GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 7: TREE-BASED METHODS

December 15, 2014

DAT11-SF

LAST TIME:

- SVM

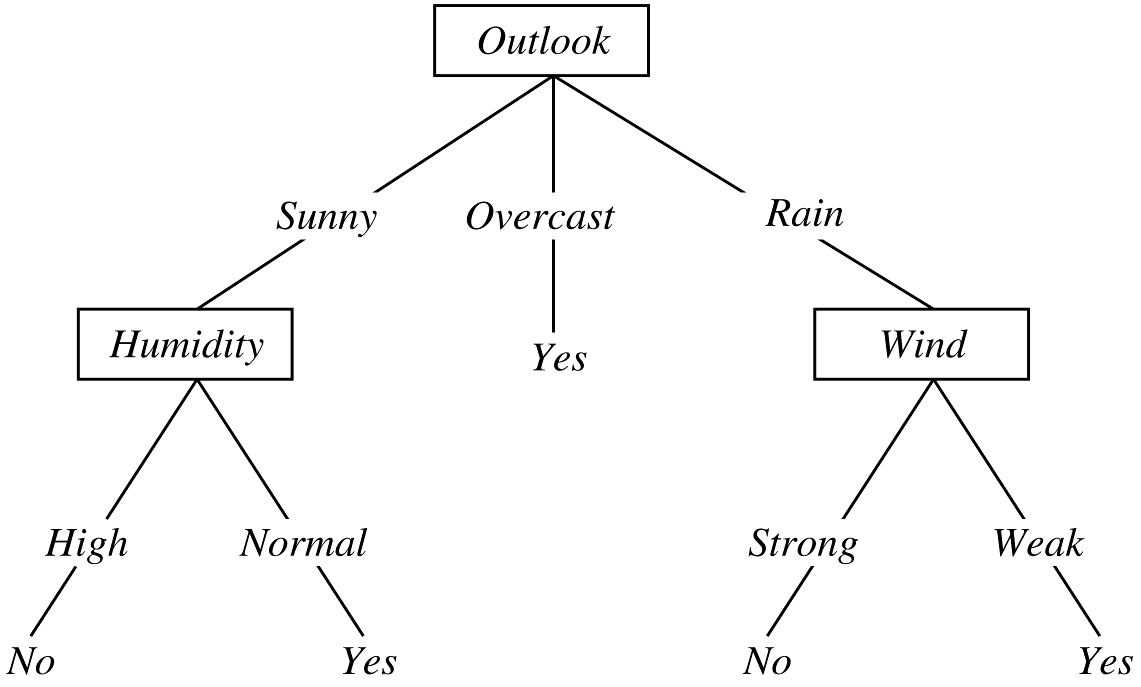
AGENDA

I. DECISION TREES

DECISION TREE LEARNING

- Decision tree learning is a method of approximating target function with a decision tree
- Tree is a representation of if-then rules
- Classification of instances done top-down the tree from the root to the leaf node

DECISION TREE



Classify an instance: <outlook=Sunny, temp = Hot, humidity=High, wind = Strong>

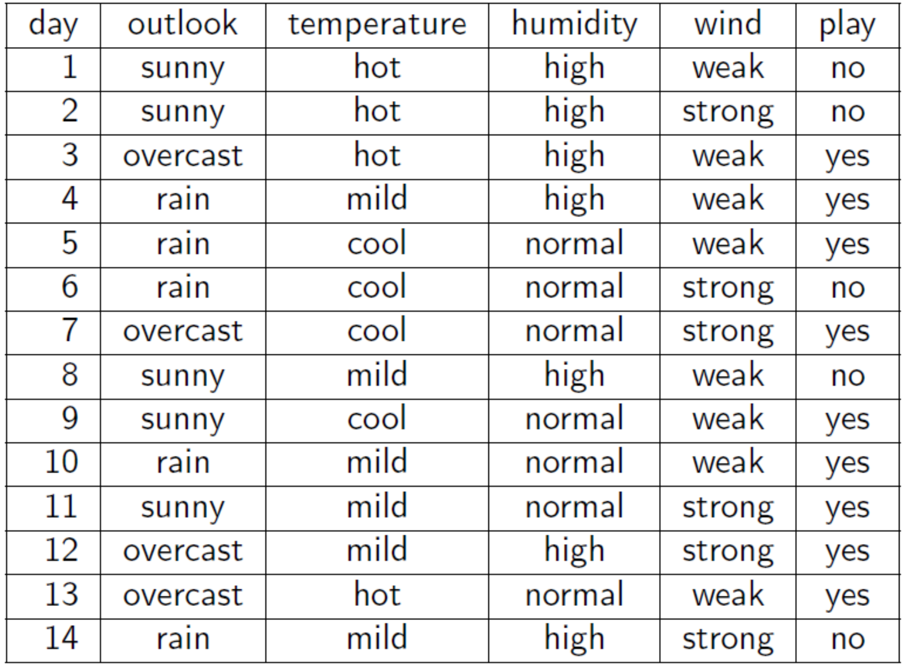
TYPES OF NODES

The top node of the tree is called the **root node**. This node has 0 incoming edges, and 2+ outgoing edges.

