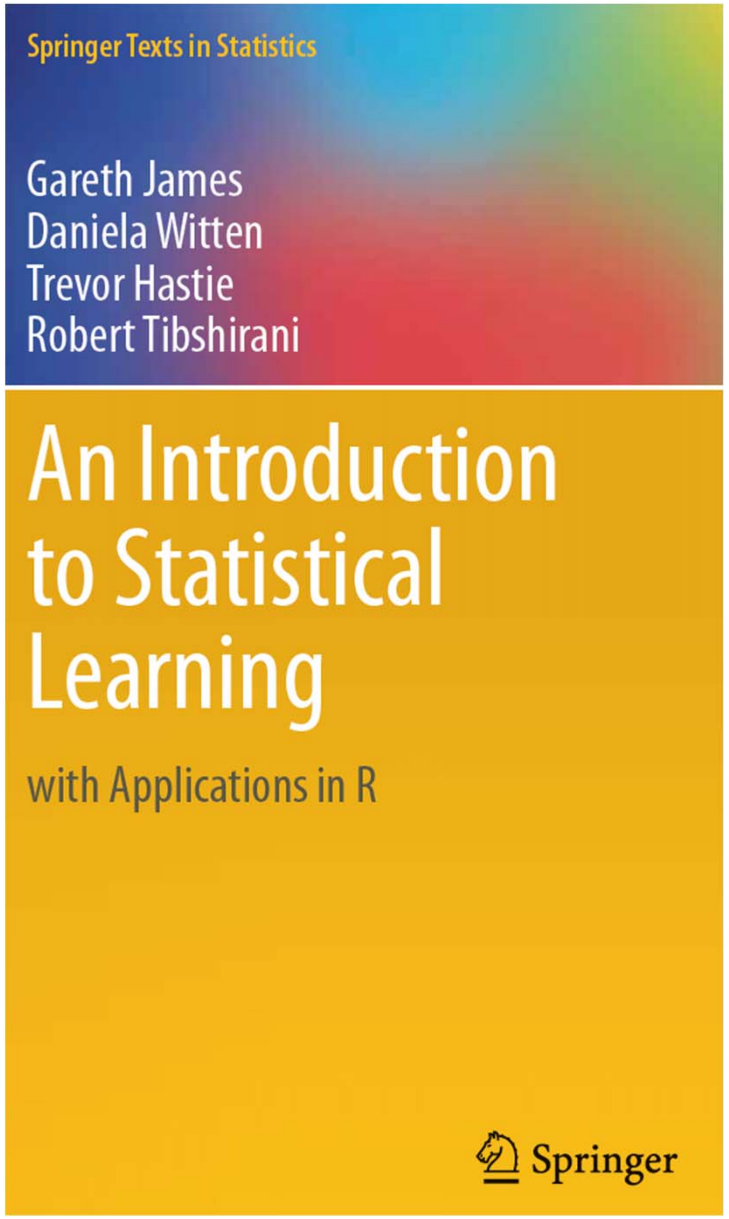
Decision Trees

Alberto Paccanaro
EMAp – FGV

z z z 1sdffdqdurode1ruj

Material and images in these slides are from (or adapted from):
Gareth, Witten, Hastie, Tibshirani – An Introduction to Statistical Learning with Applications in R, 2013
Few images are from: *Duda, Hart, Stork – Pattern classification, 2006* and *T. Mitchell, Machine Learning, 1996.*

Lecture based on this textbook:


Springer Texts in Statistics

Gareth James
Daniela Witten
Trevor Hastie
Robert Tibshirani

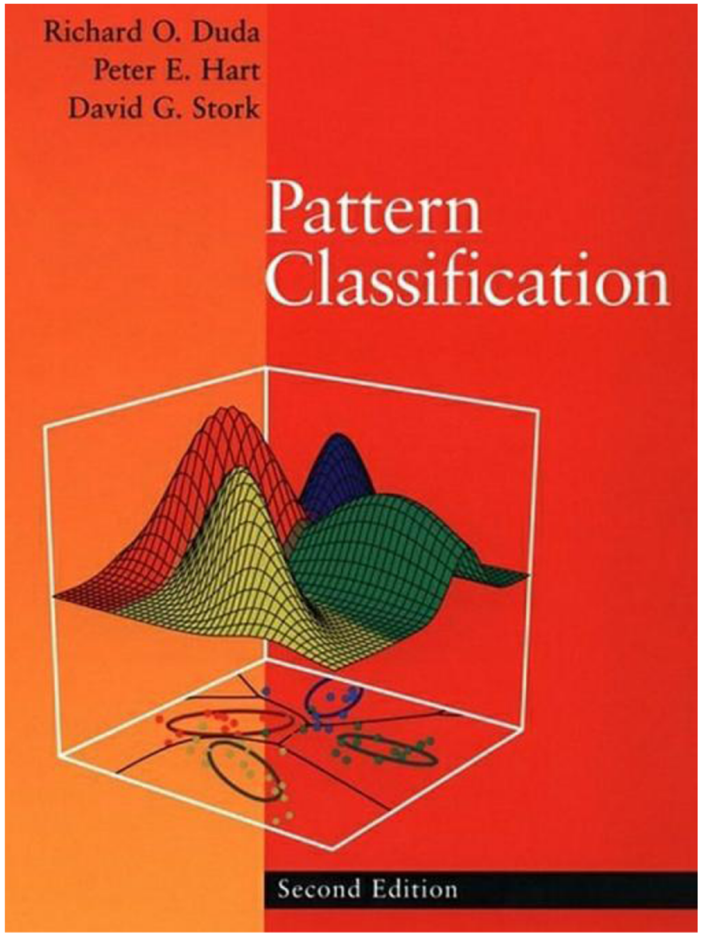
An Introduction to Statistical Learning

with Applications in R

Springer

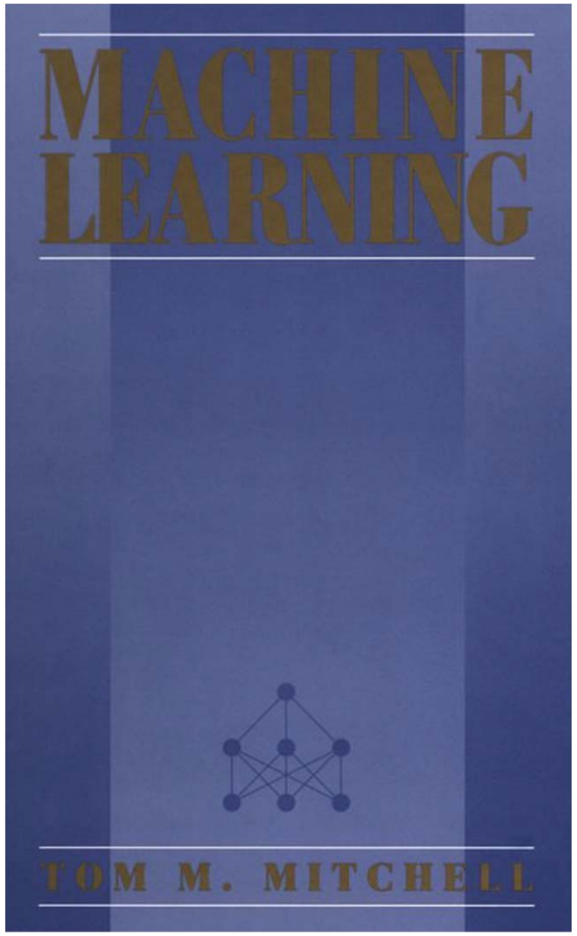
Chapter 8
Tree-Based methods

Other relevant references:



DHS 2006

Excellent reference book for machine learning
The chapter of decision trees is beautiful ☺



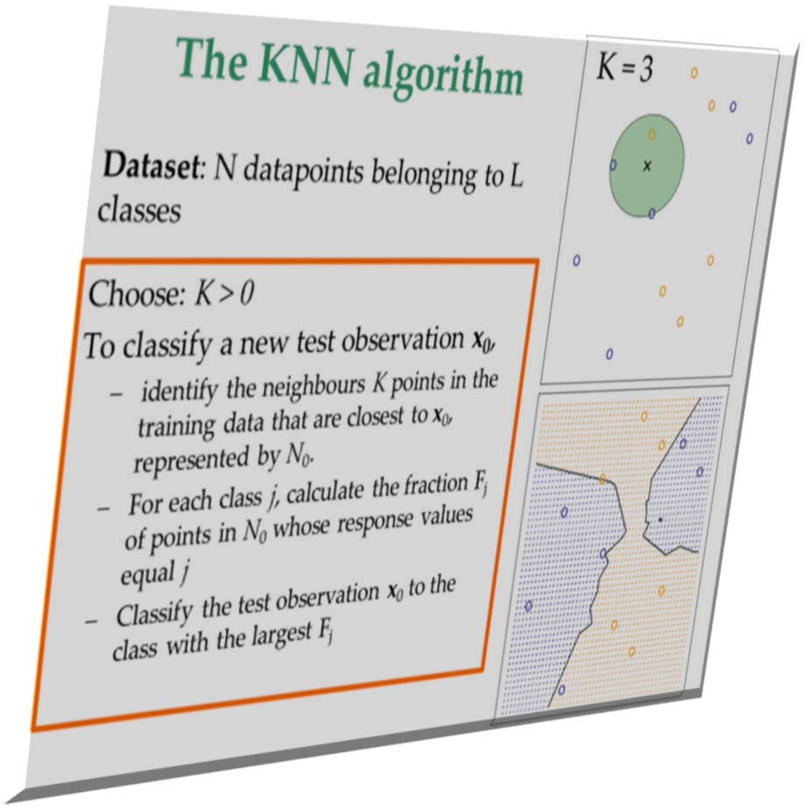
1996

A classic, very "CS" textbook

Non metric methods

Methods until now – we have a natural **measure of distance between feature vectors** representing datapoints

Input vectors sufficiently "close" lead to similar outputs.



The KNN algorithm

Dataset: N datapoints belonging to L classes

Choose: $K > 0$

To classify a new test observation x_0,

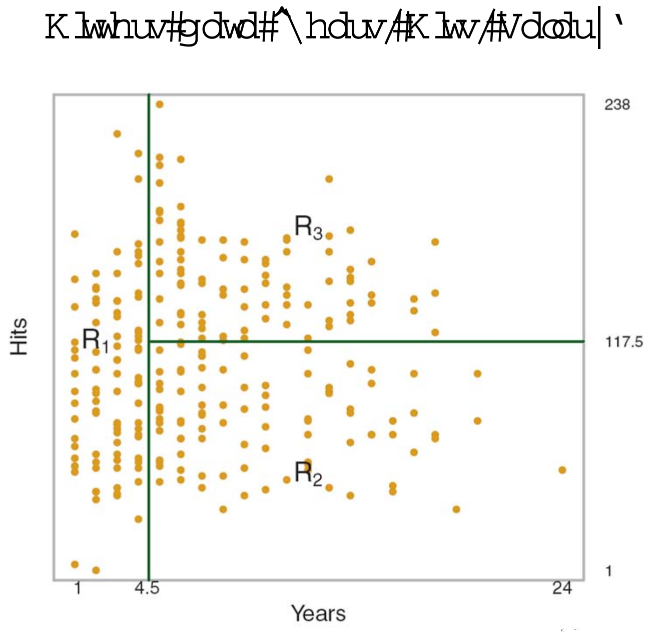
- identify the neighbours K points in the training data that are closest to x_0, represented by N_0.
- For each class j, calculate the fraction F_j of points in N_0 whose response values equal j
- Classify the test observation x_0 to the class with the largest F_j

