Erasmus School of Economics

Machine Learning

FEM31002

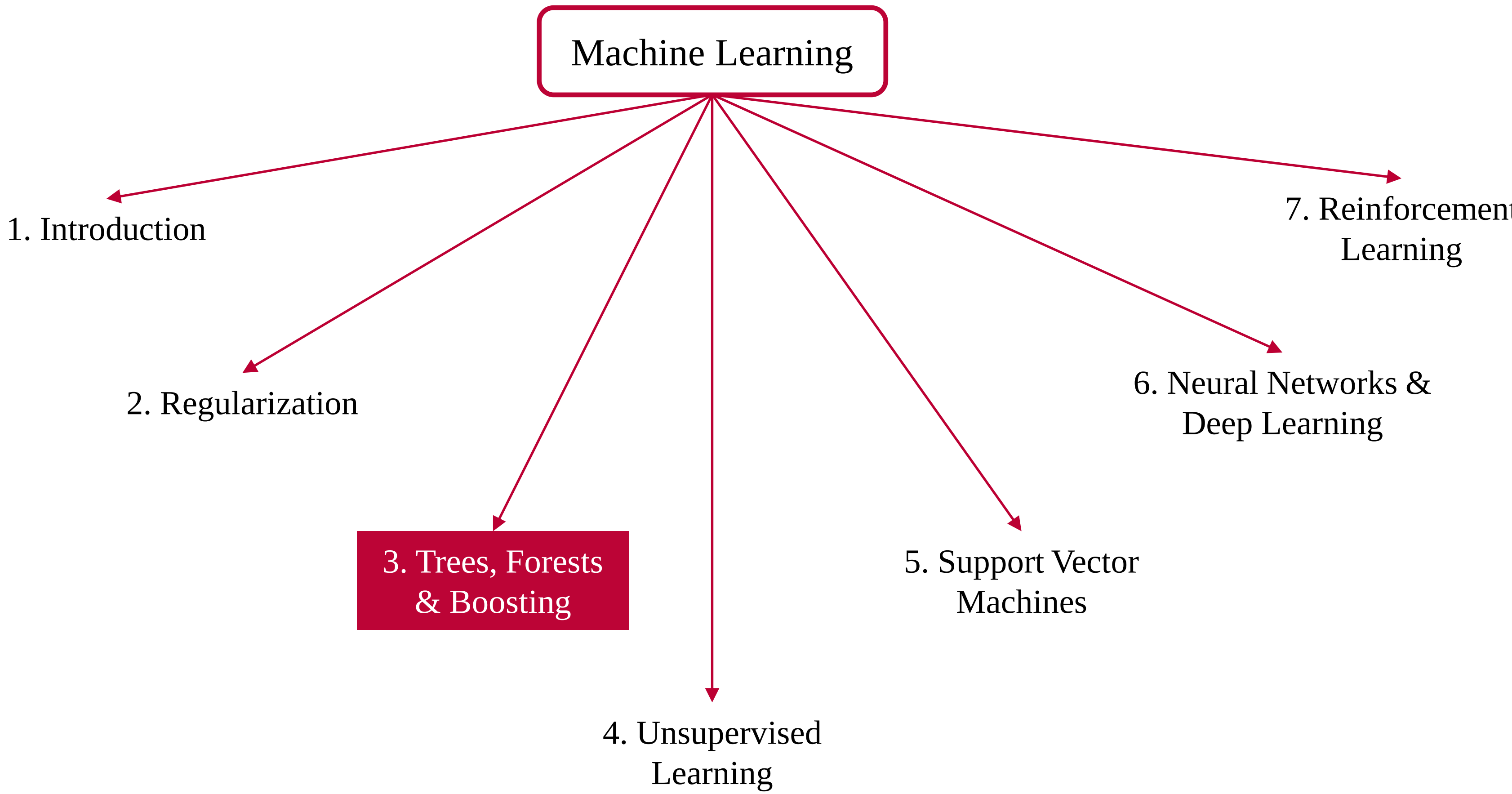
Trees, Forests and Boosting

Ilker Birbil

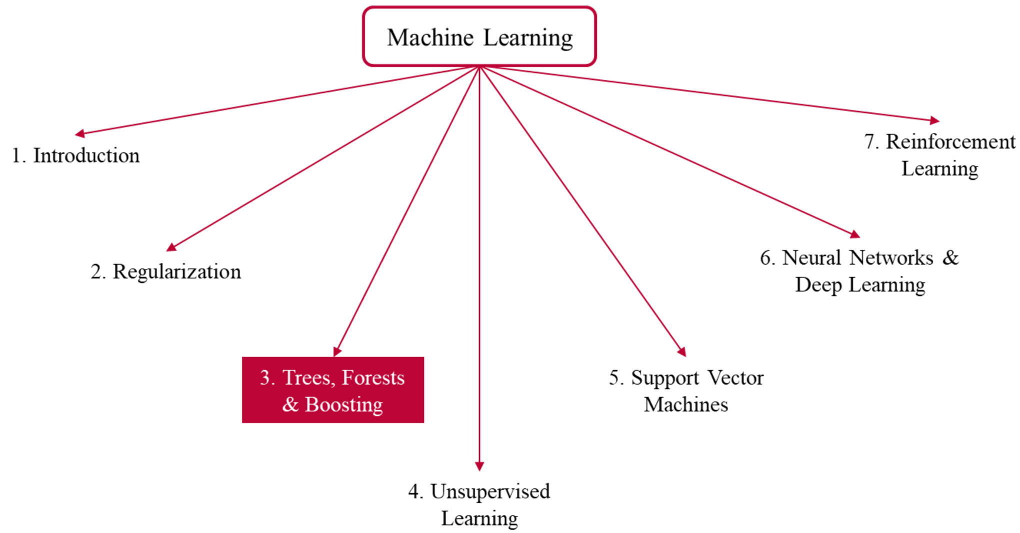
birbil@ese.eur.nl

ERASMUS UNIVERSITEIT ROTTERDAM

Outline


Machine Learning
1. Introduction
2. Regularization
3. Trees, Forests & Boosting
4. Unsupervised Learning