An **internal node** has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.

A **leaf node** has 1 incoming edge and, 0 outgoing edges. Leaf nodes correspond to *class labels*.

DECISION TREE



day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

DECISION TREE

- Instances are represented as attribute-value pairs. The best when each attributes takes on small number of possible values
- Target function has discrete output values (classification)
- Training data may contain errors and missing attribute values
- ID3, C4.5 algorithms (Quilan, 1986)

- Can be extended to continues (real-valued) attributes and regression (real valued) output function. CART trees (Breiman, Friedman, 1984)

II. CLASSIFICATION TREES

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical

Q: How do we find a practical solution that works?

A: Use a **heuristic** algorithm.

BUILDING A DECISION TREE

Build (or “grow”) a decision tree from the root

This is a **greedy recursive** algorithm that leads to a **local optimum**.

greedy – algorithm makes locally optimal decision at each step
recursive – splits task into subtasks, solves each the same way
local optimum – solution for a given neighborhood of points

BUILDING A DECISION TREE

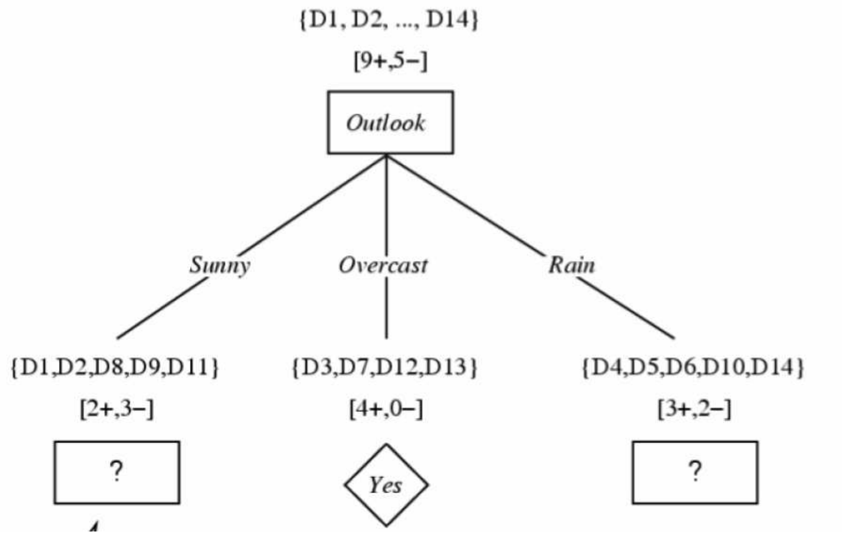
Algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called **purity**.

A partition is 100% pure when *all of its records belong to a single class*.

BUILDING A DECISION TREE

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

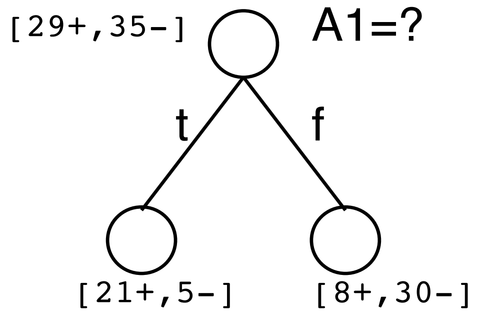
Therefore we want each step to create the partition with the highest possible purity.

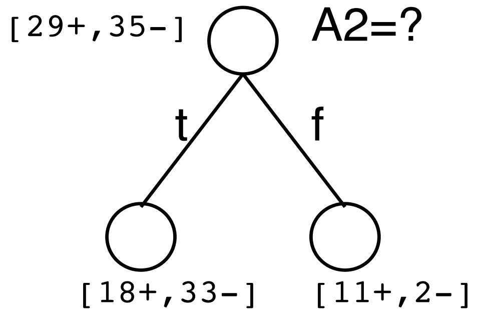
Need to define "node purity"

Need to build an objective function to measure the gain in purity from a particular split.

ATTRIBUTE SELECTION

Which attribute is best?





