Understanding Random Forests

From Theory to Practice

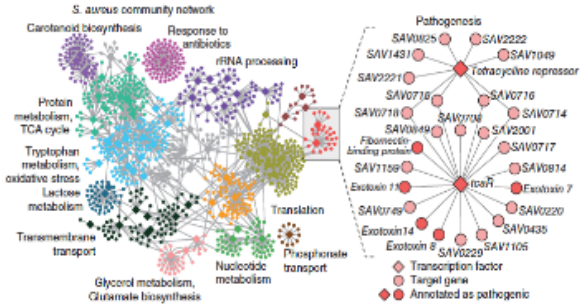
Gilles Louppe

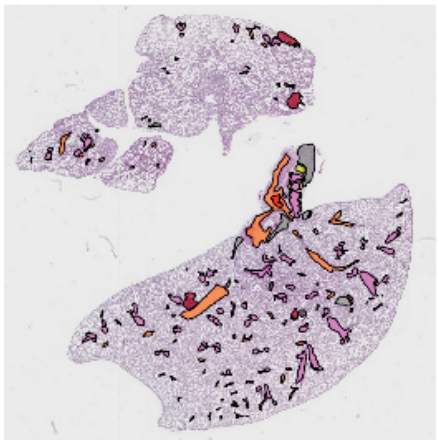
Université de Liège, Belgium

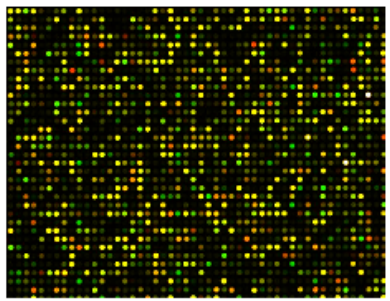
October 9, 2014

Motivation

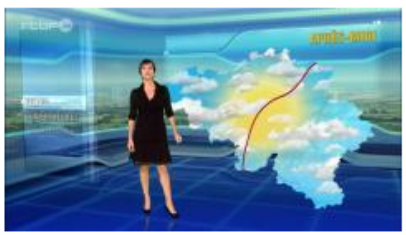

CERN


S. aureus community network
Carotenoid biosynthesis
Response to antibiotics
rRNA processing
Protein metabolism, TCA cycle
Tryptophan metabolism, oxidative stress
Lactose metabolism
Translation
Transmembrane transport
Nucleotide metabolism
Phosphonate transport
Glycerol metabolism, Glutamate biosynthesis
Pathogenesis
SAV1625
SAV2222
SAV1431
SAV1049
SAV2221
Tetracycline repressor
SAV0716
SAV0714
SAV0718
SAV2001
SAV0717
SAV0814
SAV1159
Exotoxin 7
SAV0220
SAV0435
Exotoxin 14
Exotoxin 8
SAV1105
SAV0229
Transcription factor
Target gene
Annotated as pathogenic






XBOX 360




Google
Google Search
I'm Feeling Lucky

Objective

From a set of **measurements**,

learn a **model**

to predict and understand **a phenomenon**.

Running example

From **physicochemical properties** (alcohol, acidity, sulphates, ...),

learn a **model**

to predict **wine taste preferences** (from 0 to 10).

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, *Modeling wine preferences by data mining from physicochemical properties*, 2009.

Outline

Supervised learning

- The inputs are random variables $X = X_1, \ldots, X_p$;
- The output is a random variable Y.

- Data comes as a finite learning set

$$\mathcal{L} = \{(\mathbf{x}_i, y_i) | i = 0, \ldots, N-1\},$$

where $\mathbf{x}_i \in \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_p$ and $y_i \in \mathcal{Y}$ are randomly drawn from $P_{X,Y}$.

E.g., $(\mathbf{x}_i, y_i) = ((\text{color} = \text{red}, \text{alcohol} = 12, \ldots), \text{score} = 6)$

- The goal is to find a model $\varphi_{\mathcal{L}} : \mathcal{X} \mapsto \mathcal{Y}$ minimizing

$$Err(\varphi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\{L(Y, \varphi_{\mathcal{L}}(X))\}.$$

Performance evaluation

Classification

- Symbolic output (e.g., $\mathcal{Y} = \{\text{yes}, \text{no}\}$)
- Zero-one loss

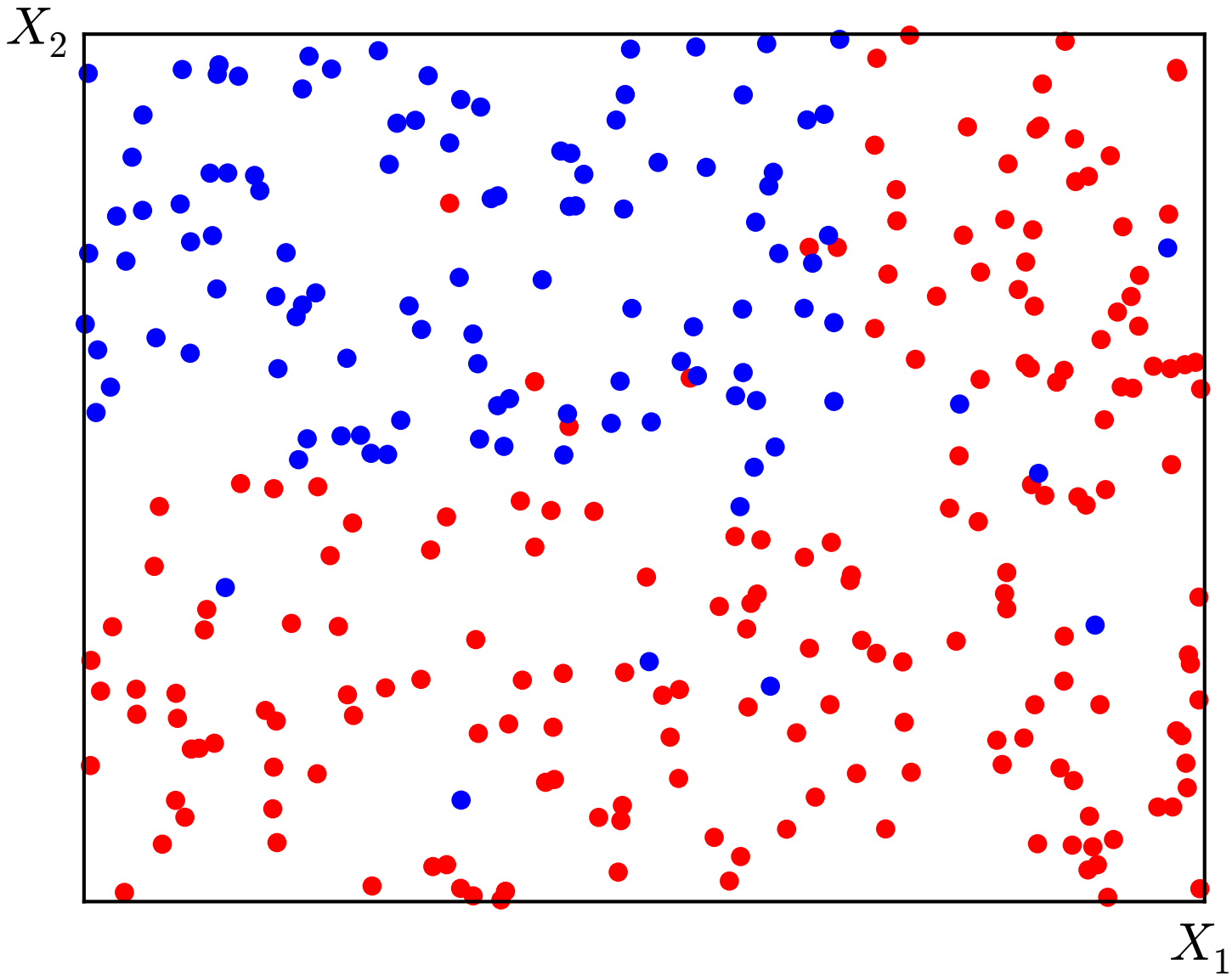
$$L(Y, \varphi_{\mathcal{L}}(X)) = 1(Y \neq \varphi_{\mathcal{L}}(X))$$

