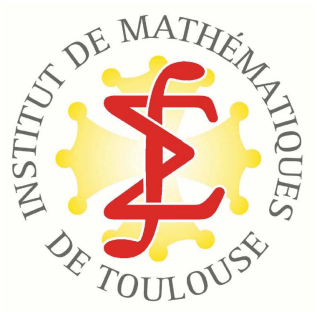


Institut de Mathématiques de Toulouse, INSA Toulouse

CART, Random Forests, Boosting

Data Mining
October 2021

Béatrice Laurent - Philippe Besse - Olivier Roustant

Outline

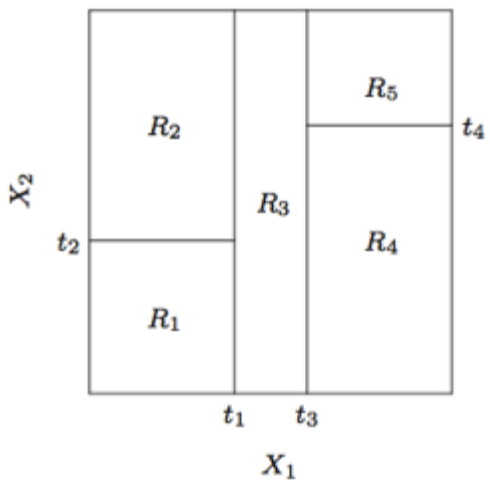
- Classification And Regression Trees (CART)
- Bagging
- Random Forests
- Boosting

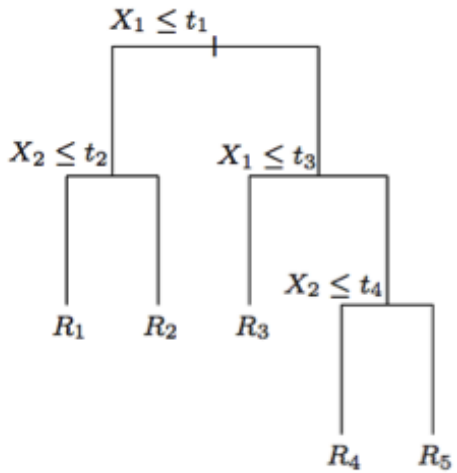
Classification And Regression Trees

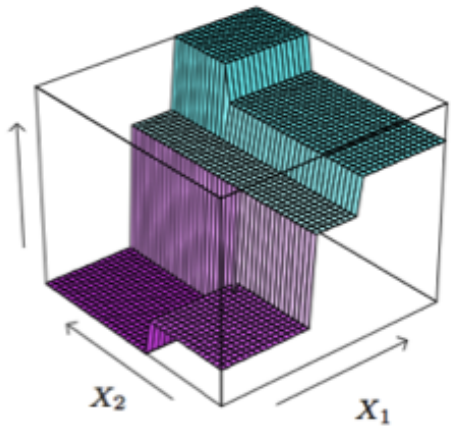
Introduction

- Classification and regression trees (CART) : Breiman et al. (1984)
- X^j explanatory variables (quantitative or qualitative)
- Y qualitative with m modalities $\{\mathcal{T}_\ell; \ell = 1 \ldots, m\}$: classification tree
- Y quantitative : regression tree
- Objective : construction of a binary decision tree easy to interpret
- No assumption on the model : non parametric procedure.

Example of binary regression tree

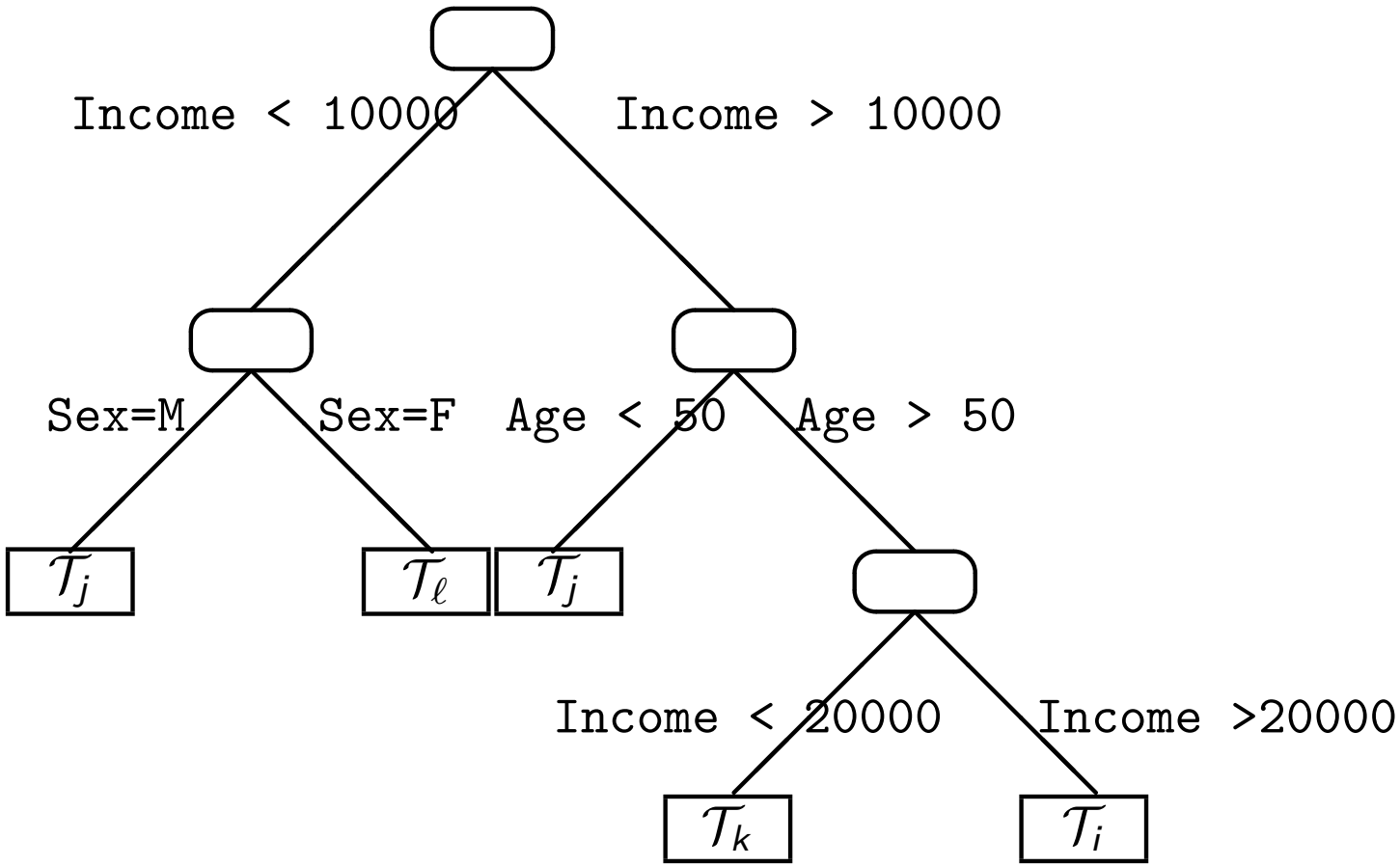






Source : Hastie, Tibshirani, Friedman (2019), "The elements of statistical learning"

Example of binary classification tree



Principles for constructing a tree

- Recursive binary split
  $\rightarrow$ Split a region in two, then split subregions in two, then ...
- Splits are defined by one variable
  $\rightarrow$ Very easy numerically : p optimizations in 1-dimension
- Clustering idea
  $\rightarrow$ Find a split that give the most homogeneous groups

Definitions

- Determine an iterative sequence of *nodes*
- *Root* : initial note : the whole sample
- *Leaf* : Terminal node
- *Node* : choice of one variable and one *division* to proceed to a dichotomie
- *Division* : threshold value or group of modalities

Rules

- We have to choose :
  1. Criterion for the "best" division among all *admissibles* ones (partition based on the values of one variable)
  2. Rules for a terminal node : *leaf*
  3. Rules to assign a leaf to a class $\mathcal{T}_\ell$ or one value for Y
- *Admissible* divisions : descendants $\neq \emptyset$
- X^j real or ordinal with c_j possible values : $(c_j - 1)$ possible divisions
- X^j nominal : $2^{(c_j-1)} - 1$ possible divisions
- Heterogeneity function D_κ of one node

Division criterion

Optimal division

- Notations
  - κ : a node
  - κ_L and κ_R the two son nodes
- The algorithm retains the division which minimizes

$$D_{\kappa_L} + D_{\kappa_R}$$

- For each node κ in the construction of the tree :

$$\max_{\{\textit{Divisions of } X^j; j=1,p\}} D_\kappa - (D_{\kappa_L} + D_{\kappa_R})$$

Stopping rule and affectation

Leaf and affectation

- A Node is a terminal node or a leaf, if it is :
  - Homogeneous
  - Number of observations below some threshold
- Affectation
  - Y quantitative : the value is the mean of the observations in the leaf
  - Y qualitative : each leaf is affected to one class $\mathcal{T}_\ell$ of Y by a majority vote.

