Machine Learning

Decision trees: regression

Sebastian Engelke

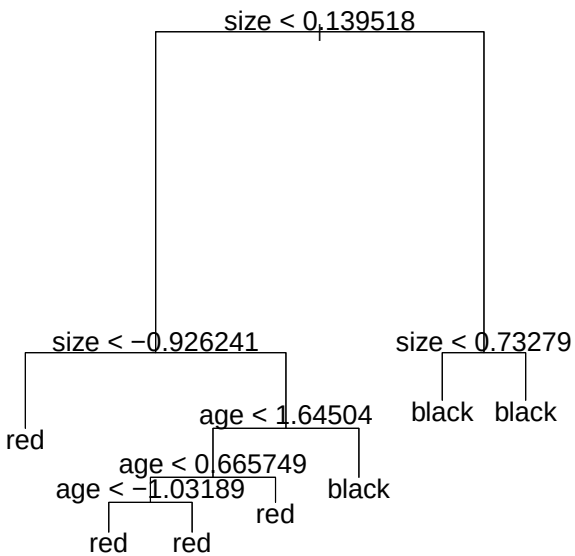
Master in Business Analytics

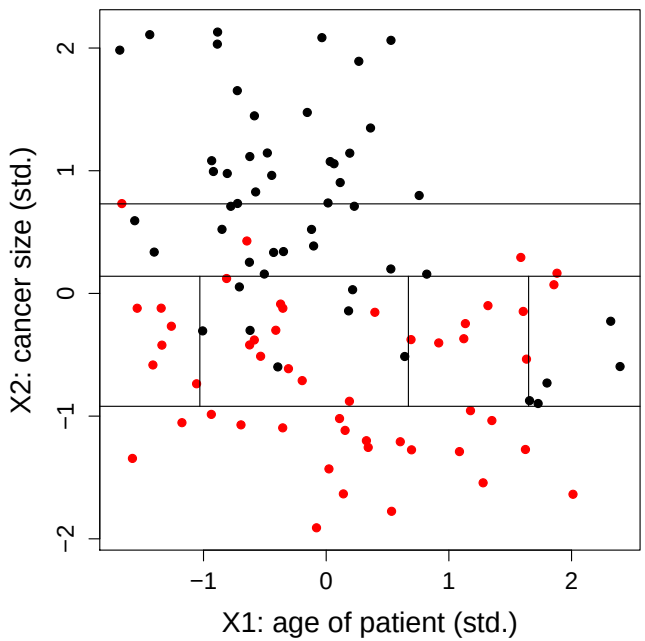
UNIVERSITÉ
DE GENÈVE

Tree-based methods

- `Tree-based methods` stratify/segment the feature space recursively into simple regions, namely rectangles $R_1, \ldots, R_J$.
- They can be applied to both `classification` and `regression`.
- To make predictions for a given observation, we use the `mean` or the `most common class` of the training observations in the region to which it belongs.