Regression

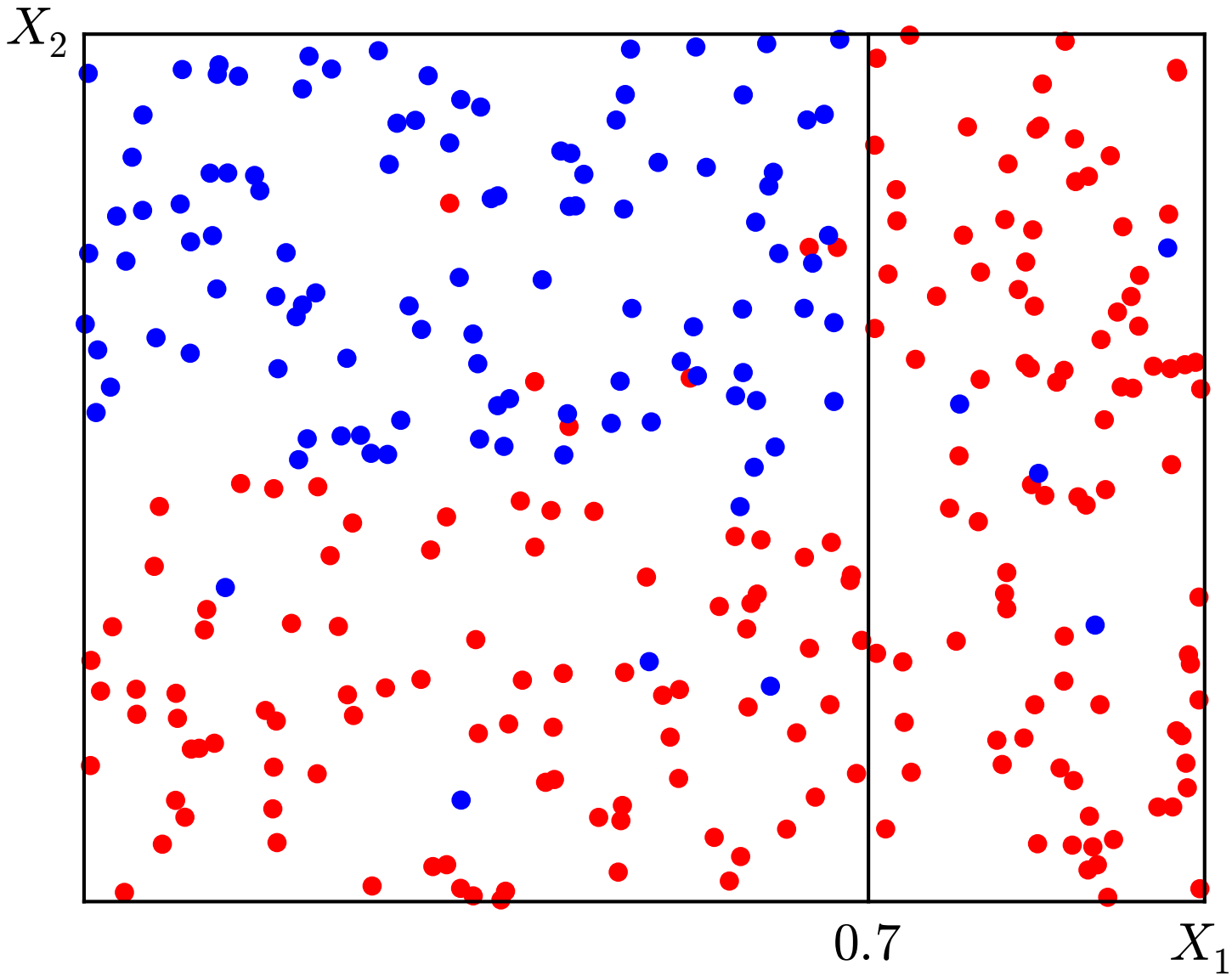
- Numerical output (e.g., $\mathcal{Y} = \mathbb{R}$)
- Squared error loss