MEASURES OF IMPURITY

$$\begin{aligned}
\text{Entropy}(t) &= -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t), \\
\text{Gini}(t) &= 1 - \sum_{i=0}^{c-1} [p(i|t)]^2, \\
\text{Classification error}(t) &= 1 - \max_i [p(i|t)],
\end{aligned}$$

MEASURES OF IMPURITY

Binary class case: $p_+ = p$, $p_- = 1 - p$

Entropy: $I = -p \log_2 p - (1-p)\log_2(1-p)$

Gini impurity: $I = 2p(1-p)$

Classification error: $I = \mathrm{Min}(p, 1-p)$

CONSISTENCY OF OBJECTIVE FUNCTIONS

Note that each measure achieves its max at 0.5, min at 0 & 1.

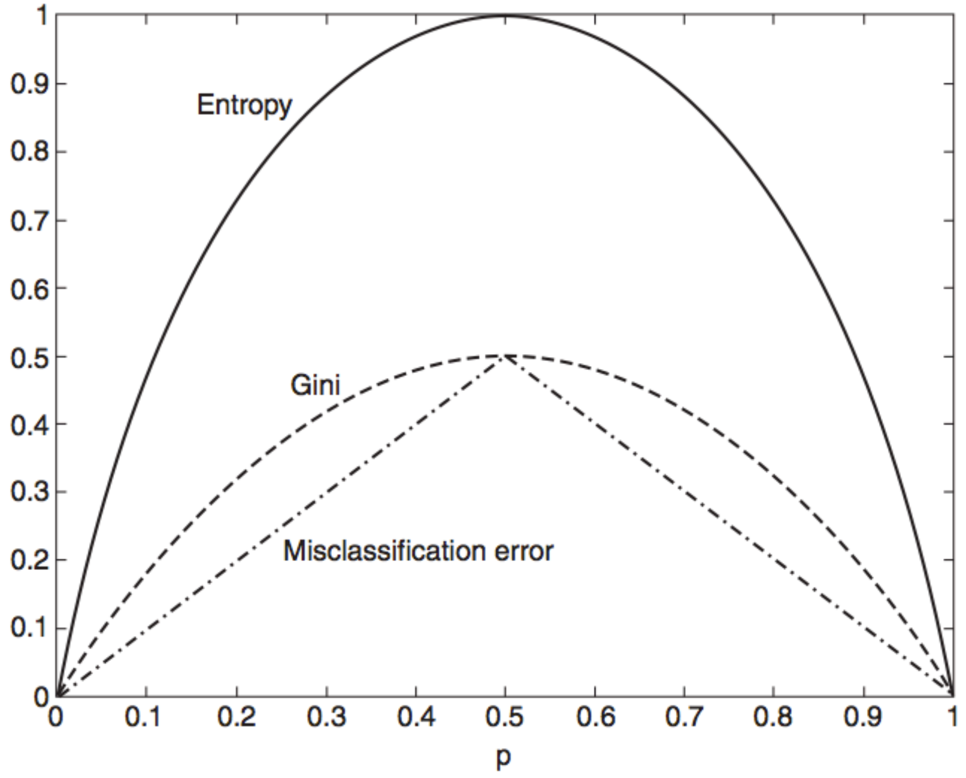


Figure 4.13. Comparison among the impurity measures for binary classification problems.

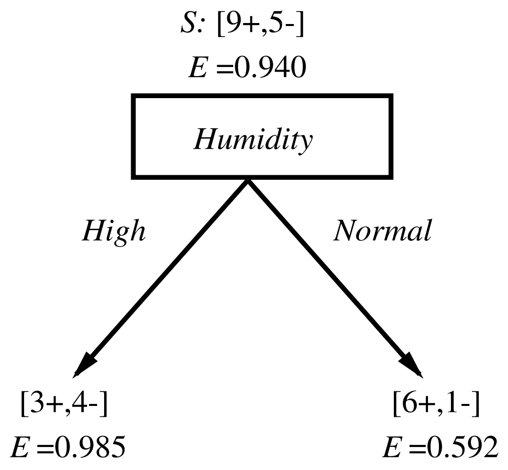
We still need to look at impurity before & after the split

We can make this comparison using the **gain (reduction in impurity)**:

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

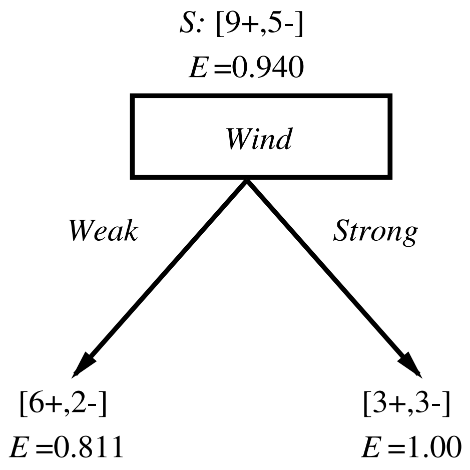
(Here I is the impurity measure, N_j denotes the number of records at child node j, and N denotes the number of records at the parent node.)