Constructing regression trees

For a given region (node) κ with size $|\kappa|$, define the heterogeneity by :

$$D_\kappa = \sum_{i \in \kappa} (y_i - \overline{y}_\kappa)^2 = |\kappa| \frac{1}{|\kappa|} \sum_{i \in \kappa} (y_i - \overline{y}_\kappa)^2$$

Splitting procedure

For a variable x_j, and a split candidate t, define left and right subregions

$$\kappa_L(t,j) = \{x_j \leq t\}, \qquad \kappa_R(t,j) = \{x_j > t\}.$$

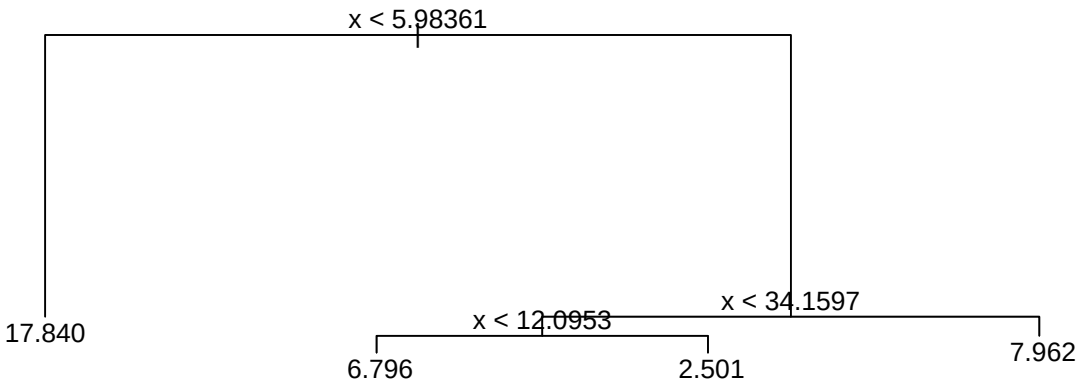
Find (j,t) in order to minimize the intra-class variance

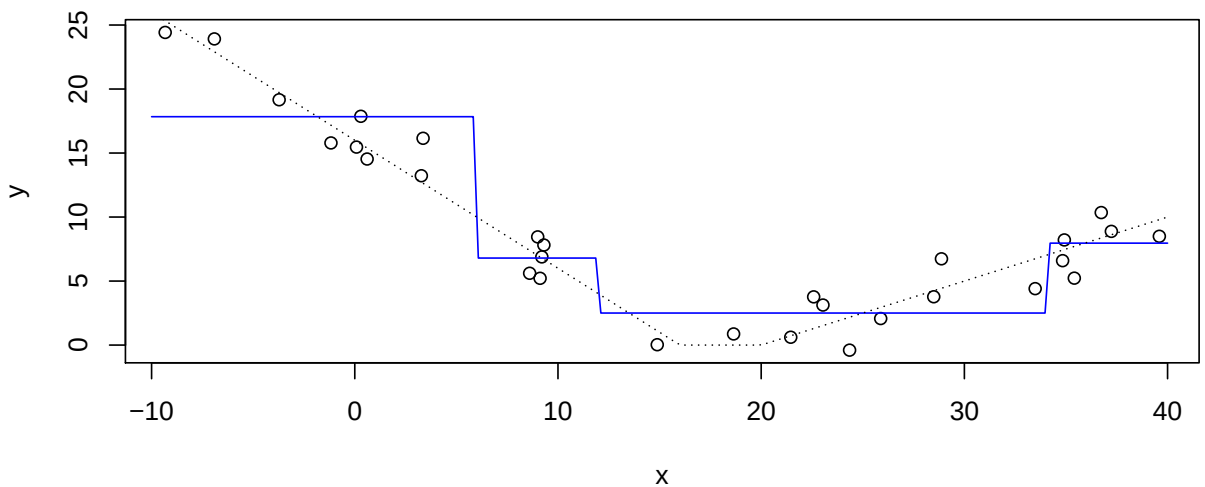
$$J(j,t) = D_{\kappa_L(t,j)} + D_{\kappa_R(t,j)},$$

or equiv. to maximize the decrease in heterogeneity (inter-class variance)

$$D_\kappa - J(j,t)$$

Illustration in 1 dimension





Constructing classification trees

This is the same procedure, with specific notions of heterogeneity

Heterogeneity measures in classification

p_κ^ℓ : proportion of the class $\mathcal{T}_\ell$ of Y in the node κ.

- Shannon Entropy

$$E_\kappa = -\sum_{\ell=1}^{m} p_\kappa^\ell \log(p_\kappa^\ell) \qquad \Rightarrow \qquad D_\kappa = -|\kappa| \sum_{\ell=1}^{m} p_\kappa^\ell \log(p_\kappa^\ell)$$

Maximal in $(\frac{1}{m}, \ldots, \frac{1}{m})$, minimal in $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$
(by continuity, we assume that $0 \log(0) = 0$)

- Gini concentration : $D_\kappa = |\kappa| \sum_{\ell=1}^{m} p_\kappa^\ell (1 - p_\kappa^\ell)$

Illustration with two classes ($m = 2$)

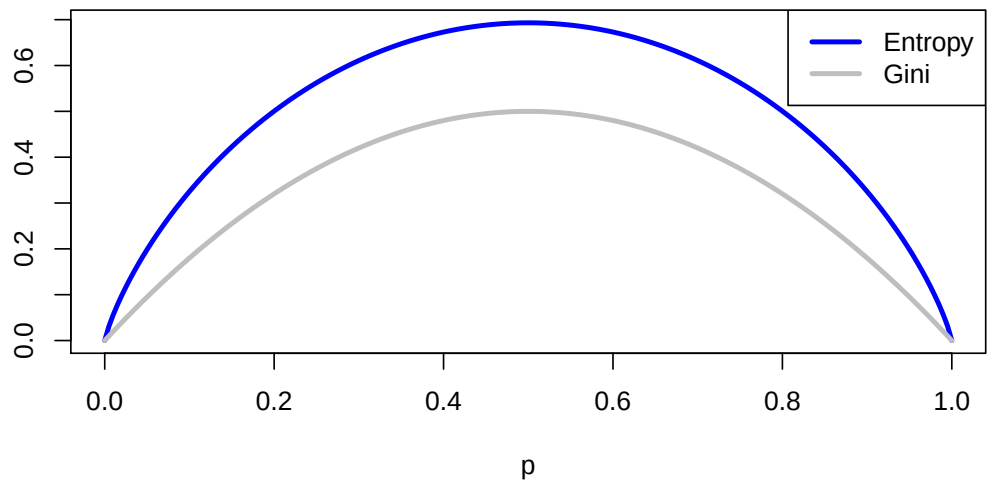
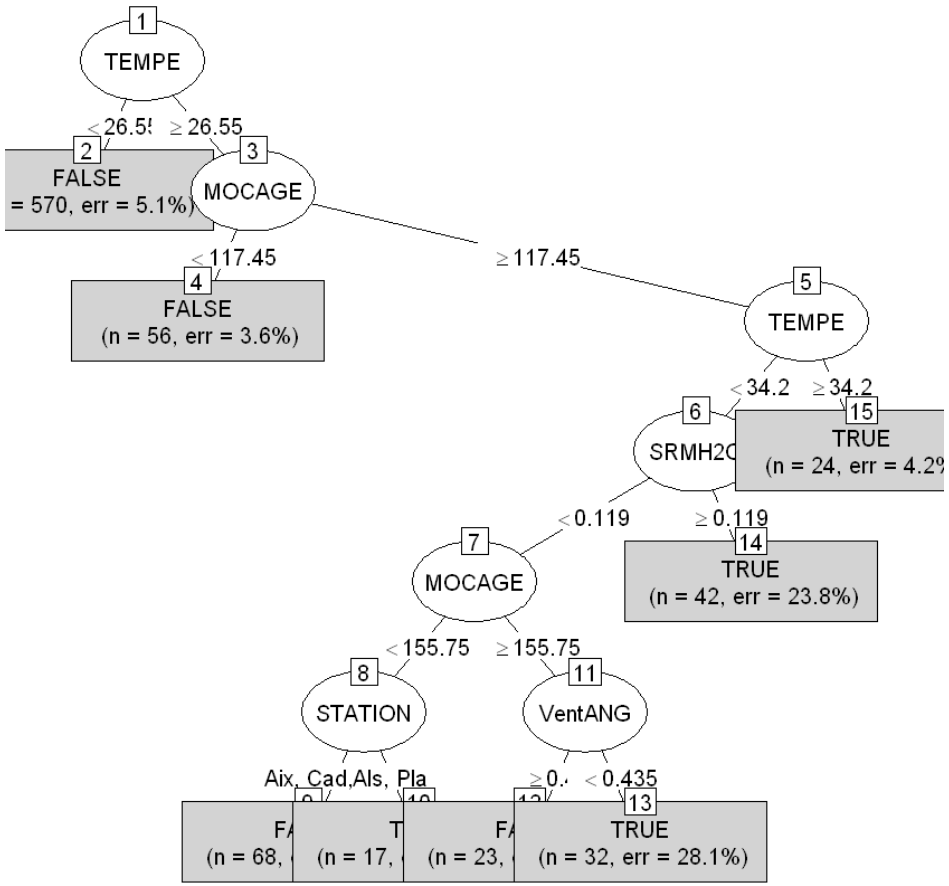


FIGURE – Heterogeneity criterions for classification. Both are minimal for $p = 0$ or $p = 1$, and maximal for $p = 1/2$.