$$L(Y, \varphi_{\mathcal{L}}(X)) = (Y - \varphi_{\mathcal{L}}(X))^2$$

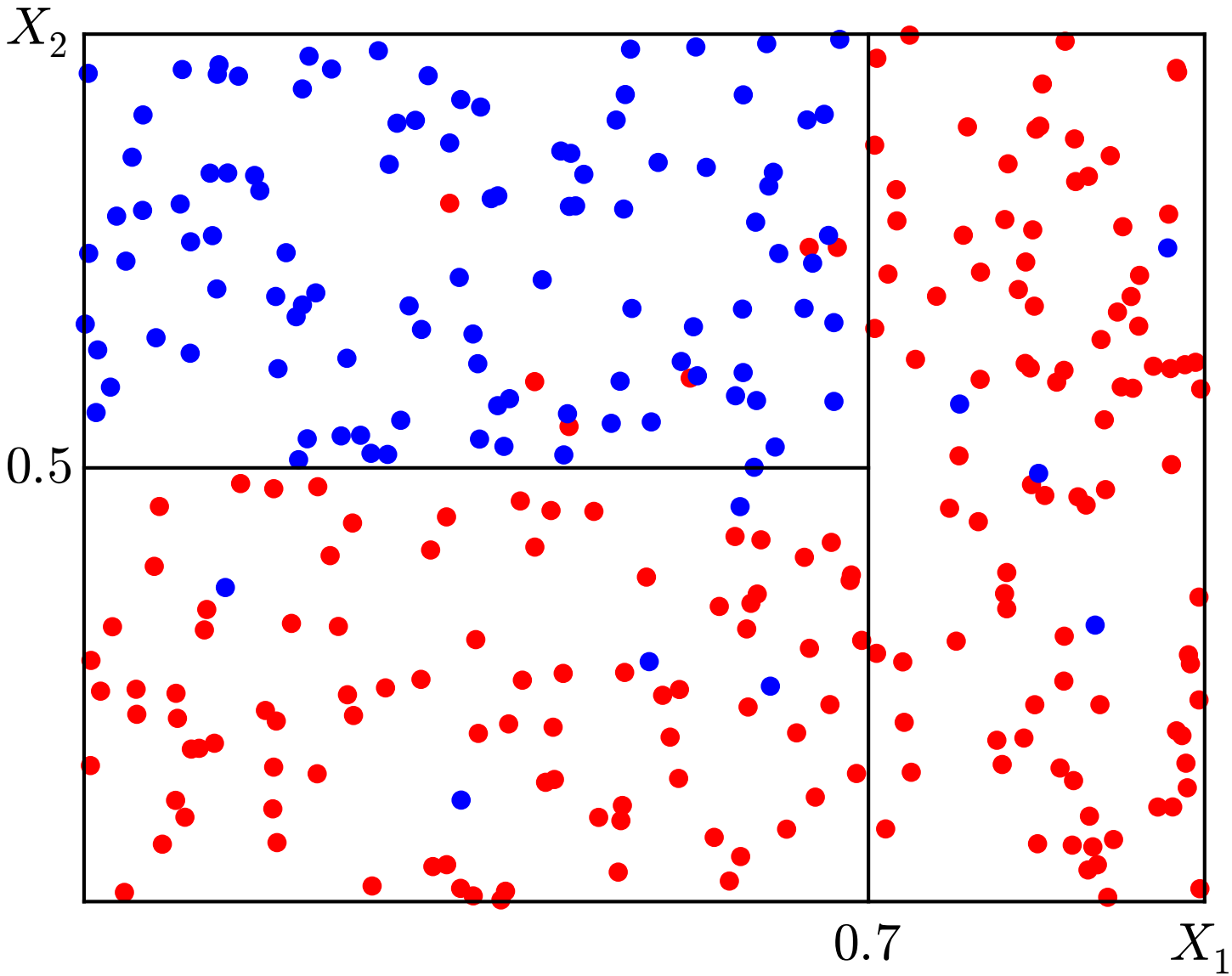
Divide and conquer



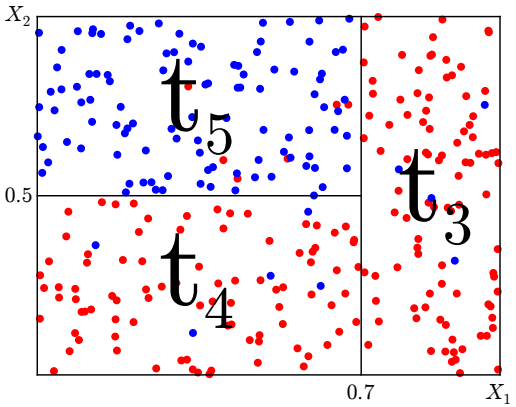
Divide and conquer

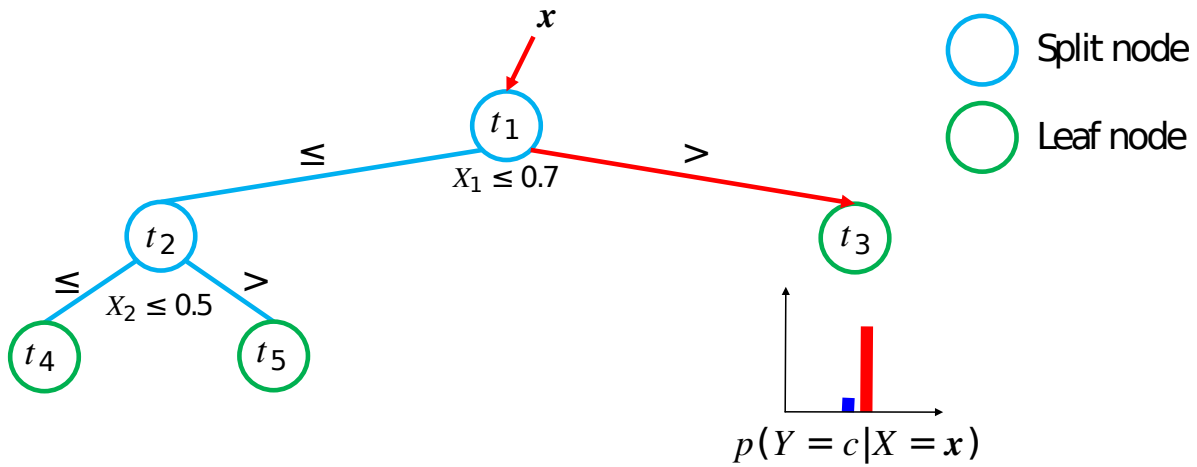


Divide and conquer



Decision trees





$t \in \varphi$: nodes of the tree φ

X_t : split variable at t

$v_t \in \mathbb{R}$: split threshold at t

$\varphi(\mathbf{x}) = \arg\max_{c \in \mathcal{Y}} p(Y = c|X = \mathbf{x})$

Learning from data (CART)

function BuildDecisionTree($\mathcal{L}$)

Create node t from the learning sample $\mathcal{L}_t = \mathcal{L}$

if the stopping criterion is met for t **then**

$\widehat{y}_t =$ some constant value

else

Find the split on $\mathcal{L}_t$ that maximizes impurity decrease

$$s^* = \arg\max_{s \in Q} \Delta i(s, t)$$

Partition $\mathcal{L}_t$ into $\mathcal{L}_{t_L} \cup \mathcal{L}_{t_R}$ according to s^*

$t_L =$ BuildDecisionTree($\mathcal{L}_L$)

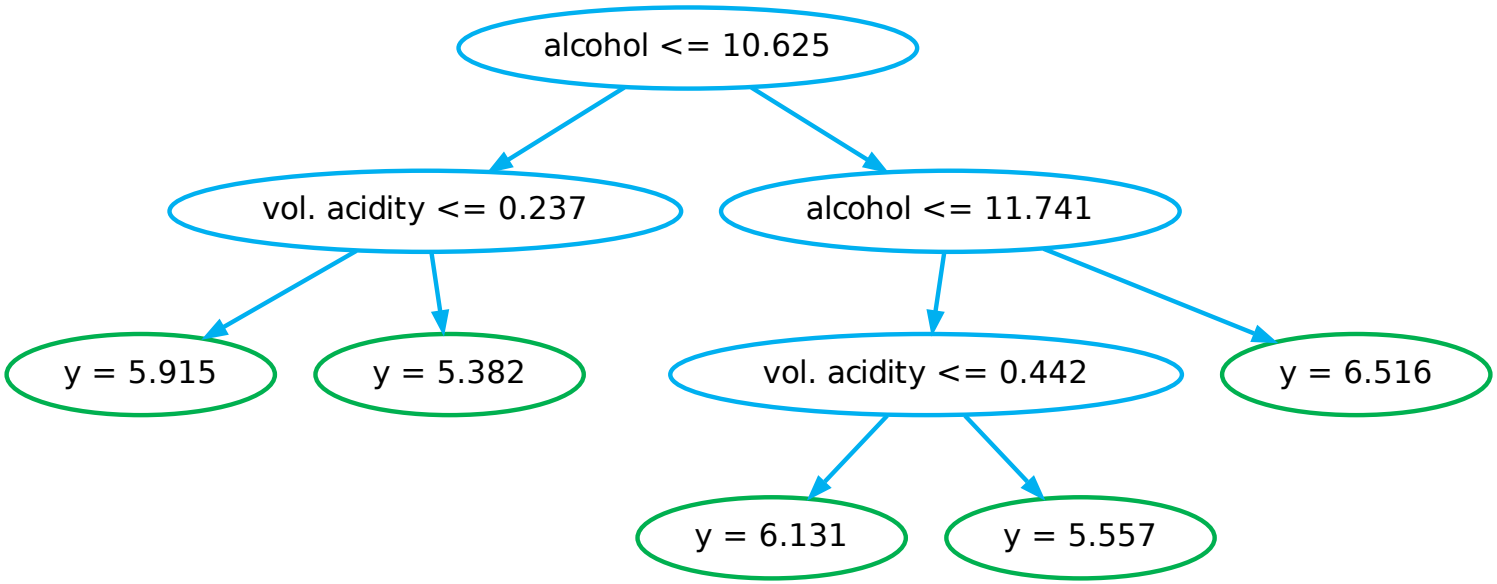
$t_R =$ BuildDecisionTree($\mathcal{L}_R$)

end if

return t

end function

Back to our example



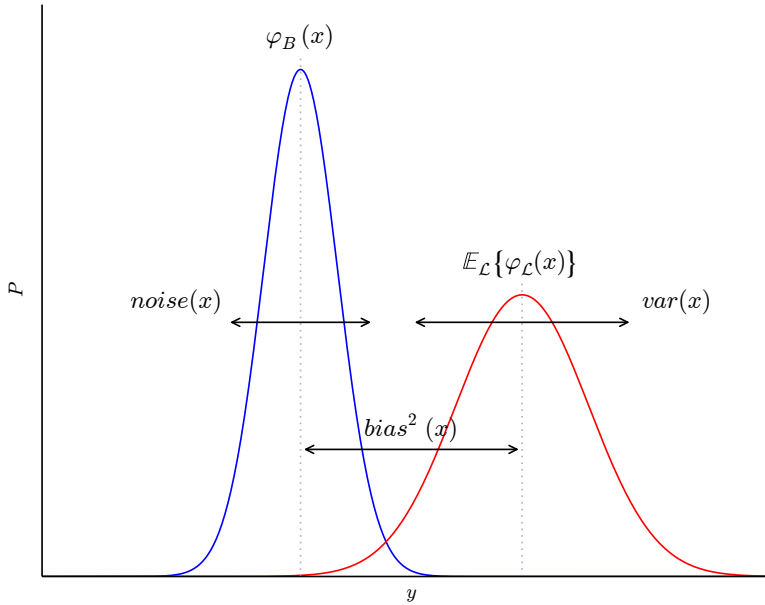
Bias-variance decomposition

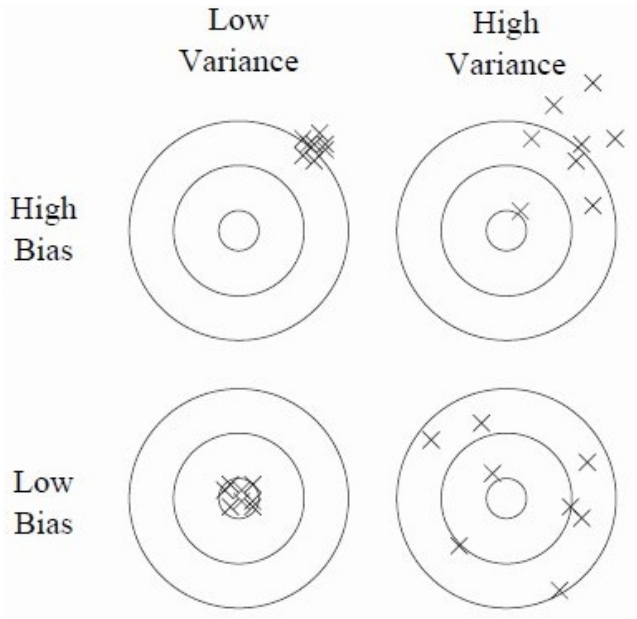
Theorem. For the squared error loss, the bias-variance decomposition of the expected generalization error at $X = \mathbf{x}$ is

