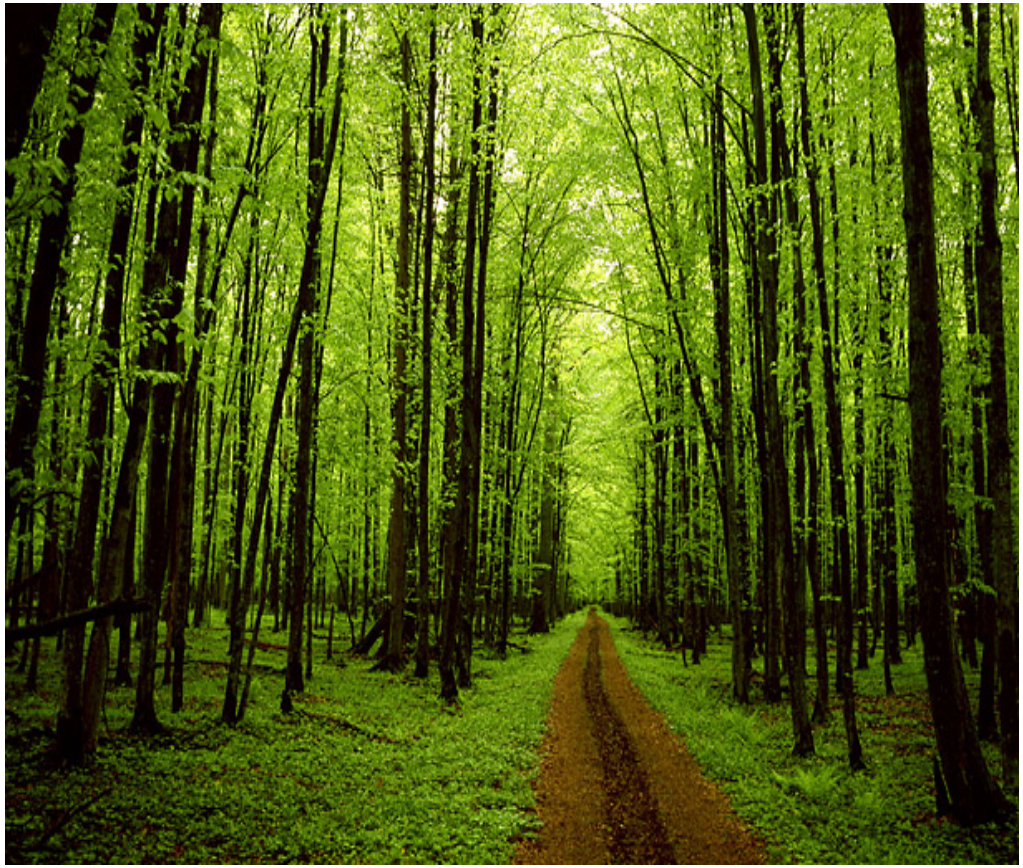
Ch5. Trees and forests

ST4240, 2016/2017

Version 0.1

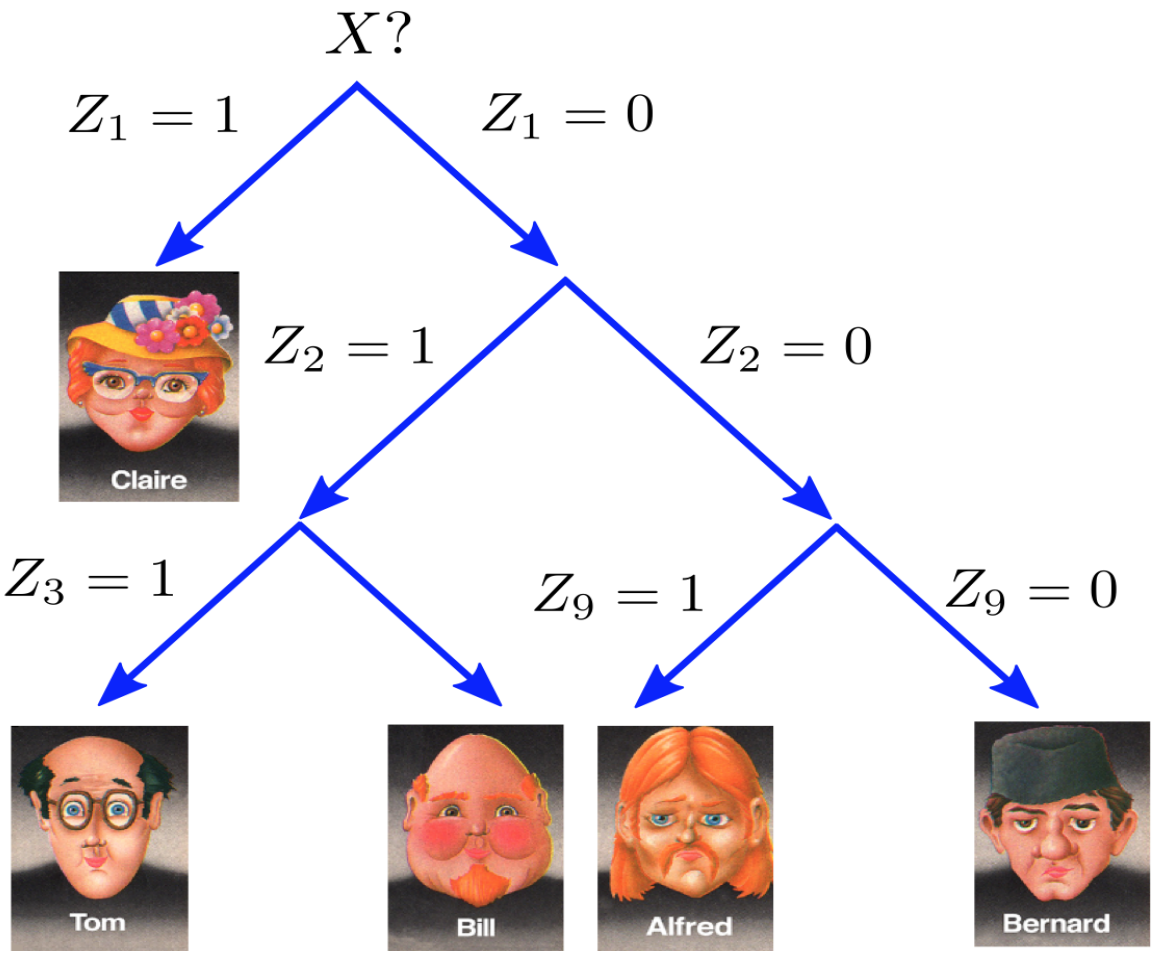
Alexandre Thiéry

Department of Statistics and Applied Probability

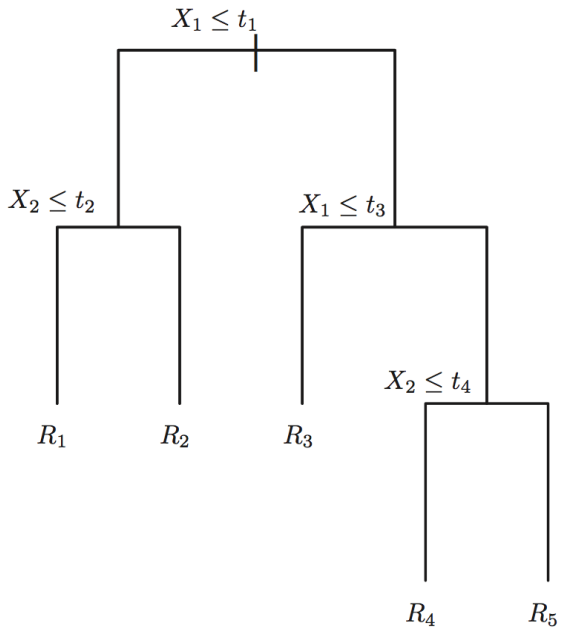


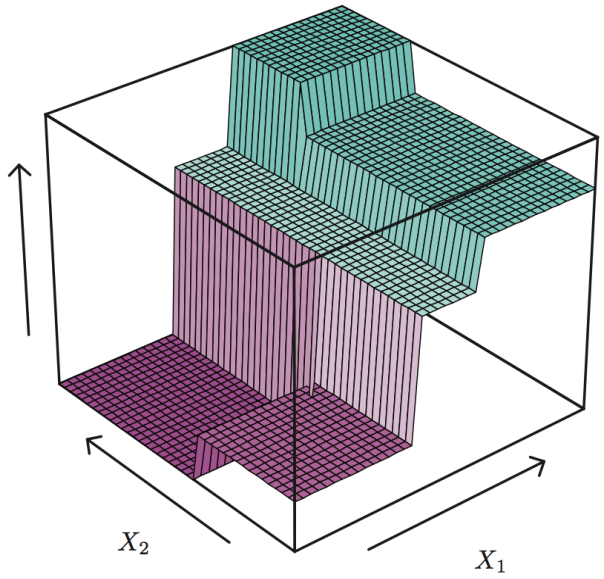
Outline

Decision Trees



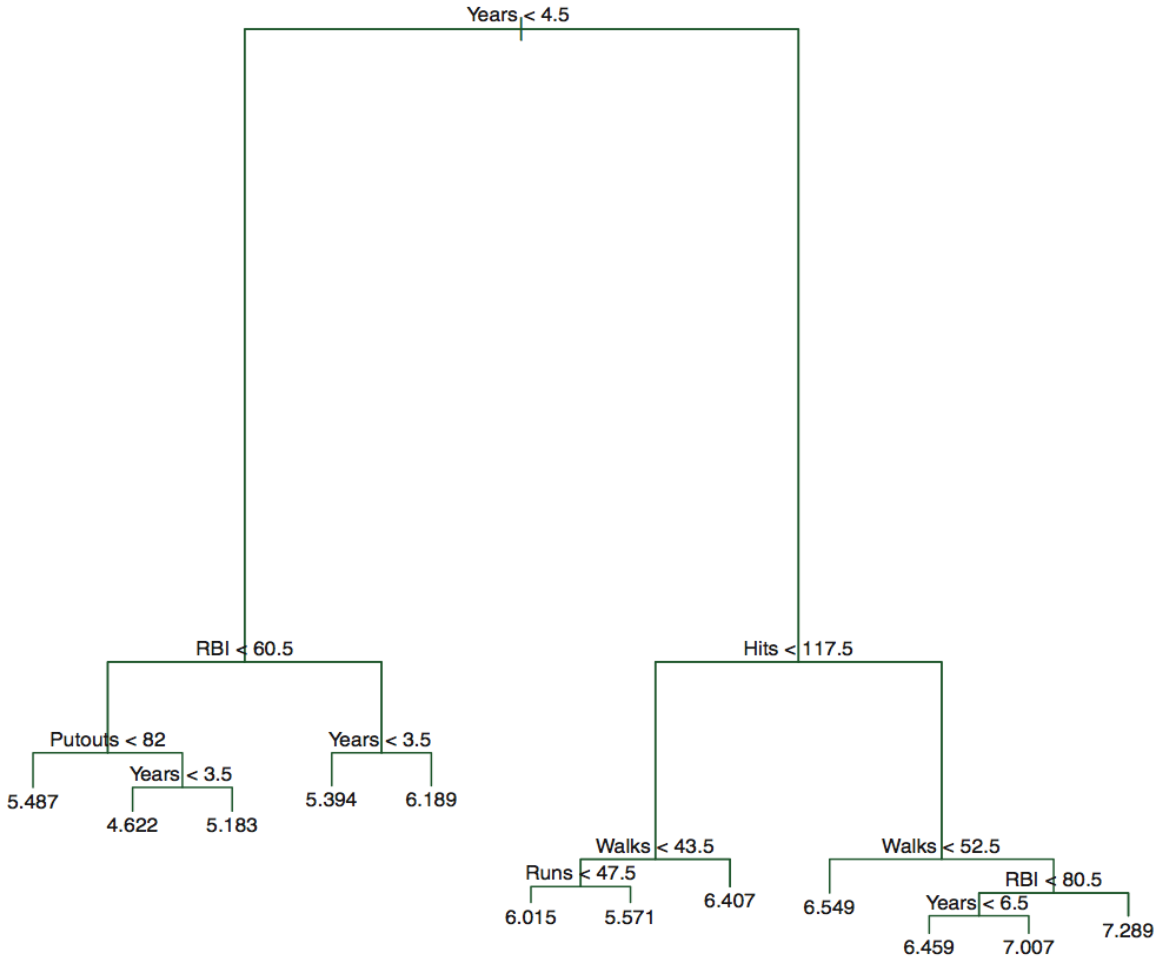
Decision Trees



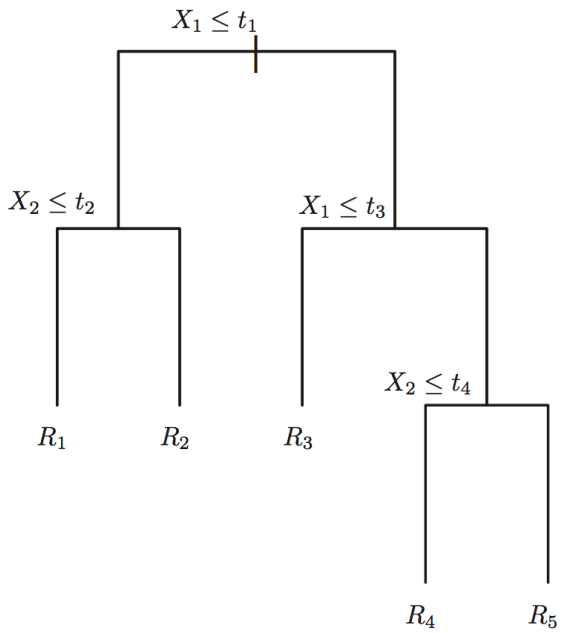


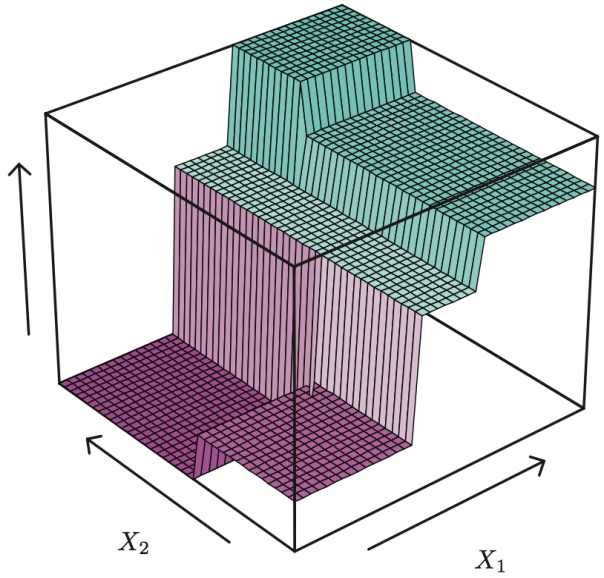
- Simple and useful for interpretation
- Typically not competitive with more advanced techniques
- Later, we will see how to aggregate many trees...

Hitters dataset: salary prediction



Decision Trees





- J distinct and non-overlapping regions $\{R_1, \ldots, R_J\}$
- In this lecture, we assume the simplistic model

$$f(x) = \sum_{j=1}^{J} c_j \, \mathbb{I}(x \in R_j)$$

Property of the mean

- [**Exercise**] consider N reals numbers $\{x_1, \dots, x_n\}$. What is

$$\operatorname{argmin} \left\{ m \mapsto \sum_{i=1}^{N} (m - x_i)^2 \right\}$$

- *[**Exercise**] consider N reals numbers $\{x_1, \dots, x_n\}$. What is

$$\operatorname{argmin} \left\{ m \mapsto \sum_{i=1}^{N} |m - x_i| \right\}$$

Trees for regression

- Training examples $\{x_i, y_i\}$ where
  - x_i is a vector of predictors
  - y_i is a real output
- Suppose that we already have a partition $\{R_1, \ldots, R_J\}$

$$f(x) = \sum_{j=1}^{J} c_j \, \mathbb{I}(x \in R_j)$$

- The mean squared error (**MSE**) on the training set is

$$(1/N) \sum_{j=1}^{J} \sum_{x_i \in R_j} (y_i - c_j)^2$$

- [**Exercise**] how should we choose the $\{c_j\}_{j=1}^{J}$ to minimise the **MSE**.

