Random Forest

Mathieu Ribatet—Full Professor of Statistics

1. Introduction

Some references

[1] Gérard Biau and Erwan Scornet. A random forest guided tour. *TEST*, 25(2):197–227, 2016.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[4] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

Quick overview

- Random forests are fairly recent learning strategy (00's)
- It is based on classification and regression trees or CART for short.
- It is a modification of bagging to mitigate dependence between trees.

ℹ Bagging and random Forest heavily rely on bootstrap.

A simple statement

Proposition 1. *Let $T_1, \ldots, T_B$ be independent copies of T with Var$(T) = \sigma^2$. We have*

$$Var\left(\bar{T}_B\right) = \frac{\sigma^2}{B}, \qquad \bar{T}_B = \frac{1}{B}\sum_{b=1}^{B} T_b.$$

Proposition 2. *Let $T_1, \ldots, T_B$ be dependent copies of T with Var$(T) = \sigma^2$ and pairwise correlation $\rho > 0$. We have*

$$Var\left(\bar{T}_B\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2, \qquad \bar{T} = \frac{1}{B}\sum_{b=1}^{B} T_b.$$

ℹ Since

$$\text{Var}\left(\bar{T}_B\right) \longrightarrow \rho\sigma^2, \qquad B \to \infty,$$

the pairwise correlation ρ mainly controls the variance of $\bar{T}_B$ as long as B is large enough. Random forests aims at reducing ρ without increasing (too much) σ^2.

2. CART

What is a binary tree?

Definition 1. A tree is a collection of connected nodes. It is often used to display a hierarchical structure in a graphical way.
Nodes without any children are called leaves or terminal nodes.

Definition 2. A binary tree is a tree whose nodes have at most two children.

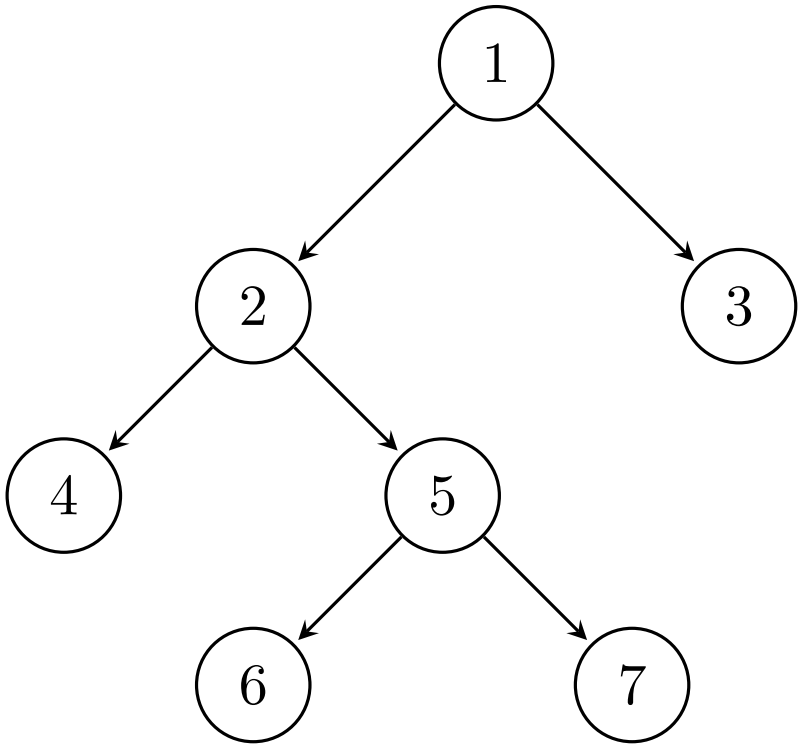

Figure 1: *A binary tree.*

Classification And Regression Trees (CART)

- CART are binary trees and are widely used in statistics
- Can be used both for regression and classification problems.
- Each terminal node is an estimator
- The estimator has the following form

$$\hat{f}(\mathbf{X}) = \sum_{j=1}^{n_{\text{terminal}}} c_j 1_{\{\mathbf{X} \in R_j\}}, \qquad \mathbf{X} \in \mathcal{X},$$

  where n_{terminal} is the number of terminal nodes, R_j is a subset of $\mathcal{X}$ and c_j an estimator for region R_j.
- CART are built from recursive binary splitting

Region R_j

- Within a CART, any $\mathbf{X}$ has to lie in a single terminal node, i.e.,

$$\mathbf{X} \in R_{j(\mathbf{X})}, \qquad \text{for some unique } j(\mathbf{X}) \in \{1, \ldots, n_{\text{terminal}}\}.$$

- Hence we get a partition of $\mathcal{X}$, i.e.,

$$\cup_{j=1}^{n_{\text{terminal}}} R_j = \mathcal{X}, \qquad R_{j_1} \cap R_{j_2} = \emptyset, \quad j_1 \neq j_2.$$

Due to binary splits, not all partitions of $\mathcal{X}$ are valid!