$$\mathbb{E}_{\mathcal{L}}\{Err(\varphi_{\mathcal{L}}(\mathbf{x}))\} = \text{noise}(\mathbf{x}) + \text{bias}^2(\mathbf{x}) + \text{var}(\mathbf{x})$$

where

$$\text{noise}(\mathbf{x}) = Err(\varphi_B(\mathbf{x})),$$
$$\text{bias}^2(\mathbf{x}) = (\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L}}\{\varphi_{\mathcal{L}}(\mathbf{x})\})^2,$$
$$\text{var}(\mathbf{x}) = \mathbb{E}_{\mathcal{L}}\{(\mathbb{E}_{\mathcal{L}}\{\varphi_{\mathcal{L}}(\mathbf{x})\} - \varphi_{\mathcal{L}}(\mathbf{x}))^2\}.$$



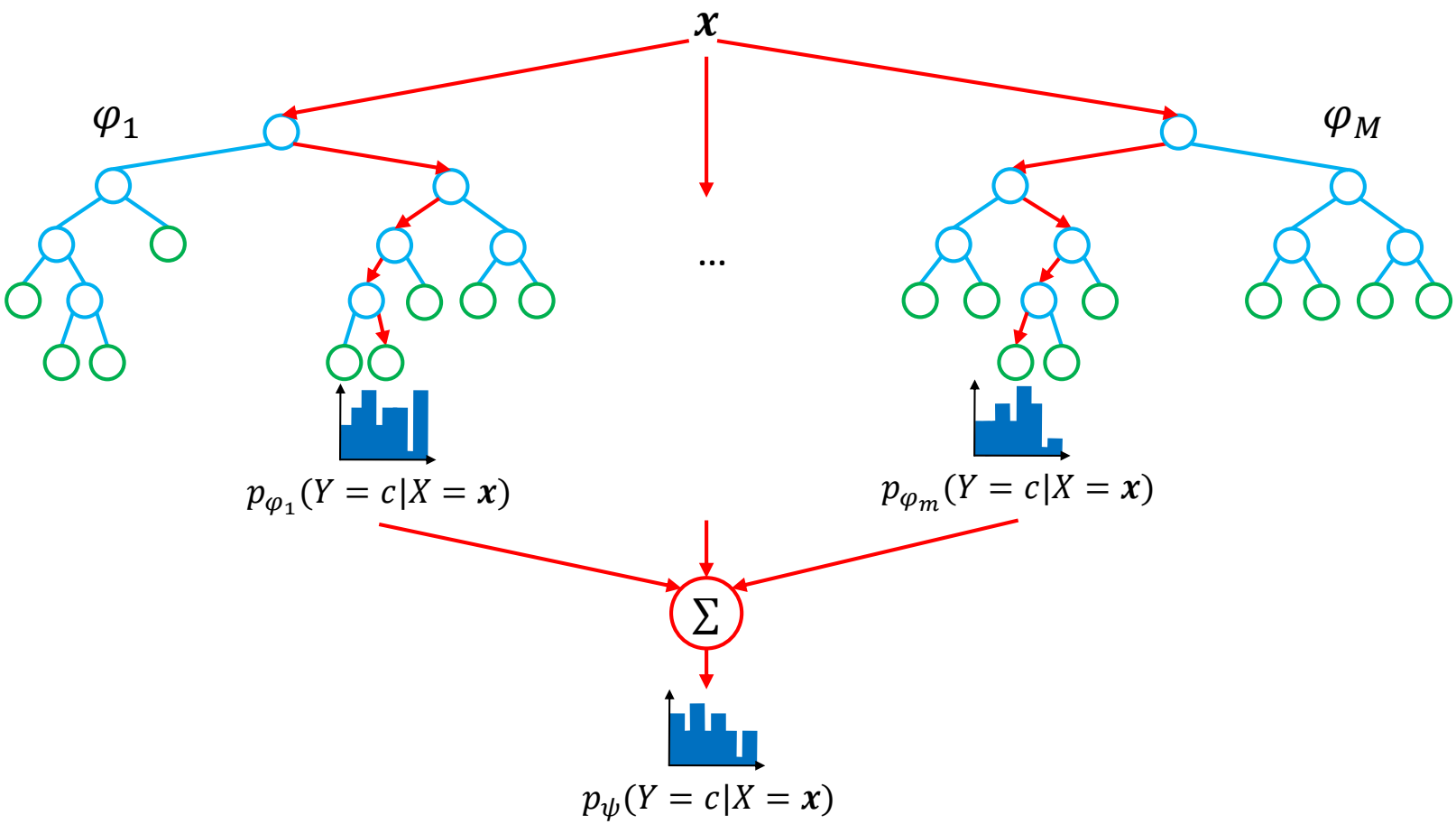


Diagnosing the generalization error of a decision tree

- (Residual error : Lowest achievable error, independent of $\varphi_{\mathcal{L}}$.)
- Bias : Decision trees usually have low bias.
- Variance : They often suffer from high variance.

- Solution : *Combine the predictions of several randomized trees into a single model.*

Random forests



Randomization

- Bootstrap samples
- Random selection of $K \leqslant p$ split variables } Random Forests
- Random selection of the threshold } Extra-Trees

Bias-variance decomposition (cont.)

Theorem. For the squared error loss, the bias-variance decomposition of the expected generalization error $\mathbb{E}_{\mathcal{L}}\{Err(\psi_{\mathcal{L},\theta_1,\ldots,\theta_M}(\mathbf{x}))\}$ at $X = \mathbf{x}$ of an ensemble of M randomized models $\varphi_{\mathcal{L},\theta_m}$ is

$$\mathbb{E}_{\mathcal{L}}\{Err(\psi_{\mathcal{L},\theta_1,\ldots,\theta_M}(\mathbf{x}))\} = \text{noise}(\mathbf{x}) + \text{bias}^2(\mathbf{x}) + \text{var}(\mathbf{x}),$$

where

$$\begin{aligned}
\text{noise}(\mathbf{x}) &= Err(\varphi_B(\mathbf{x})), \\
\text{bias}^2(\mathbf{x}) &= (\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L},\theta}\{\varphi_{\mathcal{L},\theta}(\mathbf{x})\})^2, \\
\text{var}(\mathbf{x}) &= \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{M}\sigma^2_{\mathcal{L},\theta}(\mathbf{x}).
\end{aligned}$$

and where $\rho(\mathbf{x})$ is the Pearson correlation coefficient between the predictions of two randomized trees built on the same learning set.

Diagnosing the generalization error of random forests

- Bias : Identical to the bias of a single randomized tree.
- Variance : $\text{var}(\mathbf{x}) = \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{M}\sigma^2_{\mathcal{L},\theta}(\mathbf{x})$
  As $M \to \infty$, $\text{var}(\mathbf{x}) \to \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x})$
  - The stronger the randomization, $\rho(\mathbf{x}) \to 0$, $\text{var}(\mathbf{x}) \to 0$.
  - The weaker the randomization, $\rho(\mathbf{x}) \to 1$, $\text{var}(\mathbf{x}) \to \sigma^2_{\mathcal{L},\theta}(\mathbf{x})$

Bias-variance trade-off. Randomization increases bias but makes it possible to reduce the variance of the corresponding ensemble model. The crux of the problem is to find the right trade-off.