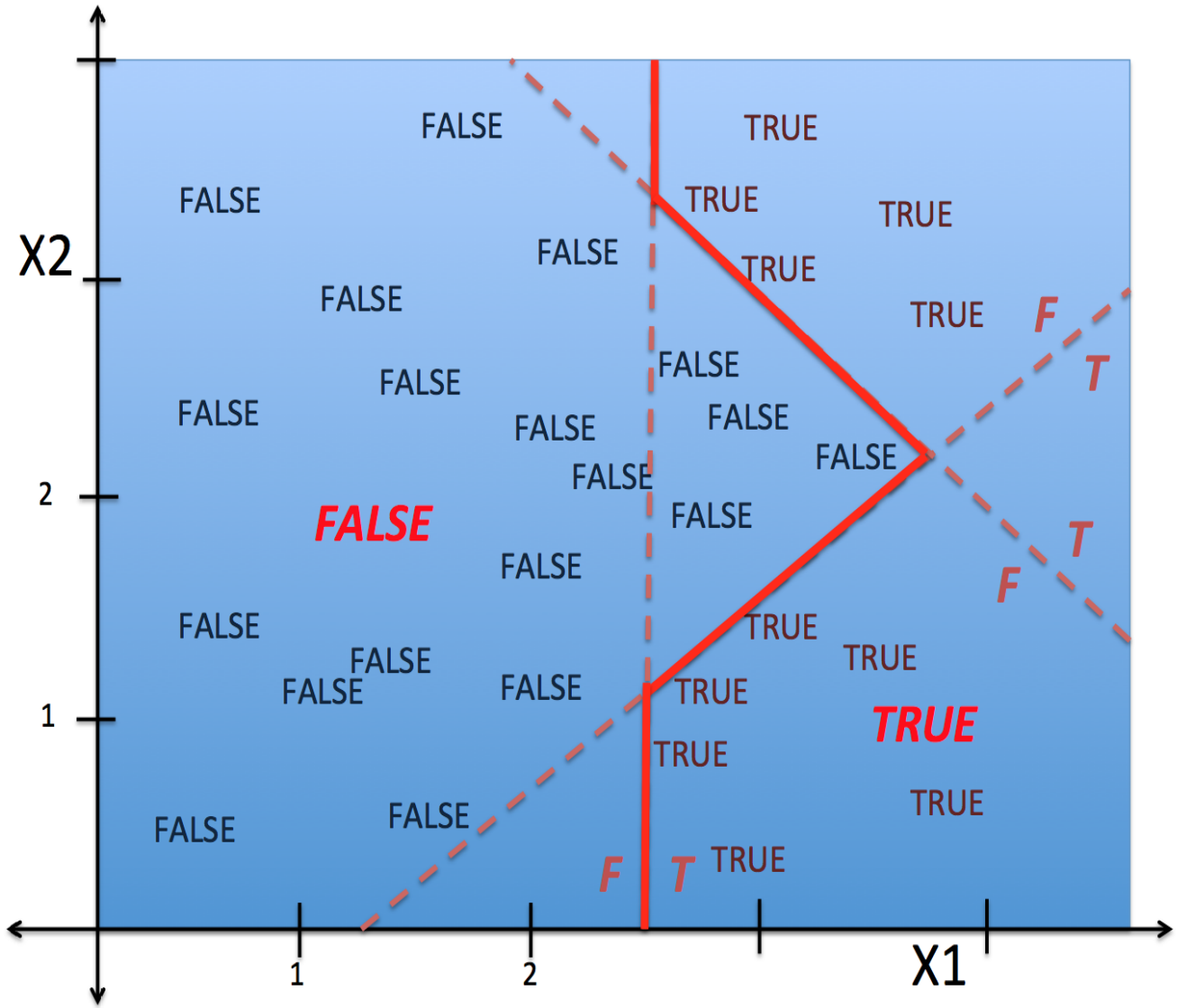
Trees for classification

- Training examples $\{x_i, y_i\}$ where
  - x_i is a vector of predictors
  - y_i is a categorical output
- Suppose that we already have a partition $\{R_1, \ldots, R_J\}$

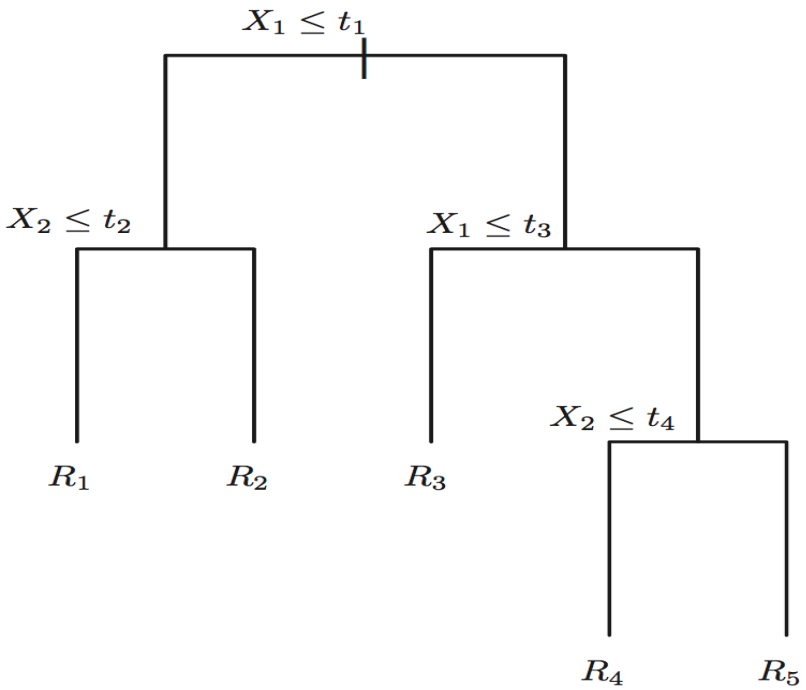
$$f(x) = \sum_{j=1}^{J} c_j \, \mathbb{I}(x \in R_j)$$

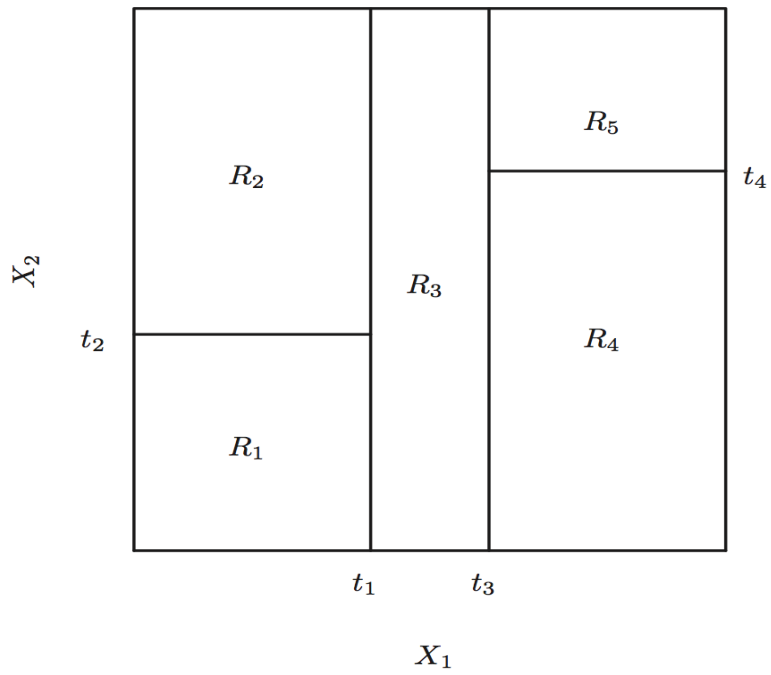
- **[Exercise]** what $\{c_j\}_{j=1}^{J}$ to minimise the error rate?

Too complicated!



Binary trees: recursive splitting





Growing trees

- For a given partition, one knows how to find the optimal $\{c_j\}_{j=1}^{J}$
- Computationally infeasible to find the best partition!
- Instead, we adopt a greedy approach
  - each step: chose the best variable and the best split
  - continue until a small number of training examples (eg. ≤ 5) live on each leave
- 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.

Best variable and best split?

- For regression, choose the variable and the split which lead to the largest decrease of the **MSE**.
- For classification with K classes, one could try to choose the variable and the split which lead to the largest decrease of the error rate. In practice, the error rate is not sensitive enough and one choose instead a measure of **purity** such as
  - **Gini index:**

$$(\text{Gini}) \equiv \sum_{j=1}^{J} \sum_{k=1}^{K} p_{j,k}(1 - p_{j,k})$$

  - **Cross-Entropy:**

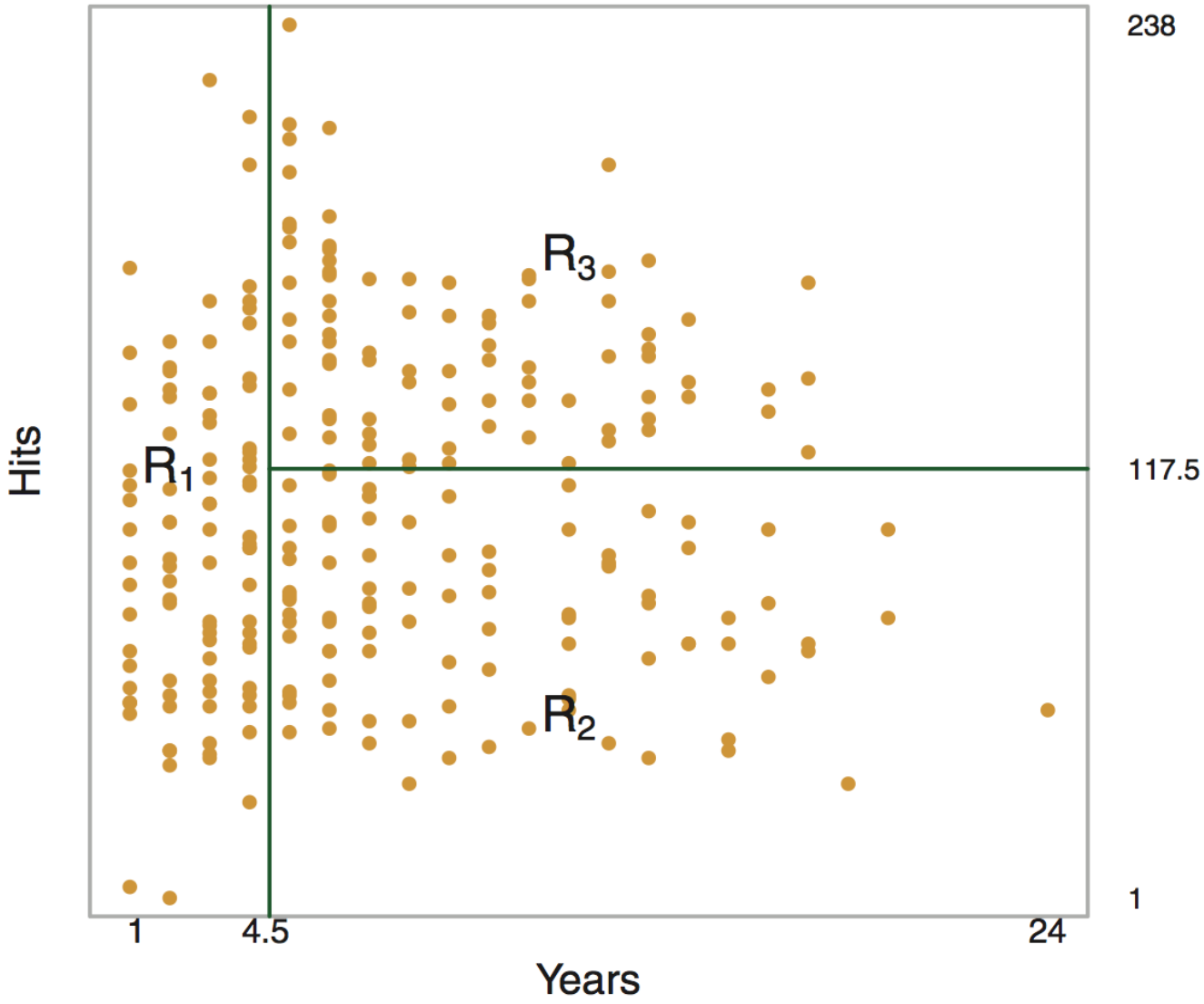
$$(\text{Cross-Entropy}) \equiv -\sum_{j=1}^{J} \sum_{k=1}^{K} p_{j,k} \log(p_{j,k})$$

with $p_{j,k}$ the proportion of training examples in R_j that belong to the k-th class.