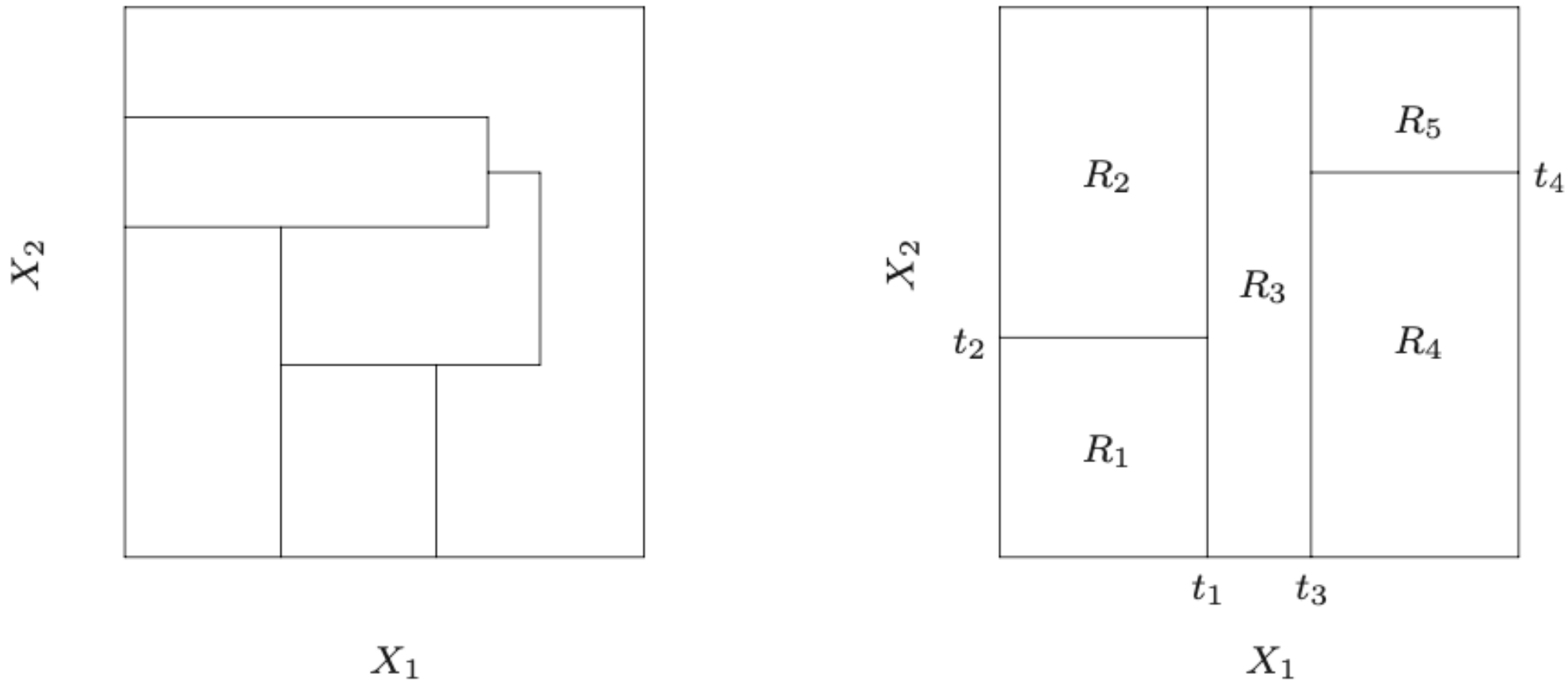

Figure 2: *Two partitions of $\mathcal{X}$. Only one is admissible! Taken from Elements of Statistical Learning (Second edition).*

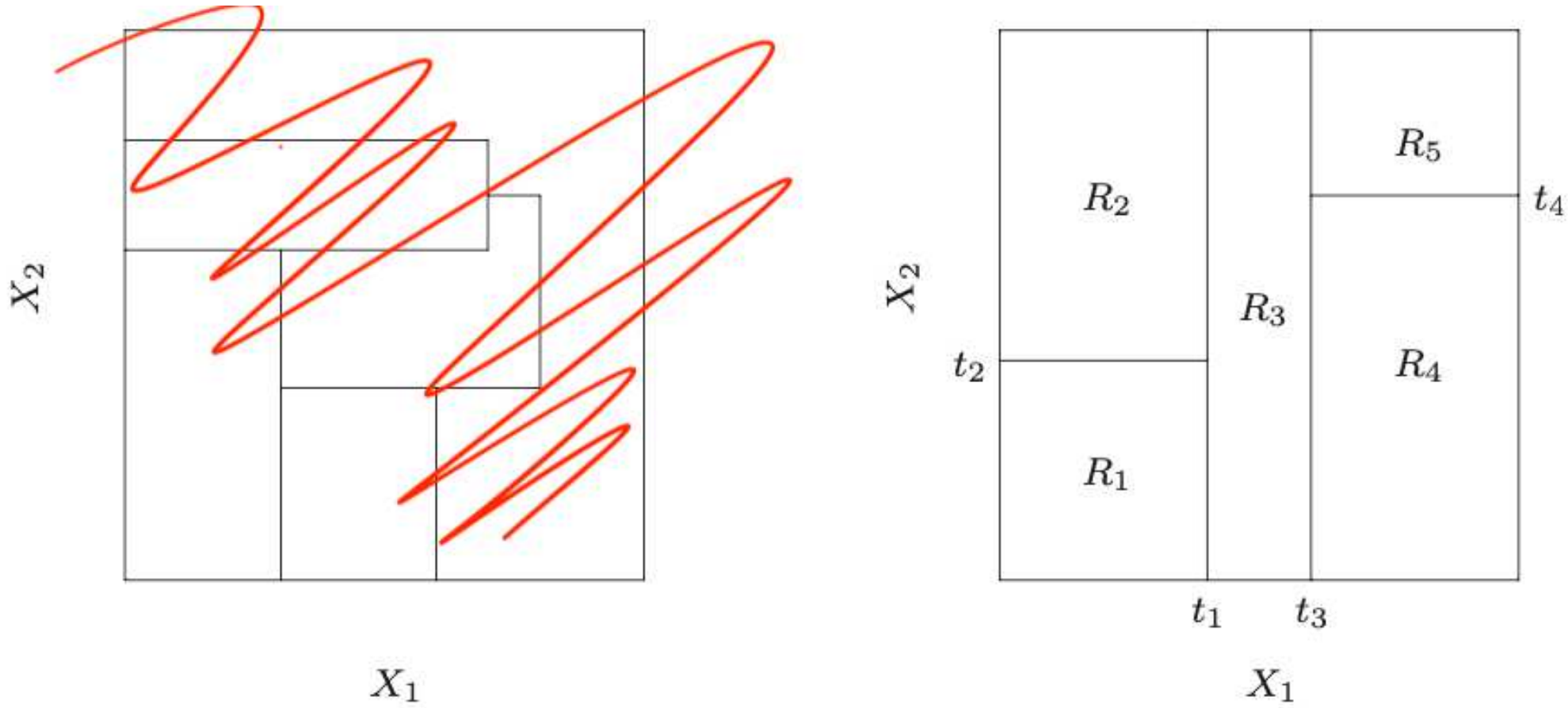

Figure 2: *Two partitions of $\mathcal{X}$. Only one is admissible! Taken from Elements of Statistical Learning (Second edition).*

How to split a node into two children?