Example for Ozone data



Ozone : Classification tree pruned by cross-validation

Stopping rule, pruning, optimal tree

- We need a tradeoff between maximal tree (overfits) and the constant tree (too rough)
- There exists a nice theory to find an optimal tree, minimizing prediction error penalized by complexity (number of leaves)
- When aggregating trees (random forest), simpler procedures are often prefered (see why after), e.g. fixing the number of leaves

Pruning and optimal tree

Pruning : notations

- We look for a parcimonious tree
- *Complexity* of a tree : K_A = numbers of leaves in A
- Adjustment error of A :

$$D(A) = \sum_{\kappa=1}^{K_A} D_\kappa$$

D_κ : heterogeneity of leaf κ

Sequence of embedded trees

- Adjustment error penalized by the complexity :

$$Crit_\gamma(A) = D(A) + \gamma \times K_A$$

- When $\gamma = 0$: $A_{\max}$ (maximal tree) minimizes $Crit_\gamma(A)$
- When γ increases, the division of A_H, for which the improvement of D is smaller than γ, is cancelled ; hence
  - two leaves are gathered (prunned)
  - there father node becomes a terminal node
  - A_K becomes A_{K-1}.
- After iteration of this process, we get a sequence of trees :

$$A_{\max} \supset A_K \supset A_{K-1} \supset \cdots A_1$$

Optimal tree

Algorithm to select the optimal tree

- Maximal tree $A_{\max}$
- Imbedded sequence $A_{\max}, A_K \dots A_1$ associated with an increasing sequence of values $\gamma_K \leq \dots \leq \gamma_1$
- V-fold cross validation error :
  for $v = 1, \dots, V$ **do**
  - Estimation of the sequence of trees associated to (γ_κ) with all the folds except v
  - Estimation of the error with the fold v.

  EndFor
- Sequence of the mean of these errors for each value of $\gamma_K, \dots, \gamma_1$
- γ_{Opt} optimal value for the tuning parameter minimizing the mean of the errors
- Tree associated to γ_{Opt} in $A_K \dots A_1$

Advantages

- Trees are easy to interpret
- Efficient algorithms to find the pruned trees
- Tolerant to missing data

$\Longrightarrow$ Success of CART for practical applications

Warnings

- Variable selection : the selected tree only depends on few explanatory variables, trees are often (wrongly) interpreted as a variable selection procedure
- High instability of the trees : not robust to the learning sample, curse of dimensionality ..
- Prediction accuracy of a tree is often poor compared to other procedures

$\Longrightarrow$ Aggregation of trees : bagging, random forests

Outline

- Classification And Regression Trees (CART)
- Bagging
- Random Forests
- Boosting

Introduction

- Combination or aggregation of models (almost) without overfitting
- Bagging is for bootstrap$^{(\star)}$ aggregating : Breiman, 1996
- Random forests : Breiman, 2001
- Allows to aggregate any modelisation method
- Efficient methods : Fernandez-Delgado et al. (2014), *Kaggle*

$^{(\star)}$ *bootstrap = sampling with replacement*

- Bagging is appropriate for unstable algorithms, with small bias and high variance (CART)

Bagging - Principle

Bootstrap AGGregatING

- Variance reduction : by aggregating independent predictions
  - Aggregation : average (regression), majority vote (classification)
- Bootstrap trick : get new data from themselves by resampling !
  - Caution : new data remain (slightly) dependent on the initial ones

