HT2015: SC4
Statistical Data Mining and Machine Learning

Dino Sejdinovic
Department of Statistics
Oxford

http://www.stats.ox.ac.uk/~sejdinov/sdmml.html

Decision Trees

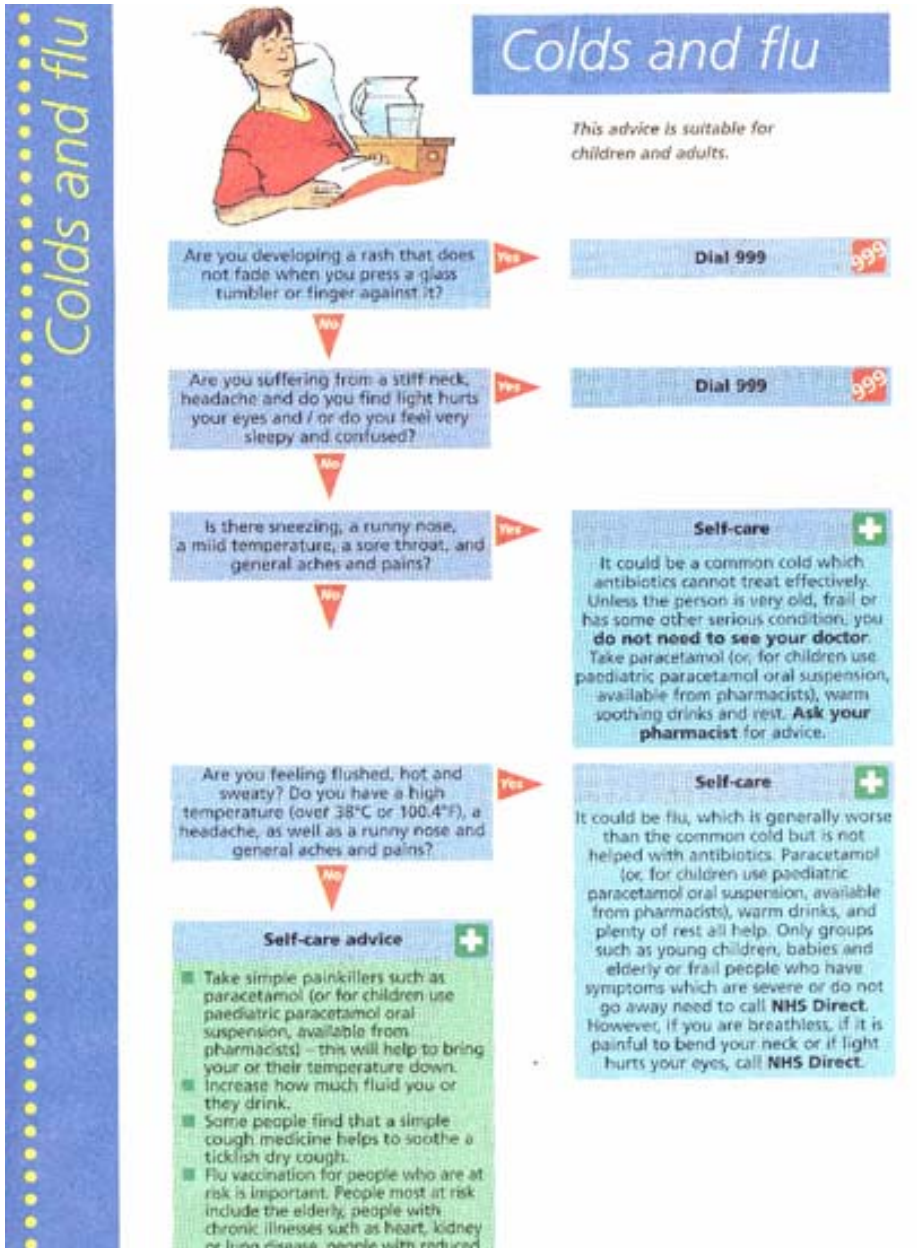
Classification and Regression Trees (**CART**)

- Denote input domain by $\mathcal{X}$ and let the output domain be $\mathcal{Y} = \{1, \ldots, K\}$ (classification) or $\mathcal{Y} = \mathbb{R}$ (regression).
- A decision tree gives a partition of $\mathcal{X}$ into R disjoint sets (regions) $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$, such that the fitted decision function is constant on each region $\mathcal{R}_j \subset \mathcal{X}$, $j = 1, \ldots, R$, i.e.

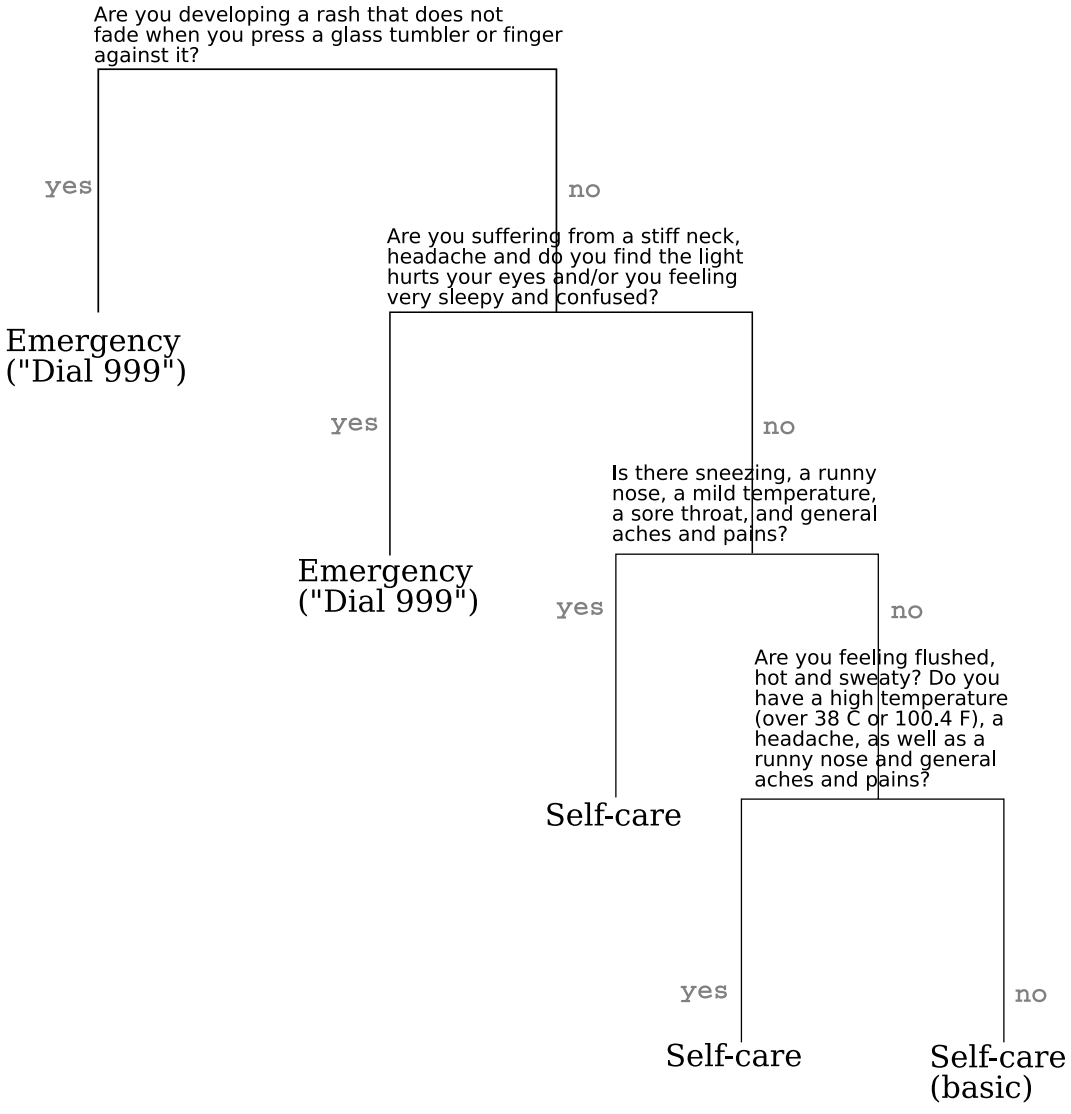
$$f_{\text{tree}}(x) = \beta_j, \forall x \in \mathcal{R}_j.$$

- Main strengths: easy to use, easy to interpret.
- Often serve as a starting point for powerful model combination and ensemble techniques: bagging, boosting (random forests).

Example: NHS Direct Self-help Guide



Example: NHS Direct Self-help Guide



Decision Trees

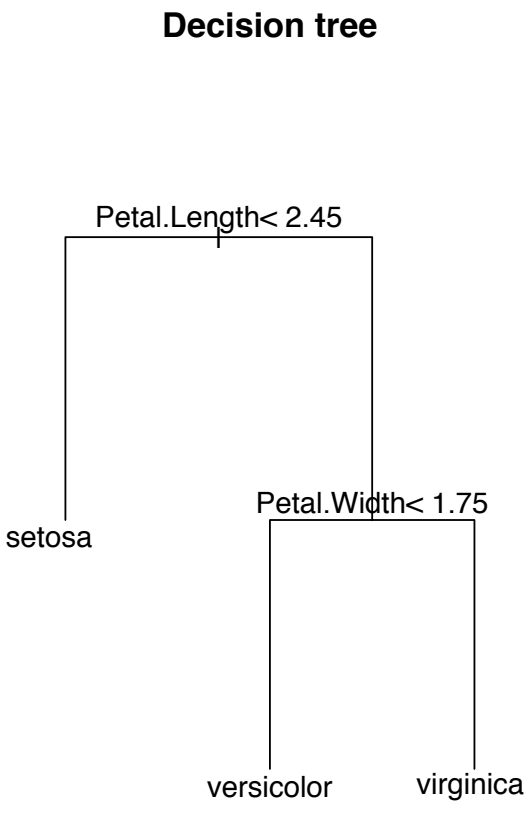
- A decision tree is a hierarchically organized structure, with each node splitting the data space into regions based on value of a single feature (attribute).
- Some terminology:
  - **Parent** of a node c is the node with an arrow pointing into c.
  - **Children** of a node c are those nodes which have node c as a parent.
  - **Root node** is the top node of the tree; the only node without parents.
  - **Leaf nodes** are nodes which do not have children.
  - **Stumps** are trees with just the root node and two leaf nodes.
  - A K**-ary tree** is a tree where each node (except for leaf nodes) has K children. Usually working with binary trees ($K = 2$).
  - The **depth** of a tree is the maximal length of a path from the root node to a leaf node.
- Partition of $\mathcal{X}$ into R disjoint sets $(\mathcal{R}_1, \ldots, \mathcal{R}_R)$ is determined by the **leaves of the tree**.
- On each region $\mathcal{R}_j$ the same decision/prediction is made: $f_{\text{tree}}(x) = \beta_j$ for all $x \in \mathcal{R}_j$ - typically as a majority vote of the data items associated to that leaf (classification) or as their mean (regression)