- Consider the case where $\mathbf{X} = (X_1, \ldots, X_p) \in \mathcal{X}$
- The X_j's can be a mix of both numerical and categorical variables.
- A node N_{parent} will have two children $N_{\text{child 1}}$ and $N_{\text{child 2}}$ such that

$$N_{\text{child 1}} \in S_{\text{parent}}, \qquad N_{\text{child 2}} \notin S_{\text{parent}},$$

  with

$$S_{\text{parent}} = \mathcal{X}^{p-1} \times C_j,$$

  where C_j is a subset of possible outcomes of feature X_j.
- At each split the feature X_j will be selected from a relevant criterion.

To ease explanations, we will assume that all covariates X_j and the outcome Y are numerical. We will go back to categorical variables later.

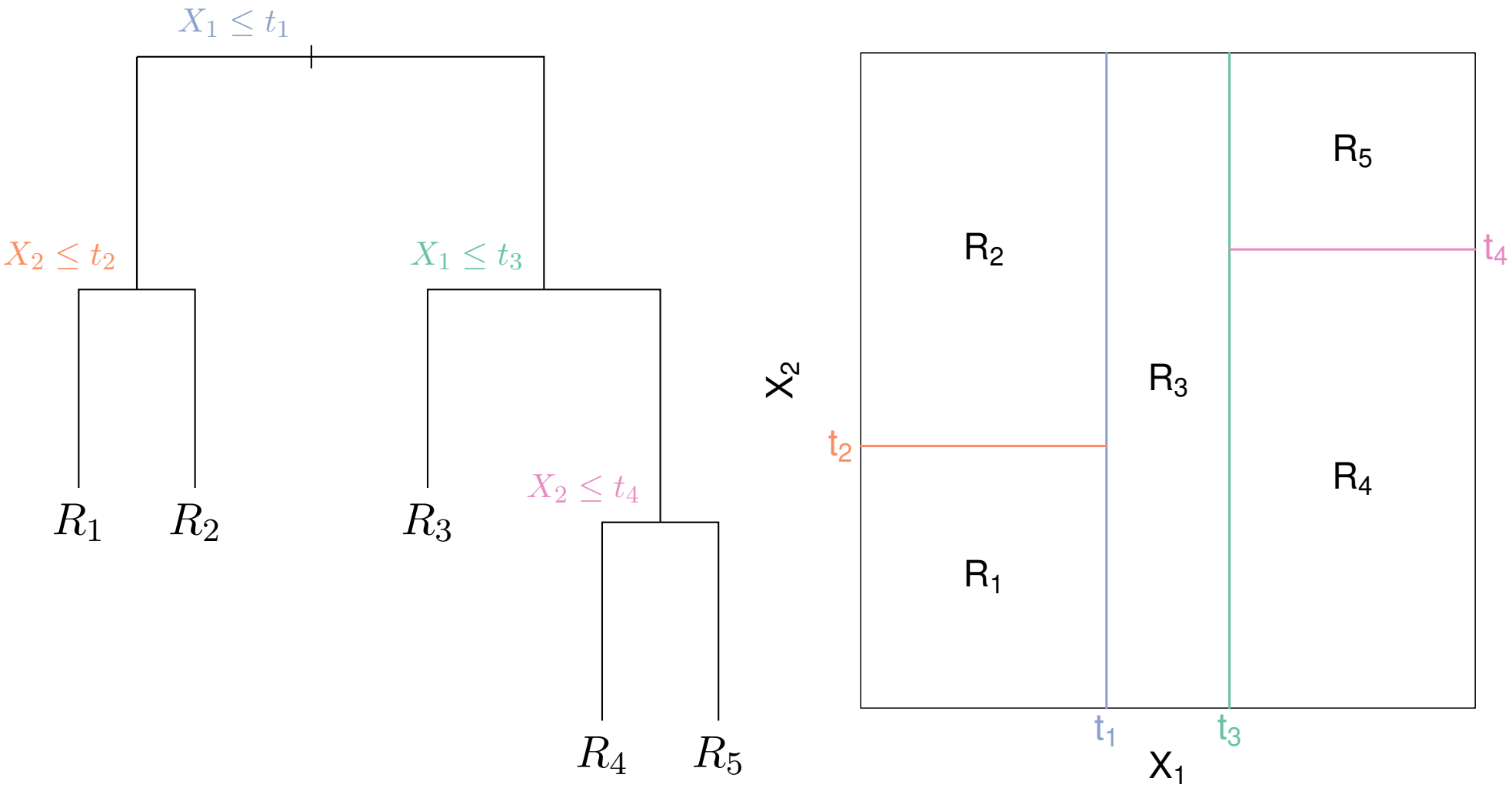

Figure 3: *A CART with $\mathcal{X} \subset \mathbb{R}^2$ (left) and the corresponding partition of $\mathcal{X}$.*

Splitting a node

Given a current node, we want to find splitting regions

$$R_1(j,s) = \{\mathbf{X} \colon X_j \leq s\}, \qquad R_2(j,s) = \{\mathbf{X} \colon X_j > s\},$$

using the following optimization problem

$$\underset{j,s}{\operatorname{argmin}} \left\{ \min_{c_1} \sum_{i \colon \mathbf{X}_i \in R_1(j,s)} (Y_i - c_1)^2 + \min_{c_2} \sum_{i \colon \mathbf{X}_i \in R_2(j,s)} (Y_i - c_2)^2 \right\}$$

