ML TECHNIQUES

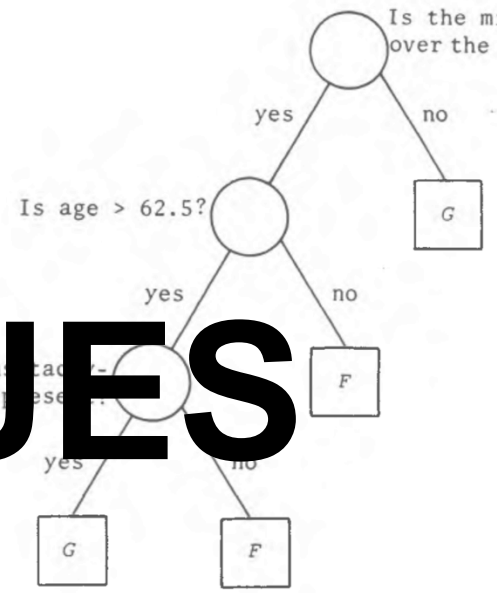


FIGURE 1.1

TREE-BASED METHODS

Instituto Balseiro - 06/11/2019

Luis G. Moyano
lgmoyano@gmail.com

UNCUYO
UNIVERSIDAD NACIONAL DE CUYO

FCEN
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
Naturaleza - Ciencia - Humanismo

CONICET

FiEstIn

Why trees?

- What is a tree?
- Why bother with trees?
- What can we do with them?

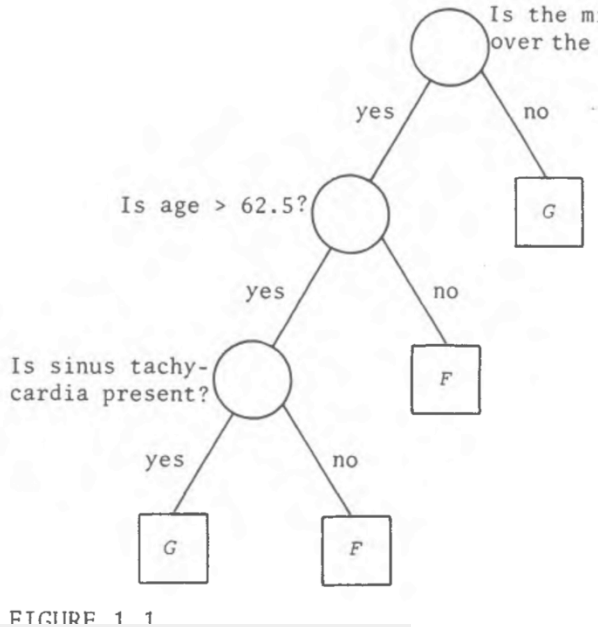
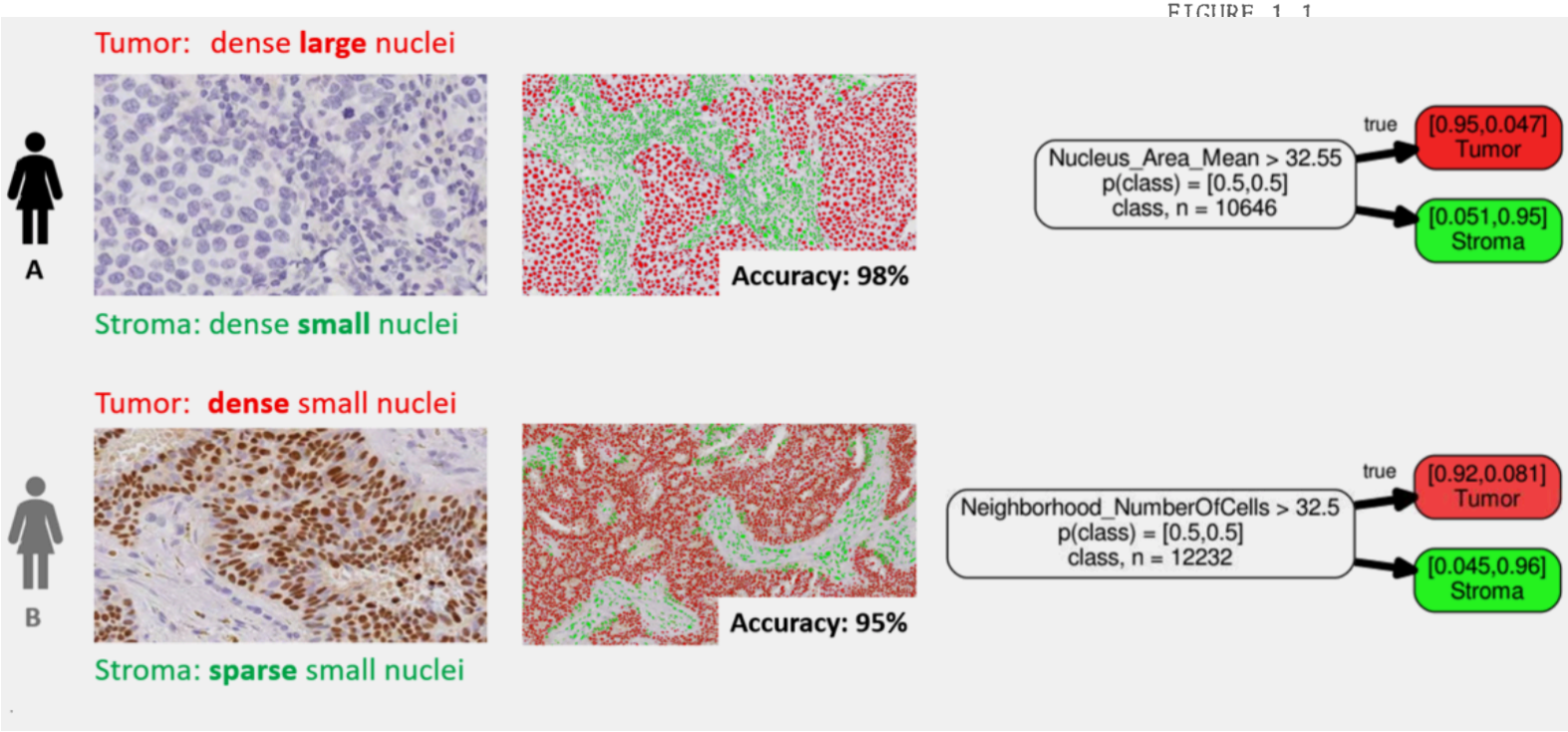


FIGURE 1.1



No free lunch theorem

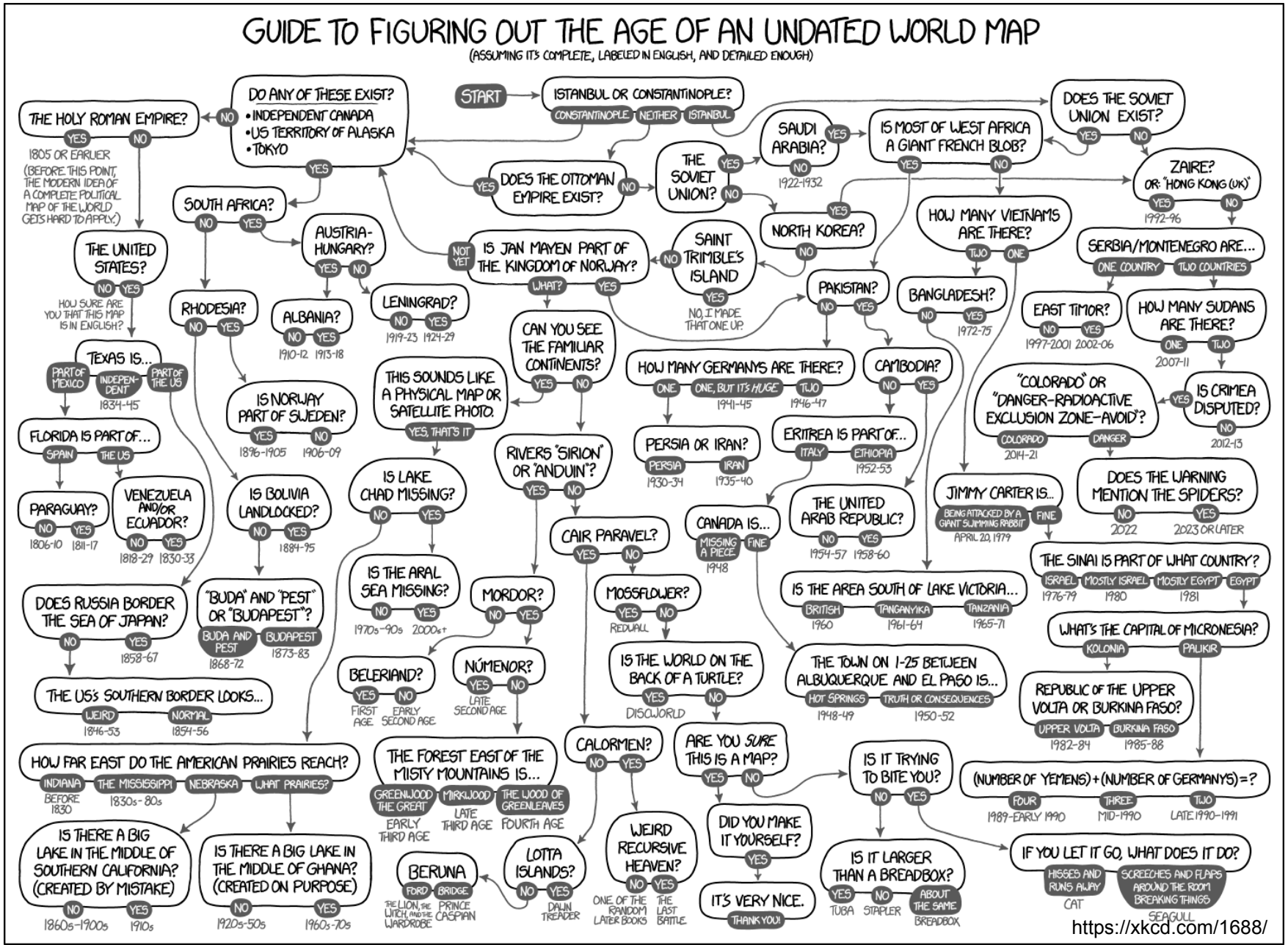
- Always check your assumptions before relying on a model or search algorithm.
- There is no “super algorithm” that will work perfectly for all datasets.