Back to our example

Method	*Trees*	*MSE*
CART	1	1.055
Random Forest	50	0.517
Extra-Trees	50	0.507

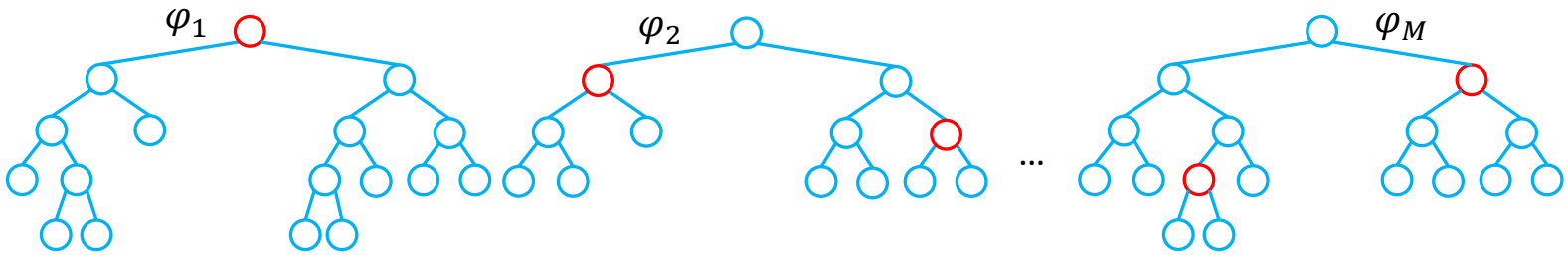
Combining several randomized trees indeed works better !

Outline

Variable importances

- Interpretability can be recovered through variable importances
- Two main importance measures :
  - **The mean decrease of impurity (MDI)** : summing total impurity reductions at all tree nodes where the variable appears (Breiman et al., 1984) ;
  - **The mean decrease of accuracy (MDA)** : measuring accuracy reduction on out-of-bag samples when the values of the variable are randomly permuted (Breiman, 2001).
- We focus here on MDI because :
  - It is faster to compute ;
  - It does not require to use bootstrap sampling ;
  - In practice, it correlates well with the MDA measure.

Mean decrease of impurity



Importance of variable X_j for an ensemble of M trees φ_m is :

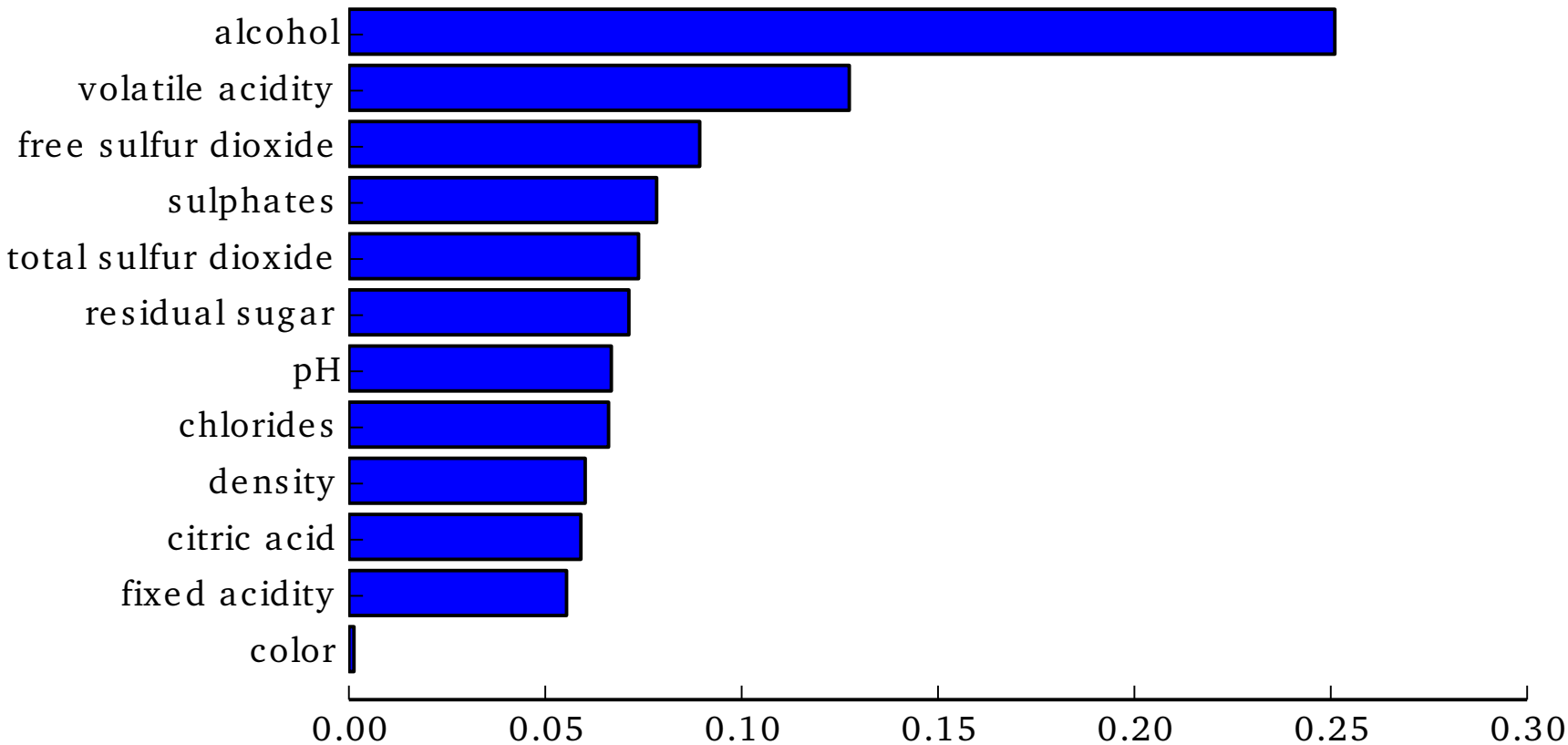
$$\text{Imp}(X_j) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t \in \varphi_m} 1(j_t = j) \Big[p(t) \Delta i(t) \Big],$$

where j_t denotes the variable used at node t, $p(t) = N_t/N$ and $\Delta i(t)$ is the impurity reduction at node t :

$$\Delta i(t) = i(t) - \frac{N_{t_L}}{N_t} i(t_L) - \frac{N_{t_r}}{N_t} i(t_R)$$

Back to our example

MDI scores as computed from a forest of 1000 fully developed trees on the Wine dataset (Random Forest, default parameters).



What does it mean ?

- MDI works well, but it is not well understood theoretically ;
- We would like to better characterize it and derive its main properties from this characterization.

- Working assumptions :
  - All variables are discrete ;
  - Multi-way splits à la C4.5 (i.e., one branch per value) ;
  - Shannon entropy as impurity measure :

$$i(t) = -\sum_c \frac{N_{t,c}}{N_t} \log \frac{N_{t,c}}{N_t}$$

  - Totally randomized trees (RF with $K = 1$) ;
  - Asymptotic conditions : $N \to \infty$, $M \to \infty$.

Result 1 : Three-level decomposition (Louppe et al., 2013)

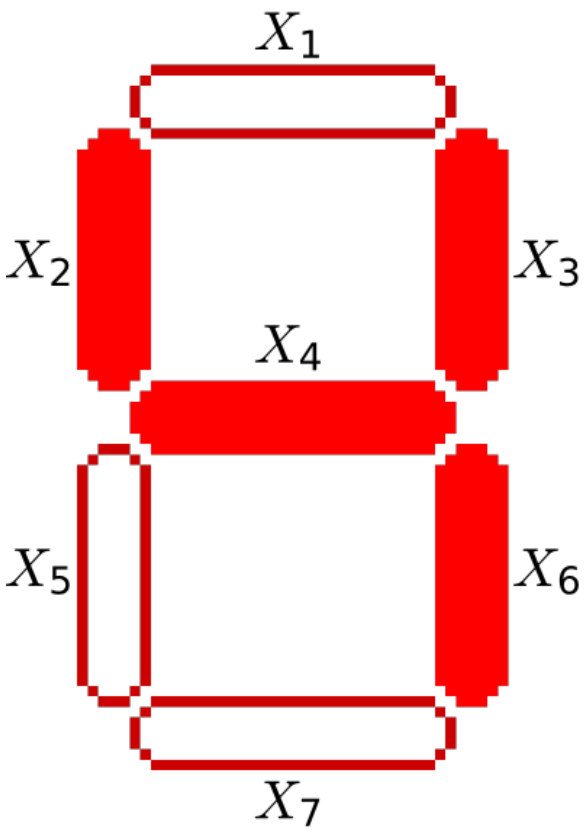
Theorem. Variable importances provide a three-level decomposition of the information jointly provided by all the input variables about the output, accounting for all interaction terms in a fair and exhaustive way.

