MA8701 Advanced methods in statistical inference and learning

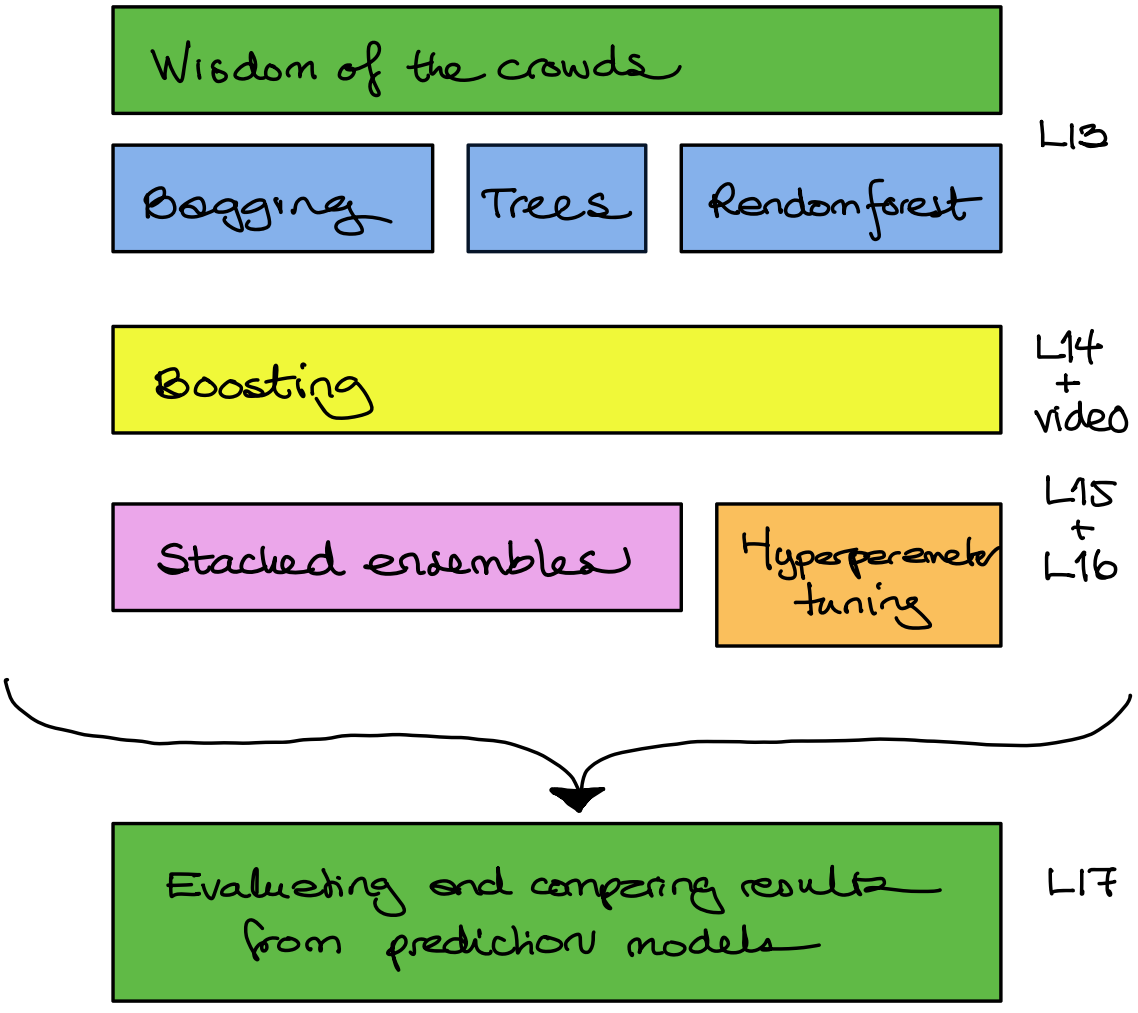
Part 3: Ensembles. L13: Bagging - trees - random forests

Mette Langaas

Lectured

~~2/23/23~~ 24.02.2023

ENSEMBLE LEARNING ← build a prediction model by combining the strengths of an ensemble of (simpler) base models



Literature this lecture (L13)

- [ESL] The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics, 2009) by Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Ebook. Chapter 8.7 (bagging), 9.2 (trees), 15 (random forest, not 15.3.3 and 15.4.3).

Wisdom of the crowds: Vox populi

Francis Galton
Nature (1907)

VOX POPULI.

IN these democratic days, any investigation into the trustworthiness and peculiarities of popular judgments is of interest. The material about to be discussed refers to a small matter, but is much to the point.

A weight-judging competition was carried on at the annual show of the West of England Fat Stock and Poultry Exhibition recently held at Plymouth. A fat ox having been selected, competitors bought stamped and numbered cards, for *6d.* each, on which to inscribe their respective names, addresses, and estimates of what the ox would weigh after it had been slaughtered and "dressed." Those who guessed most successfully received prizes. About 800 tickets were issued, which were kindly lent me for examination after they had fulfilled their immediate purpose. These afforded excellent material. The judgments were unbiassed by passion and uninfluenced by oratory and the like. The sixpenny fee deterred practical joking, and the hope of a prize and the joy of competition prompted each competitor to do his best. The competitors included butchers and farmers, some of whom were highly expert in judging the weight of cattle; others were probably guided by such information as they might pick up, and by their own fancies. The average competitor was probably as well fitted for making a just estimate of the dressed weight of the ox, as an average voter is of judging the merits of most political issues on which he votes, and the variety among the voters to judge justly was probably much the same in either case.

After weeding thirteen cards out of the collection, as being defective or illegible, there remained 787 for discussion. I arrayed them in order of the magnitudes of the estimates, and converted the *cwt.*, *quarters*, and *lbs.* in which they were made, into lbs., under which form they will be treated.

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles		Excess of Observed over Normal
		Observed deviates from 1207 lbs.	Normal p.e =37	
°				
5	1074	−133	−90	+43
10	1109	− 98	−70	+28
15	1126	− 81	−57	+24
20	1148	− 59	−46	+13
q_1 25	1162	− 45	−37	+ 8
30	1174	− 33	−29	+ 4
35	1181	− 26	−21	+ 5
40	1188	− 19	−14	+ 5
45	1197	− 10	− 7	+ 3
m 50	1207	0	0	0
55	1214	+ 7	+ 7	0
60	1219	+ 12	+14	− 2
65	1225	+ 18	+21	− 3
70	1230	+ 23	+29	− 6
q_3 75	1236	+ 29	+37	− 8
80	1243	+ 36	+46	−10
85	1254	+ 47	+57	−10
90	1267	+ 52	+70	−18
95	1293	+ 86	+90	− 4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

According to the democratic principle of "one vote one value," the middlemost estimate expresses the *vox populi*, every other estimate being condemned as too low or too high by a majority of the voters (for fuller explanation see "One Vote, One Value," NATURE, February 28, p. 414). Now the middlemost estimate is 1207 lb., and the weight of the dressed ox proved to be 1198 lb.; so the *vox populi* was in this case 9 lb., or 0·8 per cent. of the whole weight too high. The distribu-

What is a wise crowd?