Bagging - Introductive example

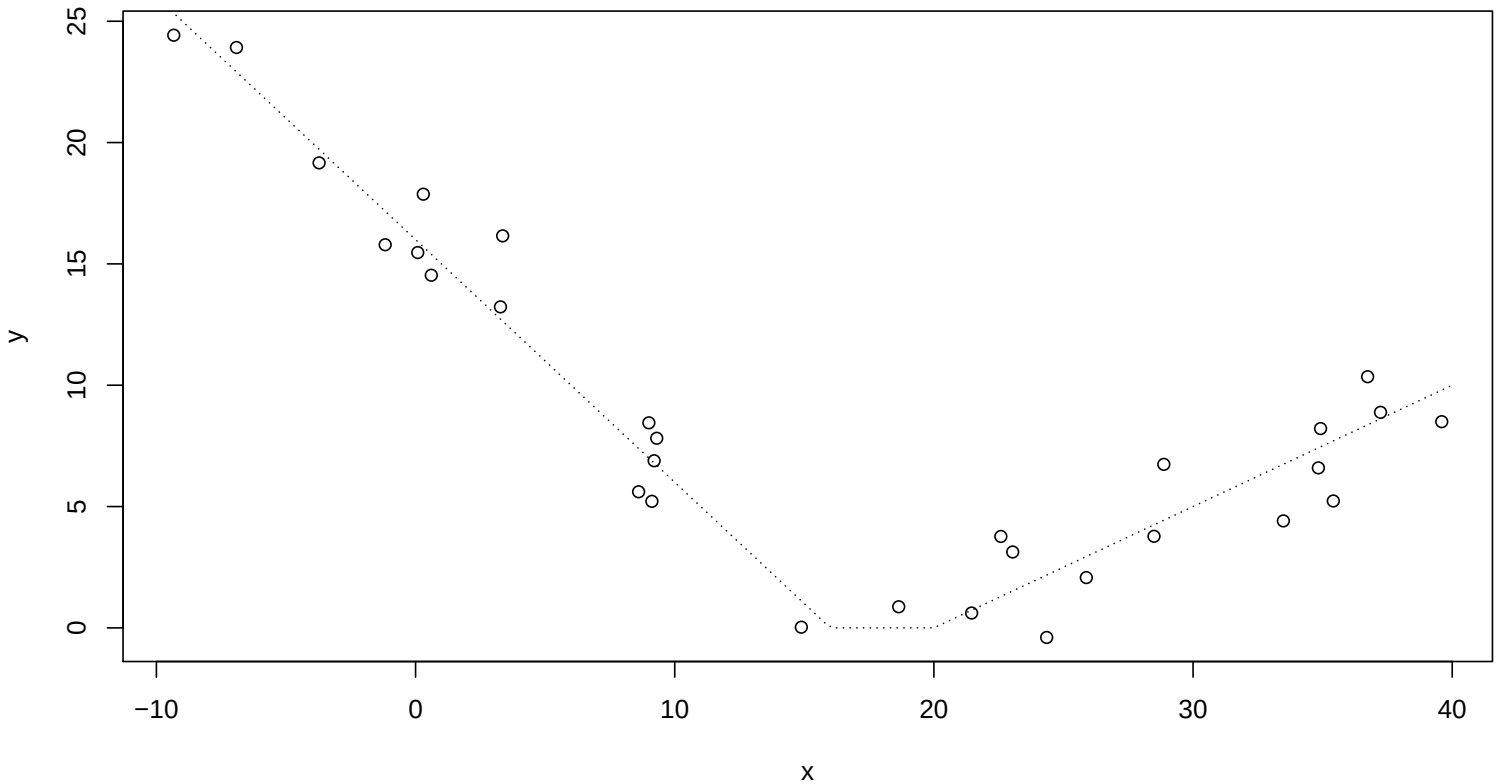


FIGURE – Original data

Bagging - Introductive example

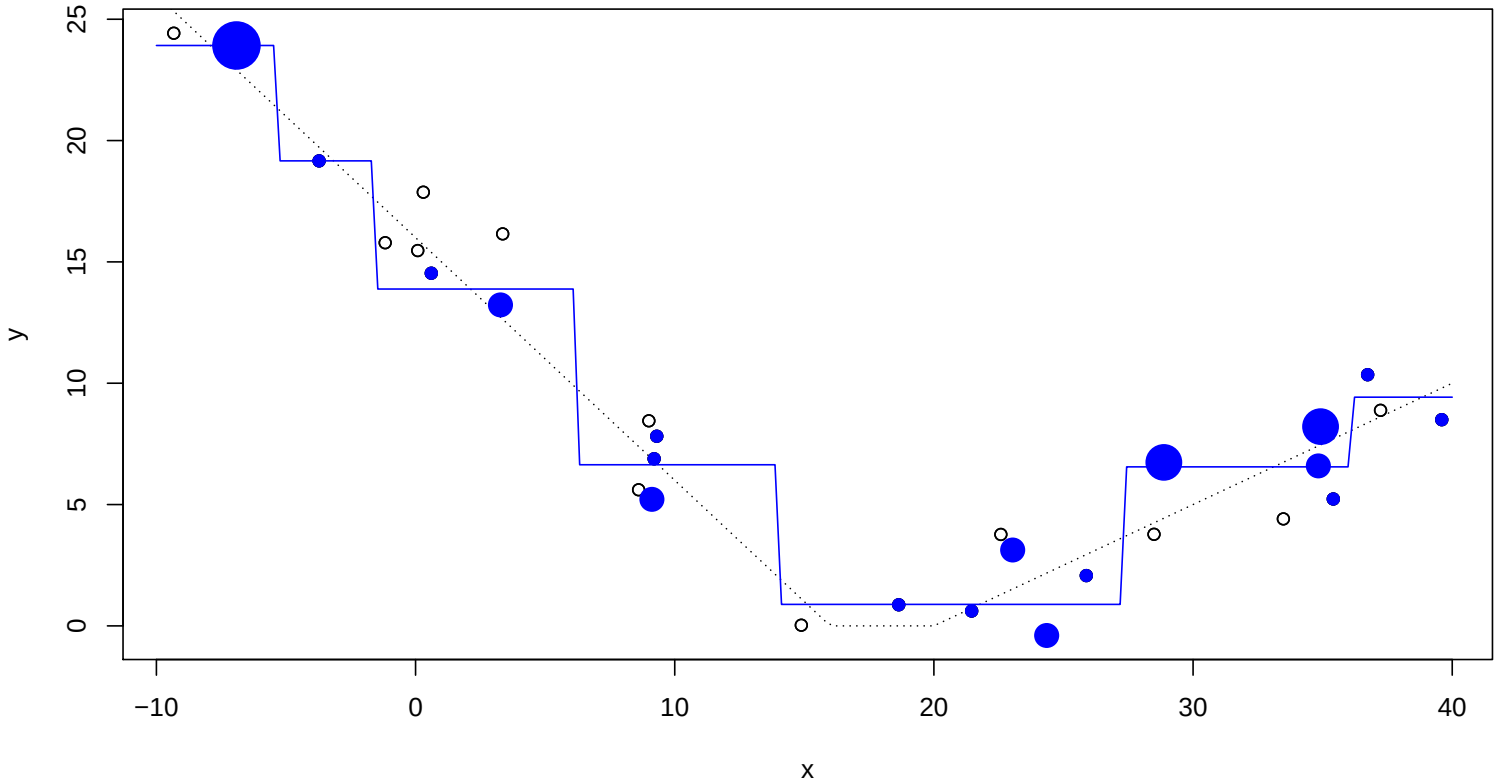


FIGURE – Bootstrap sample $n^o 1$ (in blue), and corresp. prediction with tree. The point size is proportional to the number of replicates.

Bagging - Introductive example

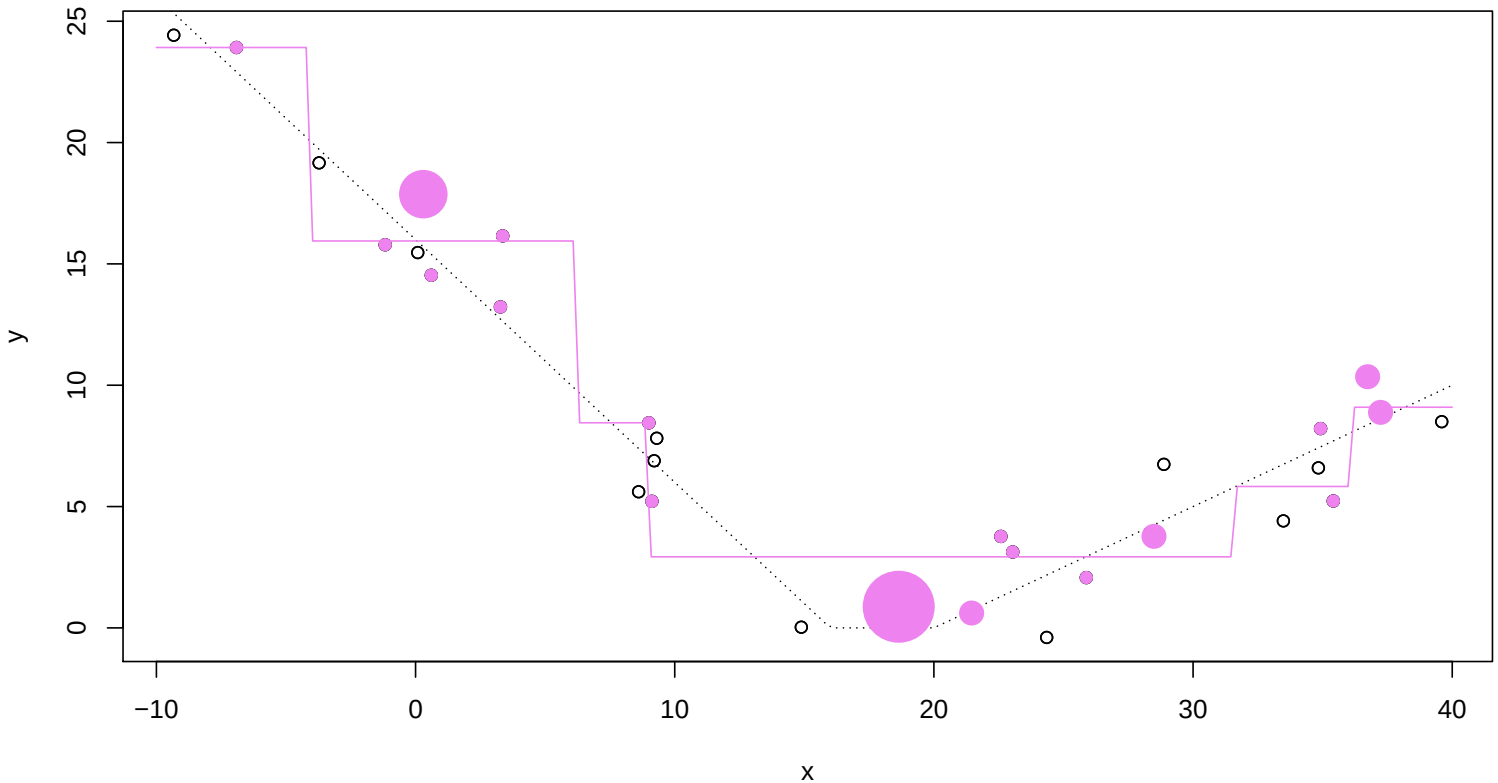


FIGURE – Bootstrap sample $n^o 2$ (in violet), and corresp. prediction with tree. The point size is proportional to the number of replicates.

Bagging - Introductive example

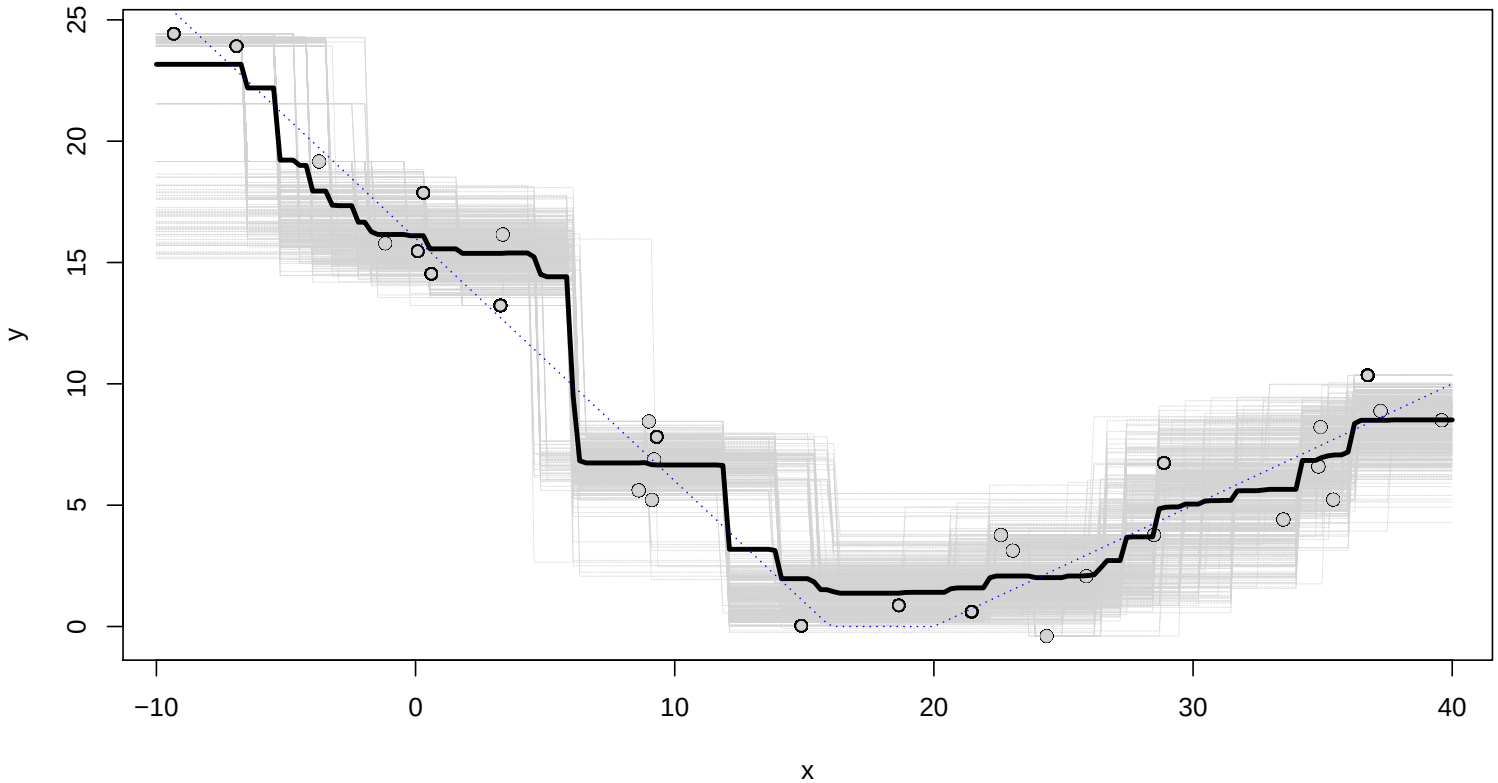


FIGURE – 500 bootstrap samples (grey), corresp. predictions with tree, and their average (bold line).