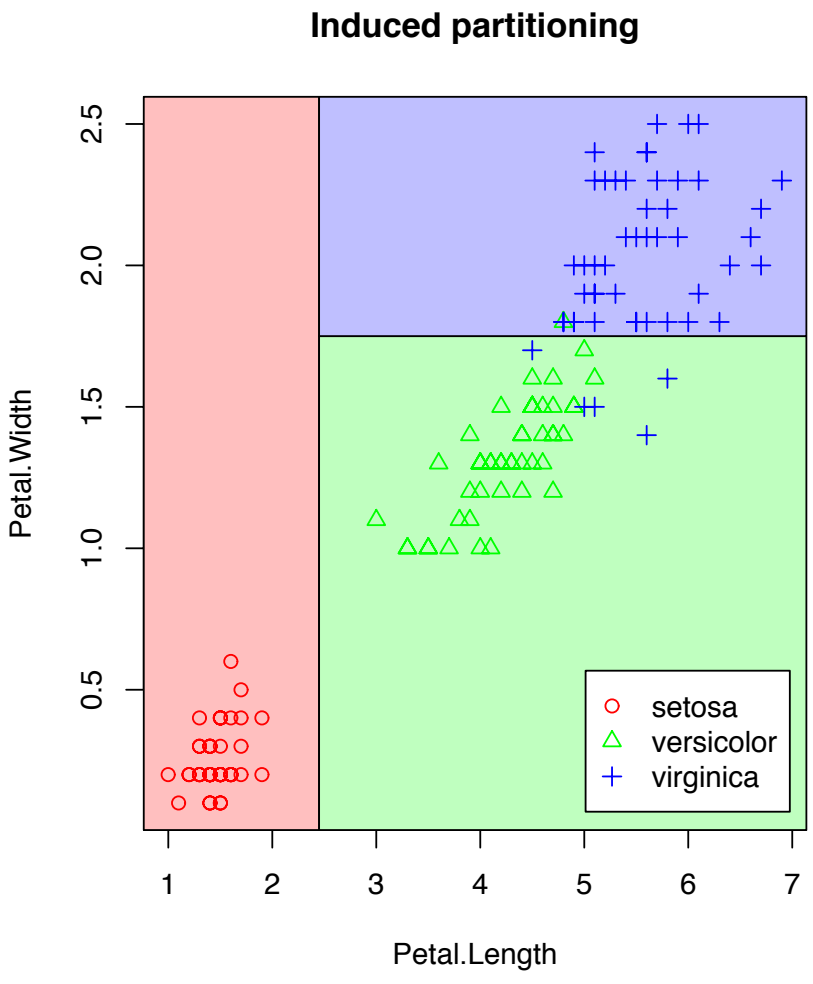
Example: Iris Data

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.4	3.2	1.3	0.2	setosa
5.9	3.0	5.1	1.8	virginica
6.3	3.3	6.0	2.5	virginica
5.3	3.7	1.5	0.2	setosa
5.5	2.5	4.0	1.3	versicolor
6.1	2.9	4.7	1.4	versicolor
6.1	3.0	4.9	1.8	virginica
5.7	2.8	4.5	1.3	versicolor
5.4	3.0	4.5	1.5	versicolor
4.8	3.4	1.6	0.2	setosa
4.6	3.1	1.5	0.2	setosa
4.9	3.1	1.5	0.2	setosa
6.4	2.9	4.3	1.3	versicolor
.....				

Previously seen Iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

Example: Iris Data





Partition of $\mathcal{X}$ into R disjoint sets $(\mathcal{R}_1, \ldots, \mathcal{R}_R)$ is determined by the **leaves of the tree**.

Decision functions based on trees

- For regression problems, the parameterized function is

$$f(x) = \sum_{j=1}^{R} \beta_j 1_{[x \in \mathcal{R}_j]},$$

  Using squared loss, optimal parameters are:

$$\hat{\beta}_j = \frac{\sum_i y_i 1_{[x_i \in \mathcal{R}_j]}}{\sum_i 1_{[x_i \in \mathcal{R}_j]}}$$

- For classification problems, the estimated probability of each class k in region $\mathcal{R}_j$ is simply:

$$\hat{\beta}_{jk} = \frac{\sum_i 1(y_i = k) 1_{[x_i \in \mathcal{R}_j]}}{\sum_i 1_{[x_i \in \mathcal{R}_j]}}$$

- These estimates can be regularized as well.

Partition Estimation

- Ideally, would like to find partition that achieves minimal risk: lowest mean-squared error for prediction or misclassification rate for classification.
- Number of potential partitions is too large to search exhaustively.
- 'Greedy' search heuristics for a good partition:
  - Start at root.
  - Determine best feature and value to split.
  - Recurse on children of node.
  - Stop at some point.

Growth Heuristic for Regression Trees

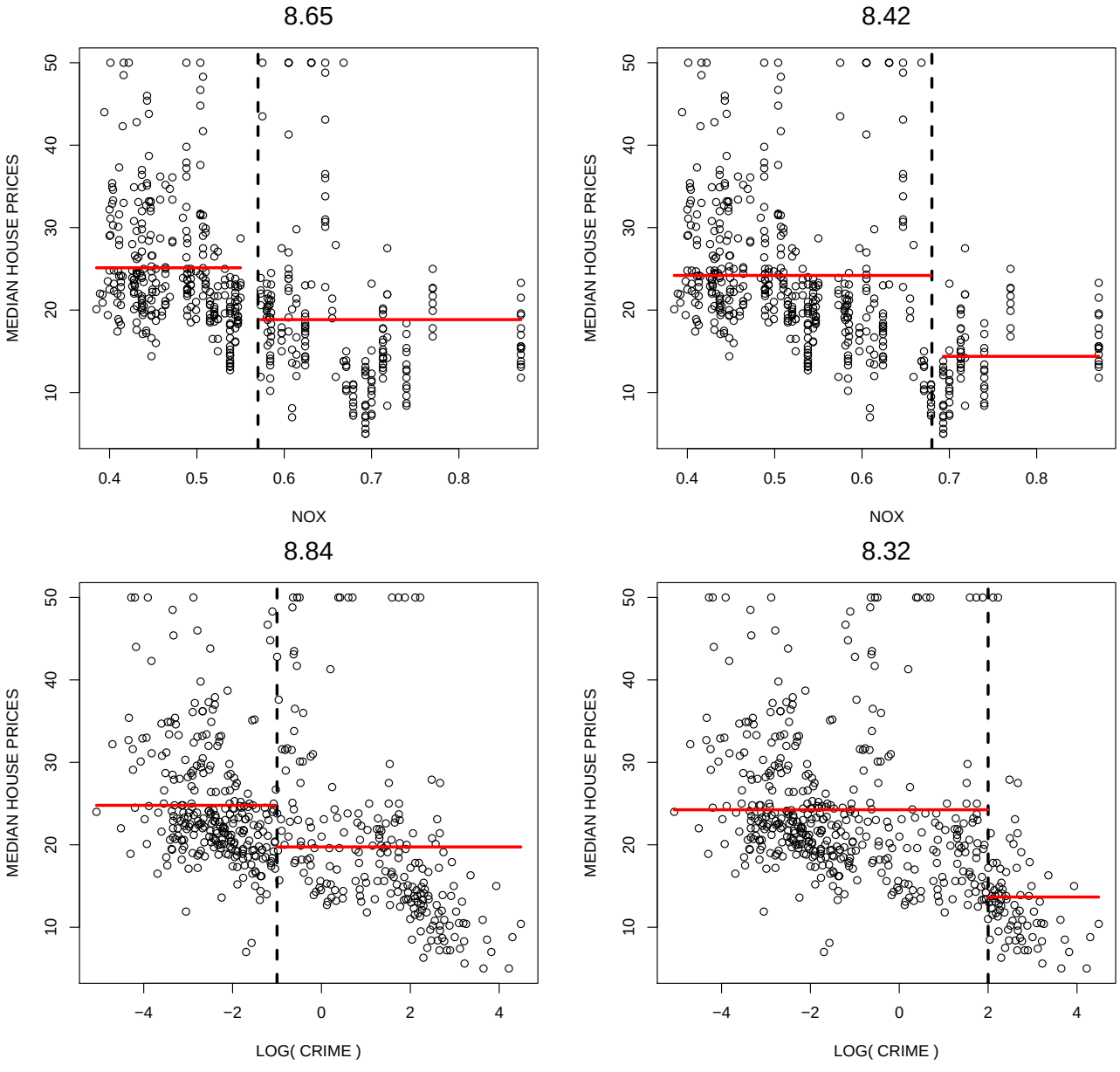
1. Start with $\mathcal{R}_1 = \mathcal{X} = \mathbb{R}^p$.
2. For each feature $j = 1, \ldots, p$, and for each value $v \in \mathbb{R}$ that we can split on:
   1. Split data set:
   $$I_< = \{i : x_i^{(j)} < v\} \qquad I_> = \{i : x_i^{(j)} \geq v\}$$
   2. Estimate parameters:
   $$\beta_< = \frac{\sum_{i \in I_<} y_i}{|I_<|} \qquad \beta_> = \frac{\sum_{i \in I_>} y_i}{|I_>|}$$
   3. Compute the **quality of split**, e.g., the square loss:
   $$\sum_{i \in I_<} (y_i - \beta_<)^2 + \sum_{i \in I_>} (y_i - \beta_>)^2$$
3. Choose split, i.e., feature j and value v, with minimal loss.
4. Recurse on both children, with datasets $(x_i, y_i)_{i \in I_<}$ and $(x_i, y_i)_{i \in I_>}$.

Boston Housing Data

```
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
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 USD 10,000
ptratio pupil-teacher ratio by town
b       1000(B - 0.63)^2 where B is the proportion of blacks by town
lstat   percentage of lower status of the population
medv    median value of owner-occupied homes in USD 1000's
```

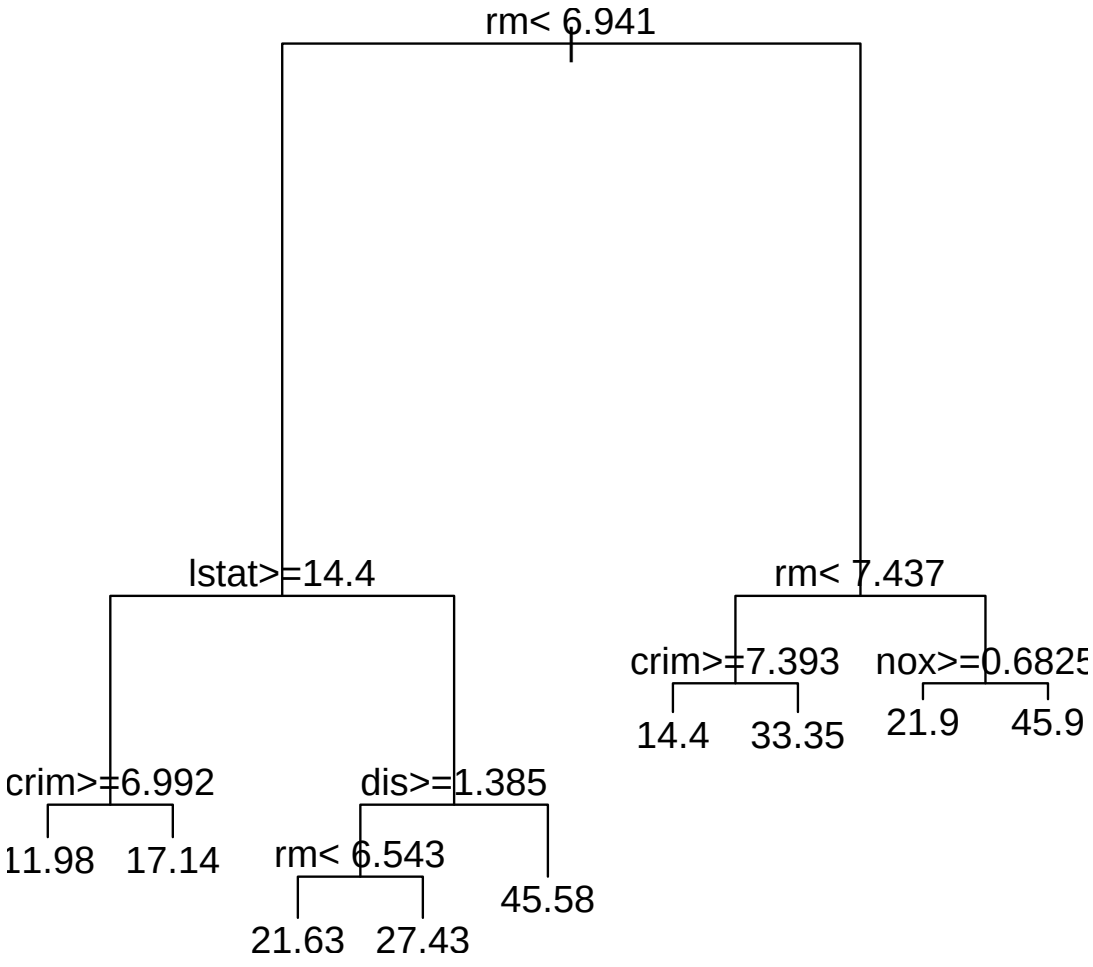
- Predict median house value.