James Surowiecki: The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations, 2004 as presented at https://en.wikipedia.org/wiki/The_Wisdom_of_Crowds

- Diversity of opinion: Each person should have private information even if it is just an eccentric interpretation of the known facts. (Chapter 2)
- Independence: People's opinions are not determined by the opinions of those around them. (Chapter 3)
- Decentralization: People are able to specialize and draw on local knowledge. (Chapter 4)
- Aggregation: Some mechanism exists for turning private judgements into a collective decision. (Chapter 5)
- Trust: Each person trusts the collective group to be fair. (Chapter 6)

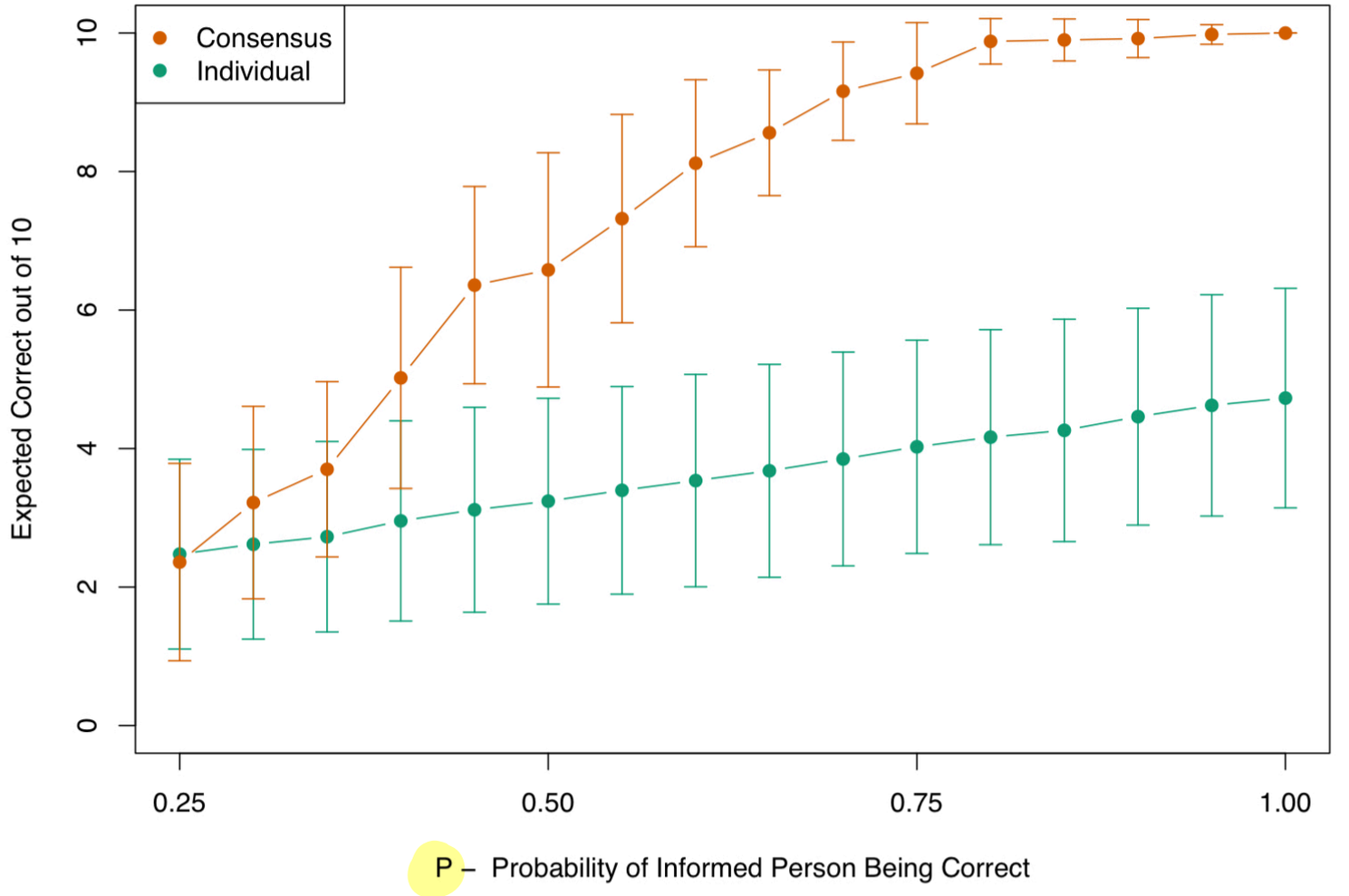
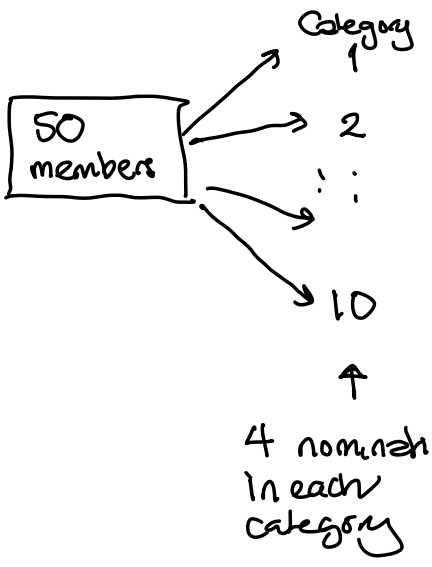


FIGURE 8.11. *Simulated academy awards voting.* 50 *members vote in* 10 *categories, each with* 4 *nominations. For any category, only* 15 *voters have some knowledge, represented by their probability of selecting the "correct" candidate in that category (so* $P = 0.25$ *means they have no knowledge). For each category, the* 15 *experts are chosen at random from the* 50*. Results show the expected correct (based on* 50 *simulations) for the consensus, as well as for the individuals. The error bars indicate one standard deviation. We see, for example, that if the* 15 *informed for a category have a* 50% *chance of selecting the correct candidate, the consensus doubles the expected performance of an individual.*

Simulated Oscer's



"There is a "true" winner in each category"

for any category

35/50 vote randomly:

P(correct vote) = $\frac{1}{4}$

15/50 have some knowledge:

P(correct vote) = p (in figure)

$\left(\frac{35}{50}\cdot\frac{1}{4} + \frac{15}{50}\cdot p\right)$ = P(correct vote in categ. j)

E(correct in 10 cat)

$= 10\cdot\left(\frac{35}{50}\cdot\frac{1}{4} + \frac{15}{50}\cdot p\right)$

$p=0 \Rightarrow EC \quad 10\cdot\frac{35}{200} = \frac{7}{4}$

How can we construct wise crowds for prediction?

- Draw many samples from a population
- Each sample: fit a model

- Take the average of the prediction

$Z = \{x_i, y_i\}_{i=1}^{N}$ original data $Z \sim P$

$f_{ag} = E_P(\hat{f}^*(x))$ where $\hat{f}^*(x)$ is the estimated model fit to $Z^* \sim P$

may estimate by $\frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$

$E_P(Y - \hat{f}^*(x))^2$ ← single model

$= E_P\left((Y - f_{ag}(x) - f_{ag}(x) - \hat{f}^*(x))^2\right)$

$= E_P\left((Y - f_{ag}(x))^2 + (\hat{f}^*(x) - f_{ag}(x))^2 + 2(Y - f_{ag}(x))(f_{ag}(x) - \hat{f}^*(x))\right)$

$= E_P\left((Y - f_{ag}(x))^2\right) + \underbrace{E_P\left((\hat{f}^*(x) - f_{ag}(x))^2\right)}_{\text{Var}_P(\hat{f}^*(x)) \geq 0} + \underbrace{E_P(\hat{f}^*(x)) = f_{ag}}_{0}$

$\geq E_P(Y - f_{ag}(x))^2$

went to estimate this

$\Rightarrow$ aggregation never increases MSE

does not hold for 0-1 Loss

often: have only one data set & focus on one good model

Bagging

(bootstrap aggregation)

1) What is it?
2) Why is it a good idea?
3) Connect to Part 1: OOB
4) When to use it?

