Lecture 16: Classification and Regression Trees

Statistical Learning and Data Mining

Xingye Qiao
Department of Mathematical Sciences
Binghamton University

E-mail: qiao@math.binghamton.edu

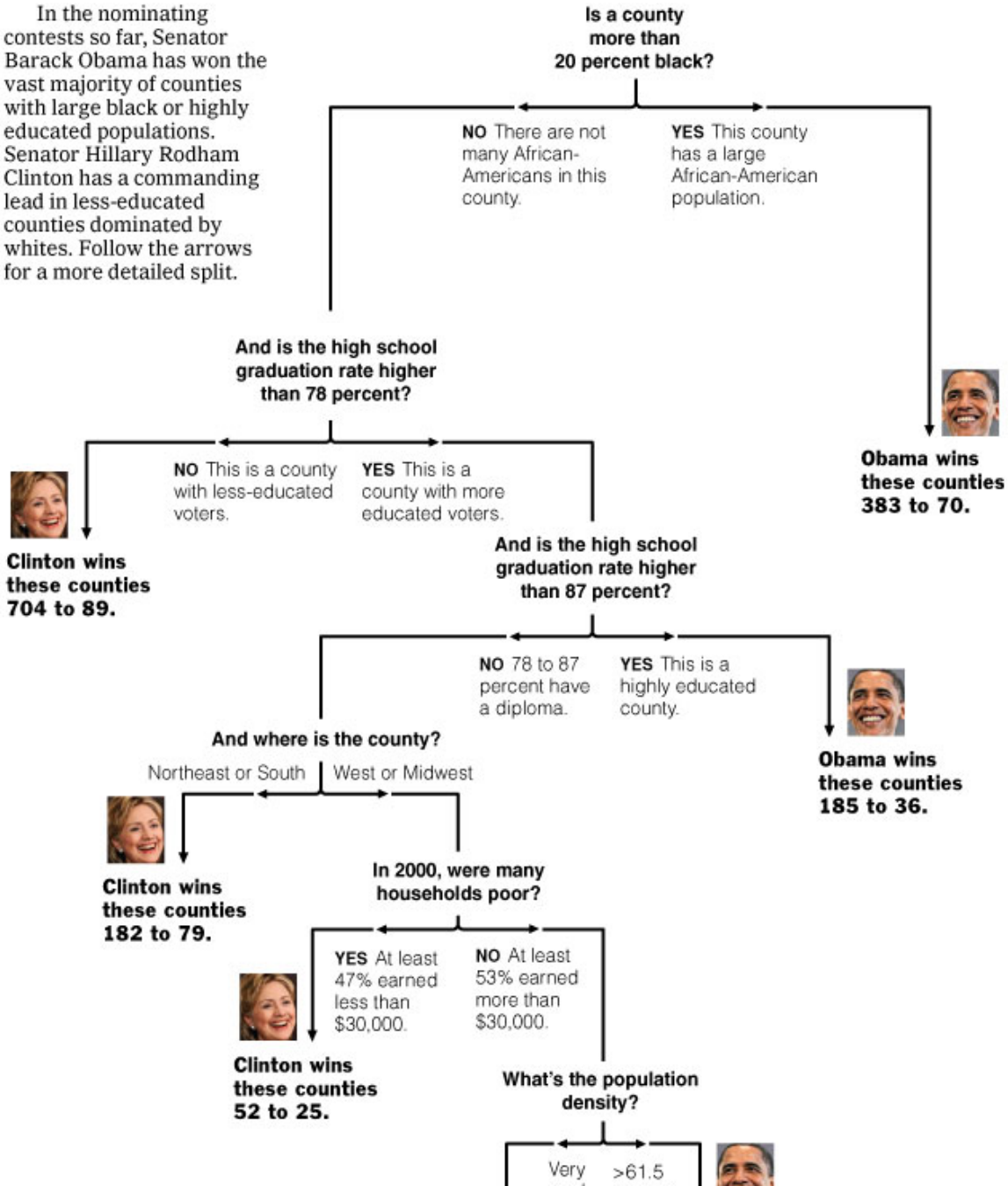
Read: ELSII Ch. 9.2, and ISLR 8.1

Outline

The next section would be . . . . . .

