hierarchical
set of rules

Decision Trees

Sourav Sen Gupta

CDS 2015 | PGDBA | 6 Oct 2015

Will you eat/wait? or leave

Deciding factors may be
~ heuristic rules

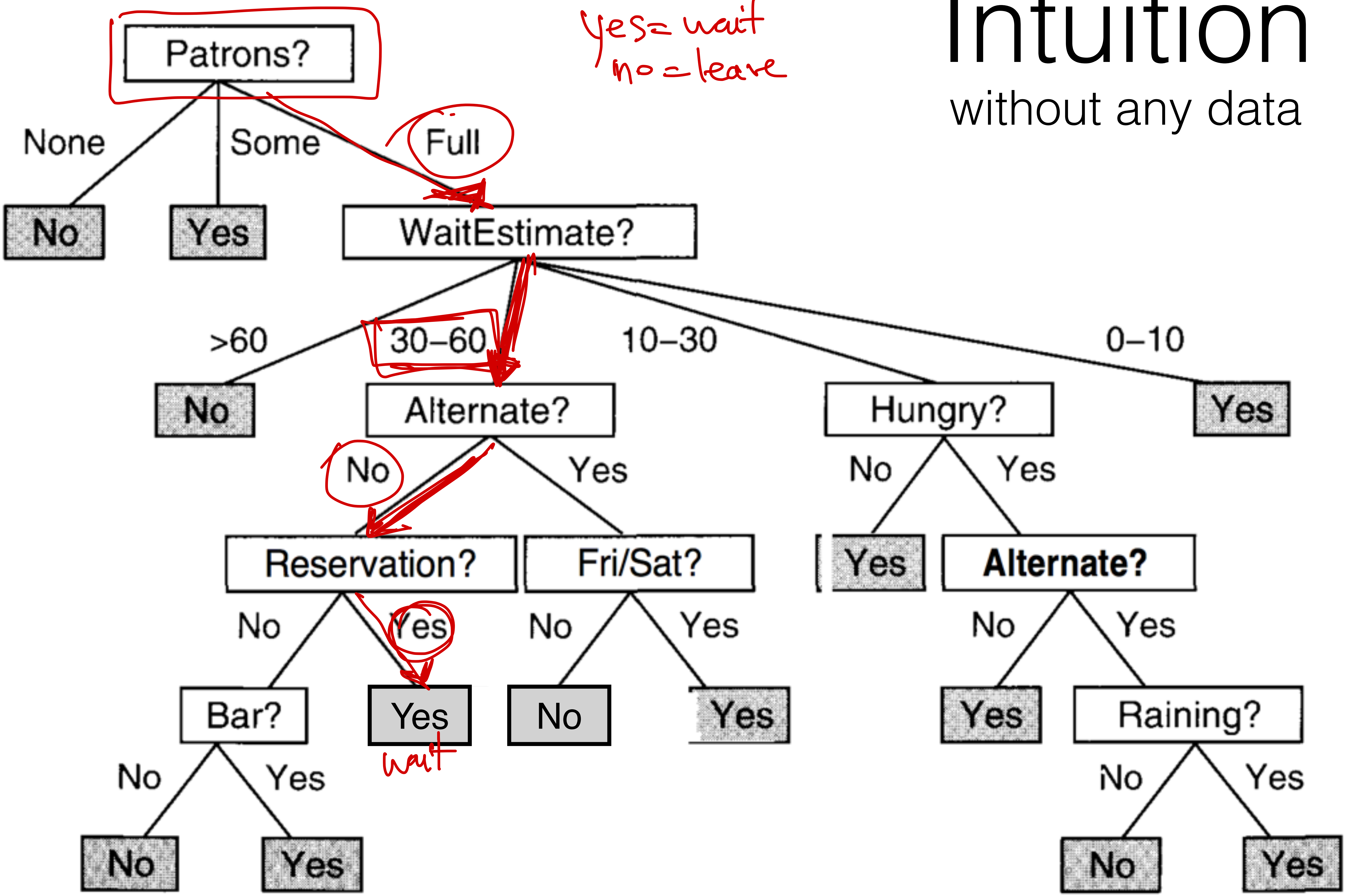
- If there are patrons (people inside) — Yes/No
- If you are hungry already — Yes / No
- Alternative options in the vicinity — Yes / No
- The estimated time for waiting — In minutes
- If you already have a reservation — Yes/No
- If it is a Friday/Saturday night — Yes/No
- If there is a Bar area to wait — Yes/No
- The range of price at the place — High/Medium/Low
- If it is raining at the time — Yes/No
- The genre of cuisine — French, Italian, Thai, Burger

Ref. — "Artificial Intelligence : A Modern Approach" — Stuart J. Russell and Peter Norvig