Wolpert, D.H., Macready, W.G. (1995-1997)

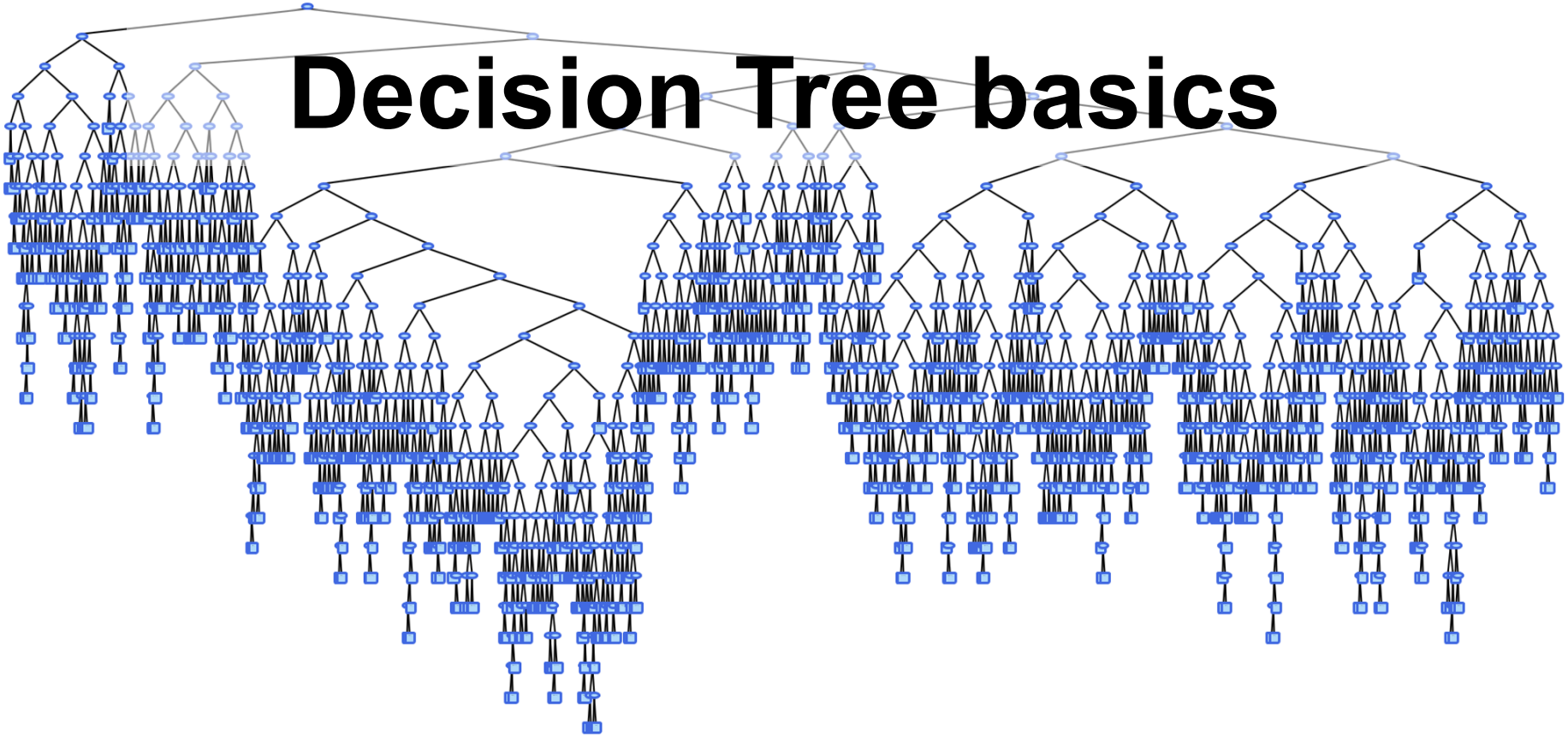
Bibliography (mostly)

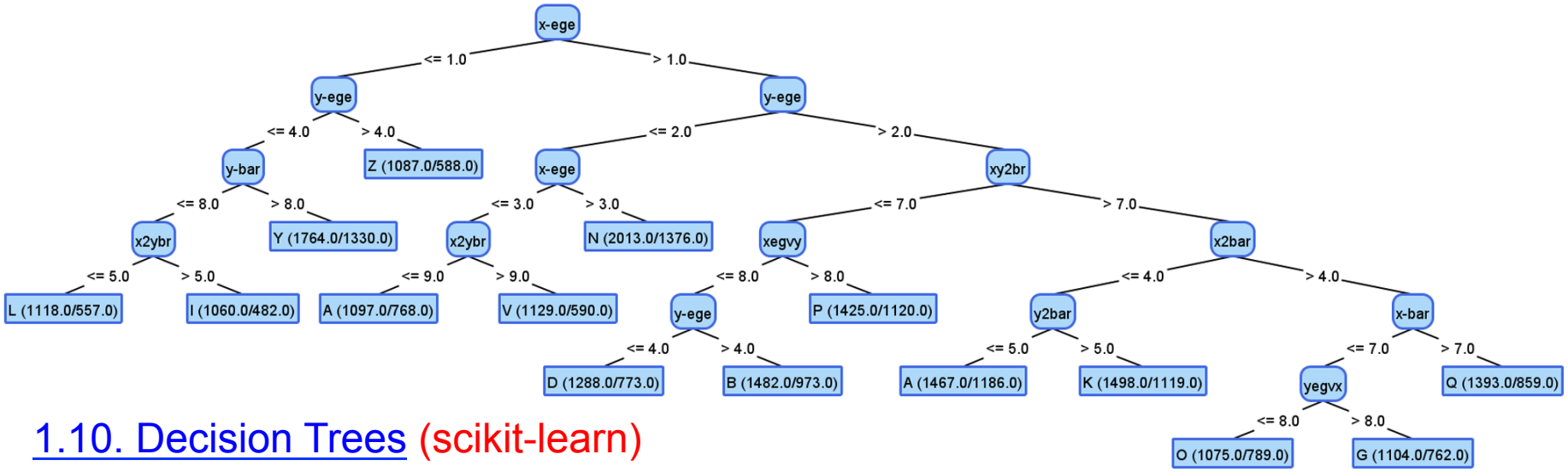
- *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Hastie et al, 2017).
- *An Introduction to Statistical Learning with Applications in R* (James et al., 2013).
- *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow Concepts, Tools, and Techniques to Build Intelligent Systems* (Gerón, 2019).

Tree-Based Methods



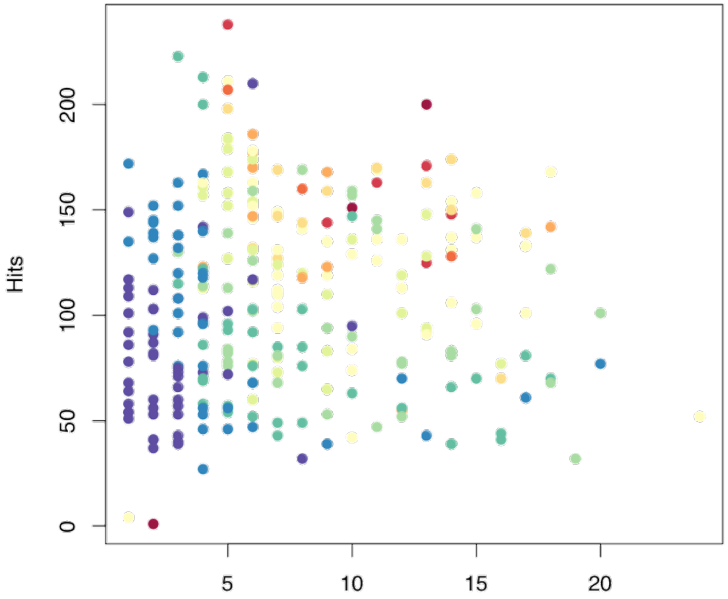
Decision Tree basics





1.10. Decision Trees (scikit-learn)

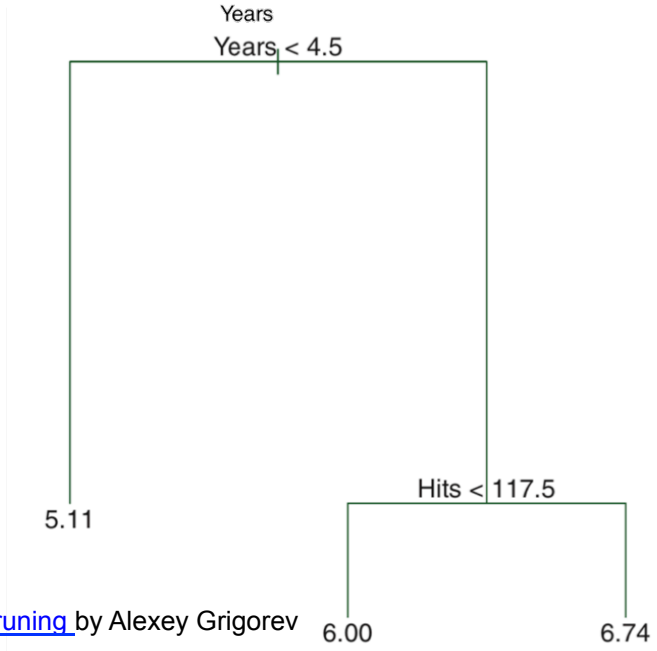
Building a regression tree

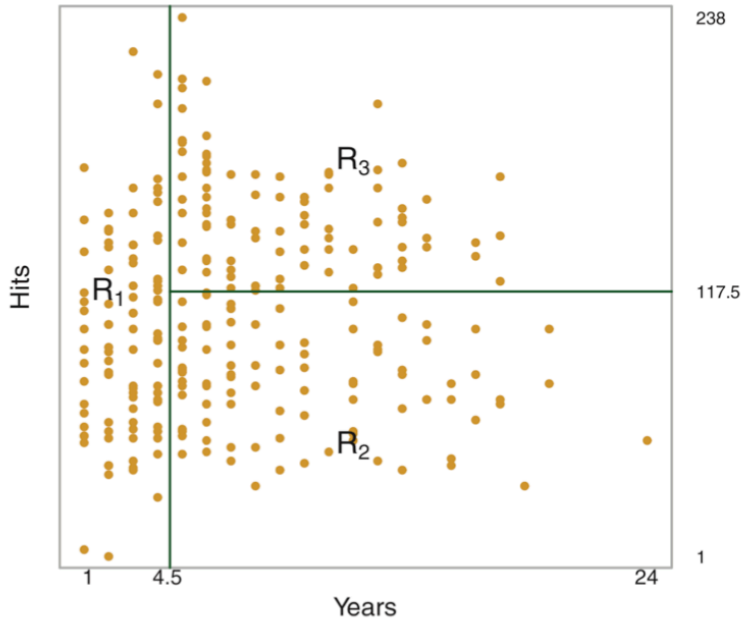


Baseball salary data
Salary is color-coded
(in Kelvins).

tree.DecisionTreeRegressor
tree.DecisionTreeClassifier

Overall, the tree stratifies or segments the players into three regions of predictor space: $R_1 = \{X \mid \texttt{Years} < 4.5\}$, $R_2 = \{X \mid \texttt{Years} >= 4.5, \texttt{Hits} < 117.5\}$, and $R_3 = \{X \mid \texttt{Years} >= 4.5, \texttt{Hits} >= 117.5\}$.

