KAZAKH-BRITISH TECHNICAL UNIVERSITY

Introduction to Machine Learning
Week 12

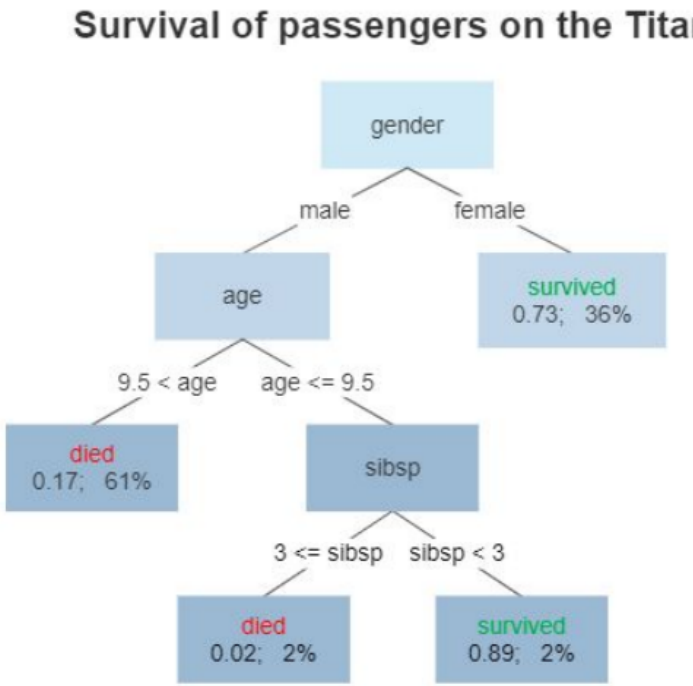
Olivier JAYLET

School of Information Technology and Engineering

Decision Tree

What is a decision tree?

A binary decision tree is an iterative algorithm that splits the training sample into two subsamples at each iteration.



Purpose of decision trees

Applied to data, *decision trees*, allow us:

- to *partition* the training observations into several *regions*.
- *estimate* a simple model in each region.
- fit a *regression* or *classification* model.
- the basis of the *random forest* algorithm.

Popular method: Classification and Regression Trees (CART), Breiman, Friedman, Olshen and Stone (1984).

Advantages of Decision Tree

Interpretation of the model, simple structure:

- The *root* of the tree (contains all observations),
- The *nodes* (from the binary divisions),
- The *leaves* (or terminal nodes).

Growing the Tree

We assume that we have a training sample of n observations described by p variables $x_1, x_2, \ldots, x_p$ and a *quantitative* variable to explain y.
There are two steps in growing a tree:

- Separate the observations into J *distinct regions* $R_1, R_2, \ldots, R_J$.
- For each region R_j, we compute the average of the values of y of observations belonging to this region:

$$\hat{y}_{R_j} = \frac{1}{\#R_j} \sum_{i \in R_j} y^{(i)} \tag{1}$$

Binary splitting

- How to construct the regions ?
- Method: recursive binary splitting.
- Idea: find the regions $R_1, \ldots, R_J$ that minimize the sum of squares residuals (SRR), defined by:

$$SSR = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y^{(i)} - \hat{y}_{R_j} \right)^2$$

Note: SSR corresponds to the training error.

Binary splitting

- For each variable x_j
- Choose a threshold value $s \in \mathcal{S}(x_j)$
- Split observations in two regions R_l and R_r such that:

$$R_l(j,s) = \{x | x_j < s\} \quad \text{and} \quad R_r(j,s) = \{x | x_j \geq s\} \tag{2}$$

Finally, keep the couple (j,s) that minimizes :

$$\sum_{i:x^{(i)} \in R_l(j,s)} \left(y^{(i)} - \hat{y}_{R_l}\right)^2 + \sum_{i:x^{(i)} \in R_r(j,s)} \left(y^{(i)} - \hat{y}_{R_r}\right)^2 \tag{3}$$

Recursive algorithm

- This process is repeated for each new region obtained.
- Until a certain stop criterion is reached (e.g., minimum number of observations in a region).
- The tree thus obtained is called the maximal tree.

Optimizing (pruning) the maximal tree

- The maximum tree obtained may be too complex (too deep).
- A smaller tree, containing less binary divisions (regions) allows a better interpretation and reduces the variance.
- Pruning consists in building a very deep tree T_0 and reducing its size in order to obtain a subtree.