Intuition

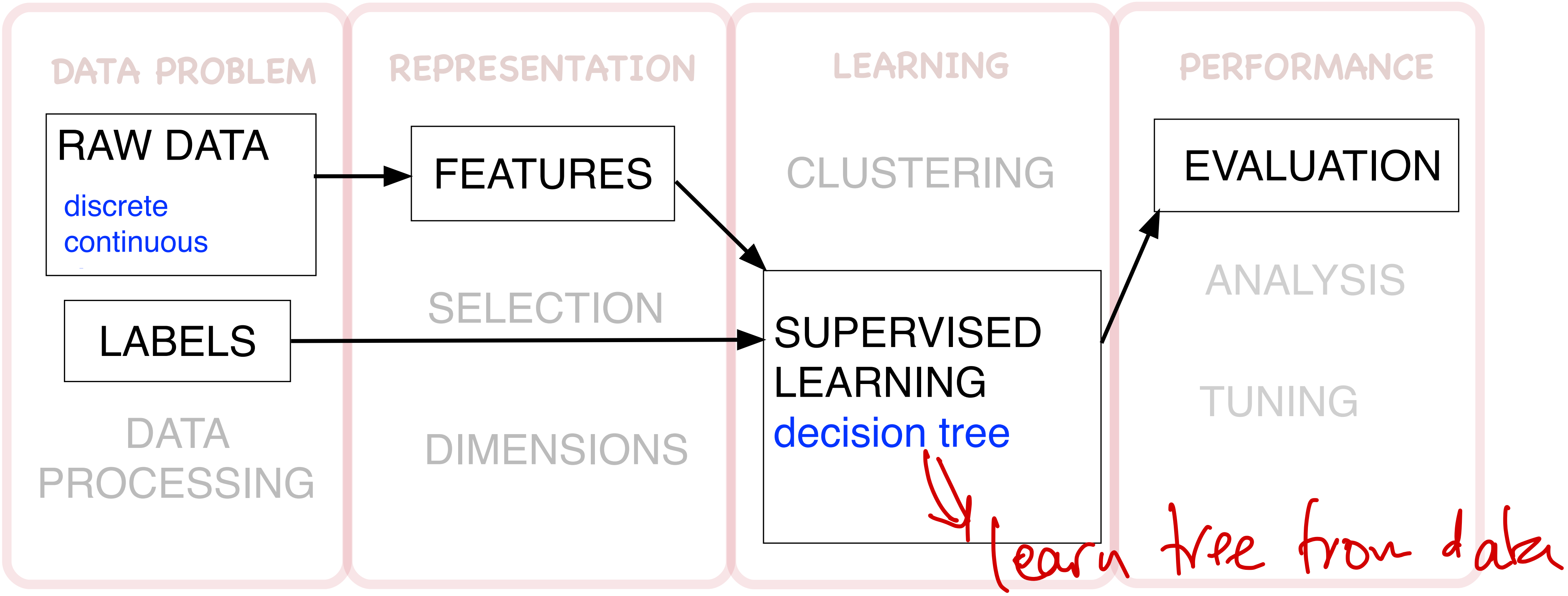
without any data

yes = wait
no = leave



Ref. — "Artificial Intelligence : A Modern Approach" — Stuart J. Russell and Peter Norvig

ML Pipeline


DATA PROBLEM
RAW DATA
discrete
continuous
LABELS
DATA PROCESSING
REPRESENTATION
FEATURES
SELECTION
DIMENSIONS
LEARNING
CLUSTERING
SUPERVISED LEARNING
decision tree
learn tree from data
PERFORMANCE
EVALUATION
ANALYSIS
TUNING

