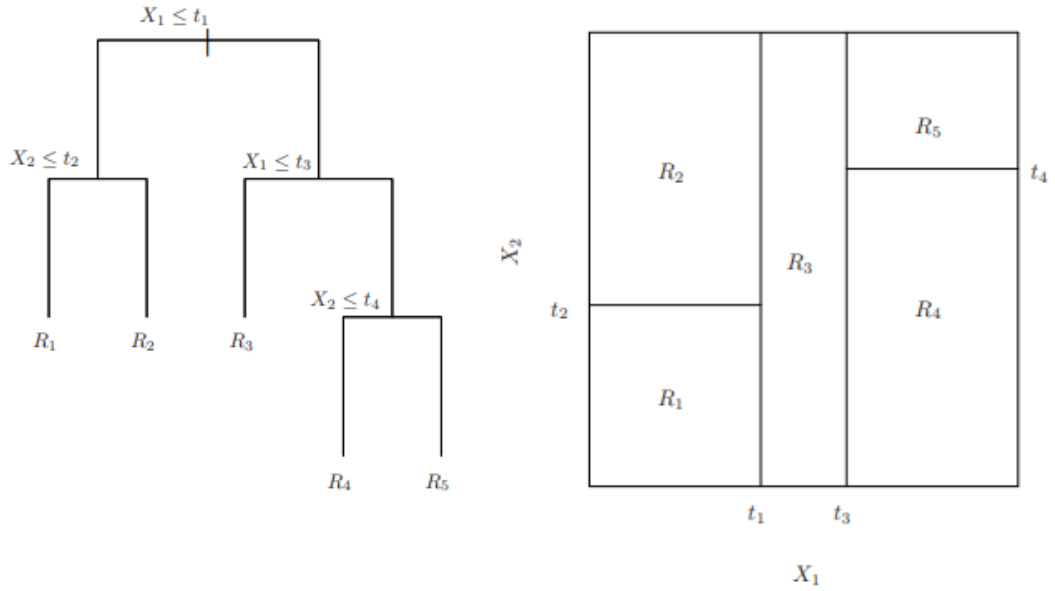
Tree and Random Forest

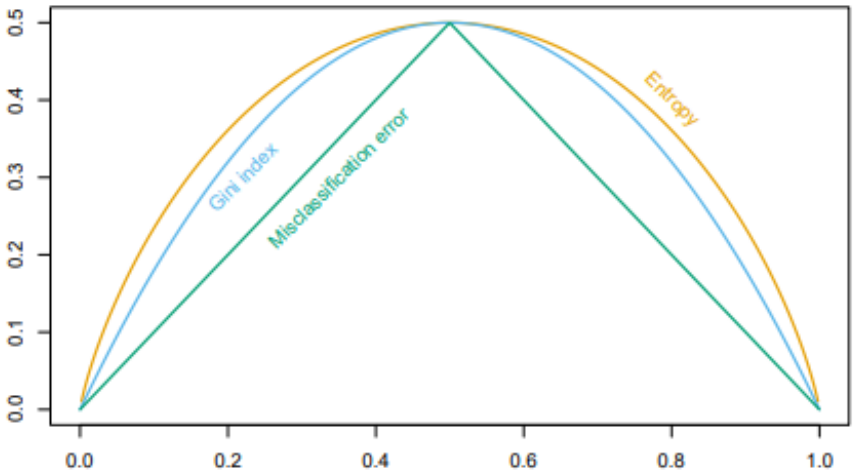
Zhuangzhuang

Tree: Greedy Splits on Feature Space

The tree model drives a sequence of decisions of splits on the feature space. Each split is performed by aligning with the axis. As an outcome, the high-dimensional “rectangles” in the feature space split by the tree have the least prediction errors.