Decision Tree: The Obama-Clinton Divide

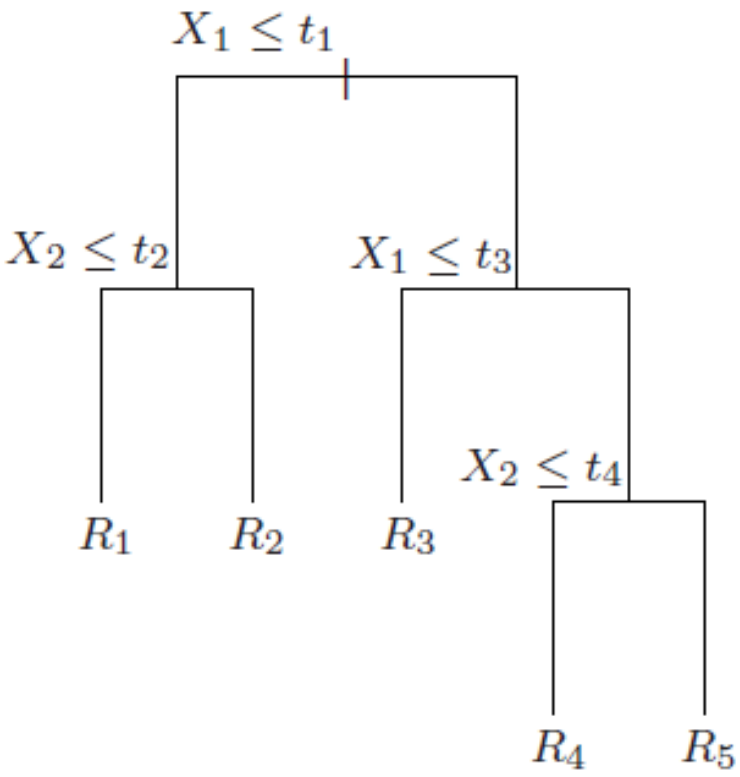
In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.

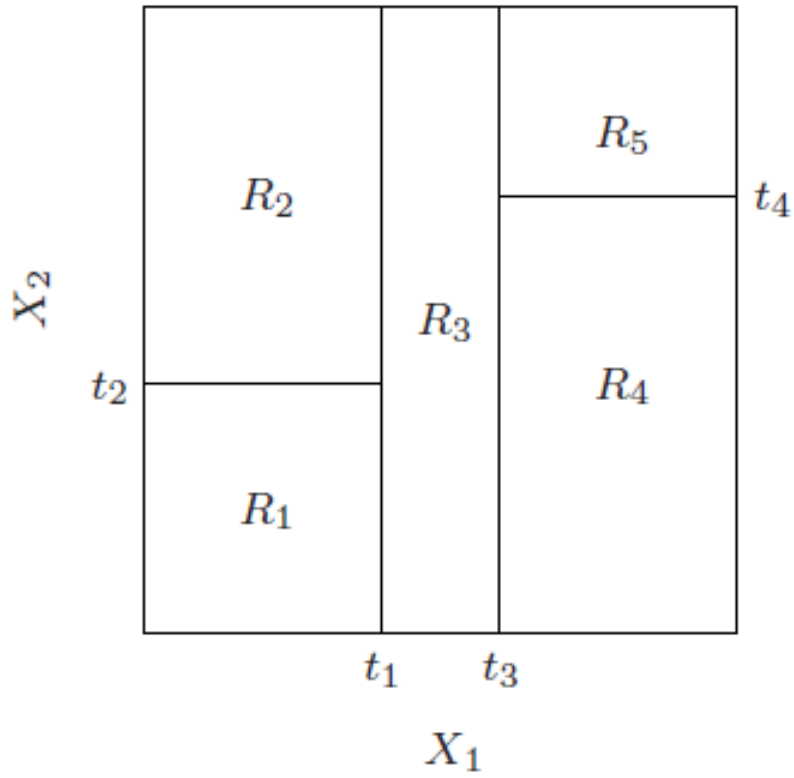


- Predict/classify a county to *Voting for Clinton* or *Voting for Obama* – a classification problem.
- The tree starts with one question, expecting a yes/no type of answer – binary independent variable.
- If the first question is enough to identify one class, then stop (see the right daughter of the first split)
- If not, then a second questions follows.
- Some questions are quantitative (continuous variables), but is converted to two outcomes by a threshold.
- Finding a threshold for a continuous variable is called splitting.