Bagging - Pause

Physical experiment !

Experiment yourself the bootstrap procedure by resampling "by hand"

Question : Choose a number between 1 and N (number of participants). What is the probability that your number does not appear in the boostrap sample ?

Bagging - Out-Of-Bag data

Out-Of-Bag (OOB) data

For each bootstrap sample :

- Let $U_1^\star, \ldots, U_N^\star$ be random variables representing the boostrapped indices. The probability that a given data z_i is not chosen is :

$$\mathbb{P}\left(z_{U_1^\star} \neq z_i, \ldots, z_{U_N^\star} \neq z_i\right) = \left(1 - \frac{1}{N}\right)^N \underset{N \to +\infty}{\longrightarrow} e^{-1} \approx 0.367$$

- The non-chosen data are called Out-Of-Bag (OOB). They can be used as a test set inside the bootstrap loop

The OOB error is obtained by averaging prediction errors over OOB data

Bagging - Out-Of-Bag data

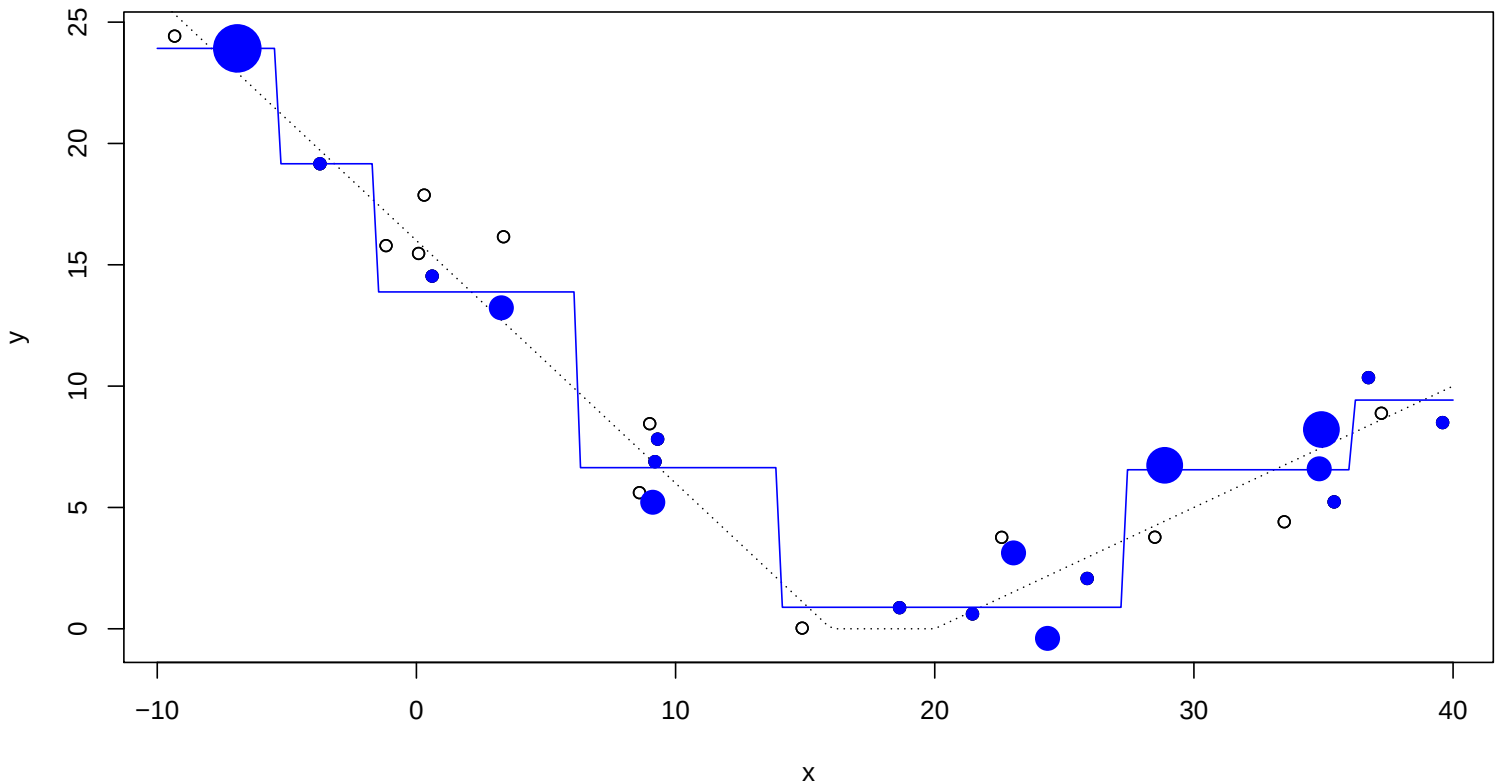


FIGURE – Residuals for the OOB bootstrap sample $n^o 1$ (red bars).

Bagging - Out-Of-Bag data

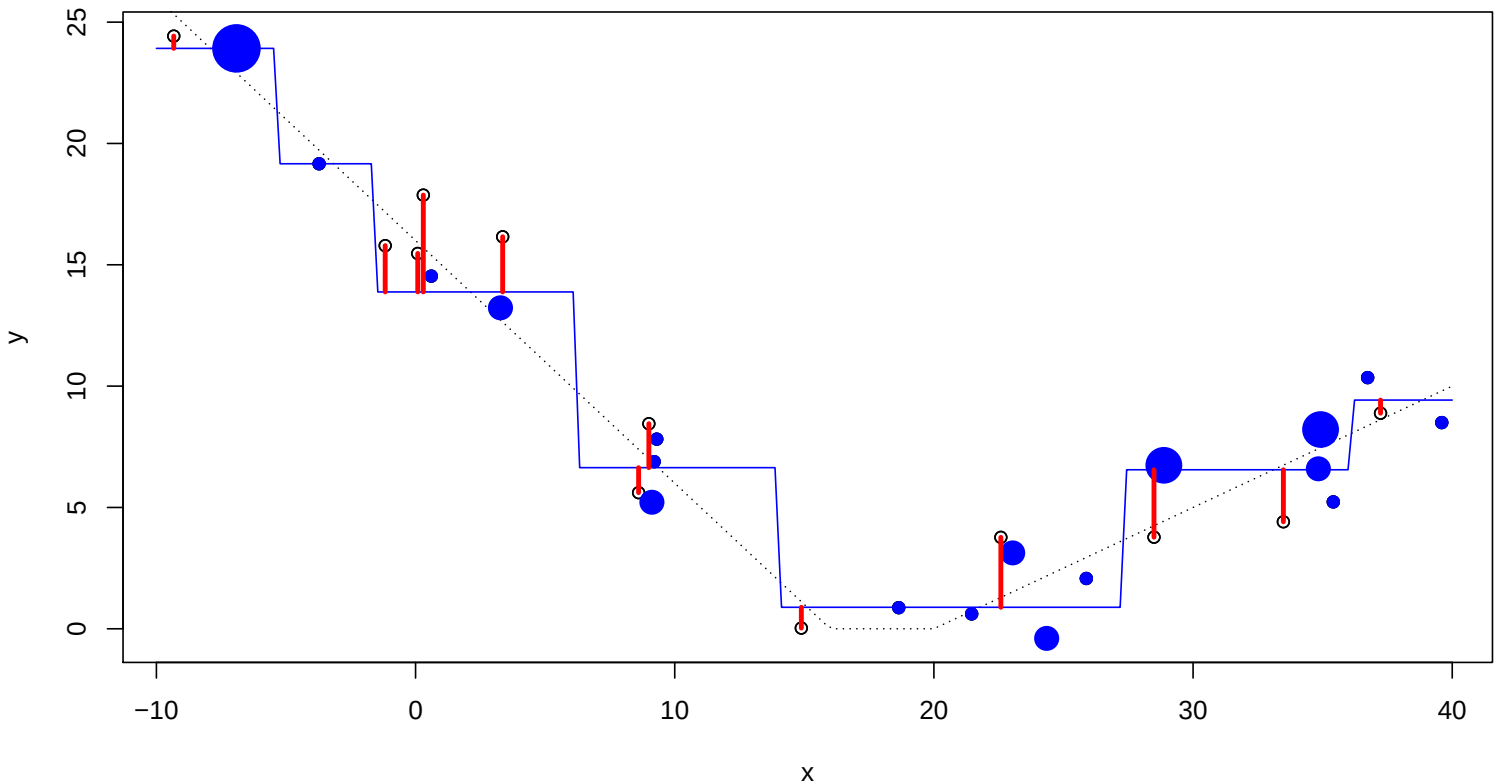


FIGURE – Residuals for the OOB bootstrap sample $n^o 1$ (red bars).