Tree: Greedy Splits on Feature Space



In other words,

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

is minimized in the regression tree, which is computationally infeasible for high-dimensional cases. Nevertheless, Iterative greedy splits offer a way out:

The feature and split location are chosen from those available as that pair which

- *minimizes the resulting within-node variance (regression)*
- *or maximizes the within-node purity (classification) of the offspring nodes.*

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

The Depth: The Complexity of Tree

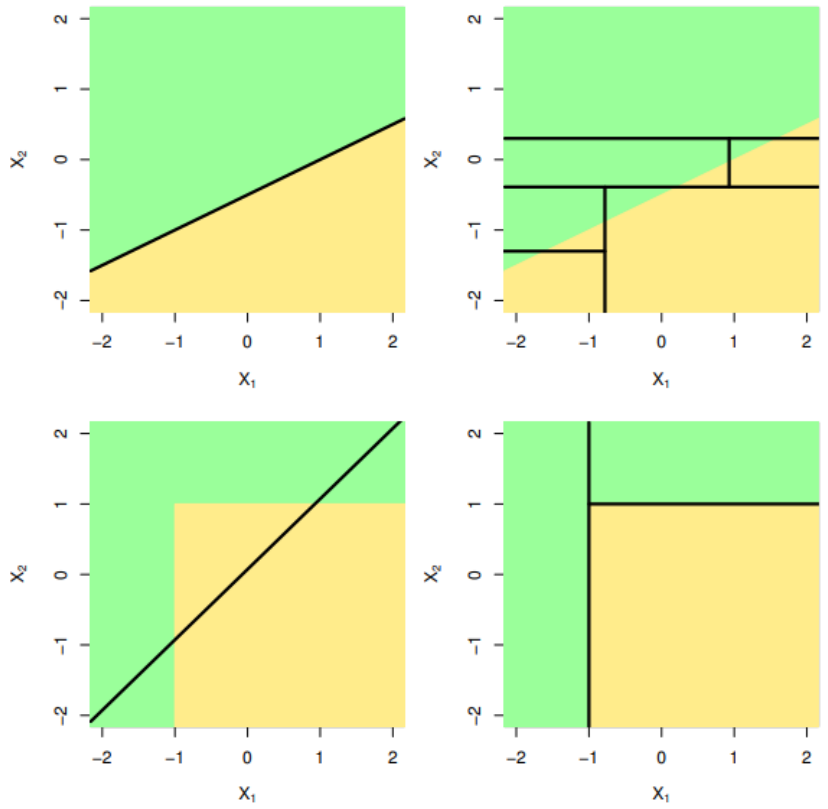
The maximum complexity is reached as the training set is interpolated. In other words, all terminal nodes predict the training sample perfectly. Other the other hand, the complexity can be controlled by

- Terminal node size
- The tree depth
- The thresholds regarding terminal nodes' RSS and purity
- Pruning

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

Tree's Downsides

- Inaccuracy and inefficiency exist in the model complexity control.
- Inefficiency also occurs in those cases where linear models fit well:

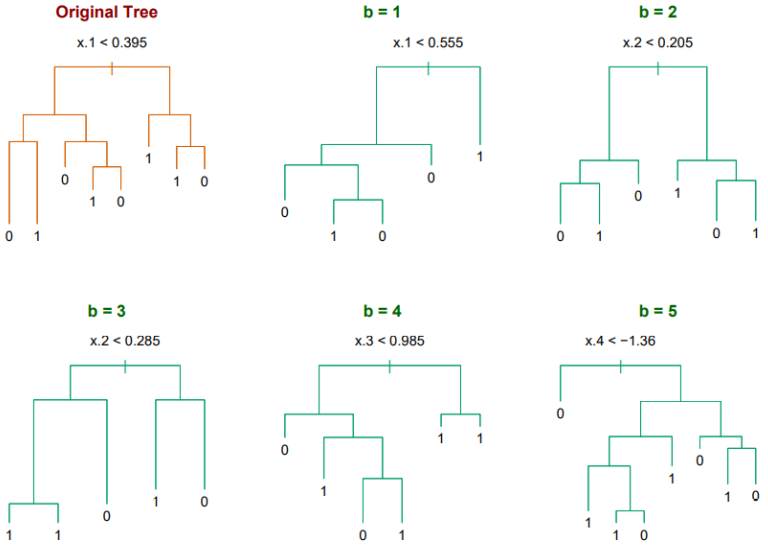


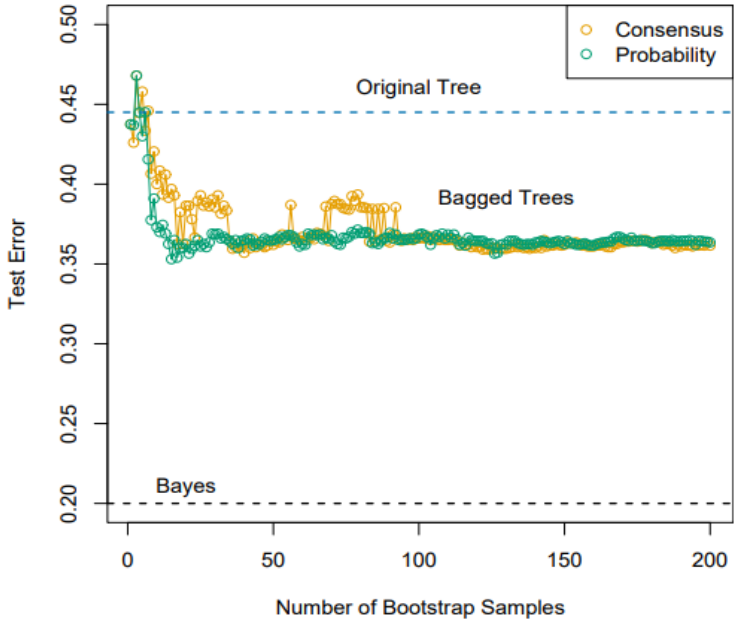
Tree Ensembles: Bagging and Boosting

Bagging

The tree approximates the preceding linear boundary by increasing its complexity (depth) but will suffer a high variance. One way out of this is to bootstrap the training set, train a tree using each bootstrapped set, and average (i.e., aggregate) the prediction outputs --- that is, bagging.

- Bootstrapping is approximately a posterior average (with non-informative prior; see ESL, 8.4).
- For classification, the majority vote is applied to aggregate predictions.





Tree Ensembles: Bagging and Boosting

Boosting adds weak learners (low-depth trees) sequentially: the next tree is learned using the residual of predicted value from the previous trees.

More on Bagging

Bagging works effectively for **non-linear/non-stable** or adaptive prediction methods by reducing the variance

- Tree-based and neural network models

But it does not have magic in those are **linear/stable**

- Linear least-squares regression, ridge regression, non-adaptive kernel, and *nearest neighbor methods (?) (Freidman and Hall, 2000)*

More on Bagging (Cont'd)

Friedman and Hall (2000)

$$\theta^*(\mathbf{x}) = \arg \max_{\theta(\mathbf{x}) \in \Theta} E_{y,\mathbf{x}} l(y, \theta(\mathbf{x}))$$

$$\hat{\theta}_b(\mathbf{x}) = \arg \max_{\theta(\mathbf{x}) \in \Theta} \frac{1}{n} \sum_{i \in S_b} l(y_i, \theta(\mathbf{x}_i)),\ b \in \{1, \cdots, B\}$$

$$\hat{\theta}_B(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_b(\mathbf{x})$$

$$\sum_{i=1}^{n} g(U_i, \theta) = 0$$

$$\sum_{i=1}^{n} \left\{ g(U_i, \theta_0) + \sum_r g_r(U_i, \theta_0) \left(\hat{\theta} - \theta_0\right)^{(r)} + \frac{1}{2} \sum_{r_1} \sum_{r_2} g_{r_1 r_2}(U_i, \theta_0) \left(\hat{\theta} - \theta_0\right)^{(r_1)} \left(\hat{\theta} - \theta_0\right)^{(r_2)} + \ldots \right\} = 0$$

Solving the very last equation, one can show that $\hat{\theta}$ is approximated by **a linear (first-order) part along with quadratic, cubic, and high-order components**.

More on Bagging (Cont'd)

Freidman and Hall's (2000) interpretation of bagging suggests that

- *bagging cannot effectively replace regularization (to trade a bit of bias for a significant reduction in variance) if the linear component is substantial.*
- *for highly non-linear models, perhaps, it does not need to use both regularization and bagging.*

More on Bagging (Cont'd): Out-of-Bag (OOB) Errors

- On average, each bagged tree utilizes around two-thirds of the observations. (Open-book HW; ISLR, Chap. 5 Ex. 2; ESL, Sec. 11 Chap. 7)
- As the number of bootstraps increases, the OOB approximates the N-fold CV. (HW)
- **OOB error is unbiased in general (no formal derivation, but tested in many)**
  - **However, OOB overestimates when p >> n. (Mitchell, 2011)**
- It is preferred regarding its computation w.r.t C.V..