Training Data

Label

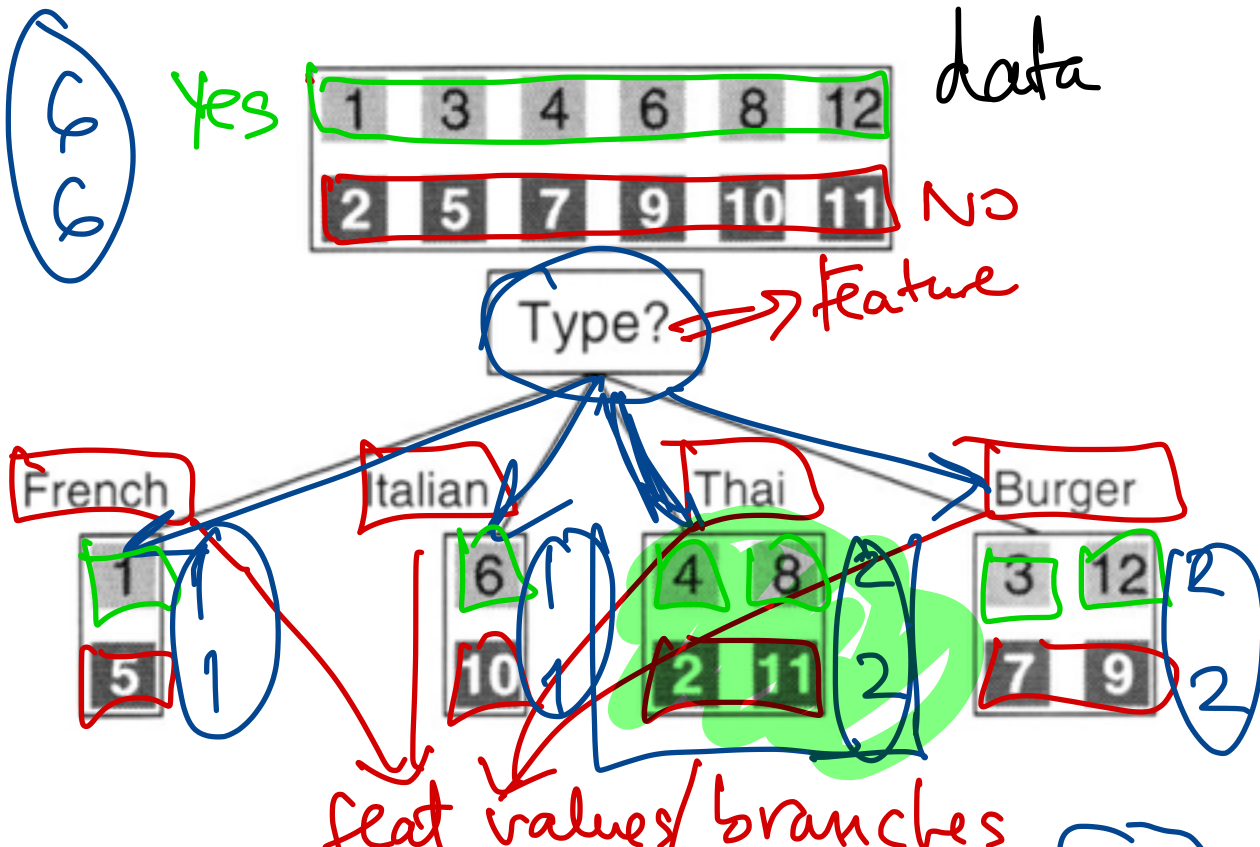
Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	*Yes*
TEST	**Yes**	**Yes**	**Yes**	**No**	**Full**	**\$\$\$**	**No**	**No**	**Thai**	**30-60**	**?**