Purity?

- [**Exercise**] Plot the function $x \mapsto x(1-x)$ and $x \mapsto -x\log(x)$ on $x \in (0,1)$. Explain why the gini index and the cross-entropy are sensible measures of purity when growing classification trees.

Too complicated!



Overfitting

- Building a tree in a greedy manner until all the leaves contain a small number (e.g. ≤ 5) of training examples will typically lead to bad predictive power i.e. **overfitting**.
- Instead of minimising the **MSE** (or Gini index, or Cross-entropy), one can instead try to minimise

$$(\text{measure of discrepancy}) + (\text{measure of complexity})$$

- For a parameter $\lambda > 0$, a typical choice is the following,

$$\textbf{MSE}(T) + \lambda \times |T|$$

where $|T|$ is the number of terminal leaves.

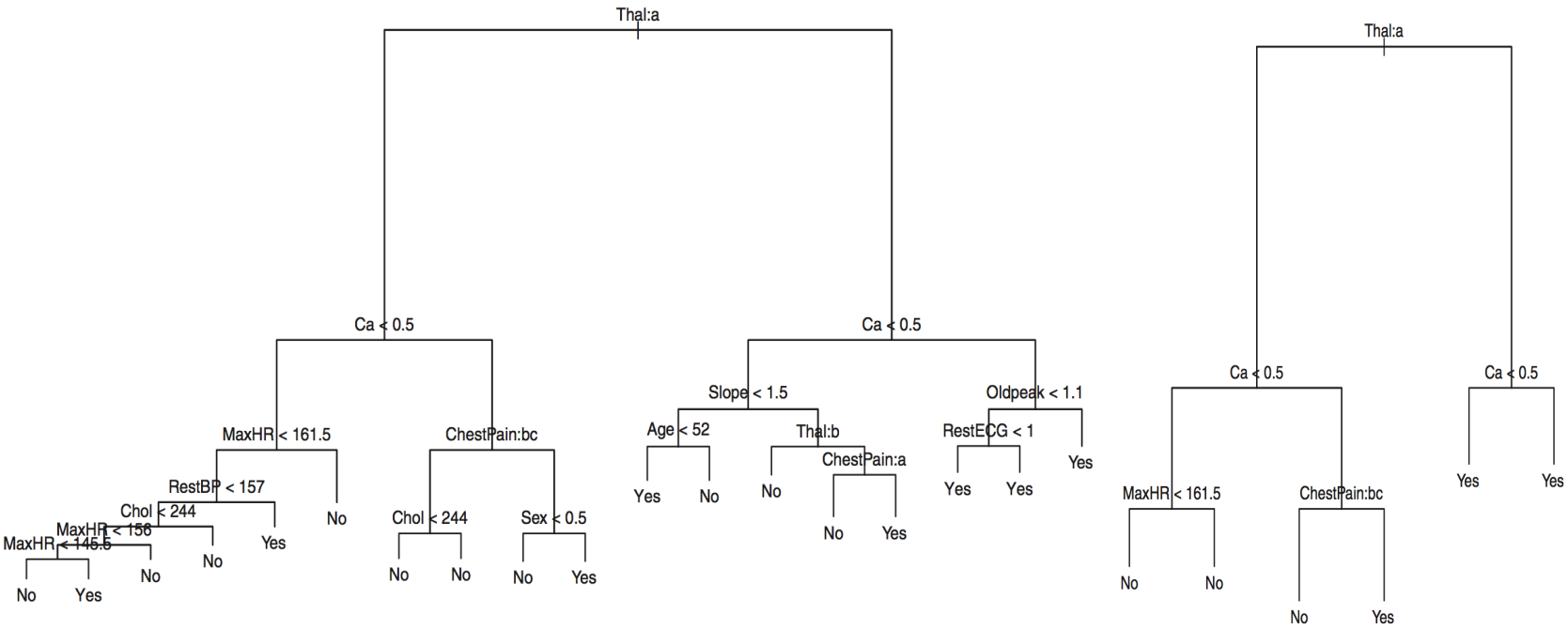
Building a tree in practice

- Use a greedy approach to build a large tree T that only has a few number of training examples in each leave.
- **Pruning**: for a regularisation parameter $\lambda > 0$, find the sub-tree T_0 of T that minimises

$$t \mapsto \mathbf{MSE}(t) + \lambda \times |t|$$

- [**Exercise**] how to choose $\lambda > 0$?

Pruning: Before and After



Building a tree in practice

- Trees can be **displayed graphically** and are easily **interpreted**
- Trees can easily handle **qualitative predictors** without the need to create dummy variables.

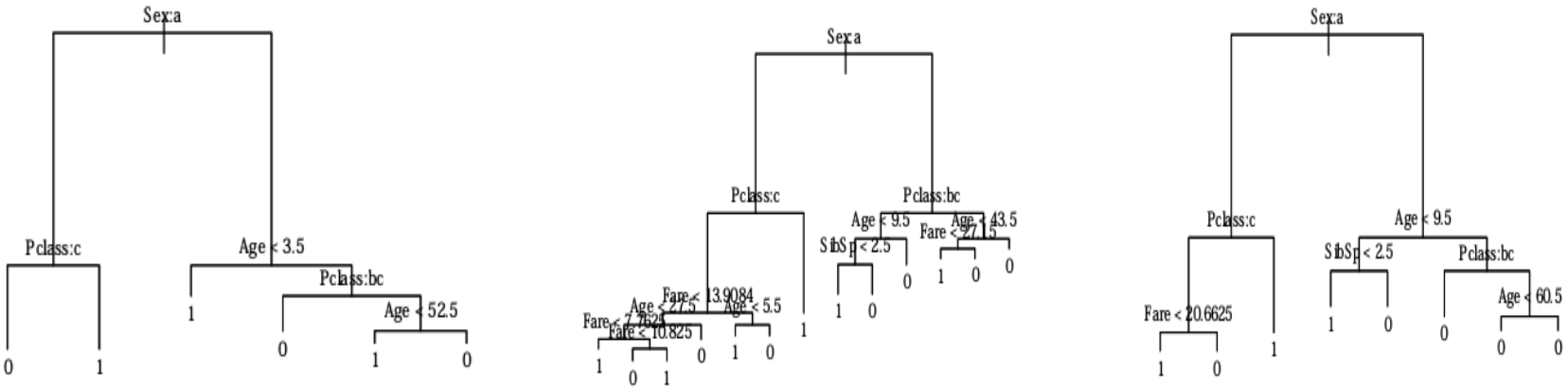
but

- Basic trees typically **do not have very good performances**.
- Trees can be **extremely sensitive** to the training examples.

Outline

Bagging = **B**ootstrap **Agg**egation

- Growing trees the greedy way is very **unstable**
- Slightly different training set, **wildly different resulting tress!**



Trees grown on the subsets of the *Titanic* dataset

Bagging

- Grow **many trees** on slightly different subsets of the training data and **average the results**
- Suppose that the training set has N elements $\{x_1, \ldots, x_N\}$
- Consider B subsets $\{S_1, \ldots, S_B\}$ where S_j is obtained by sampling with replacement N elements from the training set
- [**Exercise**] find the expected number of **different** elements in S_j