ATTRIBUTE SELECTION



Gain (S, Humidity)

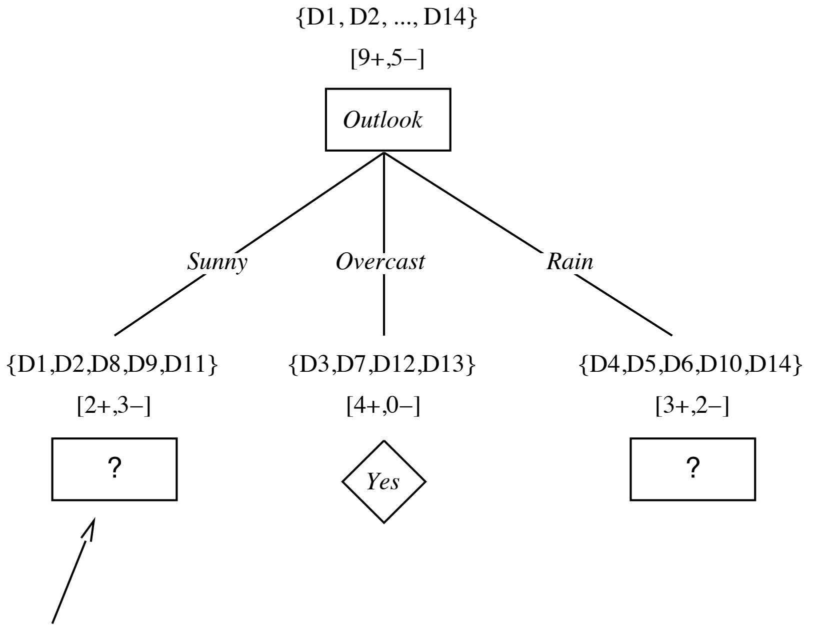
= .940 - (7/14).985 - (7/14).592
= .151



Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

ATTRIBUTE SELECTION



Which attribute should be tested here?

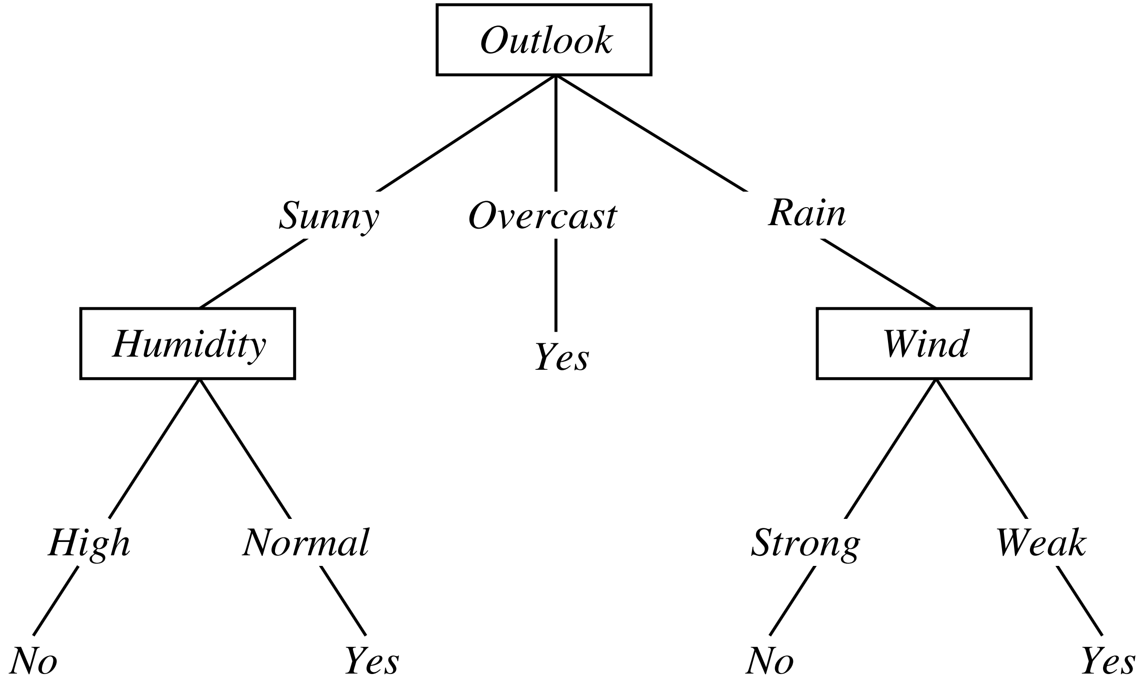
S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