$$\underbrace{I(X_1, \dots, X_p; Y)}_{\text{Information jointly provided by all input variables about the output}} = \underbrace{\sum_{j=1}^{p} \text{Imp}(X_j)}_{\text{i) Decomposition in terms of the MDI importance of each input variable}}$$

$$\text{Imp}(X_j) = \underbrace{\sum_{k=0}^{p-1} \frac{1}{C_p^k} \frac{1}{p-k}}_{\text{ii) Decomposition along the degrees } k \text{ of interaction with the other variables}} \underbrace{\sum_{B \in \mathcal{P}_k(V^{-j})} I(X_j; Y|B)}_{\text{iii) Decomposition along all interaction terms } B \text{ of a given degree } k}$$

E.g. : $p = 3, \text{Imp}(X_1) = \frac{1}{3} I(X_1; Y) + \frac{1}{6} (I(X_1; Y|X_2) + I(X_1; Y|X_3)) + \frac{1}{3} I(X_1; Y|X_2, X_3)$

Illustration : 7-segment display (Breiman et al., 1984)

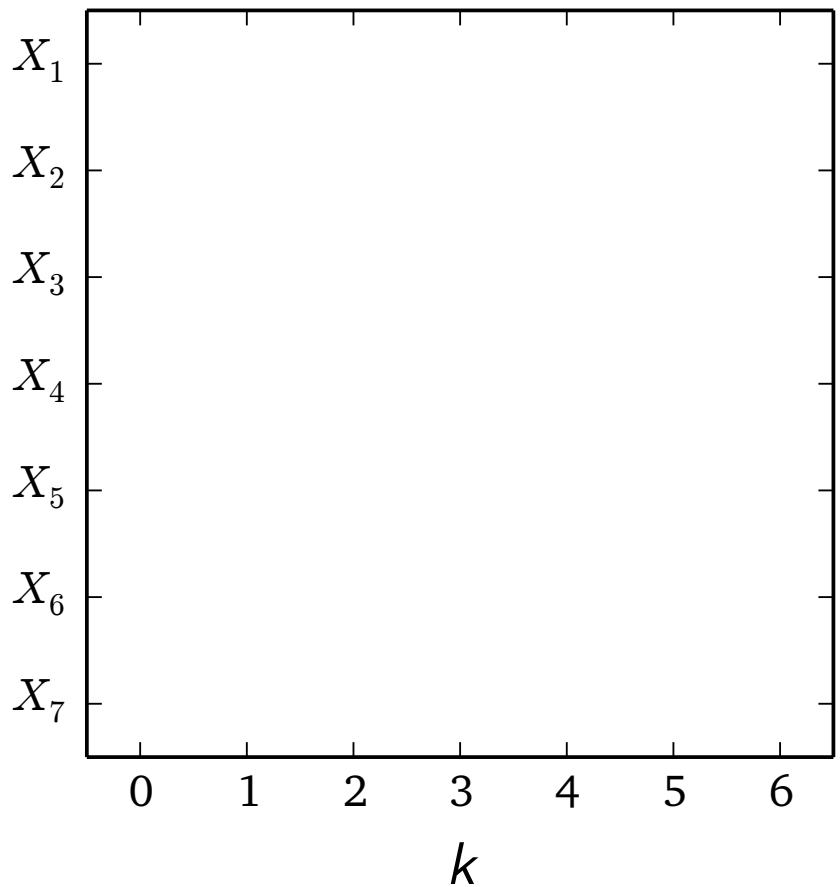


y	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	1	1	1	0	1	1	1
1	0	0	1	0	0	1	0
2	1	0	1	1	1	0	1
3	1	0	1	1	0	1	1
4	0	1	1	1	0	1	0
5	1	1	0	1	0	1	1
6	1	1	0	1	1	1	1
7	1	0	1	0	0	1	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Illustration : 7-segment display (Breiman et al., 1984)

$$\text{Imp}(X_j) = \sum_{k=0}^{p-1} \frac{1}{C_p^k} \frac{1}{p-k} \sum_{B \in \mathcal{P}_k(V^{-j})} I(X_j; Y|B)$$

Var	Imp
X_1	0.412
X_2	0.581
X_3	0.531
X_4	0.542
X_5	0.656
X_6	0.225
X_7	0.372
$\sum$	3.321



Result 2 : Irrelevant variables (Louppe et al., 2013)

Theorem. Variable importances depend only on the relevant variables.

Theorem. A variable X_j is irrelevant if and only if $\text{Imp}(X_j) = 0$.

$\Rightarrow$ The importance of a relevant variable is insensitive to the addition or the removal of irrelevant variables.

Definition (Kohavi & John, 1997). A variable X is *irrelevant* (to Y with respect to V) if, for all $B \subseteq V$, $I(X; Y|B) = 0$. A variable is *relevant* if it is not irrelevant.

Relaxing assumptions

When trees are not totally random...

- There can be relevant variables with zero importances (due to masking effects).
- The importance of relevant variables can be influenced by the number of irrelevant variables.

When the learning set is finite...

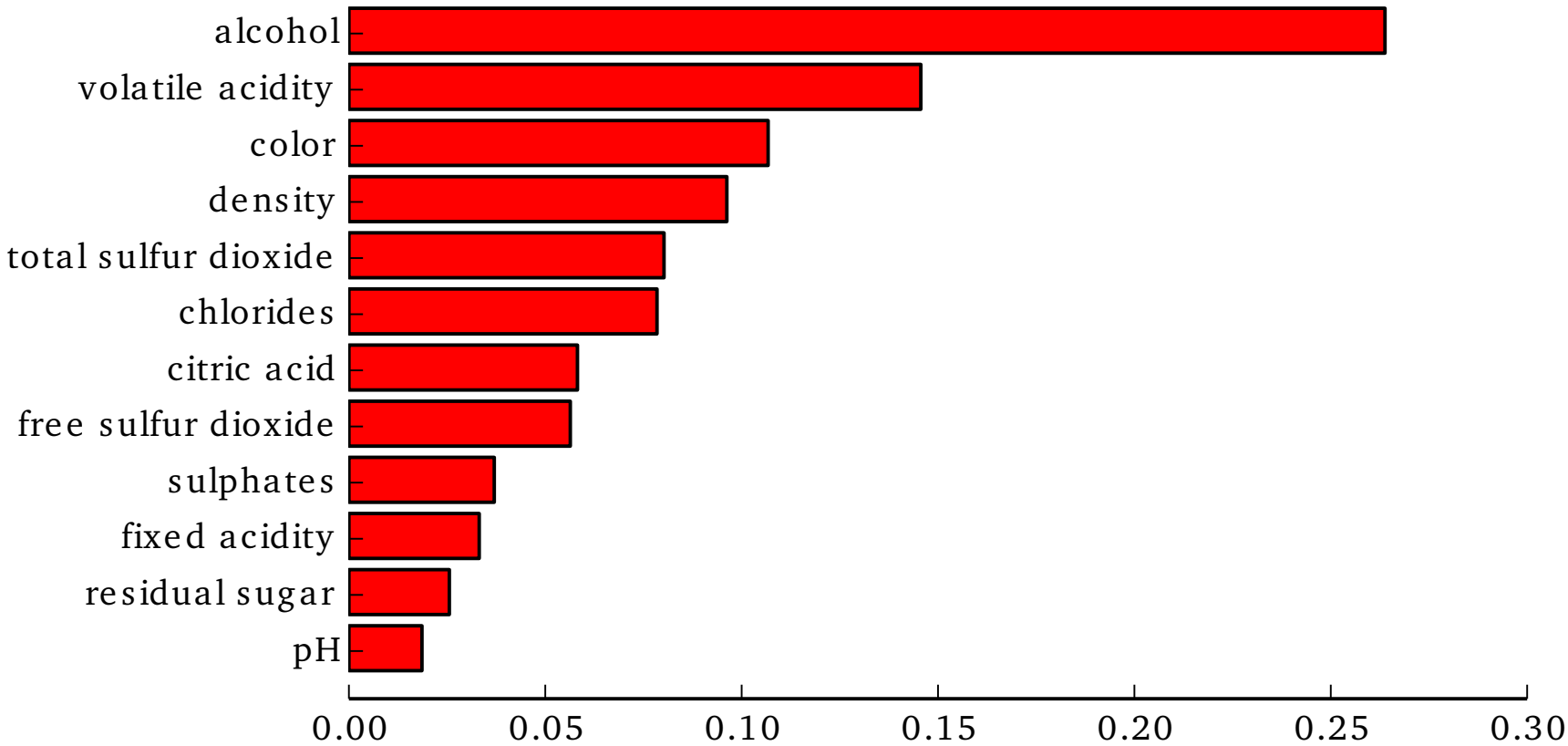
- Importances are biased towards variables of high cardinality.
- This effect can be minimized by collecting impurity terms measured from large enough sample only.