5. Support Vector Machines
6. Neural Networks & Deep Learning
7. Reinforcement Learning

Outline

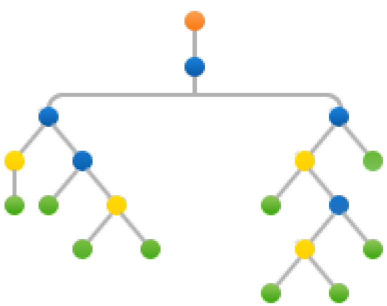


- Decision Trees
- Random Forest
- Boosting
- Interpretability

Tree-based Methods

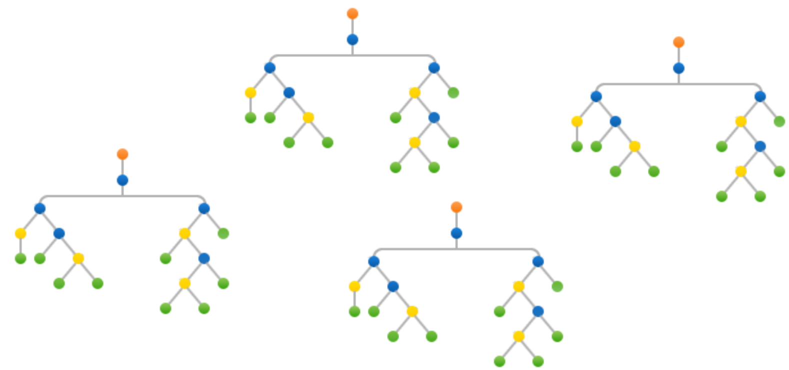
Objective: Divide the predictor space into a number of simple regions