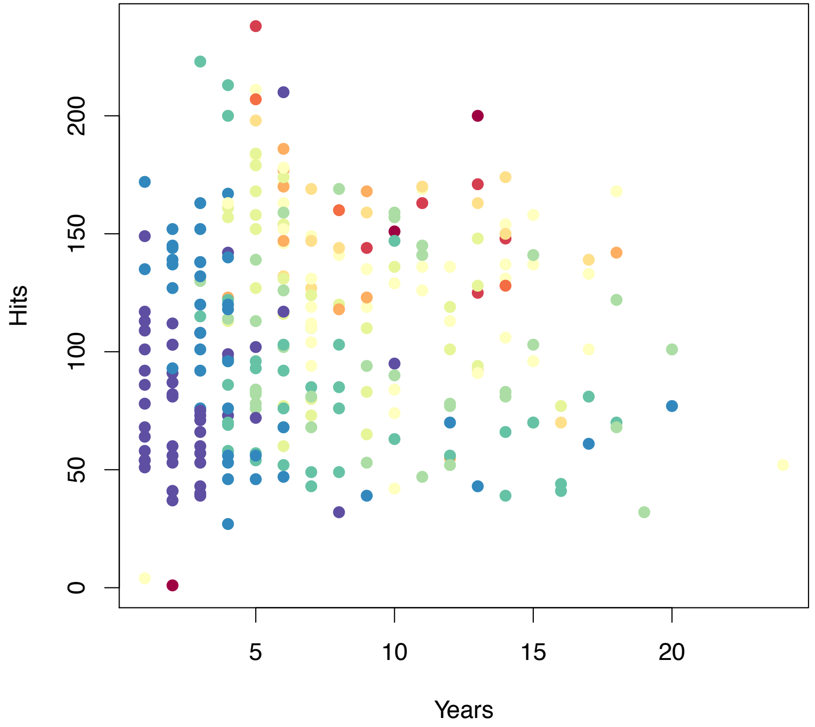
$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

