Outline

Model complexity

Which size of the tree is optimal?
Can grow tree until every leaf node contains only 1 original observation.
Clearly one should stop before. But where?
Example: Pima Indians Dataset.
The subjects were women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona. They were tested for diabetes according to World Health Organisation criteria.
The variables measured were the number of pregnancies (npreg), the plasma glucose concentration in an oral glucose tolerance test (glu), the diastolic blood pressure in mmHg (bp), the triceps skin fold thickness in mm(skin), the body mass index(bbi), the diabetes pedigree function (ped), and the age (age).

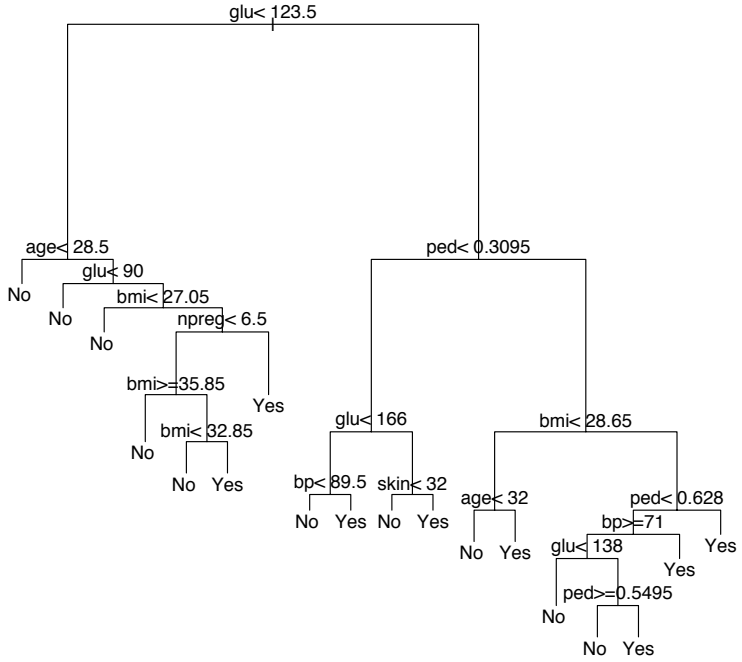
```
> library(rpart)
> library(MASS)
> data(Pima.tr)
> str(Pima.tr)

> > Pima.tr
    npreg glu  bp skin  bmi   ped age type
1       5  86  68   28 30.2 0.364  24   No
2       7 195  70   33 25.1 0.163  55  Yes
3       5  77  82   41 35.8 0.156  35   No
4       0 165  76   43 47.9 0.259  26   No
5       0 107  60   25 26.4 0.133  23   No
6       5  97  76   27 35.6 0.378  52  Yes
7       3  83  58   31 34.3 0.336  25   No
8       1 193  50   16 25.9 0.655  24   No
9       3 142  80   15 32.4 0.200  63   No
10      2 128  78   37 43.3 1.224  31  Yes
11      0 137  40   35 43.1 2.288  33  Yes
12      9 154  78   30 30.9 0.164  45   No
13      1 189  60   23 30.1 0.398  59  Yes
...
```

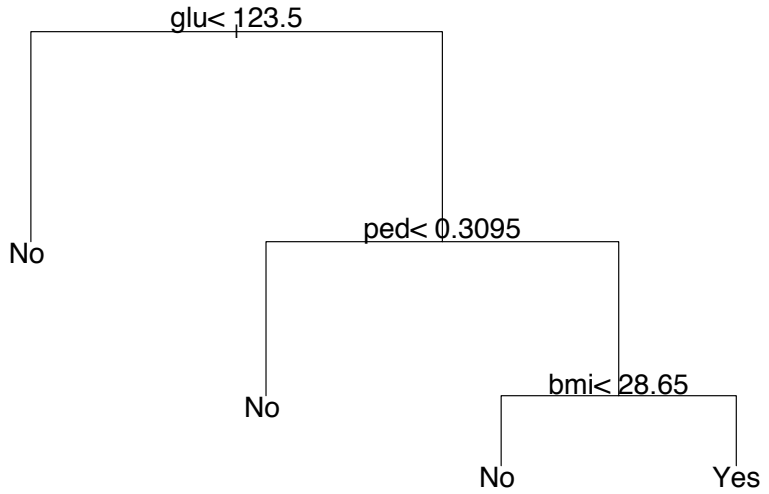
```
> rp <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8])
> rp
n= 200

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) *
```

Two possible trees.

```
> rp1 <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8])
> plot(rp1);text(rp1)
```

```
> rp2 <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8],
                        control=rpart.control(cp=0.05))
> plot(rp2);text(rp2)
```





Model Complexity

What influence has the size of the tree on predictive performance?