N=12

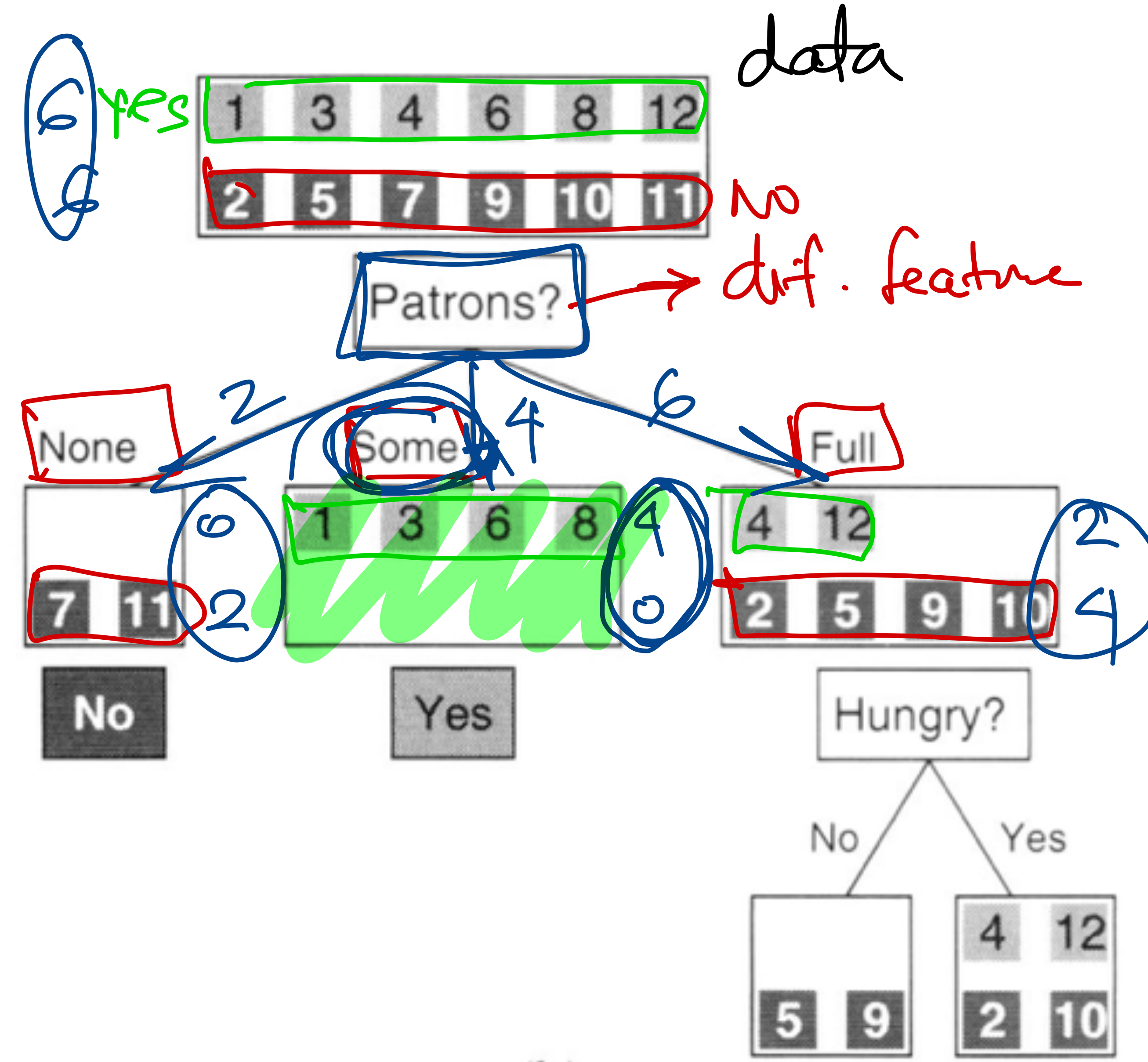
Choose First split at root

Example Split

option 1



option 2



Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Ref. — "Artificial Intelligence : A Modern Approach" — Stuart J. Russell and Peter Norvig

Entropy & Information Gain

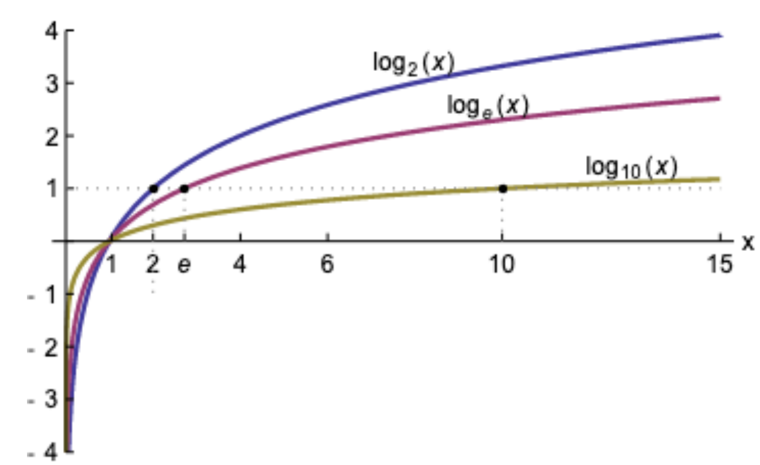
- Why a logarithm function?

$$log(p_1 \times p_2) = log(p_1) + log(p_2)$$

- **Shannon Entropy:**

$$H(p_1, \ldots, p_N) = -\sum_{i=1}^{N} p_i \,.\, log\left(p_i\right)$$

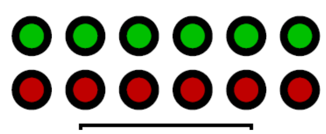
= measures randomness



Issue: increasing number of events shrinks the probability.
Solution: use **logarithm of probability** instead and take **the average.**

How do we construct the tree ?
i.e., how to pick attribute (nodes)?

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



For a training set containing p positive examples and n negative examples, we have:

$$H\left(\frac{p}{\boxed{p+n}}, \frac{n}{p+n}\right) = -\overset{\text{prob}(p)}{\frac{p}{p+n}} \log_2 \overset{\text{prob}(p)}{\frac{p}{p+n}} - \overset{\text{prob}(n)}{\frac{n}{p+n}} \log_2 \overset{\text{prob}(n)}{\frac{n}{p+n}}$$

~~Information Gain~~

Reduction in Entropy

~~Information Gain~~ = Parent Entropy — E(Child Entropy)

$$I = \underbrace{H(\mathcal{S})}_{\text{entropy of labels before split}} - \underbrace{\sum_{i \in \{\mathrm{L,R}\}} \boxed{\frac{|\mathcal{S}^i|}{|\mathcal{S}|}} \boxed{H(\mathcal{S}^i)}}_{\text{entropy (of labels) after the split}}$$

