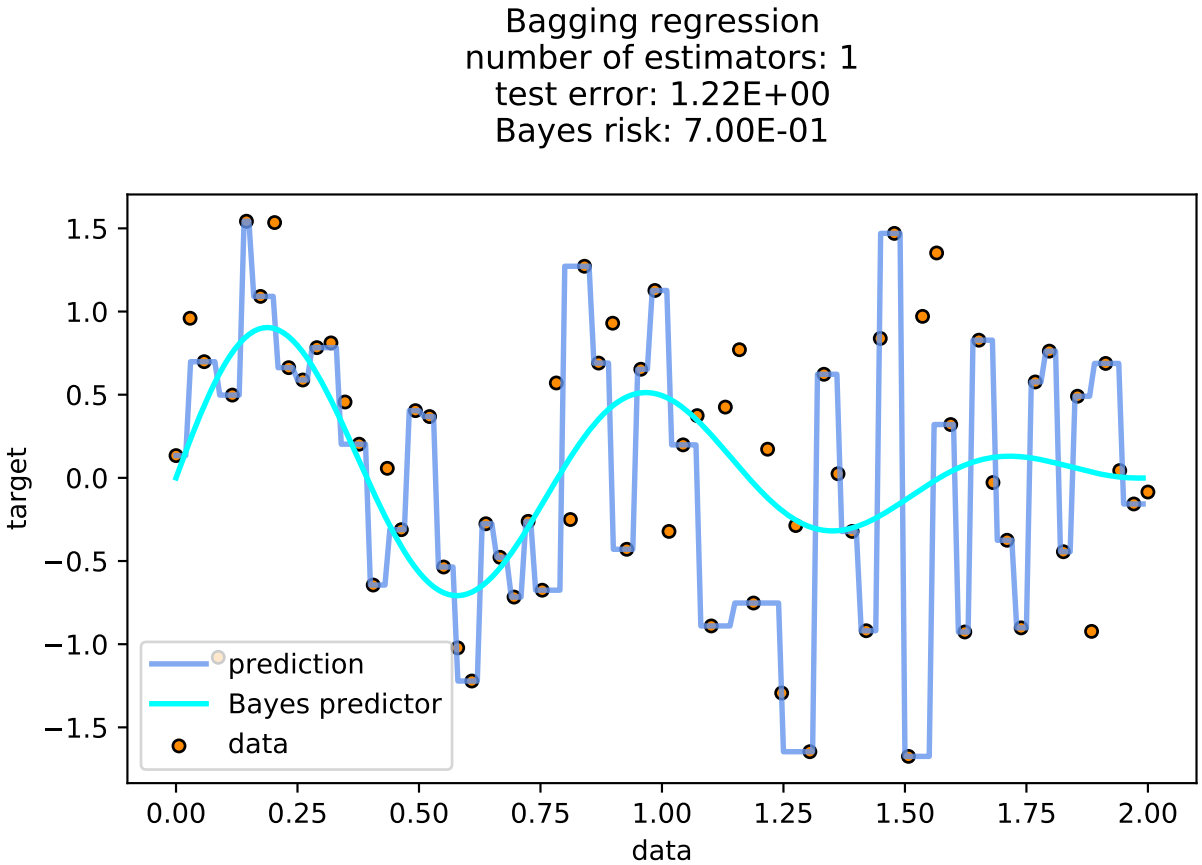
Machine learning I, supervised learning: classification and regression trees