Bagging - Out-Of-Bag data

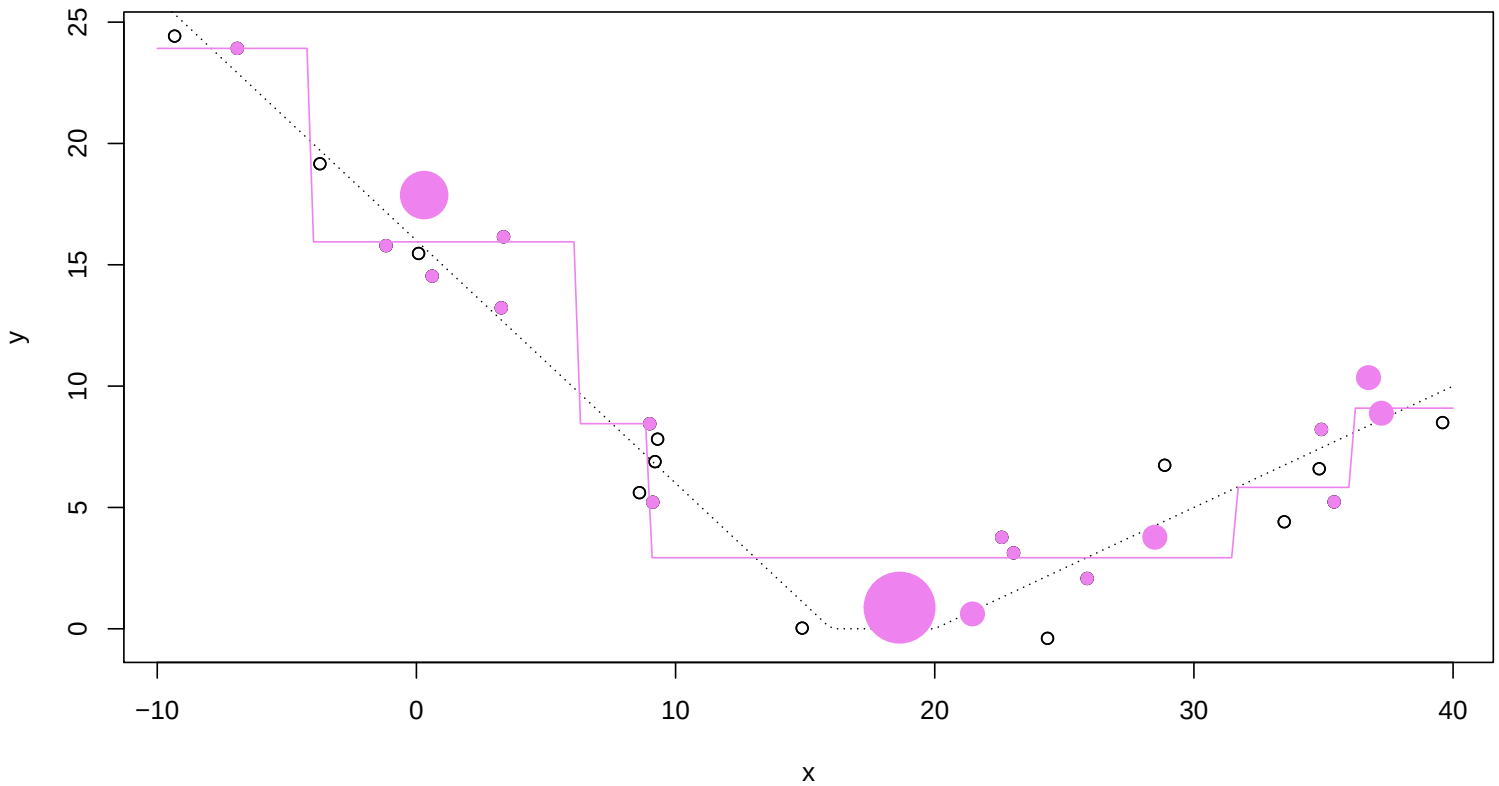


FIGURE – Residuals for the OOB bootstrap sample $n^o 2$ (red bars).

Bagging - Out-Of-Bag data

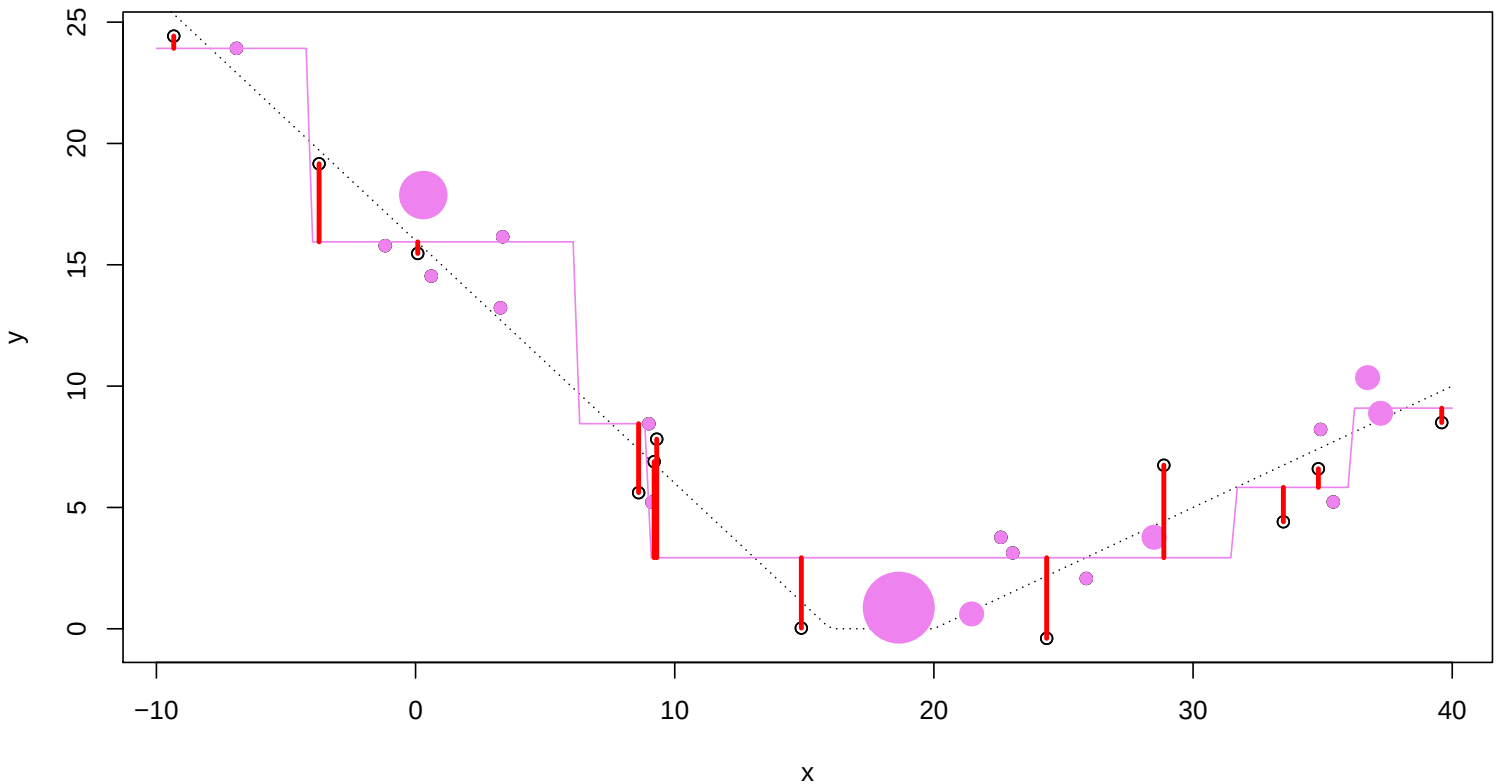


FIGURE – Residuals for the OOB bootstrap sample $n^o 2$ (red bars).

Bagging - Theory

Framework and notations

- Let Y a quantitative or qualitative variable to explain
- $X^1, \ldots, X^p$ the explanatory variables
- $f(\mathbf{x})$ a model function of $\mathbf{x} = \{x^1, \ldots, x^p\} \in \mathbb{R}^p$
- $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ learning sample with distribution F and size n
  - $\widehat{f}_{\mathbf{z}}$: predictor associated to the sample $\mathbf{z}$ with $f(.) = E_F(\widehat{f}_{\mathbf{z}})$
  - B independent samples $\{\mathbf{z}_b\}_{b=1,B}$
  - Y quantitative : $\widehat{f}_B(.) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_{\mathbf{z}_b}(.)$ (mean)
  - Y qualitative : $\widehat{f}_B(.) = \arg\max_j \text{card}\left\{b \mid \widehat{f}_{\mathbf{z}_b}(.) = j\right\}$ (majority vote)
- Principle : Take the mean of independent predictions to reduce the variance
- B independent samples replaced by B bootstrap replications

Bagging : algorithm

- Let $\mathbf{x}_0$ to point where we want to predict and
- $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ a learning sample
- **For** $b = 1$ **to** B
  - Generate a bootstrap sample $\mathbf{z}_b^*$ (with size $m_n \leq n$)
  - Estimate $\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ with the bootstrap sample
- Compute the mean estimation $\widehat{f}_B(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ or the result of the majoritary vote

- The B boostrap samples are built on the same learning sample $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$:
  $\Longrightarrow$ the estimators $\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ are not independent
- Regression case : If $\mathrm{Corr}(\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0), \widehat{f}_{\mathbf{z}_{b'}}(\mathbf{x}_0)) = \rho(x_0)$,

$$\begin{aligned}
\mathrm{E}(\widehat{f}_B(\mathbf{x}_0)) &= f(\mathbf{x}_0) \\
\mathrm{Var}(\widehat{f}_B(\mathbf{x}_0)) &= \rho(x_0)\mathrm{Var}(\widehat{f}_b(\mathbf{x}_0)) + \frac{(1-\rho(x_0))}{B}\mathrm{Var}(\widehat{f}_b(\mathbf{x}_0)) \\
&\to \rho(x_0)\mathrm{Var}(\widehat{f}_b(\mathbf{x}_0)) \text{ as } B \to +\infty
\end{aligned}$$