- The larger the tree is (the more final leaf nodes), the better is the prediction on the **training samples**.
- However, performance on **new data** / **test data** is deteriorating –in general– after a certain complexity (size) of the tree is surpassed.

Want to find the optimal complexity / tree size, giving best predictive performance for new (unseen) data.

Training and test error rate

Let $L(Y, \hat{Y})$ be a loss function that measures the loss when observing Y under a predition $\hat{Y}$.

- For regression trees,

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2.$$

- For classification trees

$$L(Y, \hat{Y}) = 1\{Y \neq \hat{Y}\}.$$

There are two important error rates, when using observations $(X_1, Y_1), \ldots, (X_n, Y_n)$ and a predictor $\hat{Y} = \hat{Y}(x)$. The fitted values at the n observations are $\hat{Y}_i := \hat{Y}(X_i)$.

- **Training error** rate R (or apparent error rate) is the loss for the training sample,

$$R_{train} = n^{-1} \sum_{i=1}^{n} L(Y_i, \hat{Y}_i).$$

- True error is the expected error rate/risk for new data (X, Y)

$$R_{test} = E\big(L(Y, \hat{Y})\big),$$

  where the expectation is with respect to drawing new random pairs (X, Y) and using the predictor $\hat{Y} = \hat{Y}(X)$ at the newly observed X.

Cross-Validation

Suppose we had

- **training data** (X_i, Y_i), $i = 1, \ldots, n$
- and a separate set of **test data** $(\tilde{X}_j, \tilde{Y}_j)$, $j = 1, \ldots, n_{test}$.

One possibility of estimating the true error rate is to

- **fit** the predictor $\hat{Y}$ (a tree here) on the **training data** and then
- **evaluate** the error rate on the **test data** (which have not been used for fitting of the tree),

$$\hat{R}_{test} = n_{test}^{-1} \sum_{j=1}^{n_{test}} L(\tilde{Y}_j, \hat{Y}(\tilde{X}_j)).$$

Disadvantage: if we have n_{test} additional samples, we could have used test data to get a larger training set and thus a better predictor.

Leave-one out cross-validation (LOO-CV)

For all $i = 1, \ldots, n$:

- fit the tree $T^{(-i)}$ by using all n observations **except** the i-th observation.
- compute prediction $\hat{Y}^{(-i)}(X_i)$ by running X_i down this tree.

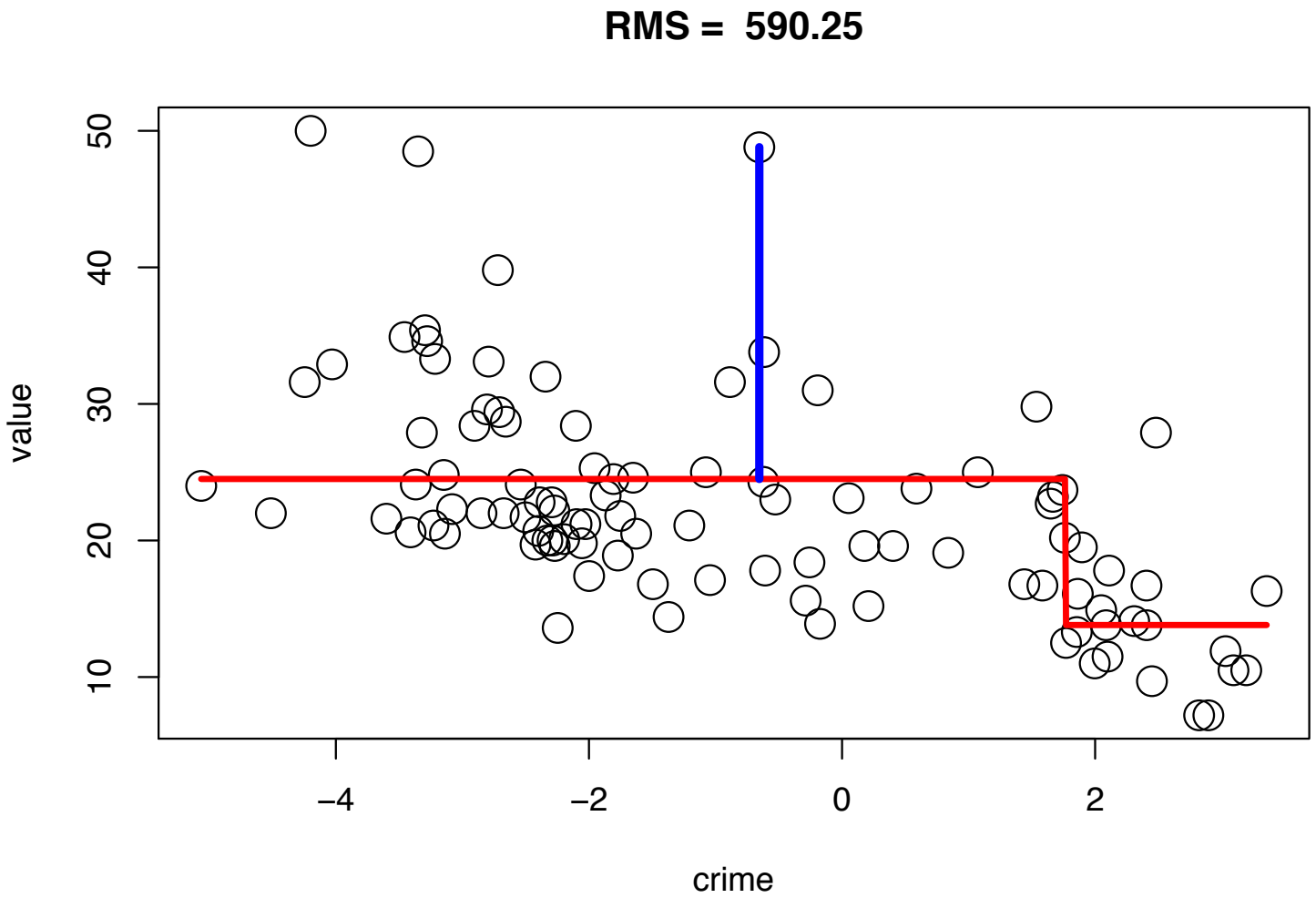
Compute the LOO-CV estimate of generalization error as