Boston Housing Data



Boston Housing Data

- Overall, the best first split is on variable `rm`, average number of rooms per dwelling.
- Final tree contains predictions in leaf nodes.



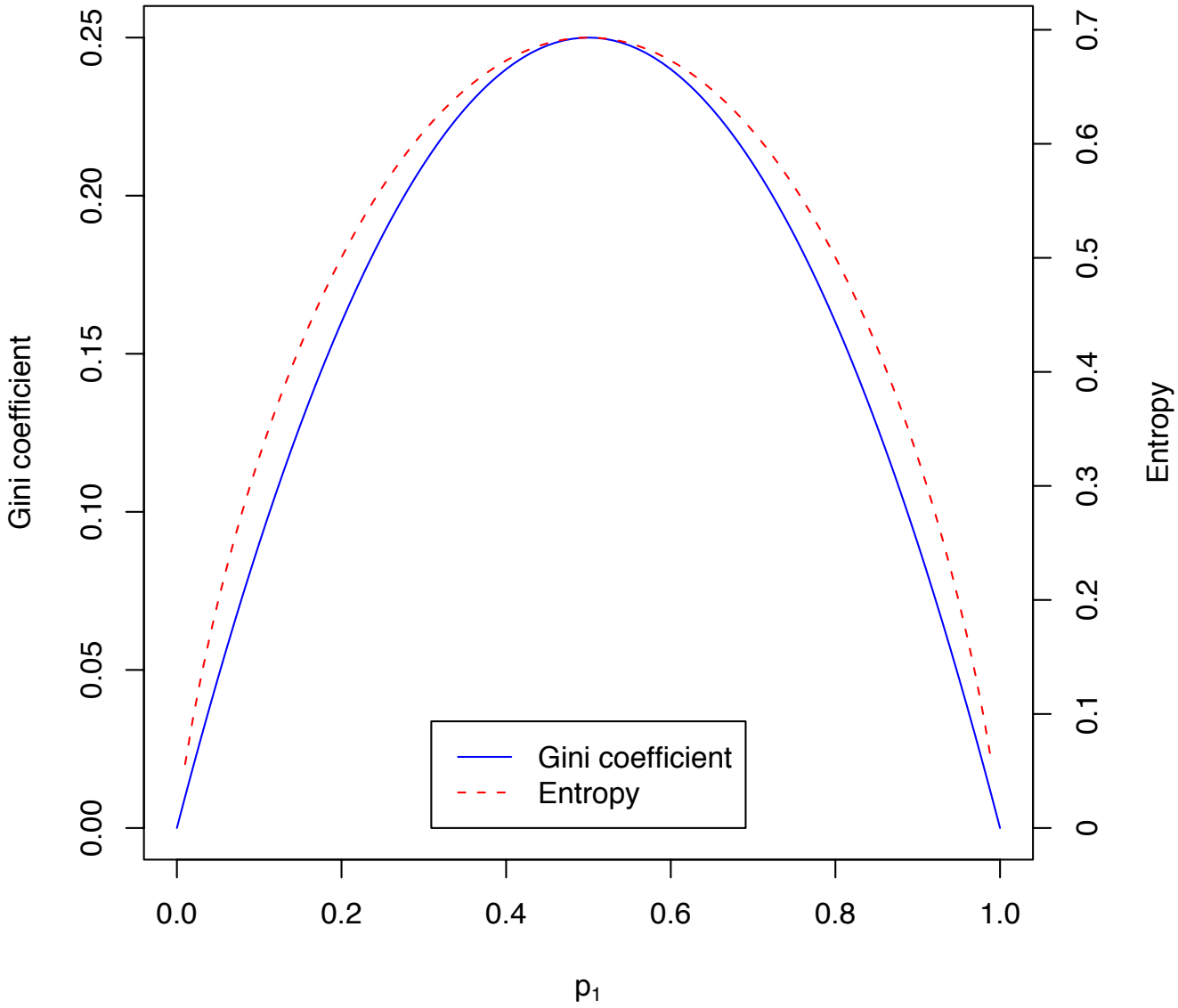
Growth Heuristics for Classification Trees

- For binary classification, the proportion of class 1 items in node corresponding to region $\mathcal{R}_j$ is given by

$$\hat{\beta}_{j1} = \frac{\sum_i 1(y_i = 1) 1_{[x_i \in \mathcal{R}_j]}}{\sum_i 1_{[x_i \in \mathcal{R}_j]}}$$

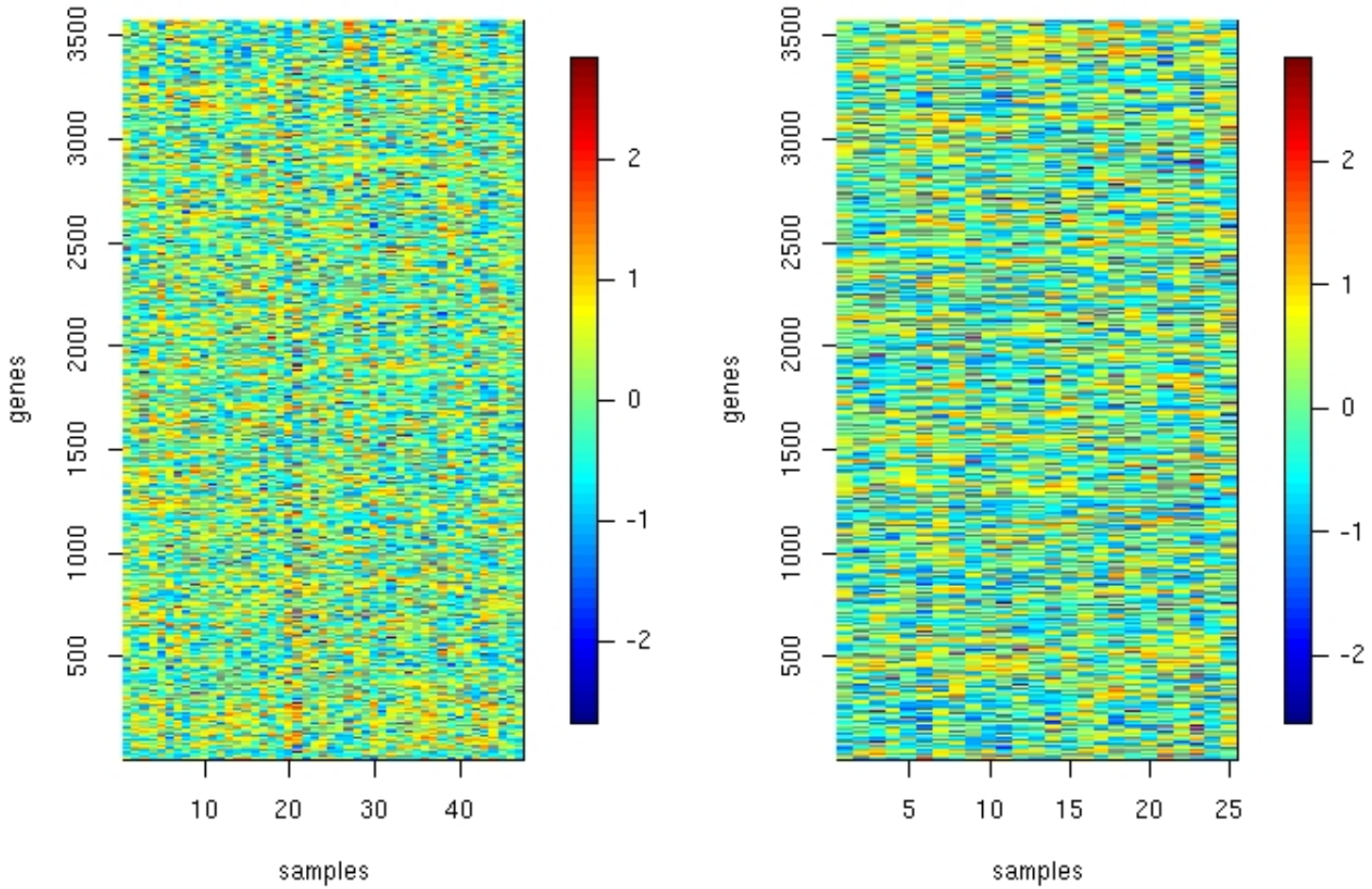
- A split is good if both sides are more **pure**, i.e. $\hat{\beta}_{j1}$ is closer to 0 or 1.
- Different measures of node impurity:
  - **Misclassification error**: $1 - \max\{\hat{\beta}_{j1}, 1 - \hat{\beta}_{j1}\}$.
  - **Gini impurity**: $2\hat{\beta}_{j1}(1 - \hat{\beta}_{j1})$.
  - **Entropy**: $-\hat{\beta}_{j1} \log \hat{\beta}_{j1} - (1 - \hat{\beta}_{j1}) \log(1 - \hat{\beta}_{j1})$.
- Gini and entropy preferred: differentiable and produce purer nodes.
- Extension to multi-class:
  - **Misclassification error**: $1 - \max_k \hat{\beta}_{jk}$.
  - **Gini impurity**: $\sum_{k=1}^{K} \hat{\beta}_{jk}(1 - \hat{\beta}_{jk})$.
  - **Entropy**: $-\sum_{k=1}^{K} \hat{\beta}_{jk} \log \hat{\beta}_{jk}$.
- Stops once a node has insufficient number of items, or is pure.

Growth Heuristics for Classification Trees



Misclassification error?

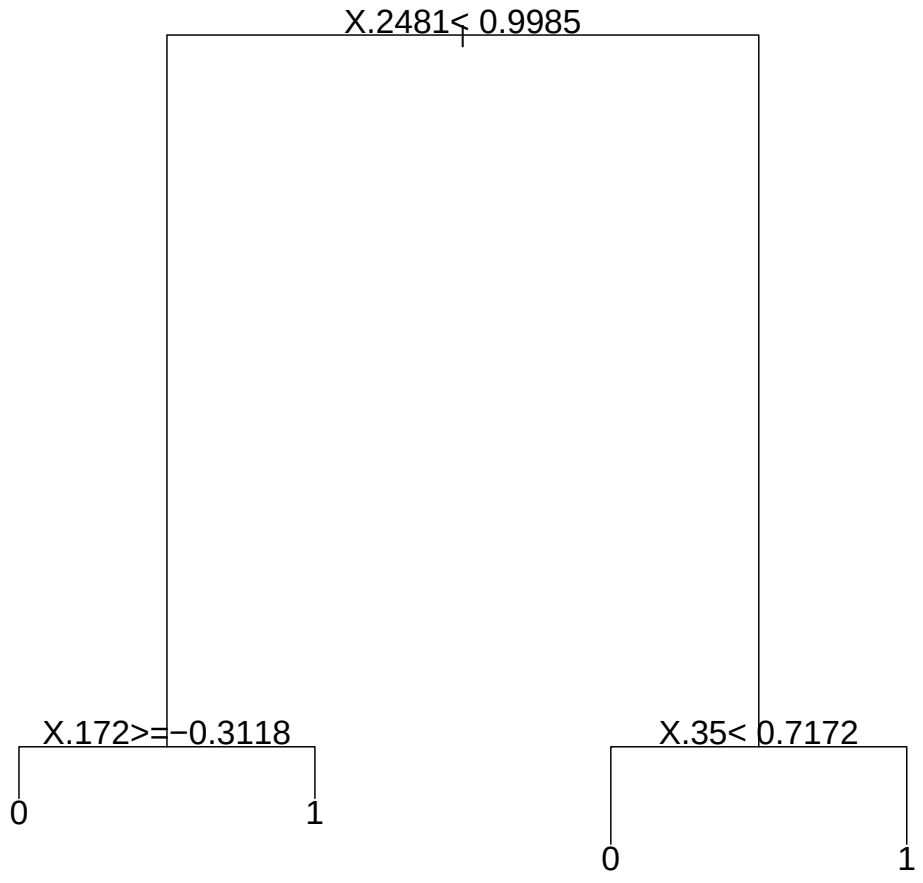
Example: Leukemia Prediction



Leukemia Dataset: Expression values of 3541 genes for 47 patients with Leukemia ALL subtype (left) and 25 patients with AML (right).