ATTRIBUTE SELECTION



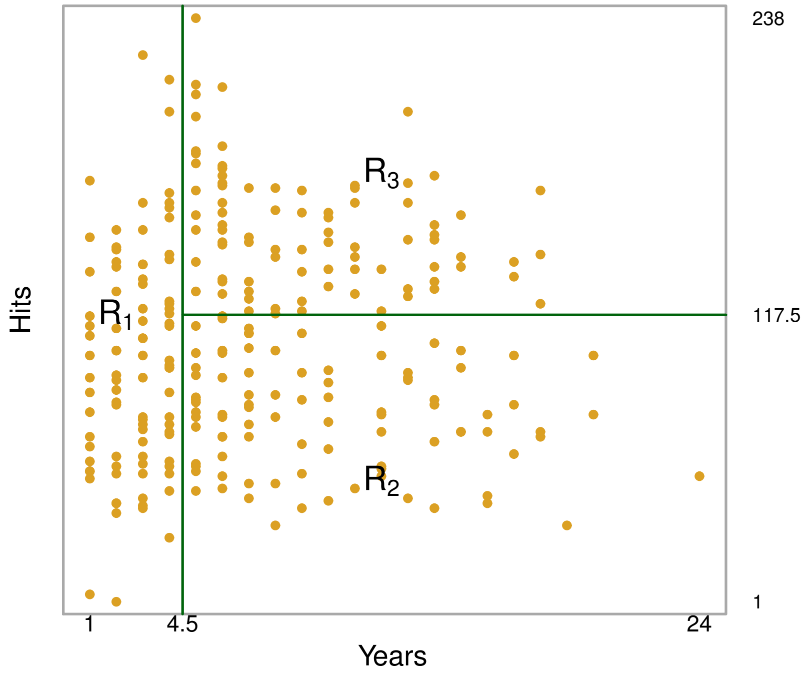
III. REGRESSION TREES

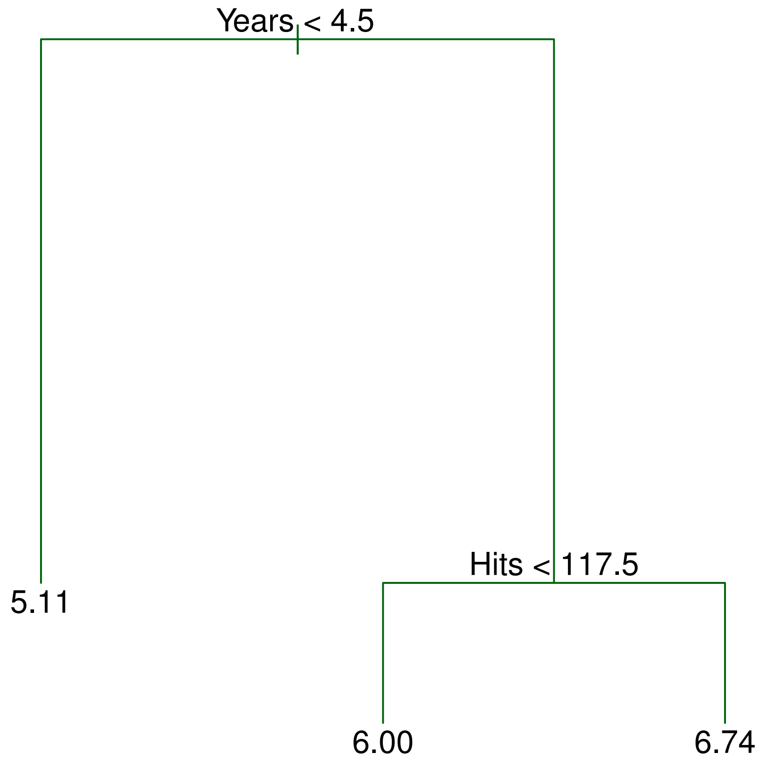
Real-valued data



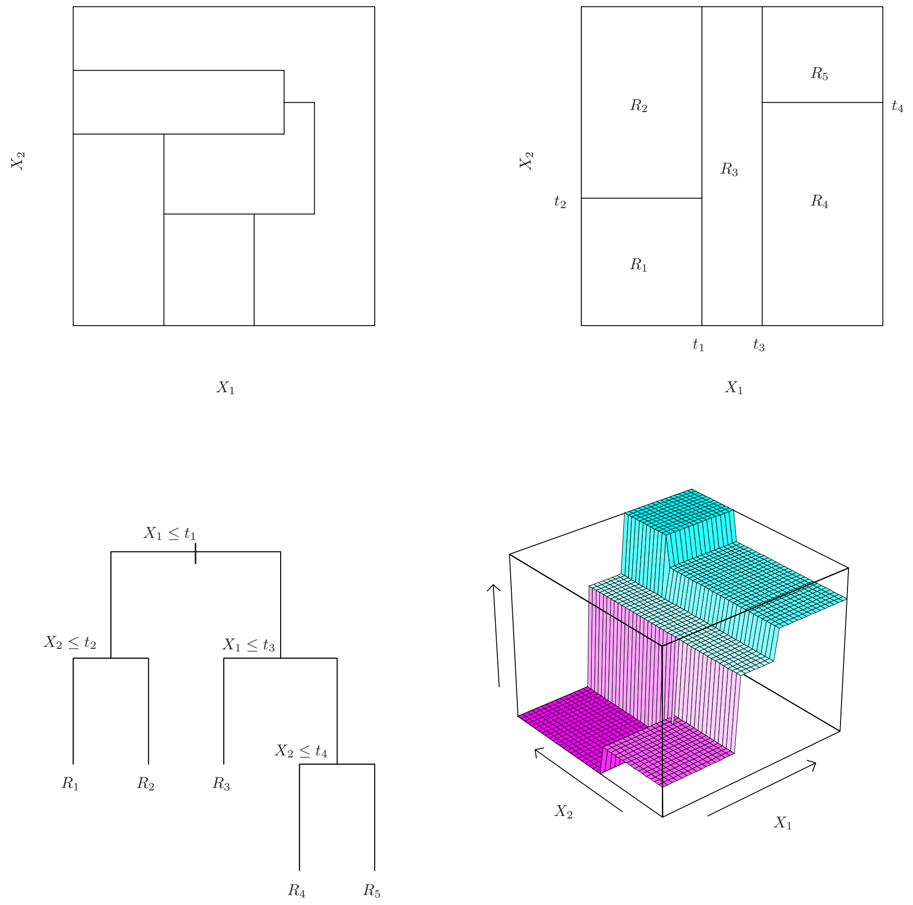
Salary is color-coded from low (blue, green) to high (yellow, red)

CART TREES


238
117.5
1
R1
R2
R3
Hits
1
4.5
24
Years


Years < 4.5
Hits < 117.5
5.11
6.00
6.74

CART TREES


X_2
X_1
R_2
R_3
R_5
R_4
R_1
t_1
t_2
t_3
t_4
$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_1
R_2
R_3
R_4
R_5