$$\hat{R}_{test} = n^{-1} \sum_{i=1}^{n} (\hat{Y}^{(-i)}(X_i) - Y_i)^2$$

for regression and mis-classification error or entropy criterion for classification. LOO-CV is a nearly unbiased estimate of generalization error. It can be expensive to compute as the tree (or other predictor) needs to be re-computed n times.

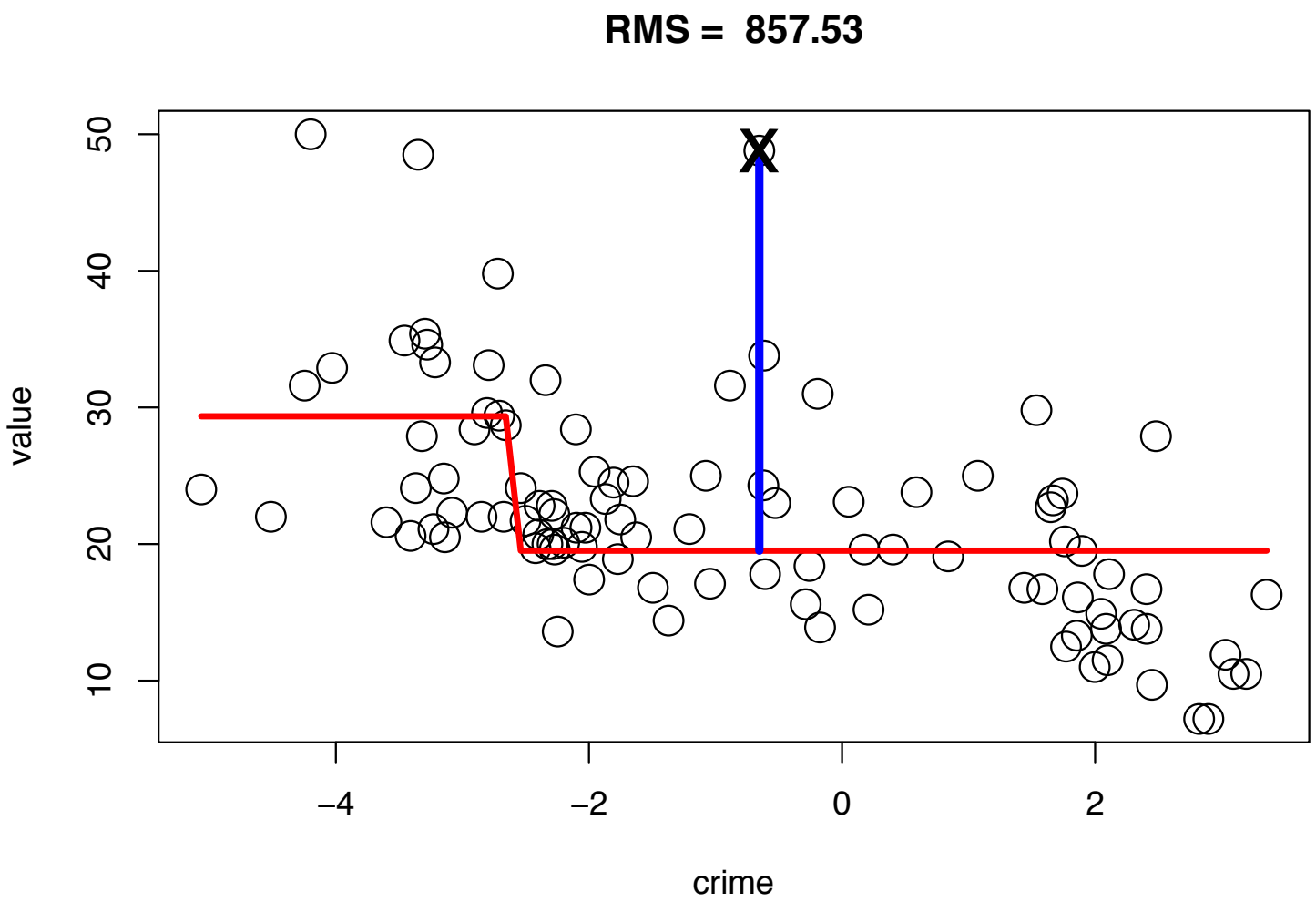
Example: Boston Housing Data

Again try to predict median house prices by using for simplicity just a single predictor variable, (logarithm of) crime rate.



Red line is fitted curve $\hat{Y}(x)$ for a tree of depth 1 (a stump). Blue vertical bar corresponds to residual of $i = 54$th observation with a squared residual of 590. Observation i was used to fit $\hat{Y}$ here !