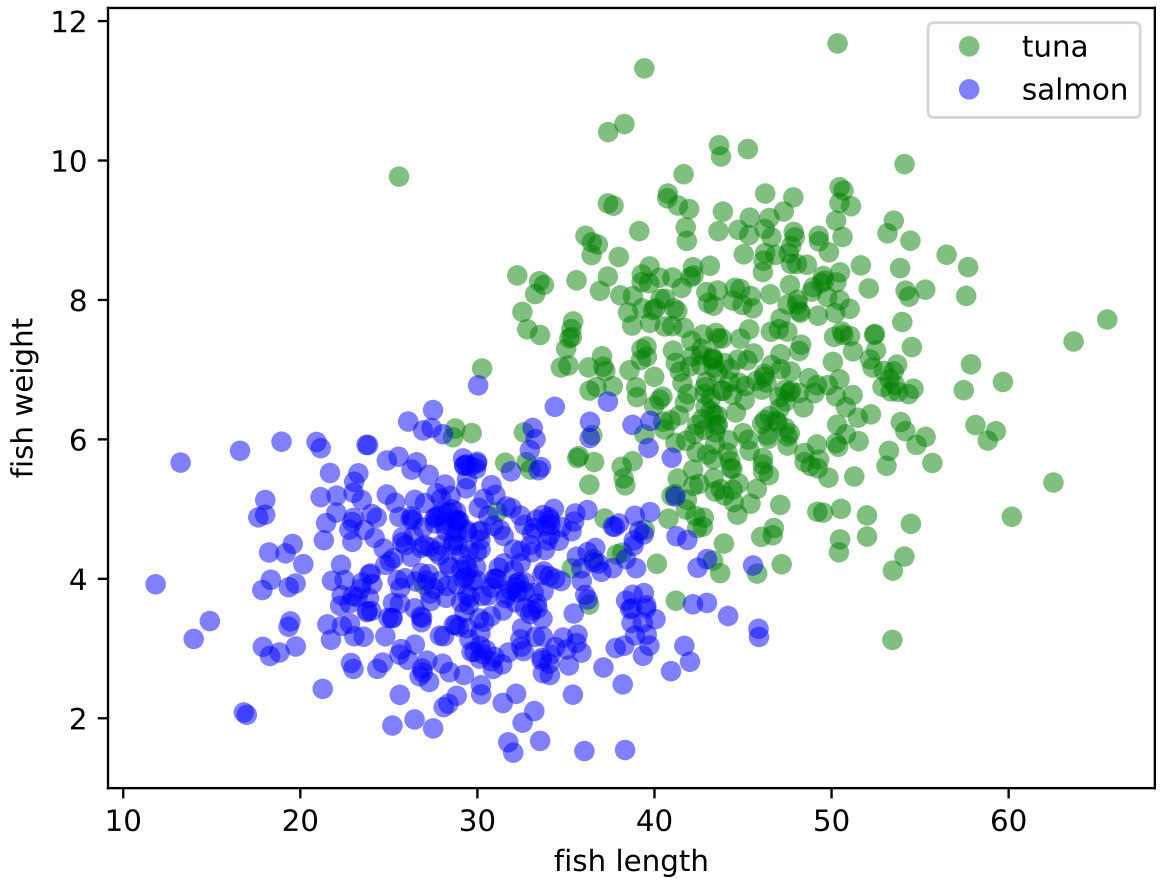
Decision trees

- Decision trees
- Construction of a decision tree
- Tree pruning

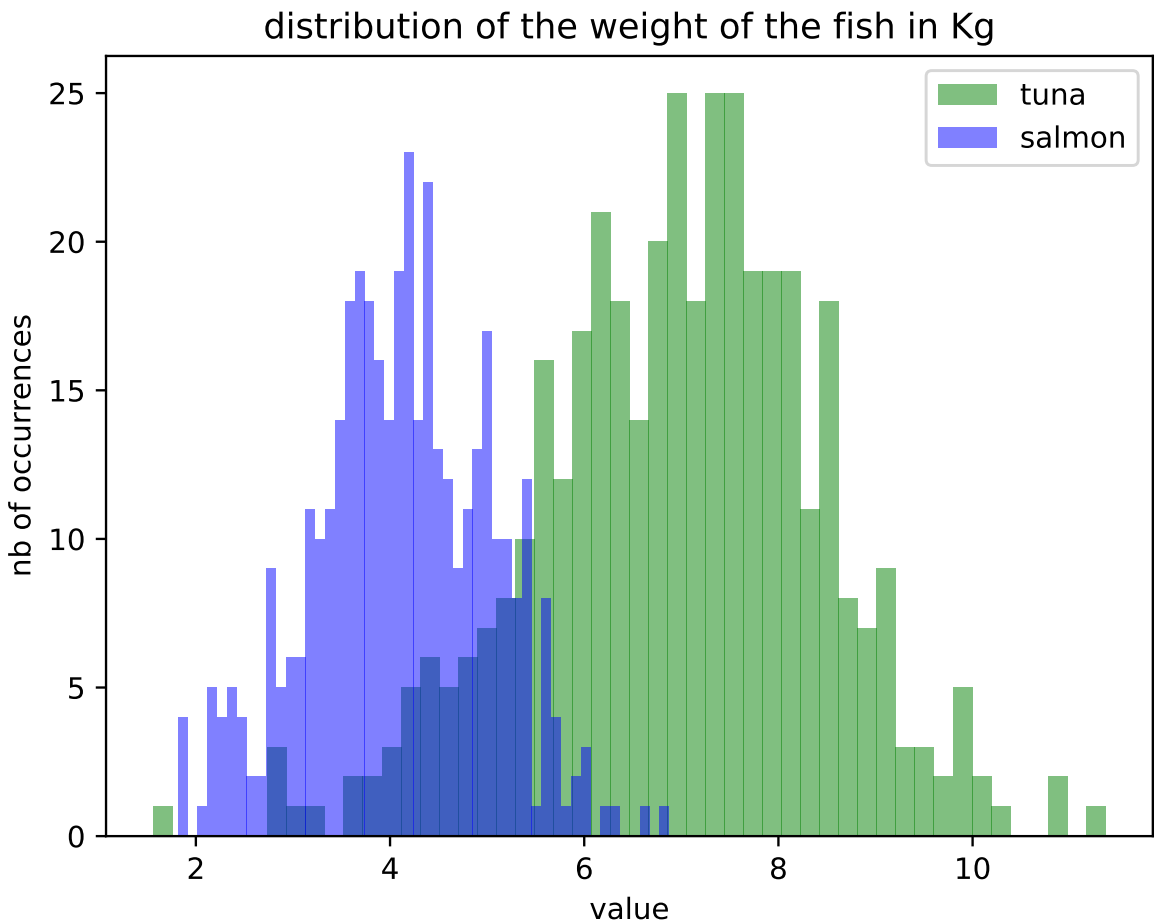
Ensemble learning

- Bagging
- Random forest

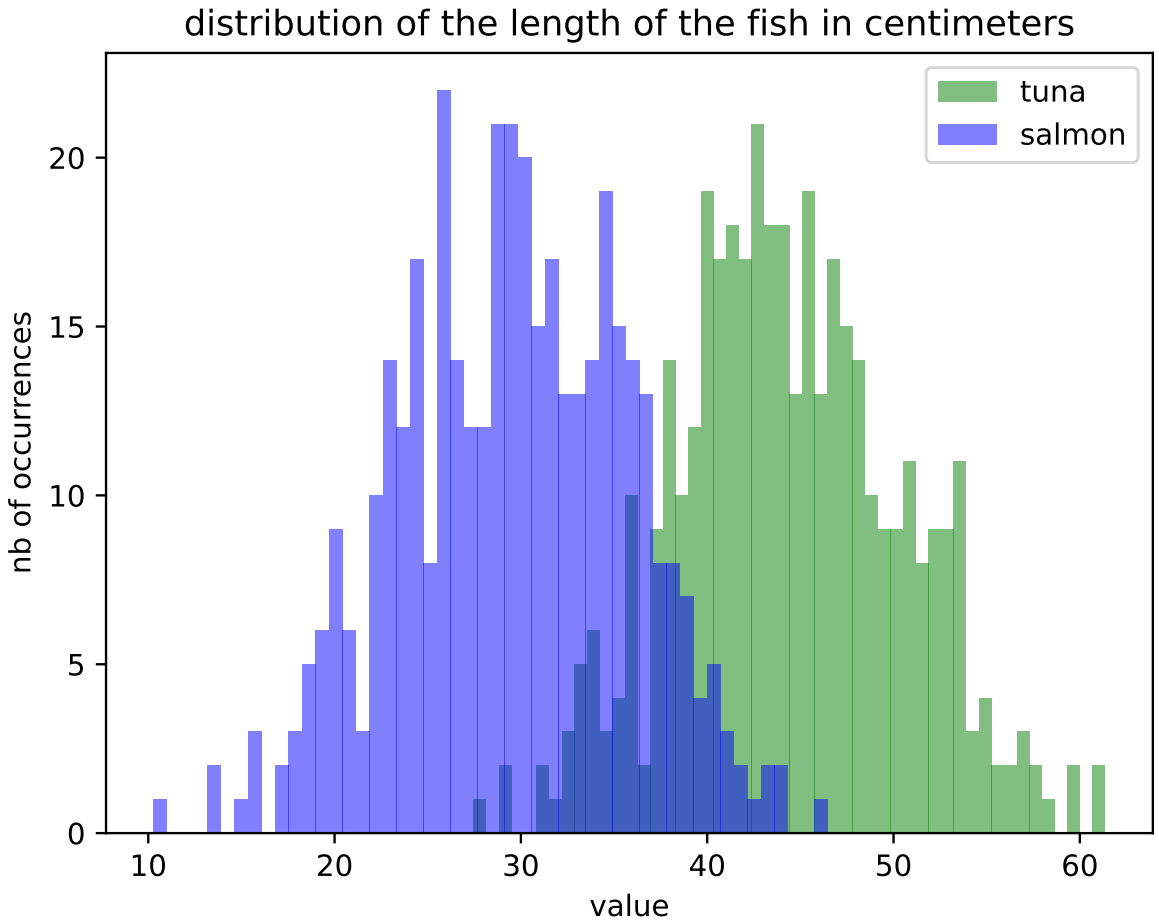
Example dataset : fishes



Fish dataset



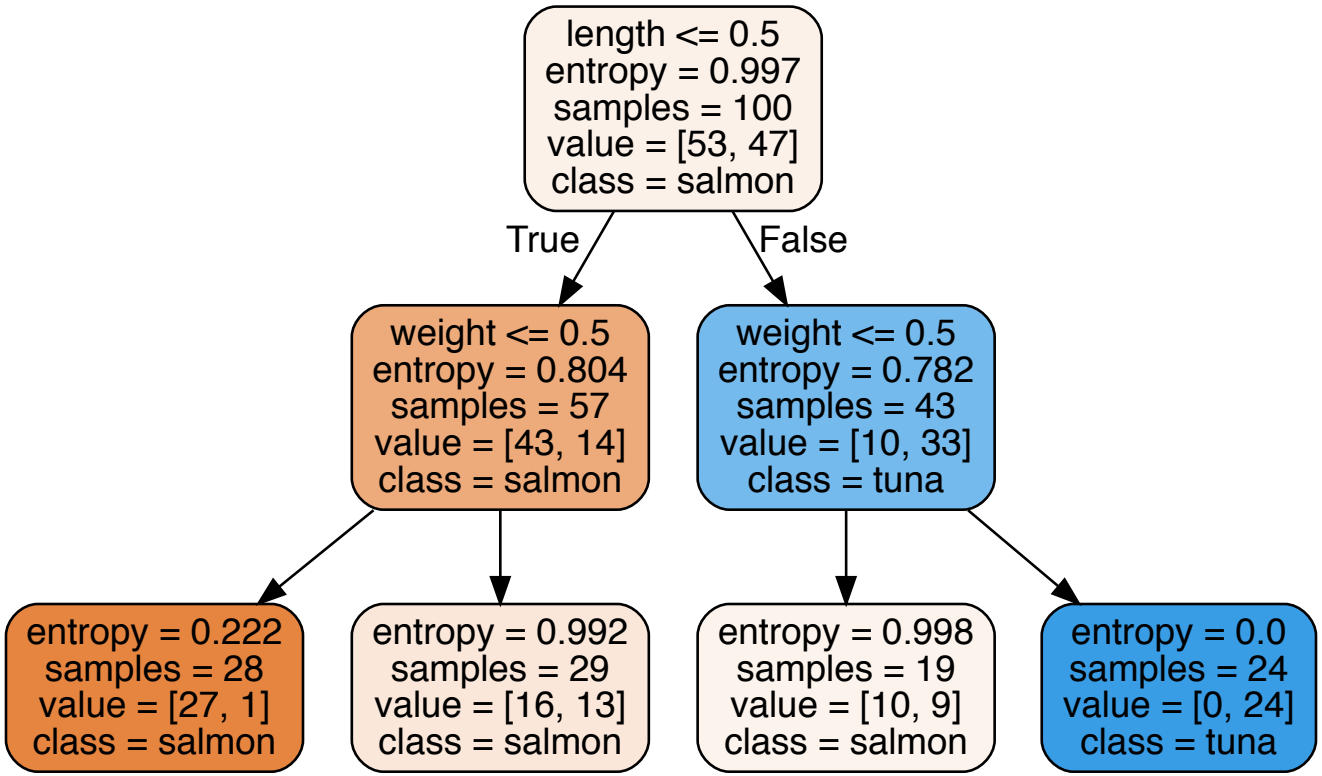
Fish dataset



CART

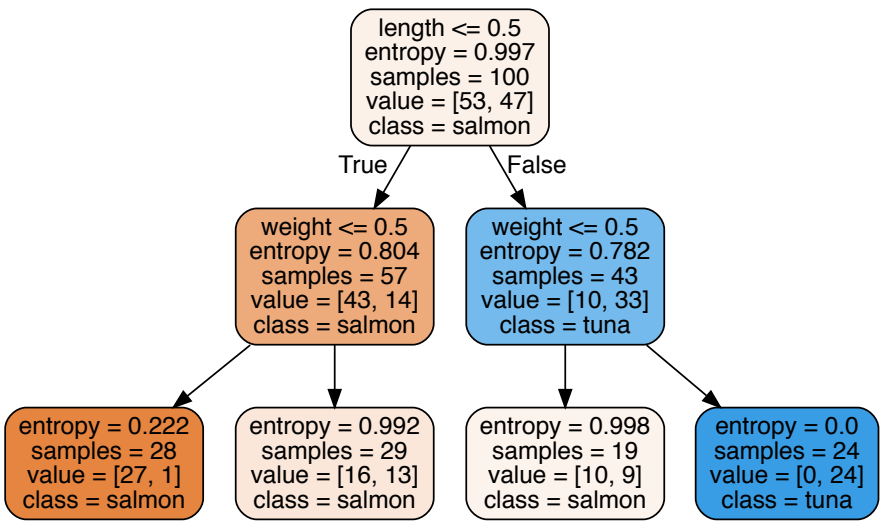
- Segmentation method (partitionnement récursif), **binary splits.**
- First algorithm : **CART** (classification and regression tree, Breiman, 1984) [Breiman et al., 2017]

Example tree



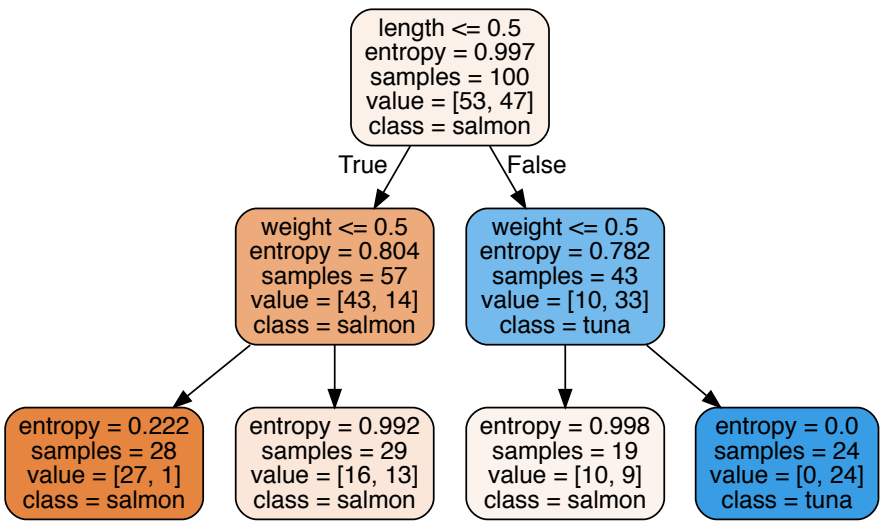
Splitting nodes

Each split should lead to more **homogeneous** nodes (in a sense that is to be formally defined)

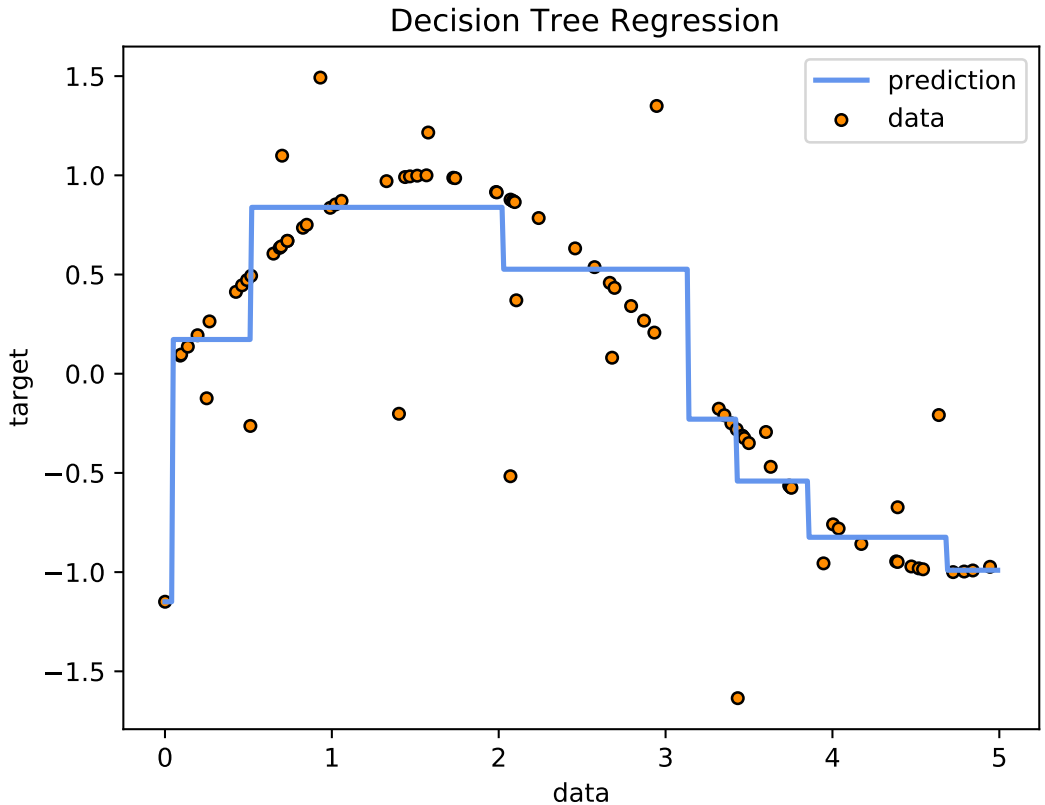


Splitting nodes

Leaf nodes correspond to a predicted value (class or real number for instance if $\mathcal{Y} = \mathbb{R}$)).



Decision trees can also be used for regression. In that case, the prediction is **piecewise constant**. It can be seen as a form of local averaging.



Construction of a tree

A split node corresponds to :

- a segmentation variable V
- a segmentation value / threshold if V is quantitative. If V is a class, it is a two-fold partitionning of the set of classes.