REGRESSION TREE

Split variable j into regions by split point s:

$$R_1(j, s) = \{X|X_j \le s\} \text{ and } R_2(j, s) = \{X|X_j > s\}.$$

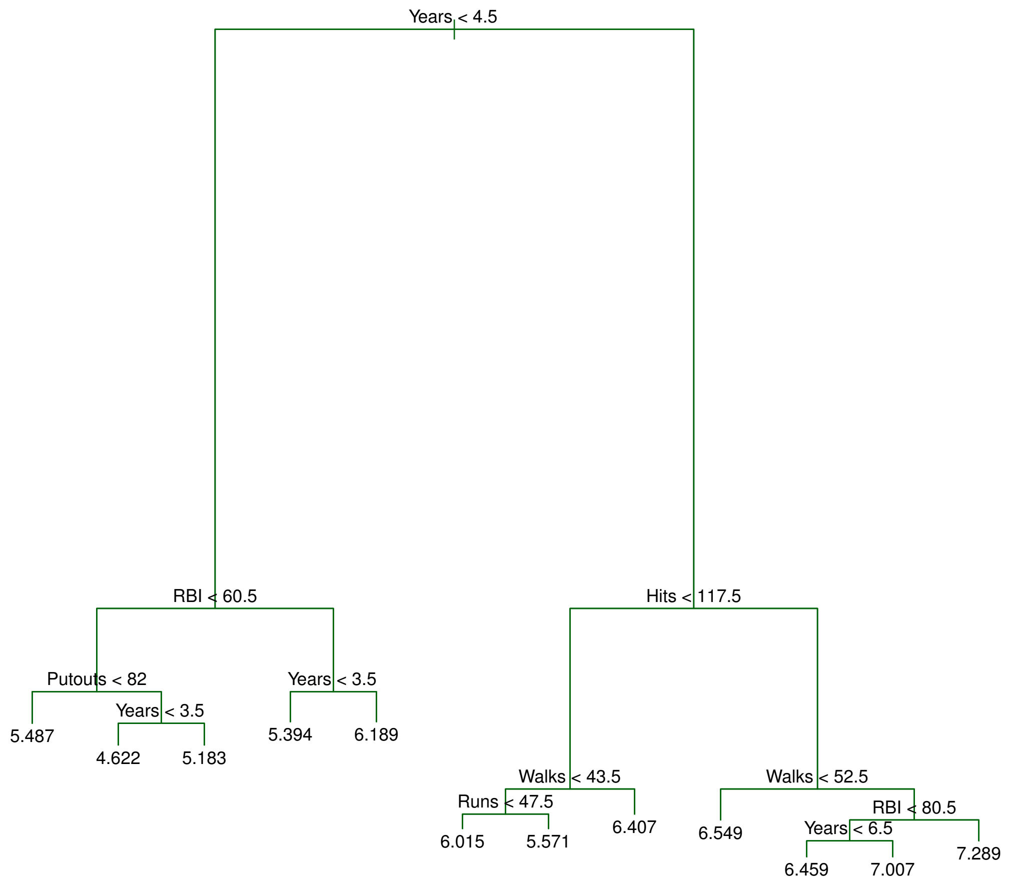
Constant value within the region:

$$\hat{c}_1 = \text{ave}(y_i|x_i \in R_1(j, s)) \text{ and } \hat{c}_2 = \text{ave}(y_i|x_i \in R_2(j, s)).$$