Bagging

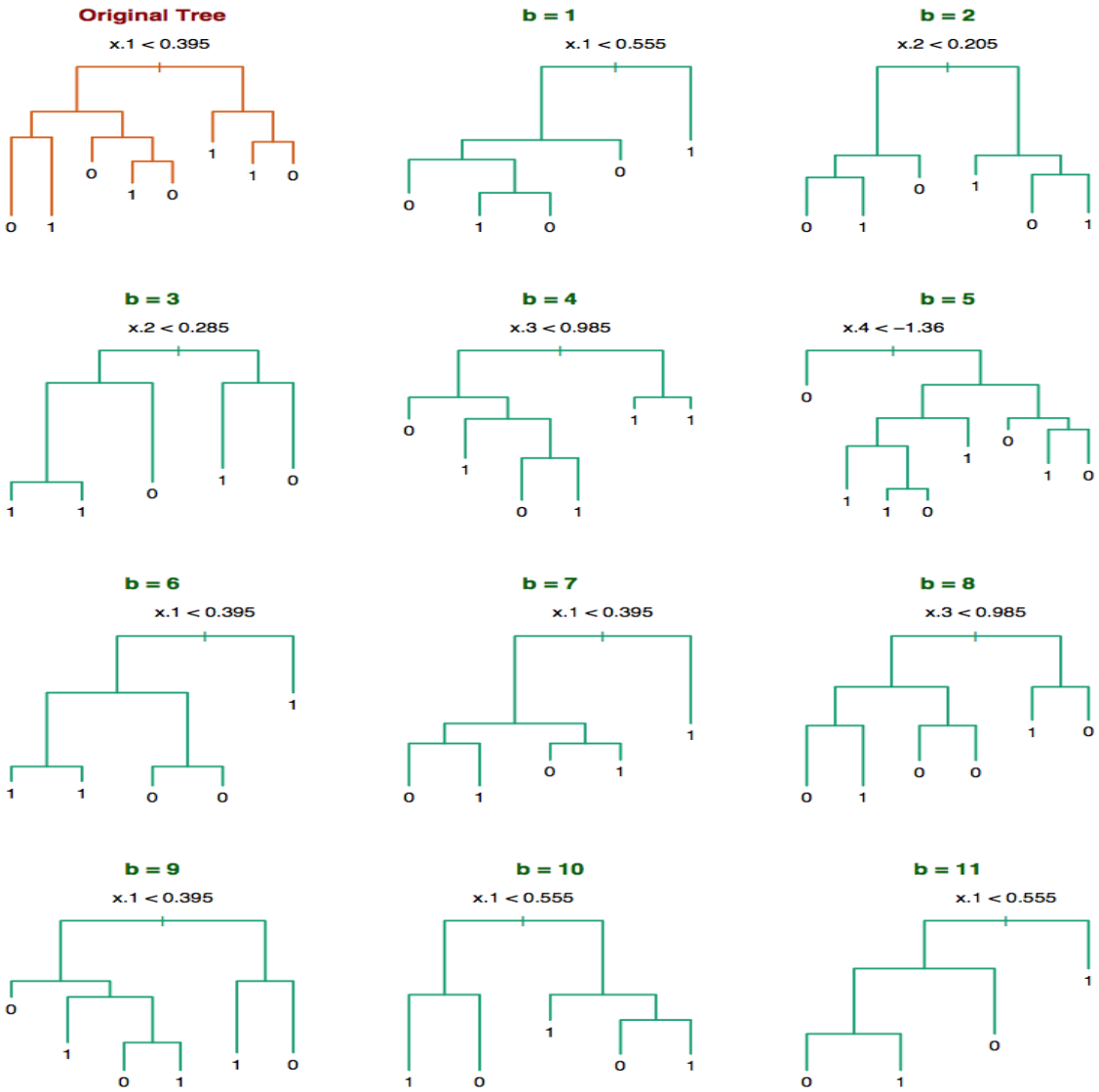
- Grow a tree T_{S_j} on each data set S_j
- Regression:

$$\widehat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} T_{S_j}(x)$$

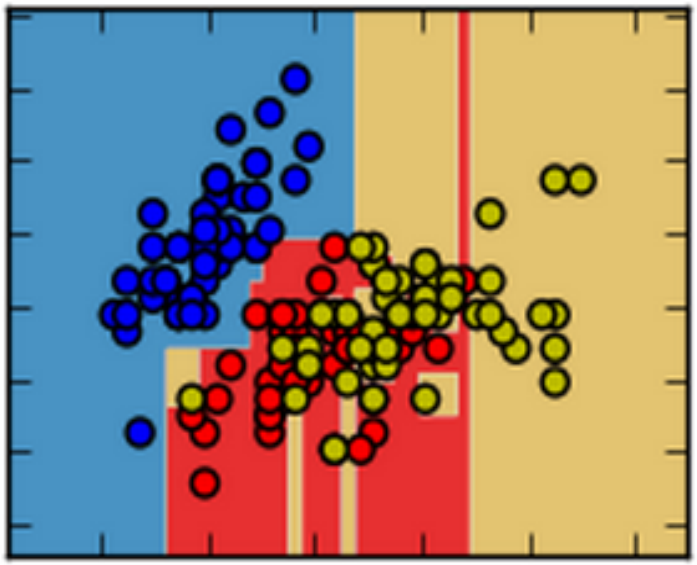
- Classification: bagging can produce a probabilistic output

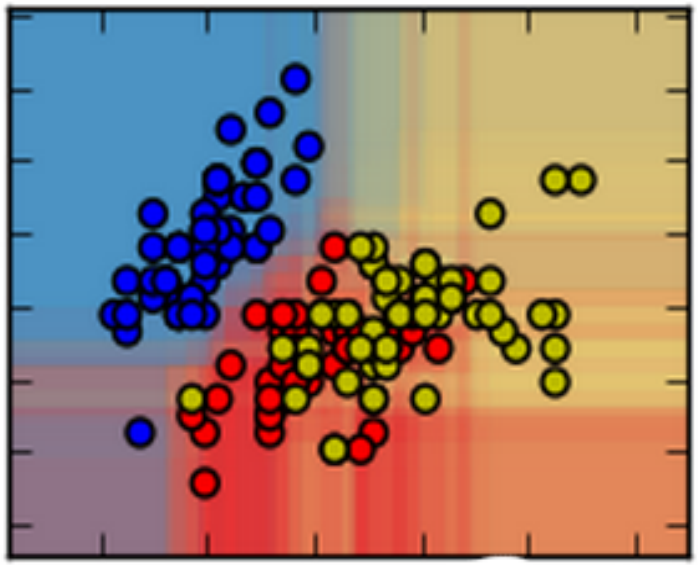
$$\widehat{\mathbb{P}}(y = k \mid x) = \frac{1}{B} \sum_{b=1}^{B} \mathbb{I}\left(T_{S_j}(x) = k\right)$$