1. How about classification problem involves *nominal* data?

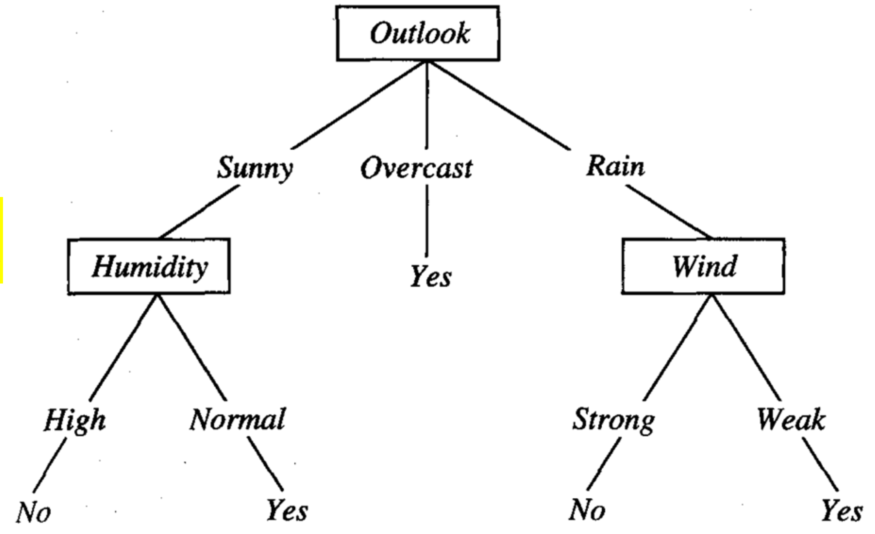
2. KNN, de facto, divides the input space. *How about diving up the input space directly, without defining a "distance"?*

The idea

1. Segment the **input space** into regions.
2. To predict for a new datapoint: use the mean (for *regression*) or the majority label (for *classification*) of the training points in the region to which it belongs.



The set of splitting rules for segmenting the input space can be summarized in a **tree** ➔ *simple and useful for interpretation*



Points to note…

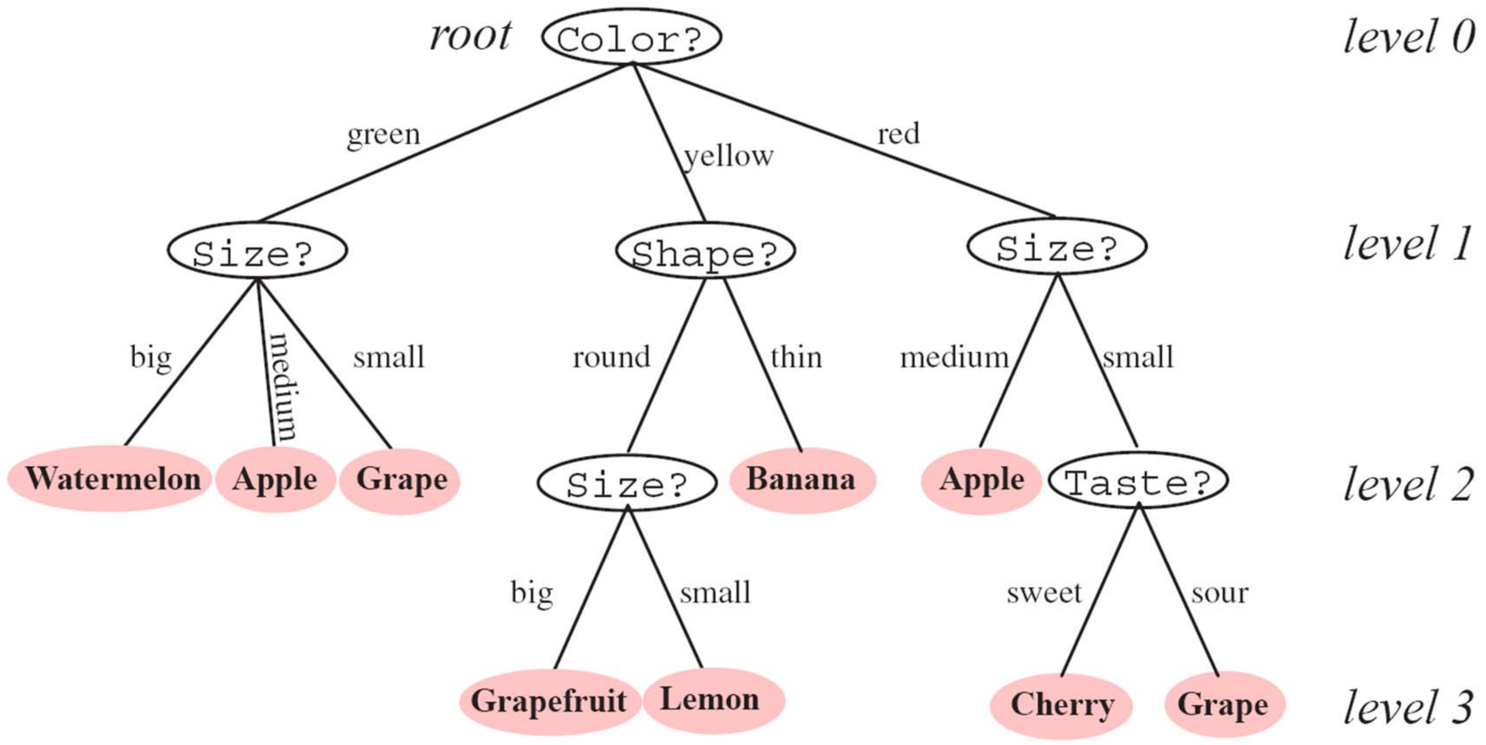


Figure 8.1: Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, `Size?`, appears in different places in the tree, and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**).

