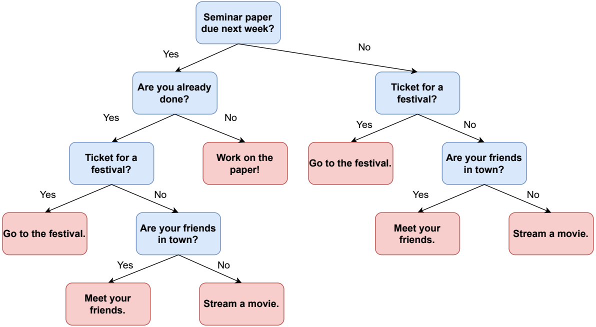
Basic idea: Divide the predictor space into sub-regions, and for each region, we learn from the training set by taking the average (mean), most frequent (mode), or middle value (median) of the response variable among the training examples belonging to that specific segment.

CART or trees are a class of models that can:

- model non-linear feature effect
- facilitate interactions of features
- be inherently interpreted

WHAT IS A TREE?

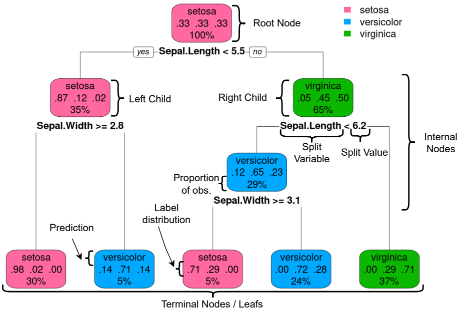
A decision tree is a set of hierarchical binary partitions.
An example from your daily life where your decision could be based on a decision tree may look like this:



WHAT IS A TREE?

- Instead of life choices you can partition a target y through feature space $\mathcal{X}$.
- For example, we can predict the $\mathrm{Species}$ of the flowers described in the iris data set using the features $\mathrm{Sepal.Width}$ and $\mathrm{Sepal.Length}$.

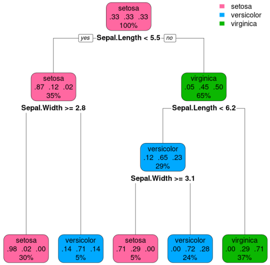
A CART can be fully (visually) decribed by this (annotated) plot:

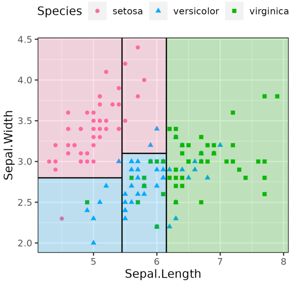


CART AS A PREDICTOR

Instead of the visual description, we can also describe trees through their division of the feature space $\mathcal{X}$ into **rectangular regions**, Q_m:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(\mathbf{x} \in Q_m),$$

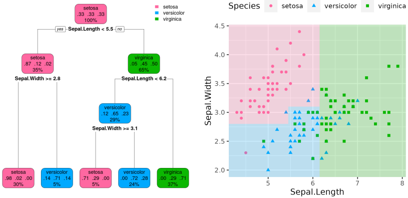




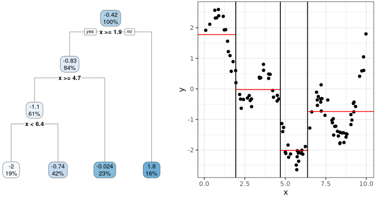
TASKS FOR CART

- **C**lassification **A**nd **R**egression **T**rees can have categorical and numeric targets
- However, CART have also been applied to more exotic tasks like survival analysis, e.g. ▸ Davis and Anderson, 1989
- In both cases, the leafs (the ultimate nodes) define the predictions

Categorical target:

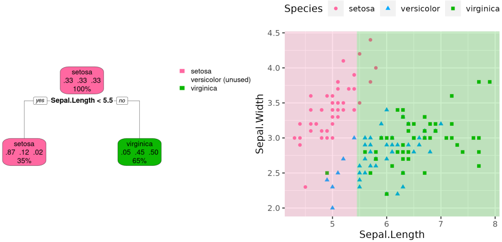


Numeric target:



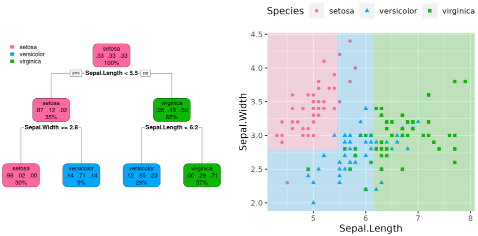
HOW TO FIT A CART

- A *recursive* greedy search in the feature space optimizes CARTs
- In each iteration the (currently) best binary split out of all possible splits is selected
- The best split is determined via empirical risk minimization
- This recursive fitting looks like this, starting with iteration 1:

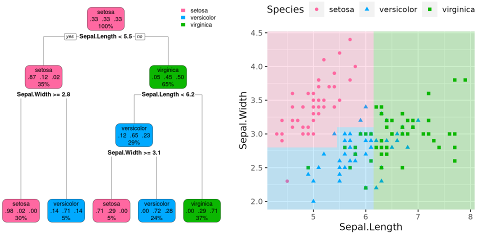


HOW TO FIT A CART

Iteration 2:

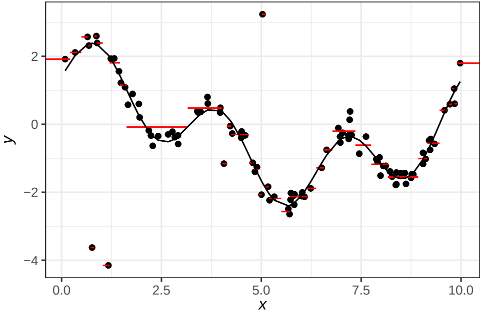


Iteration 3:



HOW TO FIT A CART

- This procedure would / could run until each observation has its own leaf.
- This is, however, not a good idea as the tree will not generalize well and overfit:



- Thus, we need techniques to keep the tree short or shallow.
- In fact, we used some of these techniques for the fit of the tree.

HOW TO FIT A CART

- There are two options for this:
  - Stop the fitting at some point
  - Fit a deep tree and shorten ("prune") it afterwards
- The fitting is typically stopped either by specifying a maximum depth of the tree or minimum number of observations per leaf.
- Another option is to specify a minimum decrease of the empirical risk that a split has to exceed.
- A common library for fitting trees includes the following options:

```
rpart.control(minsplit = 20, minbucket = round(minsplit/3), cp = 0.01,
              maxcompete = 4, maxsurrogate = 5, usesurrogate = 2, xval = 10,
              surrogatestyle = 0, maxdepth = 30, ...)
```

TREES IN PRACTICE

- Trees are an attractive learner:
  - (Usually) cheap to compute
  - Interpretable
  - Can capture complex feature effects
- However, CART has high variance, hence not the best predictor

$\Rightarrow$ In practice, trees are mostly used as base learners for ensemble learners like Random Forests.