Terminology

- Root node – the point on the top
- Non-terminal node (parent node) – a node that splits into two daughter nodes
- Terminal node (leaf node) – a node that does not split.
- A single-split tree with only one root node and two terminal nodes is called a **stump**.
- CART – Classification and Regression Trees.





- Each leaf node correspond to a (basic) region.
- Each split divides the observations in a region into two smaller regions

Variable types

For each variable, how many possible splits are there?

- Ordinal or continuous variables.
  - Ordinal variable: the number of possible splits is the number of unique levels less 1.
  - Continuous variable: the number possible splits is the number of unique values less 1.
- Nominal or categorical variables.
  - Suppose that there are M categories for a variable, then the possible number of splits is $2^{M-1} - 1 = \frac{1}{2}\left(2^M - 2\right)$
  - Missing values are treated as new categories.

At each node, one need to (1) choose the best split within each variable, and then (2) choose the best variable to split.

The next section would be . . . . . .

Impurity function

- Within a node, and within a variable, we need to find a split so that the observations on this node are divided into two regions. Ideally, we hope that within each region, there are observations from only 1 class (prefect purity). If not, try to make the impurity small.
- We define an impurity function $\phi(p_1, p_2, \ldots, p_K) : \mathbb{R}^K \mapsto \mathbb{R}$
  - p_j is the estimated class proportion for Class j in the current node; hence $p_1 + p_2 + \cdots + p_K = 1$, i.e. $(p_1, p_2, \ldots, p_K)$ is on the $(K + 1)$ dimensional simplex.
  - We wish $\phi(\cdot) = 0$ at corners of the simplex, e.g. $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, etc.
  - We wish $\phi(\cdot)$ reaches its maximal value at $(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$.

Impurity function: entropy

- $(p_1, p_2, \ldots, p_K)$ can be viewed as the probability mass function for a discrete random variable X. The entropy function, defined by

$$\mathcal{H}(X) := -\sum_{j=1}^{K} p_j \log(p_j)$$

  satisfies the desired properties.
- In the case of $K = 2$, it boils down to

$$-p_1 \log(p_1) - (1 - p_1) \log(1 - p_1)$$

Impurity function: Gini index

- Another candidate function for the impurtiy is the Gini index.

$$i(\tau) := \sum_{j \neq k} p_j p_k = 1 - \sum_{j=1}^{K} p_j^2$$

  is the Gini index for node τ
- In the case of $K = 2$, $i(\tau) = 2p(1-p)$ (this is the variance of Bernoulli r.v.)