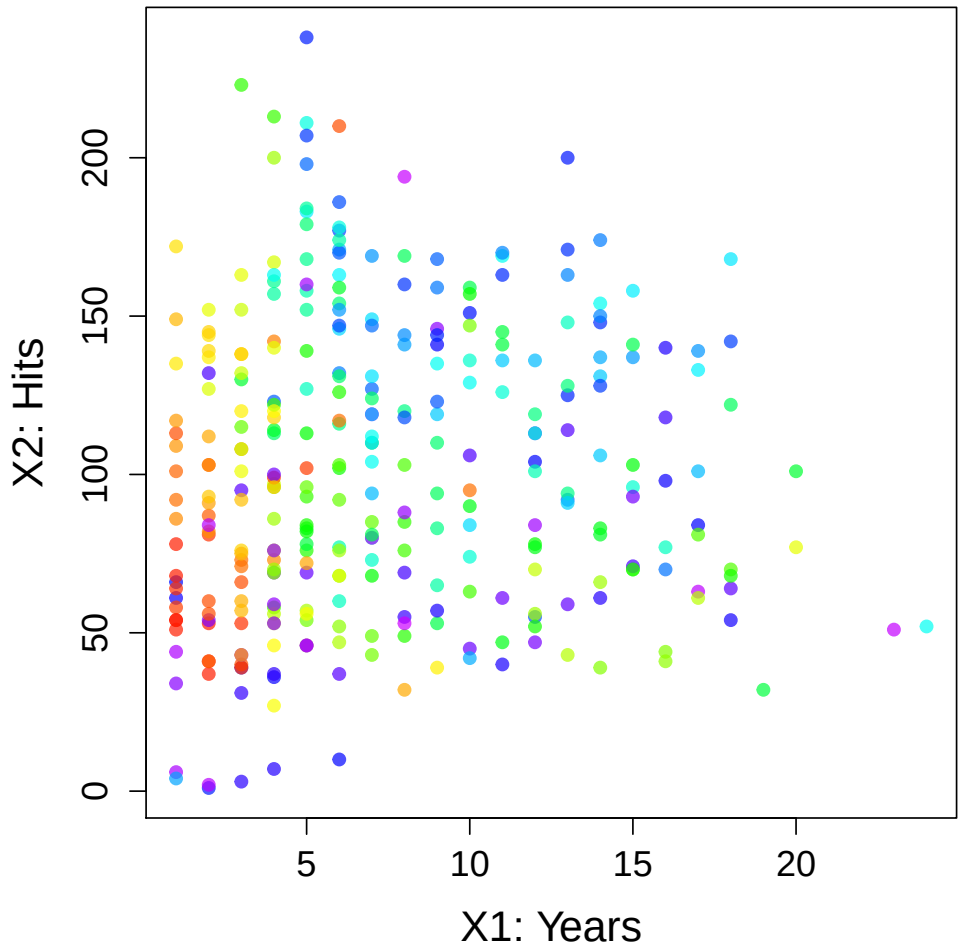
1. **Decision trees**: Single tree with good `interpretability`.
2. **Bagging/Boosting**: Combining mulitple trees to `improve predition`.
3. **Random Forest**: `Decorrelation` of bagged trees.





Regression example: Hitters data set

The `Hitters` data set from the `ISLR` package contains the annual `Salary` of 263 baseball players and $p = 19$ predictors such as numbers `Years` played in major leagues and the number of `Hits`. `Salary` color-code: low (yellow-red) to high (blue, green).



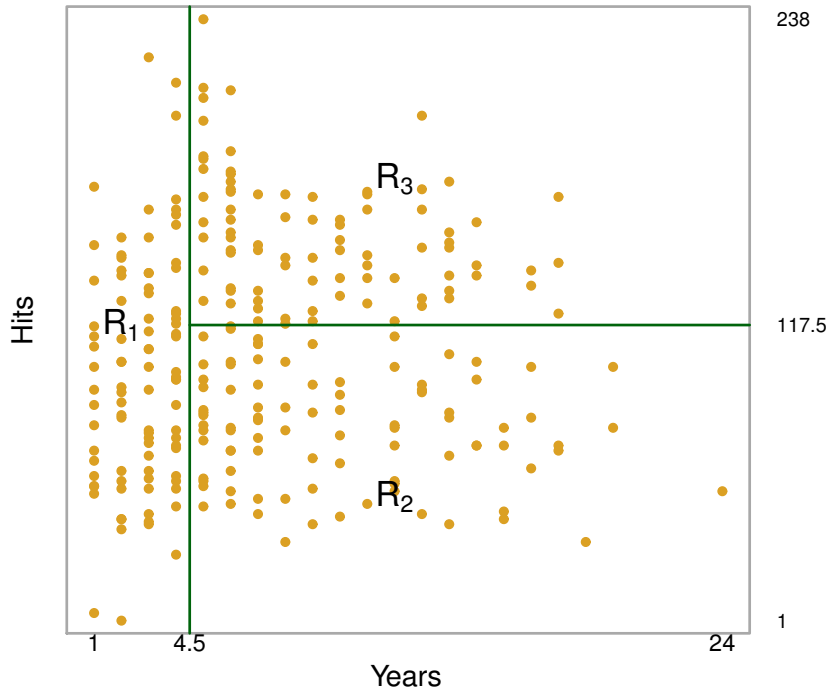
Regression trees: terminology

Regression tree for predicting the log `Salary` of a baseball player using only the variables `Years` and `Hits` as predictors.