Tree Pruning

- Also called Cost-complexity pruning.
- Idea: define a measure that takes into account the learning error and the complexity (depth) of the tree.
- Goal: For a given value of α (complexity parameter), find a subtree $T \subset T_0$ that minimizes

$$\sum_{j=1}^{|T|} \sum_{i:x^{(i)} \in R_j} \left(y^{(i)} - \hat{y}_{R_j} \right)^2 + \alpha |T| \tag{4}$$

where T is the number of terminal nodes and α is a tuning parameter that penalizes large trees.

Example :

Data: *Hitters* dataset (Major League Baseball, season 1986–1987): 322 observations and 20 variables.

Goal: predict players' salary in function of the number of years played in MLB and the number of hits in the previous season.

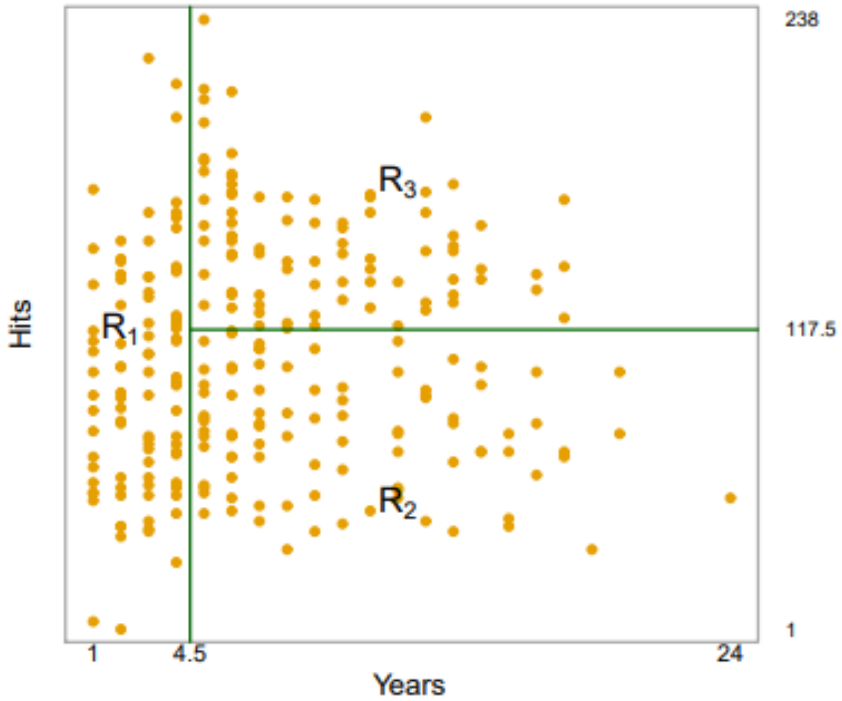
Example :

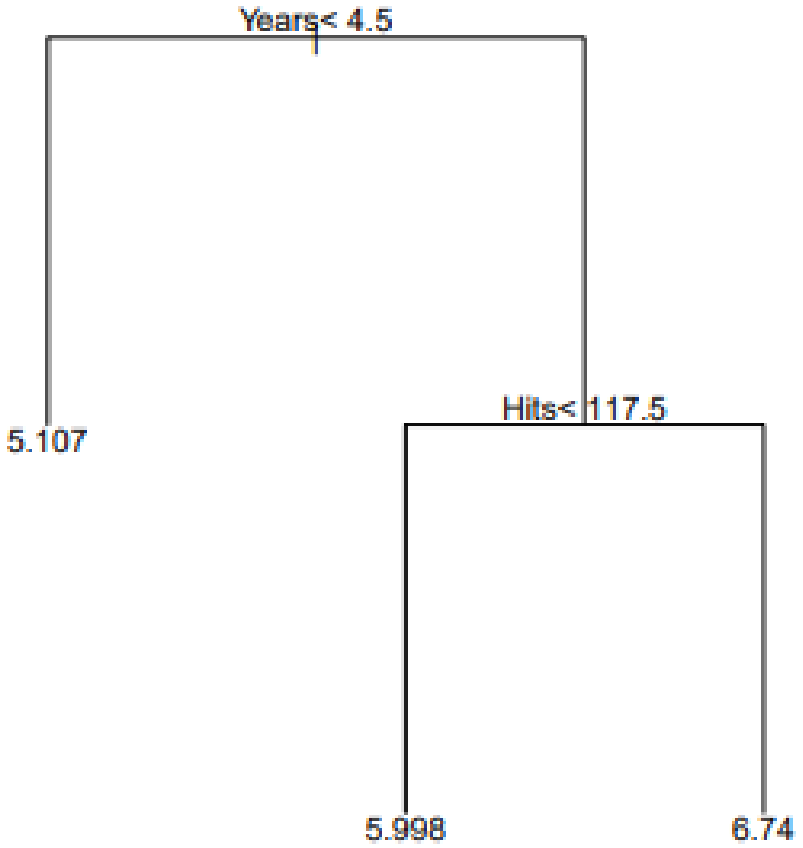
Name	Years	Hits	Salary
Alan Ashby	14	81	475.0
Alvin Davis	3	130	480.0
Andre Dawson	11	141	500.0
Andres Galarraga	2	87	91.5
Alfredo Griffin	11	169	750.0