Training data $Z = (x_i, y_i)_{i=1}^N$

↓

Draw B bootstrap samples $Z^* = \{x_i^*, y_i^*\}_1^N$

↓

Fit a model $\hat{f}^b$ (regression)

or $\hat{G}^b$ (classification) or $\hat{p}_k^b$

to each boot sample $k = 1, \ldots, K$

↓

Regression: $\hat{f}_{bag}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^b(x)$

Classification: $\hat{G}_{bag}(x) =$ majority vote $\{\hat{G}^b(x)\}_{k=1}^K$

or $\underset{k}{\text{argmax}} \frac{1}{B} \sum_{b=1}^{B} \hat{p}_k^b(x)$

Why is it a good idea?

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{b}(x) \xrightarrow{B\to\infty} \text{is an estimator of } E_{\hat{P}}(\hat{f}^{*}(x))$$

$\hat{P}$ is empirical distribution with $\frac{1}{N}$ for each of (x_i, y_i)

$$(x_i^*, y_i^*) \sim \hat{P}$$

thus - an estimate of $f_{bag}(x)$

In addition:

$T_1, \dots, T_B$ i.i.d $\quad E(T_b) = \mu$

$Var(T_b) = \sigma^2$

$Cov(T_b, T_c) = 0 \quad b \neq c$

some RV could be $\hat{f} = 0$

$$\bar{T} = \frac{1}{B} \sum_{b=1}^{B} T_i \qquad E(\bar{T}) = \mu$$

$$Var(\bar{T}) = \frac{\sigma^2}{B}$$

Average is a more precise estimate!

If $T_1, \dots, T_B$ is based on bootstrapping then

$Cov(T_b, T_c) \neq 0$.

What if $Corr(T_b, T_c) = \rho$ compound symmetry

What is then $Var(\bar{T})$?

Exercise: $\rho \sigma^2 + \frac{1-\rho}{B} \cdot \sigma^2$

Interpret this result!

$B \to \infty$

$\rho \sigma^2$ ↓ 0

↑

Improvement possible if the T's are decorrelated:

Connect to Part 1: Out-of-bag error estimation

- We use a subset of the observations in each bootstrap sample. We know that the probability that an observation is in the bootstrap sample is approximately $1 - e^{-1} = 0.6321206$ (0.63212).
- when an observation is left out of the bootstrap sample it is not used to build the ~~tree~~ model, and we can use this observation as a part of a "test set" to measure the predictive performance and error of the fitted model, $f^{*b}(x)$.

In other words: Since each observation i has a probability of approximately $2/3$ to be in a bootstrap sample, and we make B bootstrap samples, then observation i will be outside the bootstrap sample in approximately $B/3$ of the fitted ~~trees.~~ models

The observations left out are referred to as the *out-of-bag* observations, and the measured error of the $B/3$ predictions is called the *out-of-bag error*.

When should we use bagging?

Breiman originally contructed bagging for classification and regression trees! Aim: combat the high variance of trees!

Bagging can be used for many types of predictors in addition to trees (regression and classification) according to Breiman (1996):

- the vital element is the instability of the prediction method
- if perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

Breiman (1996) suggests that these methods should be suitable for bagging:

- neural nets, classification and regression trees (CART), subset selection in linear regression

however not nearest neighbours - since

- the stability of nearest neighbour classification methods with respect to perturbations of the data distinguishes them from competitors such as trees and neural nets.

Q: $\hat{f}(x) = X(X^TX)^{-1}X^TY$ $\qquad Y = X\beta + \varepsilon$

Parametric bootstrap: $y^* = \hat{f}(x) + \varepsilon^*$ ← draw

keep X, generate new $y = y^*$

$$\hat{f}^{*b}(x) = X(X^TX)^{-1}X^T y^{*b} = X(X^TX)^{-1}X^T(\hat{f}(x) + \varepsilon_b^*)$$

$$= X(X^TX)^{-1}X^T\left(X(X^TX)^{-1}X^T y + \varepsilon_b^*\right)$$

$$= \underbrace{X(X^TX)^{-1}X^T}_{H} y + H\varepsilon_b^*$$

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) = \frac{1}{B}\sum_{b=1}^{B}\left[HY + H\varepsilon_b^*\right]$$

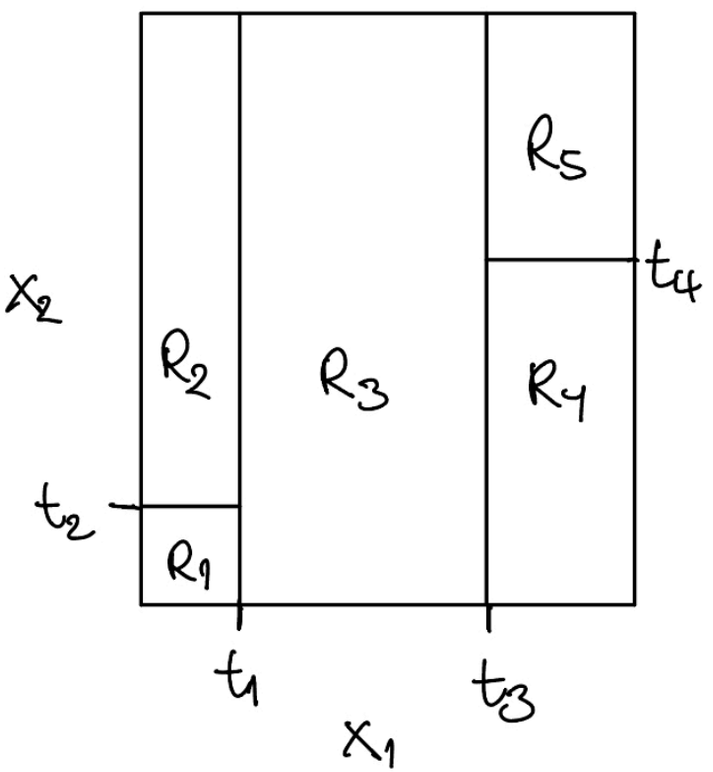
$$= HY + H\underbrace{\frac{1}{B}\sum_{b=1}^{B}\varepsilon_b^*}_{\to 0} \approx HY = \hat{f}(x)$$

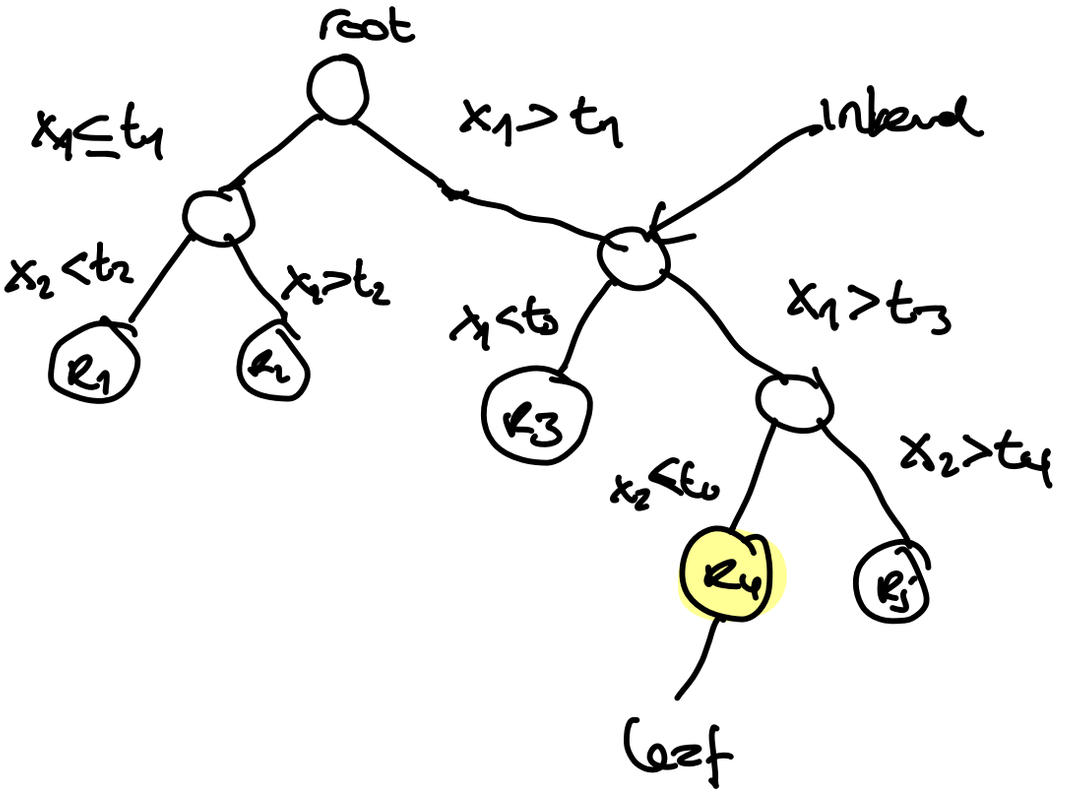
$\hat{f}_{bag}(x)$ "reproduces" the original fit ← more effect of using bagging for nonlinear and adaptive function of data