- Each split is called `node`.
- A terminal node is called a `leaf`.
- The interior nodes lead to `branches`.
- The label $(X_j < t)$ indicates the `left-hand branch` from that split and the right-hand split corresponds to $(X_j \geq t)$.
- The number in each leaf is the `mean` of the responses for the observations that fall there.
- Note that an advantage of trees is the `interpretability`: the structure corresponds to the way people might think/act in a `decision problems`.

Regression trees: segmentation of feature space

- The tree segments the feature space into three regions corresponding to the `terminal nodes`.
- $R_1 = \{X \mid \texttt{Years} < 4.5\}$
- $R_2 = \{X \mid \texttt{Years} \geq 4.5, \texttt{Hits} < 117.5\}$
- $R_3 = \{X \mid \texttt{Years} \geq 4.5, \texttt{Hits} \geq 117.5\}$



Interpretation:

- `Years` is the most important factor in determining `Salary`, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of `Hits` seems to play little role in his `Salary`.
- But among players who have been in the major leagues for five or more years, the number of `Hits` does affect `Salary`.

How to grow a regression tree?

Given a training data set $(x_1, y_1), \ldots, (x_n, y_n)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, how do we grow a regression tree?

- We want to divide the predictor space of $(X_1, \ldots, X_p)$ into J distinct non-overlapping regions $R_1, \ldots, R_J$.
- The regions could have any shape, but for trees we choose high-dimensional rectangles or boxes for the R_j.
- For every obervation that falls in to region R_j we predict the same value

$$\widehat{y}_{R_j} = \frac{1}{n_j} \sum_{i: x_i \in R_j} y_i,$$

namely the mean of the responses of all training observations in R_j, where n_j denotes the number of training training observations in R_j.

- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS given by

$$\sum_{j=1}^{J} \sum_{i: x_i \in R_j} (y_i - \widehat{y}_{R_j})^2.$$

Tree growing: greedy algorithm

- Unfortunately it is computationally impossible to search over all possible partitions of the feature space in J boxes.
- Instead, one uses a top-down, greedy algorithm that is also known as recursive binary splitting.
- It is top-down because it begins at the top of the tree and then successively splits the predictor space. Each split is indicated via two new branches further down on the tree.
- It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Tree growing: greedy algorithm

The greedy algorithm:

(1) For any predictor j and cutpoint t define the half-planes

$$R_1(j,t) = \{X \mid X_j < t\} \text{ and } R_2(j,t) = \{X \mid X_j \geq t\}.$$

(2) We seek the values of j^* and t^* that minimize the RSS

$$\sum_{i:x_i \in R_1(j,t)} (y_i - \widehat{y}_{R_1(j,t)})^2 + \sum_{i:x_i \in R_2(j,t)} (y_i - \widehat{y}_{R_2(j,t)})^2.$$

We split the predictor space in the two regions $R_1(j^*, t^*)$ and $R_2(j^*, t^*)$.