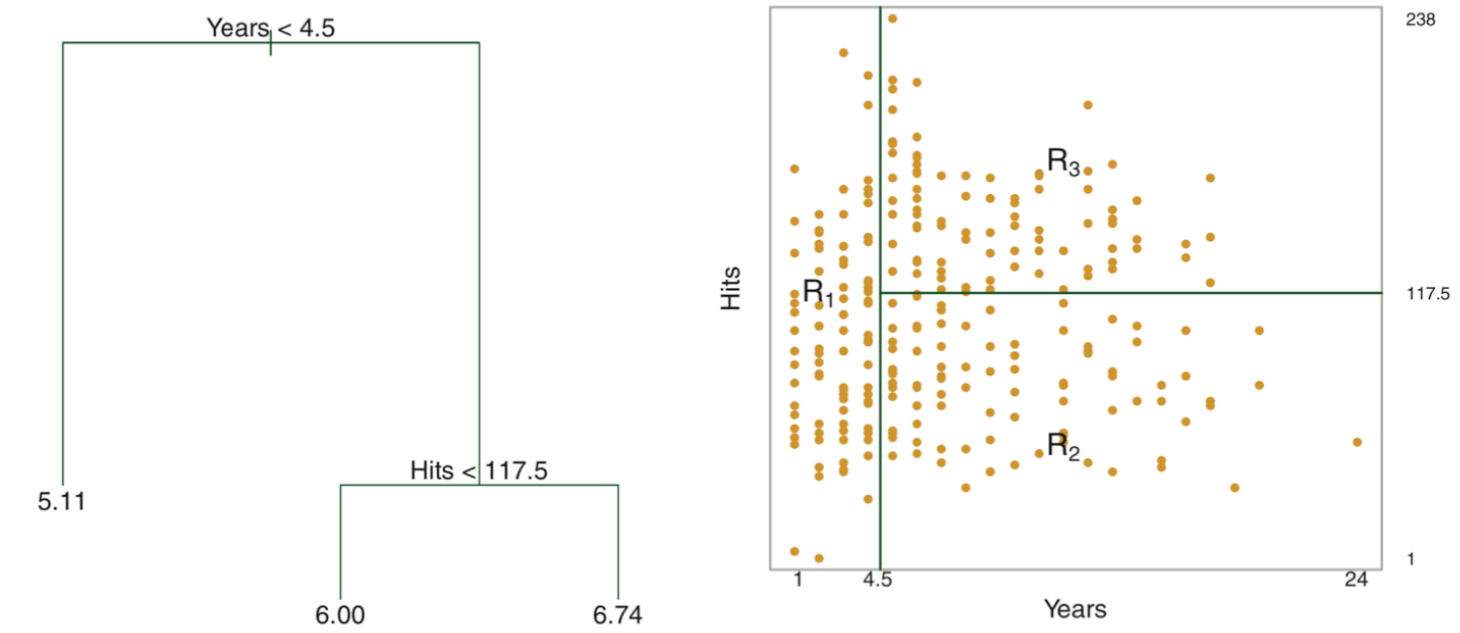




cost-complexity pruning by Alexey Grigorev

Recursive binary splitting

1. Top-down: splits predictor space ***recursively***



2. Greedy: best split at each step (horizon effect)

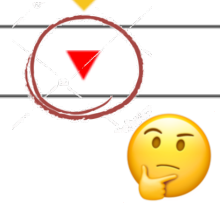
$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

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

3. Stop by some criterion

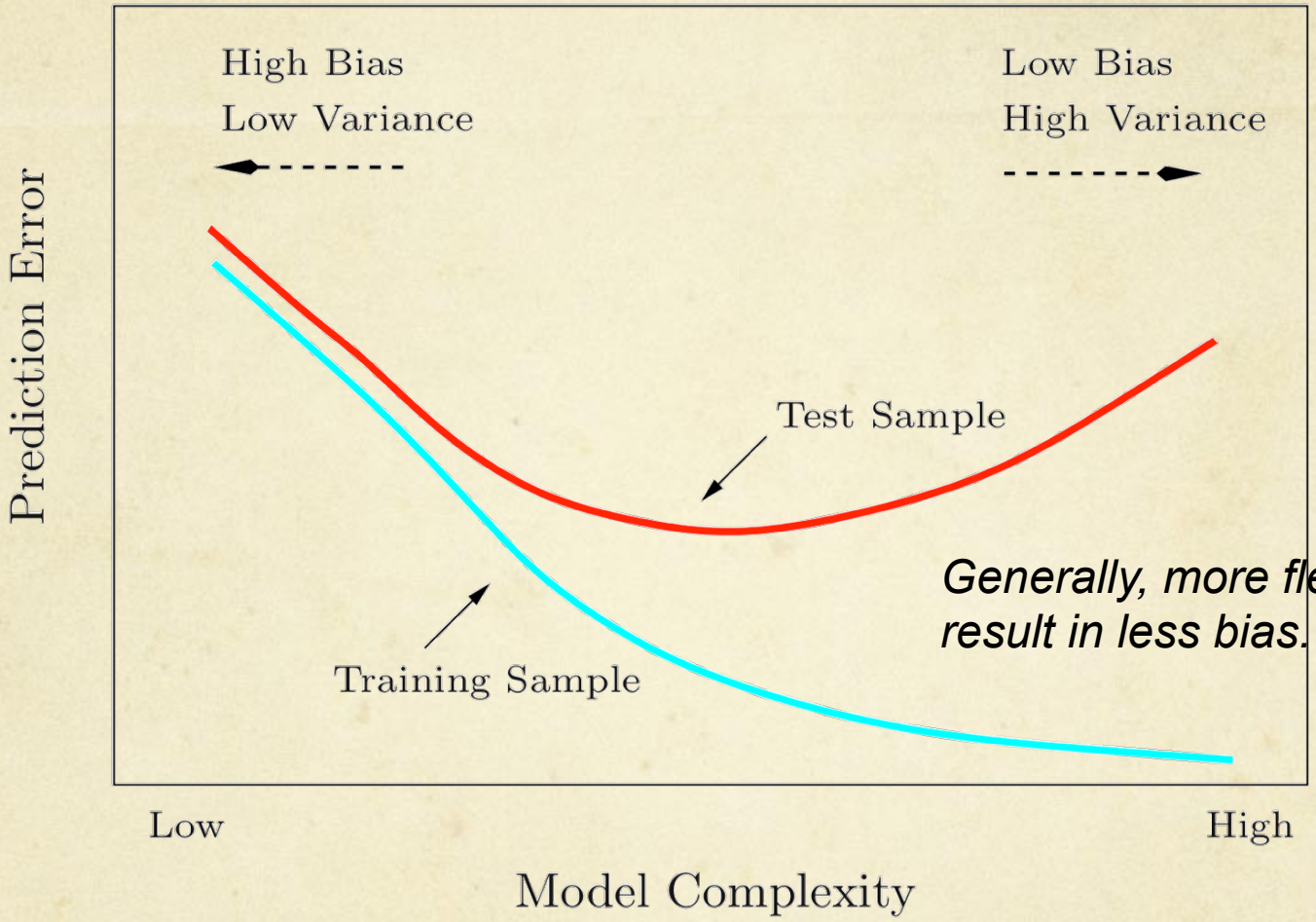
TABLE 10.1. *Some characteristics of different learning methods. Key:* ▲ *= good,* ◆ *=fair, and* ▼ *=poor.*

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrelevant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲



Review of some ML concepts

The Bias–Variance Tradeoff



Generally, more flexible methods result in less bias.

$$\mathrm{E}\left[\left(y-\hat{f}\left(x\right)\right)^2\right]=\left(\mathrm{Bias}\left[\hat{f}\left(x\right)\right]\right)^2+\mathrm{Var}\left[\hat{f}\left(x\right)\right]+\sigma^2$$

where

$$\mathrm{Bias}\left[\hat{f}\left(x\right)\right]=\mathrm{E}\left[\hat{f}\left(x\right)\right]-\mathrm{E}\left[f(x)\right]$$

and

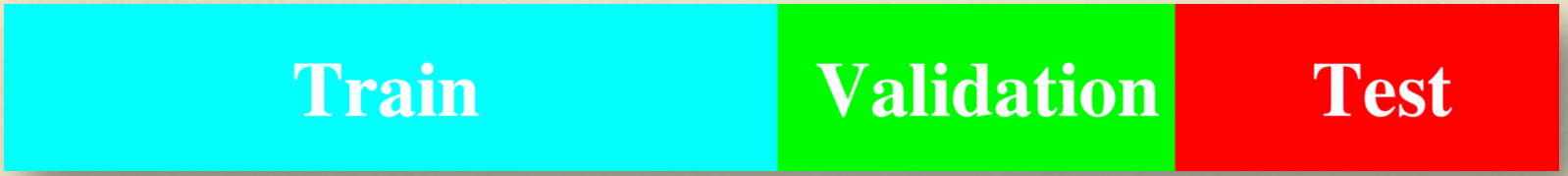
$$\mathrm{Var}\left[\hat{f}\left(x\right)\right]=\mathrm{E}[\hat{f}\left(x\right)^2]-\mathrm{E}[\hat{f}\left(x\right)]^2.$$

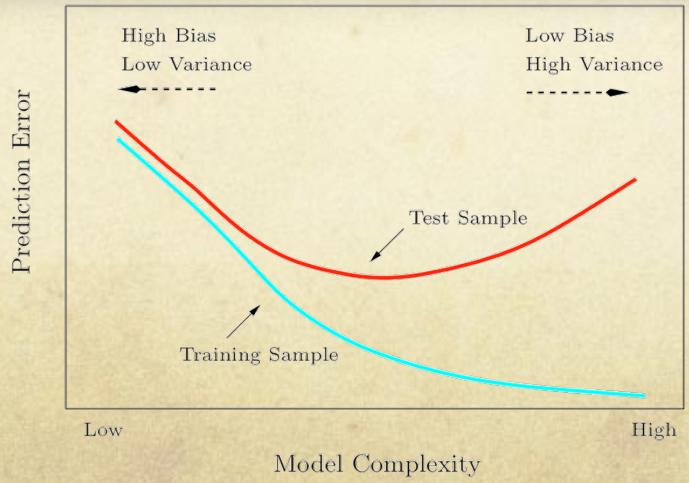
Single estimator versus bagging: bias-variance decomposition

Assessing Model Accuracy

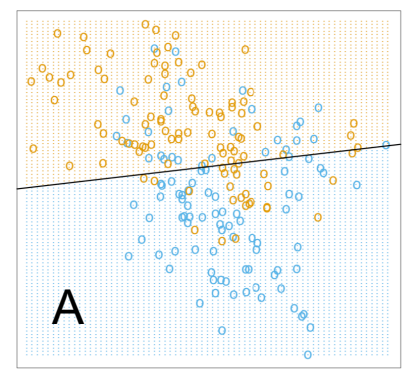
- Model selection: estimating the performance of different models in order to choose the best one.

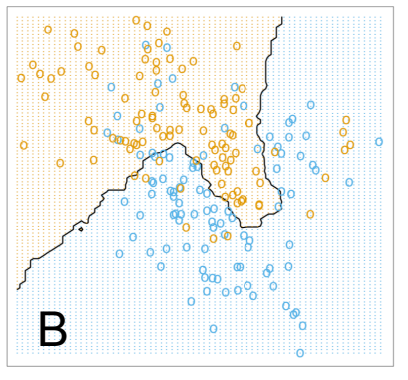
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.