For any j, s, the optimal c_1 and c_2 are always

$$\hat{c}_1 = \frac{1}{|R_1(j,s)|} \sum_{i \colon \mathbf{X}_i \in R_1(j,s)} Y_i, \qquad \hat{c}_2 = \frac{1}{|R_2(j,s)|} \sum_{i \colon \mathbf{X}_i \in R_2(j,s)} Y_i$$

$$\underset{j,s}{\operatorname{argmin}} \left\{ \sum_{i\colon \mathbf{X}_i \in R_1(j,s)} (Y_i - \hat{c}_1)^2 + \sum_{i\colon \mathbf{X}_i \in R_2(j,s)} (Y_i - \hat{c}_2)^2 \right\}$$

- Finding the optimal cutoff value s and feature j are relatively easy.
- For feature X_j, possible cutoff values s are the observed outcome of X_j
- Hence the above optimization problem is solved using a brute–force search.

All possible cutoff values s can be computed once for all at the beginning of the learning stage.

Growing and pruning a tree

- The main strategy is to:
  1. Repetitively split nodes until some minimum node size is reached;
  2. "simplify" the obtained tree
- The last stage is called pruning a tree
- It consists in "collapsing the internal nodes of a tree" based on some criterion.
- The criterion is often cost–complexity pruning

$$\mathcal{P}_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|, \qquad \alpha \geq 0,$$

where T is a binary tree with $|T|$ terminal nodes $R_1, \ldots, R_{|T|}$ and

$$N_m = \sum_{i=1}^{n} 1_{\{\mathbf{X}_i \in R_m\}}, \qquad Q_m(T) = \frac{1}{N_m} \sum_{i=1}^{n} (Y_i - \hat{c}_m)^2 1_{\{\mathbf{X}_i \in R_m\}}.$$

$$\mathcal{P}_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|, \qquad \alpha \geq 0.$$

- Tuning parameter α drives the tradeoff between goodness of fit and complexity:
  - large values of α yields to small trees `(simple models)`
  - small values of α gives large trees. `(complex models)`
- Pruning is done in an iterative way by
  1. collapsing the internal node that gives the smallest quadratic loss increase to get a sub–tree $\tilde{T}$
  2. iterate on the new tree until we get the single–node tree, i.e., new tree is the root.

Dealing with categorical covariates

- Suppose that X_j is a categorical variable with levels $E = \{1, \ldots, K\}$.
- When $K = 2$, splitting is straightforward since $X_j = 1$ or $X_j \neq 1$.
- When $K > 2$, we quickly face a computational burden since there are

$$\frac{\overbrace{2^K}^{\#\text{ partitions}} - \overbrace{2}^{\text{omit } \emptyset \text{ and } E}}{\underbrace{2}_{\text{symmetry}}} = 2^{K-1} - 1$$

  ways to partition E with 2 non overlapping sets.
- The optimal split can be found from only $K - 1$ evaluations (not trivial).

Classification trees

- For classification problems, i.e., $Y \in \{1, \ldots, K\}$, we cannot use the quadratic loss anymore
$$\mathbb{E}_{(Y,\mathbf{X})}\left[(Y - \hat{Y})^2\right].$$

- We must use a measure of non homogeneity, i.e., node impurity,

  Classification error
$$1 - \Pr_{(Y,\mathbf{X})}(\hat{Y} = Y)$$

  Gini's index
$$\mathbb{E}_{(Y,\mathbf{X})}\left[1 - \Pr(\hat{Y} = Y)\right]$$

  Cross–entropy
$$-\mathbb{E}_{(Y,\mathbf{X})}\left[\log \Pr(\hat{Y} = Y)\right]$$

Remember that $\hat{Y}$ obviously is a function of $\mathbf{X}$.

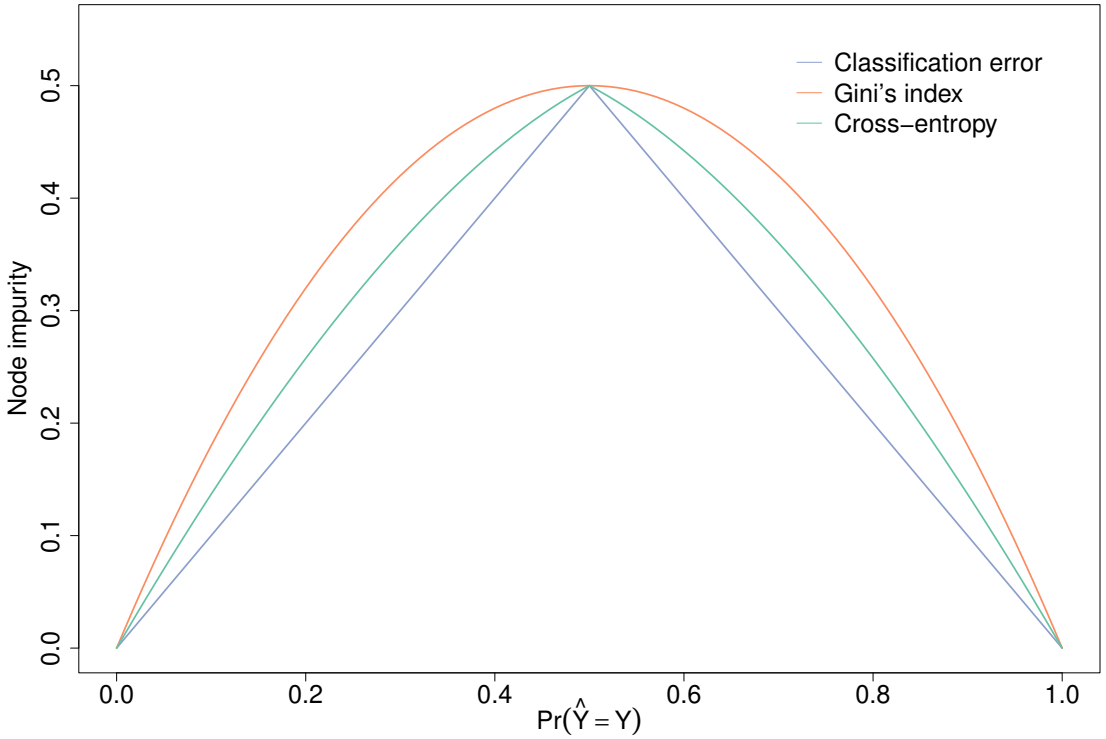

Figure 4: *Different node impurity measures for a binary classification problem. The cross–entropy impurity has been scaled by* $1/\log 2$ *so that it is equal to 0.5 at* $x = 0.5$.