When splits are not multiway...

- $i(t)$ does not actually measure the mutual information.

Back to our example

MDI scores as computed from a forest of 1000 fixed-depth trees on the Wine dataset (Extra-Trees, $K = 1$, $\texttt{max\_depth} = 5$).



Taking into account (some of) the biases
results in quite a different story !

Outline

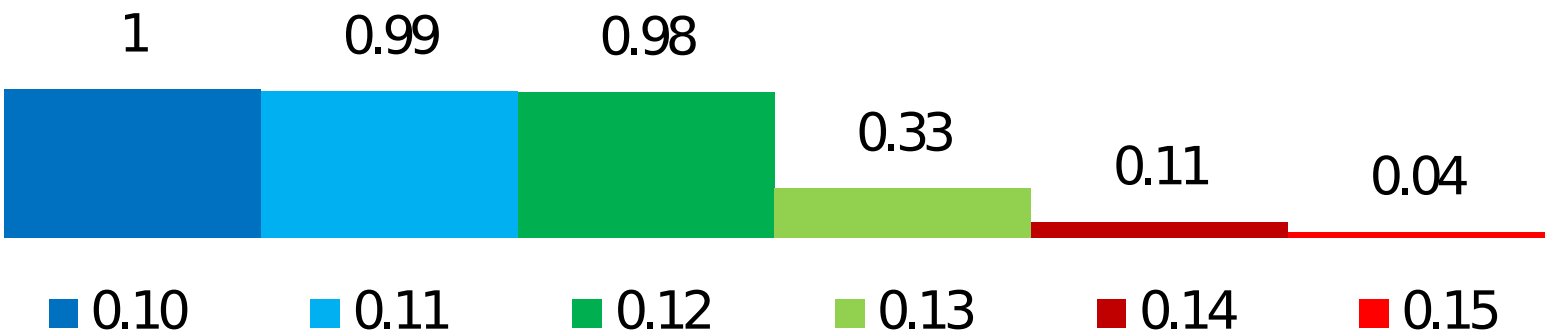
Implementation (Buitinck et al., 2013)

Scikit-Learn

- Open source **machine learning** library for Python
- Classical and **well-established algorithms**
- Emphasis on **code quality** and **usability**

A long team effort

Time for building a Random Forest (relative to version 0.10)



Implementation overview

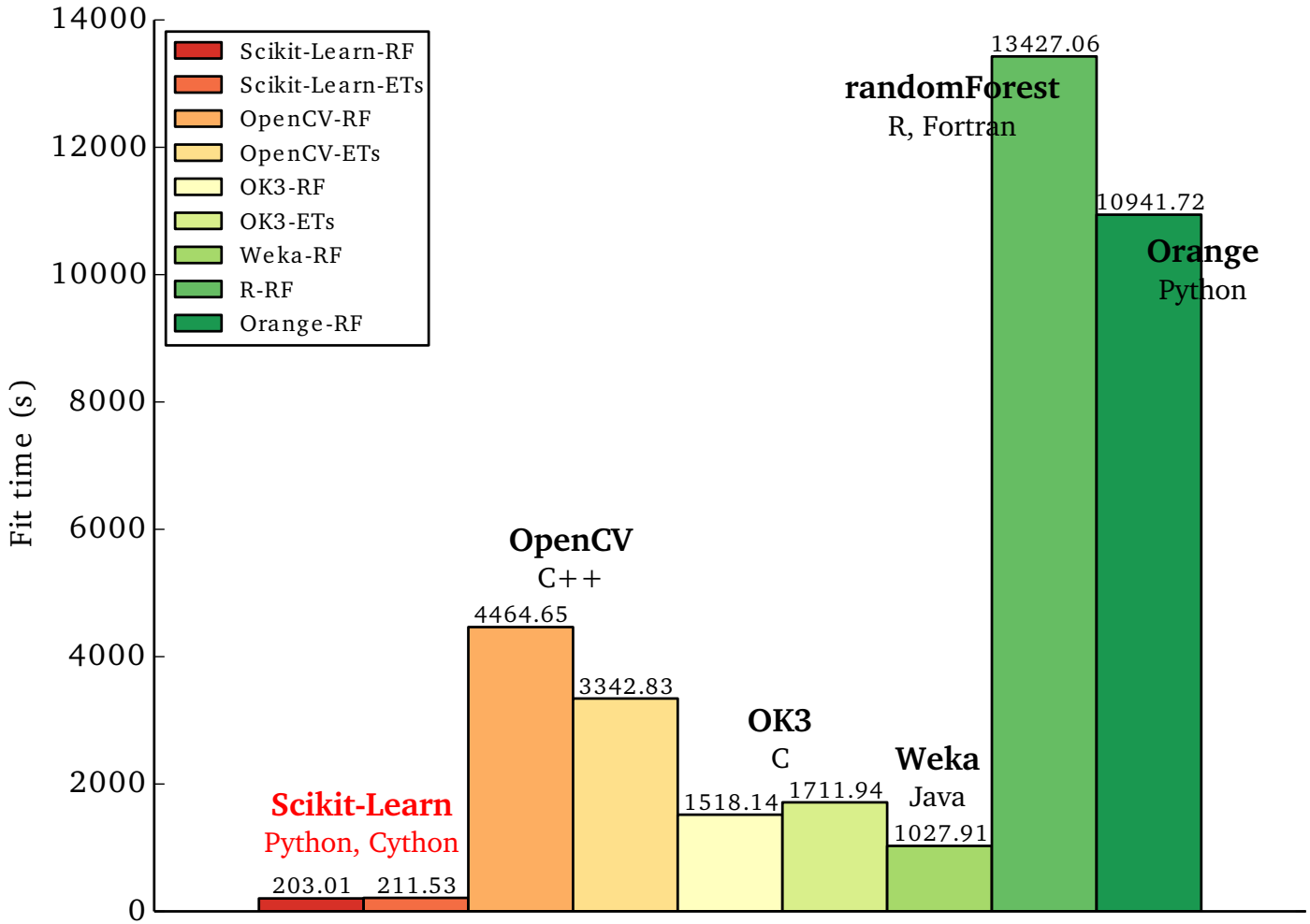
- Modular implementation, designed with a strict **separation of concerns**
  - Builders : for building and connecting nodes into a tree
  - Splitters : for finding a split
  - Criteria : for evaluating the goodness of a split
  - Tree : dedicated data structure
- Efficient **algorithmic formulation** [See Louppe, 2014]
  - Dedicated sorting procedure
  - Efficient evaluation of consecutive splits
- **Close to the metal**, carefully coded, implementation