Random Forest (RF): A Special Kind of Bagging

RF differs from the bagged trees in

- *Requiring only randomly choosing "mtry" of features to grow each bootstrapped tree.*
  - *This step de-correlates the bootstrapped trees to reduce variance (See ESL Sec. 2 Chap. 15).*

- *Pruning becomes less necessary as the random selection of features regularizes the model complexity.*

Note that some use subsamples instead of bootstrapped samples and obtain similar performance.

Main Course Today: Why does RF Perform Better? – An Answer by Degrees of Freedom (DF)

RF simply "is better" than bagging or other similar randomized approaches.

-- by recent empirical evidence (Mentch and Zhou, 2020)

RF Performs Better? – An Answer by DF (Cont'd)

Breiman (2001) offered a motivation rather than an explanation for using RF:

- Two inherent random components---bootstrapped samples and randomly selected pre-fixed number of features (mtry)---serve as the regularization like the lasso in a linear model, allowing RF to benefit from the bias-variance trade-off.
- RF may "*act to reduce the bias in some ways*." Later studies showed such reductions are limited to variance. (See Hastie 2009)

RF Performs Better? – An Answer by DF (Cont'd)

Wyner (2017), taking a more definitive stance, claimed that

- RF's superior performance can be credited to its *"self-averaging interpolation"* --- i.e., fitting the training data perfectly while retaining some degree of smoothness due to the averaging.
- This explanation works when the data contains high signals with low noises. Put differently, each Tree in RF tends to overfit both noises and signals, but eventually, the overfitting of signals prevails by averaging. (Wyner calls this effect "localizing.")

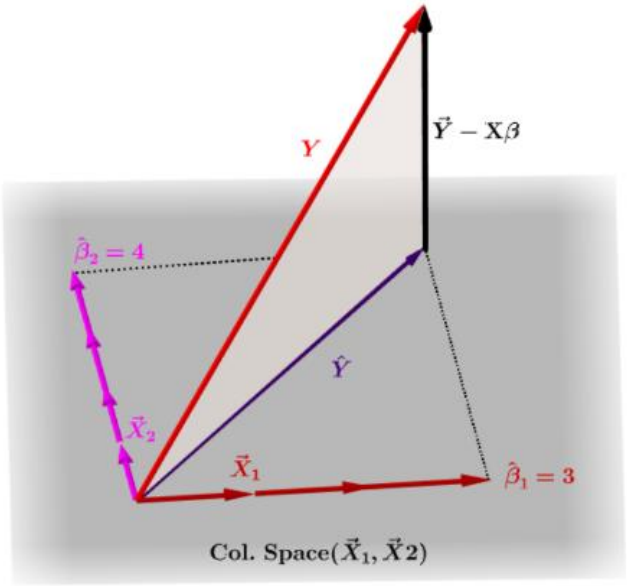
RF Performs Better? – An Answer by DF (Cont'd)

Following Breiman's (2001) intuitions, Mentch and Zhou (2020) investigated the complexity of RF and tree-based bagging regarding the effective DF, implying that the "mtry" lets RF trade bias for variance like regularizing.

A Quick Recap of Effective DF

For a linear prediction method (e.g., least-square regression, non-adaptive kernel, and regularized regression), one can obtain

$$\hat{y} = Hy$$



A Quick Recap of Effective DF (Cont'd)

What if one applies the expression for linear models to those non-linear? Mallows (1973) borrowed the below expression from the linear regression

$$\mathrm{EPE} = \mathbb{E}[\mathrm{RSS}] + 2\sigma^2 p$$

Efron (1986) brought y_i^*, a new independent copy of y_i at $\boldsymbol{x}_i$, to the above expression and have

$$\mathbb{E}\left[\sum_{i=1}^{n}(y_i^* - \hat{y}_i^{(FIT_k)})^2\right] = \mathbb{E}\left[\sum_{i=1}^{n}(y_i - \hat{y}_i^{(FIT_k)})^2\right] + 2\sigma^2 \cdot \mathrm{DF}(\boldsymbol{\beta}, \boldsymbol{X}, \mathrm{FIT}_k)$$