Example: Leukemia Prediction

- Tree found is of depth 2.
- Very interpretable as it selects 3 out of 4088 genes and bases prediction only on these.



Example: Pima Indians Diabetes Dataset

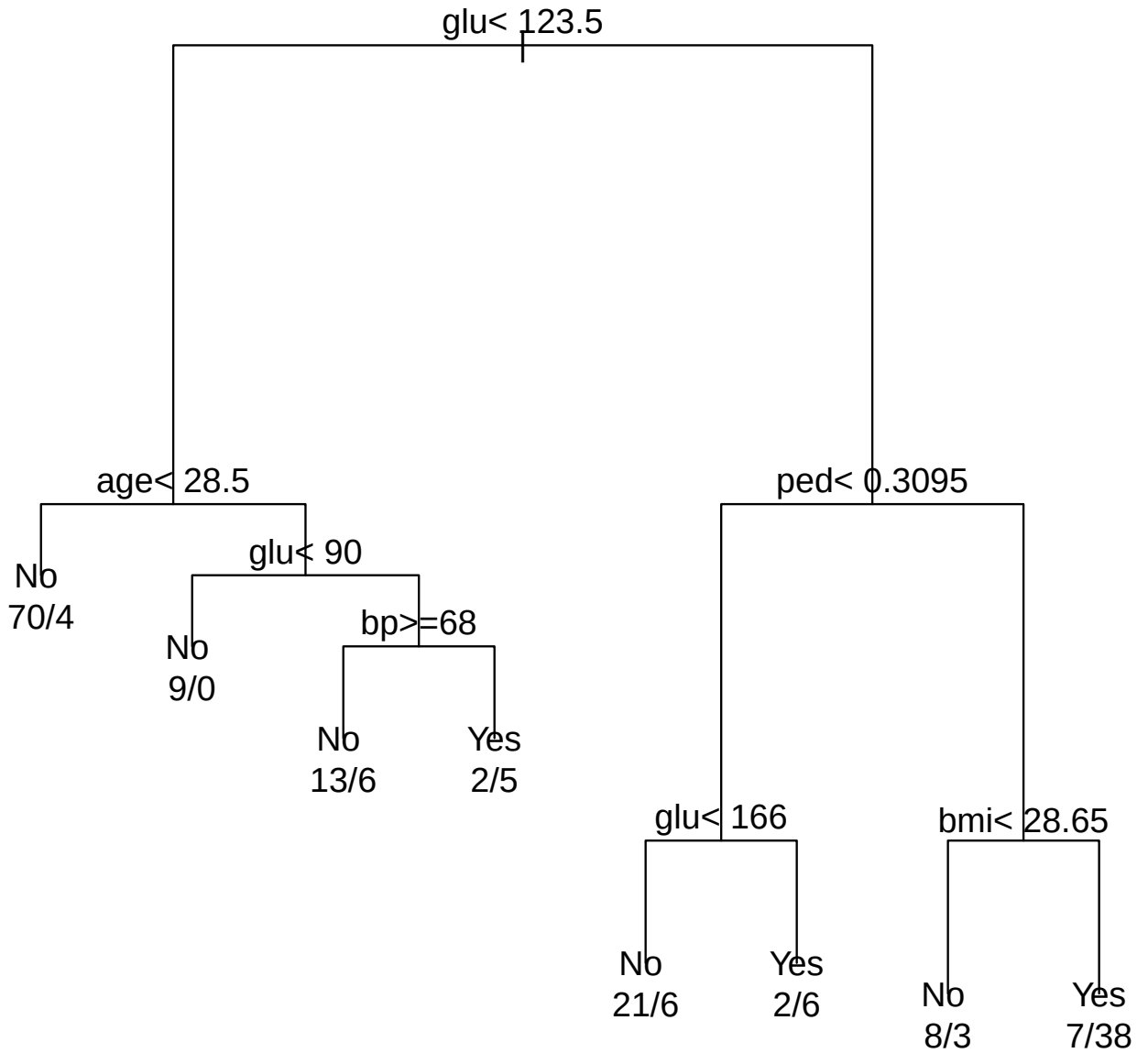
- The subjects: women who were at least 21 years old, of Pima Indian heritage living near Phoenix, Arizona.
- Tested for diabetes according to World Health Organisation criteria.
- Features:
  - number of pregnancies (npreg)
  - plasma glucose concentration (glu)
  - diastolic blood pressure (bp)
  - tricep skin fold thickness (skin)
  - body mass index(bbi)
  - diabetes pedigree function (ped)
  - age (age)

Example: Pima Indians Diabetes Dataset

```
> library(rpart)
> library(MASS)
> data(Pima.tr)
> rp <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8])
> rp
n= 200

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 200 68 No (0.66000000 0.34000000)
   2) glu< 123.5 109 15 No (0.86238532 0.13761468)
     4) age< 28.5 74  4 No (0.94594595 0.05405405) *
     5) age>=28.5 35 11 No (0.68571429 0.31428571)
      10) glu< 90 9  0 No (1.00000000 0.00000000) *
      11) glu>=90 26 11 No (0.57692308 0.42307692)
        22) bp>=68 19  6 No (0.68421053 0.31578947) *
        23) bp< 68 7  2 Yes (0.28571429 0.71428571) *
   3) glu>=123.5 91 38 Yes (0.41758242 0.58241758)
     6) ped< 0.3095 35 12 No (0.65714286 0.34285714)
      12) glu< 166 27  6 No (0.77777778 0.22222222) *
      13) glu>=166 8  2 Yes (0.25000000 0.75000000) *
     7) ped>=0.3095 56 15 Yes (0.26785714 0.73214286)
      14) bmi< 28.65 11  3 No (0.72727273 0.27272727) *
      15) bmi>=28.65 45  7 Yes (0.15555556 0.84444444) *
```

Example: Pima Indians Diabetes Dataset

```
> plot(rp,margin=0.1); text(rp,use.n=T)
```



Model Complexity

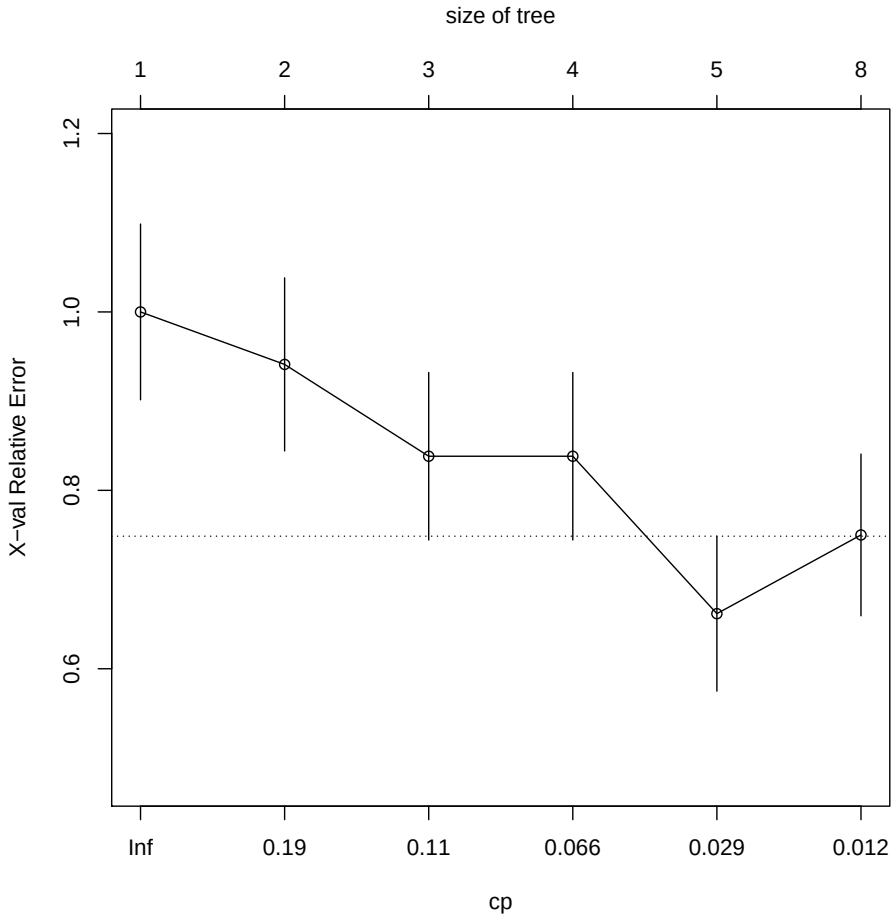
- When should tree growing be stopped?
- Will need to control complexity to prevent overfitting, and in general find optimal tree size with best predictive performance.
- A regularized objective

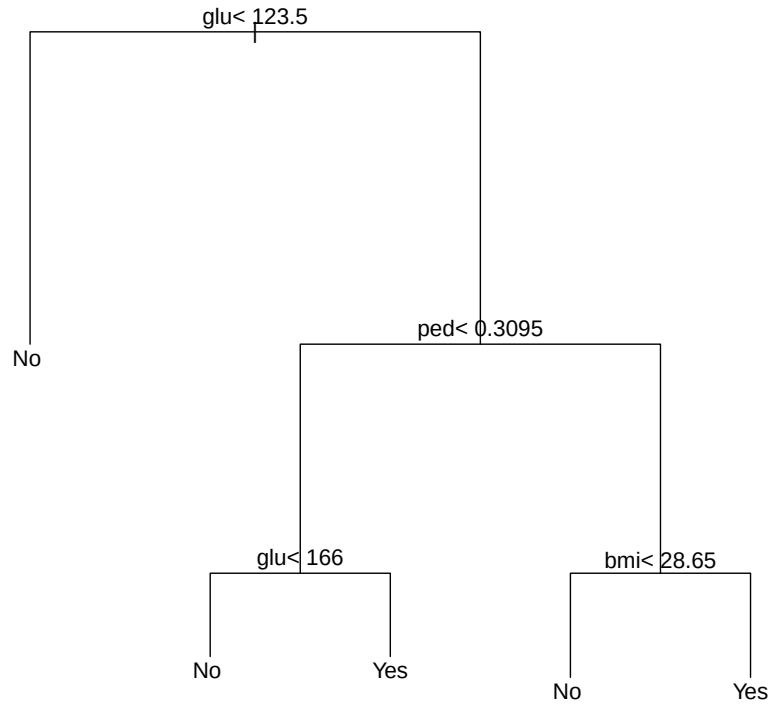
$$R^{\text{emp}}(T) + C \times \text{size}(T)$$

  - Grow the tree from scratch and stop once the criterion objective starts to increase.
  - First grow the full tree and prune nodes (starting at leaves), until the objective starts to increase.
- Second option is preferred as the choice of tree is less sensitive to "wrong" choices of split points and variables to split on in the first stages of tree fitting.
- Use cross-validation to determine optimal C.