Regression Trees
Classification Trees

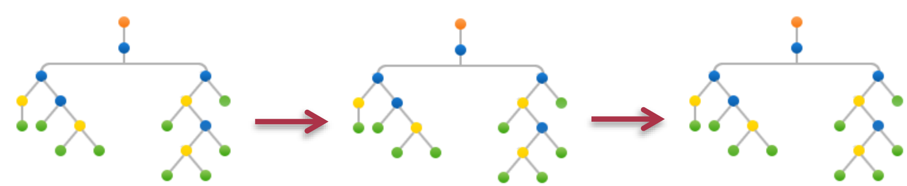


CART
C4.5

Bagging & Random Forest

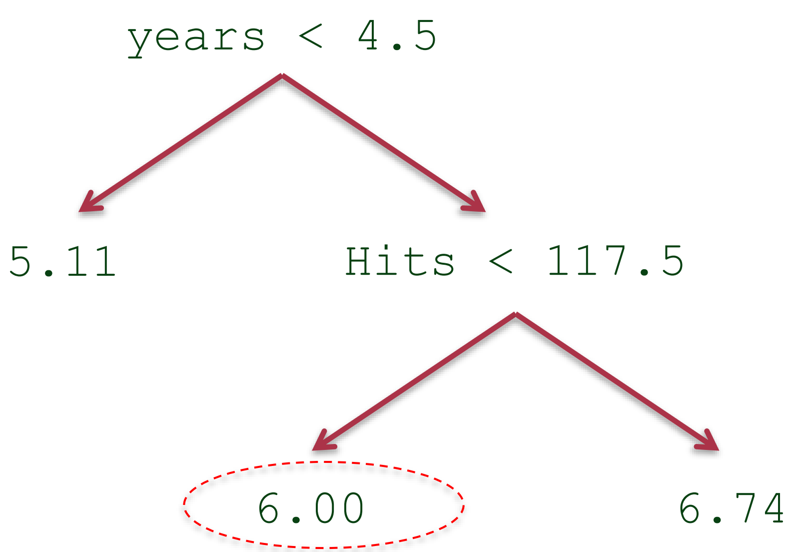


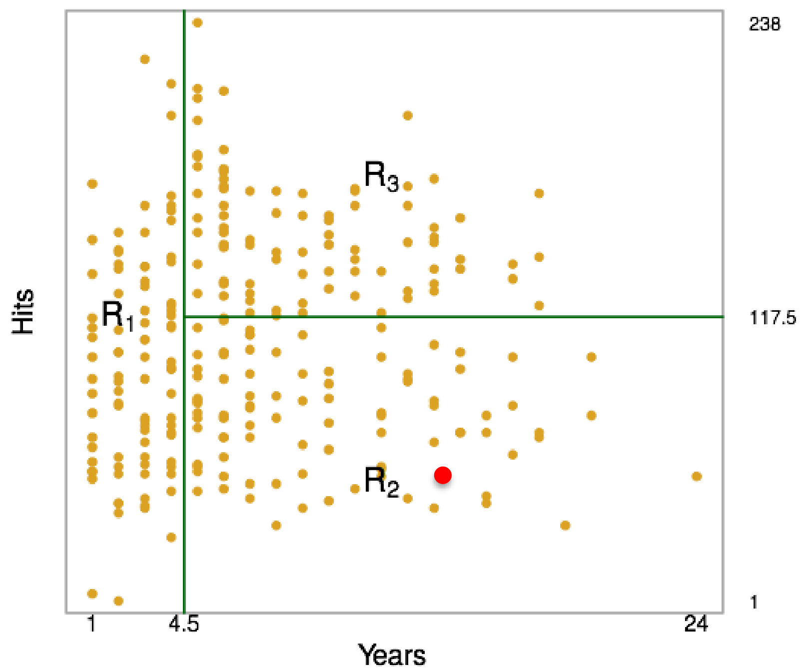
Boosting



Example Tree

Predicting **the log salary** of a player as a function of the **number of hits** and **the years of experience**





Regression Trees

$$X_1, X_2, \ldots, X_p \xrightarrow{\text{distinct and nonoverlapping regions}} R_1, R_2, \ldots, R_J$$

Prediction: Mean of the response values for the training observations in R_j

$R_1, R_2, \ldots, R_J$ **?**

Goal: Finding the regions such that sum of squares is minimized

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

$\hat{y}_{R_j}$: mean response within R_j

Regression Trees

Recursive Binary Splitting

$$R_1(j,s) = \{X | X_j < s\} \qquad R_2(j,s) = \{X | X_j \geq s\}$$

Find *j* and *s* that minimizes

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Stop

when each terminal node has a very small number of observations

decrease in sum of squares is below a threshold (myopic)

Regression Trees

Tree Pruning

Goal: Avoiding overfitting with a fully grown or large tree
(Selecting a subtree that leads to a lowest test error rate)

Minimize the cost complexity criterion:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