- Similar overall pattern
- Miss-classification is not differentiable
- Gini and cross–entropy are:
  - differentiable: helps optimization
  - favour pure nodes

Gini and cross–entropy should be used to grow and one can use any impurity measure while pruning—miss-classification rate is often use though.

Pros and cons

- 👍 Almost no data pre-processing, e.g., data scaling
- 👍 Robust to outliers
- 👍 Allows for missing values
- 👍 Intuitive and easy to explain to non specialist
- 👍 Kind of interpretable
- 👎 Highly instable: small change in $\mathbf{X}$ may give a completly different answer
- 👎 CPU demanding
- 👎 Prone to overfitting—pruning mitigates this drawback
- 👎 Non continuous predictor in regression
- 👎 High bias for unbalanced designs—must "re-balanced" it

❗ It is highly recommended to not use CART but rather random forests.

3. Random forest

Towards random forest

$$\text{Var}\left(\frac{1}{B}\sum_{b=1}^{B} T_b\right) \approx \rho\sigma^2, \qquad \text{Cov}(T_b, T_{b'}) = \rho\sigma^2, \qquad B \gg 1,\ \rho > 0.$$

- If we have a low (positive) correlation, the variance is reduced.
- We need to find a way to get almost uncorrelated trees.
- We use the same rationale as bagging, i.e.,

  1. Generate a synthetic data set by boostraping both individuals and covariates;
  2. Fit a CART
  3. Repeat

  where
- Having "different dataset" reduces correlations.

Algorithm 1: Random forest for regression and classification.

input : Supervised data set $\mathcal{D}_n = \{(\mathbf{X}_i, Y_i) \colon i = 1, \ldots, n\}$, number of trees B

for $b \leftarrow 1$ **to** B **do**

  Draw a boostrap sample $\mathcal{D}_b$ of size n from $\mathcal{D}_n$;

  Grow a tree T_b from $\mathcal{D}_b$ using the following steps:

  1. Select $\tilde{p}$ variables at random from the p variables
  2. Pick the best variable / split–point among the m
  3. Split the node into two children nodes

Output the ensemble of trees $\{T_b \colon b = 1, \ldots, B\}$ and predictors

Regression (`averaging`)

$$\hat{f}_B \colon \mathbf{x} \longmapsto \frac{1}{B} \sum_{b=1}^{B} T_b(\mathbf{x}),$$

Classification (`majority vote`)

$$\hat{C}_B \colon \mathbf{x} \longmapsto \operatorname*{argmax}_k \sum_{b=1}^{B} 1_{\{T_b(\mathbf{x})=k\}}$$

Recommendations, for regression use $m = \lfloor p/3 \rfloor$ with minimum node size is 5; for classification use $m = \lfloor \sqrt{p} \rfloor$ with minimum node size 1. But these are just guidelines and in practice you should consider fine tuning.

Out of bag samples

Algorithm 2: Bootstraping the observations give OOB sample.

input : Supervised data set $\mathcal{D}_n = \{(\mathbf{X}_i, Y_i) : i = 1, \ldots, n\}$, number of trees B

for $b \leftarrow 1$ **to** B **do**
- Draw a boostrap sample $\mathcal{D}_b$ of size n from $\mathcal{D}_n$;
- Grow a tree T_b from $\mathcal{D}_b$ using the following steps:
  1. Select $\tilde{p}$ variables at random from the p variables
  2. Pick the best variable / split–point among the m
  3. Split the node into two children nodes

- By construction, some observations will be discarded while fitting tree T_b.
- These observations are called Out Of Bag (OOB)
- As a consequence we can estimate the generalization error based on OOB samples

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{N_i}\sum_{b=1}^{B}\text{loss}\{Y_i, T_b(\mathbf{X}_i)\}1_{\{(\mathbf{X}_i,Y_i)\notin\mathcal{D}_b\}}, \qquad N_i = \sum_{b=1}^{B}1_{\{(\mathbf{X}_i,Y_i)\notin\mathcal{D}_b\}}$$

Variable importance

- Within a tree, variable importance can be assessed from the improvement in the splitting criterion.
- Since random forest are collection of tree, we just accumulate these importances over all trees.

4. Feature importance

Motivation

- A CART is (quite) interpretable since it is based on binary splits
- It is more challenging for random forests since we are "averaging" over multiple binary trees.
- How to know which feature has a large impact on predictions?
- This stage is known as variable importance
- Not all statistical models enable variable importance measures but random forests do!
- Let's see how

Mean decrease impurity

- Binary trees split node N into (N_L, N_R) maximizing the decrease in impurity

$$\Delta i(N) = i(N) - \underbrace{\{p(L)i(N_L) + p(R)i(N_R)\}}_{\text{impurity decrease after splitting } N}, \qquad p(L) = \frac{|N_L|}{|N|},\ p(R) = 1 - p(L),$$

where $i(T)$ and $|T|$ are the impurity and cardinal of node T.