Review of trees - through 4 questions

1) From non-overlapping regions in predictor space to a roted decision tree

Draw the binary decision tree corresponding to the predictor space regions. Mark root, branch, internal node, leaf node.





2) Tree prediction: what are the missing estimates?

Regression

$$\hat{f}(X_i) = \sum_{m=1}^{M} \hat{c}_m I(X_i \in R_m)$$

where $\hat{c}_m$ is the estimate for region R_m.

Classification

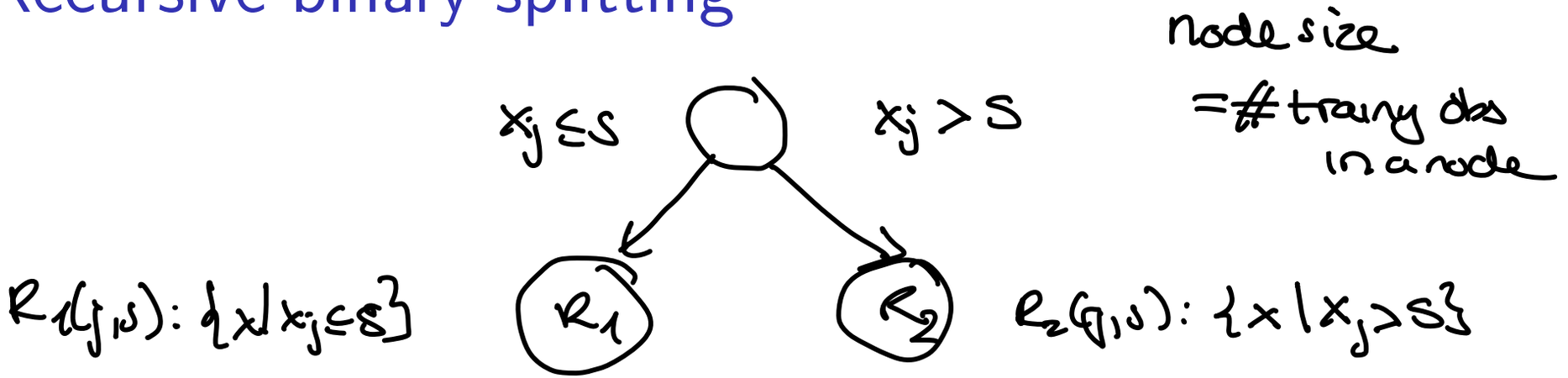
- Majority vote: Predict that the observation belongs to the most commonly occurring class of the training observations in R_m.
- Estimate the probability that an observation x_i belongs to a class k, $\hat{p}_{mk}(x_i)$, and then classify according to a threshold value.

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

$k = 1, \ldots K$

$m = 1, \ldots, M$

3) Recursive binary splitting



a ▶ We look for a split point s on variable j. What to minimize?
b ▶ Why recursive binary splitting?
c ▶ When to stop growing a tree?

a) Regression

$$\min_{(j,s)} \left[\min_{c_1} \sum_{i: x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{i: x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

b) Two challenges: partition predictor space and construct tree → can only be done optimally for small number of covariates.
→ go for a greedy method

↑ only optimal at each step – not necessarily globally

c) Node size, number of splits, full tree + prune
one split = stump

Classification

Some *measure of impurity* of the node. For leaf node (region) m and class $k = 1, \ldots, K$:

Gini index:

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

Cross entropy:

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

Here $\hat{p}_{mk}$ is the proportion of training observation in region m that are from class k.

Remark: the deviance is a scaled version of the cross entropy. $-2\sum_{k=1}^{K} n_{mk} \log \hat{p}_{mk}$ where $\hat{p}_{mk} = \frac{n_{mk}}{N_m}$. Ripley (1996, page 219).

When making a split in our classification tree, we want to minimize the Gini index or the cross-entropy.

The Gini index can be interpreted as the expected error rate if the label is chosen randomly from the class distribution of the node. According to Ripley (1996, page 217) Breiman et al (CART) preferred the Gini index.

4) Pros and cons of trees

- model nonlinearity
- handle NA's
- graphically nice ← human readable
- too big + lose interpret of bagging trees
- sensitive to change in data
- automatically model interactions

Advantages (+) of using trees

- Trees automatically select variables
- Tree-growing algorithms scale well to large n, growing a tree greedily
- Trees can handle mixed features (continuouos, categorical) seamlessly, and can deal with missing data
- Small trees are easy to interpret and explain to people
- Some believe that decision trees mirror human decision making
- Trees can be displayed graphically
- Trees model non-linear effects
- Trees model interactions between covariates
- Trees handle missing data in a smart way!
- Outliers and irrelevant inputs will not affect the tree.

There is no need to specify the functional form of the regression curve or classification border - this is found by the tree automatically.

Disadvantages (-) of using trees

- Large trees are not easy to interpret
- Trees do not generally have good prediction performance (high variance)
- Trees are not very robust, a small change in the data may cause a large change in the final estimated tree
- Trees do not produce a smooth regression surface.

Regression example: Boston housing

James et al. (2013) Section 8.3.4.

Information from https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html.

- Collected by the U.S Census Service concerning housing in the area of Boston Massachusetts, US.
- Two tasks often performed: predict nitrous oxide level (nox), or predict the median value of a house with in a "town" (medv).

Variables

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - #1000(Bk - 0.63)^2# where Bk is the proportion of African Americans by town (black below)
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's (seems to be a truncation)

Handling missing covariates in trees

Instead of removing observation with missing values, or performing single or multiple imputation, there are two popular solutions to the problem for trees:

Make a "missing category"

If you believe that missing covariates behave in a particular way (differently from the non-missing values), we may construct a new category for that variable.