weight i

entropy for branch i

One notion of entropy is that of Shannon Entropy

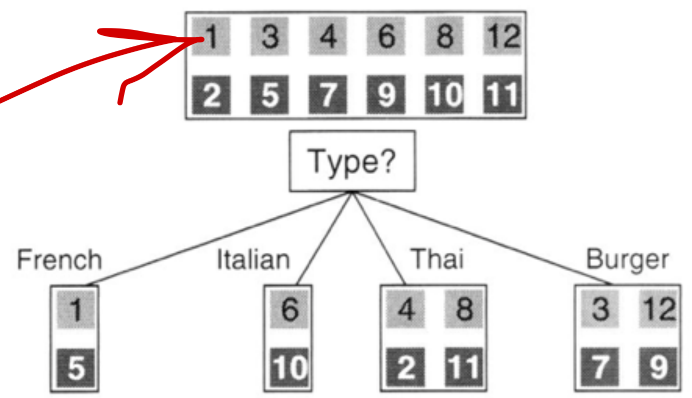
$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c))$$

Ref. — "Decision Forests" — Antonio Criminisi, Jamie Shotton, and Ender Konukoglu

Compare Gain

= reduction in entropy

$H_{parent} - \text{weight } H_{child}$



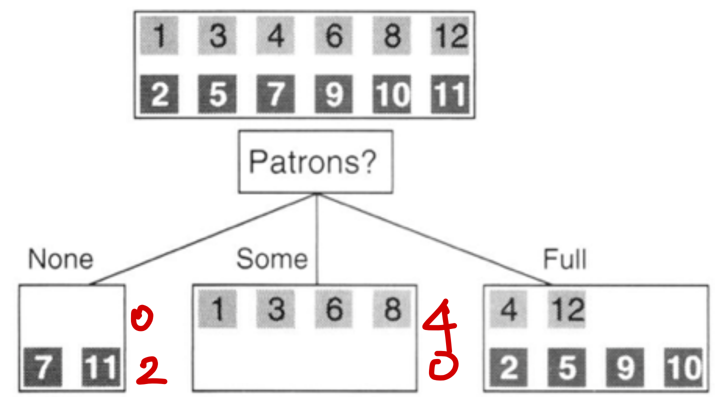
French	Italian	Thai	Burger
p = 1/6	p = 1/6	p = 1/3	p = 1/3
H = 1	H = 1	H = 1	H = 1

Parent Entropy

$$H = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

E(Child Entropy)

$$H = \frac{1}{6}\cdot 1 + \frac{1}{6}\cdot 1 + \frac{1}{3}\cdot 1 + \frac{1}{3}\cdot 1$$



None	Some	Full
p = 1/6	p = 1/3	p = 1/2
H = 0	H = 0	H = 0.918 $= -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}$

no randomness
100% one label

Parent Entropy

$$H = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

E(Child Entropy)

$$H = \frac{1}{6}\cdot 0 + \frac{1}{3}\cdot 0 + \frac{1}{2}\cdot 0.918$$

How to pick nodes?

- A chosen attribute A, with K distinct values, divides the training set E into subsets $E_1, \ldots, E_K$.
- The **Expected Entropy (EH) remaining** after trying attribute A (with branches $i=1,2,\ldots,K$) is

points in child i

$$EH(A) = \sum_{i=1}^{K} \frac{p_i + n_i}{\boxed{p+n}} H\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- **Information gain (I)** or **reduction in entropy** for this attribute is:

$$I(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(A)$$

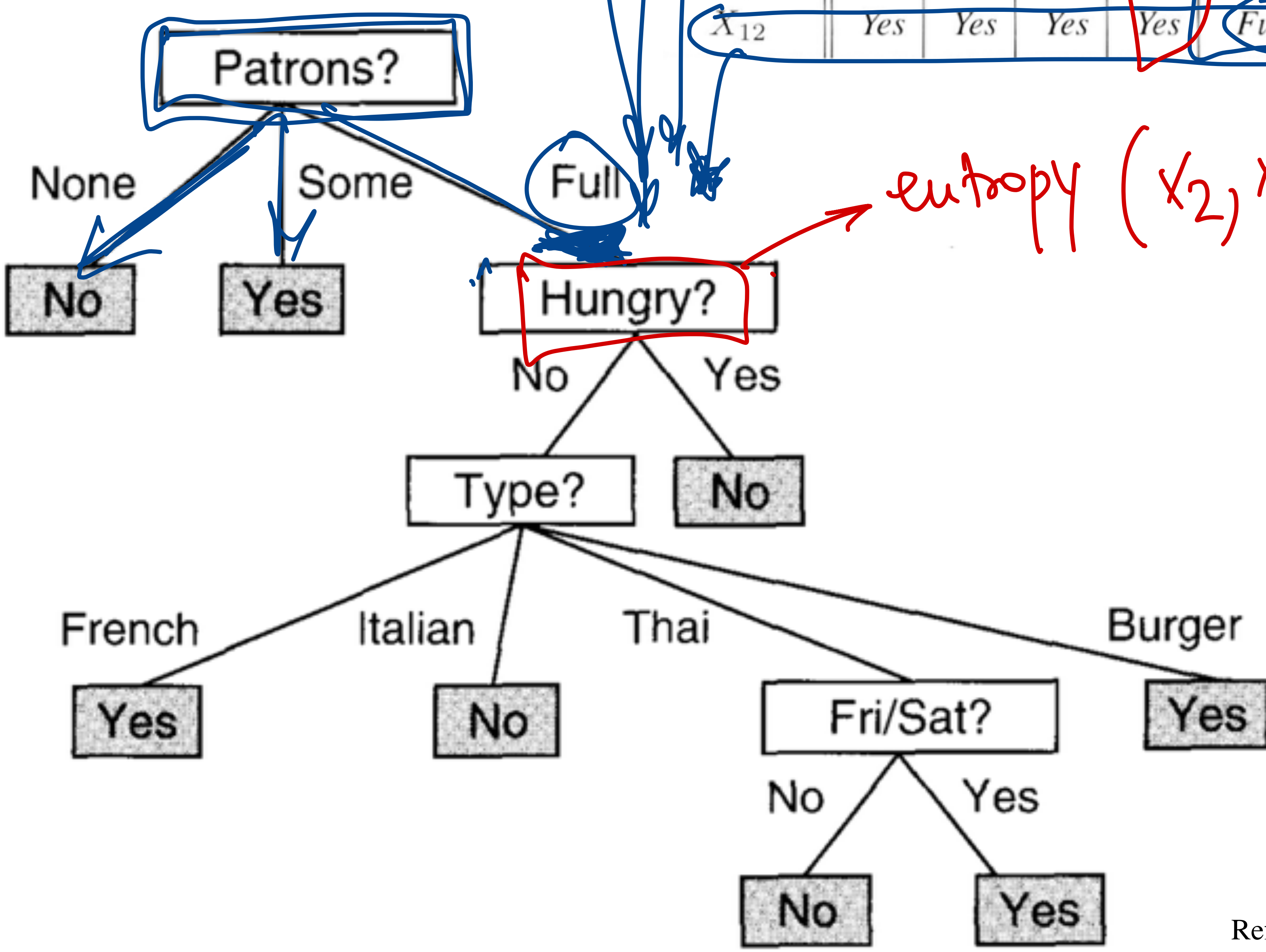
= **Entropy** in the **parent node -** remaining **Expected Entropy** in the **child nodes**

[Hwee Tou Ng & Stuart Russell]

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	Yes
X_2	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	No
X_3	No	Yes	No	No	Some	$	No	No	Burger	0–10	Yes
X_4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	Yes
X_5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	Yes
X_7	No	Yes	No	No	None	$	Yes	No	Burger	0–10	No
X_8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	Yes
X_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	No
X_{11}	No	No	No	No	None	$	No	No	Thai	0–10	No
X_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	Yes