The construction of a tree requires a criterion to :

- choose the segmentation variable and value
- decide when a node is terminal (leaf node)
- associate a prediction value to all leaf nodes

Homogeneity

We want the homogeneity in the child nodes to be greater than in the parent node. Equivalently, we can define a non-negative heterogeneity function H that describes each node and that must be :

- 0 if the node is homogeneous (only one class or one output value is represented in this node)
- maximal if the values of y are uniformly distributed within the node.
- if $H(L)$ it the value of H for the left child node (and $H(R)$ for the right one), then the segmentation should minimize

$$H(L) + H(R) \tag{1}$$

Homogeneity criterion for regression

In the case of regression, $\mathcal{Y} = \mathbb{R}$ and the heterogeneity $H(n)$ of node n is its empirical variance :

$$H(n) = \frac{1}{|n|} \sum_{i \in n} (y_i - \bar{y}_n)^2 \tag{2}$$

Homogeneity criterion for classification

In the case of classification, $\mathcal{Y} = [1, \ldots, L]$ and several criteria exist.

Homogeneity criterion for classification : entropy (information gain)

- We use the convention that $0 \log(0) = 0$
- p_n^l is the proportion of class l in node n.

The **entropy** writes :

$$H(n) = -\sum_{l=1}^{L} p_n^l \log(p_n^l) \tag{3}$$

Exercice 1 : What are the maximum and minimum values of the entropy ?

Homogeneity criterion for classification : Gini impurity

The Gini impurity writes :

$$H(n) = \sum_{l=1}^{L} p_n^l (1 - p_n^l) \tag{4}$$

Homogeneity criterion for classification : Gini impurity

$$H(n) = \sum_{l=1}^{L} p_n^l (1 - p_n^l) \tag{5}$$

If we predict the classes in node n according to the proportions of the labels in n, then the Gini impurity is the probability of making a mistake, given that we are in node n.

Homogeneity criterion for classification : misclassification probability

$$H(n) = 1 - \max_{l}(p_n^l) \tag{6}$$

If we predict the most represented class in n, then this is the misclassification probability.

Prediction for a leaf node

- **regression** : predict the average value in the node
- **classification** : several possibilities
  - most represented class in the node
  - most probable class *a posteriori* if the *a priori* probabilities are known (Bayesian probabilities)
  - class that costs the less in case of missclassification

Pruning

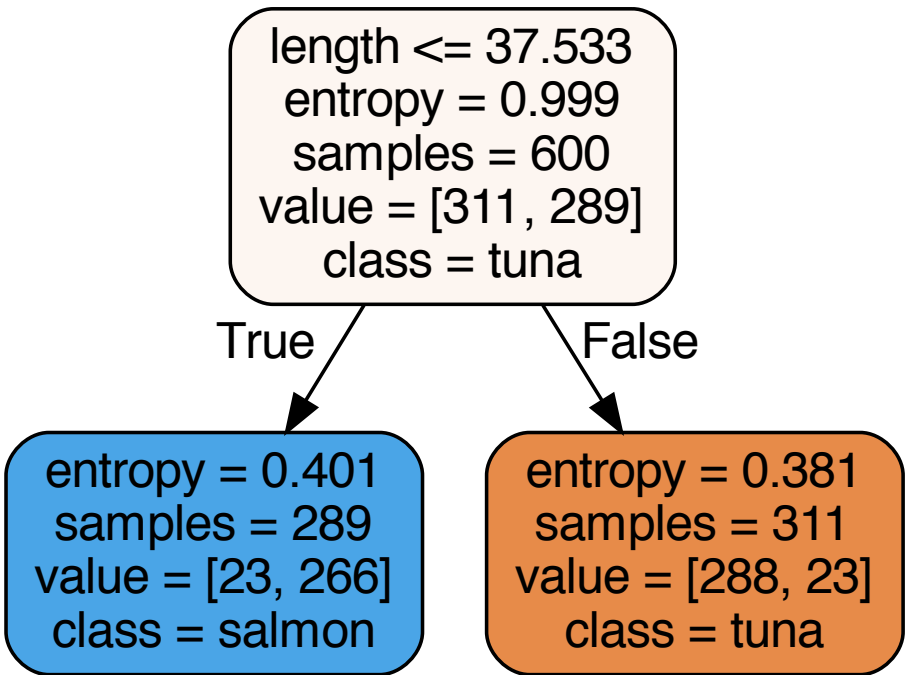
- If the segmentation is not restricted, the tree can fit the training dataset perfectly (overfitting).
- learning would then be highly dependent on the choice of the training dataset, leading to **instability.**
- To avoid it, **pruning** (élaguage) is used : it consists in restricting the size of the tree (and can be seen as an equivalent to regularization when we perform ridge regression of logistic regression).

Pre-pruning by restricting divisions

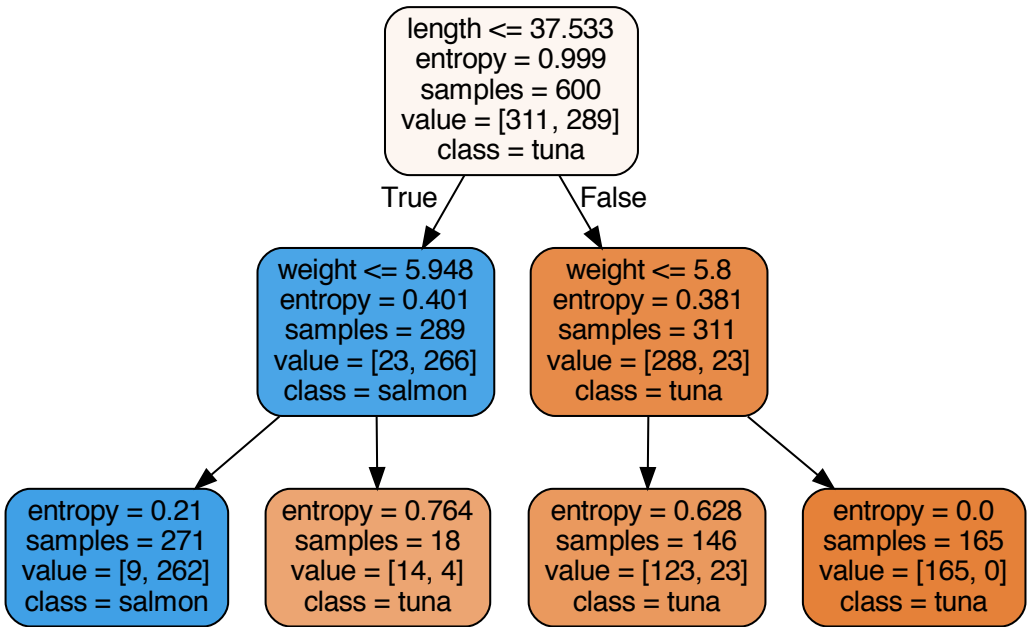
We can impose

- minimum impurity decrease
- a minimum number of samples in a leaf node
- a minimum number of samples to authorize splitting a node