Bootstrapped trees

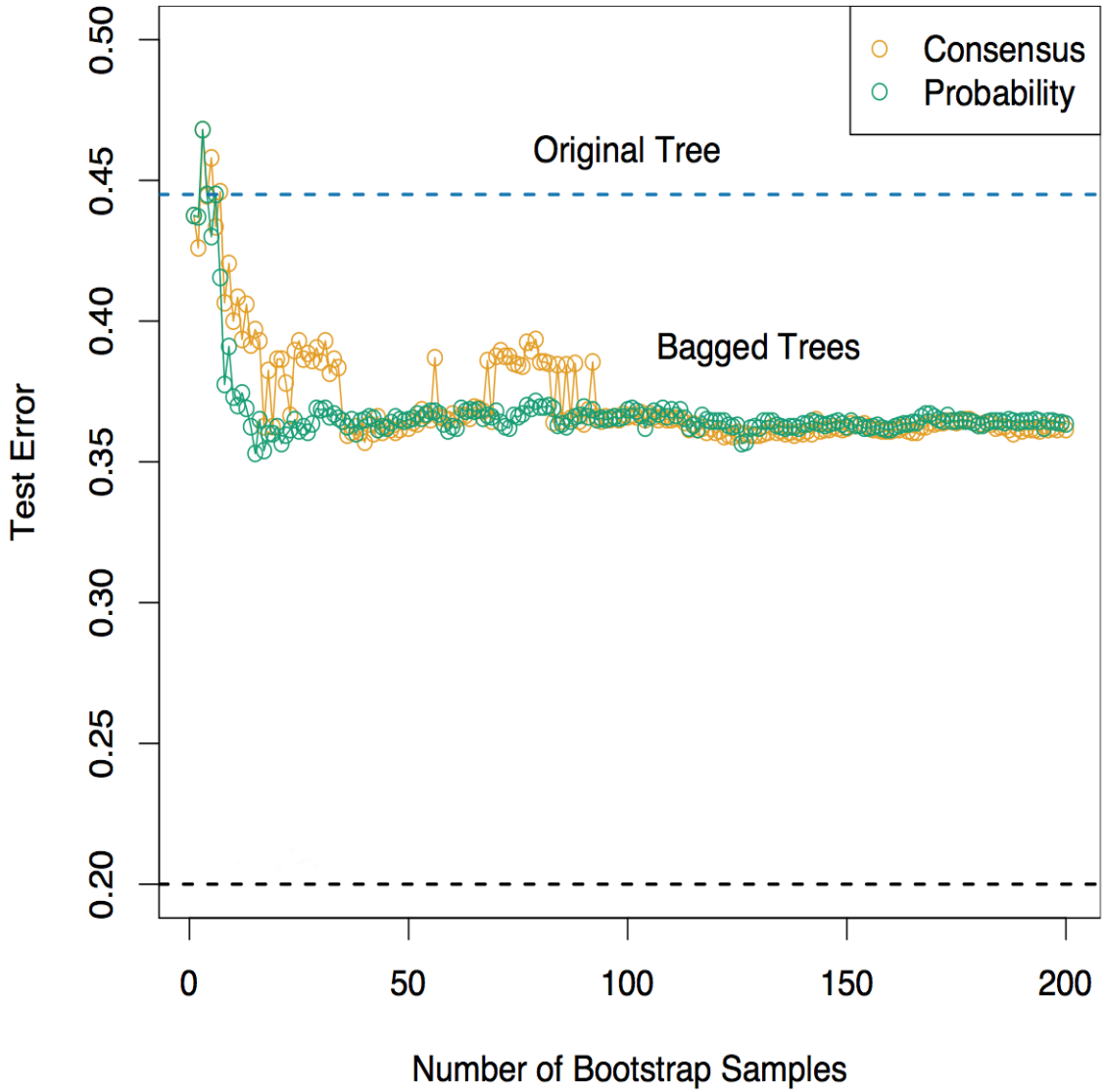


Decision Tree v.s. Bagged estimate





Error v.s. Boostrap size



Regression: bias-variance

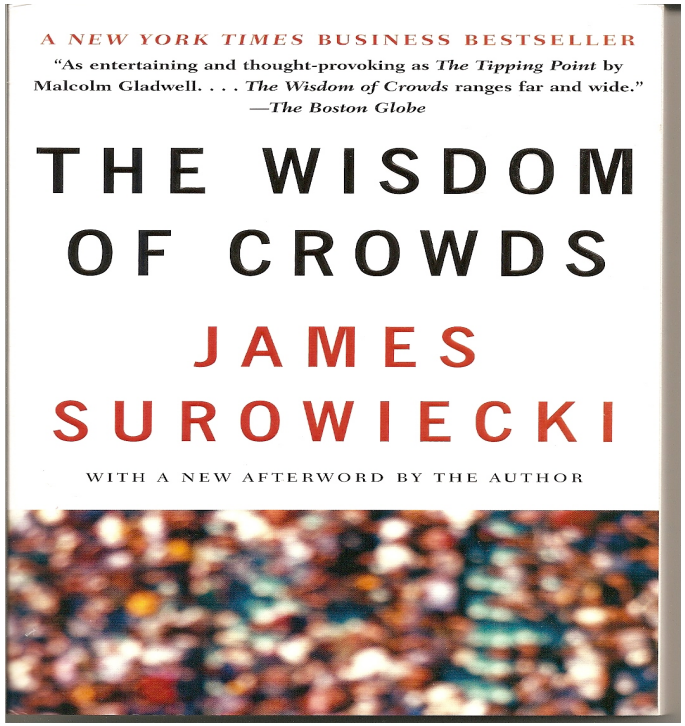
- [**Exercise**] Consider n estimators $X_1, \ldots, X_n$ of a quantity $X_\star$. Assume that the estimators are i.i.d. The pooled estimate is

$$\overline{X} = \frac{X_1 + \ldots + X_n}{n}$$

What is the MSE of $\overline{X}$ in terms of

$$(\text{bias}) = \mathbb{E}[X_i] - X_\star \qquad \text{and} \qquad \text{Var}(X_j)$$

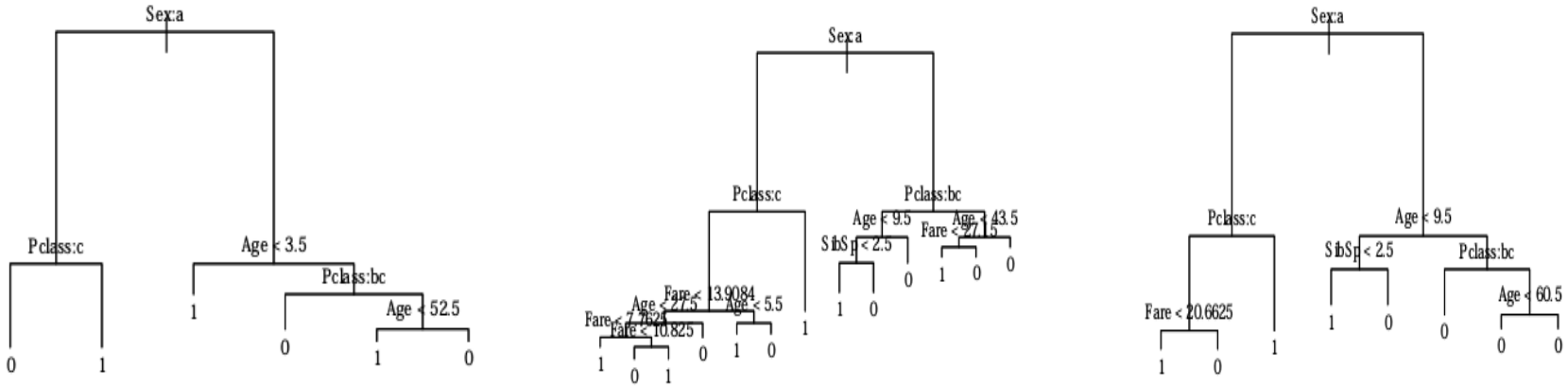
Classification: Wisdom of crowds



- [**Exercise**] consider n (independent) persons answering a Yes/No answer. Suppose that each person has a probability $p > 0.5$ of getting the right question. What is the probability that the *crowd* gives the right answer?

Bagging and independence

- In the two previous slides, we assumed **independence** !



Does it look like **independent**?

- **[Exercise]** what is the MSE of $\overline{X}$ if one assume pairwise correlation of $\rho > 0$?

Outline

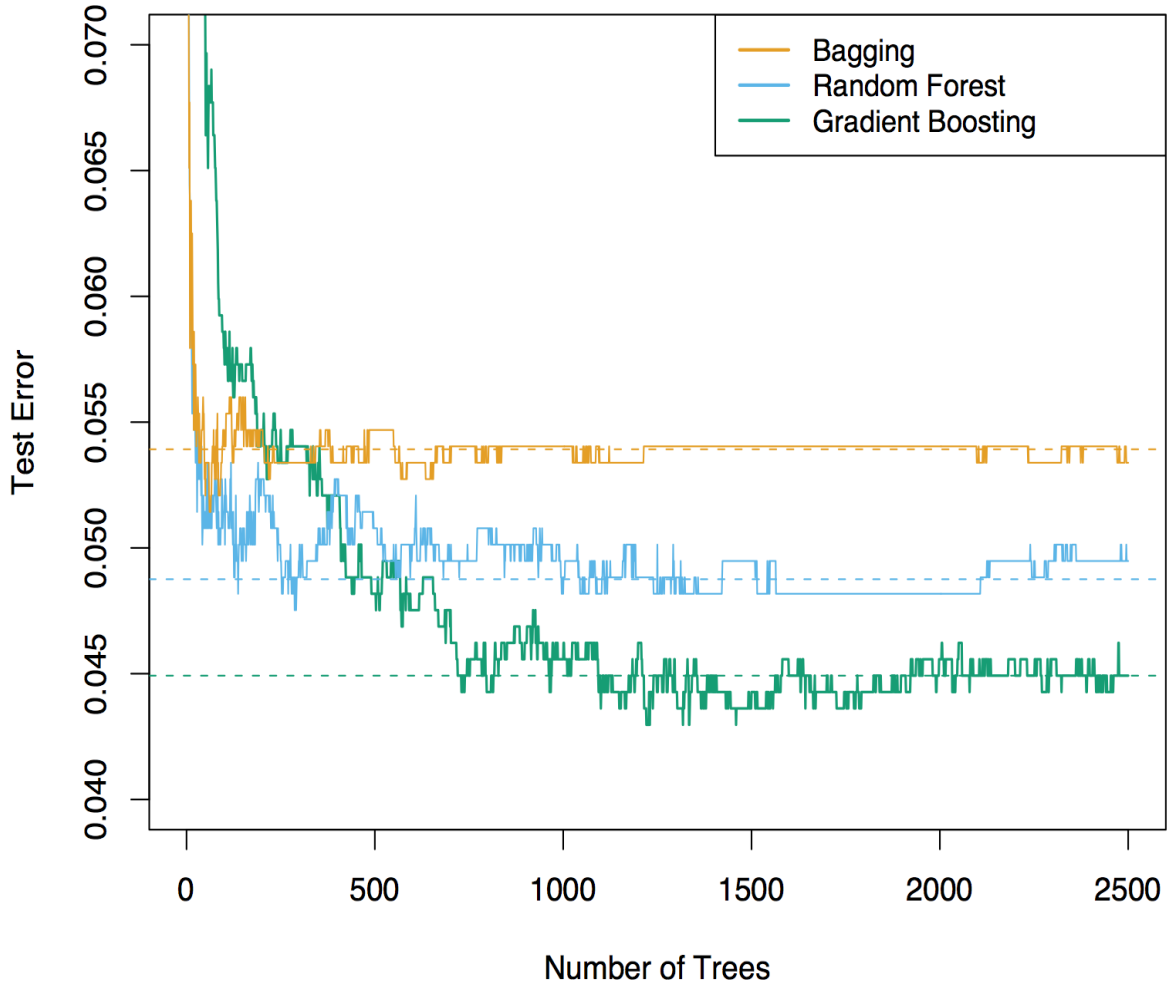
Independence helps

- A crowd of independent people give better advice than a crowd of very similar people!
- The *Random Forest* algorithm also grows trees on subsamples dataset
- To make trees less correlated than the simple bagging procedure, only subsets of predictors are allowed at each split.

Subsets of predictors

- p-dimensional predictors: $x = (x_1, \ldots, x_p)$
- At each split, **only $m < p$ random predictors are allowed**
- Random forest algorithm:
  1. Select m predictors at random from the p available predictors.
  2. Pick the best variable/split-point among the m.
  3. Split the node into two daughter nodes.
  4. Iterate until a full tree is grown
- In practice $m \approx \sqrt{p}$ is often a reasonable choice
- **[Exercise]** how can one find a good m in practice?