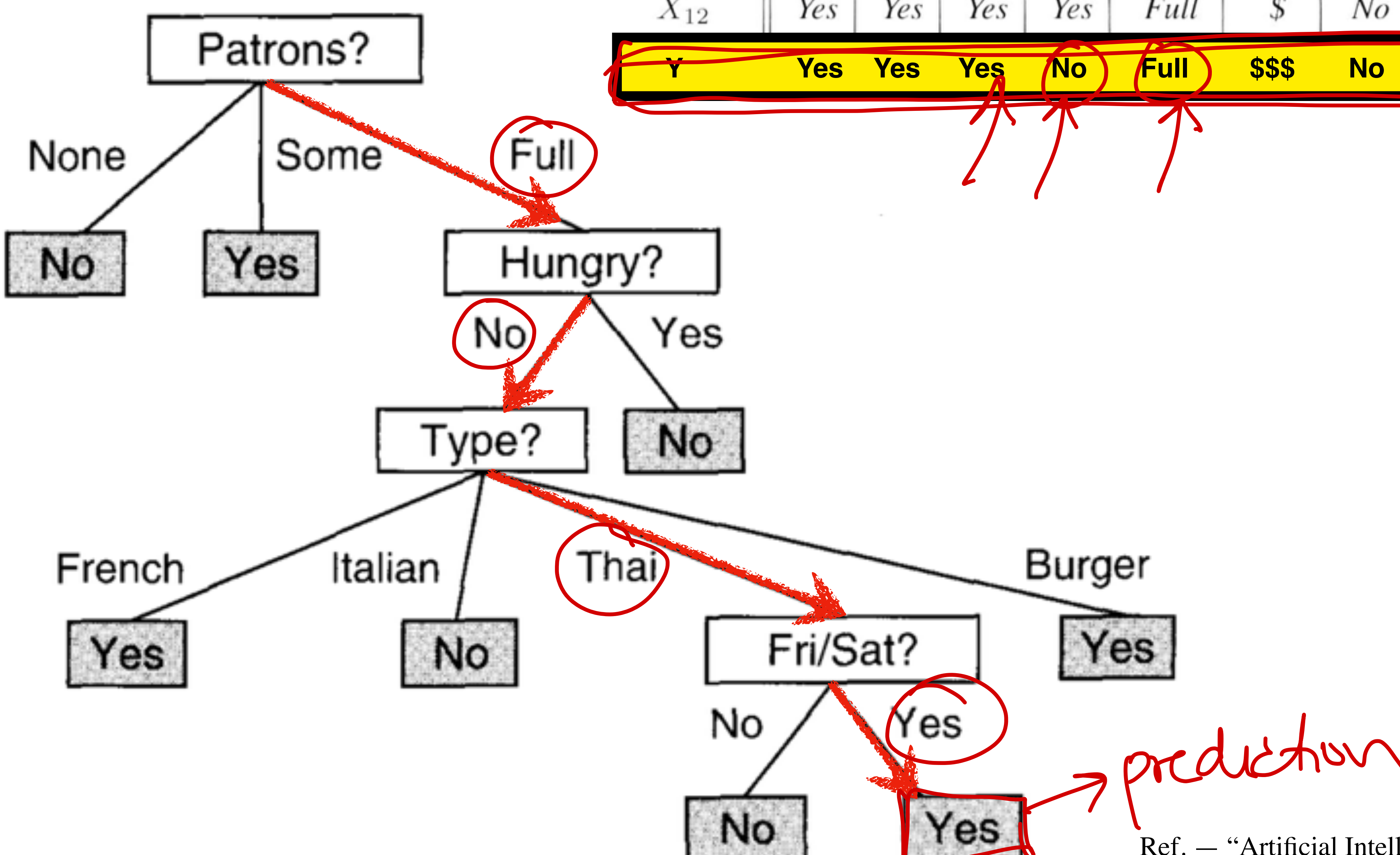
data at this node.

entropy $(X_2, X_4, X_5, X_9, X_{10}, X_{12})$



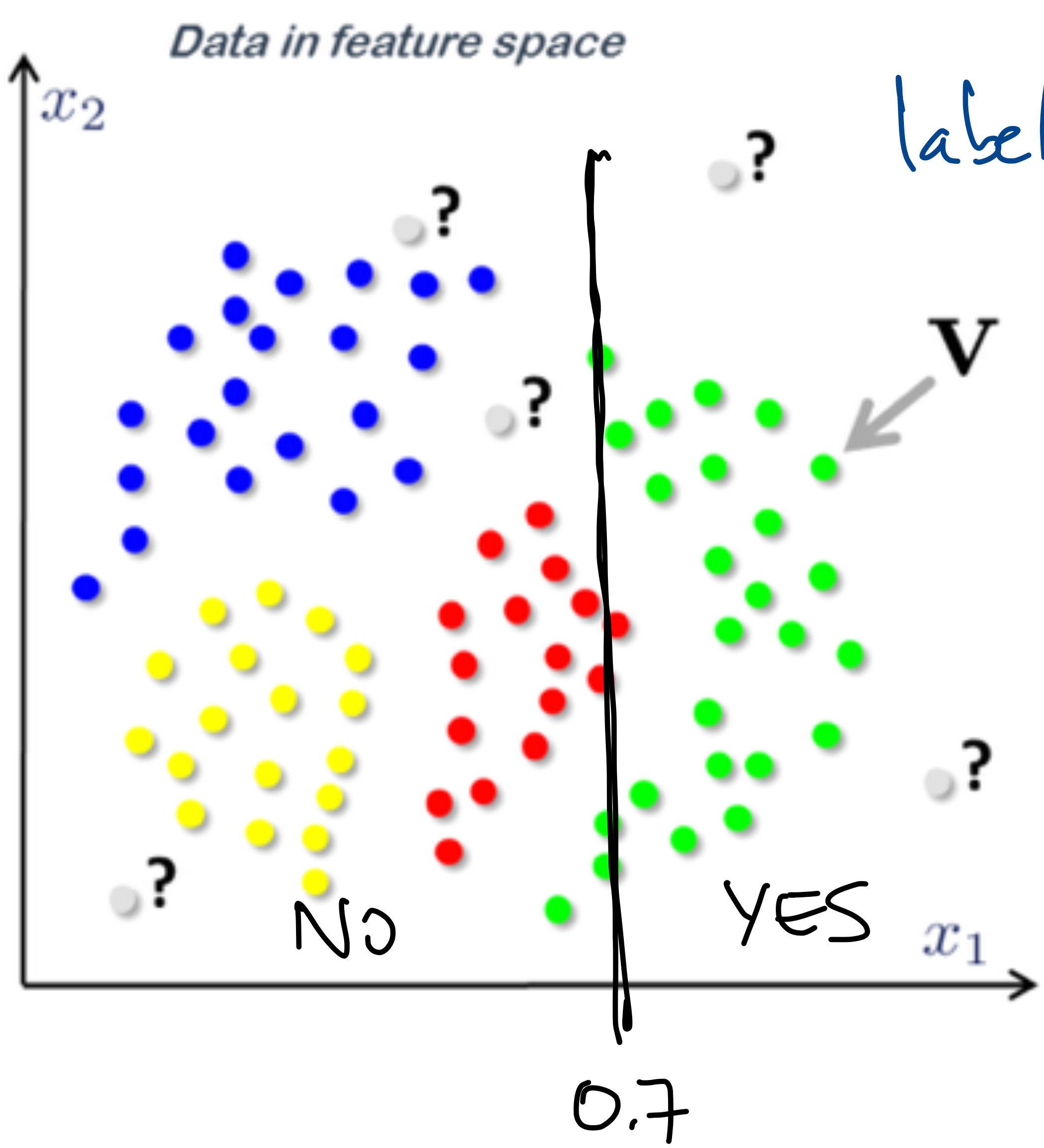
Ref. — "Artificial Intelligence" — Stuart J. Russell and Peter Norvig