Model Complexity

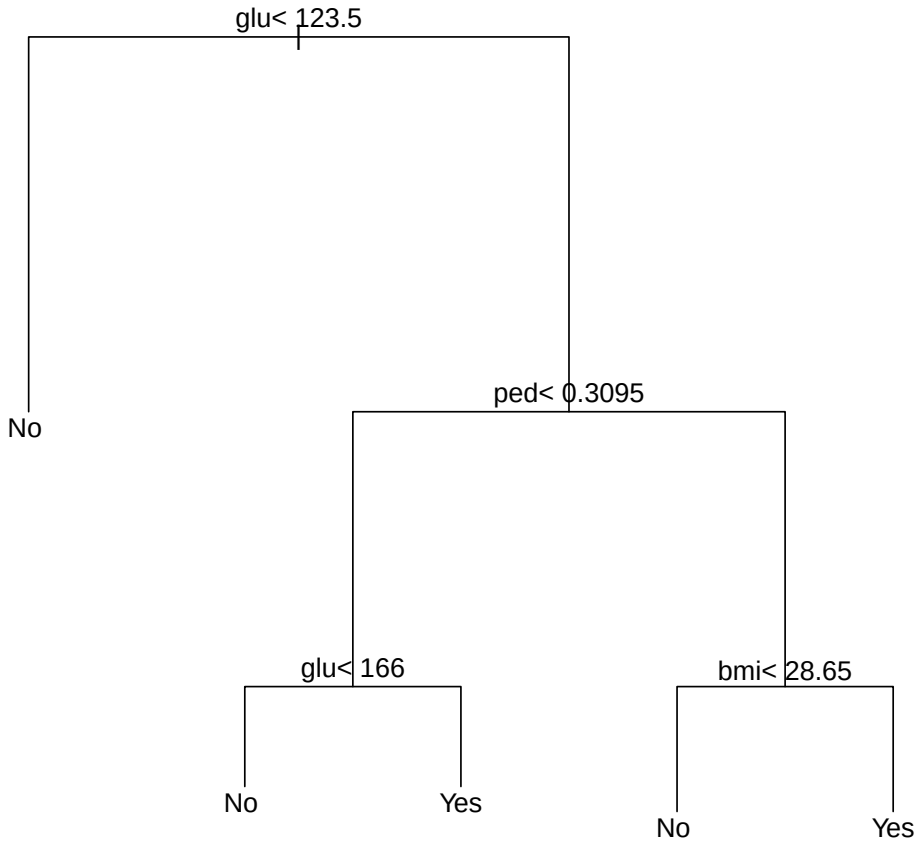
```
> rp <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8],
                    control=rpart.control(xval=10))) ## 10-fold CV
> plotcp(rp)
> rp2 <- prune.rpart(rp,.029)
> plot(rp2); text(rp2)
```





Bagging

Model Variability

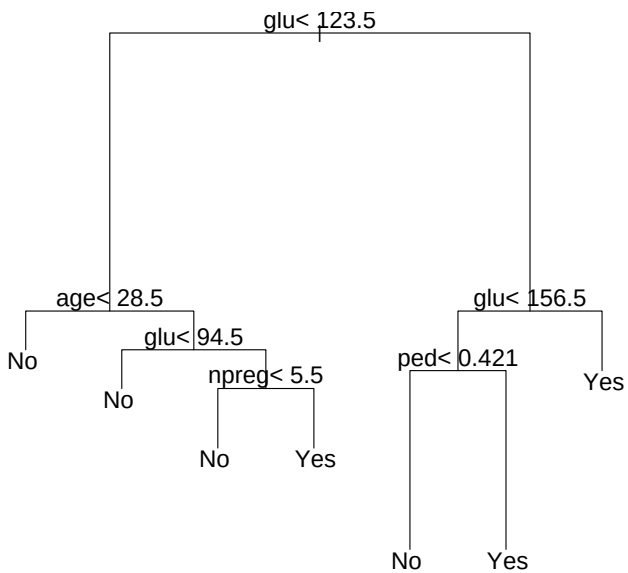


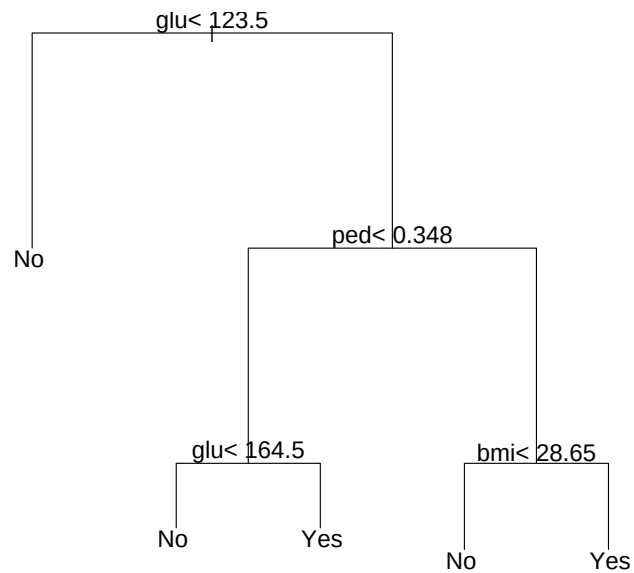
- Is the tree ‘stable’ if training data were slightly different?

Bootstrap for Classification Trees

- The **bootstrap** is a way to assess the variance of estimators.
- Fit multiple trees, each on a **bootstrapped sample**. This is a data set obtained by **sampling with replacement** n times from training set.

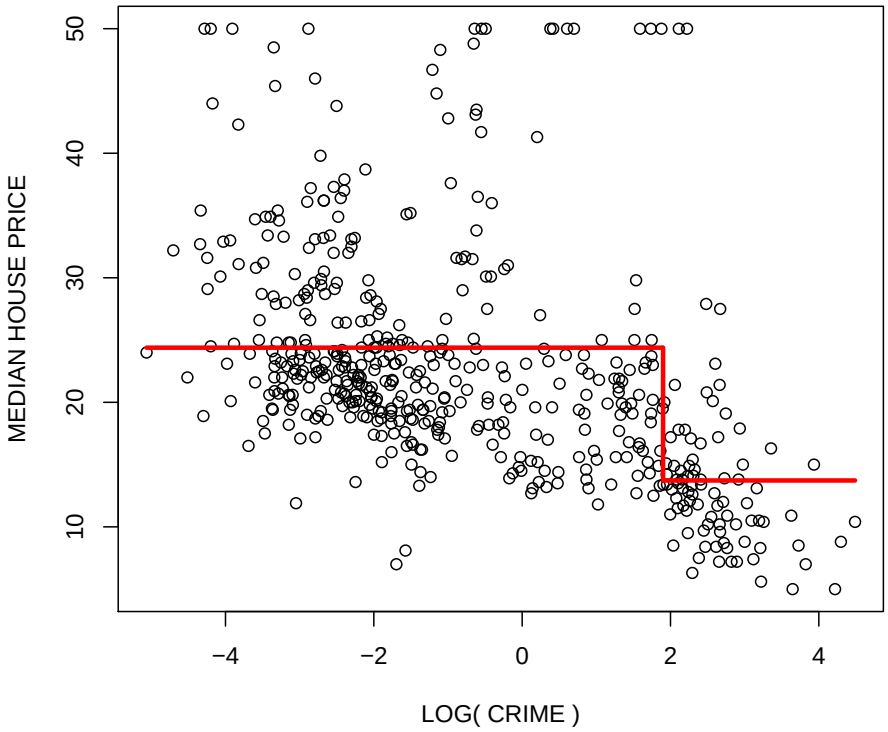
```
> n <- nrow(Pima.tr)
> bss <- sample(1:n, n , replace=TRUE)
> sort(bss)
[1]  2 4 4 5 6 7 9 10 11 12 12 12 12 13 13 15 15 20 ...

> tree_boot <- rpart(Pima.tr[bss,8] ~ ., data=Pima.tr[bss,-8],
                     control=rpart.control(xval=10)) ## 10-fold CV
```

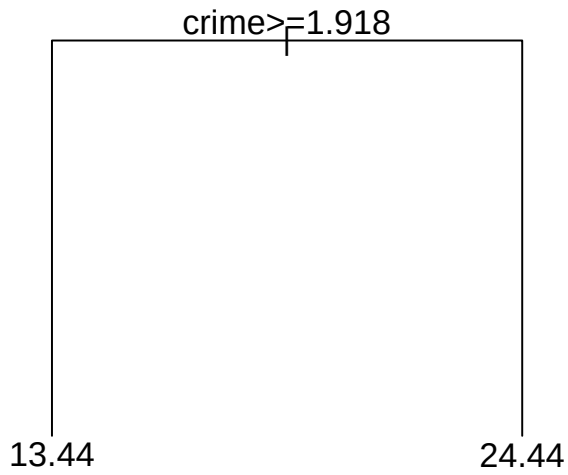




Bootstrap for Regression Trees

- Regression for Boston housing data.
- Predict median house prices based only on crime rate.
- Use decision **stump**—the simplest tree with a single split at root.