- Mean Decrease Impurity (MDI) aggregates over nodes of T

$$\mathsf{MDI}_T(X_j) = \sum_{N \in T} p(N)\,\{i(N) - \Delta i(N)\}\,1_{\{\text{split of } N \text{ uses } X_j\}}.$$

- For a random forest $\mathcal{F} = \{T_1, \ldots, T_B\}$, we average over trees, i.e.,

$$\mathsf{MDI}(\mathcal{F}) = \frac{1}{B}\sum_{b=1}^{B} \mathsf{MDI}(T_b).$$

Boston data set

```
> head(Boston)
     crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat medv
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94 33.4
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33 36.2
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21 28.7
```

- Aim is to predict the price of house (regression problem)
- Sample size is $n = 506$
- We have $p = 13$ covariates (only a few categorical)

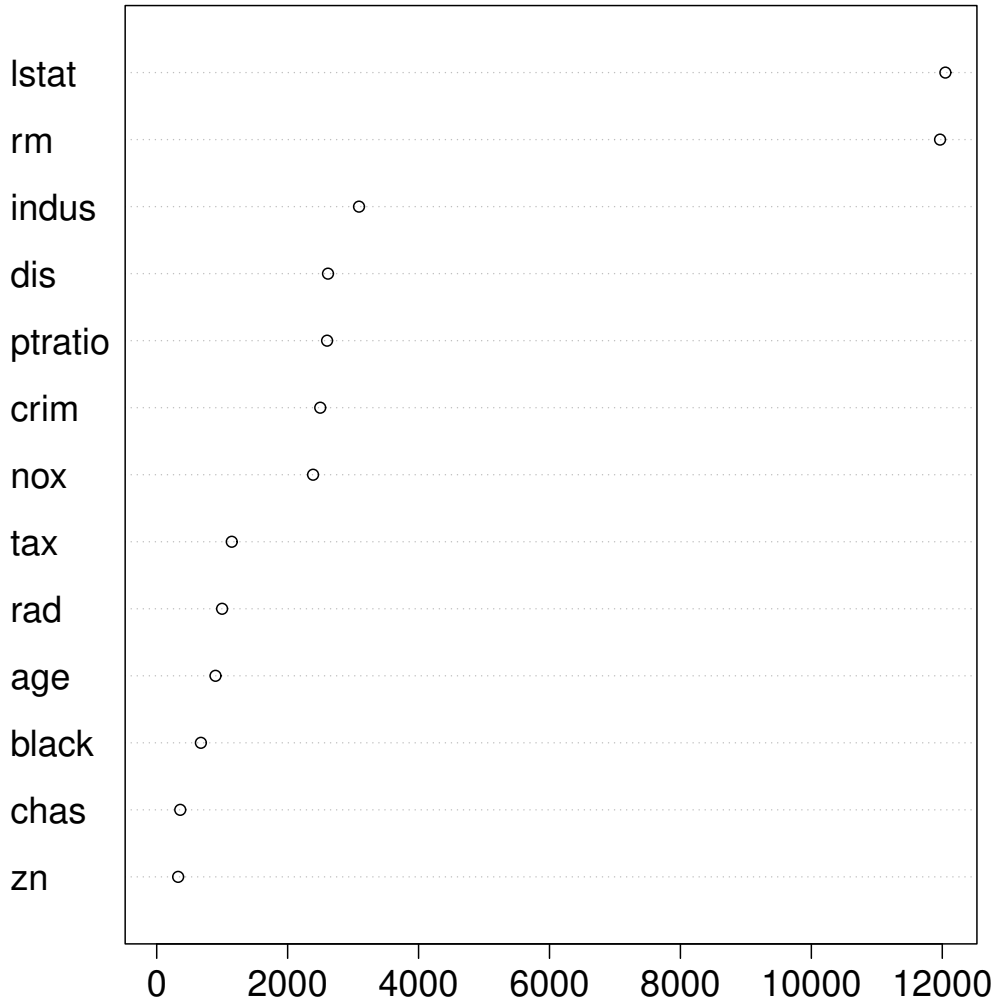

Figure 5: *Mean decrease impurity for the Boston housing regression problem.*

Mean decrease accuracy (a.k.a. permutation importance)

- The Mean Decrease Accuracy (MDA) for a tree (fitted) T is

$$\text{MDA}_T(X_j; \mathcal{D}_n) = \frac{1}{n}\sum_{i=1}^{n} \text{loss}\left\{Y_i, T(\mathcal{D}_{j,n})\right\} - \frac{1}{n}\sum_{i=1}^{n} \text{loss}\left\{Y_i, T(\mathcal{D}_n)\right\},$$

  where $\mathcal{D}_{j,n}$ is similar to the original data $\mathcal{D}_n$ except that feature X_j has been randomly shuffled.
- For a random forest $\mathcal{F} = (T_1, \ldots, T_B)$, we average over all trees, i.e.,

$$\text{MDA}(\mathcal{F}, \mathcal{D}_n) = \frac{1}{B}\sum_{b=1}^{B} \text{MDA}_{T_b}(X_j, \tilde{\mathcal{D}}_{n,b})$$

  where $\tilde{\mathcal{D}}_{n,b}$ is the out-of-bag sample of tree T_b.

Intuitively, if X_j is not influential, prediction performance should not be degraded too much hence MDA should be small.