Do the same fit but leave-out observation $i = 54$.



Red line is fitted curve $\hat{Y}^{(-54)}(x)$. Blue vertical bar corresponds to LOO-CV residual of $i = 54$th observation with a squared residual of 590. Observation $i = 54$ was now NOT used to fit $\hat{Y}^{(-54)}$ here!
Repeat for all $i = 1, \ldots, n$.

V-fold cross-validation

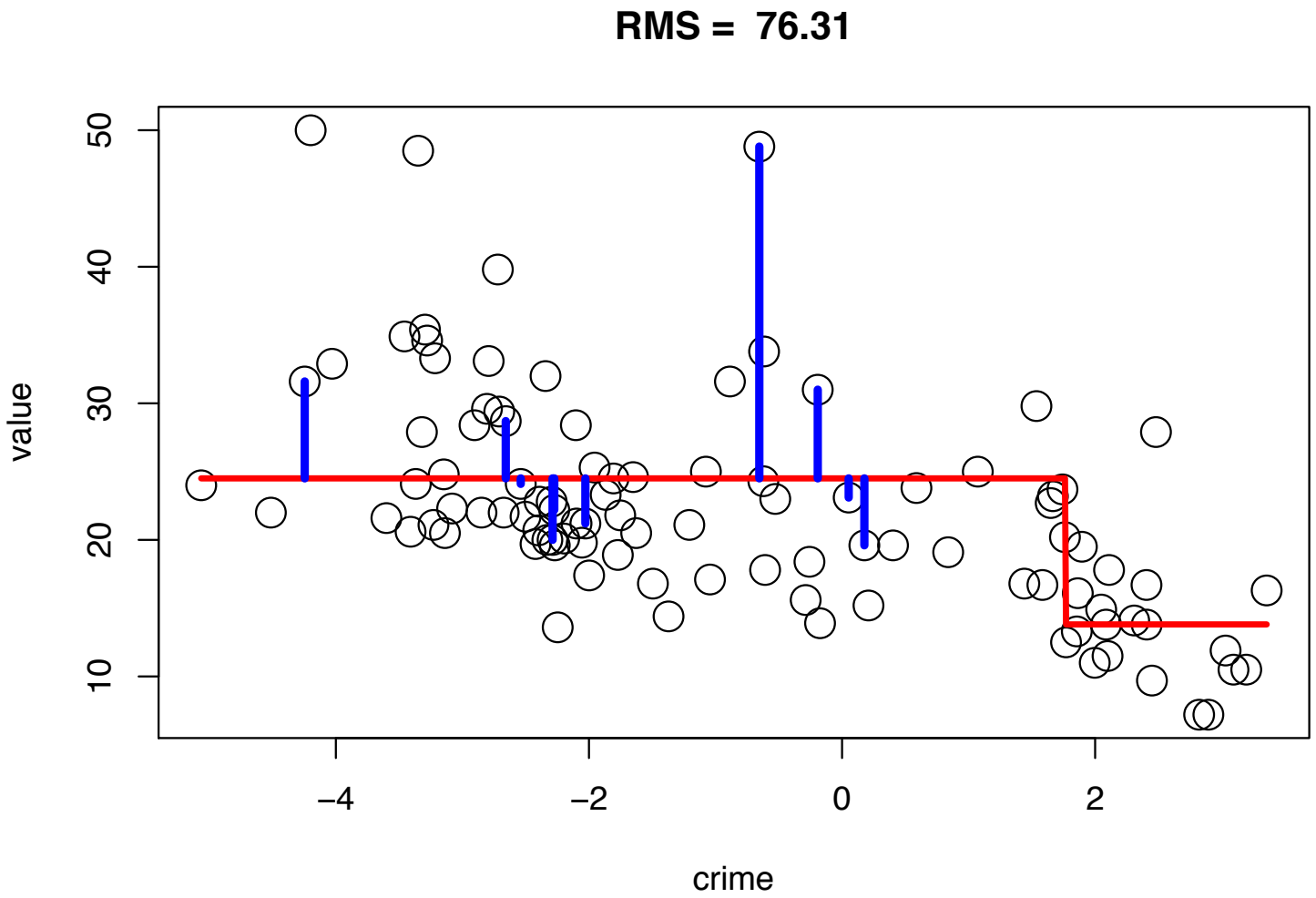
Is computationally cheaper than LOO-CV and yields comparable results.
V-fold cross-validation works by splitting the dataset randomly into V sets of equal size $S_1, \ldots, S_V$, so that $S_k \cap S_{k'} = \emptyset$ for all $k \neq k'$ and $\cup_k S_k = \{1, \ldots, n\}$.
For each $v = 1, \ldots, V$