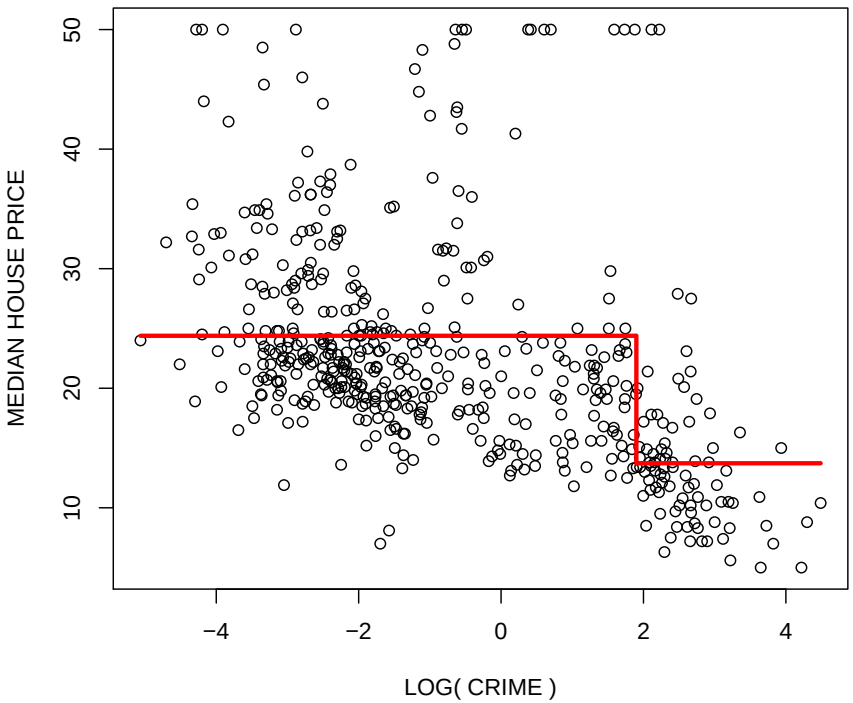


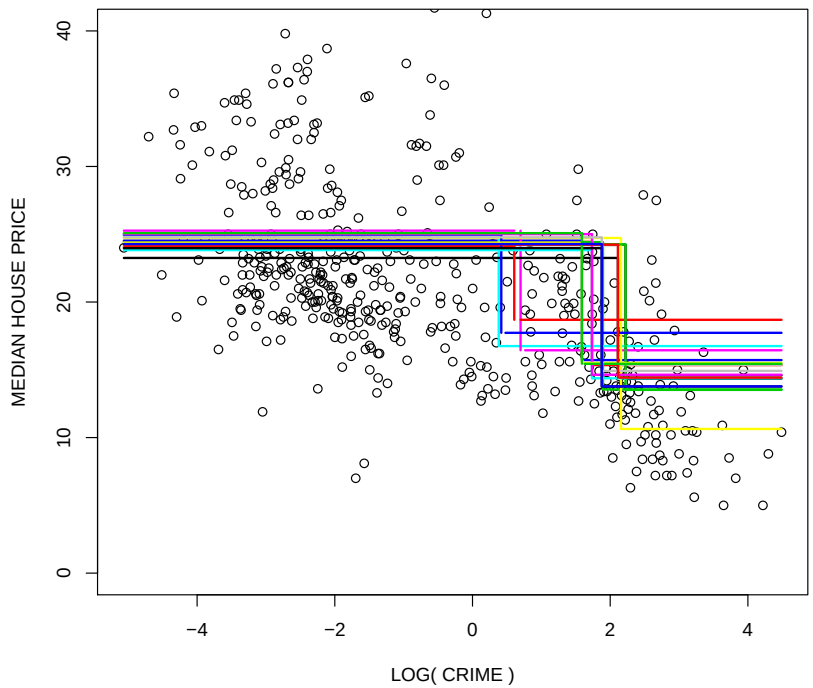
Bootstrap for Regression Trees

- We fit a predictor $\hat{f}(x)$ on the data $\{(x_i, y_i)\}_{i=1}^n$.
- Assess the variance of $\hat{f}(x)$ by taking $B = 20$ bootstrap samples of the original data, and obtaining bootstrap estimators

$$\hat{f}^b(x), \qquad b = 1, \ldots, B$$

- Each tree $\hat{f}^b$ is fitted on the resampled data $(x_{j_i}, y_{j_i})_{i=1}^n$ where each j_i is chosen randomly from $\{1, \ldots, n\}$ with replacement.





Bagging

- **Bagging** (**B**ootstrap **Agg**regation): average across all B trees fitted on different bootstrap samples.

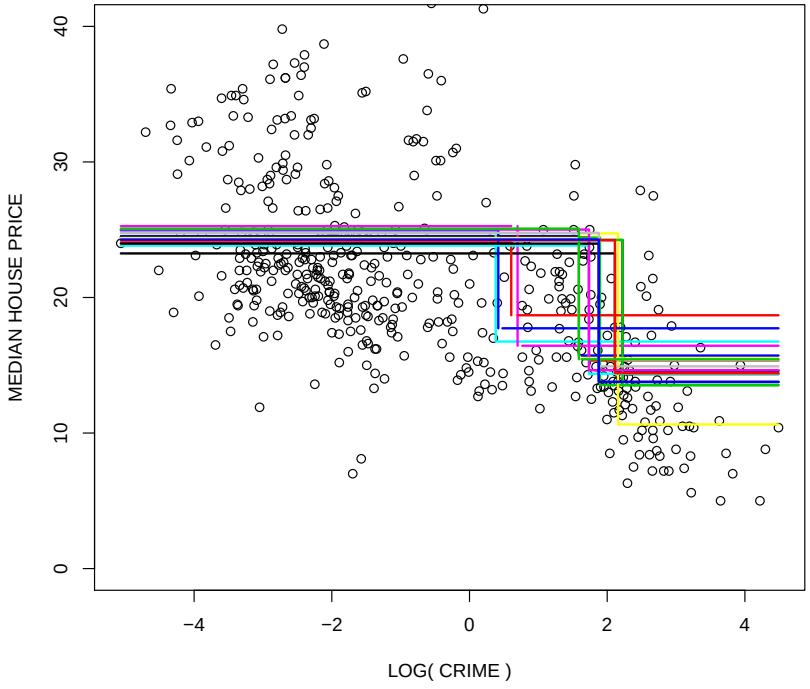
1. For $b = 1, \ldots, B$:
   1. Draw indices $(j_1, \ldots, j_n)$ from the set $\{1, \ldots, n\}$ with replacement.
   2. Fit the model, and form predictor $\hat{f}^b(x)$ based on bootstrap sample

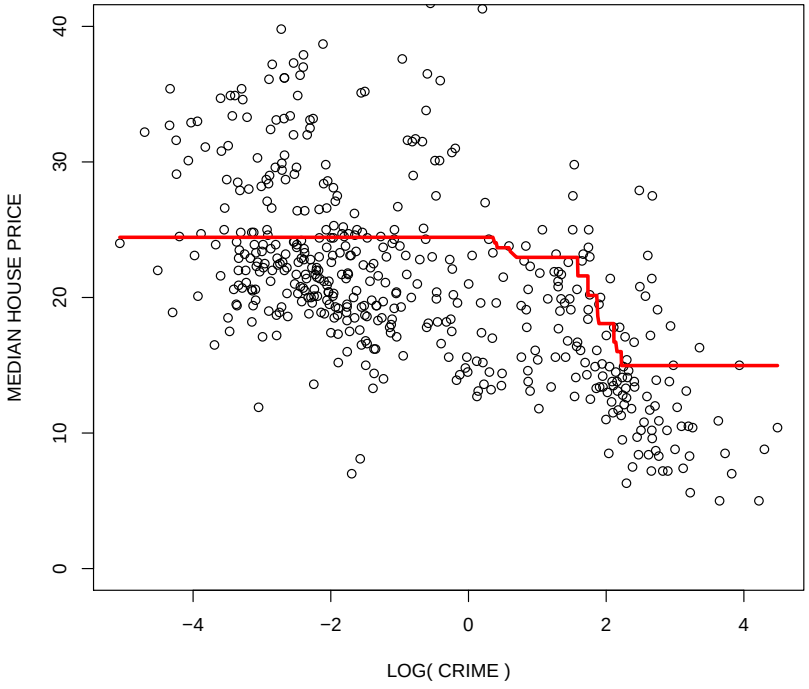
$$(x_{j_1}, y_{j_1}), \ldots, (x_{j_n}, y_{j_n})$$

2. Form bagged estimator

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

Bagging





- Bagging smooths out the drop in the estimate of median house prices.
- Bagging reduces the variance of predictions, i.e. when taking expectations over a random dataset $\mathcal{D}$:

$$\mathbb{E}_{\mathcal{D}}\left[(\hat{f}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}(x)])^2\right] \quad \geq \quad \mathbb{E}_{\mathcal{D}}\left[(\hat{f}_{Bag}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{Bag}(x)])^2\right]$$

Variance Reduction in Bagging

- Suppose, in an ideal world, our estimators $\hat{f}^b$ are each based on different independent datasets of size n from the true joint distribution of X, Y.
- The aggregated estimator would then be

$$\hat{f}_{ag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x) \to \bar{f}(x) = \mathbb{E}_{\mathcal{D}}[\hat{f}(x)] \quad \text{as } B \to \infty$$

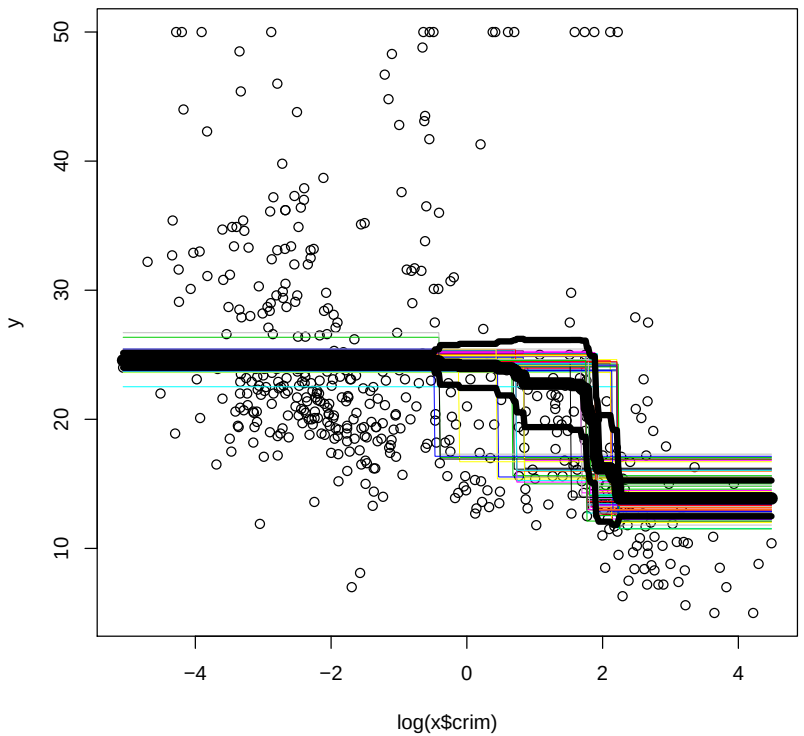
  where expectation is with respect to datasets of size n.
- The squared-loss is:

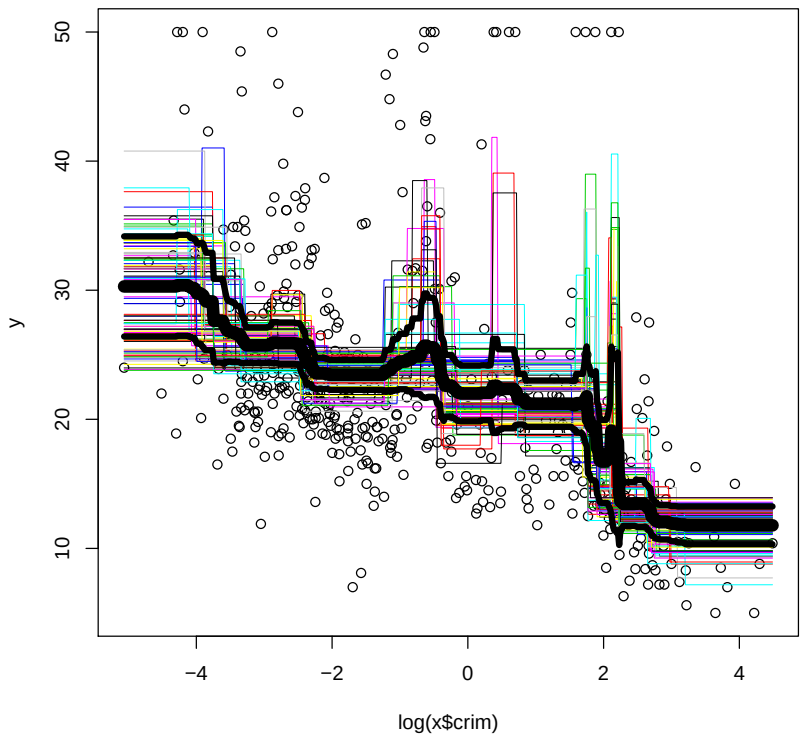
$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[(Y-\hat{f}_{ag}(X))^2|X=x] &= \mathbb{E}_{\mathcal{D}}[(Y-\bar{f}(X))^2|X=x] + \mathbb{E}_{\mathcal{D}}[(\bar{f}(X)-\hat{f}_{ag}(X))^2|X=x] \\ &\to \mathbb{E}_{\mathcal{D}}[(Y-\bar{f}(X))^2|X=x] \quad \text{as } B \to \infty.\end{aligned}$$

  Aggregation reduces the squared loss by eliminating variance of $\hat{f}(x)$.
- In bagging, variance reduction still applies at the cost of a small increase in bias.
- Bagging is most useful for **flexible estimators with high variance** (and low bias).