T : subtree

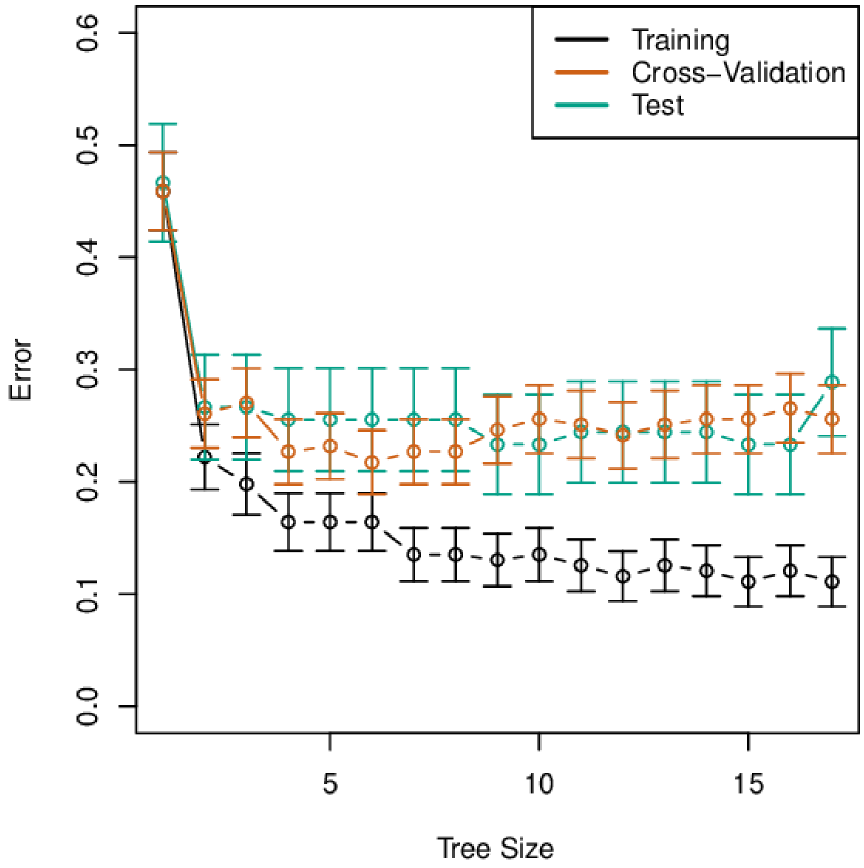
$|T|$: number of terminal nodes in T

R_m : region corresponding to the mth **terminal node**

α : tuning hyperparameter

$\alpha \uparrow \qquad |T| \downarrow$

Use k-fold cross validation to choose α


Training
Cross−Validation
Test
Error
Tree Size
0.0
0.1
0.2
0.3
0.4
0.5
0.6
5
10
15

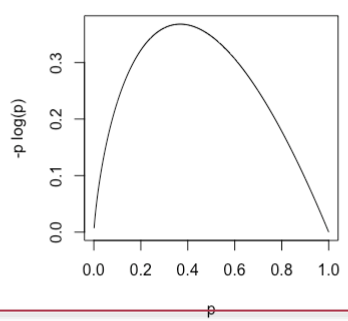
Classification Trees

Similar to a regression tree but uses different error measures based on *purity* of a region – classification is done with **majority voting**

$\hat{p}_{mk}$: proportion of class-*k* training observations in the *m*th region

Classification Error Rate

$$E = 1 - \max_k \{\hat{p}_{mk}\}$$

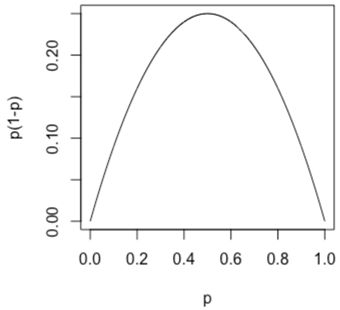


(Cross) Entropy

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

Gini Index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



Other Points

- **Categorical variables:** Need to consider all subsets of the possible values

  a) Binary output – order the categories according to the proportion falling in one class, then split as if it is an ordered predictor (optimal)

  b) Quantitative outcome – square error loss: Same as a)