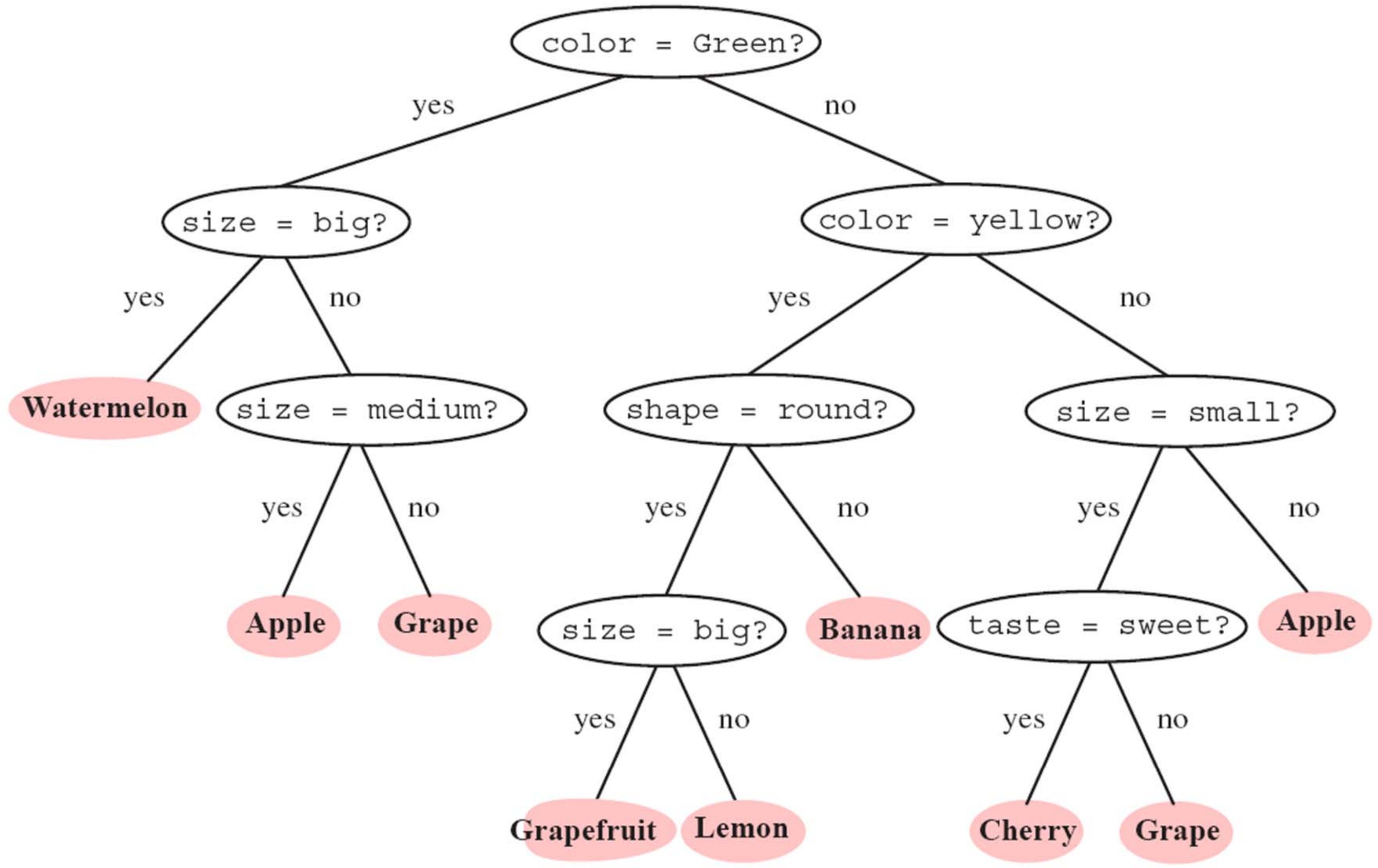


Figure 8.2: A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree, i.e., one having branching factor $B = 2$ throughout. By convention the "yes" branch is on the left, the "no" branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1.