Variance Reduction in Bagging

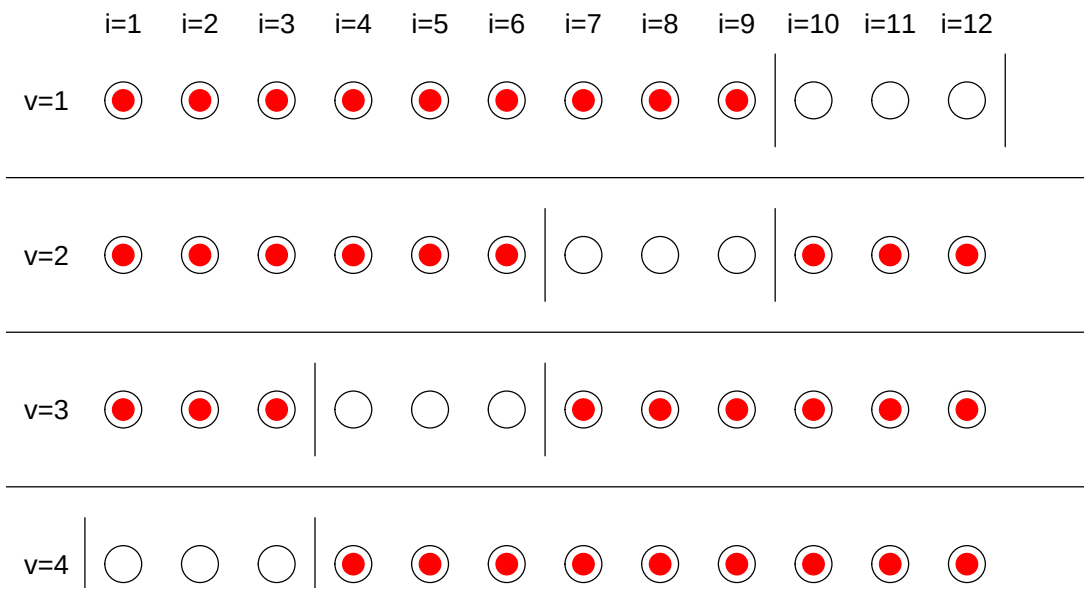
- Deeper trees have higher complexity and variance.
- Compare bagging trees of depth 1 and 3.





Out-of-bag Test Error Estimation

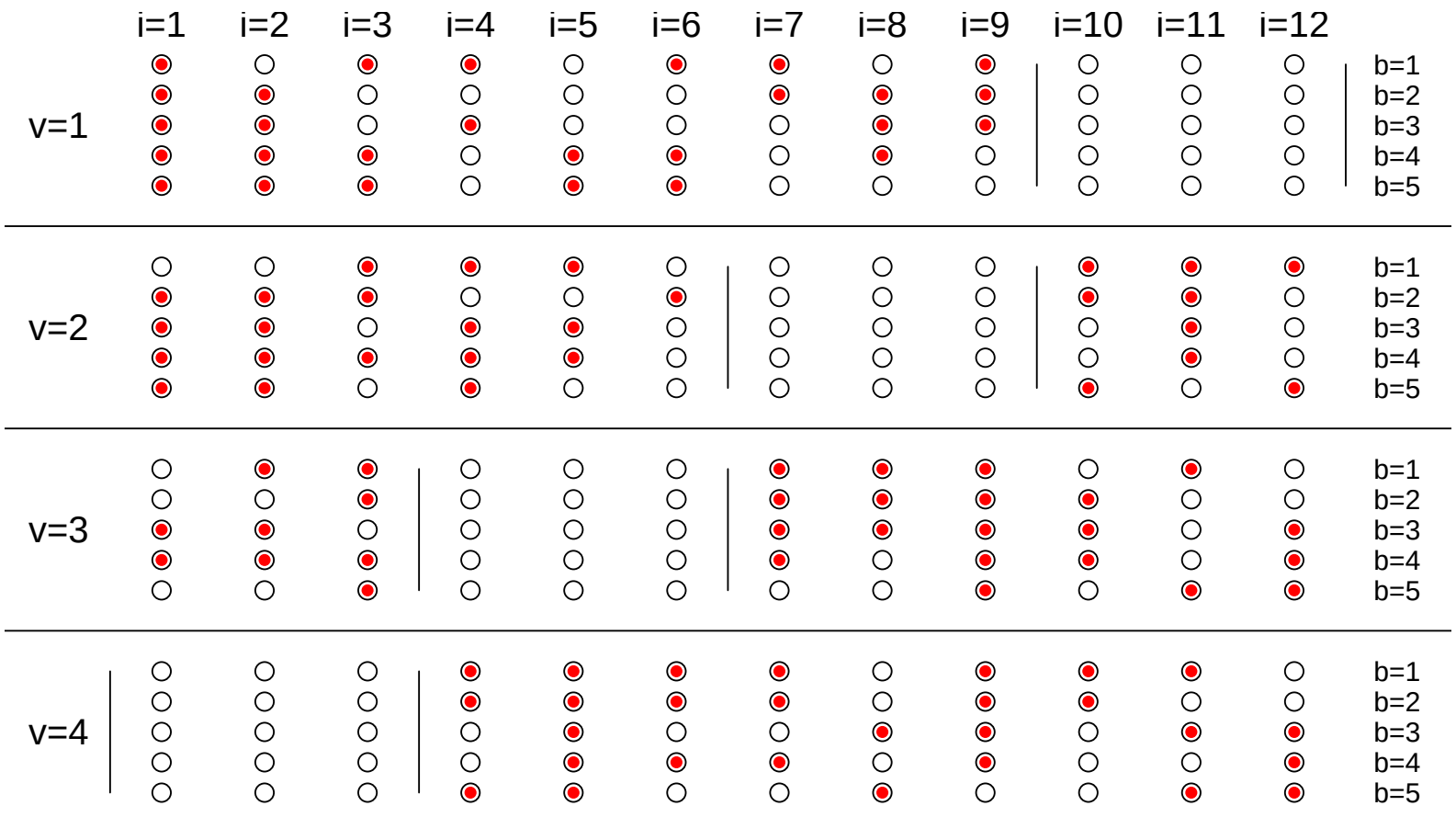
- How well does bagging to? Can we estimate generalization performance, and tune hyperparameters?
- Answer 1: cross-validation.



- For each $v = 1, \ldots, V$,
  - fit $\hat{f}_{Bag}$ on the training samples.
  - predict on validation set.
- Compute the CV error by averaging the loss across all test observations.

Out-of-bag Test Error Estimation

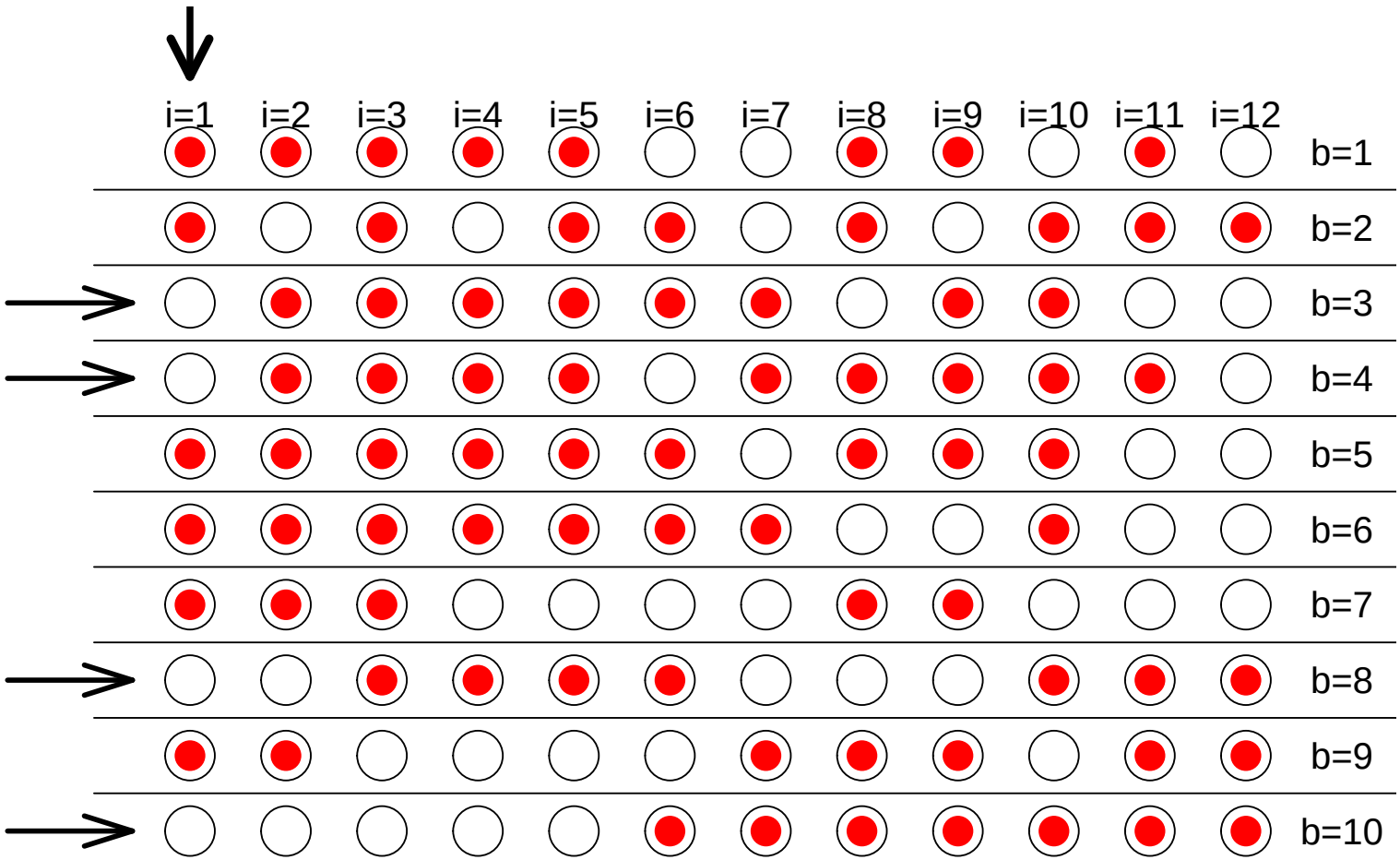
- But to fit $\hat{f}_{Bag}$ on the training set for each $v = 1, \ldots, V$, we have to train on B bootstrap samples!



- Answer 2: **Out-of-bag** test error estimation.