- compute the predictor (tree) using samples $\{1, \ldots, n\} \setminus S_v$.
- predict the response for samples in set S_v with the found predictor
- record the test error for the set S_v.

Average the test error over all V sets.
Typical choices are $V = 5$ or $V = 10$.

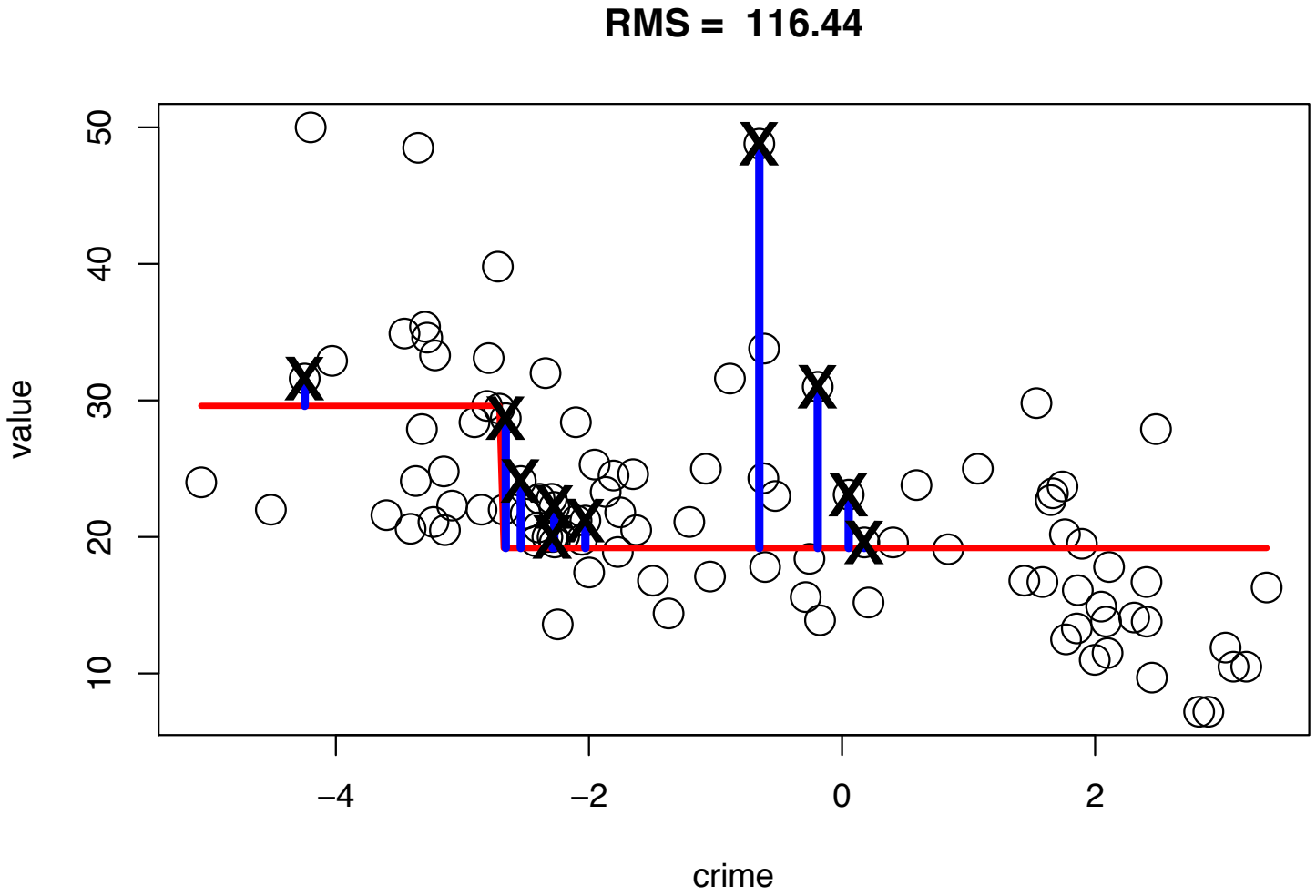
Example: Boston Housing Data

Assess now a whole block S_v of about $n/10$ of all n observations (V=10 fold CV).



Red line is fitted curve $\hat{Y}(x)$ for a tree of depth 1 (a stump). Blue vertical bar corresponds to residuals of ith observation, where i is in the to be assessed block v.

Do the same fit but leave-out observation the whole block of observations S_v.



Red line is fitted curve without using observations in block v. Blue vertical bar corresponds to LOO-CV residuals.
Repeat for all $V = 10$ blocks of all observations, each containing about $n/10$ of all samples.

Choosing the optimal tree

We would like to choose the tree that minimizes the true error rate. We dont have the test error, but can use CV-approximation $\hat{R}_{test}$ instead and choose the optimal tree T^* as

$$T^* = \text{argmin}_T \, \hat{R}_{test}(T).$$

This would require searching across all possible trees T and is clearly infeasible.
With CV, we can however search for the optimal value of one-dimensional so-called 'tuning' parameter. Here, we use tuning parameter α for tree pruning and find α by CV.

Pruning

Let $R_{train}(T)$ be the training error as a function of tree T (squared error on the training set for regression, mis-classification or entropy for classification). Minimizing $R_{train}(T)$ leads to a tree with maximal size. Minimize instead

$$(*) \qquad R_{train}(T) + \alpha \cdot \text{size}(T),$$

where the size of a tree T is measured by the number of leaf nodes.

- Either grow the tree from scratch and stop once the criterion $(*)$ starts to increase.
- Or first grow the full tree and start to delete nodes (starting at the leaf nodes), until the criterion $(*)$ starts to increase.

Second option is preferred as the choice of tree is less sensitive to “wrong” choices of splitpoints and variables to split on in the first stages of tree fitting.

Choice of α

Which value of α should be chosen ? Let T_α for $\alpha \in \mathbb{R}^+$ be the tree that is the minimizer of

$$T_\alpha = \text{argmin}_T \{R_{train}(T) + \alpha \cdot \text{size}(T)\}.$$

Want to pick α^* such that the resulting tree has minimal test error:

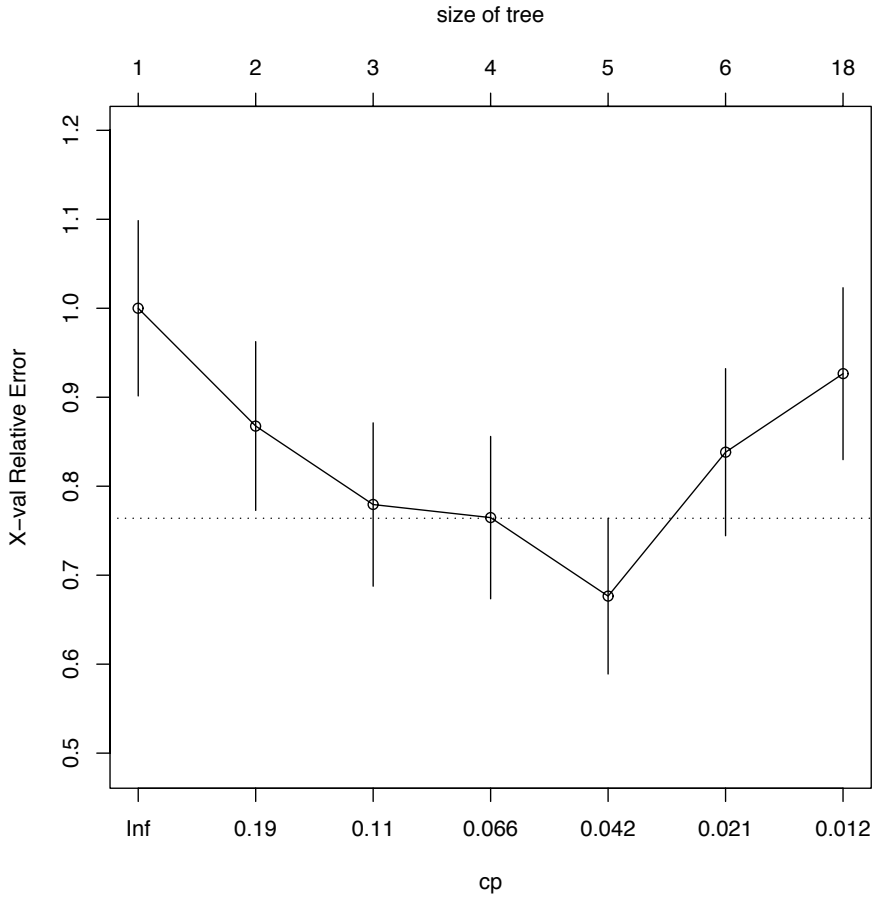
$$T_{\alpha^*} = \text{argmin}_{T_\alpha; \alpha \in \mathbb{R}^+} \hat{R}_{test}(T_\alpha).$$

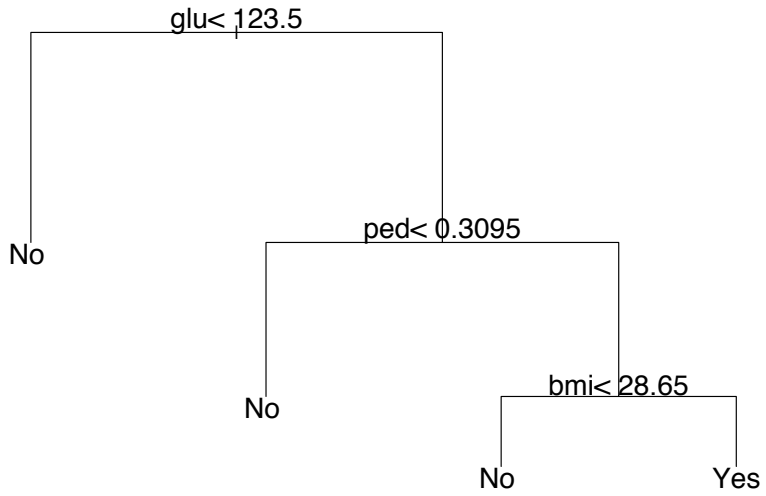
where we compute $\hat{R}_{test}$ using CV.
Its best to visualize $\hat{R}_{test}(T_\alpha)$ as a function of α.

Can plot the generalization error $\hat{R}_{test}$ of the optimal tree under criterion

$$R_{train}(T) + \alpha \cdot \text{size}(T)$$

as a function of α and pick the value of α which yields the smallest estimate of the generalization error.
For Pima Indians example:





Bias-Variance Tradeoff

Suppose

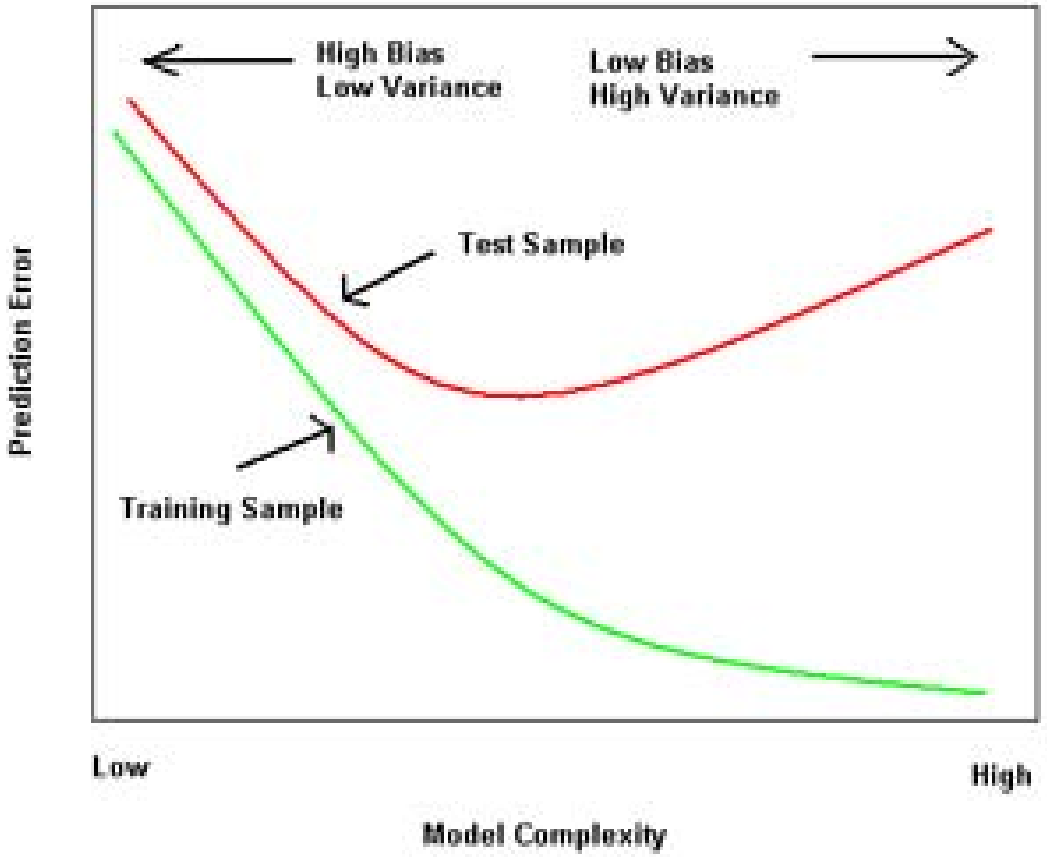
$$y = f^*(x) + \mathcal{N}(0, \sigma^2)$$

Given a dataset (X, Y), train a model $f(x; X, Y)$. How did we do, averaging over datasets?

$$\begin{aligned} & E_{X,Y}[(y - f(x; X, Y))^2] & \\ = \; & (\bar{f}(x) - f^*(x))^2 & \text{bias}^2 \\ & + E_{X,Y}[(\bar{f}(x) - f(x; X, Y))^2] & \text{variance} \\ & + (y - f^*(x))^2 & \text{noise} \end{aligned}$$

where $\bar{f}(x) = E_{X,Y}[f(x; X, Y)]$ is average prediction (averaged over datasets).

Choosing Model Complexity



The training error will always decrease if the model is made more complex (the tree grown larger). The test error will have reach a minimum at a certain model complexity (tree size) and grow if the tree is made either larger or smaller.

Pitfalls of using the training error rate

How deceptive can the training error be ?

- Assume we have n data samples $(X_1, Y_1), \ldots, (X_n, Y_n)$ and

$$Y_i \sim \mathcal{N}(0, 1)$$

  so there is no information about Y in the predictor variables X.
- Assume we take a tree with size d (the size is the number of leaf nodes), which is chosen independently of Y, so that each leaf node contains the same number of samples.

What is the expected training (apparent) error rate, as a function of tree size d?

Assume

- In total d leaf nodes.
- In each final leaf node, there are n/d samples $j_1, \ldots, j_{n/d}$.

The value of $\hat{\beta}_k$ in each leaf node k is simply the mean $\overline{Y}_k$ over all observations in node k.
The test error rate in each leaf node k is

$$R_{test} = E((Y - \overline{Y}_k)^2) = E((Y - E(Y))^2) + E((\overline{Y}_k - E(Y))^2) = 1 + \overline{Y}_k^2.$$

Averaged over independent realizations of the new test data, the expected test error rate is

$$E(R_{test}) = 1 + E(\overline{Y}_k^2)$$

The training error in each node is

$$R_{train} = \frac{d}{n} \sum_{j_1}^{j_{n/d}} (Y_i - \overline{Y}_k)^2 = \left(\frac{d}{n} \sum_{j_1}^{j_{n/d}} (Y_i - E(Y))^2 \right) - \overline{Y}_k^2.$$

The expected value of the training error rate is

$$E(R_{train}) = 1 - E(\overline{Y}_k^2).$$

The mean $\overline{Y}_k$ has a distribution $\sim \mathcal{N}(0, d/n)$. Then $\frac{n}{d}\overline{Y}_k^2 \sim \chi_1^2$ and $E(\overline{Y}_k^2) = d/n$.
The expected value of the test error rate is thus

$$E(R_{test}) = 1 + d/n$$

The expected value of the training error rate is

$$E(R_{train}) = 1 - d/n$$

In this extreme example, choosing the number of leaf nodes according to the

- training error rate leads you to choose maximal tree size $d = n$,
- test error rate leads you choose minimal tree size $d = 0$.