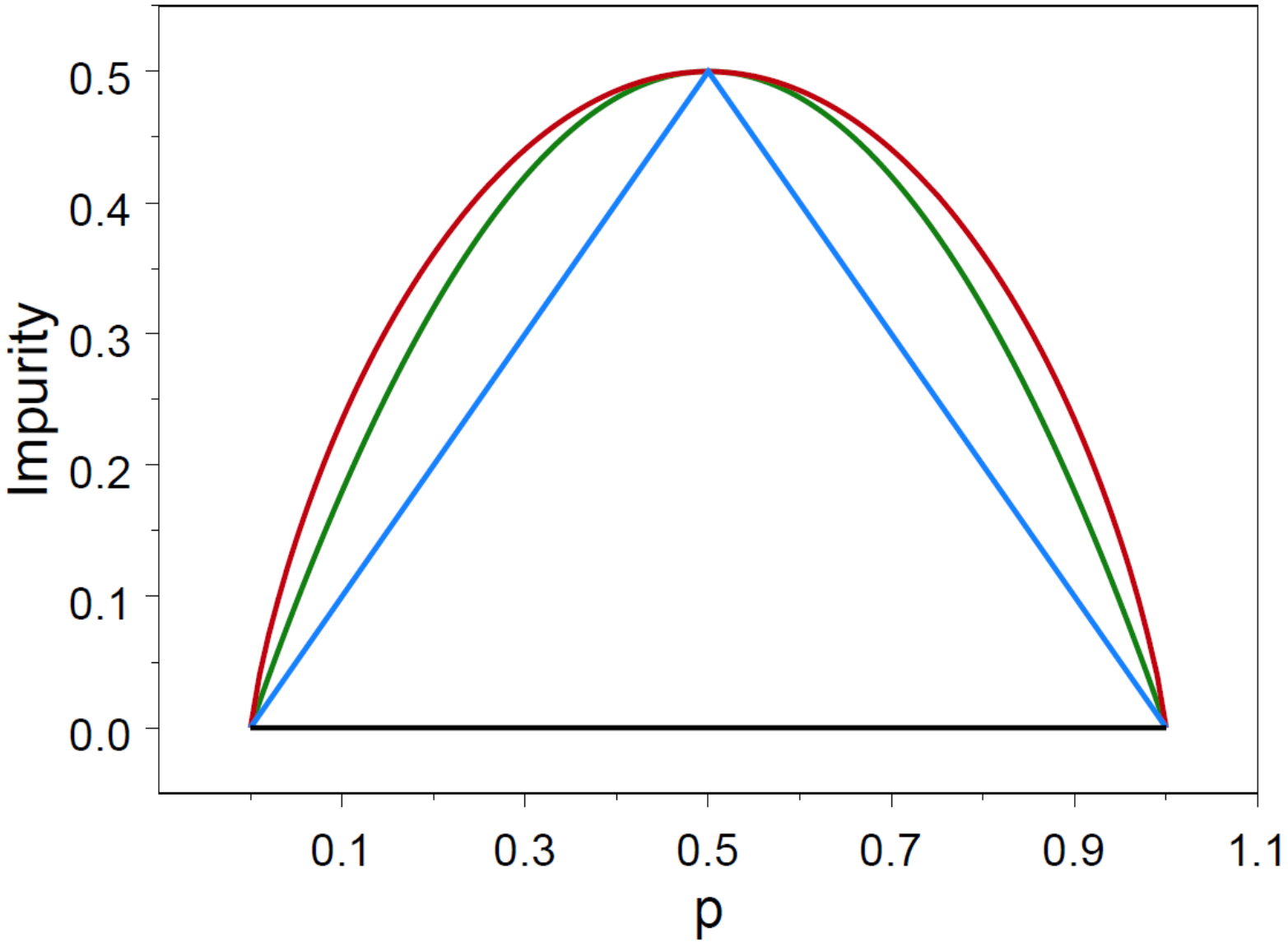


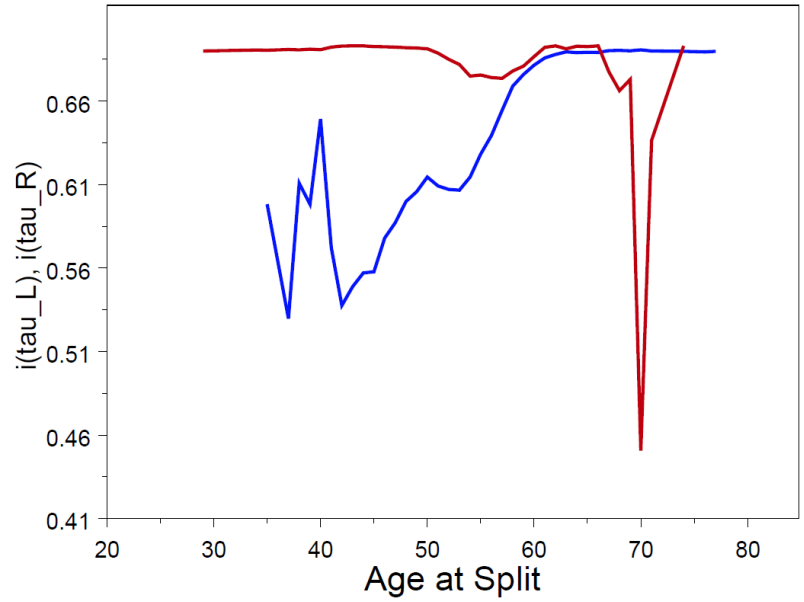
Figure: Blue: misclassification rate; Red: entropy; Green: Gini index