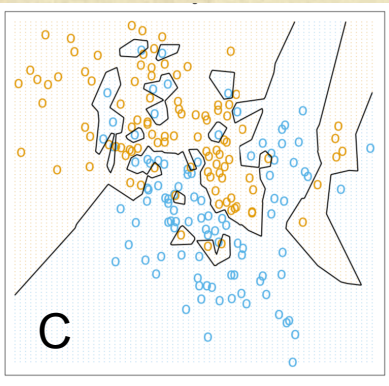


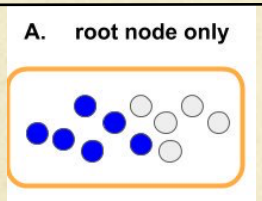


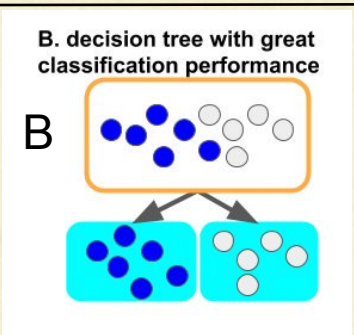
Overfitting

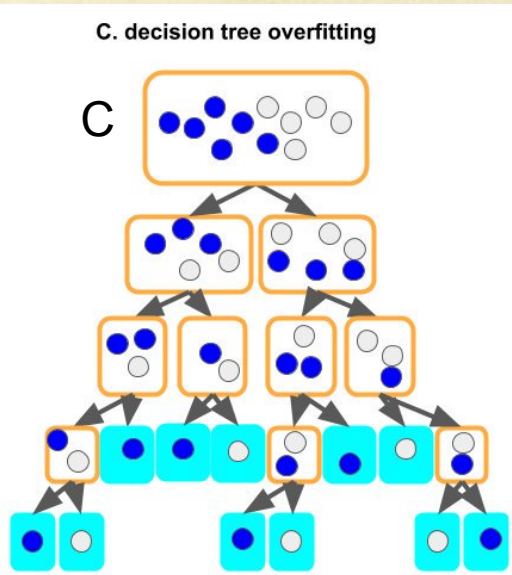


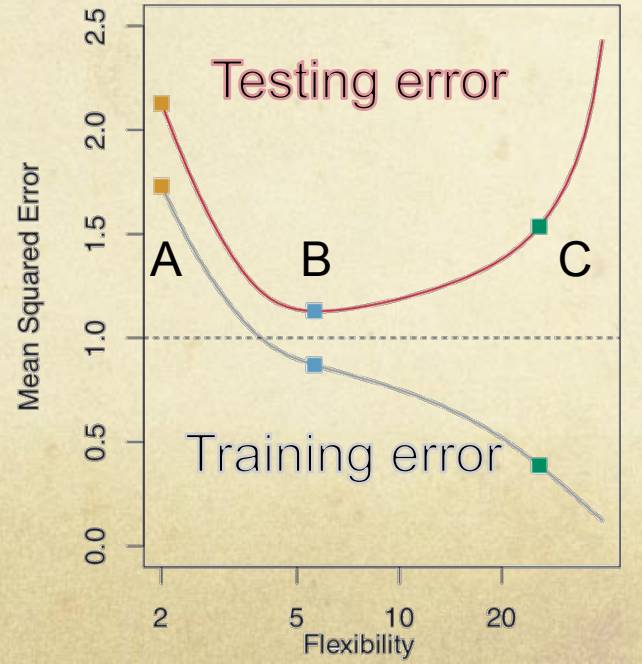








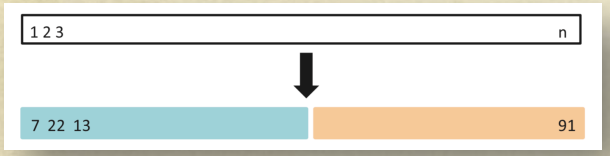


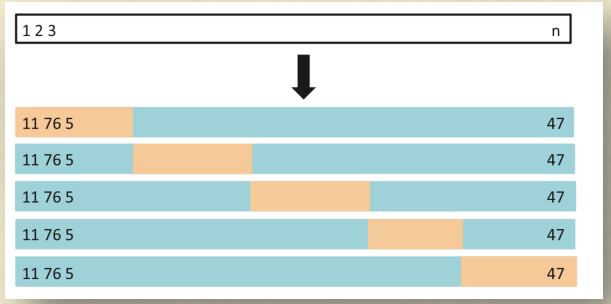


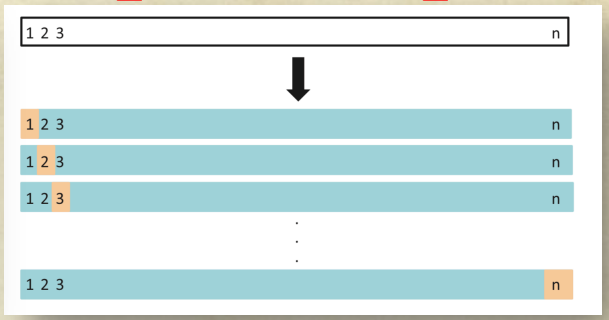
Resampling methods

model_selection.LeaveOneOut

model_selection.Kfold
model_selection.cross_validate







Validation set

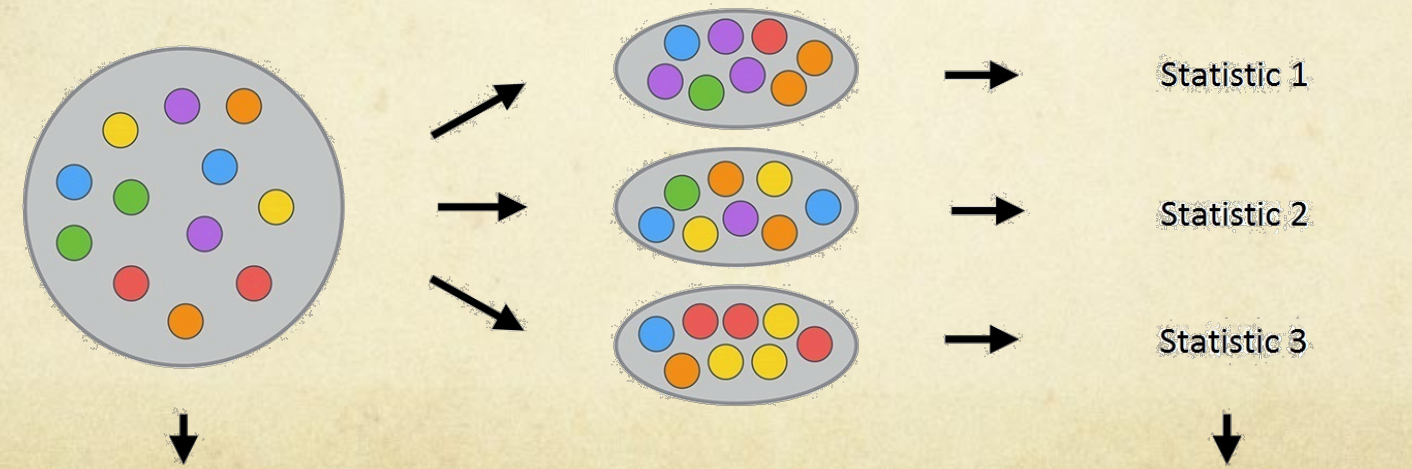
LOOC

K-fold cross-validation

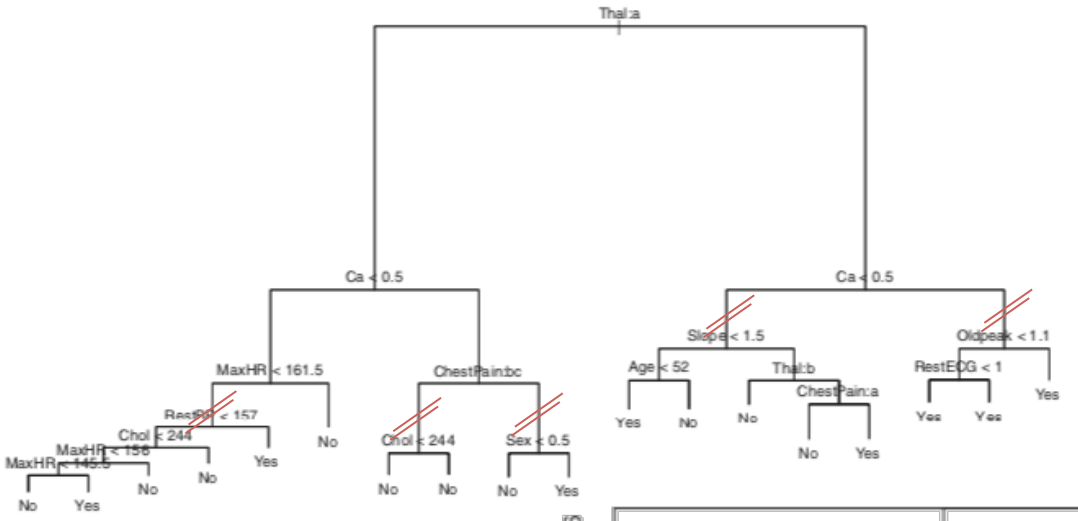
Initial Sample

Bootstrap

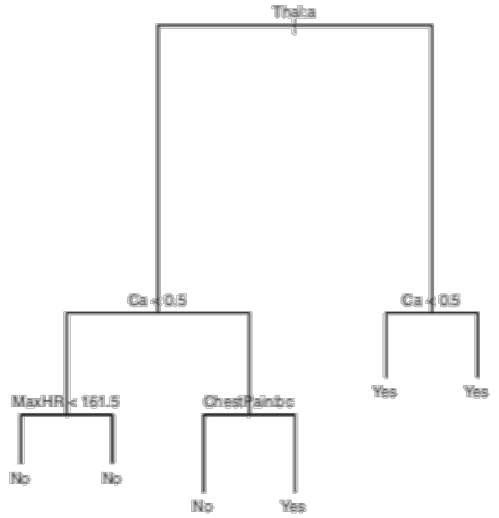
Statistics

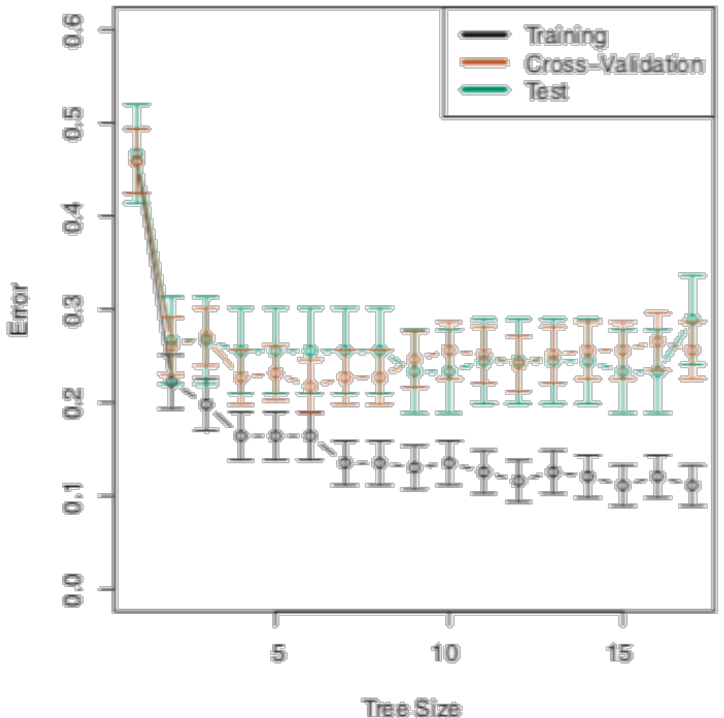


Prunning



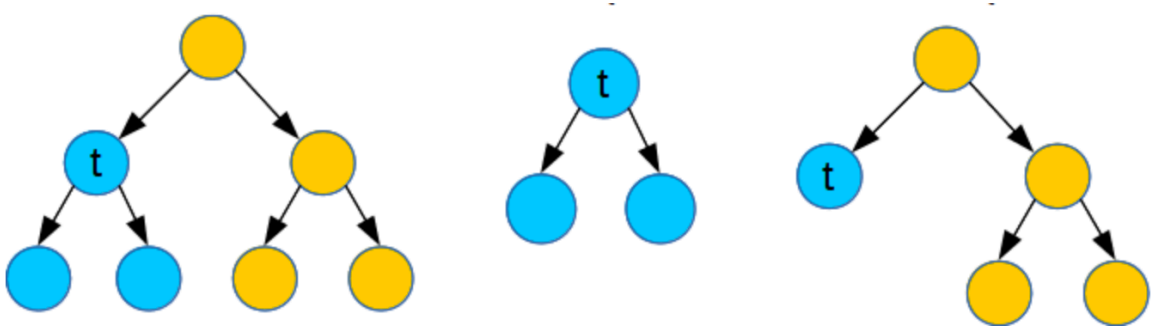
Prune





Prunning

- **Weakest link pruning:**
  - Starting with T_0, substitute a subtree with a leaf to obtain T_1, by minimizing:
$$\frac{RSS(T_1) - RSS(T_0)}{|T_0| - |T_1|}.$$
  - Iterate this pruning to obtain a sequence $T_0, T_1, T_2, \ldots, T_m$ where T_m is the null tree.
  - Select the optimal tree T_i by cross validation.



$$\text{RSS} = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Prunning

- **Cost complexity pruning:**
  - Solve the problem:

$$\text{minimize}_T \sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2 + \alpha |T|.$$

  - When $\alpha = \infty$, we select the null tree.
  - When $\alpha = 0$, we select the full tree.
  - The solution for each α is among $T_1, T_2, \ldots, T_m$ from weakest link pruning.
  - Choose the optimal α (the optimal T_i) by cross validation.