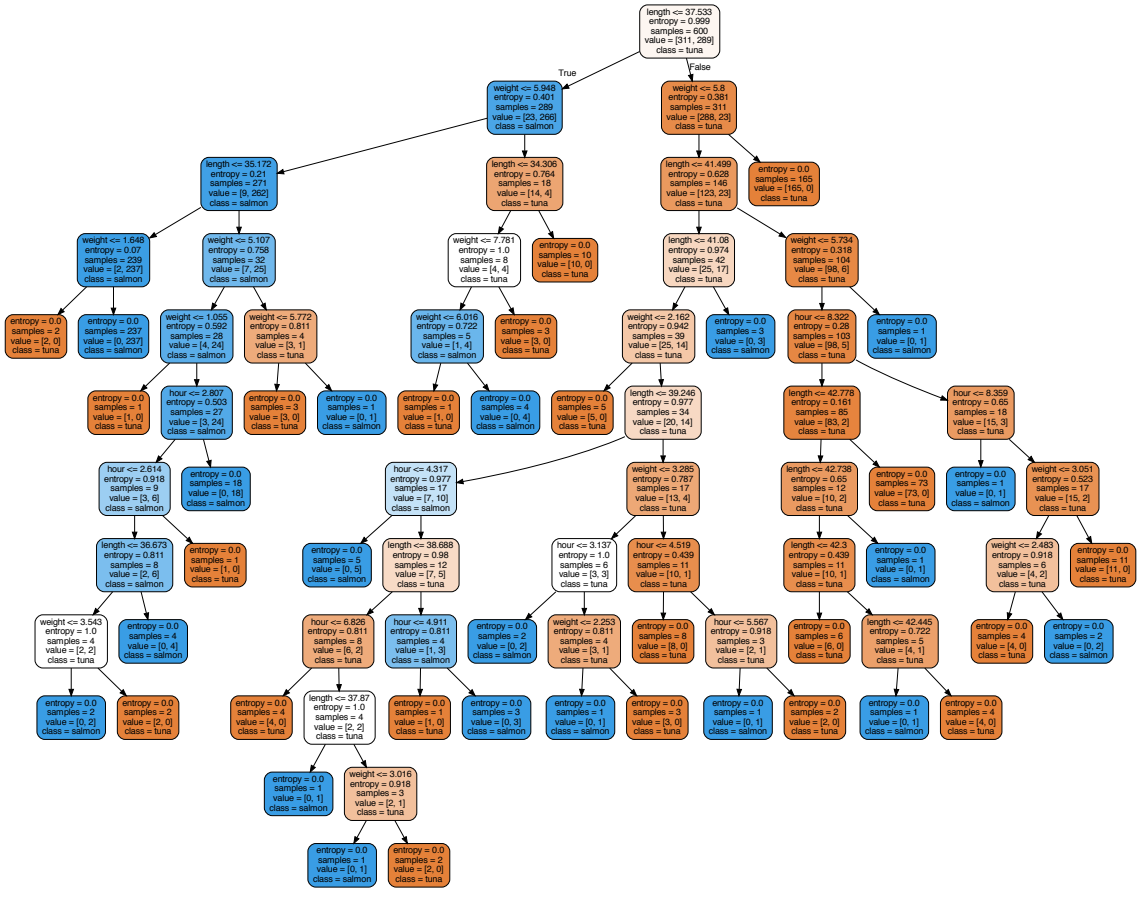
Simulation : fish dataset, max depth=1



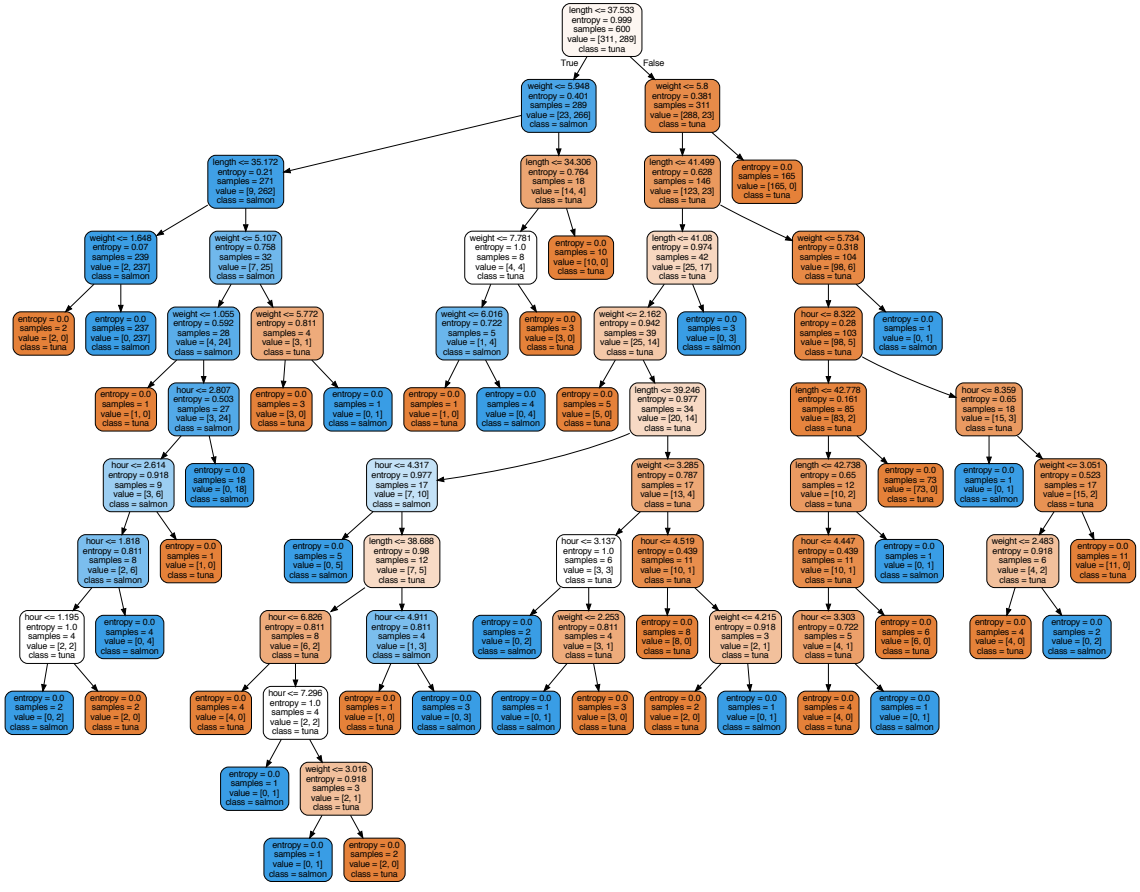
Simulation : fish dataset



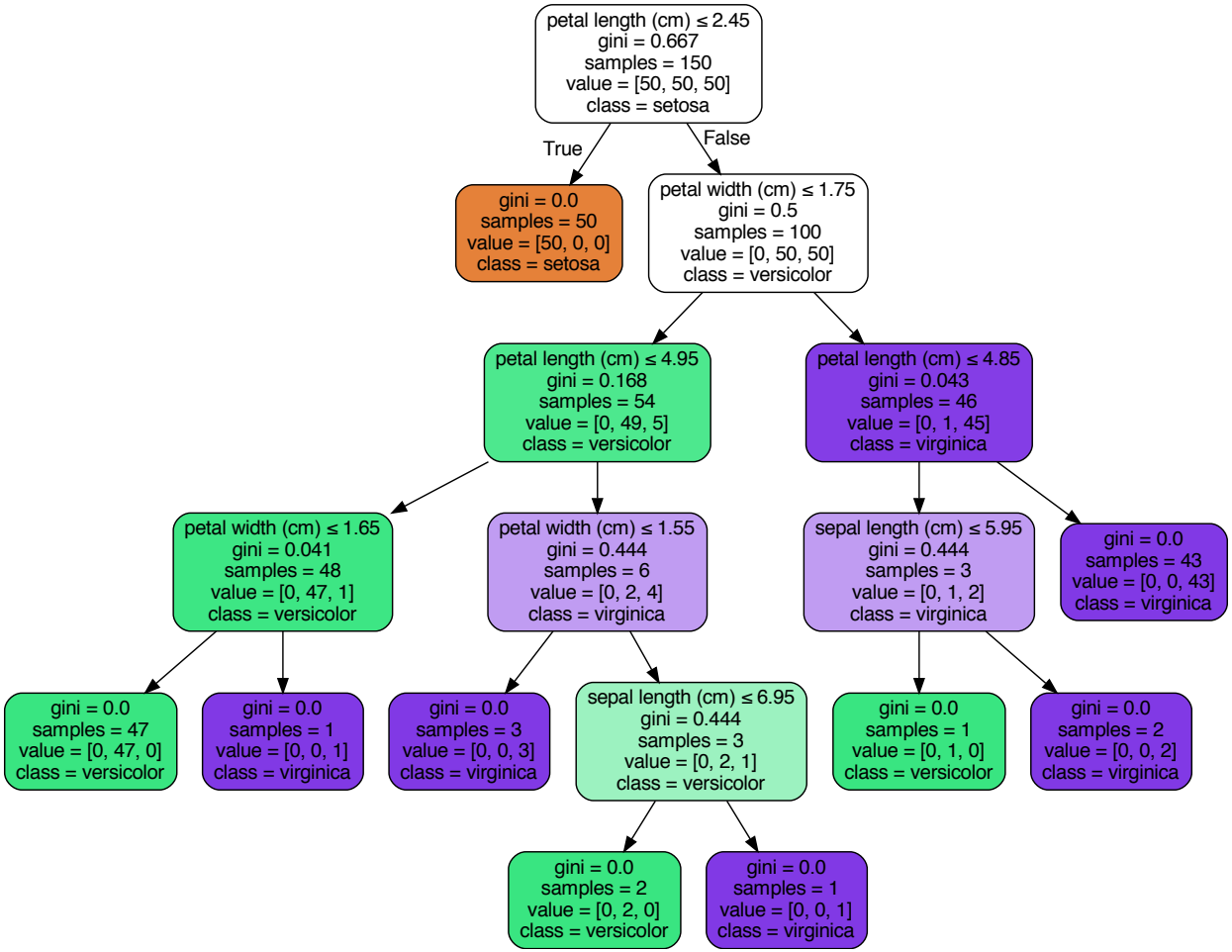
Simulation : fish dataset



Simulation : fish dataset



Overfitting : Iris dataset, max depth=34



Avoiding overfitting : Iris dataset

Exercice 2 :

Avoid overfitting by choosing a pruning method on the iris dataset (**cart/iris/**)

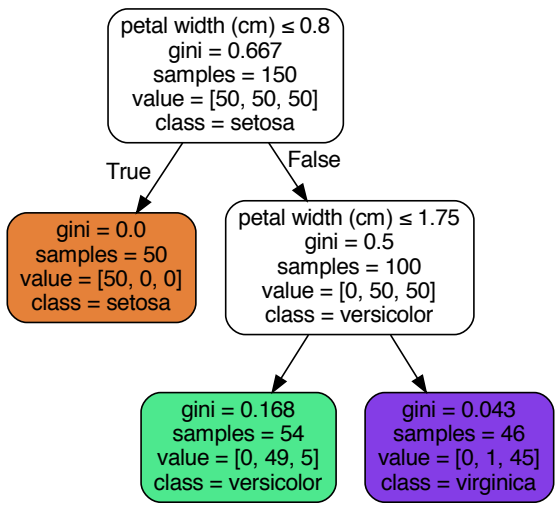


Figure – Avoid overfitting with pre-pruning.

Post-pruning

It is also possible to prune after the construction of a large tree.

Remarks on decision trees I

- Decision trees do not require hypotheses on the distribution of the features.
- Feature selection is integrated in the algorithm, so they might be relevant in a situation with many features (variables explicatives)
- Decision trees allow a straightforward **interpretation** of the model.

Remarks on decision trees II

- Decision trees are greedy : they might find a local minimum
- they are hierarchical : a bad choice in a segmentation at the top of the tree propagates in the whole tree
- Hence their instability and sensibility to the choice of the training set.

Ensemble learning

- *Bagging* and *boosting* are methods that combine estimators (in parallel or sequentially)
- they are often applied to decision trees
- they reach state-of-the-art performance in several supervised learning tasks [Fernández-Delgado et al., 2014] (pdf in documents/ at the root of the repo)
- they are an active area of research (both theoretically and for practical applications).

`https://scikit-learn.org/stable/modules/ensemble.html`

Aggregating

- usual setup : input space $\mathcal{X}$, output space $\mathcal{Y}$.
- we note z_b a dataset sampled from the unknown distribution ρ. If we sample b different datasets, $b \in [1, B]$,

$$z_b = \{(x_{1b}, y_{1b}), \dots (x_{nb}, y_{nb})\} \tag{7}$$

- we note $\hat{f}_{z_b}$ the estimator obtained after learning with z_b.

Aggregating

- we note z_b a dataset sampled from the unknown distribution ρ. If we sample b different dataset, $b \in [1, B]$,

$$z_b = \{(x_{1b}, y_{1b}), \dots (x_{nb}, y_{nb})\} \tag{8}$$

- we note $\hat{f}_{z_b}$ the estimator obtained after learning with z_b.

Aggregating consists in using as an estimator :

- for regression

$$\hat{f}_B = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_{z_b} \tag{9}$$

- for classification

$$\hat{f}_B(x) = \arg\max_j |\{b, \hat{f}_{z_b}(x) = j\}| \tag{10}$$

Aggregating

We note $\hat{f}_{z_b}$ the estimator obtained after learning with z_b.
Aggregating consists in using as an estimator :

- for regression

$$\hat{f}_B = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_{z_b} \tag{11}$$

- for classification (voting)

$$\hat{f}_B(x) = \arg\max_j |\{b, \hat{f}_{z_b}(x) = j\}| \tag{12}$$

The goal is to reduce the variance of the estimator.

Bootstrapping

If B is large, it is not possible to sample B independent datasets with n samples, from a finite dataset.
Bootstrapping consists in sampling B times a sample dataset with n elements **with replacement**.

Bagging

Bagging is the combination of bootstrapping and aggregating.

Out-of-bag error

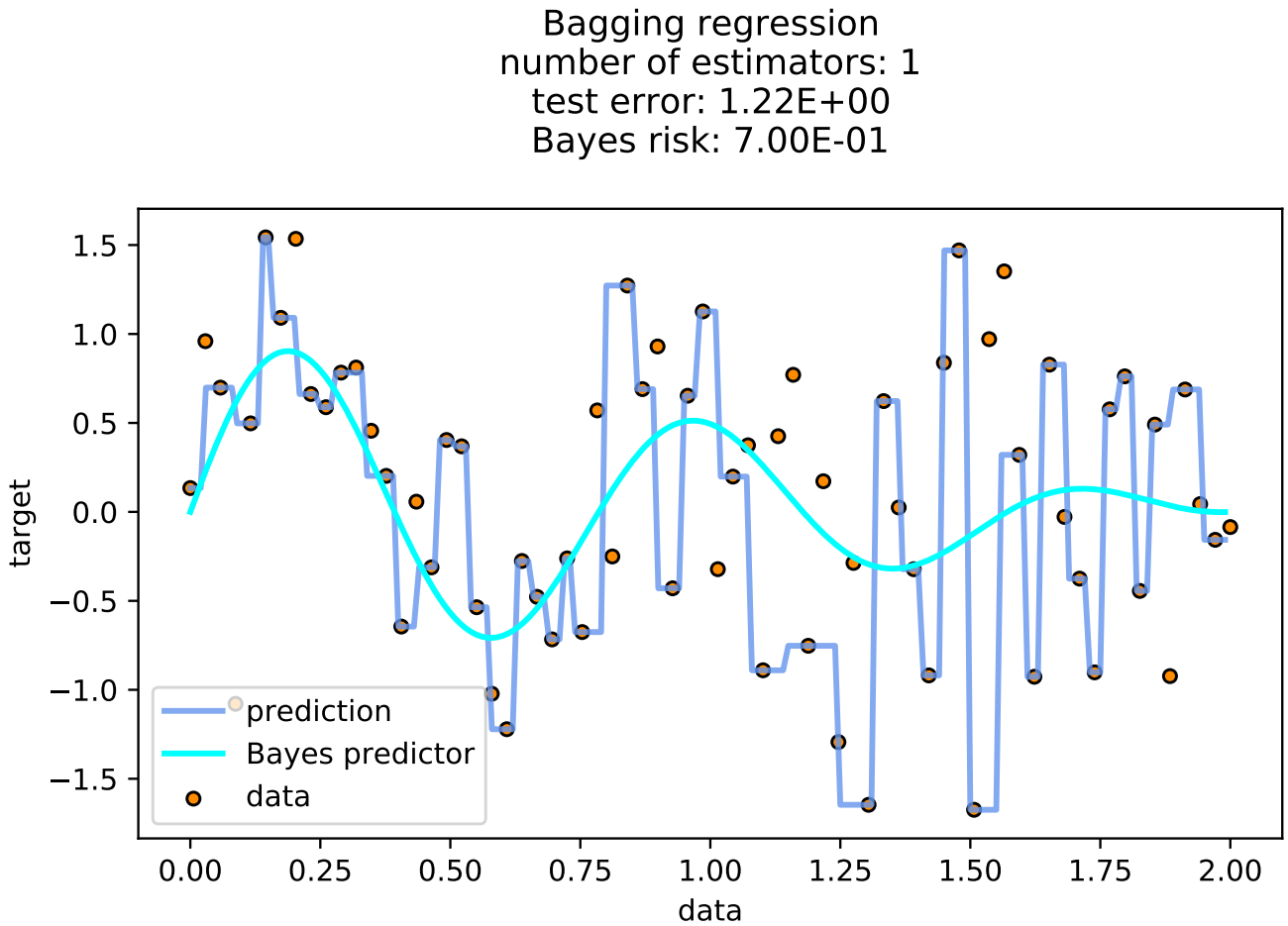
For an observation (x_i, y_i), the **out-of-bag error** is the mean error on this observation among the estimators that were trained on a bootstrap dataset **not** containing it.