Minimizing the total impurity

- For each variable (say j) and at each node (say τ), we wish to find a split, so that the total impurity between the two daughter nodes is the smallest.
- More precisely speaking, any split would find a partition for $\tau = \tau_L \cup \tau_R$. Within both daughter nodes τ_L and τ_R, the impurity can be calculated. We wish both impurities are small.
- The goodness-of-split is measured by the reduction in impurity caused by the split s on the jth variable, from node τ to two separated τ_L and τ_R:

$$\Delta i(\tau, j, s) := i(\tau) - [p_L i(\tau_L) + p_R i(\tau_R)]$$

  where the total impurity of the daughters is calculated as weighted sum, with the proportions p_L and p_R as the weights.
- Since $i(\tau)$ is independent of the split, to maximize $\Delta i(\tau, j, s)$ is to minimize $p_L i(\tau_L) + p_R i(\tau_R)$.



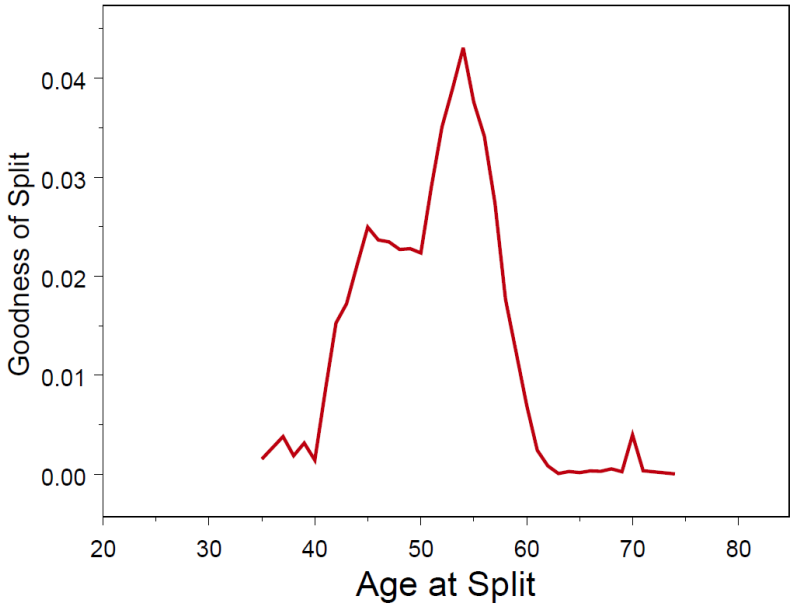


Figure: Variable "age". Left: impurities for the left and right daughters with different split values. Right: Goodness-of-split (negative correlated

Recursive Growing

- At each node, we first select the best possible split for each variable, and then select the variable which minimizes the total (weighted) impurity of the daughter nodes.
  - Note that all the variables are visited (including those which have been split before). If all variables are categorical, this should not take too long.
- Recursively split all nodes (including daughter nodes) until there is only one observation at each node. Then the tree has been **saturated**. Stop.
- Early stopping rule:
  - Stop splitting a node if there are fewer observations in it than a prespecified threshold.
  - Stop splitting a node if $\max_{j,s} \Delta(\tau, j, s)$ is not greater than a prespecified threshold [improvement is too small].

Class assignment

- For saturated tree, the class assignment for future observations in each leaf node is just the class of the only training observation on it.
- For other trees, the class assignment for future observations in each leaf node is the class with the majority presence at the node. When we view a tree as a classifier $\phi : \mathcal{X} \mapsto \mathbb{R}$

$$\phi(\boldsymbol{x}) = \operatorname*{argmax}_j p_j(\tau), \text{ where } \boldsymbol{x} \in \tau$$

where

$$p_j(\tau) := \frac{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau, Y_i = j]}}{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau]}}$$

is the proportion of the jth class at node τ (from the training data)

Training misclassification rate

- Data points whose true labels are $\operatorname{argmax}_j p_j(\tau)$ are correctly classified. The rest are misclassified. Hence, at leaf node τ, the misclassification rate is

$$R(\tau) = 1 - \max_j p_j(\tau)$$

  For $K = 2$, reduce to $R(\tau) = \min(p, 1 - p)$
- Let

$$q(\tau) := \frac{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau]}}{n}$$

  be the proportion of observations at leaf node τ, then the total training data misclassification rate for a tree T is

$$R(T) = \sum_{\ell=1}^{|T|} R(\tau_\ell) q(\tau_\ell)$$

  where ℓ is the leaf node index and $|T|$ is the number of leaf nodes in T.