Prunning

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

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

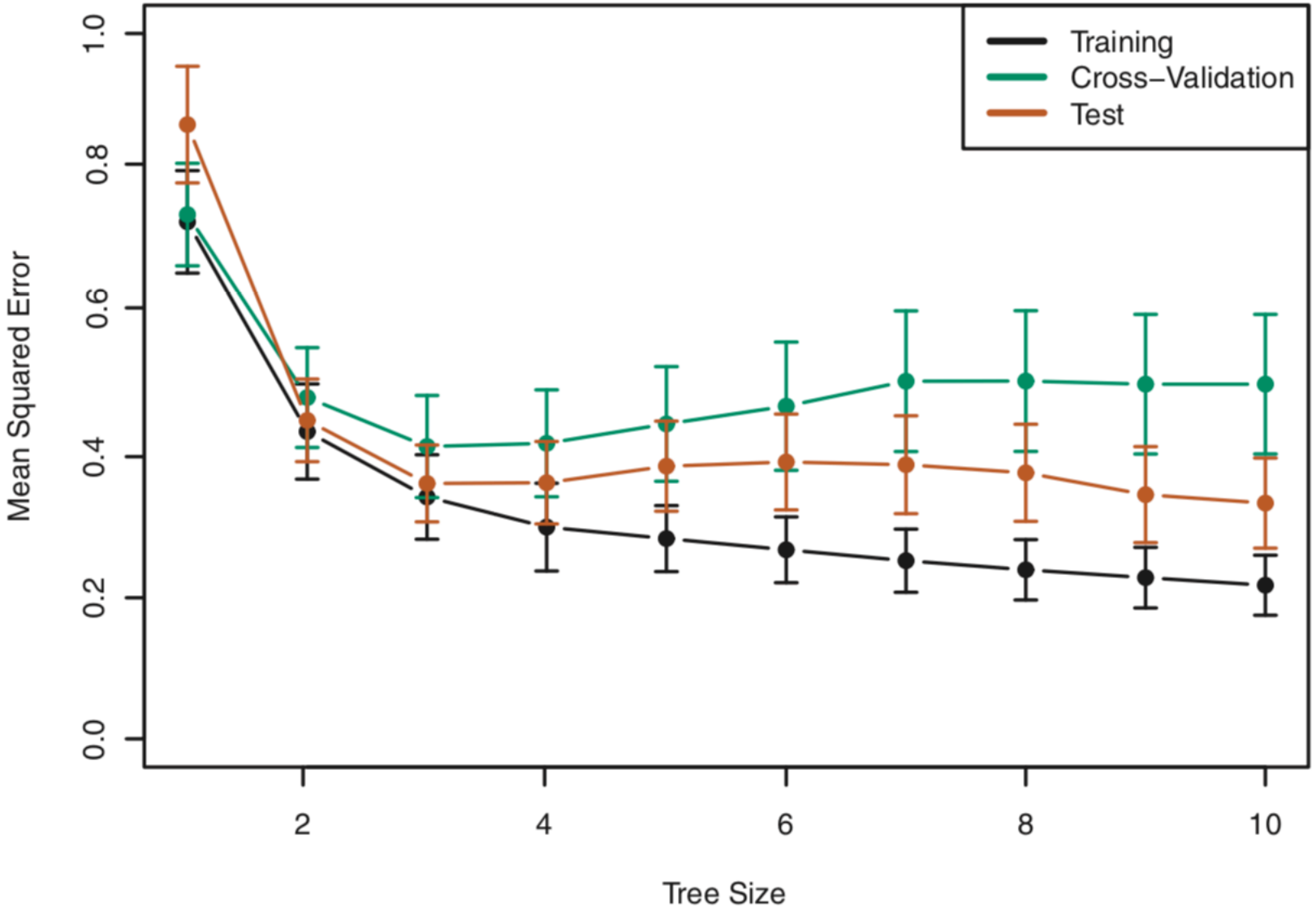
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k = 1,..., K:

   1. Repeat Steps 1 and 2 on all but the kth fold of the training data.
   2. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

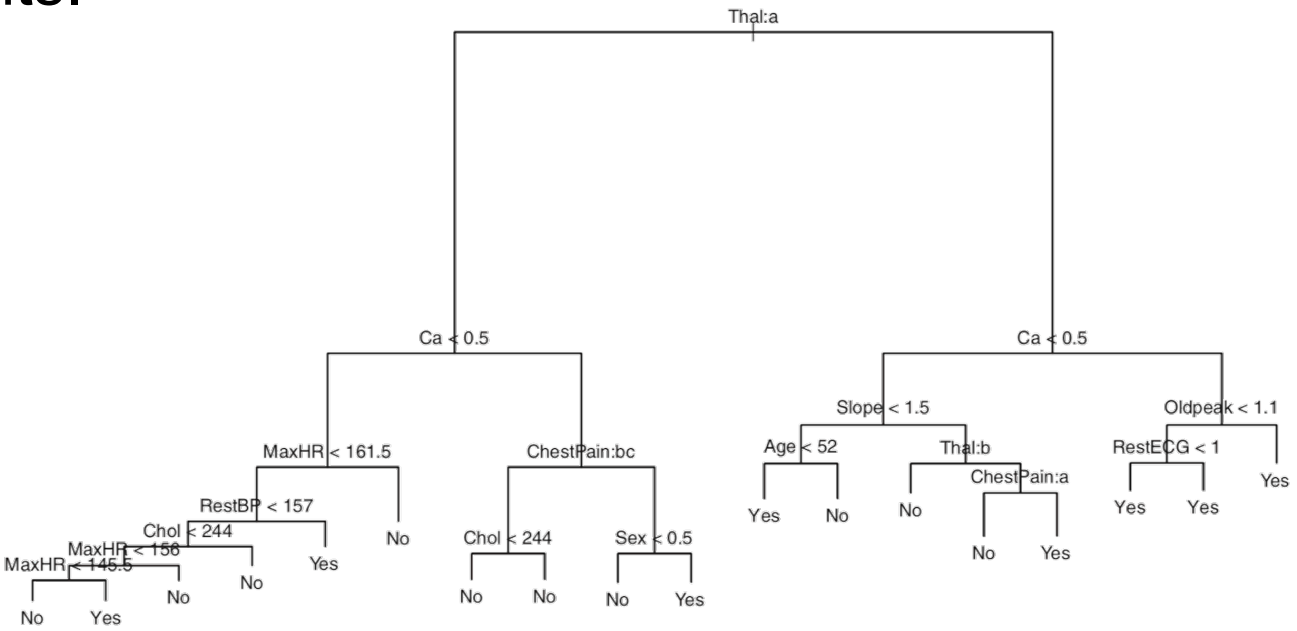
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Training, validation and testing errors

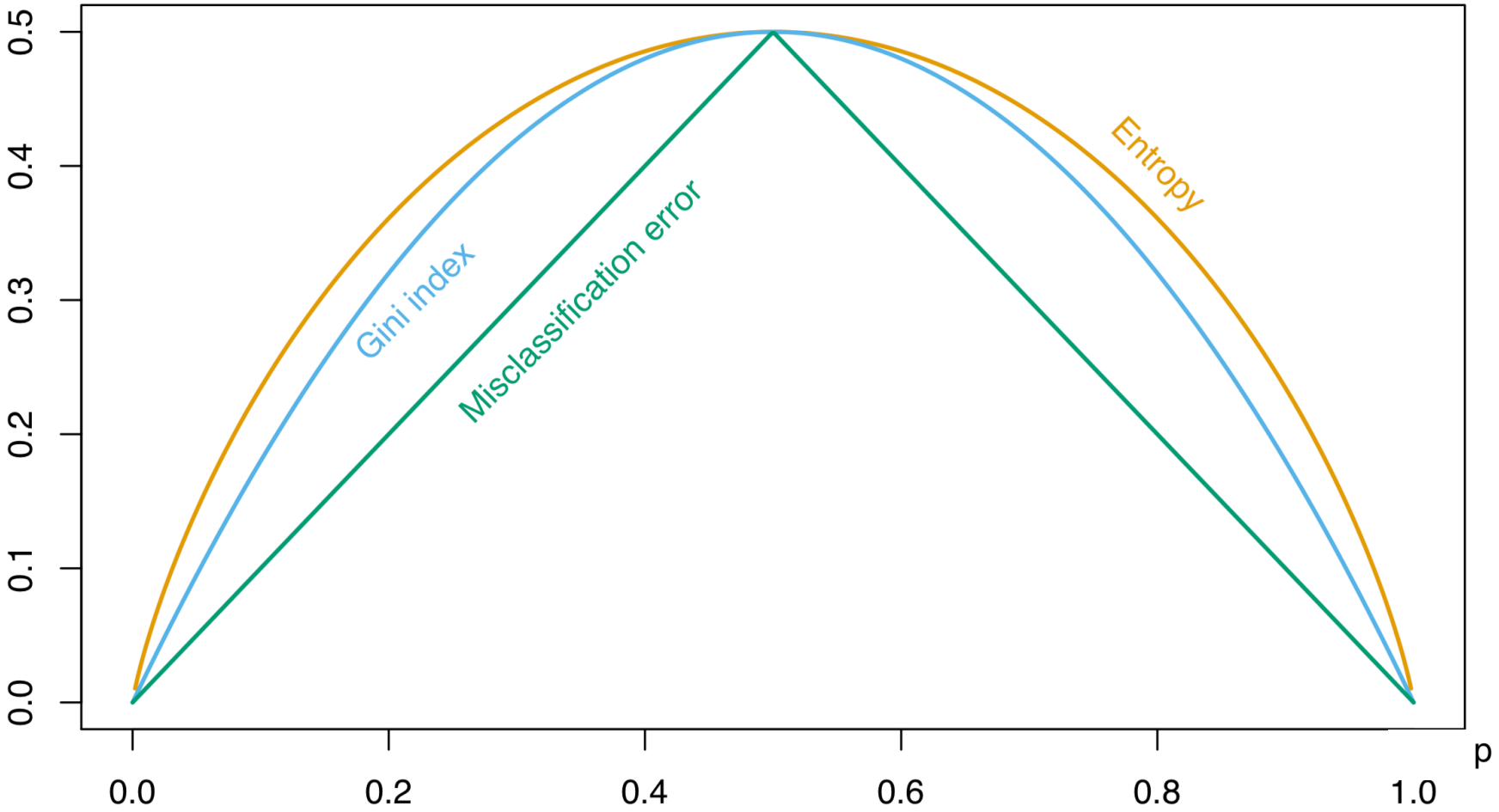


Building a classification tree

- Similar to regression, but for **qualitative** response.
- Observation belongs to **most commonly occurring class** of training observations in the region to which it belongs (i.e., *Mode* instead of *Mean*).
- RSS cannot be used as a criterion for making the binary splits.