Trees and bagging

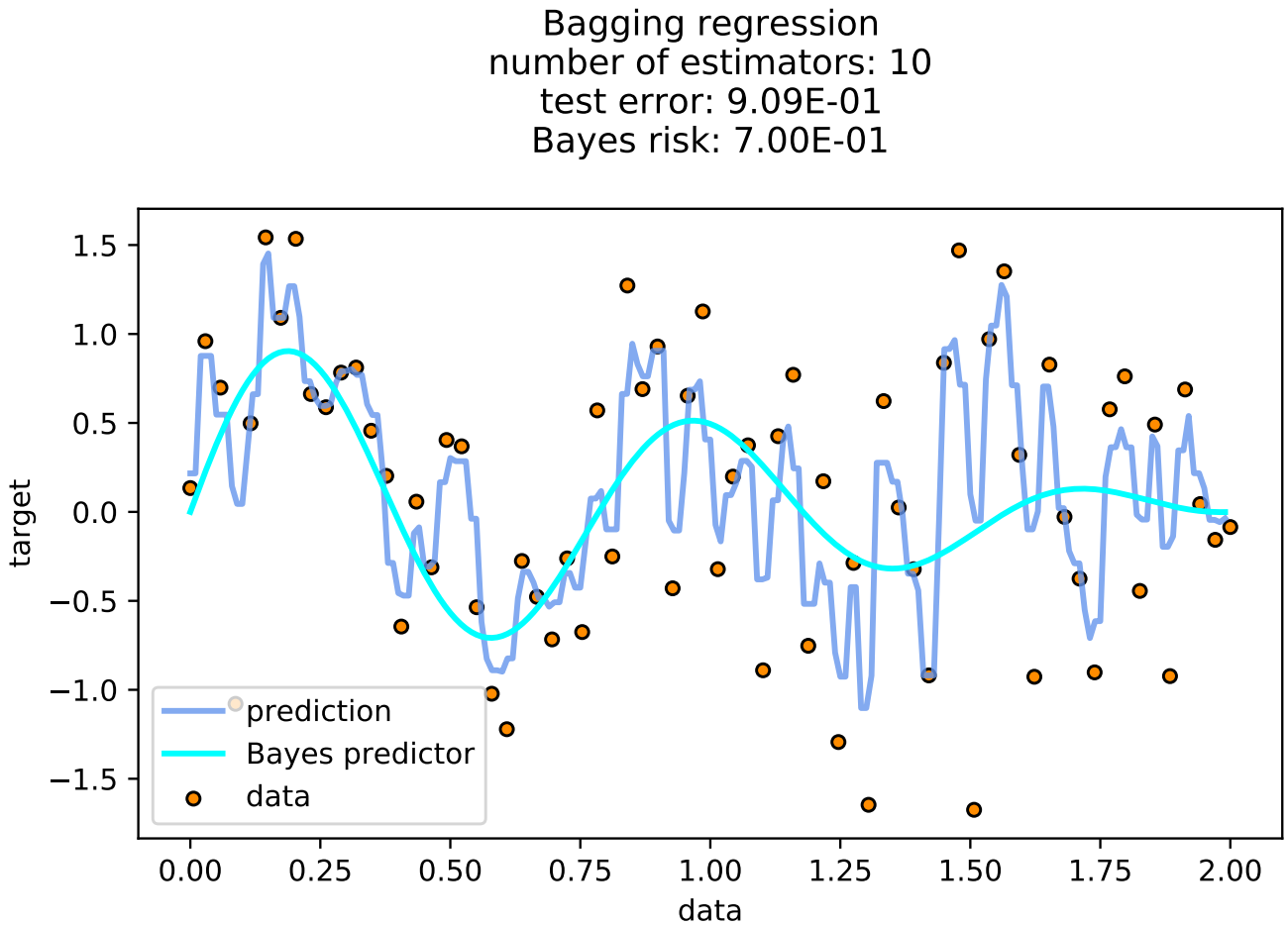
Bagging is most often used with decision trees. Several strategies are possible in order to build the base estimators $\hat{f}_{z_b}$ and to prune them.

- pre-pruning by enforcing a minimal number of samples per leaf node (e.g. 5) : good tradeoff between computation speed and quality of the prediction. The goal is to reduce high variance of the base estimators by averaging.
- pre-pruning by enforcing a maximal tree depth or maximal number of leaves.
- post-pruning with cross validation (more computationally expensive with less gains).

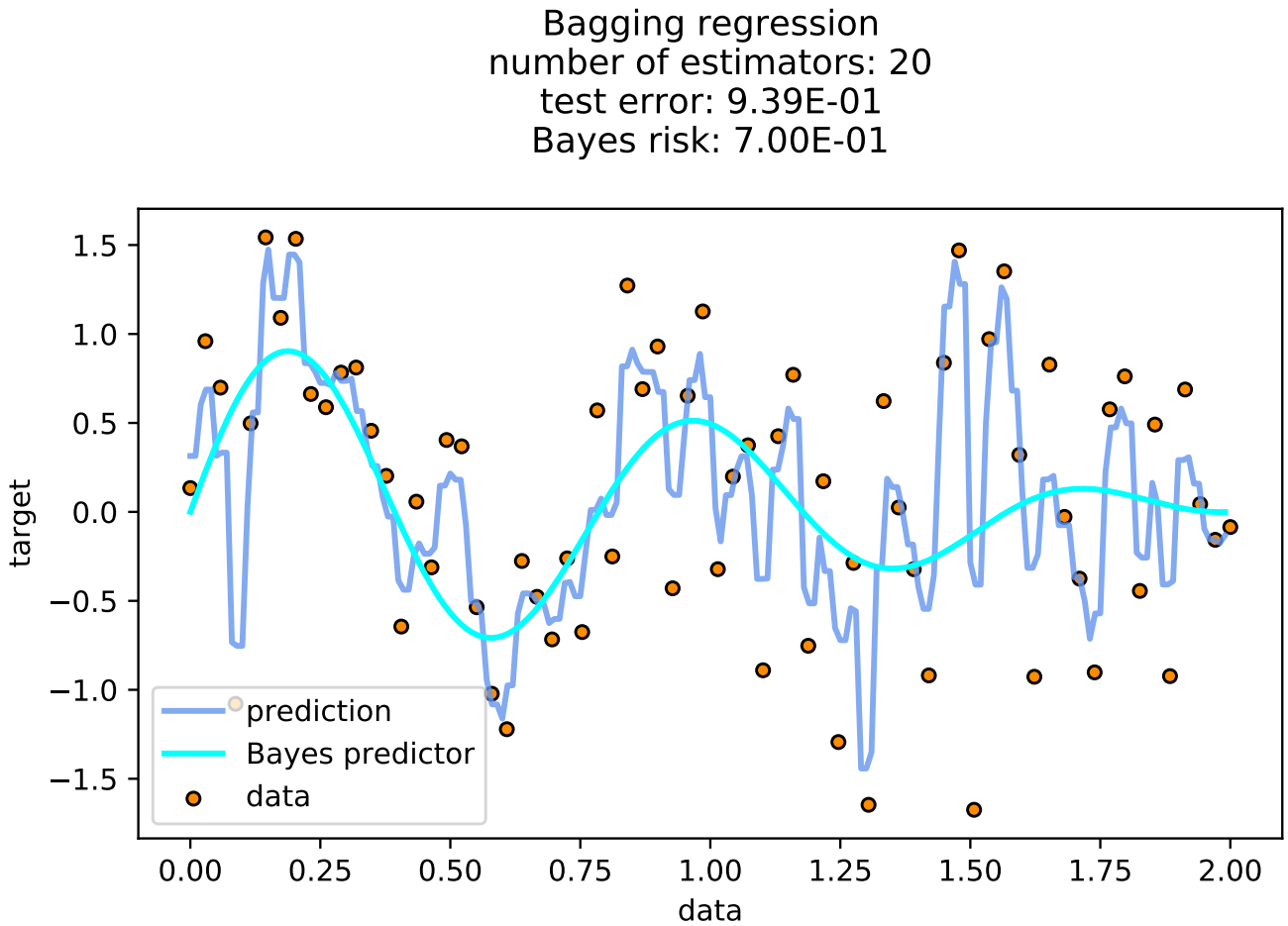
Simulation


Bagging regression
number of estimators: 1
test error: 1.22E+00
Bayes risk: 7.00E-01
target
data
prediction
Bayes predictor
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