  2300+ lines of Python, 3000+ lines of Cython, 1700+ lines of tests

```
# But we kept it stupid simple for users!
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

A winning strategy

Scikit-Learn implementation proves to be **one of the fastest** among all libraries and programming languages.



Computational complexity (Louppe, 2014)

	Average time complexity
CART	$\Theta(pN \log^2 N)$
Random Forest	$\Theta(MK\widetilde{N} \log^2 \widetilde{N})$
Extra-Trees	$\Theta(MKN \log N)$

- N : number of samples in $\mathcal{L}$
- p : number of input variables
- K : the number of variables randomly drawn at each node
- $\widetilde{N} = 0.632N$.

Improving scalability through randomization

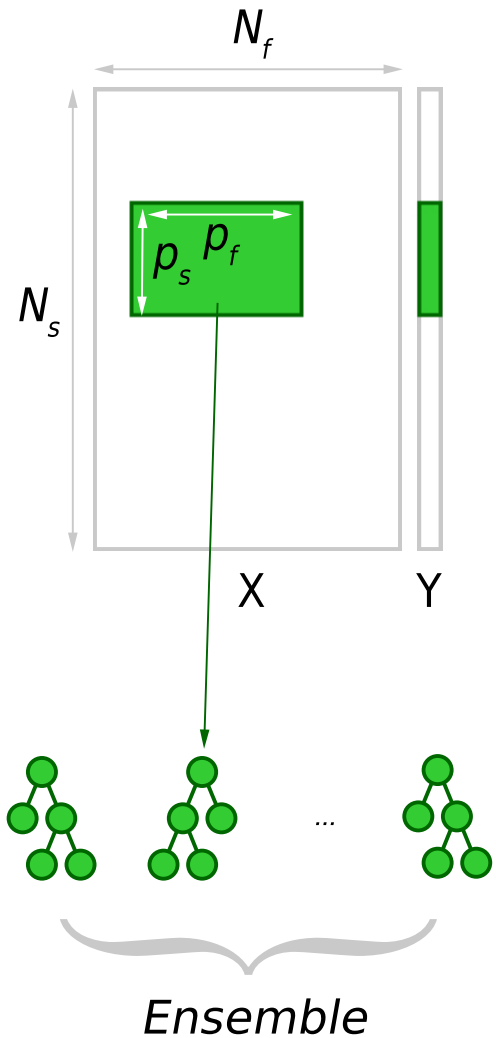
Motivation

- Randomization and averaging allow to improve accuracy by reducing variance.
- As a nice side-effect, the resulting algorithms are fast and embarrassingly parallel.
- Why not purposely exploit randomization to make the algorithm even more scalable (and at least as accurate) ?

Problem

- Let assume a supervised learning problem of N_s samples defined over N_f features. Let also assume T computing nodes, each with a memory capacity limited to M_{max}, with $M_{max} \ll N_s \times N_f$.
- How to best exploit the memory constraint to obtain the most accurate model, as quickly as possible ?

A straightforward solution : Random Patches (Louppe et al., 2012)

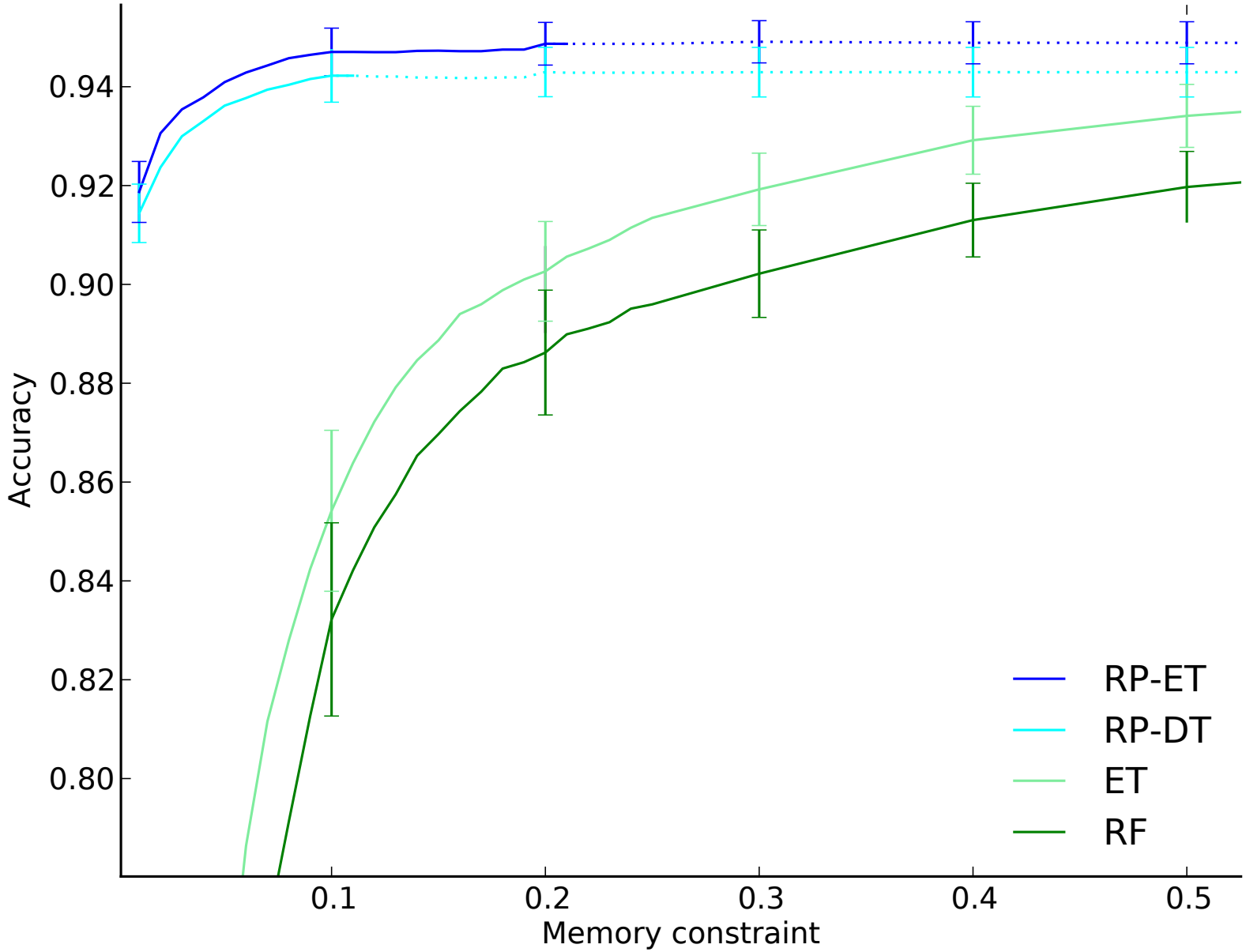


1. Draw a subsample r of $p_s N_s$ random examples, with $p_f N_f$ random features.
2. Build a *base estimator* on r.
3. Repeat 1-2 for a number T of estimators.
4. Aggregate the predictions by voting.

p_s and p_f are two meta-parameters that should be selected

- such that $p_s N_s \times p_f N_f \leqslant M_{max}$
- to optimize accuracy

Impact of memory constraint



Lessons learned from subsampling

- Training each estimator on the whole data is (often) useless. The size of the random patches can be reduced without (significant) loss in accuracy.
- As a result, both memory consumption and training time can be reduced, at low cost.
- With strong memory constraints, RP can exploit data better than the other methods.
- Sampling features is critical to improve accuracy. Sampling the examples only is often ineffective.

Outline

Opening the black box

- Random forests constitute one of the most robust and effective machine learning algorithms for many problems.
- While simple in design and easy to use, random forests remain however
  - hard to analyze theoretically,
  - non-trivial to interpret,
  - difficult to implement properly.
- Through an in-depth re-assessment of the method, this dissertation has proposed original contributions on these issues.

Future works

Variable importances

- Theoretical characterization of variable importances in a finite setting.
- (Re-analysis of) empirical studies based on variable importances, in light of the results and conclusions of the thesis.
- Study of variable importances in boosting.

Subsampling

- Finer study of subsampling statistical mechanisms.
- Smart sampling.

Questions ?

Backup slides

Condorcet's jury theorem

Let consider a group of M voters.

If each voter has an independent probability $p > \frac{1}{2}$ of voting for the correct decision, then adding more voters increases the probability of the majority decision to be correct.

When $M \to \infty$, the probability that the decision taken by the group is correct approaches 1.

Interpretation of $\rho(\mathbf{x})$ (Louppe, 2014)

Theorem. $\rho(\mathbf{x}) = \frac{\mathbb{V}_{\mathcal{L}}\{\mathbb{E}_{\theta|\mathcal{L}}\{\varphi_{\mathcal{L},\theta}(\mathbf{x})\}\}}{\mathbb{V}_{\mathcal{L}}\{\mathbb{E}_{\theta|\mathcal{L}}\{\varphi_{\mathcal{L},\theta}(\mathbf{x})\}\}+\mathbb{E}_{\mathcal{L}}\{\mathbb{V}_{\theta|\mathcal{L}}\{\varphi_{\mathcal{L},\theta}(\mathbf{x})\}\}}$

In other words, it is the ratio between

- the variance due to the learning set and
- the total variance, accounting for random effects due to both the learning set and the random perburbations.

$\rho(\mathbf{x}) \to 1$ when variance is mostly due to the learning set ;
$\rho(\mathbf{x}) \to 0$ when variance is mostly due to the random perturbations ;
$\rho(\mathbf{x}) \geqslant 0$.