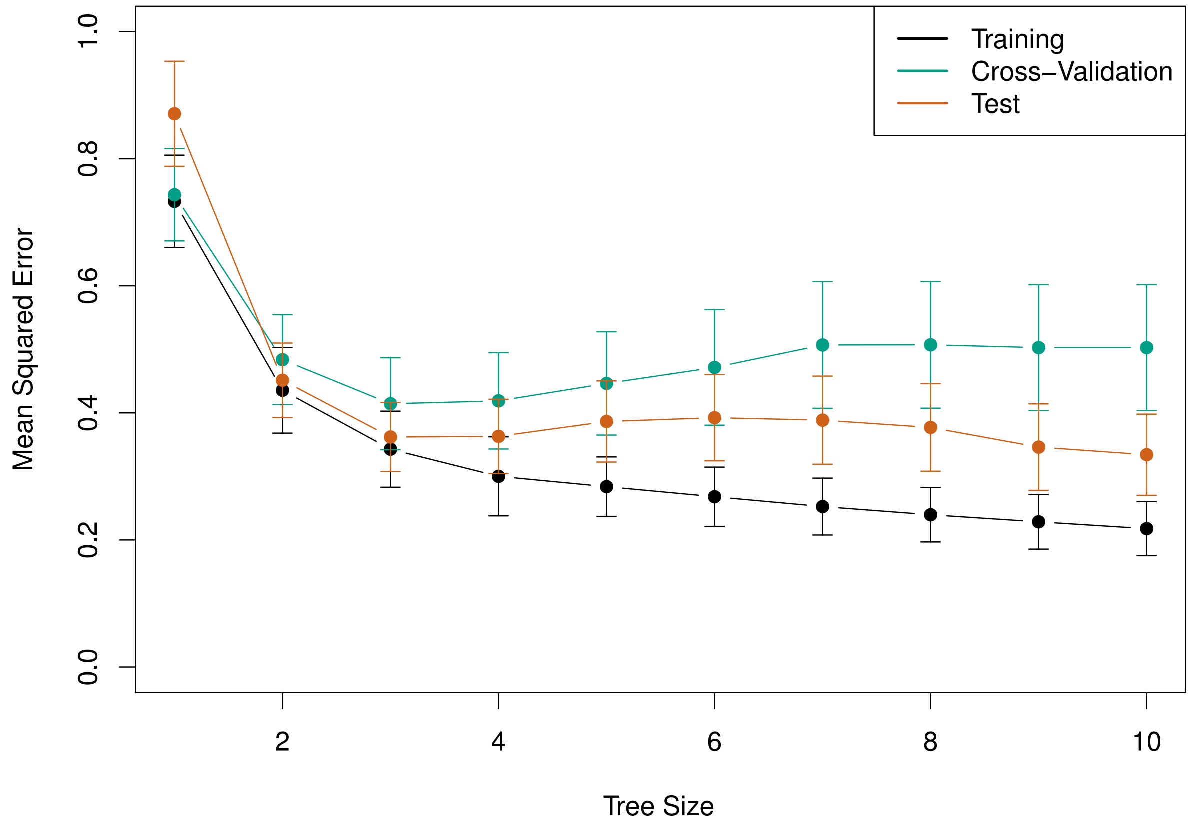
Optimization:

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

CART TREES


Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
5.487
4.622
5.183
5.394
6.189
Walks < 43.5
Walks < 52.5
Runs < 47.5
6.015
5.571
6.407
6.549
RBI < 80.5
Years < 6.5
6.459
7.007
7.289

OVERFITTING



Alternatively we could build the full tree, and then perform **pruning** as a post-processing step.

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

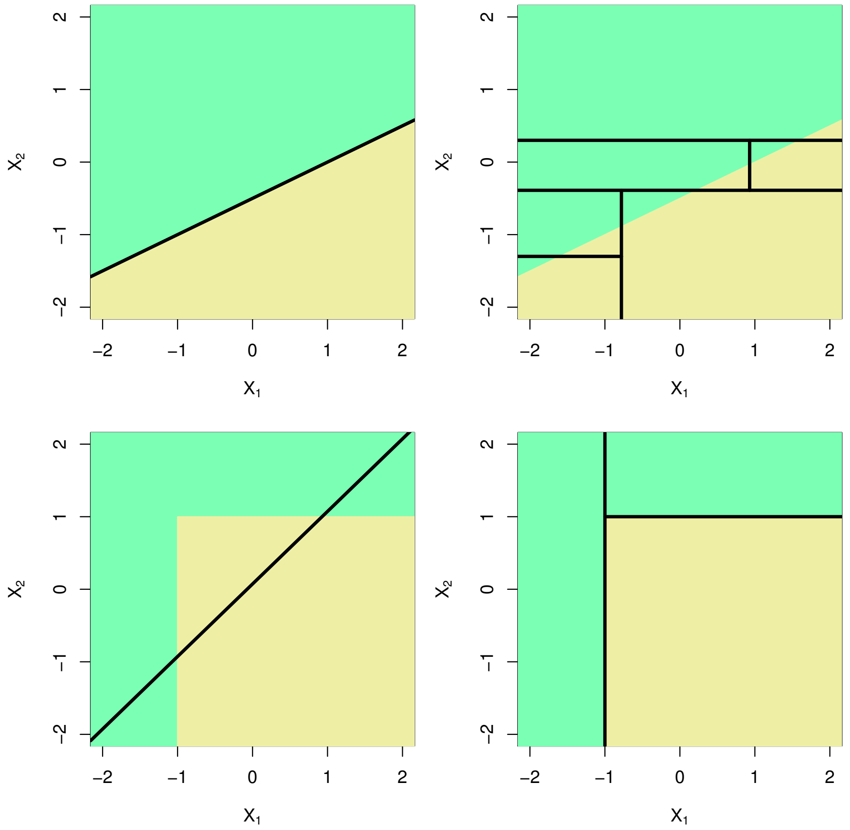
Complicated subtrees can be replaced either with a single node, or with a simpler (child) subtree.

Cost complexity prunning.

a subtree $T \subset T_0$ such that

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

TREES VS LINEAR MODELS



LAB: DECISION TREES IN SCIKIT-LEARN