Performances

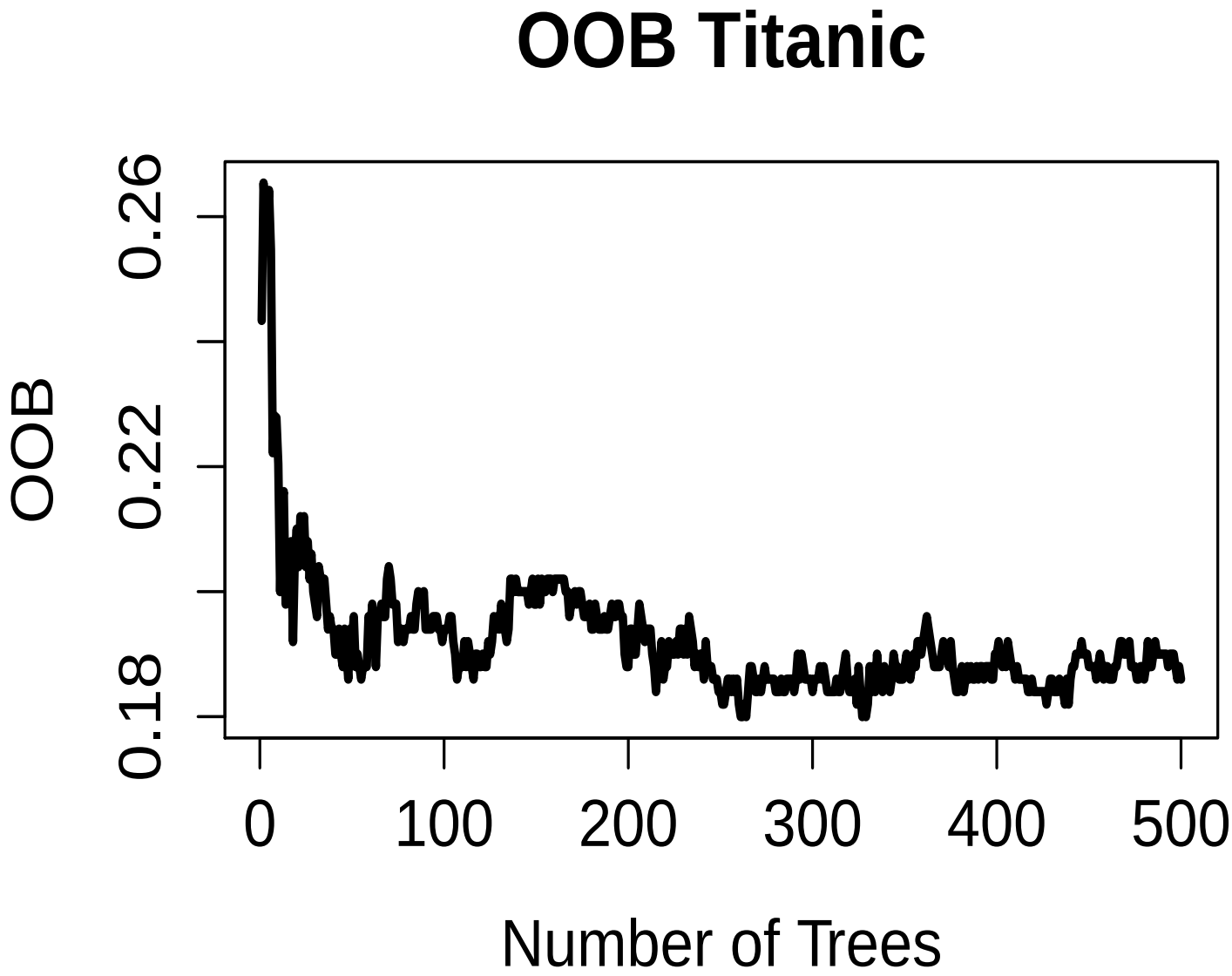


OOB: Out Of Bag estimate

- No need of cross validation to estimate the performance of a random forest!
- Regression Training dataset: $\{(x_1, y_1), \ldots, x_n, (y_n)\}$
- Random forest: $\{T_{S_1}, \ldots, T_{S_B}\}$
- Define J_k the indexes j such that $x_k \notin S_j$

$$\frac{1}{n} \sum_{k=1}^{n} \left\{ y_k - \frac{1}{|J_k|} \sum_{j \in J_k} T_{S_j}(x_k) \right\}^2$$

- [**Exercise**] how would define the OOB estimate for the error rate when random forest is used for classification?


OOB Titanic
OOB
0.18
0.22
0.26
0
100
200
300
400
500
Number of Trees

Variable Importance: permutation test approach

- If a predictor is not important, randomly shuffling this predictor in the training-set should not change much the OOB estimate
- [**Exercise**] how can one estimate the importance of each predictor after a run of the *Random Forest* algorithm?

Titanic: variable importance

Outline

Boosting fathers

Yoav Freund, Robert Schapire and Leo Breiman

Basic Idea

- Consider a regression problem with training data $\{x_i, y_i\}_{i=1}^N$
- Our objective is to construct a function F such that

$$y_i \approx F(x_i)$$

- One can first fit a tree T_0 to the data

$$y_i \approx T_0(x_i) \equiv F^{[0]}(x_i).$$

- The **residual** is defined as

$$R_i^{[0]} = y_i - F^{[0]}(x_i).$$

- One gets a first approximation $F^{[0]} = T_0$

Basic Idea

- To improve things, one can fit a second tree T_1 to the residuals

$$R_i^{[0]} \approx T_1(x_i)$$

- One obtains an hopefully better approximation

$$y_i \approx T_0(x_i) + T_1(x_i) \equiv F^{[1]}(x_i)$$

with new approximation function: $F^{[1]} = T_0 + T_1$

- Indeed, one can iterate!

$$F^{[n]} = T_0 + T_1 + \ldots + T_n$$

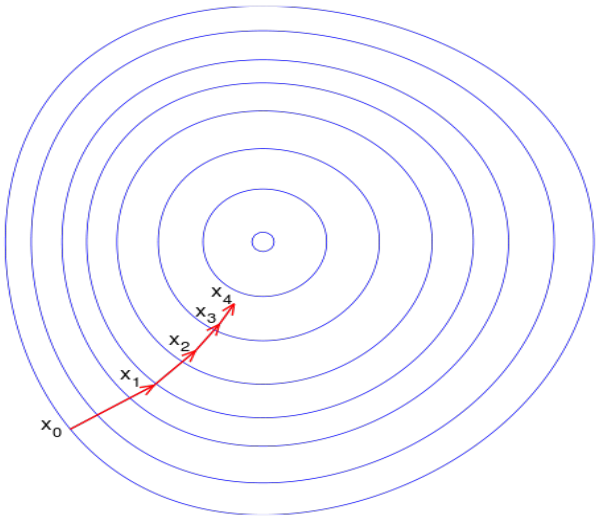
- **Questions:** Can one keep iterating? How to deal with classification? And more general loss function?

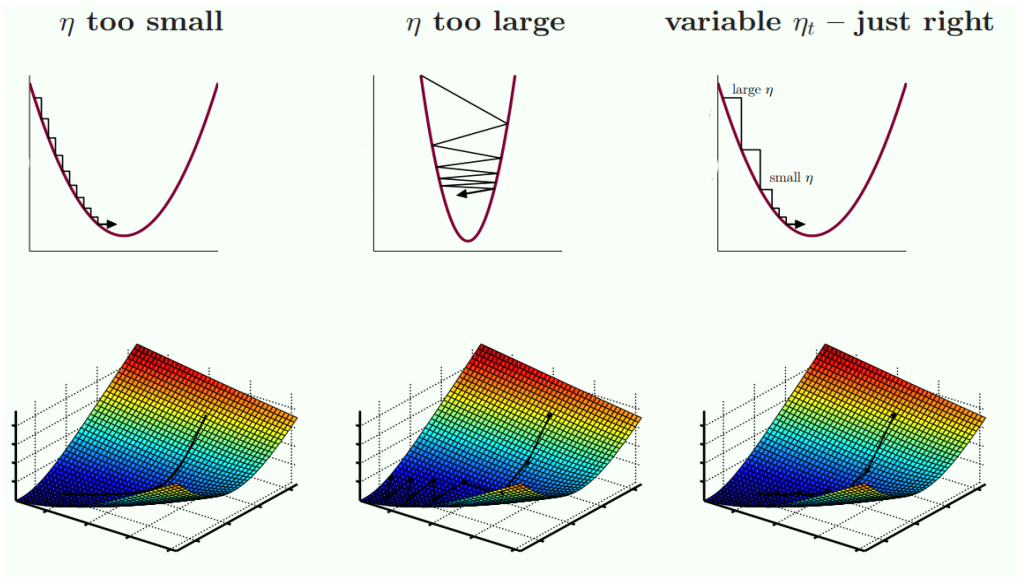
Gradient descent

- Function to minimize: $x \mapsto \Psi(x)$
- Current approximation x_n of the minimiser
- Gradient descent update:

$$x_{n+1} = x_n - \eta \nabla \Psi(x_n)$$

- Learning rate $\eta > 0$
  - the smaller the learning rate, the slower the convergence
  - too large a learning rate can lead to divergence





Loss function

- Training examples $\{x_i, y_i\}_{i=1}^N$
- Quality of a predictive function $F(\cdot)$ is measured by

$$\Psi(F) \equiv \sum_{i=1}^{N} L\left(F(x_i), y_i\right)$$

- [**Exercise**] Linear regression? Logistic regression? SVM?

Boosting

- Training examples $\{x_i, y_i\}_{i=1}^N$
- Try to minimise

$$\Psi(F) \equiv \sum_{i=1}^{N} L\left(F(x_i), y_i\right)$$

by looking (greedily) for a function of the form

$$F = T_0 + \alpha_1 T_1 + \alpha_2 T_2 + \ldots$$

- **General idea:**
  - Fit a first tree: $F^{[0]} \equiv T_0$
  - Try to improve performances by adding another tree: $F^{[1]} = T_0 + \alpha_1 T_1$
  - Try to improve performances by adding another tree: $F^{[2]} = T_0 + \alpha_1 T_1 + \alpha_2 T_2$
  - iterate...