The next section would be . . . . . .

Size of the tree matters

- $R(T)$ is clearly not a good measure to assess the performance of T. Otherwise, one would always choose the saturated tree for classification, which would be overfitting.
- Recall R^2 in regression.
- Over- and under-fitting exist, like in other methods.
- Leo Breiman and his colleagues showed that a better strategy, compared to early stopping or saturated trees, is to let the tree grow to saturation, and prune (cut) some branches to make the tree smaller, so as to find a best subtree of the fully saturated tree (called T_{max}.)

Cost complexity pruning

- If $R(T)$ is viewed as the loss function (as a matter fact, 0-1 misclassification loss), then introduce a regularization term/penalty term to discourage trees that are too big.
- Prune the tree from bottom up, until

$$R(T^*) + \alpha|T^*|$$

  reaches the minimal value, where T^* is a subtree of T.
- α is called the *complexity parameter* (cp) by some software.
- For each α there is a (candidate) 'optimal' tree.
- There is a solution path, with x-axis being the value of α and the optimally pruned tree at each α value. Note that they are a unique sequence of nested trees.
- We can use an independent tuning data set or a cross-validated error to choose the best tuning parameter value α, like in other methods.

The next section would be . . . . . .

CART for regression

- CART can be used for regression as well
- The prediction value for the leaf node τ is a constant, and equals to the average of the response values of the observations at node τ, i.e.

$$\widehat{y}(\boldsymbol{x}|\tau) = \frac{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau]} Y_j}{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau]}}$$

- The mean squared error for the training data at leaf node τ is then

$$\frac{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau]} (y_i - \widehat{y}(\boldsymbol{x}_i|\tau))^2}{\sum_{i=1}^{n} \mathbb{1}_{[\boldsymbol{x}_i \in \tau]}}$$

  Note that this is essentially the same as the *sample variance* of the observations at node τ

Splitting strategy

- In classification, we used some impurity measure.
- In regression, we minimize the (weighted) sum of the prediction mean squared errors for the two daughter nodes. That is, we want to find the jth variable and split s, to minimize

$$p_L \widehat{\sigma}^2(\tau_L | j, s, \tau) + p_R \widehat{\sigma}^2(\tau_R | j, s, \tau)$$

where $\widehat{\sigma}^2(\cdot)$ is the sample variance at a daughter node.

Pruning and tuning parameter

- Again, we could let the tree to grow to saturation. But there would be severe overfitting if not controlled.
- Similar to the classification case, we introduce a penalty term for the size of a tree. Want the best subtree of T_{max} so that

$$R(T) + \alpha|T|$$

  is minimized.
- Each α leads to an "optimal" subtree.
- To choose between different tuning parameter value α, we can use an independent tuning data set or cross validation.