Look at the Boston default tree with `tree` and `rpart` to see how the two handles ONE missing value that we have CONSTRUCTED

```
[1] "tree package"
```

lstat = NA

ONE OBS

```
     crim zn indus chas   nox    rm  age  dis rad tax ptrat
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.09   1 296    15
```

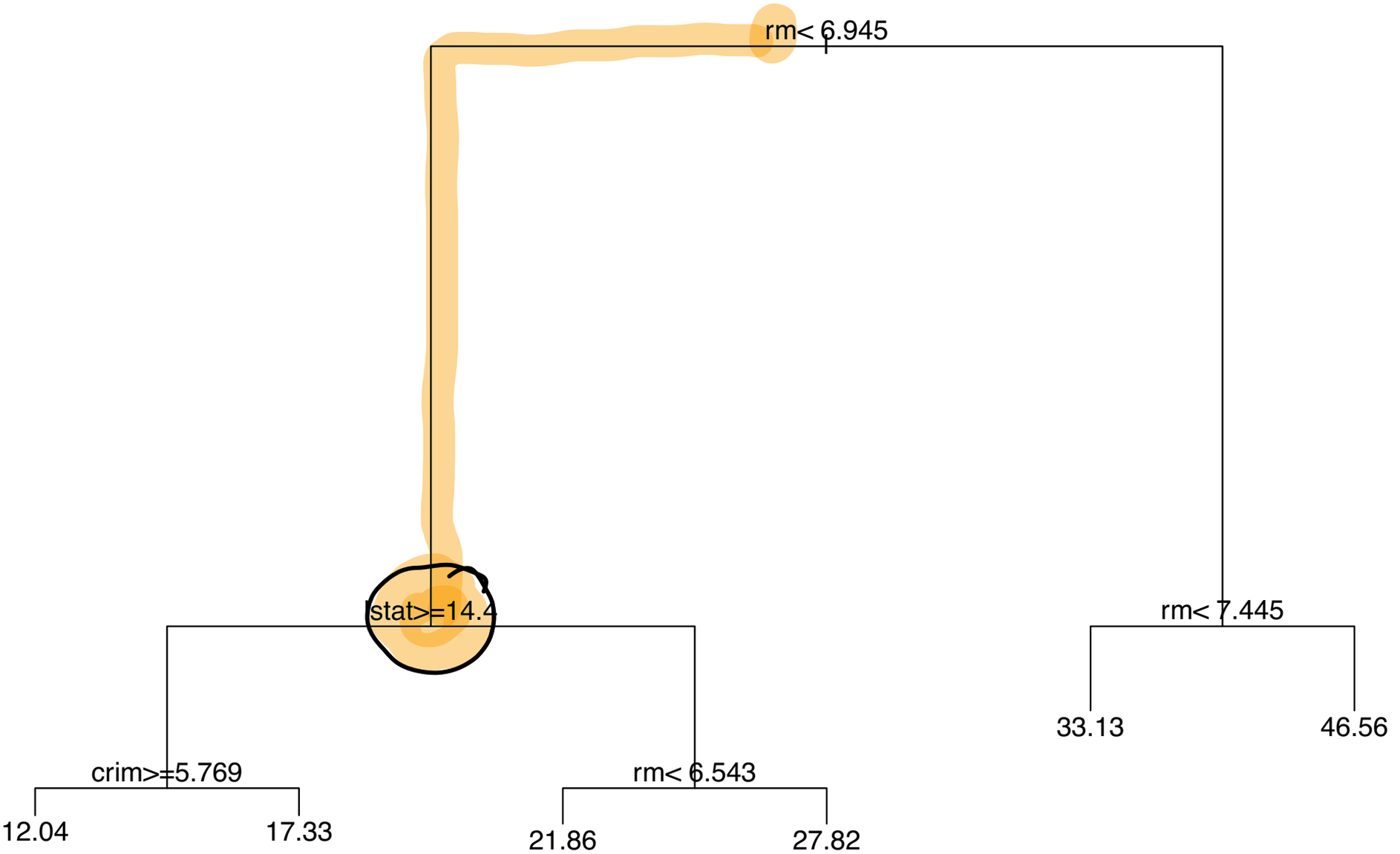
```
       1
19.8223
```

1 → f(x) → NA for lstat

```
[1] "rpart package"
```

```
       1
27.82308
```

```r
boston.rpart <- rpart(formula = medv~. , data = Boston,subset=train)
plot(boston.rpart)
text(boston.rpart,pretty=0)
```



```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 354 32270.0 22.95
   2) rm < 6.945 296 10830.0 19.82
     4) lstat < 14.405 177  3681.0 23.17
       8) rm < 6.543 138  1690.0 21.86 *
       9) rm > 6.543 39   908.2 27.82 *
     5) lstat > 14.405 119  2215.0 14.84
      10) crim < 5.76921 63   749.9 17.33 *
      11) crim > 5.76921 56   636.1 12.04 *
   3) rm > 6.945 58  3754.0 38.92
     6) rm < 7.445 33   749.7 33.13 *
     7) rm > 7.445 25   438.0 46.56 *
```

Use surrogate splits

The best split at a node is called the *primary split*.

An observation with missing value for variable x_1 is dropped down the tree, and arrive at a split made on x_1.

A "fake" tree is built to predict the split, and the observation follows the predicted direction in the tree. This means that the correlation between covariates are exploited - and the higher the correlation between the primary and predicted primary split - the better.

This is called a *surrogate split*.

If the observation is missing the surrogate variable, there is also a back-up surrogate variable that can be used (found in a similar fashion.)

If the surrogate variable is not giving more information than following the majority of the observations at the primary split, it will not be regarded as a surrogate variable.

Look at the Boston default tree with `tree` and `rpart` to see how the two handles ONE missing value that we have CONSTRUCTED

```
[1] "tree package"
```

one obs

```
      crim zn indus chas   nox    rm  age  dis rad tax ptrat
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.09   1 296    15
```

```
       1
19.8223
```