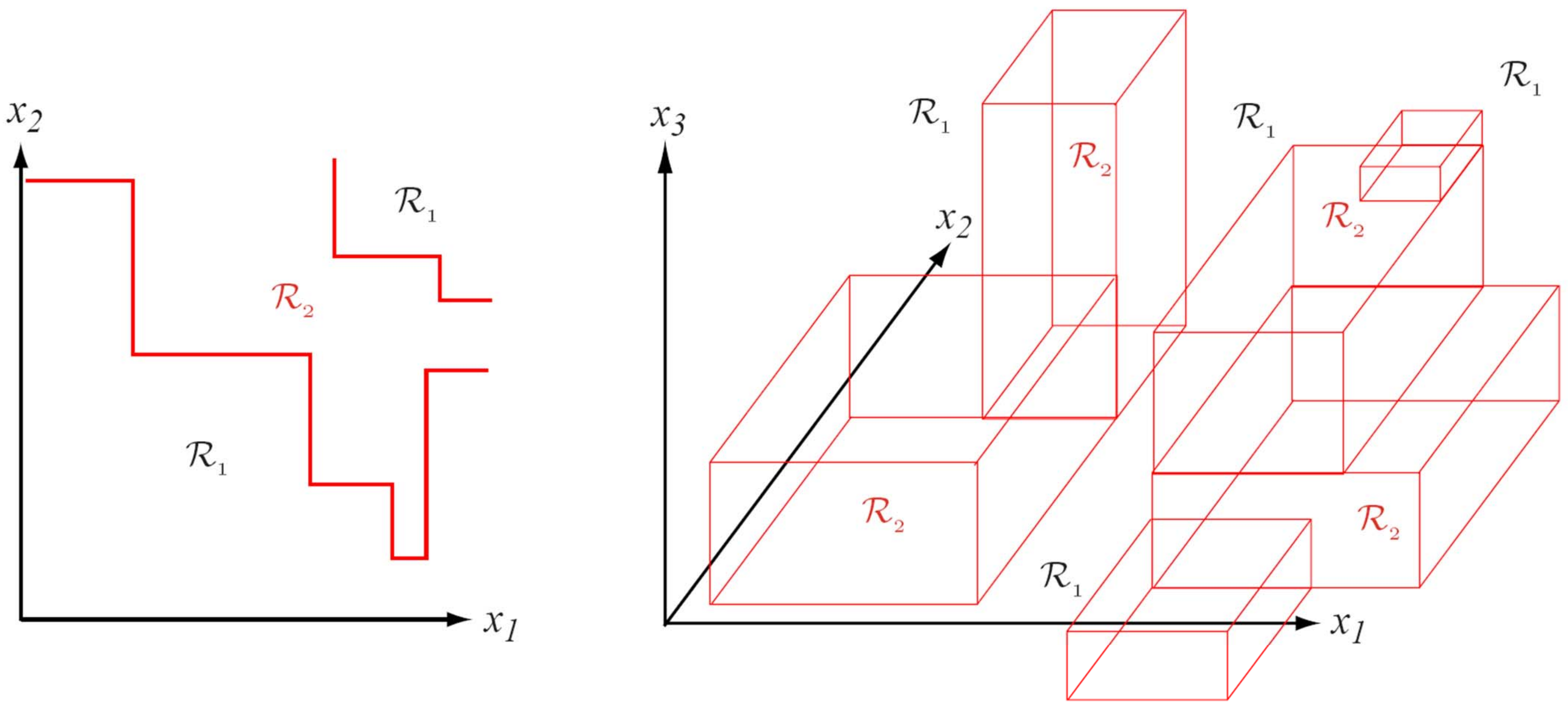
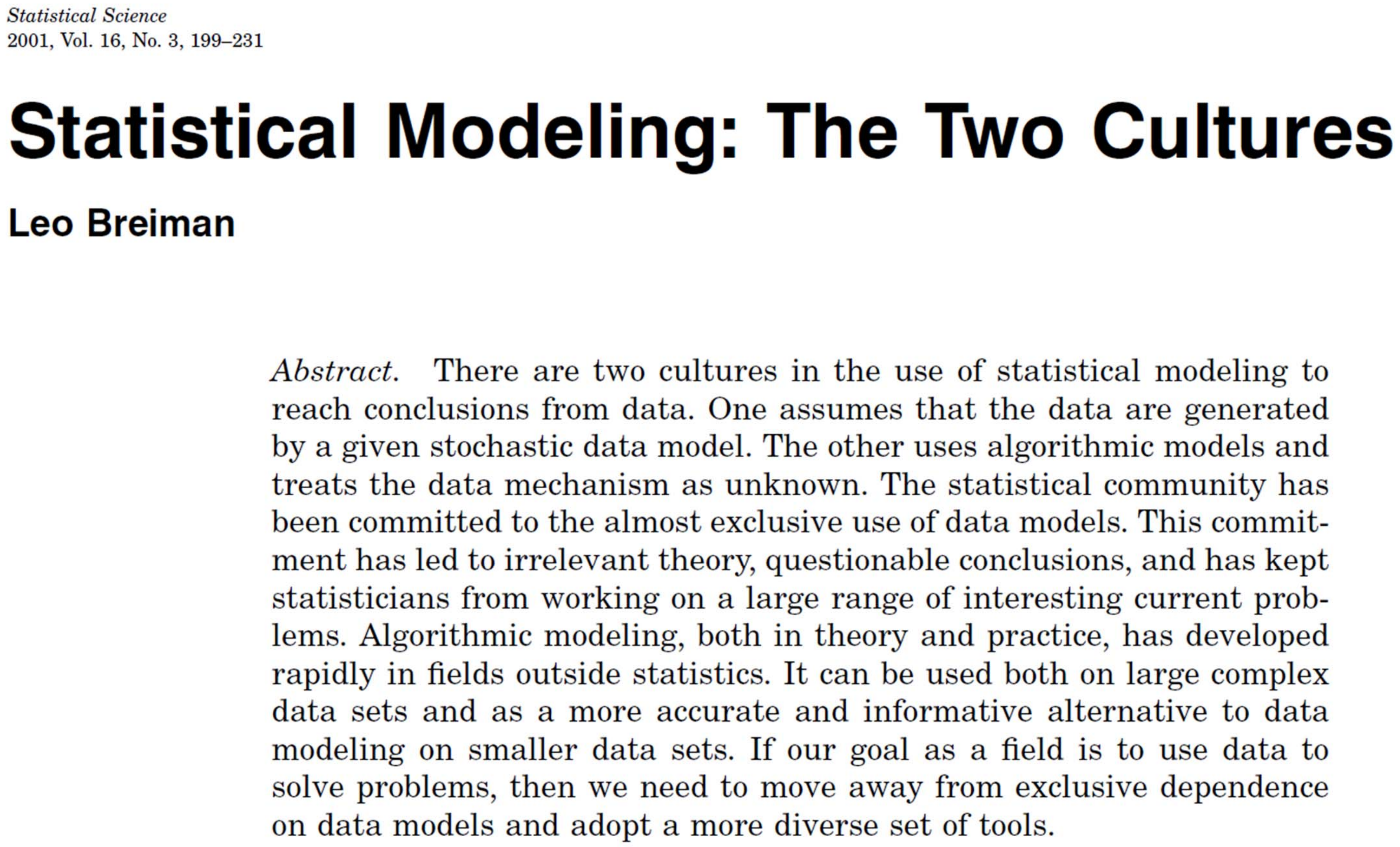


Figure 8.3: Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked $\mathcal{R}_1$ and $\mathcal{R}_2$ in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary can be approximated arbitrarily well.