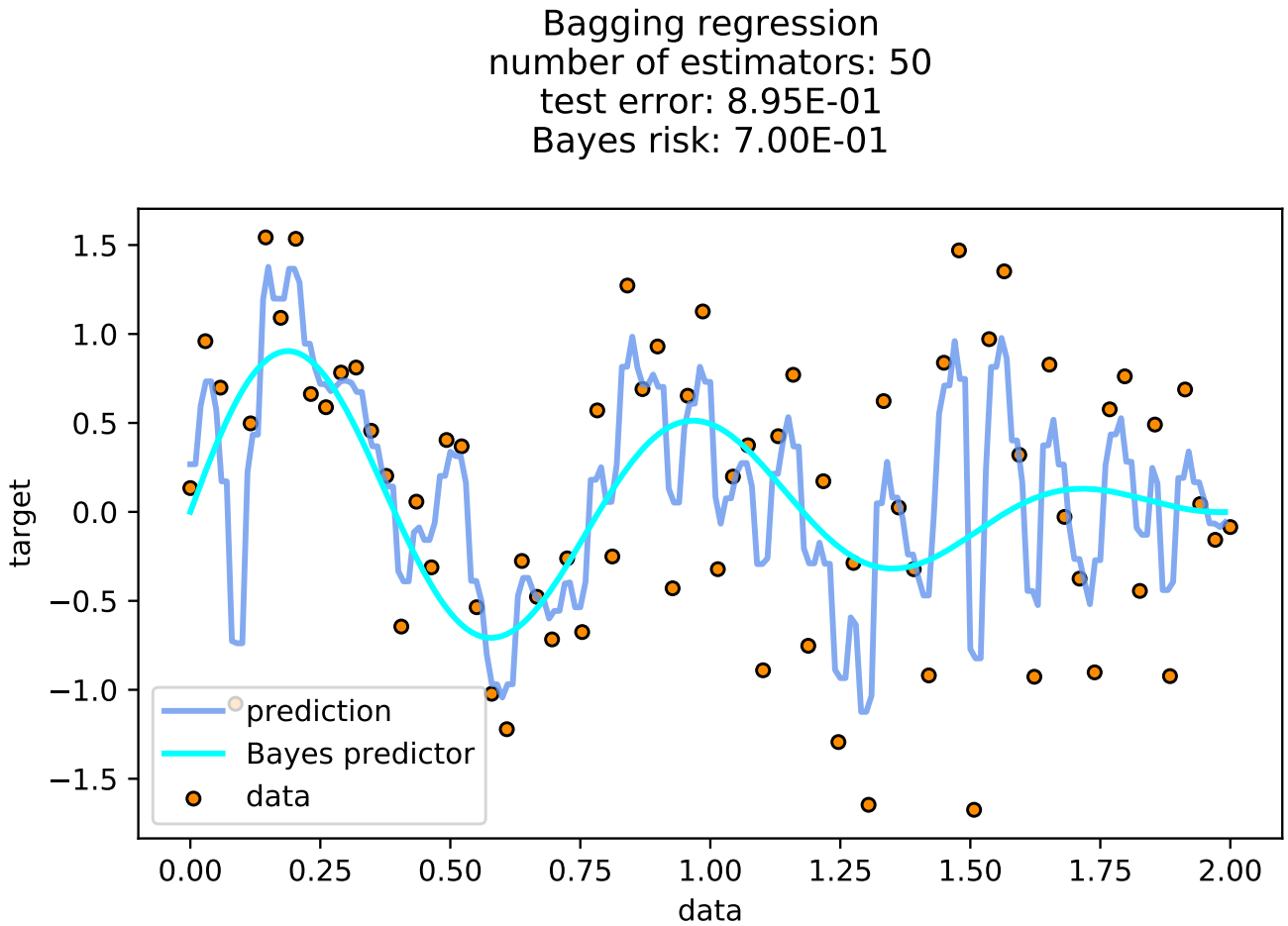
Simulation


Bagging regression
number of estimators: 10
test error: 9.09E-01
Bayes risk: 7.00E-01
target
data
prediction
Bayes predictor
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

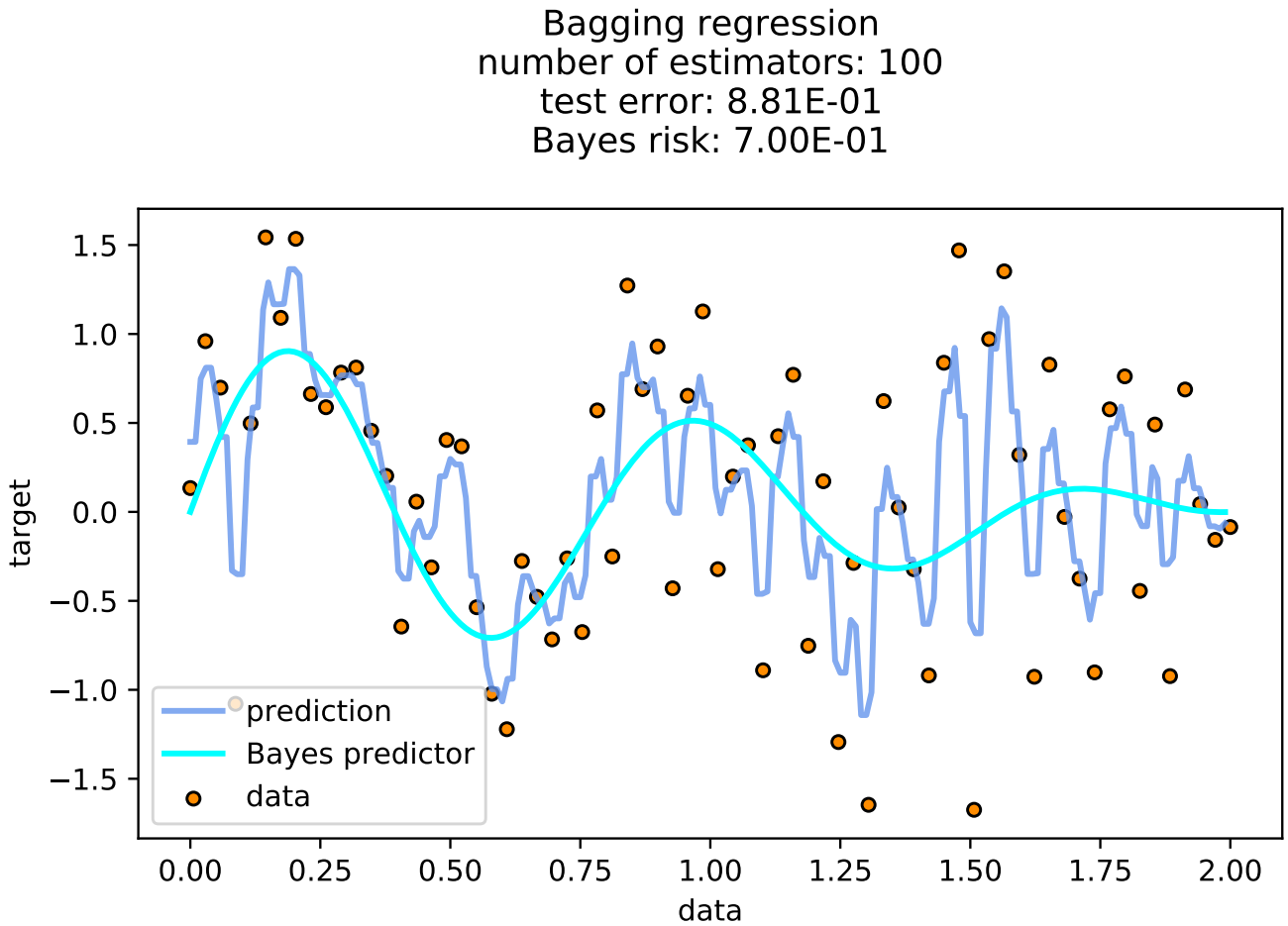
Simulation


Bagging regression
number of estimators: 20
test error: 9.39E-01
Bayes risk: 7.00E-01
target
data
prediction
Bayes predictor
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Simulation


Bagging regression
number of estimators: 50
test error: 8.95E-01
Bayes risk: 7.00E-01
target
data
prediction
Bayes predictor
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Simulation


Bagging regression
number of estimators: 100
test error: 8.81E-01
Bayes risk: 7.00E-01
target
data
prediction
Bayes predictor
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Individual estimators

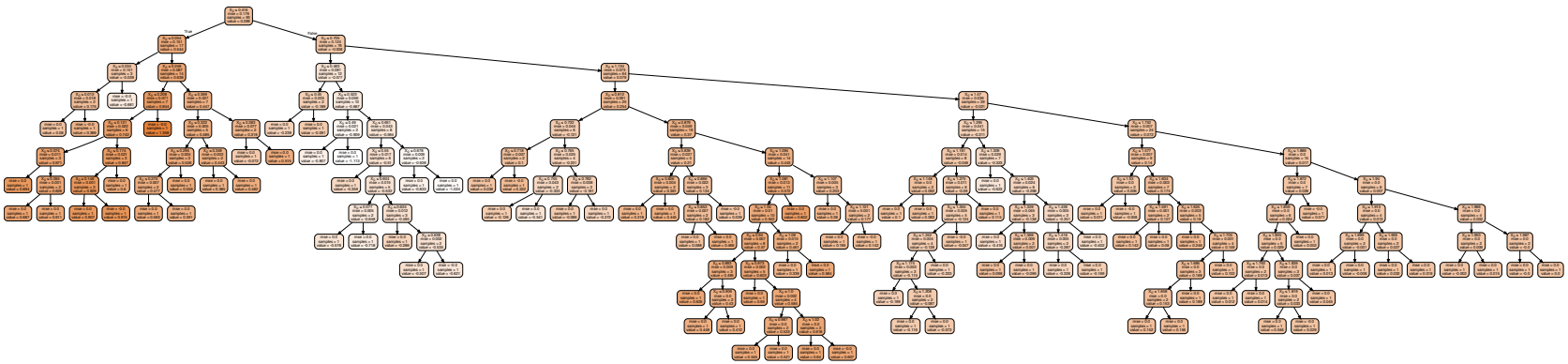


Figure – Estimator used in the averaging

Individual estimators

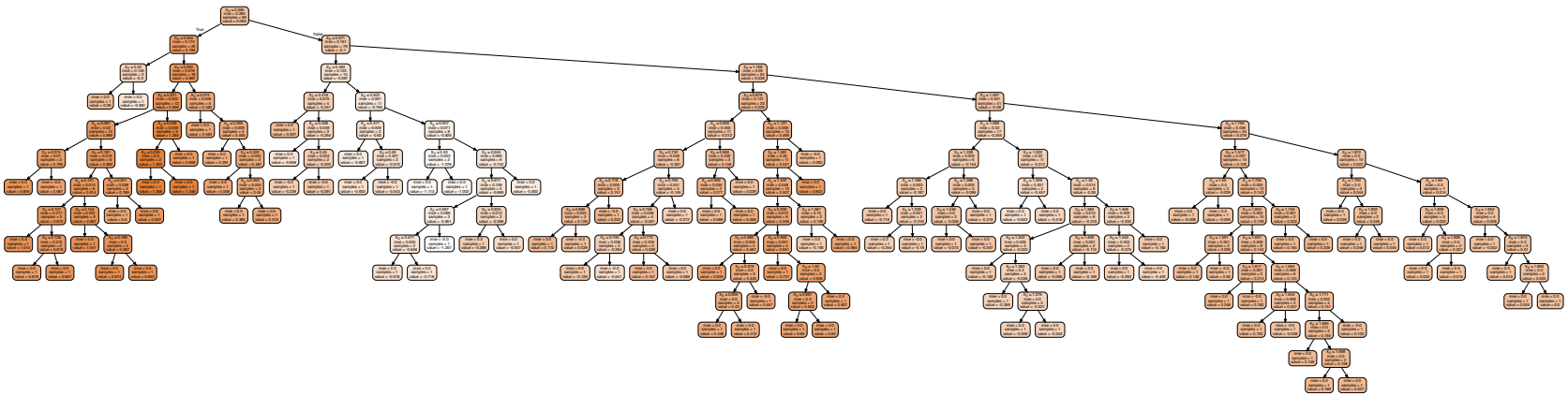


Figure – Other estimator used in the averaging

Possible issues

- number of trees to compute (B) in order to have a stable out-of-bag error.
- need for storing all the base estimators
- the final estimator is a *black-box* model.