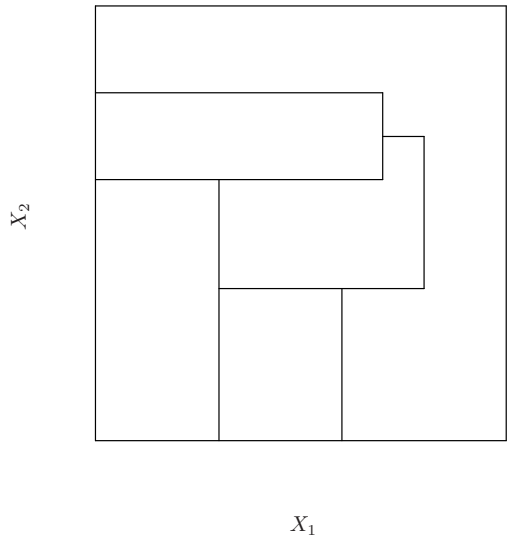
(3) We continue the process by looking for the best predictor and cutpoint, but now, instead of splitting the entire predictor space, we split one of the two previously identified regions.

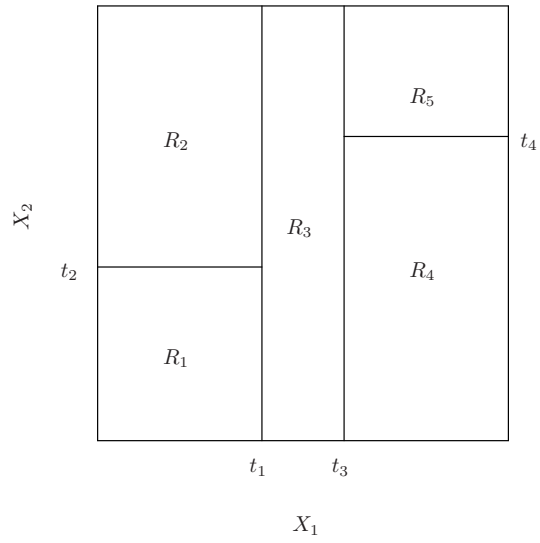
(4) We continue the process until a stopping criterion is reached, e.g., until no region contains more than five observations.

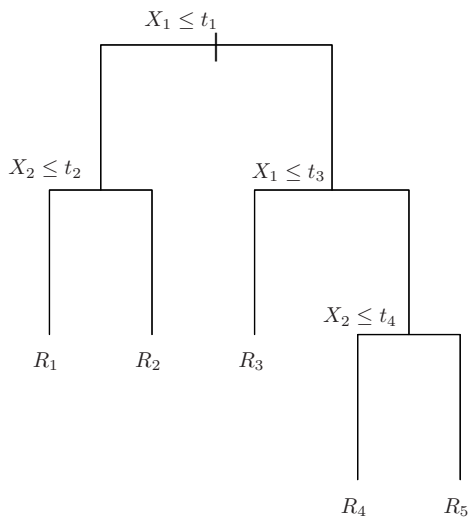
(5) Once the regions $R_1, \ldots, R_J$ are created, we predict the response for a given test observation using the mean $\widehat{y}_{R_j}$ of the training observations in the region R_j to which that test observation belongs, that is, a tree corresponds to a piece-wise constant predictive function

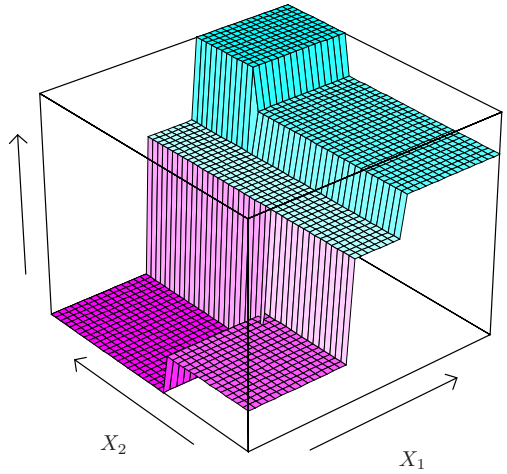
$$\widehat{f}(x) = \sum_{j=1}^{J} \widehat{y}_{R_j} \mathbf{1}\{x \in R_j\}.$$

Regression tree: two-dimensional feature space









Tree pruning

Tree properties

- Intuitively, the `complexity` of the predictive model corresponding to a tree T is the number of terminal nodes, denoted by $|T|$, e.g., the number J of boxes $R_1, \ldots, R_J$.