With some simple derivations, Efron (1986) rewrote it as

$$\mathbb{E}\left[\sum_{i=1}^{n}(y_i^* - \hat{y}_i)^2\right] - \mathbb{E}\left[\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right] = 2\sum_{i=1}^{n}\mathrm{Cov}(y_i, \hat{y}_i)$$

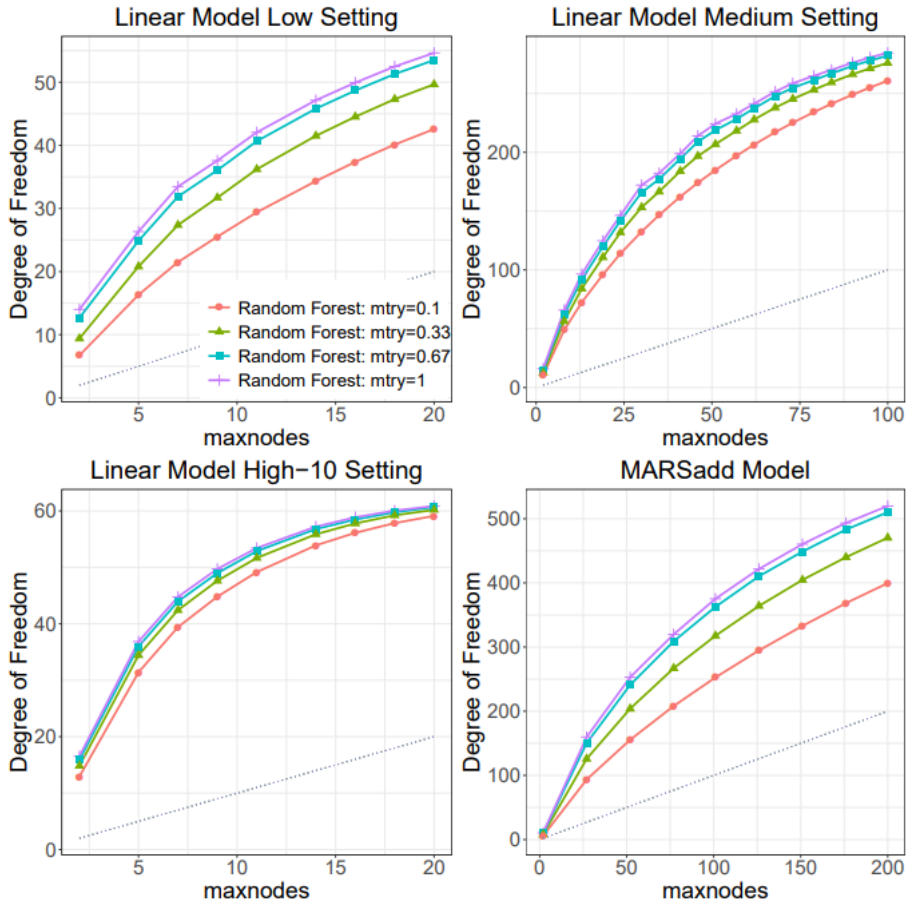
RF Performs Better? – An Answer by DF (Cont'd)

$$Y = X\beta + \epsilon = X_1\beta_1 + \cdots + X_p\beta_p + \epsilon$$

$$Y = 0.1e^{4X_1} + \frac{4}{1 + e^{-20(X_2 - 0.5)}} + 3X_3 + 2X_4 + X_5 + \epsilon$$

- **Low**: $n = 100,\ p = 10,\ s = 5$
- **Medium**: $n = 500,\ p = 100,\ s = 5$
- **High-10**: $n = 100,\ p = 1000,\ s = 10$

RF Performs Better? – An Answer by DF (Cont'd)



$$\mathrm{SNR} = \frac{\mathrm{Var}(f(x))}{\mathrm{Var}(\epsilon)}$$