$\Longrightarrow$ Importance to find low correlated predictors $(\widehat{f}_b(\mathbf{x}_0))_{1 \leq b \leq B}$.
$\hookrightarrow$ **Random forests**

Bagging : practical use

- *Bootstrap out-of-bag* estimation of the prediction error : control of the quality and avoid overfitting
- CART to built a sequence of binary trees
- Three pruning strategies are possible :
  1. keep a whole tree for each of the samples
  2. tree with at most q leaves
  3. whole tree pruned by cross-validation
- First strategy compromise between computations and prediction accuracy : small bias for each tree and reduced variance by aggregation

Bagging : limitations

- Computation time and control of the error
- Storage of all the models to aggregate
- Black box model

Outline

- Classification And Regression Trees (CART)
- Bagging
- Random Forests
- Boosting

Random forests

Random forests : principle

- Random forests means initially any aggregation method for classification or regression trees
- Now, it refers to the **Random input** method introduced by Breiman and Cutler (2005)
  `http://www.stat.berkeley.edu/users/breiman/RandomForests/`
- Improvement of the bagging of binary trees
- Variance of B correlated variables : $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$
- Add a randomisation to get more independent trees
- Random choice of variables
- Interesting in high dimension

Random forests : algorithm

- Let $\mathbf{x}_0$ the point where we want to predict,
  $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ a learning sample.
- **for** $b = 1$ **to** B
  - Generate a bootstrap sample $\mathbf{z}_b^*$
  - Estimate a tree with randomization of the variables : for each node, random sample of $m < p$ predictors to built the subdivision
- Compute the mean estimation $\widehat{f}_B(\mathbf{x}_0) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$
  or the majority vote

Random forests : utilisation

- Pruning : tree with q leaves, or complete tree,
- Random selection of m predictors : $m = \sqrt{p}$ for classification, $\frac{p}{3}$ for regression.
- Small m increases the variability of each tree but reduces the correlation between them : Each model is less accurate but aggregation is performing
- Iterative computation of the out-of-bag error.
- Good accuracy, easily implementable, parallelisable but not easy to interpret

To help interpretation

- Index of importance for each variable

Interpretation - Variable importance

How can we quantifiy the importance of a variable X^j in random forest ?

Decrease in heterogeneity

Average the decrease of heterogeneity when X^j is chosen as a split.

- Mean Decrease Accuracy
- Mean Decrease Gini

Permutation of variables

Compute the OOB error for the subsample of OOB data.
Compare with the OOB error when permuting at random the values of the variable X^j in the learning sample (but keeping the output Y unchanged).

Example on ozone data

```
> library(randomForest)
> rf.reg <- randomForest(O3obs ., data = datappr, xtest = datestr[, -2],
    ytest = datestr[, "O3obs"], ntree = 500, do.trace = 50, importance = TRUE)
```

Tree	Out -of - bag MSE	%Var(y)	Test set MSE	%Var(y)
50	697.9	40.77	568.5	36.75
100	689.5	40.28	555.9	35.93
150	683.8	39.95	563.2	36.41
200	685.4	40.04	561	36.27
250	678.2	39.62	564.2	36.47
300	675.1	39.44	569.2	36.79
350	676.8	39.54	572.8	37.02
400	674.3	39.39	571.4	36.93
450	673.9	39.37	571.5	36.94
500	674.3	39.39	569.6	36.82

```
> round(importance(rf.reg), 2)

          %IncMSE    IncNodePurity
JOUR        1.98         11011.79
MOCAGE     41.46        388657.27
TEMPE      51.73        409018.57
STATION    21.73         75350.42
VentMOD    12.95         91387.20
VentANG    18.81        124908.37
SRMH2O     16.76        114463.05
LNO2        7.73         84152.34
LNO        10.04         74387.32
```

Details (from R help file of function `importance`)

"The first measure [%IncMSE] is computed from permuting OOB data : For each tree, the prediction error on the out-of-bag portion of the data is recorded (error rate for classification, MSE for regression). Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences.

The second measure [IncNodePurity] is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares."

rf.reg

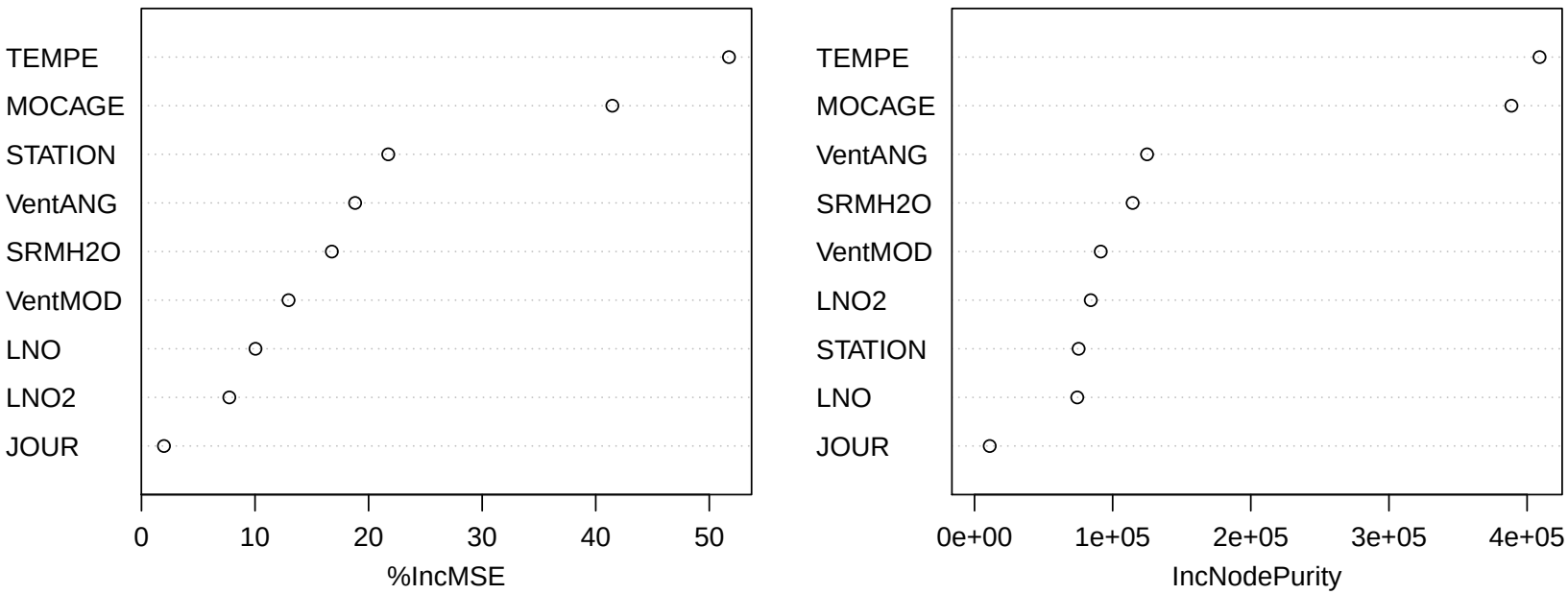


FIGURE – Variable importance plot, returned by the R function `importance`

Other possible applications

- Proximity or similarity between observations
- Anomaly detection with `IsolationForest`
- Imputation of missing data with `missForest`
- Survival analysis with `survival forest`
- ...

Outline

- Classification And Regression Trees (CART)
- Bagging , Random Forests
- Boosting

Boosting principle

Boosting : principle

- Improve the performances of a weak classifier (Schapire, 1990 ; Freund and Schapire, 1996)
- AdaBoost (Adaptative boosting) prediction of a binary variable
- Reduction of the variance but also the bias
- Aggregation of a family of recurrent models : *Each model is an adaptive version of the previous one giving more weight, in the next estimate, to the badly adjusted observations*
- Variants : type of variable to predict (binary, k classes, real), loss fonction

Basic algorithm

AdaBoost

- AdaBoost is a boosting method to combine several binary classifiers $f_1, \ldots, f_k$ with values in $\{-1, 1\}$.
- $\mathbf{z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ a learning sample, $y_i \in \{-1, 1\}$.
- The aim is to minimize the empirical risk for the exponential loss function over linear combination of the classifiers

$$\hat{f} = \underset{f \in \text{span}(f_1, \ldots, f_k)}{\text{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \exp(-y_i f(\mathbf{x}_i)) \right\}.$$

AdaBoost

- To approximate the solution, Adaboost computes a sequence of functions $\hat{f}_m$ for $m = 0, \ldots M$ with

$$\begin{aligned} \hat{f}_0 &= 0 \\ \hat{f}_m &= \hat{f}_{m-1} + \beta_m f_{j_m} \end{aligned}$$

where (β_m, j_m) minimizes the empirical risk

$$\underset{\beta \in \mathbb{R}, j=1,\ldots,p}{\operatorname{argmin}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \exp(-y_i(\hat{f}_{m-1}(\mathbf{x}_i) + \beta f_j(\mathbf{x}_i))) \right\}.$$