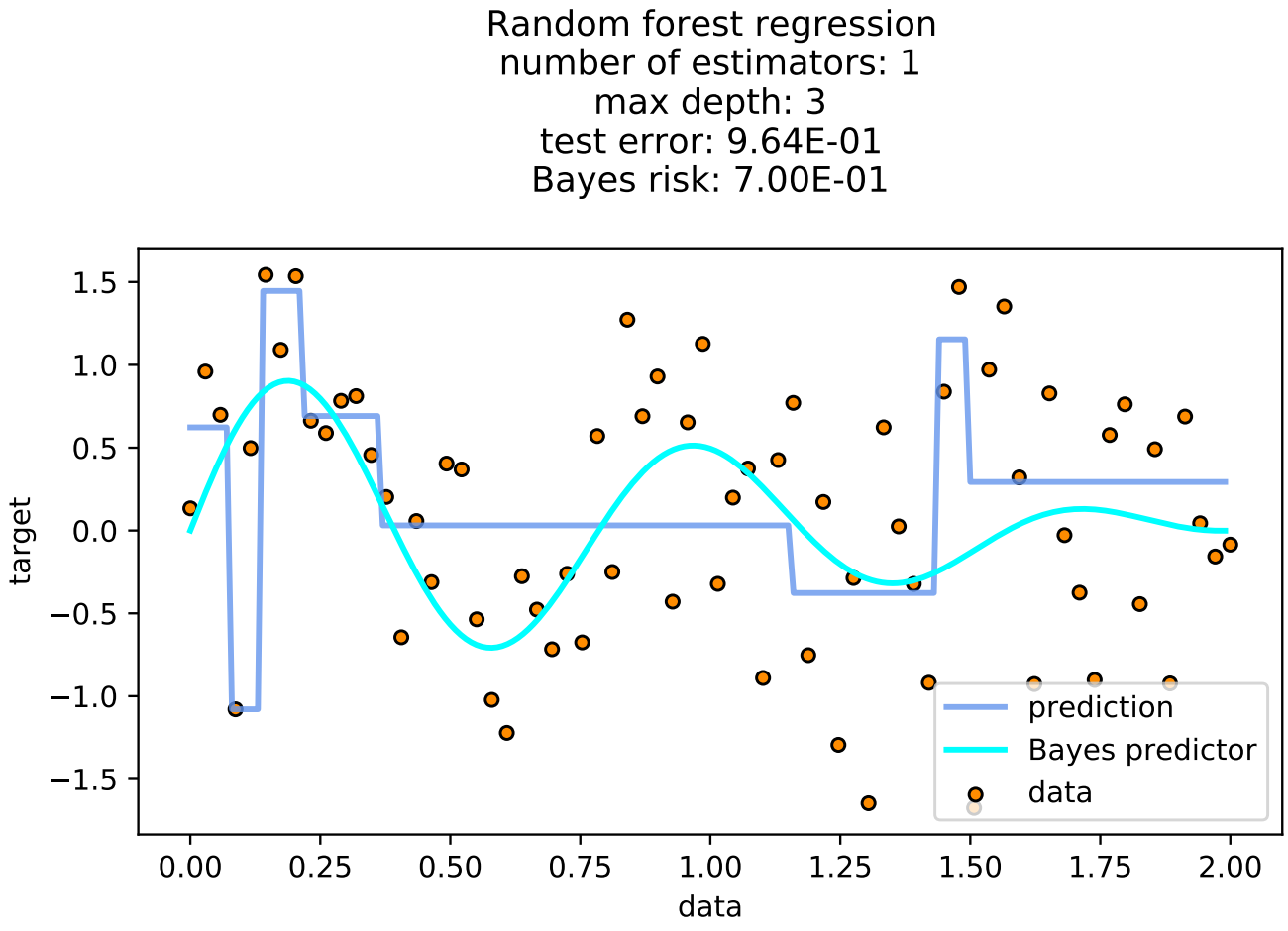
Random forest

- Random forest [Breiman, 2001] is a bagging method with binary CART estimators, with additional randomness added to the choice of segmentation variables.
- the goal is to increase the **independence** between the base trees.
- although the theoretical properties of random forest are not yet fully understood, it is an important benchmark in supervised learning (but for instance not when the problem can be solved in a linear way).

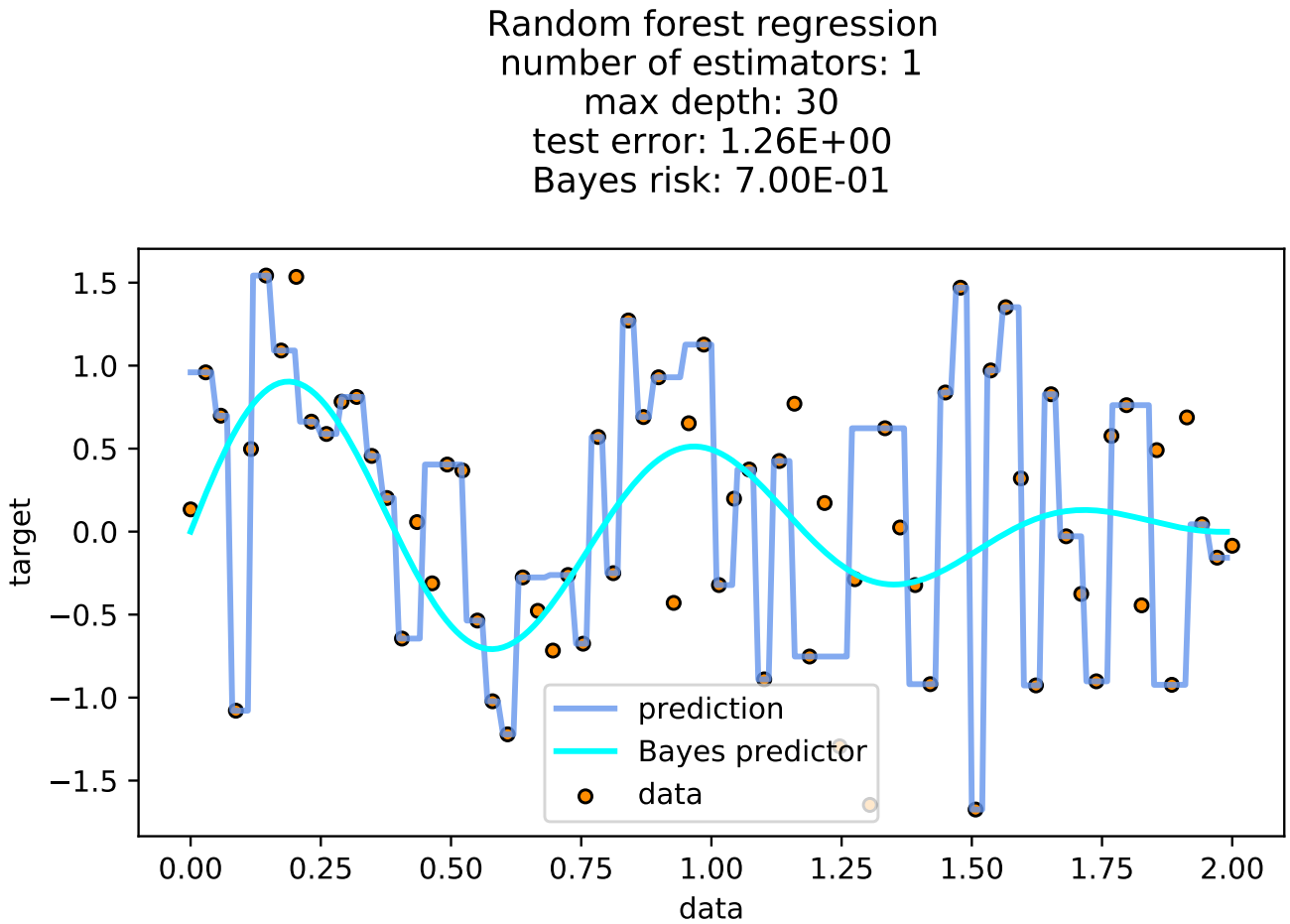
Random forest II

- With random forests, it is possible to use a more brutal pruning (for instance using only trees of depth 2)
- To tune m, heuristics are used. Common ones include :
  - $m = \sqrt{d}$ for classification (d is the number of features)
  - $m = d/3$ for regression
  - cross validation.