Out-of-bag Test Error Estimation

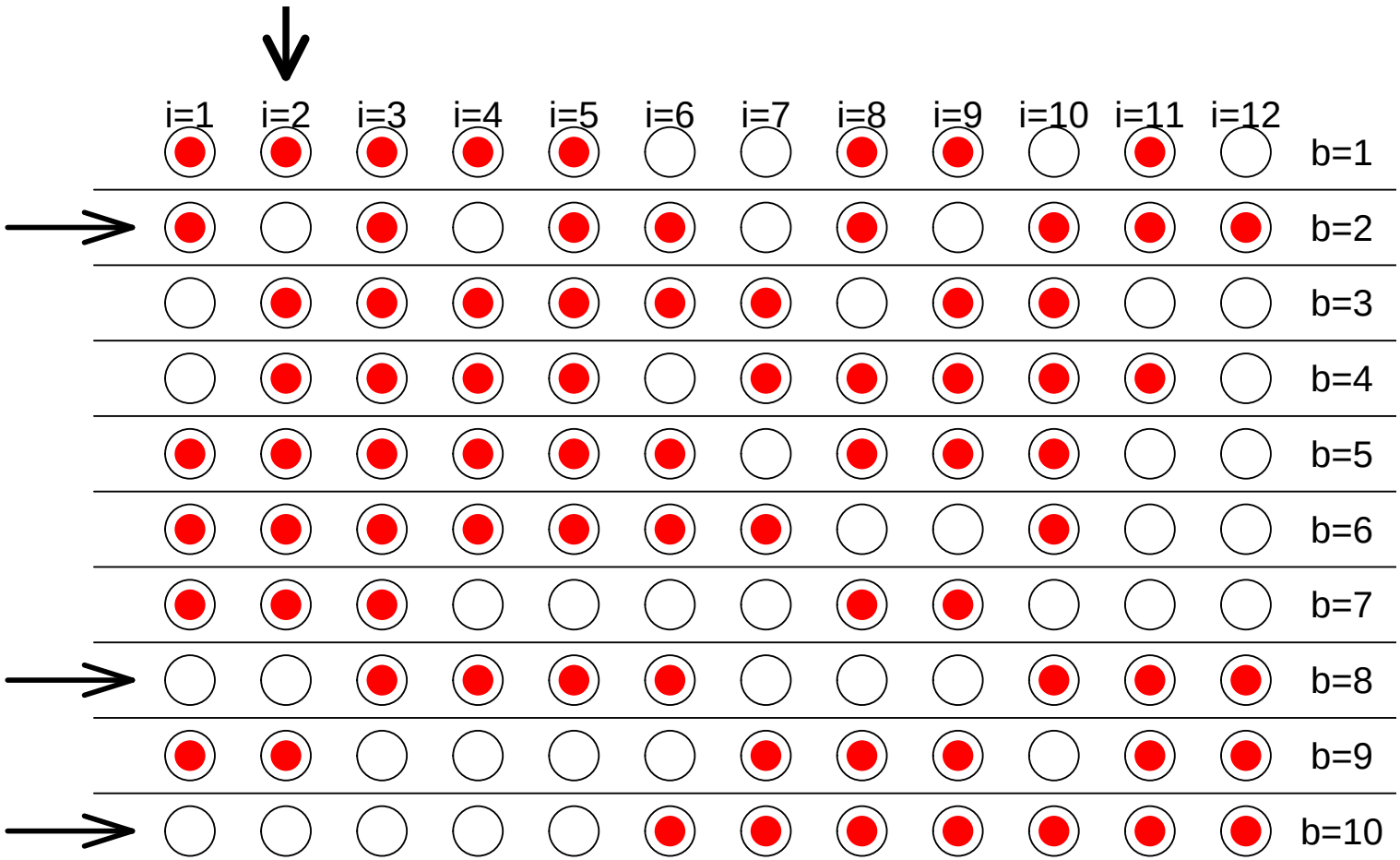
- Idea: test on the “unused” data points in each bootstrap iteration to estimate the test error.



$$\hat{f}^{\text{oob}}(x_1) = \frac{1}{4} \sum_{b \in \{3,4,8,10\}} \hat{f}^{b}(x_1)$$

Out-of-bag Test Error Estimation

- Idea: test on the "unused" data points in each bootstrap iteration to estimate the test error.



$$\hat{f}^{\text{oob}}(x_2) = \frac{1}{3} \sum_{b \in \{2,8,10\}} \hat{f}^b(x_2)$$

Out-of-bag Test Error Estimation

- For each $i = 1, \ldots, n$, the out-of-bag sample is:

$$\tilde{B}_i = \{b : x_i \text{ is not in training set}\} \subseteq \{1, \ldots, B\}.$$

- Construct the out-of-bag estimate at x_i:

$$\hat{f}^{\text{oob}}(x_i) = \frac{1}{|\tilde{B}_i|} \sum_{b \in \tilde{B}_i} \hat{f}^b(i_i)$$

- Out-of-bag risk:

$$R^{\text{oob}} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}^{\text{oob}}(x_i))$$

Out-of-bag Test Error Estimation

- We need $|\tilde{B}_i|$ to be reasonably large for all $i = 1, \ldots, n$.
- The probability π^{oob} of an observation NOT being included in a bootstrap sample $(j_1, \ldots, j_n)$ (and hence being 'out-of-bag') is:

$$\pi^{\text{oob}} = \prod_{i=1}^{n} \left(1 - \frac{1}{n}\right) \quad \stackrel{n \to \infty}{\longrightarrow} \quad \frac{1}{e} \approx 0.367.$$

- Hence $\mathbb{E}[|\tilde{B}_i|] \approx 0.367B$
- In practice, number of bootstrap samples B is typically between 200 and 1000, meaning that the number $|\tilde{B}_i|$ of out-of-bag samples will be approximately in the range $70 - 350$.
- The obtained test error estimate is asymptotically unbiased for large number B of bootstrap samples and large sample size n.

Example: Boston Housing Dataset

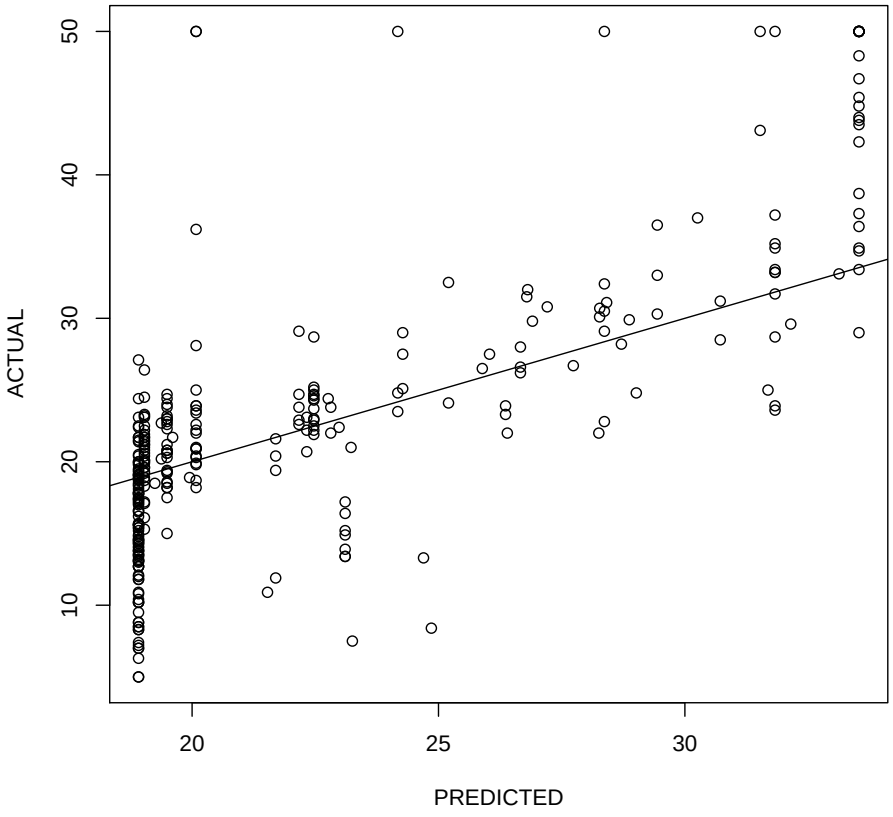
- Apply out of bag test error estimation to select optimal tree depth and assess performance of bagged trees for Boston Housing data.
- Use the entire dataset with $p = 13$ predictor variables.

```
n <- nrow(BostonHousing)   ## n samples
X <- BostonHousing[,-14]
Y <- BostonHousing[,14]
B <- 100
maxdepth <- 3
prediction_oob <- rep(0,length(Y))      ## vector with oob predictions
numbertrees_oob <- rep(0,length(Y))     ## number pf oob trees
for (b in 1:B) {                        ## loop over bootstrap samples
  subsample <- sample(1:n,n,replace=TRUE)       ## "in-bag" samples
  outofbag <- (1:n)[-subsample]                 ## "out-of-bag" samples
                                        ## fit tree on "in-bag" samples
  treeboot <- rpart(Y ~ ., data=X, subset=subsample,
        control=rpart.control(maxdepth=maxdepth,minsplit=2))
                                        ## predict on oob-samples
  prediction_oob[outofbag] <- prediction_oob[outofbag] +
                      predict(treeboot, newdata=X[outofbag,])
  numbertrees_oob[outofbag] <- numbertrees_oob[outofbag] +  1
}
## final oob-prediction is average across all "out-of-bag" trees
prediction_oob <- prediction_oob / numbertrees_oob
```

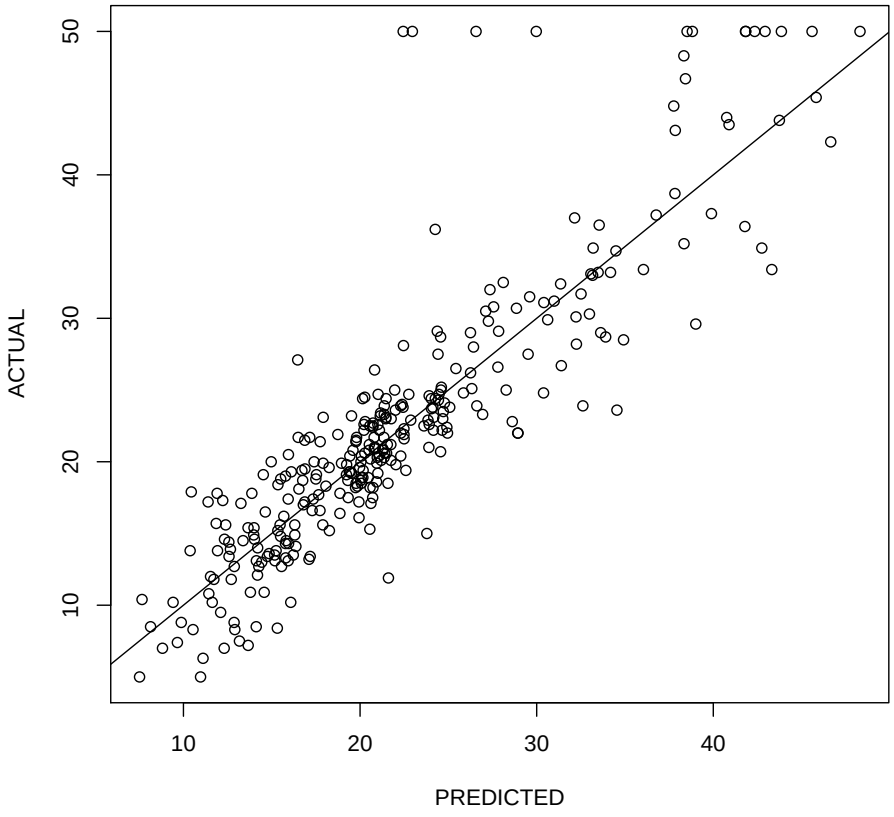
Example: Boston Housing Dataset

```
plot(prediction_oob, Y,  xlab="PREDICTED",  ylab="ACTUAL")
```

For depth $d = 1$.



For depth $d = 10$.



Example: Boston Housing Dataset

- Out-of-bag error as a function of tree depth d:

tree depth d	1	2	3	4	5	10	30
single tree $\hat{f}$	60.7	44.8	32.8	31.2	27.7	26.5	27.3
bagged trees $\hat{f}_{Bag}$	43.4	27.0	22.8	21.5	20.7	20.1	20.1

- Without bagging, the optimal tree depth seems to be $d = 10$.
- With bagging, we could also take the depth up to $d = 30$.

Summary:

- Bagging reduces variance and prevents overfitting
- Often improves accuracy in practice.
- Bagged trees cannot be displayed as nicely as single trees and some of the interpretability of trees is lost.