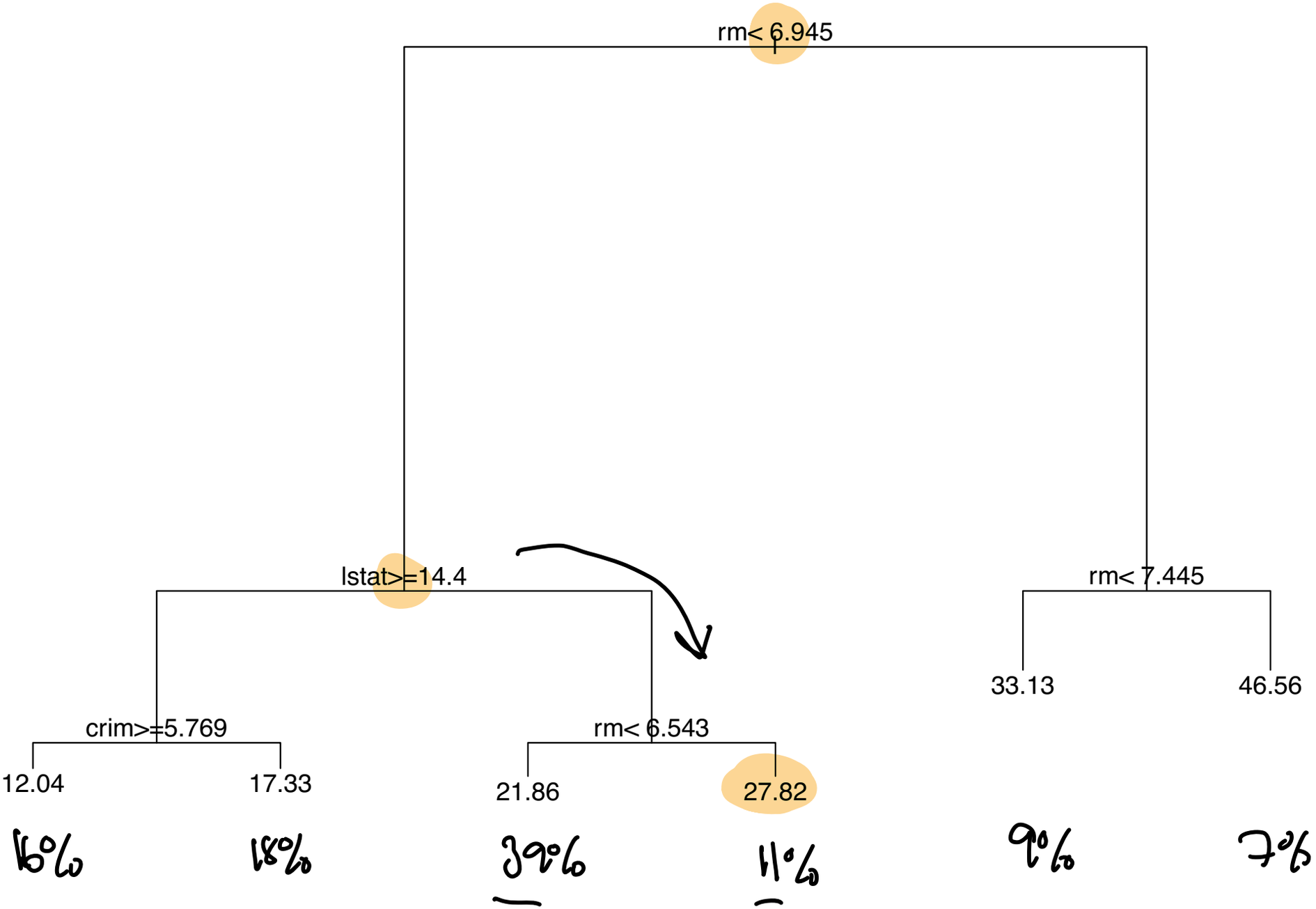
FORCESET

lstat = NA

```
[1] "rpart package"
```

```
        1
27.82308
```

```r
boston.rpart <- rpart(formula = medv~. , data = Boston,subset=train)
plot(boston.rpart)
text(boston.rpart,pretty=0)
```



ANOTHER POSSIBILITY

The R package `rpart` vignette page 18 gives the following example:

- Assume that the split (age $\leqslant$40, age$>$40) has been chosen.
- Surrogate variables are found by *re-applying the partitioning algorithm* (without recursion=only one split?) to predict the two categories age $<$40 vs. age$\geqslant$40 using the other covariates.
- Using "number of misclassified"/"number of observations" as the criterion: the optimal split point is found for each covariate.
- A competitor is the majority rule - that is, go in the direction of the split where the majority of the training data goes. This is given misclassification error min(p, 1 $-$ p) where p = (# in A with age $<$ 40) / nA.
- A ranking of the surrogate variables is done based on the misclassification error for each surrogate variable, and variables performing better than the majority rule is kept.

Choosing B

- The number B is chosen to be as large as "necessary".
- An increase in B will not lead to overfitting, and B is not regarded as a tuning parameter.
- If a goodness of fit measure is plotted as a function of B (soon) we see that (given that B is large enough) increasing B will not change the goodness of fit measure.

Bagging with trees - summing up

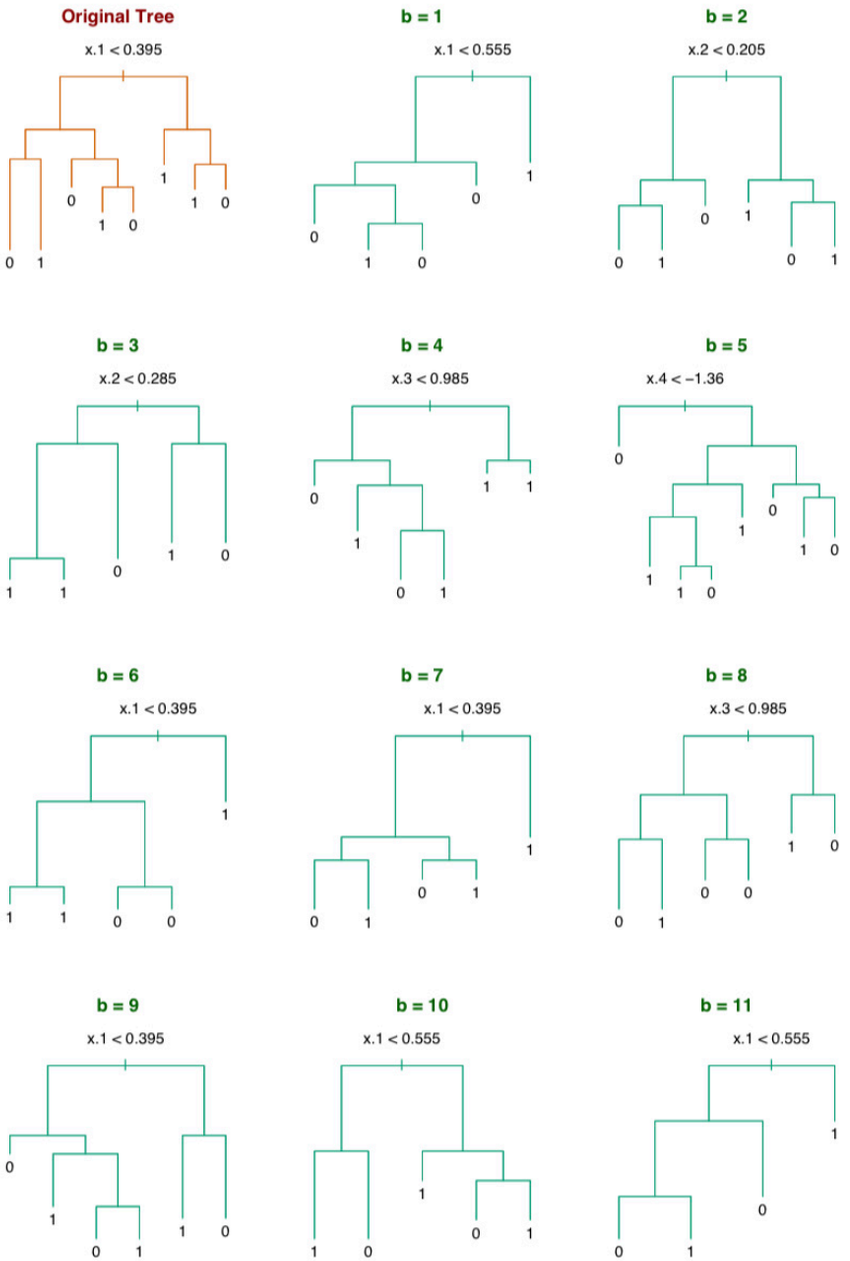


FIGURE 8.9. *Bagging trees on simulated dataset. The top left panel shows the original tree. Eleven trees grown on bootstrap samples are shown. For each tree, the top split is annotated.*

high variance →

bagging improves this!