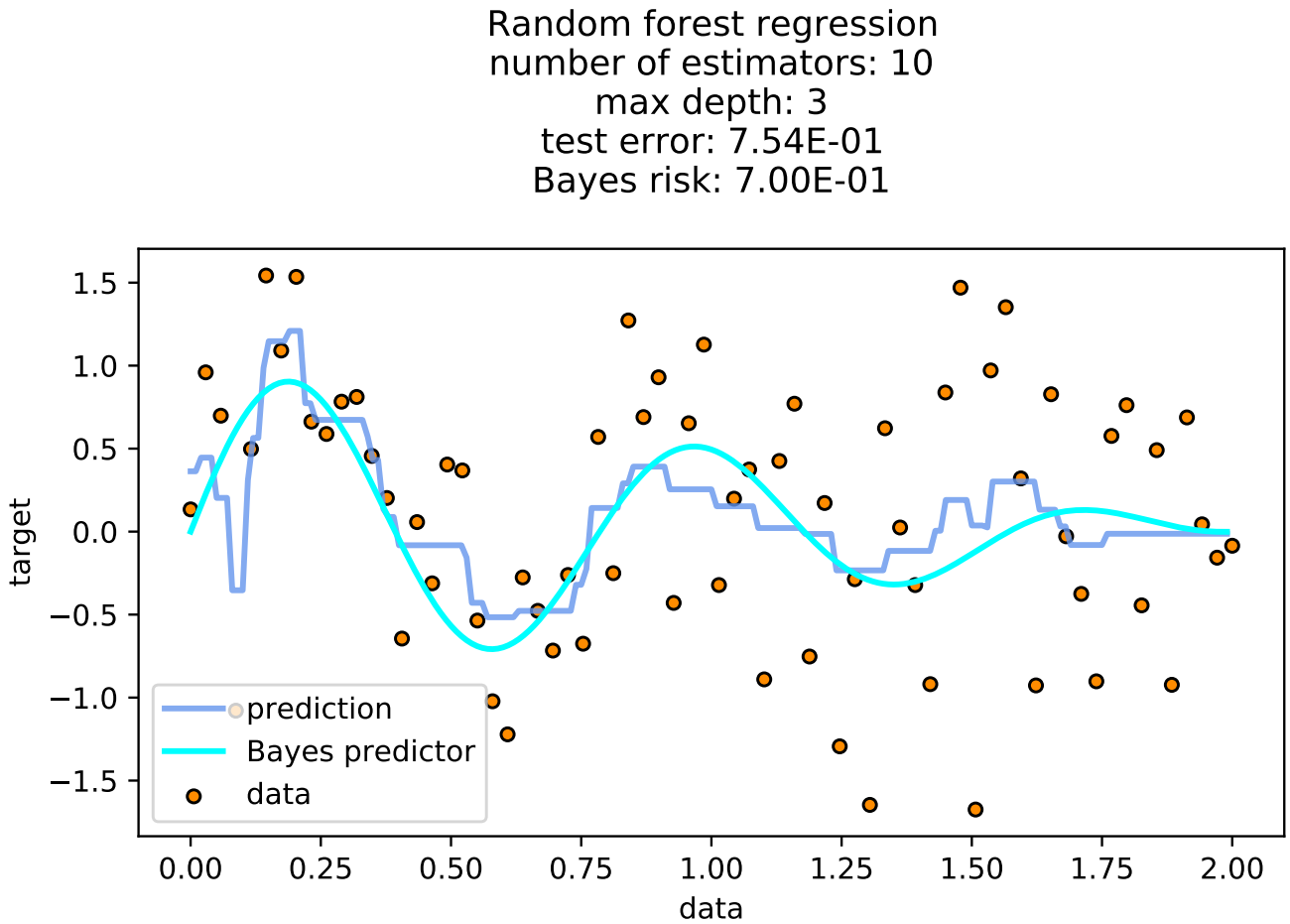
Example in 1D



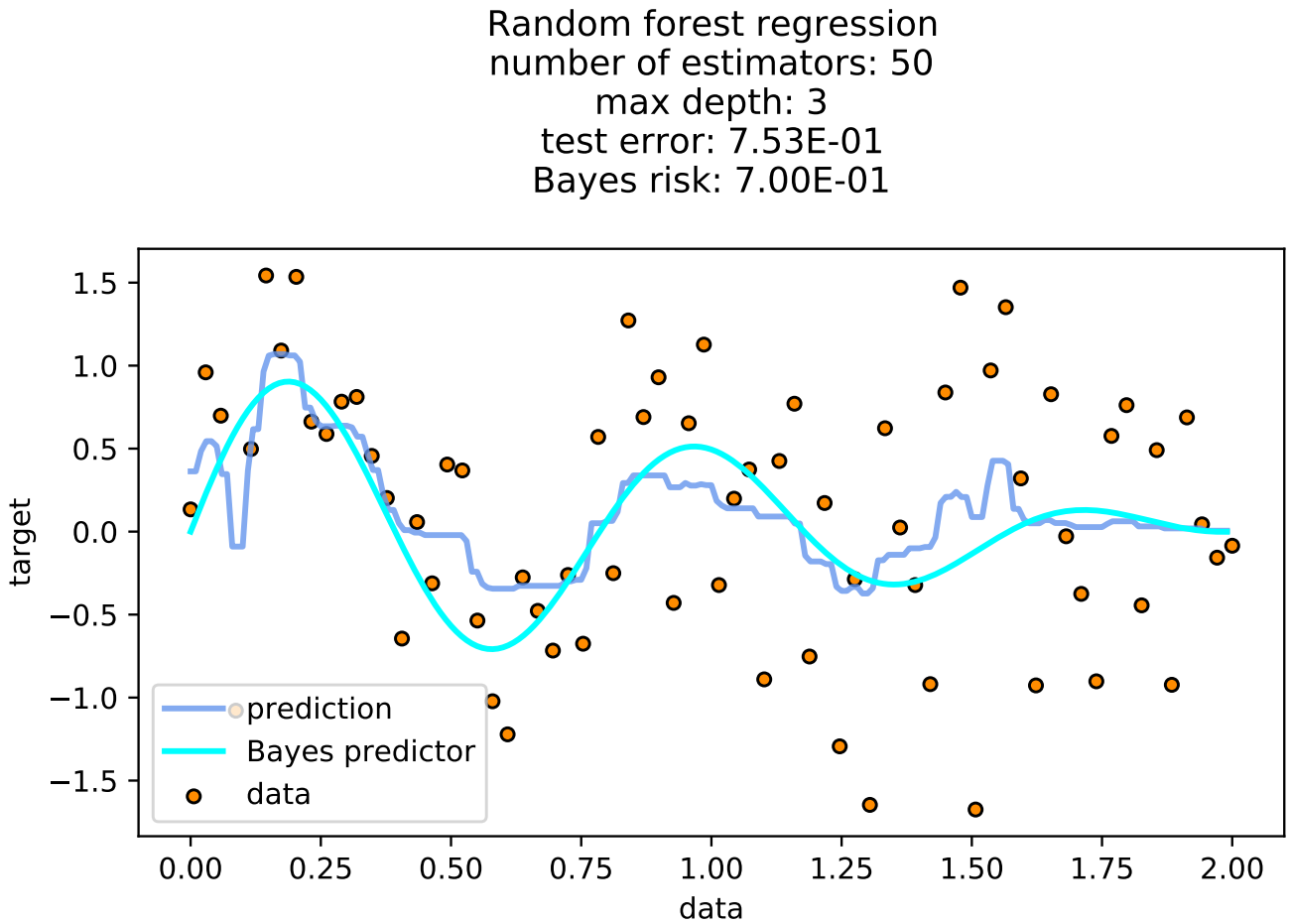
Example in 1D



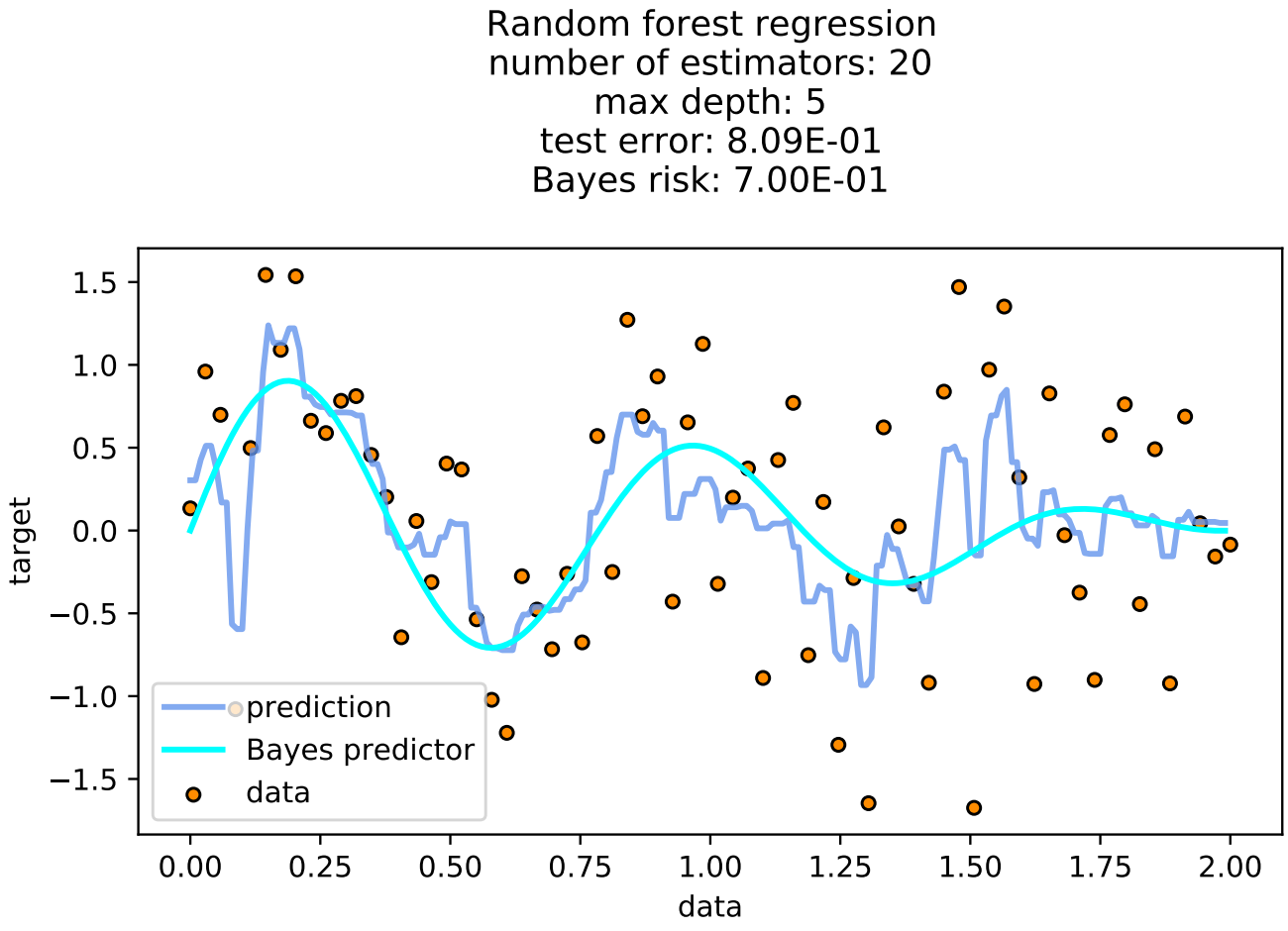
Example in 1D



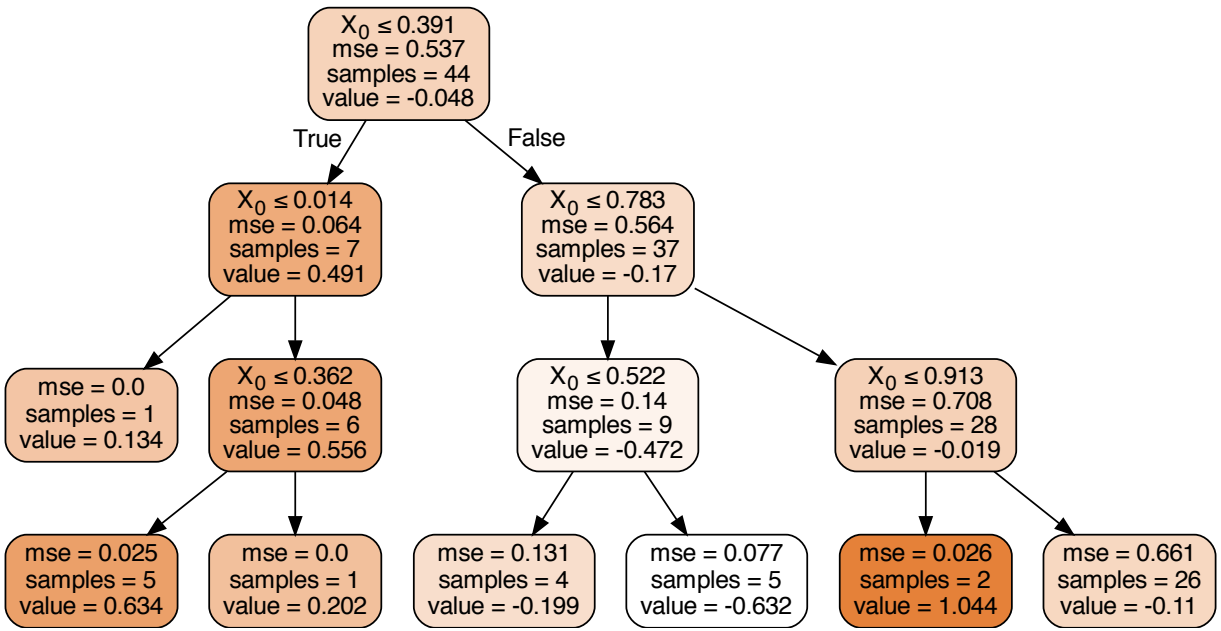
Example in 1D



Example in 1D



Estimators


$X_0 \leq 0.391$
mse = 0.537
samples = 44
value = -0.048
True
False
$X_0 \leq 0.014$
mse = 0.064
samples = 7
value = 0.491
$X_0 \leq 0.783$
mse = 0.564
samples = 37
value = -0.17
mse = 0.0
samples = 1
value = 0.134
$X_0 \leq 0.362$
mse = 0.048
samples = 6
value = 0.556
$X_0 \leq 0.522$
mse = 0.14
samples = 9
value = -0.472
$X_0 \leq 0.913$
mse = 0.708
samples = 28
value = -0.019
mse = 0.025
samples = 5
value = 0.634
mse = 0.0
samples = 1
value = 0.202
mse = 0.131
samples = 4
value = -0.199
mse = 0.077
samples = 5
value = -0.632
mse = 0.026
samples = 2
value = 1.044
mse = 0.661
samples = 26
value = -0.11

Importance of the variables

Although the obtained estimator is hard to interpret, we can still study the importance of variables.

Mean decrease accuracy

Consider a variable j. For a given tree b

- compute the out-of-bag error
- shuffle the values of j in the out-of-bag sample.
- compute the new out-of-bag error
- store the decrease in prediction quality, D_j^b.

Average the D_j^b on b in order to measure the importance of the variable j.
See also "Mean decrease Gini".

Other parameters

- **subsampling** : sampling bootstrap datasets of size $s < n$. Smaller variance because of more independence between the trees, but higher bias.
- number of bins in each variable (for instance, impose a maximum number of bins).

See also

- boosting
- gradient tree boosting
- XGBoost : variant of gradient boosting, very efficient on several benchmarks [Chen and Guestrin, 2016]
- can exploit parallelization
- `https://xgboost.readthedocs.io/en/latest/parameter.html`
- `https://github.com/dmlc/xgboost`

Catboost

- See also : Catboost
  `https://catboost.ai/`

References I


Breiman, L. (2001).
Random Forests.
Machine Learning, 45(1) :5–32.


Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (2017).
Classification And Regression Trees.
Routledge.


Chen, T. and Guestrin, C. (2016).
XGBoost : A scalable tree boosting system.
Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-Augu :785–794.

References II

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014).
Do we need hundreds of classifiers to solve real world classification problems ?
Journal of Machine Learning Research, 15 :3133–3181.