Decision trees development

Ross Quinlan (CS)
1986: ID3
1992: C4.5
Currently: C5.0

Leo Breiman (Stats)
1984: CART
(1996: Bagging)
2001: Random forests



Statistical Science
2001, Vol. 16, No. 3, 199–231

Statistical Modeling: The Two Cultures

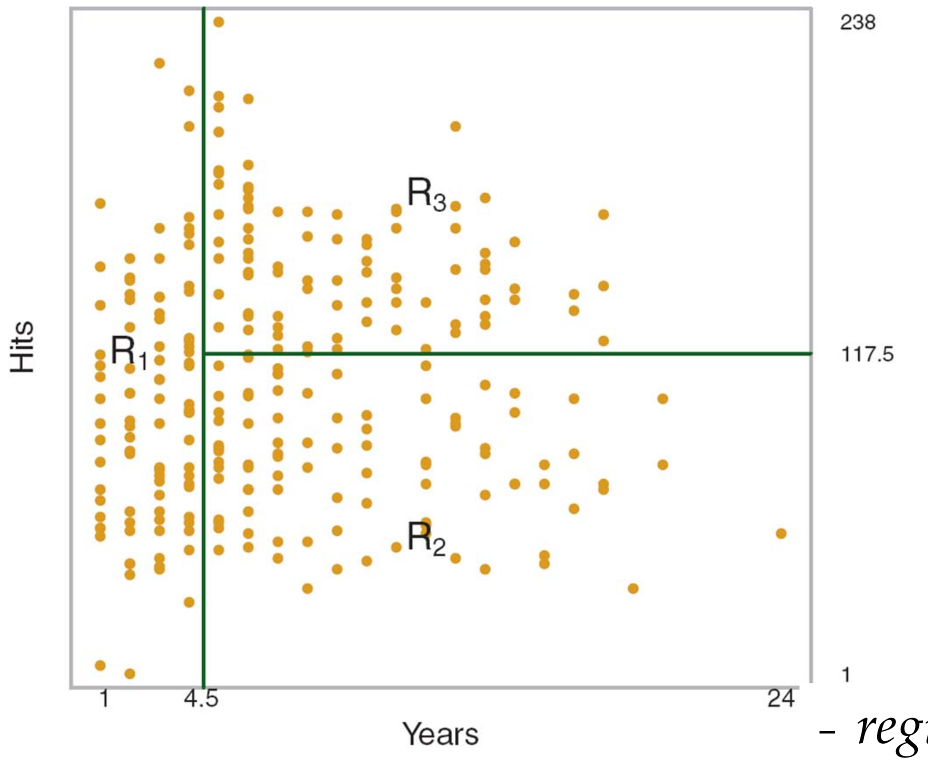
Leo Breiman

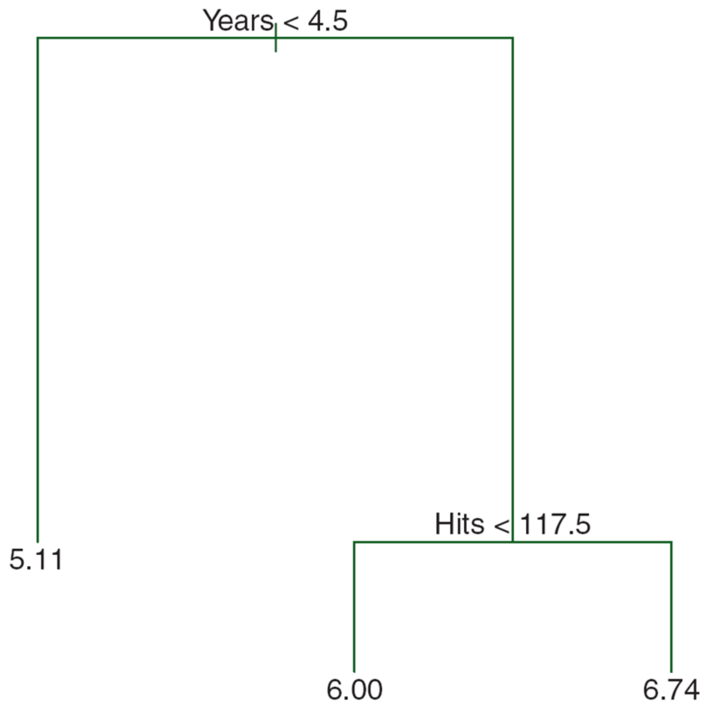
Abstract. There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models. This commitment has led to irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems. Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.

(I will put the paper on the class website, in case you are interested).

Regression trees

Example: K lwwhuv#gdwd#^\hduv/#K lwv/#Vdodu|`
we want to predict salary from years and hits.





- *regions (high-dimensional* ***rectangles****, or "boxes")*
- *leaves (terminal nodes)*
- *internal nodes*

Two steps

1. Divide the set of possible values for each feature $X_1, X_2, \ldots, X_p$ into j distinct and non-overlapping regions, $R_1, R_2, \ldots, R_j$.

2. For every observation that falls into the region R_j, make the same prediction: <u>the mean of the response values</u> for the training observations in R_j . ✓

Step 1: constructing the regions

Goal: find boxes $R_1, \ldots, R_j$ that minimize:

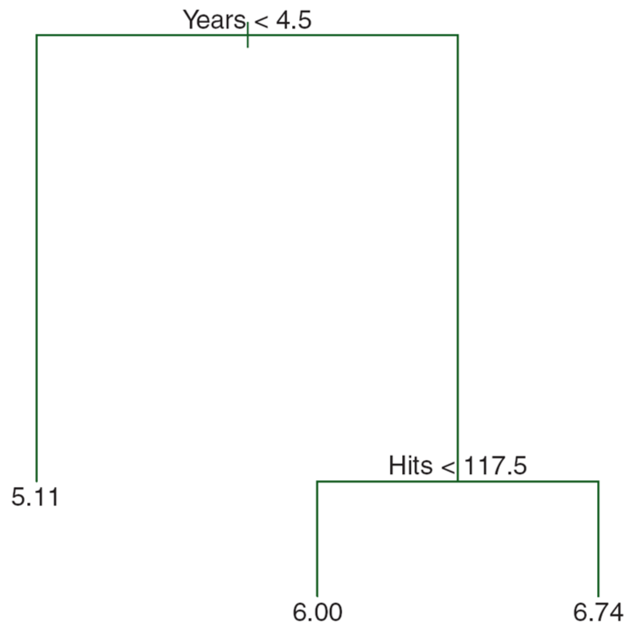
infeasible to consider every possible partition…

$$\boxed{\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2}$$

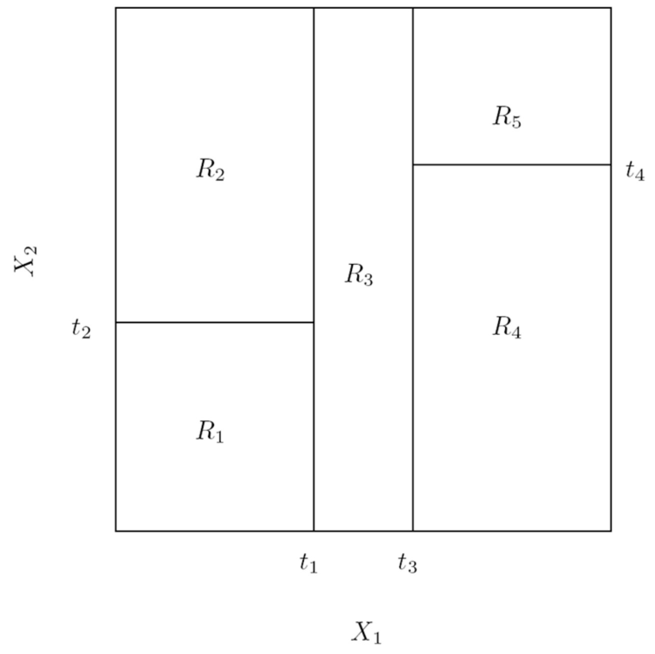
$\hat{y}_{R_j}$ mean response for the training points within the j box