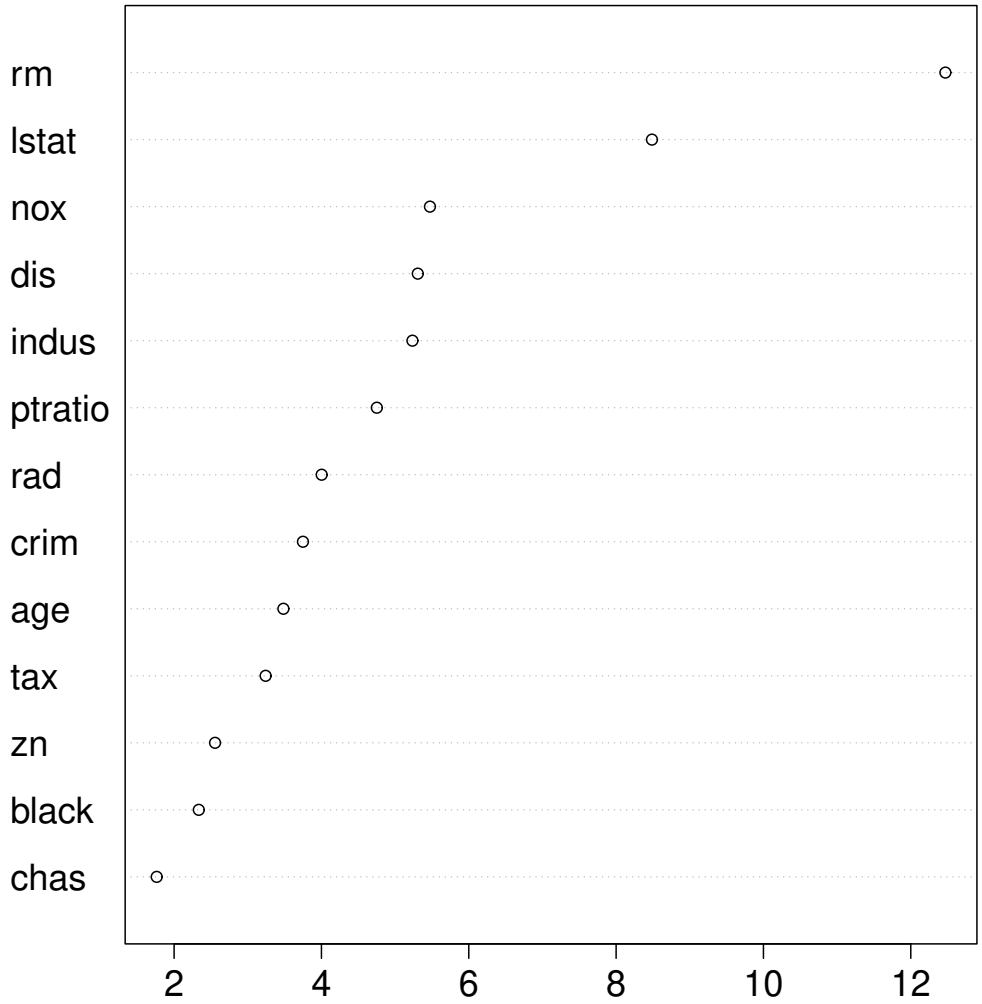

Figure 6: *Mean decrease accuracy for the Boston housing regression problem.*

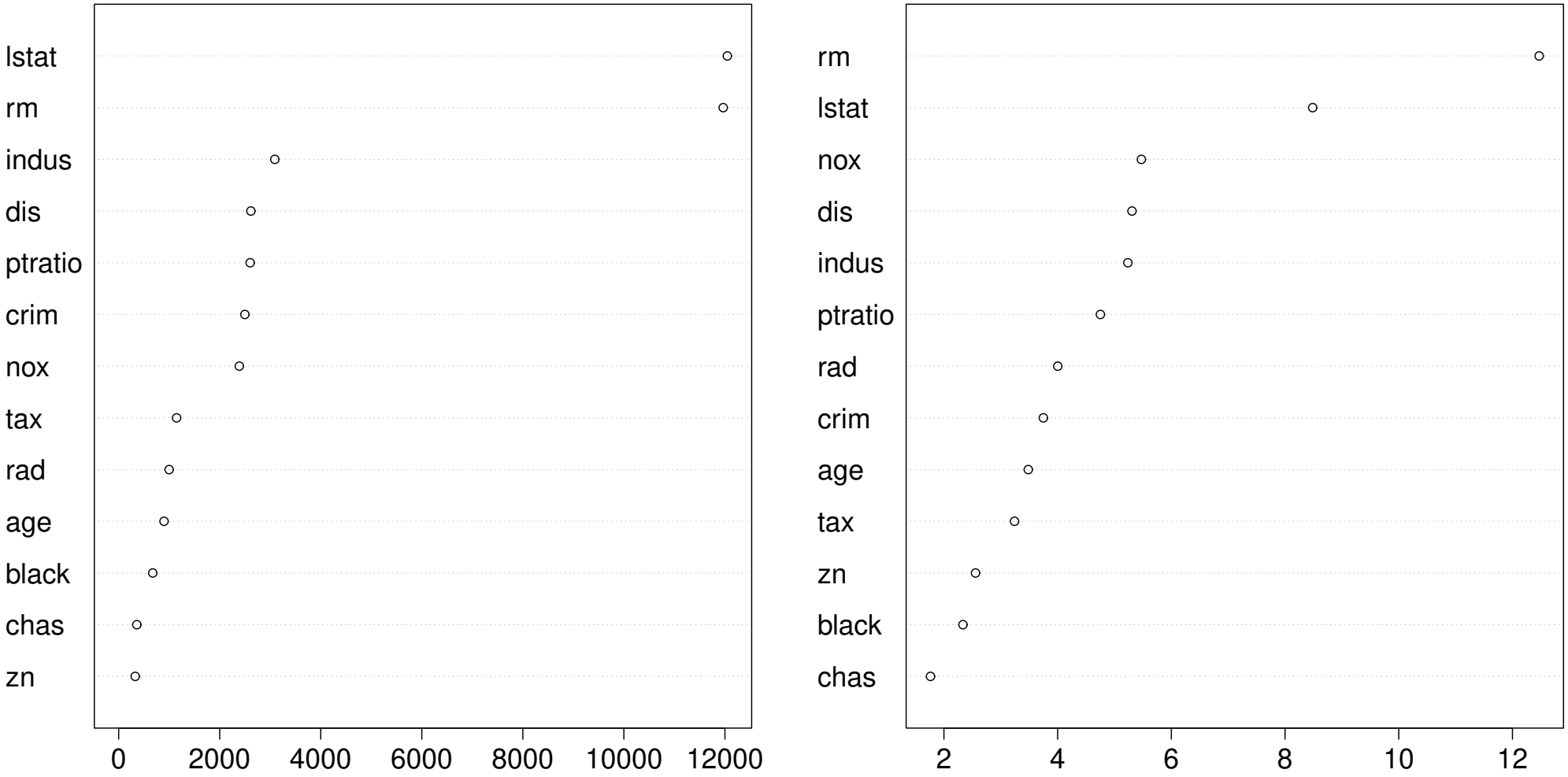

Figure 7: *Comparison of feature importance measures.*

Shapley values

- Shapley values came from game theory and hence are agnostic.
- Within a (coalitional) game with n players, they give the "fair" distribution of the (maximal) profit
- For our concern, we have

  Game predict Y given $\mathbf{X}$;
  Players features;
  Profit model's prediction for an observation $(Y_i, \mathbf{X}_i)$.