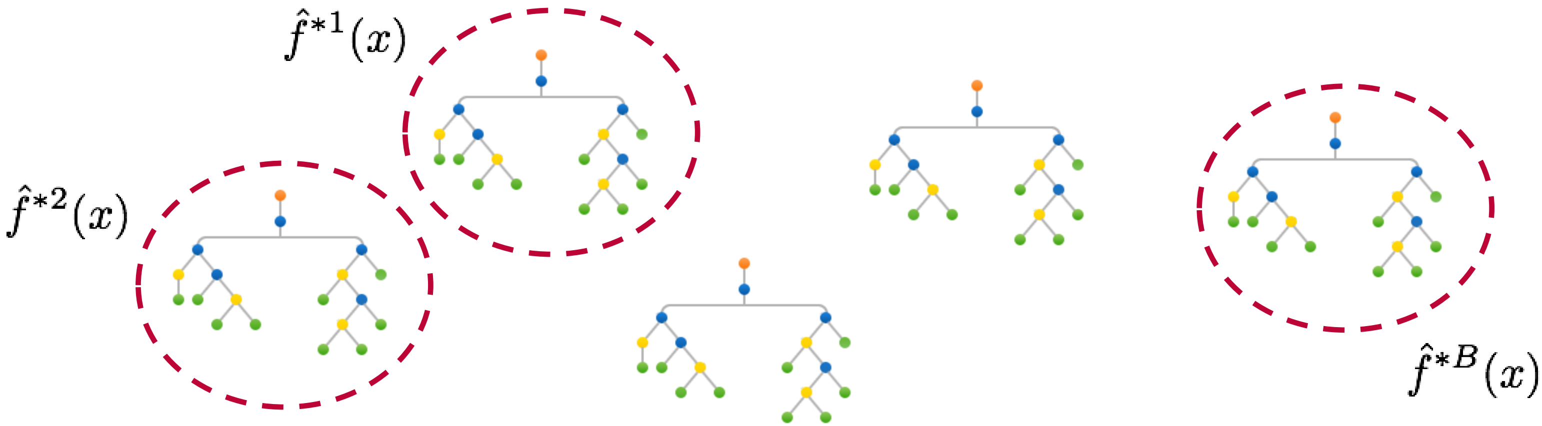
  c) Multi-category output – Trick in a) does not work (approximations)

- **Instability:** Trees have high variance due their hierarchical structure

- **CART alternatives:** ID3, C4.5, C.5.0

Bagging

Goal: Use bootstrapping to grow separate *deep* trees and average all the predictions (regression) or apply majority rule (classification) to reduce the variance