Recursive Binary Splitting:

- **top-down:** begins at the top of the tree (all points are in a single region) and iteratively splits the space (each split = two new branches).
- **Greedy**: a best split is made at each specific step



At each step: we select the feature X_j and the cutpoint s



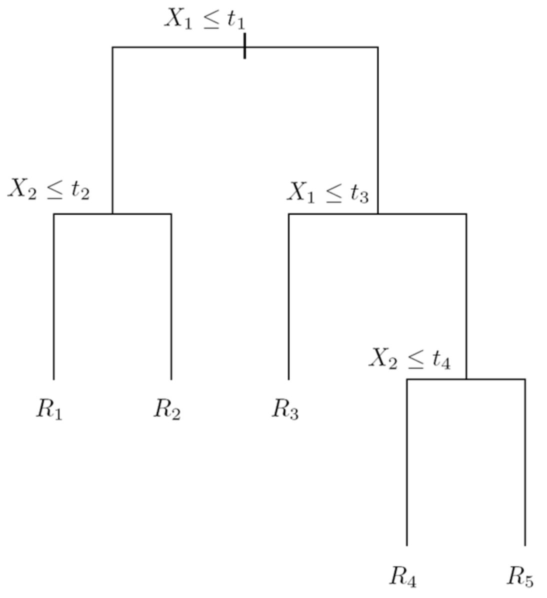
We define the pair of half-planes:

$$R_1(j, s) = \{X | X_j < s\}$$

$$R_2(j, s) = \{X | X_j \geq s\}$$

choosing *j* and *s* that minimize:

$$\sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$



Continue the process until stopping criterion is reached (e.g. maximum number of points per leaf)

Pruning (and overfitting)

Question for you: what characterizes the ***complexity*** *of a tree? What are the "parameters" here ???*

A complex tree will overfit the data

"In the extreme but rare case, each leaf corresponds to a single training point and the full tree is merely a convenient implementation of a lookup table; it thus cannot be expected to generalize well in (noisy) problems…" (DHS, section 8.3.3)

The idea: we grow a very large tree T_0, and then prune it back in order to obtain a subtree (*pruning*).

Given a subtree, we can estimate its test error using cross-validation. But it cannot be done for every subtree…

Cost complexity pruning: we consider a sequence of trees indexed by a nonnegative tuning parameter α.

For each value of α there corresponds a subtree $T \subset T_0$:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$ *

$|T|$ = *number of leaves*

- α controls the subtree's complexity
- Increasing α from zero branches get pruned
- Select α using cross-validation.

* can be seen as using a Lagrange multiplier for size, so finding the minimizing trees for all α is equivalent to finding the trees with minimum of $\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2$ for each size (more details in *B.D.Ripley, Pattern Recognition and Neural Networks, 1996*)

In practice…

You can think of α as controlling the total number of leaves in the tree.

1. all pairs of neighboring leaf nodes (i.e., ones linked to a common antecedent node, one level above) are considered for elimination.

2. Any pair whose elimination yields a satisfactory (small) increase in performance on the crossvalidation set is eliminated, and the common antecedent node declared a leaf.

Algorithm 8.1 *Building a Regression Tree*

```
1. Use recursive binary splitting to grow a large tree on the training
   data, stopping only when each terminal node has fewer than some
   minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a
   sequence of best subtrees, as a function of $\alpha$.

3. Use K-fold cross-validation to choose $\alpha$. That is, divide the training
   observations into $K$ folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 1 and 2 on all but the $k$th fold of the training data.

   (b) Evaluate the mean squared prediction error on the data in the
   left-out $k$th fold, as a function of $\alpha$.

   Average the results for each value of $\alpha$, and pick $\alpha$ to minimize the
   average error.

4. Return the subtree from Step 2 that corresponds to the chosen value
   of $\alpha$.
```

Classification trees

- Differences with Regression trees:
  1. The prediction is the most commonly occurring class of training datapoints
  2. Often interested in the *class proportions* among the training datapoints in that region

- In regression trees $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$ quantifies "*how much the desired output for the training points in the region is different from the output that we are assigning to that region*". We need something else ☺

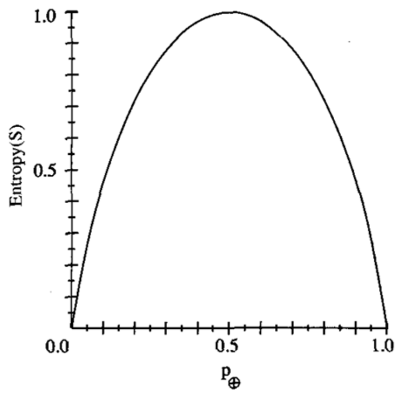
Node purity

Since, for a region, we plan predict the most common class in that region, we need to “quantify” how frequently points do not belong to that class.

Gini index: $$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

small value if all $\hat{p}_{mk}$ are close to 0 or 1.