- Shapley value of the i–th observation and j–th feature (p features in total) is

$$\text{Shapley}(X_{i,j}) = \frac{1}{p} \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \underbrace{\binom{n-1}{|S|}^{-1}}_{\substack{\text{number of partitions} \\ \text{of size } |S| \text{ without } j}} \underbrace{\{\nu(X_{i,S} \cup X_{i,j}) - \nu(X_{i,S})\}}_{\substack{\text{marginal contribution of} \\ X_{i,j} \text{ to } X_{i,S} \cup X_{i,j}}}$$

Intuitively, if $X_{i,j}$ is not influential, as before prediction performance should not be degraded too much hence Shapley values should be small.

Shapley value estimation

- Recall that Shapley values uses terms of the form $\nu(X_S)$, $S \subseteq \{1, \ldots, p\}$.
- For our concern, it implies to fit the model to all subset S, i.e., $2^p - 1$ models.
- To reduce the computational burden, one can use an estimation based on a marginalization approach to get $\nu(X_{S_1})$ from $\nu(X_{S_1} \cup X_{S_2})$, $S_1 \cap S_2 = \emptyset$, i.e.,

$$\hat{\nu}(X_{i,S_1}) = \frac{1}{n} \sum_{\ell=1}^{n} \nu(X_{i,S_1} \cup X_{\ell,S_2})$$

The above estimator is called Shapley sampling values.

Shapley additive explanation values

- Shapley Addidtive exPlanation (SHAP) values are Shapley values with

$$\nu(X_S) = \mathbb{E}\left[\hat{f}(\mathbf{X}) \mid X_S\right]$$

- From basic probability theory, we easily get

$$\mathbb{E}\left[\hat{f}(\mathbf{X}) \mid X_S = x_S\right] = \int \hat{f}(\mathbf{x}) p(\mathbf{x} \mid x_S) \mathrm{d}x_{-S}.$$

- SHAP values can be estimated in two ways:
  - using the same marginalization strategy, i.e., SHAP sampling values;
  - using a Kernel approach, i.e., Kernel SHAP.
- We will now focus on Kernel SHAP.

Kernel SHAP

- Kernel SHAP assumes independence between covariates, i.e., $X_{S_1} \mid X_{S_2} \sim X_{S_1}$ when $S_1 \cap S_2 = \emptyset$, so that

$$\mathbb{E}\left[\hat{f}(\mathbf{X}) \mid X_S\right] = \int \hat{f}(\mathbf{x}) p(\mathbf{x} \mid x_S) \mathrm{d}x_{-S} = \int \hat{f}(\mathbf{x}) p(x_{-S}) \mathrm{d}x_{-S}.$$

- Using (as often), we can define the Kernel SHAP estimator

$$\hat{\nu}(X_{i,j}) = \frac{1}{L} \sum_{\ell=1}^{J} \hat{f}(X_{i,j}, \tilde{X}_{-j}),$$

where the $X_{\ell,-j}$'s are sampled independently from $X_{i,j}$.

! If covariates are highly dependent, estimates will be completely off. Some variations exists to enable dependent features, e.g., using a multivariate Gaussian distribution.

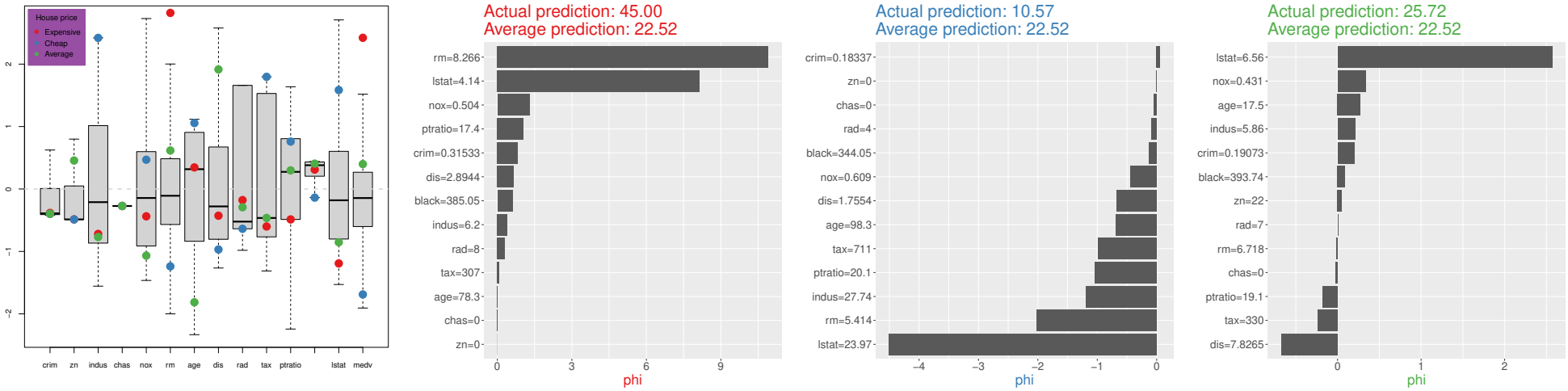

Figure 8: *Boxplot (scaled Boston) and Shapley values for the Boston housing regression problem.*