Regression

B : number of separate training sets

$\hat{f}^{*b}(x)$: prediction obtained with the *b*th training set

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

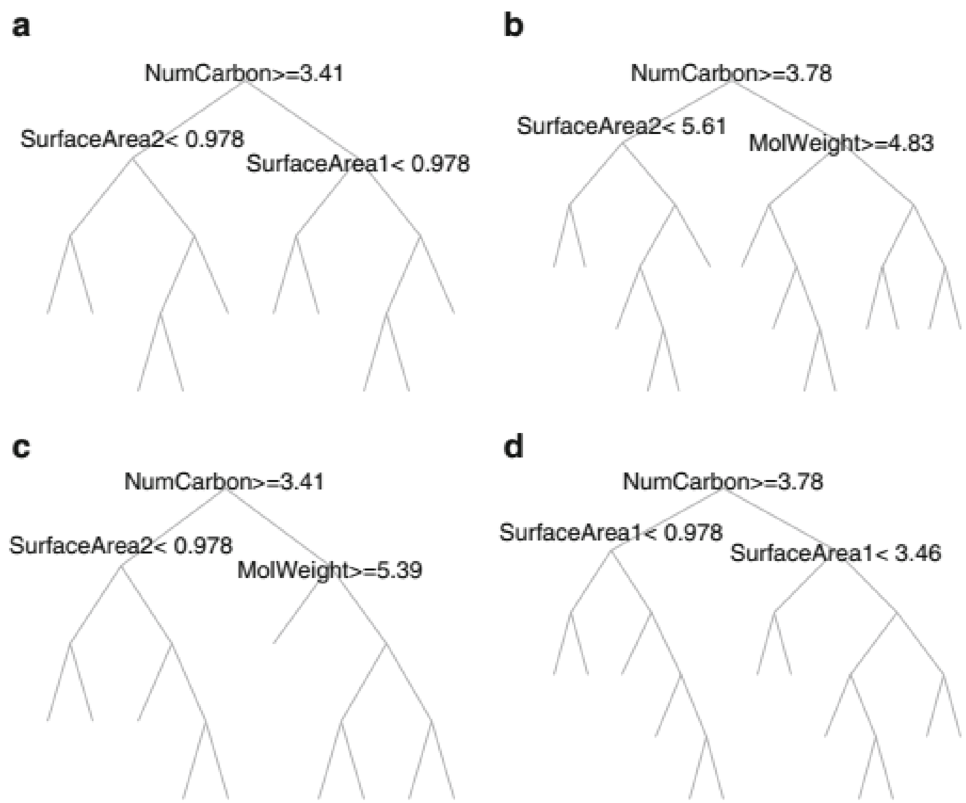
Classification

Apply majority voting

Random Forest

Goal: Smart bagging to *de-correlate* the trees – only a random sample of *m* predictors is chosen as candidates for splitting

Why?