Measures of node impurity



Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}.$

Gini index: $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$

Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$

donde $\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$

Trees - Pros & Cons

▲ Trees are easy to interpret.

▲ They closely mirror human decision-making (compared to other regression and classification approaches).

▲ Easy to visualize graphically (especially if they are small).

▲ Trees can easily handle qualitative predictors and missing data.

▼ They are often bad predictors!

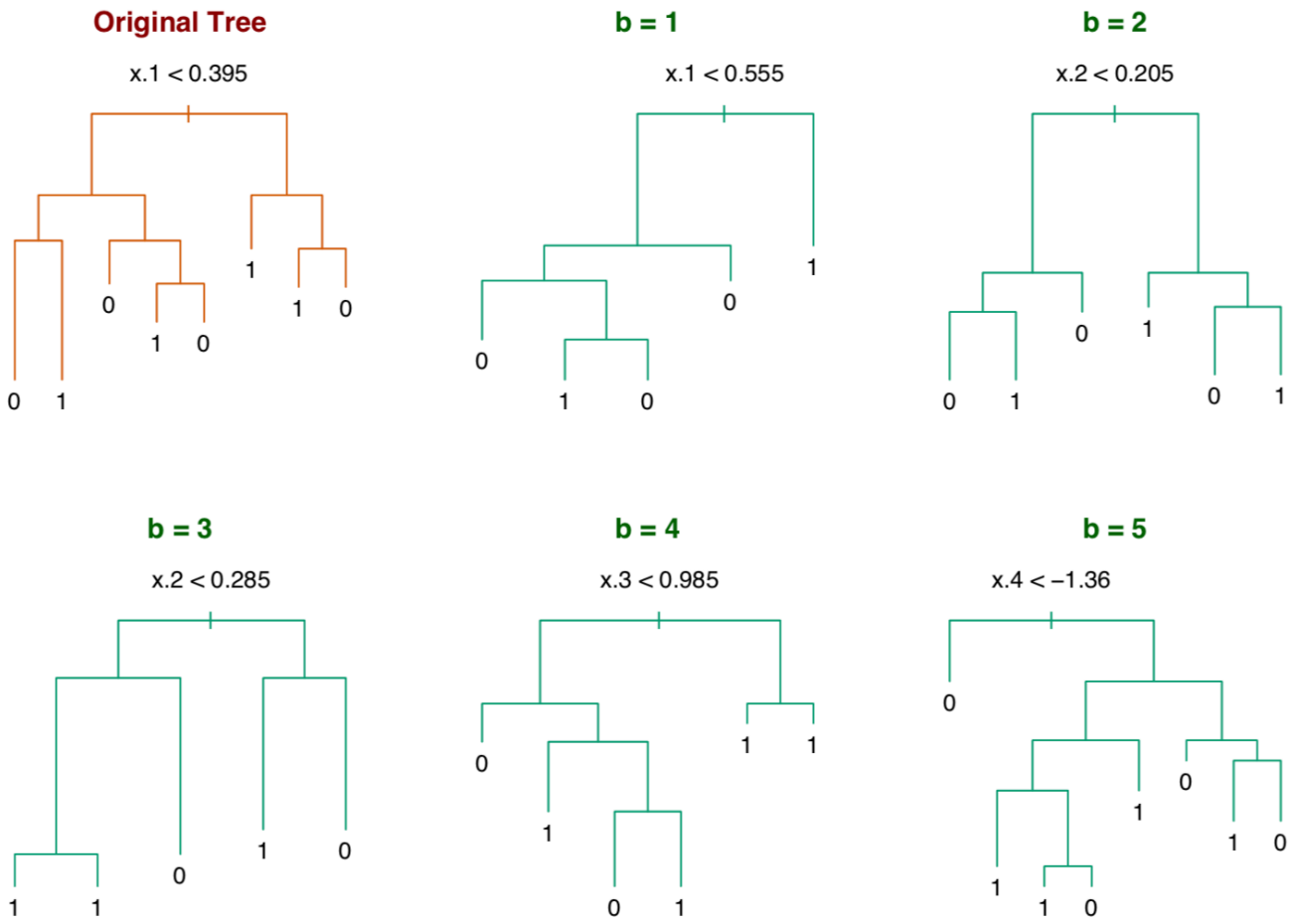
Ensemble learning

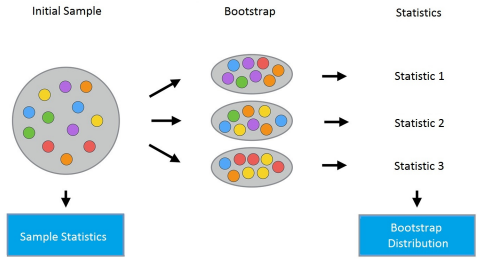
Can a set of weak learners create a single strong learner?

sklearn.ensemble module

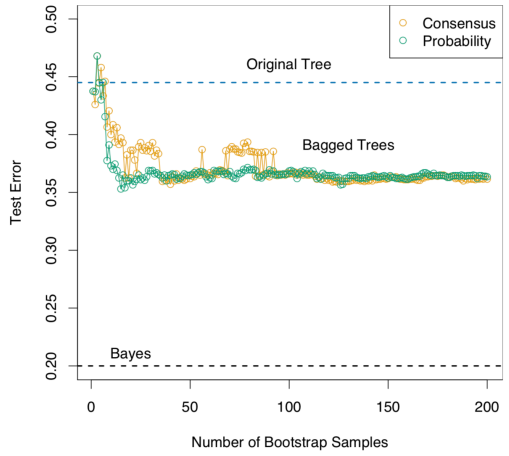
Bagging

Average B trees out of a bootstrapped sample



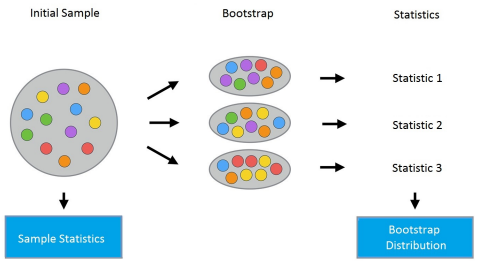


$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$



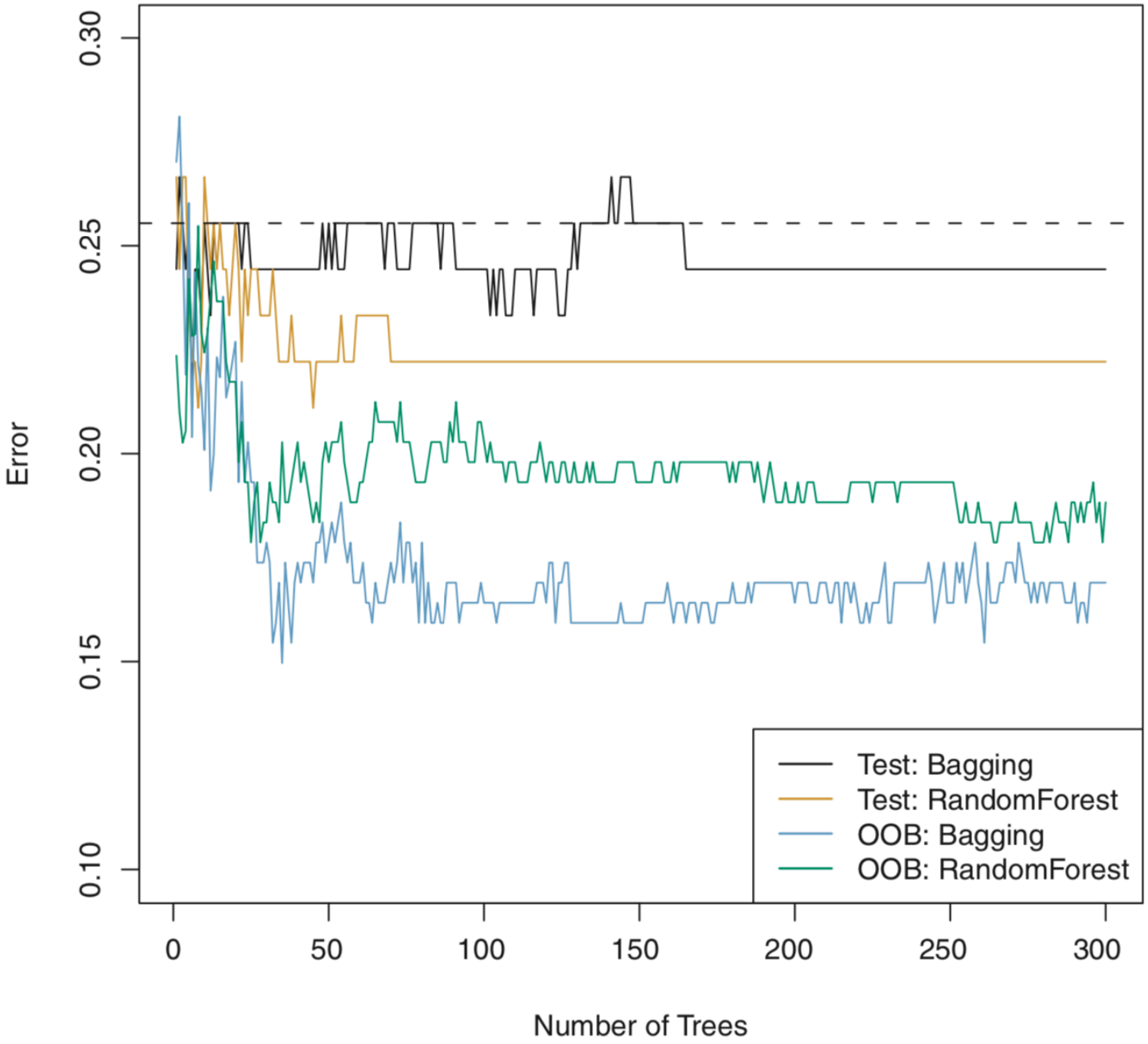
ensemble.BaggingClassifier
ensemble.BaggingRegressor