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



Entropy as the proportion of positives varies between 0 and 1

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

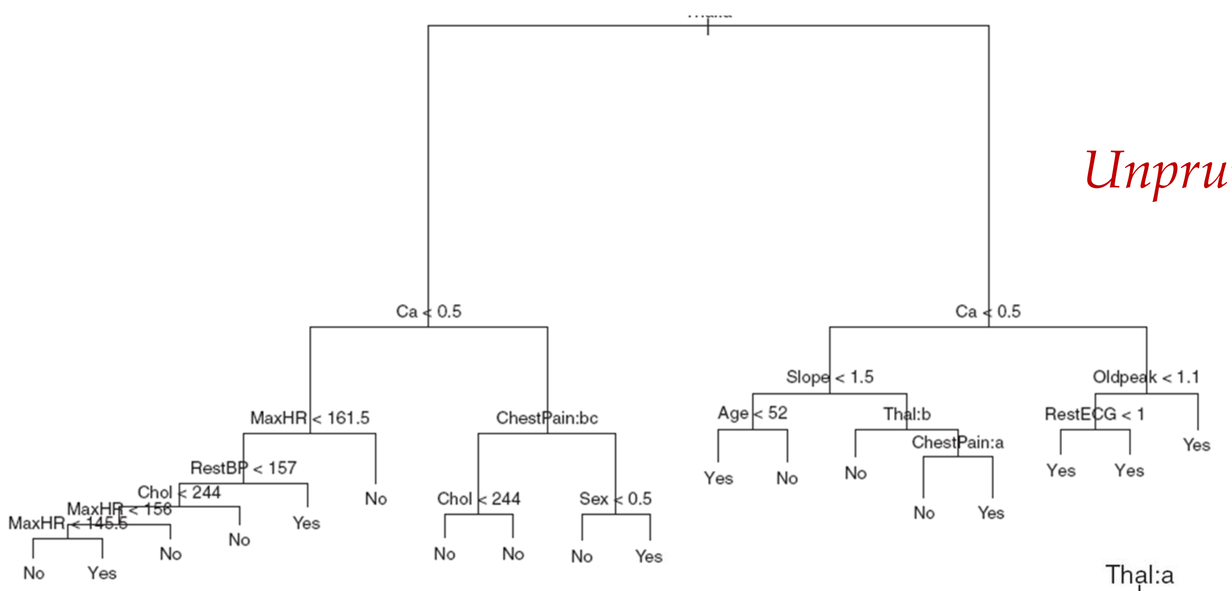
Information gain/Gini gain

We need to quantify the difference in Entropy or Gini index between *before* and *after* the split.

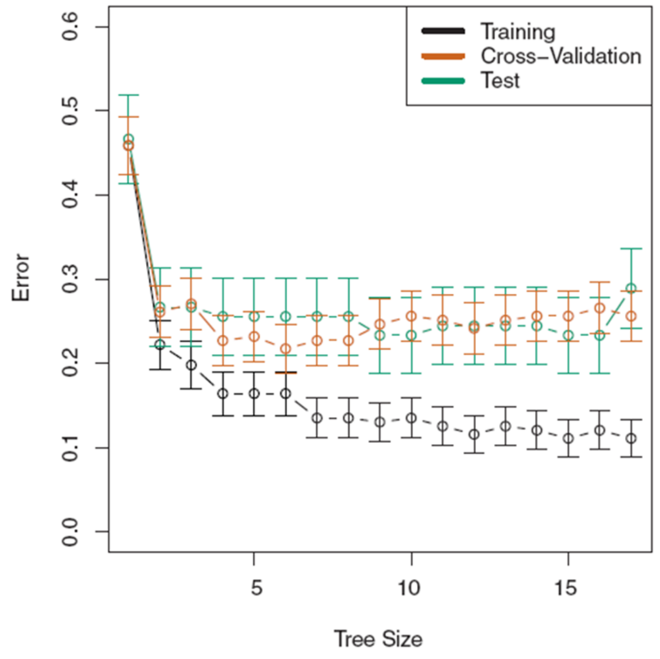
$$Gain(S, X_j) \equiv Entropy\ (S) - \sum_{v \in values(X_j)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where: S is a set of points and $values(X_j)$ is the set of possible values for X_j and S_v is the subset of S for which X_j has value v

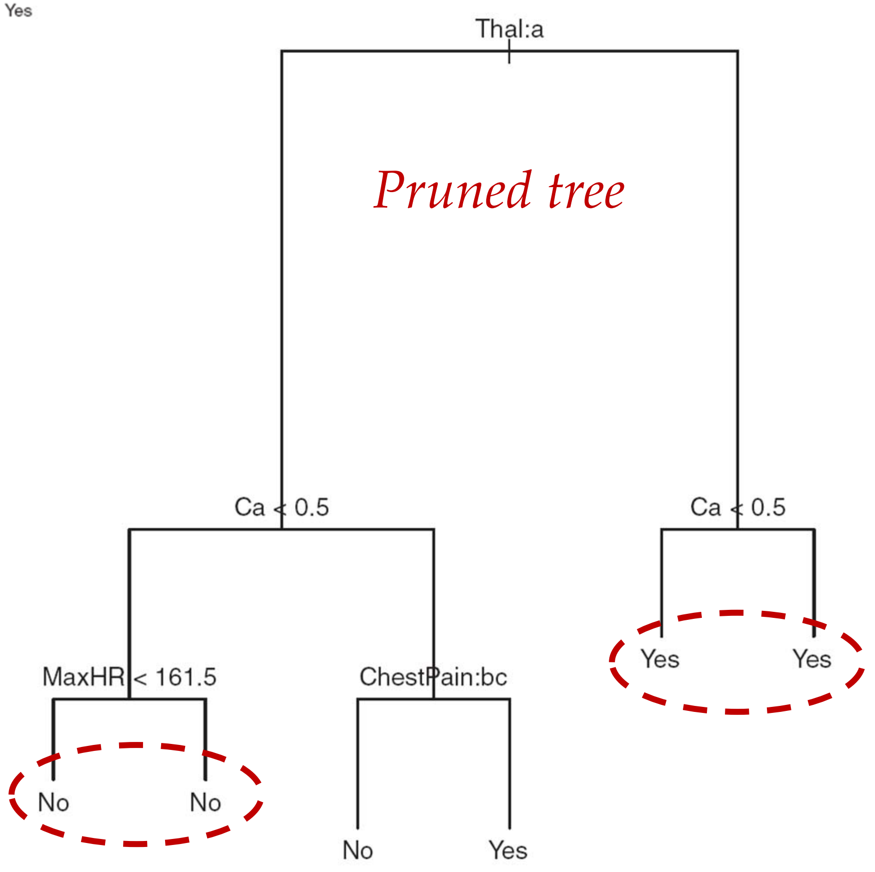
An equivalent formula can be written for the Gini index.



Unpruned tree



Pruned tree



Pruning vs stopping tree growth

- Small trees can be obtained also by preventing trees to grow too much (i.e. stopping the splitting early)
- Several criteria have been developed for deciding when to stop splitting – *these use the same ideas as those used for pruning*
- It is computationally cheaper than growing + pruning.

“In general, pruning is to be preferred over stopped training and cross-validation, since it takes advantage of more of the information in the training set; however, pruning large training sets can be computationally expensive. (DHS, 8.4.3)”