Bagging with trees - summing up

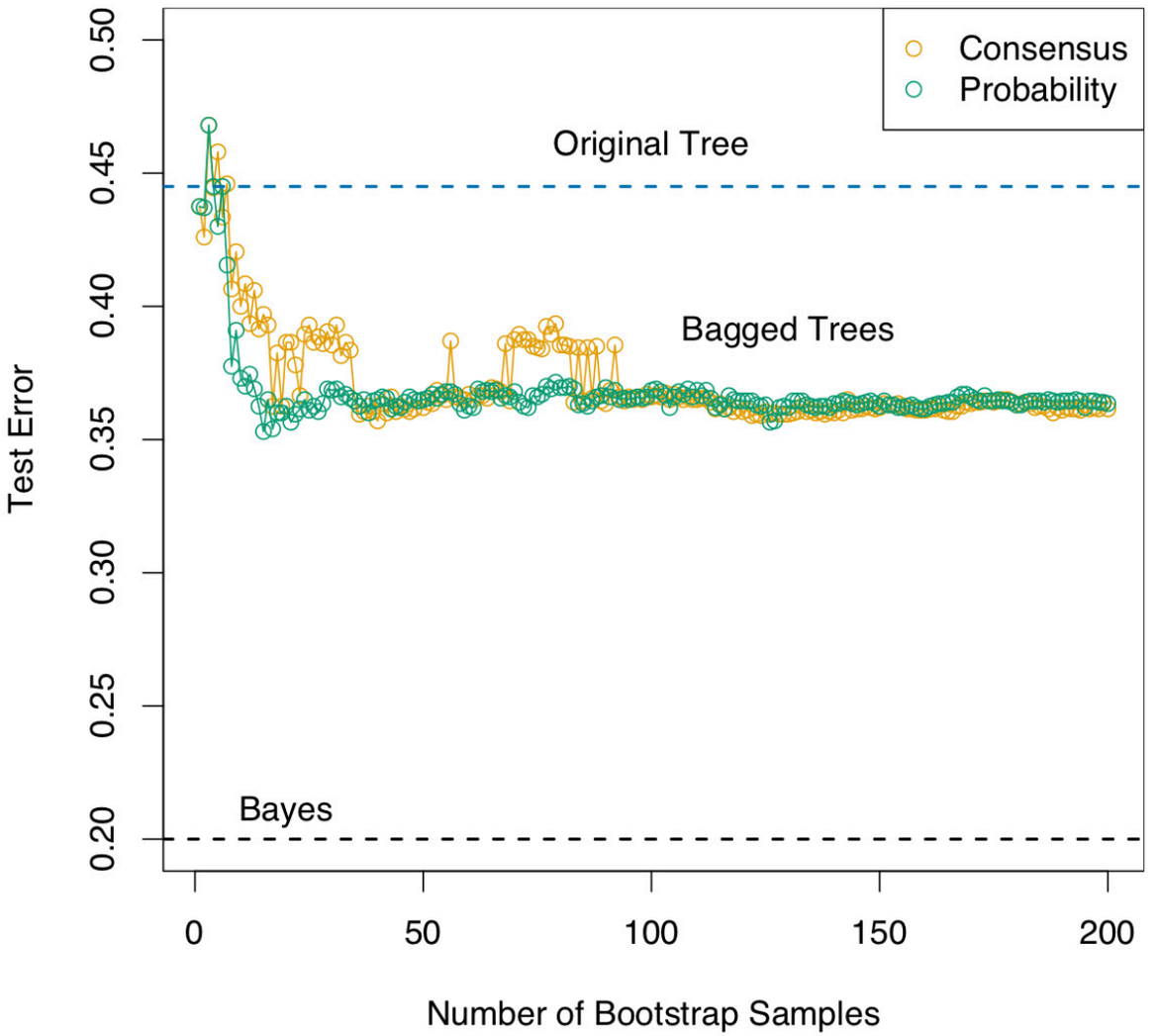


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

Random forest ← is a method to decorrelate trees

If there is a strong predictor in the dataset, the decision trees produced by each of the bootstrap samples in the bagging algorithm becomes very similar: Most of the trees will use the same strong predictor in the top split.

Random forests is a solution to this problem and a method for decorrelating the trees. The hope is to improve the variance reduction.

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.

only difference to bagging

bagging $m = p$

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Figure 8: Hastie, Tibshirani, and Friedman (2009) Figure 15.1

OOB

When the OOB error stabilizes the B is large enough and we may stop training.

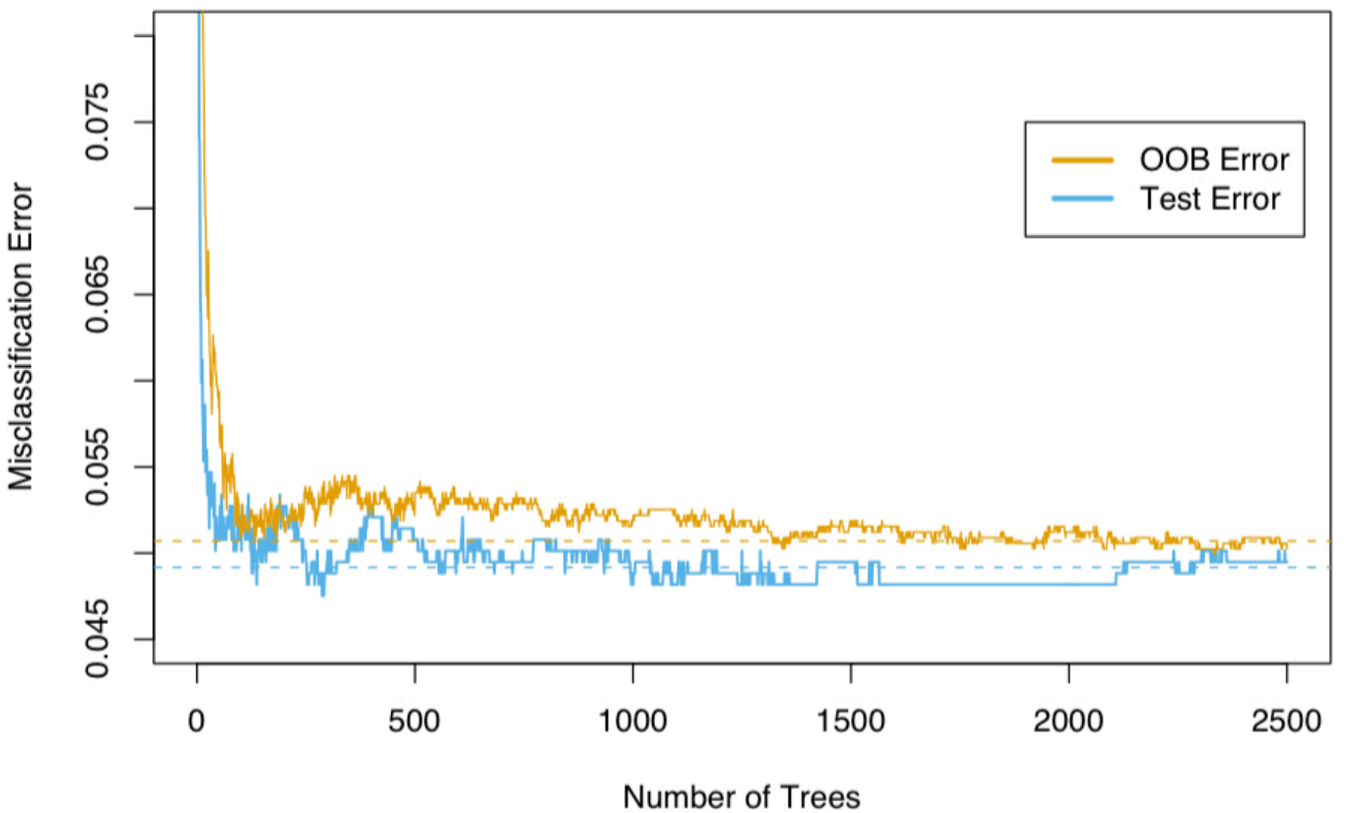


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Figure 9: Hastie, Tibshirani, and Friedman (2009) Figure 15.4