With:

- x_1: *Years* in Major League
- x_2: *Hits* in 1986
- y: *Salary* (log-transformed)

Results


Hits
238
117.5
1
1
4.5
24
Years
R_1
R_2
R_3


Years< 4.5
5.107
Hits< 117.5
5.998
6.74

Implementation in Scikit-Learn

- We use the function `DecisionTreeRegressor`.
- Many hyperparameters (with default values!):
  - `maxdepth`: maximal depth of the tree
  - `min_samples_split`: minimum number of observations to split a node
  - `min_samples_leaf`: minimum number of observations to have a leaf
- Tuning the hyper-parameters with cross-validation and grid search.

Difference with Regression Trees

- A classification tree is similar to a regression tree in terms of structure.
- It is used to predict a qualitative variable: $y \in \{1, \ldots, K\}, K \in \mathbb{N}$.
- We assign to each region R_j the most occurring class among the observations in R_j.

Problem: we do not use the RSS in classification, we use the classification error rate.

Classification Error

Classification error rate in a region R_j is the proportion of training observations in R_j that do not belong to the most occurring class:

$$\epsilon_j = 1 - \max_k(\hat{p}_{jk}) \tag{5}$$

where $\hat{p}_{jk}$ is the proportion of observations in R_j that belong to class k.

Note: in practice, two other metrics may be used, the Gini index or the cross entropy.

Gini index

The Gini index in a region R_j is defined by:

$$G_j = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk}) \tag{6}$$

This is a measure of total variance (or impurity) across the K classes.
Interpretation: a small value indicates that the node (or region) contains a predominant class (we say that the node is pure).

Cross entropy

An alternative to the Gini index :

$$D = -\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk} \tag{7}$$

It also measures the impurity of a node: low values indicate that the node is pure.

Note: The two indices are very similar, but cross-entropy is more sensitive to small changes in probabilities than Gini & penalizes impurity more harshly.

Variable importance

Let t be the node of a tree, let t_l and t_r be its child nodes.
p_l and p_r are the proportions of observations sent to each child node.
We will use the impurity criterion of node t :

- $R(t)$ in the case of regression (e.g. SSR)
- $I(t)$ in the case of classification (e.g. error rate, Gini index, cross-entropy)

Variable importance

The importance of a variable x_j is defined by:

$$VI(x_j) = \begin{cases} \sum_{t \in T} \left(R(t) - R(t_l) - R(t_r) \right) & \text{in regression} \\ \sum_{t \in T} \left(I(t) - p_l I(t_l) - p_r I(t_r) \right) & \text{in classification} \end{cases}$$

where :

- $R(t)$ & $I(t)$ measure the impurity of the parent node t.
- $R(t_l)$ & $R(t_r)$ measure impurity of the child nodes after splitting at t.
- $p_l I(t_l)$ & $p_r I(t_r)$ measure weighted impurities of the left and right child nodes.

Note: At each node t , the binary split is based on the same variable x_j , it is the competitive division.

Pro

- Easy to implement with Scikit-Learn
- Graphical decision making tool
- Non-parametric interpretable model (*"white box"*)
- Regression and classification tasks.

Cons

- Choice of hyperparameters.
- Requires extensive model optimization (pruning, grid seach cross-validation)
- Trees are unstable.