Out-of-Bag (OOB) Error Estimation



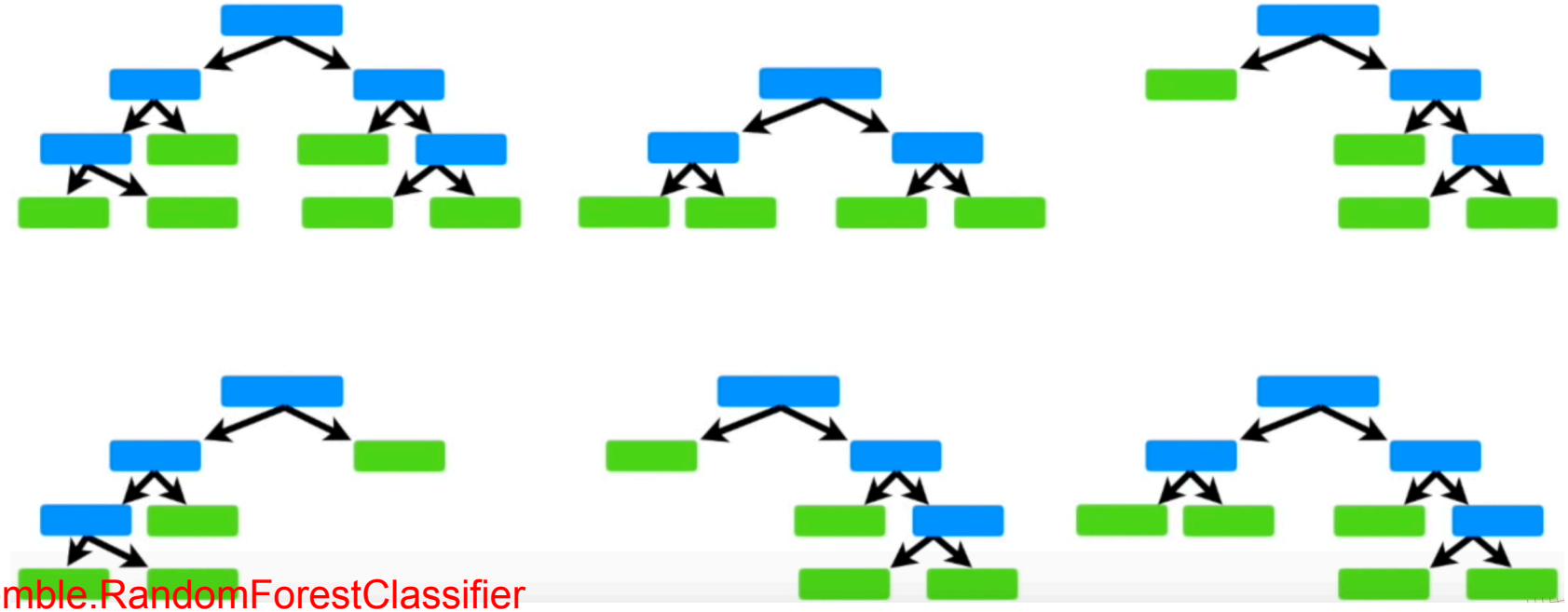
- OOB error estimation is a method **to estimate the test error** of a bagged model.
- In bagging, trees are repeatedly fit to bootstrapped subsets of the observations. On average, each bagged tree makes use of around 2/3 of the observations.
- The remaining 1/3 are referred to as the **out-of-bag (OOB) observations.**
- We can predict the response for the i^{th} observation using each of the trees in which that observation was OOB. This will yield around B/3 predictions for the ith observation, which we can average and then estimate the error.
- This estimate is essentially the **LOO cross-validation error for bagging**, if B is large.

OOB Errors for Random Forests (scikit-learn)


Error
0.10
0.15
0.20
0.25
0.30
0
50
100
150
200
250
300
Number of Trees
Test: Bagging
Test: RandomForest
OOB: Bagging
OOB: RandomForest

Random Forests

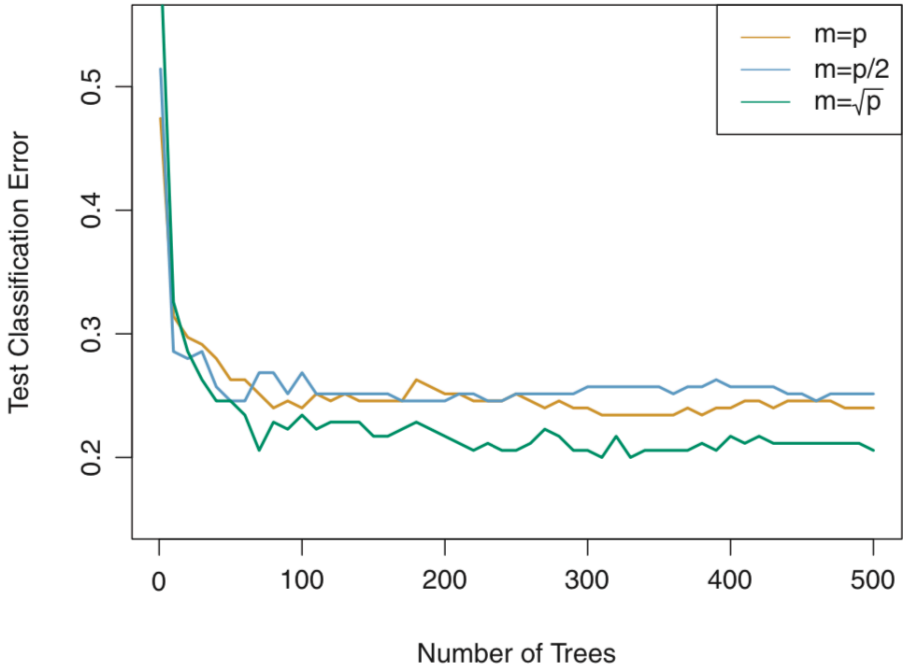
- **Random forests** provide an improvement over bagged trees by way of a small tweak that **decorrelates** the trees, reducing the variance when we average the trees.
- The central idea is to **randomly select a subgroup of predictors** for each bagged tree. ...Less is more



ensemble.RandomForestClassifier
ensemble.RandomForestRegressor

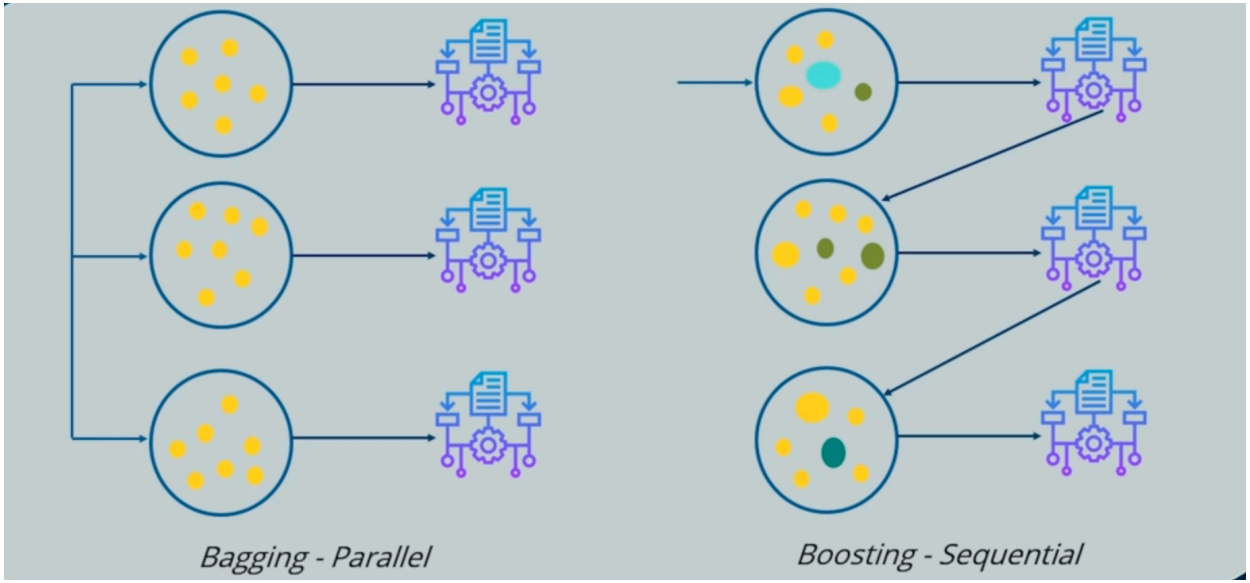
Random Forests

- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, a **random selection of m (or mtry) predictors** is chosen as split candidates from the full set of p. predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically we choose m ≈ √p. (We recover Bagging when m = p).



Boosting methods

- Boosting is a **general approach** for improving the predictions resulting from statistical learning methods.
- As in *Bagging*, Boosting also grows several trees.
- But Boosting **does not** involve bootstrap sampling.
- Rather, Boosting works growing the trees **sequentially**
  - **AdaBoost.M1**, Freund and Schapire (1997) and **Gradient Boosting Machines** (GBM) are the two most frequently used algorithms.