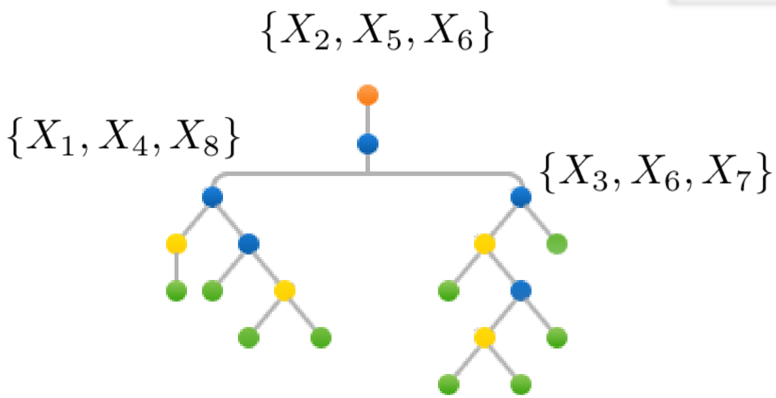


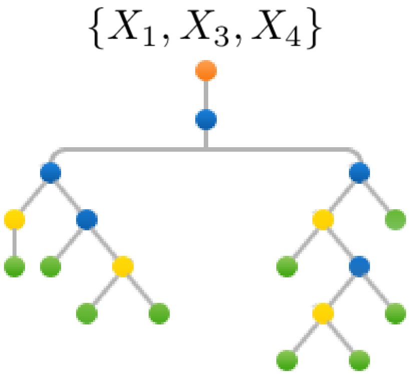
*

**Applied Predictive Modeling*, M. Kuhn, K. Johnson., Springer, 2013, pg. 195.

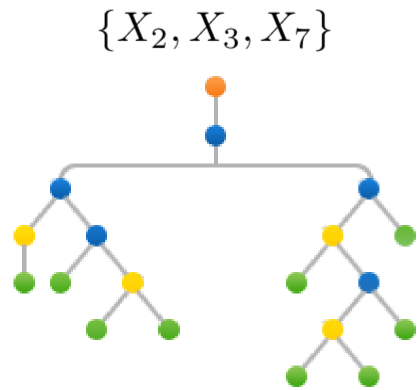
Random Forest

$$\boxed{m \approx \sqrt{p}}$$





random subsets of size *m*



Regression

B : number of grown trees

$\hat{f}^{*b}(x)$: prediction obtained with the *b*th tree

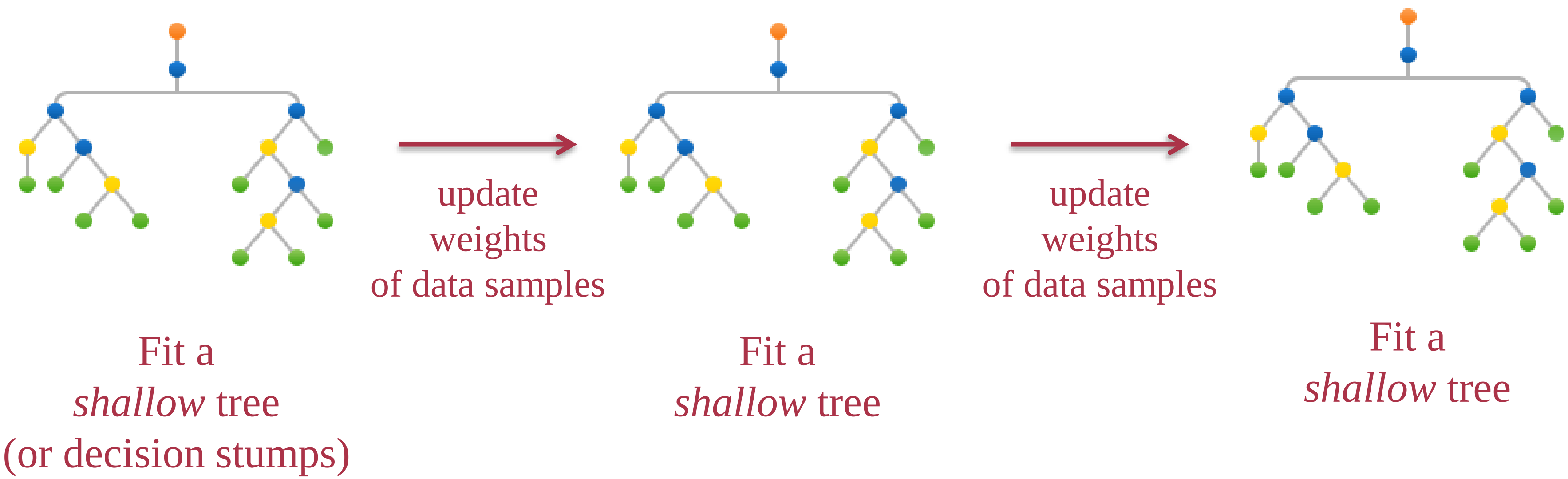
$$\hat{f}_{\mathrm{rf}}^{B}(x) = \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Classification

Apply majority voting

Boosting

Idea: Combining mediocre classifiers sequentially to boost their collective performance through weighted data sampling



Update Rule: Give more weights to incorrectly classified samples

AdaBoost – Binary Classification {+1, -1}

Each sample has the same starting weight (*1/n*)

for *k=1 to K* **do**

Fit a tree with *d* splits using the weighted samples and compute misclassification error (ϵ_k)

Compute stage weight value $\ln \frac{1-\epsilon_k}{\epsilon_k}$

Update the sample weights – give more weights to incorrectly classified samples

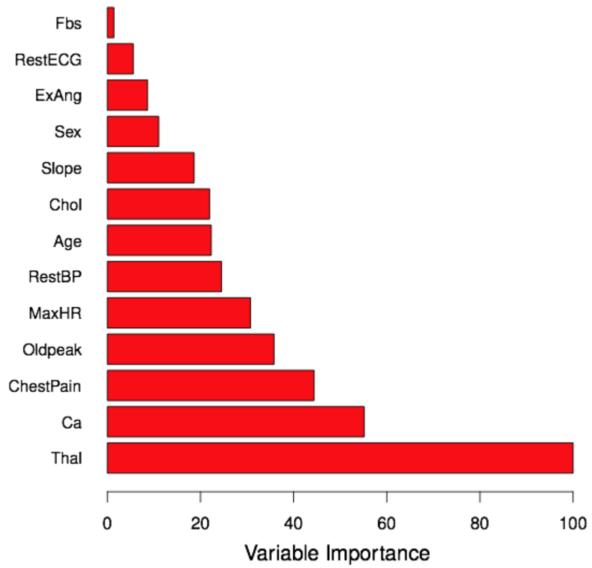
end

Compute the boosted classifier's prediction for each sample by multiplying the *k*th stage value by the *k*th model prediction and add these quantities across k. If the sum is positive, then classify the sample as +1 otherwise as -1

Details **for general classifiers** are in the supplementary note on Canvas

Notes on (Tree-based) Ensemble Methods

- Both boosting and bagging can be applied to different learning methods
- While bagging allows direct parallel implementation, the sequential structure of boosting prevents parallelization
- Ensemble methods cause loss of interpretability
- Variable importance plots can be used



Variable importance is computed by sorting the predictors according to the mean decrease they achieve in Gini index or entropy

(assign 100 to the largest and scale the rest)