Bias-variance trade off

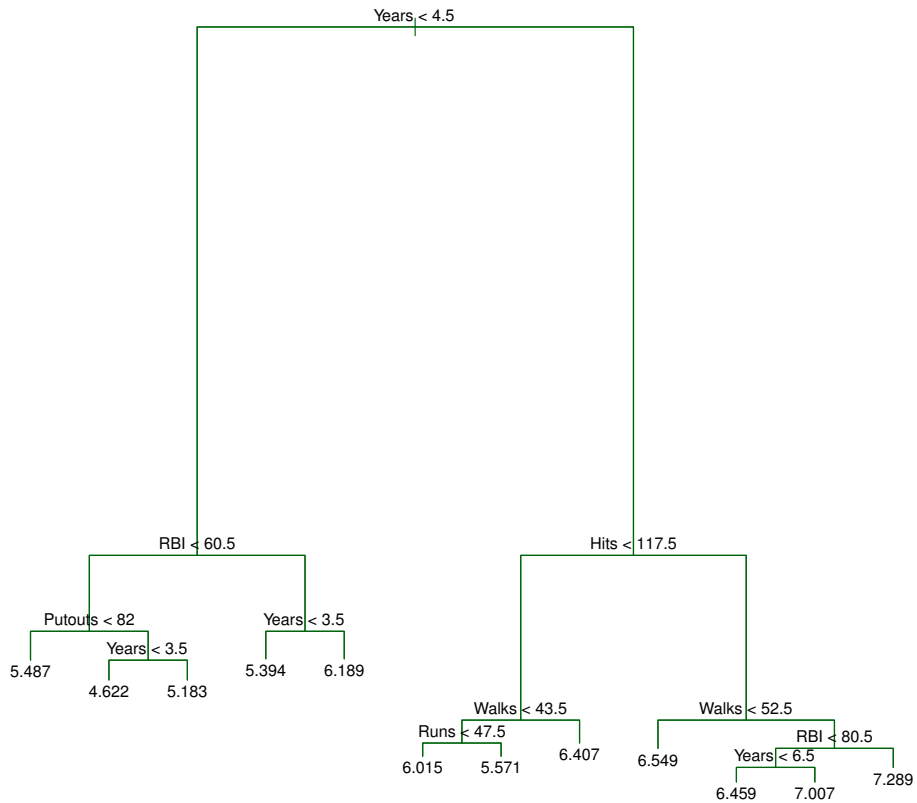
- A very complex, large tree may produce good prediction on the training set, but it is likely to `overfit` the data leading to a poor test error.
- A `less complex, smaller tree` with fewer splits/regions might lead to lower variance but to higher bias, and *vice versa* for more complex trees.
- A way of avoiding overfitting is to first grow a large tree T_0 and then `prune it back` in order to obtain a subtree.

Cost-complexity pruning

We can consider a sequence of trees indexed by a tuning parameter $\alpha \geq 0$. For each value of α there corresponds a subtree $T_\alpha \subset T_0$ that minimizes

$$\sum_{j=1}^{|T_\alpha|} \sum_{i:x_i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T_\alpha|.$$

- This is called `cost-complexity pruning` or `weakest-link pruning`.
- The number $|T_\alpha|$ of terminal nodes of T_α indicates the complexity of the tree.
- The `regularization parameter` α controls the tree complexity.
- It has a similar interpretation as the parameter λ in lasso or ridge regression.
- The optimal value is chosen by `cross-validation`.



Regression example: Hitters data set

Tree growing and pruning illustrated on the `Hitters` data set. The data is split into 132 training observations and 131 test observations. A large regression tree is built on the 132 training data and α `is varied` to obtain pruned subtrees with different numbers of terminal nodes. `Six-fold CV` is performed on the 132 training data to estimate the MSE of the trees. The test error is computed on the 131 test observations that are `not used` for model fitting or selection.