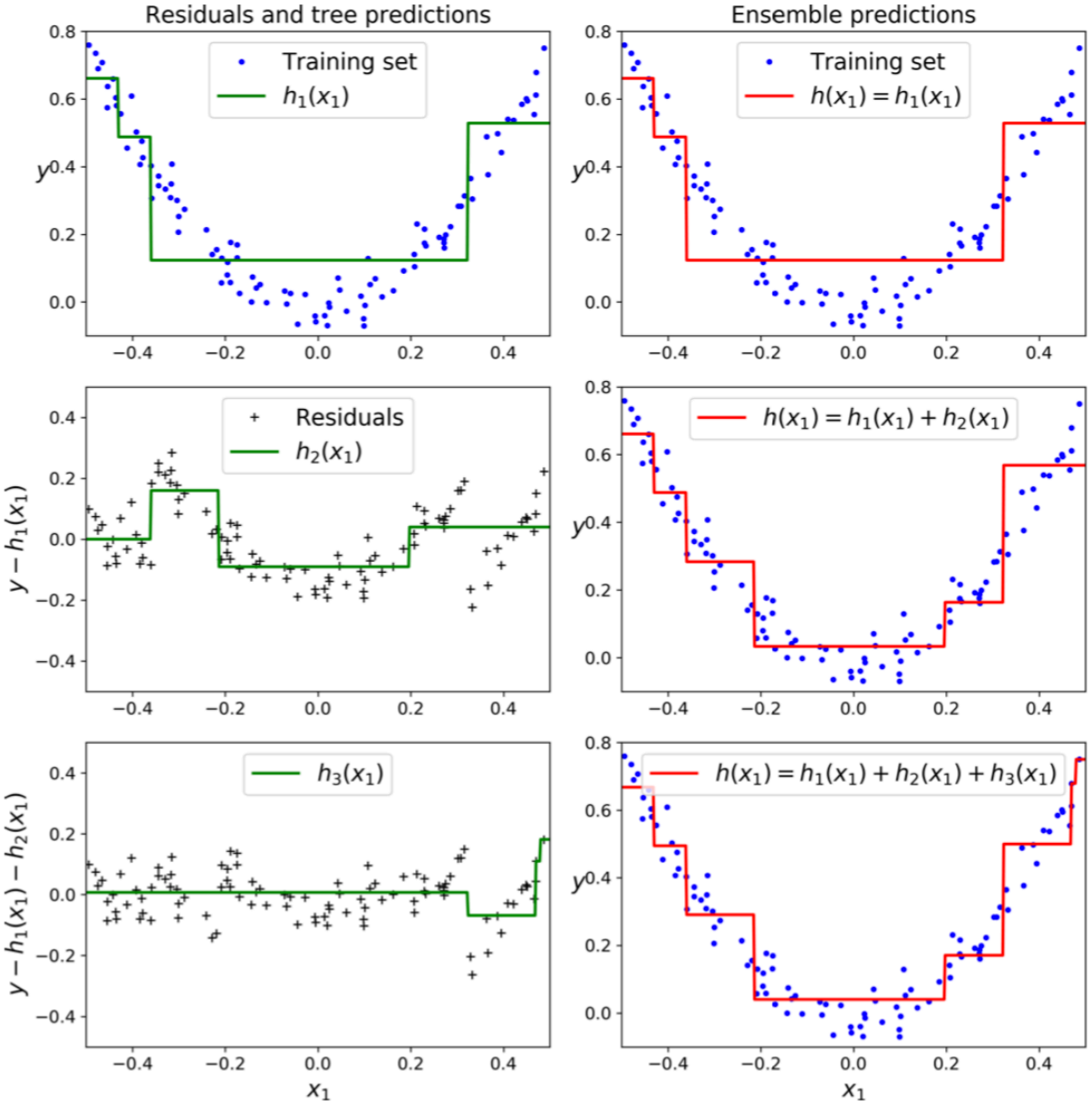


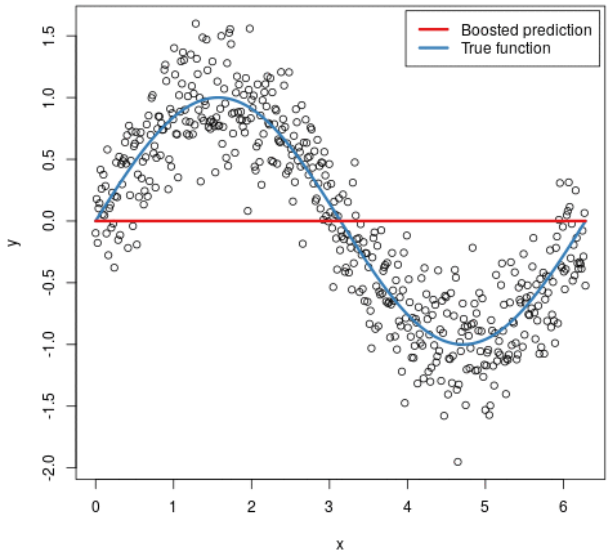
ensemble.GradientBoostingClassifier and ensemble.AdaBoostClassifier, also check "Multi-Class AdaBoost," J. Zhu et al. (2006).

Boosting methods

- Fit a decision tree **to the residuals** from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be **rather small**, with just a few terminal nodes, determined by the parameter *d* (depth) in the algorithm.
- By fitting small trees to the residuals, we slowly **improve** the model in areas **where it does not perform well**.
  - The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Boosting methods





Boosting methods

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

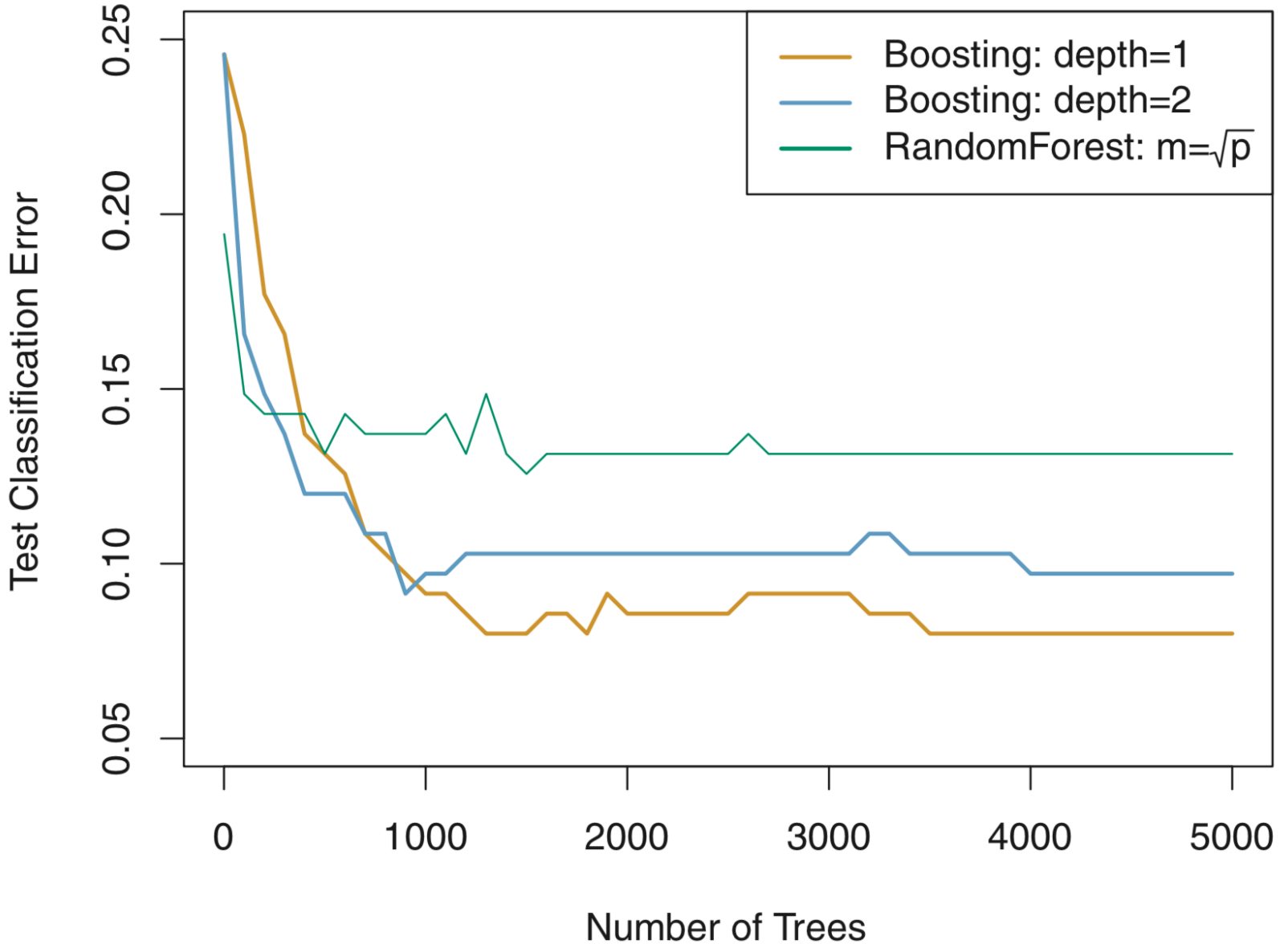
   - 2.3 Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

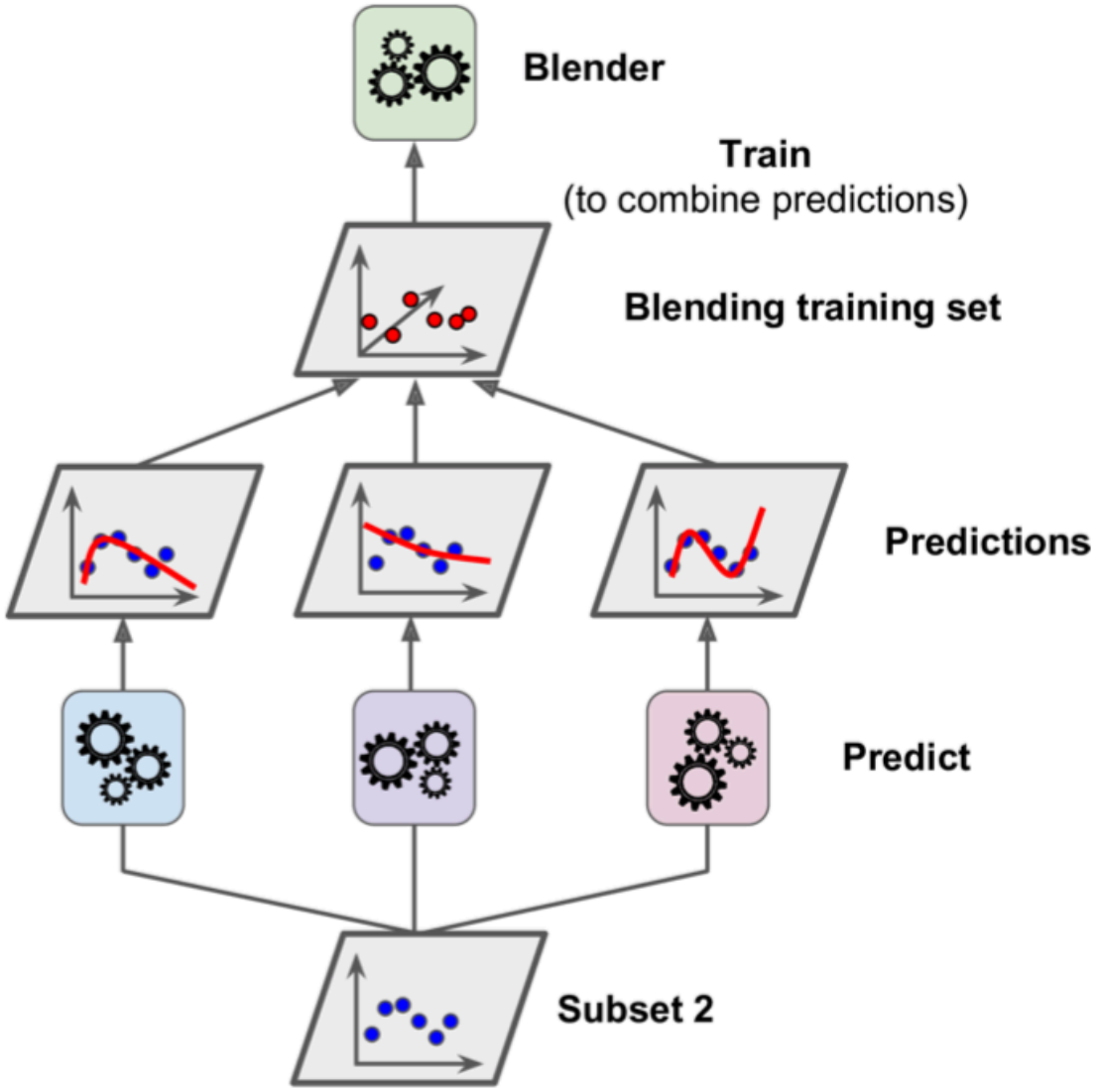
3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Boosting methods



Stacking



http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

Tree-based methods: Take-home messages

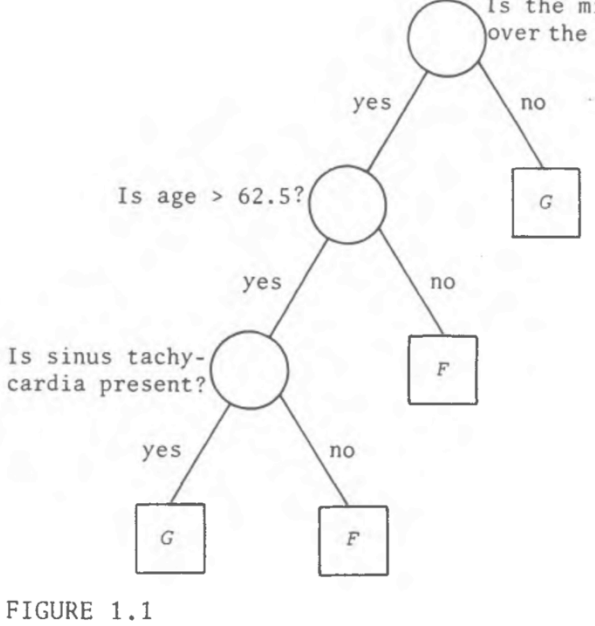


FIGURE 1.1

- There's life outside the NN planet.
- Decision trees is a simple and powerful approach for supervised learning.
- Weak learners combined into ensembles (bagging, RF, boosting and *stacking*) produce powerful results.
- Have different tools in your ML toolkit!

Luis G. Moyano - lgmoyano@gmail.com

FiEstIn