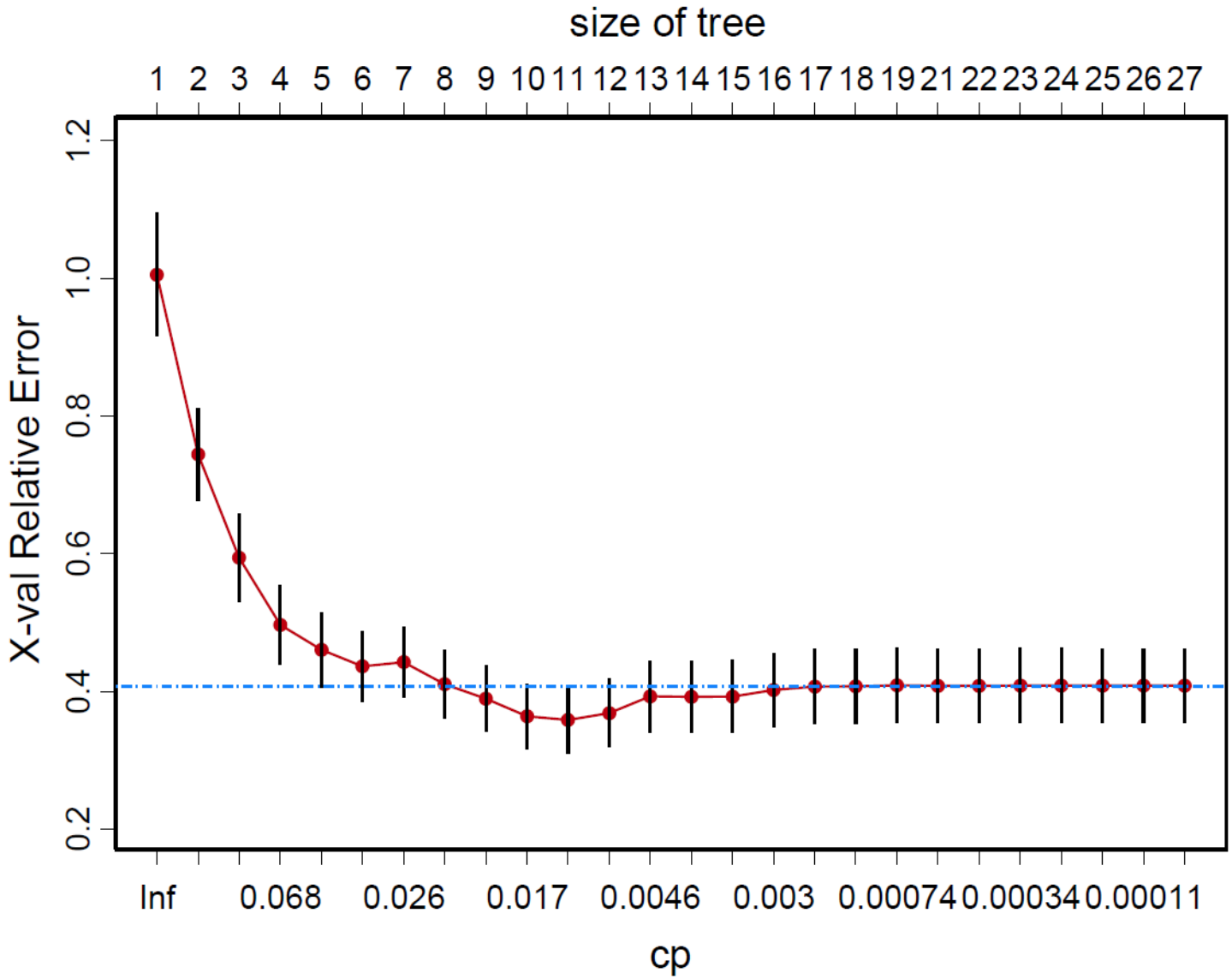
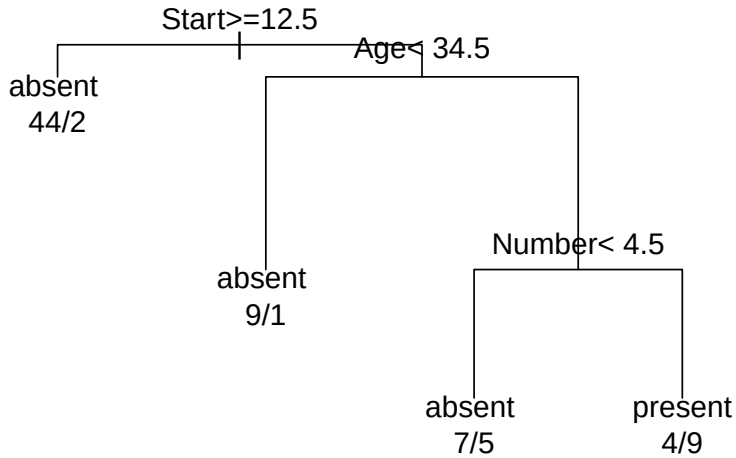