Boosting: follow the gradient

- Suppose that the current estimate if $F^{[m]}$
- First order expansion

$$\Psi\left(F^{[m]}+g\right) \approx \Psi\left(F^{[m]}\right) + \sum_{i=1}^{N} \nabla_x L\left(F^{[m]}(x_i), y_i\right) \times g(x_i)$$
$$= \Psi\left(F^{[m]}\right) + \langle \nabla\Psi, g \rangle$$

- **Idea**: fit a tree T_{m+1} to the dataset

$$\left\{x_i, \nabla_x L\left(F^{[m]}(x_i), y_i\right)\right\}_{i=1}^{n}$$

- The next approximation is defined as

$$\mathbf{F^{[m+1]}} = \mathbf{F^{[m]}} - \eta\, \mathbf{T_{m+1}}$$

for a learning rate η.

Boosting: regression case

- Squared error loss function

$$L(F(x), y) = \frac{1}{2}(f(x) - y)^2$$

- **[Exercise]** write the boosting algorithm.
- A more robust alternative (Laplace loss)

$$L(F(x), y) = |f(x) - y|$$

- **[Exercise]** write the boosting algorithm.

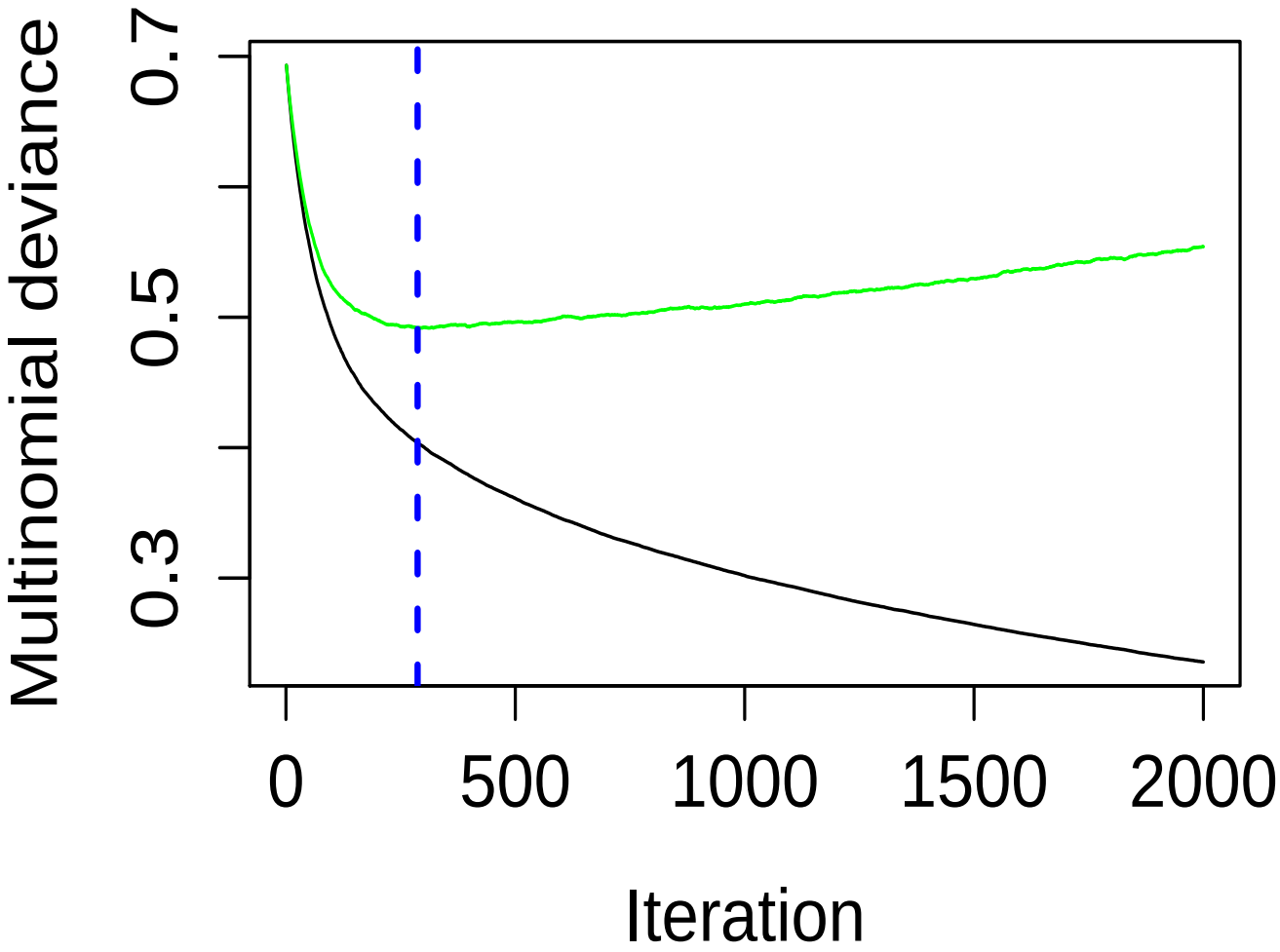
Boosting: classification

- Training examples $\{x_i, y_i\}_{i=1}^N$ with $y_i \in \{-1, 1\}$.
- Suppose that one is trying to approximate the log-odd function F
- A possible loss function is given by the negative log-likelihood

$$L\left(F(x), y\right) = \log\left(1 + e^{-y\,F(x)}\right)$$

- **[Exercise]** write the boosting algorithm.

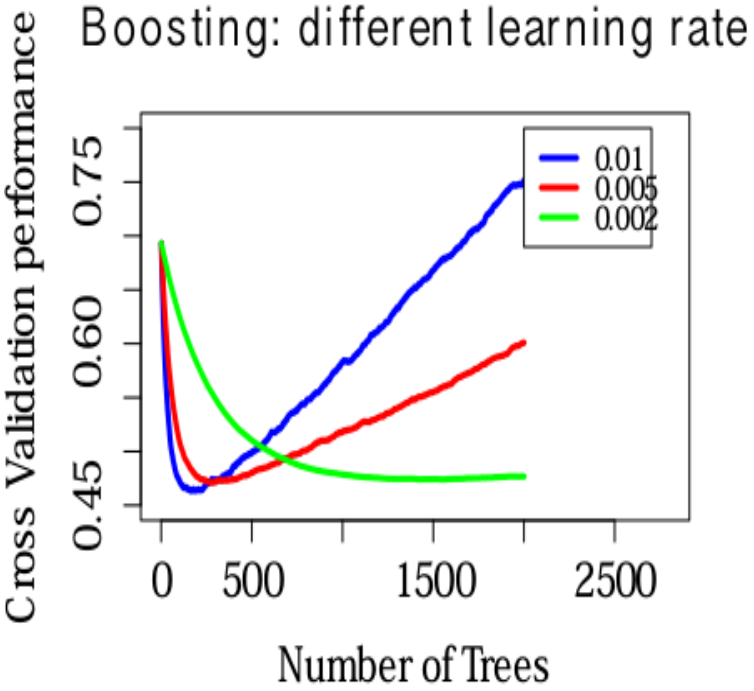
Optimal number of trees

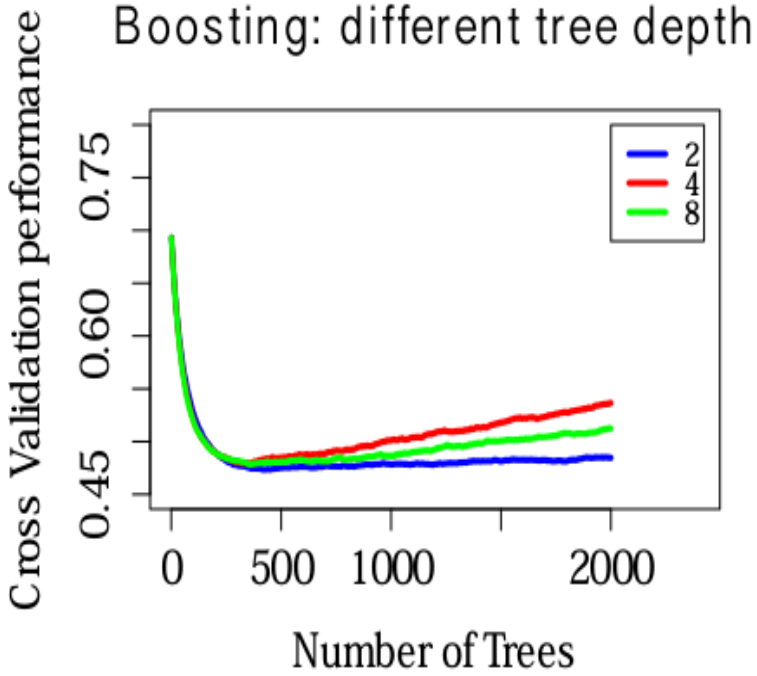


Tuning parameters

- Depth of the tree: small values (≤ 8) is usually recommended
- Learning rate
- Proportion of training example

Tuning parameters





Variable importance

```
> summary(titanic.boosting)
                var   rel.inf
Sex             Sex 33.064059
Fare           Fare 20.902456
Age             Age 20.178804
Pclass       Pclass 19.097227
Embarked Embarked  3.032399
SibSp         SibSp  2.479412
Parch         Parch  1.245643
```