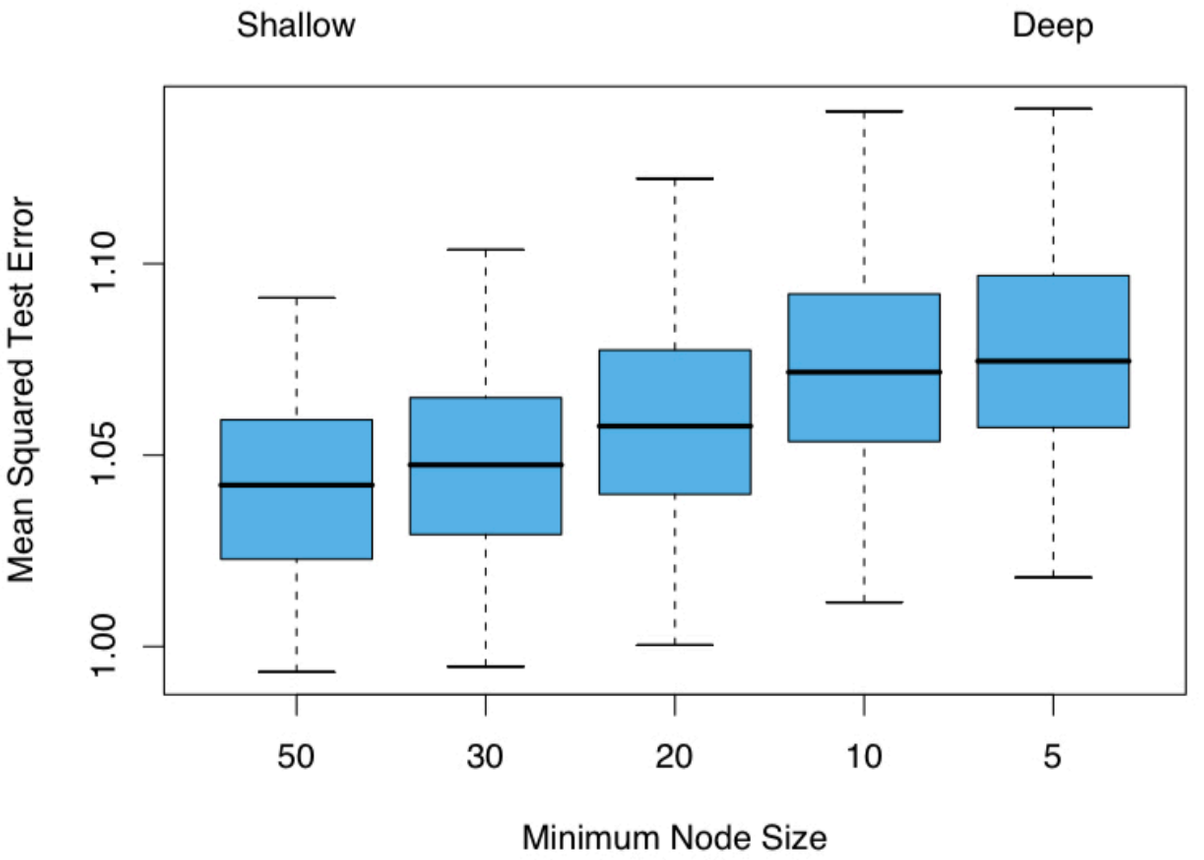


FIGURE 15.8. *The effect of* tree size *on the error in random forest regression. In this example, the true surface was additive in two of the* 12 *variables, plus additive unit-variance Gaussian noise. Tree depth is controlled here by the minimum node size; the smaller the minimum node size, the deeper the trees.*

Figure 7: Hastie, Tibshirani, and Friedman (2009) Figure 15.8

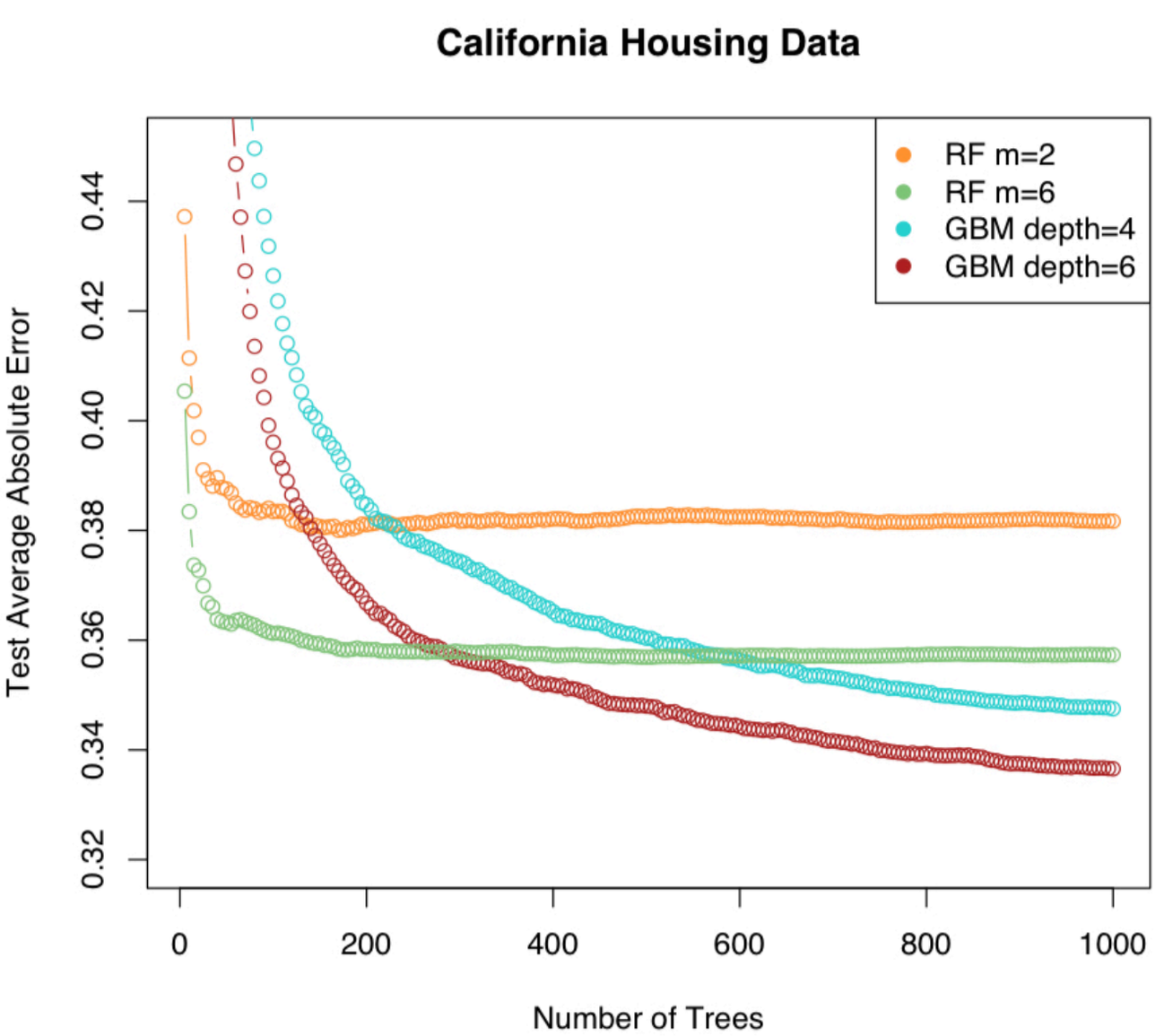


FIGURE 15.3. *Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with* $m = 2$ *and* $m = 6$*. The two gradient boosted models use a shrinkage parameter* $\nu = 0.05$ *in (10.41), and have interaction depths of* 4 *and* 6*. The boosted models outperform random forests.*

NEXT WEEK: BOOSTING

MON 27.02: Lecture w/ Mette Ch 10 ESL

FRI 03.02: Watch video w/ Benat on Ch 10 + xgboost article

→ LINK PUT ON Bb under announcements

↑

Penopto & slides