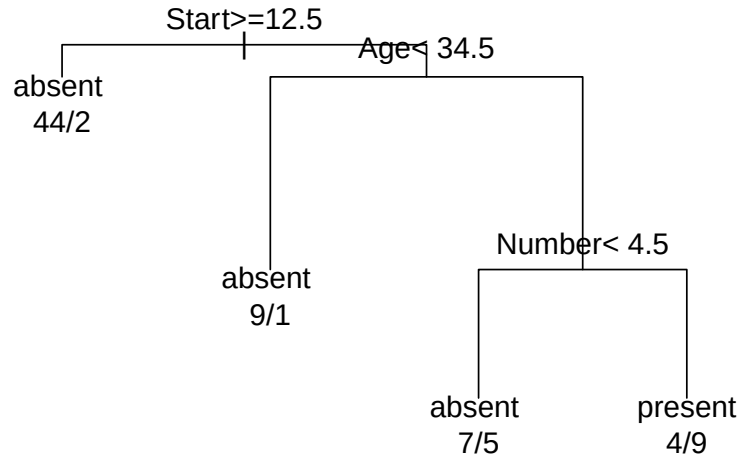
Figure: 10-fold CV results of a regression tree.

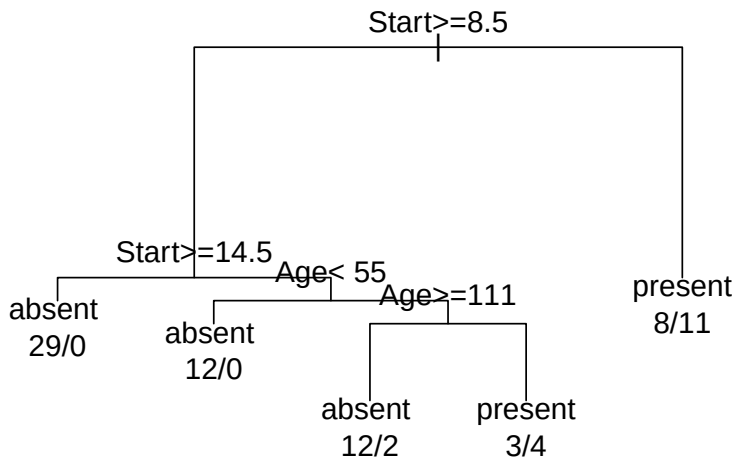
Data on Children who have had Corrective Spinal Surgery

- The kyphosis data frame has 81 rows and 4 columns. representing data on children who have had corrective spinal surgery
- Kyphosis – a factor with levels (absent, present) indicating if a kyphosis (a type of deformation) was present after the operation.
- Age – in months
- Number – the number of vertebrae involved
- Start – the number of the first (topmost) vertebra operated on.

```
> kyphosis
   Kyphosis Age Number Start
1    absent  71      3     5
2    absent 158      3    14
3   present 128      4     5
4    absent   2      5     1
5    absent   1      4    15
6    absent   1      2    16
7    absent  61      2    17
8    absent  37      3    16
9    absent 113      2    16
10  present  59      6    12
11  present  82      5    14
12   absent 148      3    16
```


Start>=12.5
Age< 34.5
absent
44/2
absent
9/1
Number< 4.5
absent
7/5
present
4/9


Start>=12.5
Age< 34.5
absent
44/2
absent
9/1
Number< 4.5
absent
7/5
present
4/9


Start>=8.5
Start>=14.5
Age< 55
Age>=111
absent
29/0
absent
12/0
absent
12/2
present
3/4
present
8/11


Start>=8.5
absent
56/6
present
8/11

R: use rpart package

```
library(rpart)
data(kyphosis)
head(kyphosis)
fit1 <- rpart(Kyphosis ~ Age + Number + Start, data = kypho
              parms=list(split = "information"))

fit2 <- rpart(Kyphosis ~ Age + Number + Start, data = kypho
              parms=list(split = "information"),
              control = rpart.control(cp = 0.03))

fit3 <- rpart(Kyphosis ~ Age + Number + Start, data = kypho
              parms=list(split = "gini"))

fit4 <- rpart(Kyphosis ~ Age + Number + Start, data = kypho
              parms=list(split = "gini"),
              control = rpart.control(cp = 0.05))
```

Strengths & Weaknesses of Trees.

Strength

- Trees are very easy to explain to lay people
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Weakness

- Cannot find hidden structure or model, that can be found by other classification methods. Mainly prediction-driven.
- CARTs are not very stable. A small change to the data can lead to a very different tree.
- Lack of smoothness. If the *true* classification boundary turns out to be a smooth hyperplane, CART cannot reconstruct that (merely approximate it at best.)
- Three-way or higher-order way splitting is possible, but a binary split is preferred (due to computational concerns).
- Cannot model additive structure.