- The final classification rule is given by

$$\hat{f} = \operatorname{sign}(\hat{f}_M).$$

AdaBoost

- We denote
$$w_i^{(m)} = \frac{1}{n} \exp(-y_i \hat{f}_{m-1}(\mathbf{x}_i))$$
- We assume that for all $j = 1, \ldots, k$,
$$\widehat{\mathcal{E}_m}(j) = \frac{\sum_{i=1}^n w_i^{(m)} \mathbb{1}_{f_j(\mathbf{x}_i) \neq y_i}}{\sum_{i=1}^n w_i^{(m)}} \in]0, 1[.$$
- Then, we have :
$$j_m = \underset{j=1,\ldots,k}{\operatorname{argmin}} \widehat{\mathcal{E}_m}(j),$$
and
$$\beta_m = \frac{1}{2} \log \left(\frac{1 - \widehat{\mathcal{E}_m}(j_m)}{\widehat{\mathcal{E}_m}(j_m)} \right).$$

AdaBoost

AdaBoost algorithm

- $w_i^{(1)} = 1/n$ for $i = 1, \ldots, n$.
- For $m = 1, \ldots, M$

$$j_m = \underset{j=1,\ldots,p}{\operatorname{argmin}} \widehat{\mathcal{E}_m}(j), \beta_m = \frac{1}{2} \log \left(\frac{1 - \widehat{\mathcal{E}_m}(j_m)}{\widehat{\mathcal{E}_m}(j_m)} \right).$$

$$\begin{aligned} w_i^{(m+1)} &= w_i^{(m)} \exp(-y_i \beta_m f_{j_m}(\mathbf{x}_i)) \text{ for } i = 1, \ldots, n \\ &= \frac{1}{n} \exp(-y_i \hat{f}_{m-1}(\mathbf{x}_i)) \exp(-y_i \beta_m f_{j_m}(\mathbf{x}_i)) \\ &= \frac{1}{n} \exp(-y_i \hat{f}_m(\mathbf{x}_i)) \end{aligned}$$

- $\hat{f}_M(x) = \sum_{m=1}^{M} \beta_m f_{j_m}(x)$.
- $\hat{f} = \text{sign}(\hat{f}_M)$.

Gradient Boosting Models

GBM : Principle 1

- Gradient Boosting Models (Friedman, 2002-2009)
- In the case of a differentiable loss function
- Principle :
  - Construct a sequence of models in such a way that at each step, each model added to the linear combination, appears as a step towards a better solution
  - This step is done in the direction of the gradient of the loss function approximated by a regression tree

GBM : Principle 2

- Previous Adaptative model (AdaBoost) :

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \beta_m f_{j_m}(\boldsymbol{x})$$

- Transformed in a gradient descent

$$f_m(\boldsymbol{x}_i) = f_{m-1}(\boldsymbol{x}_i) - \gamma_m \frac{\partial l(y_i, f_{m-1}(\boldsymbol{x}_i))}{\partial f_{m-1}(\boldsymbol{x}_i)}.$$

- Search for a best step in the descent γ :

$$min_\gamma \sum_{i=1}^{n} \left[l\left(y_i, f_{m-1}(\boldsymbol{x}_i) - \gamma \frac{\partial l(y_i, f_{m-1}(\boldsymbol{x}_i))}{\partial f_{m-1}(\boldsymbol{x}_i)} \right) \right].$$

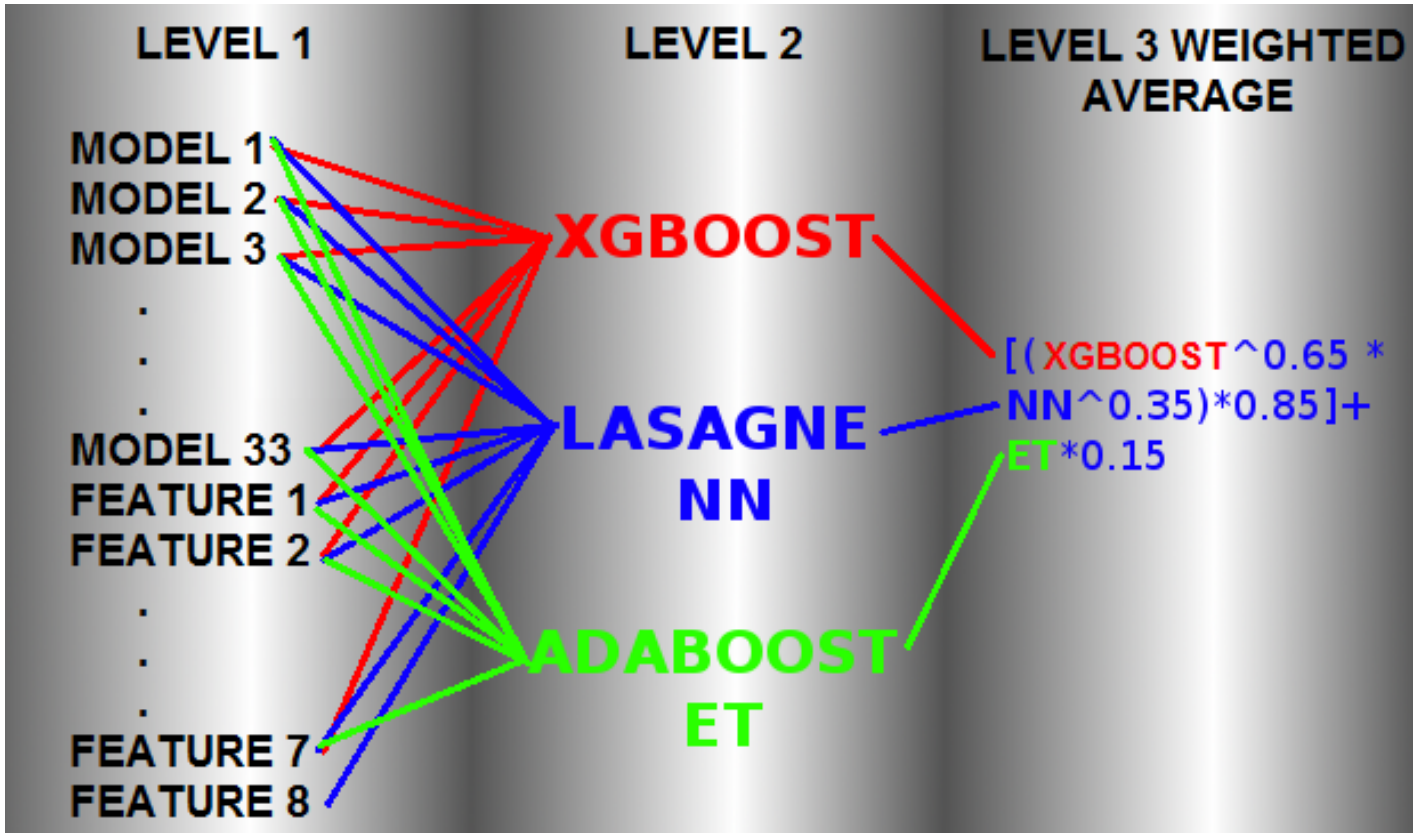
GBM in regression : algorithm

- Let $\mathbf{x}_0$ the point where we want to predict
- Initialize $\widehat{f}_0 = \arg\min_\gamma \sum_{i=1}^n l(y_i, \gamma)$
- **For** $m = 1$ **to** M
  - Compute $r_{m,i} = -\left[\frac{\partial l(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f=f_{m-1}}$; $i = 1, \ldots, n$
  - Adjust a regression tree δ_m to $(\mathbf{x}_i, r_{m,i})$
  - Calculate $\gamma_m = \arg min_\gamma \sum_{i=1}^n l(y_i, f_{m-1}(\boldsymbol{x}_i) + \gamma \delta_m(\boldsymbol{x}_i))$
  - Update : $\widehat{f}_m(\boldsymbol{x}) = \widehat{f}_{m-1}(\boldsymbol{x}) + \gamma_m \delta_m(\boldsymbol{x})$
- Result : $\widehat{f}_M(\boldsymbol{x}_0)$

Extreme gradient boosting

`XGBoost` : motivation

- Algorithm introduced by Chen et Guestrin (2016)
- Additional Penalization to control overfitting
- Problem : number of parameters to optimize
- Implementation tips for parallelisation
- Environments : R (`caret`), Python, Julia, GPU, *Amazon Web Service*, Spark...
- Winning solutions for *Kaggle* competitions



Kaggle : Identify people who have a high degree of Psychopathy based on Twitter usage

XGBoost : penalization

- Loss function L penalized version of l

$$L(f) = \sum_{i=1}^{n} l(f(\boldsymbol{x}_i), y_i) + \sum_{m=1}^{M} \Omega(\delta_m)$$

$$\Omega(\delta) = \alpha|\delta| + \beta||\boldsymbol{w}||^2$$

- $|\delta|$ number of leaves in the tree δ
- $\boldsymbol{w}$ vector of values assigned to each leaf
- Ω Mix of l_1 and l_2 penalization

References

- L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen (1984). *Classification and regression trees*. Chapman et Hall. CRC Press, Boca Raton.
- Giraud C. (2015) *Introduction to High-Dimensional Statistics* Vol. 139 of Monographs on Statistics and Applied Probability. CRC Press, Boca Raton, FL.
- Hastie, T. and Tibshirani, R. and Friedman, J, (2009), *The elements of statistical learning : data mining, inference, and prediction*, Springer.
- R.E. Shapire and Y. Freund *Boosting : Fundation and Algorithms* MIT Press, 2012