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*
Y	**Yes**	**Yes**	**Yes**	**No**	**Full**	**$$$**	**No**	**No**	**Thai**	**30-60**	**?**



Ref. — “Artificial Intelligence” — Stuart J. Russell and Peter Norvig

Classification Tree



labels = colors ⟹ 4 labels B, Y, R, G

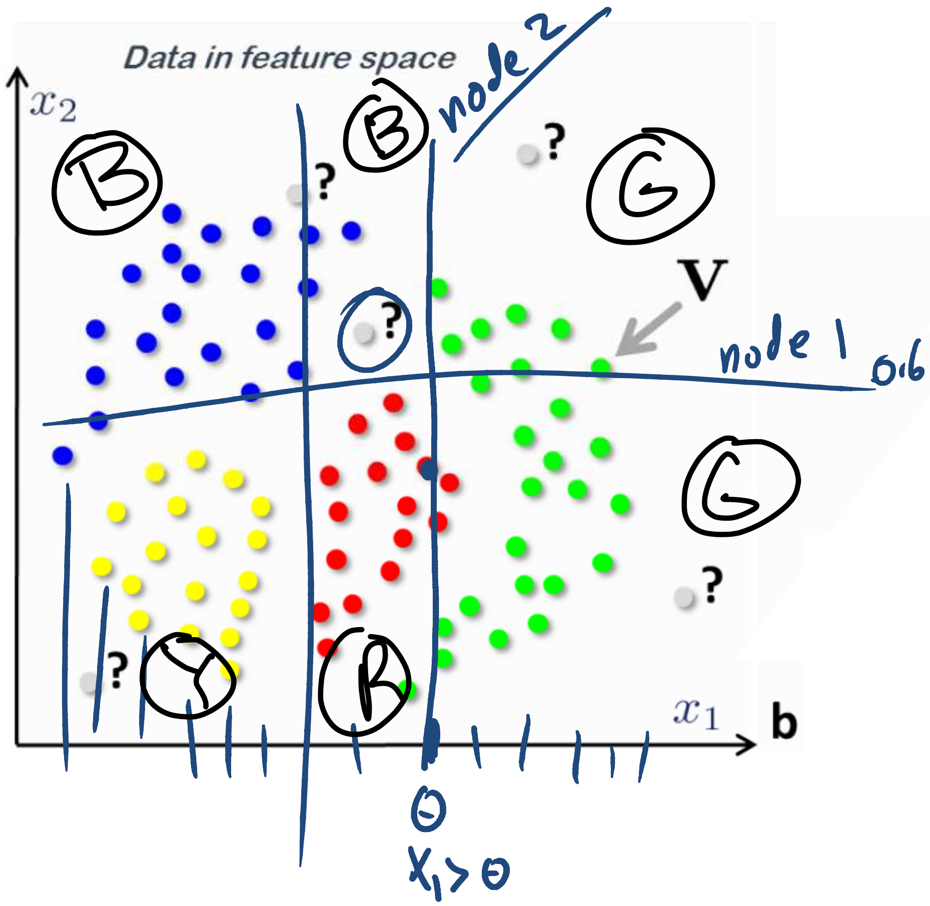
splits = thresholding by coordinat

ex. ($x_1 > 0.7$?) → yes / → no

Ref. — "Decision Forests" — Antonio Criminisi, Jamie Shotton, and Ender Konukoglu

Classification tree

3 splits
⇓
6 regions



- How to deal with ***continuous features***?
  - Create the splits **randomly**
  - Compute **information gain** for each split
  - Choose the one with **maximum gain**

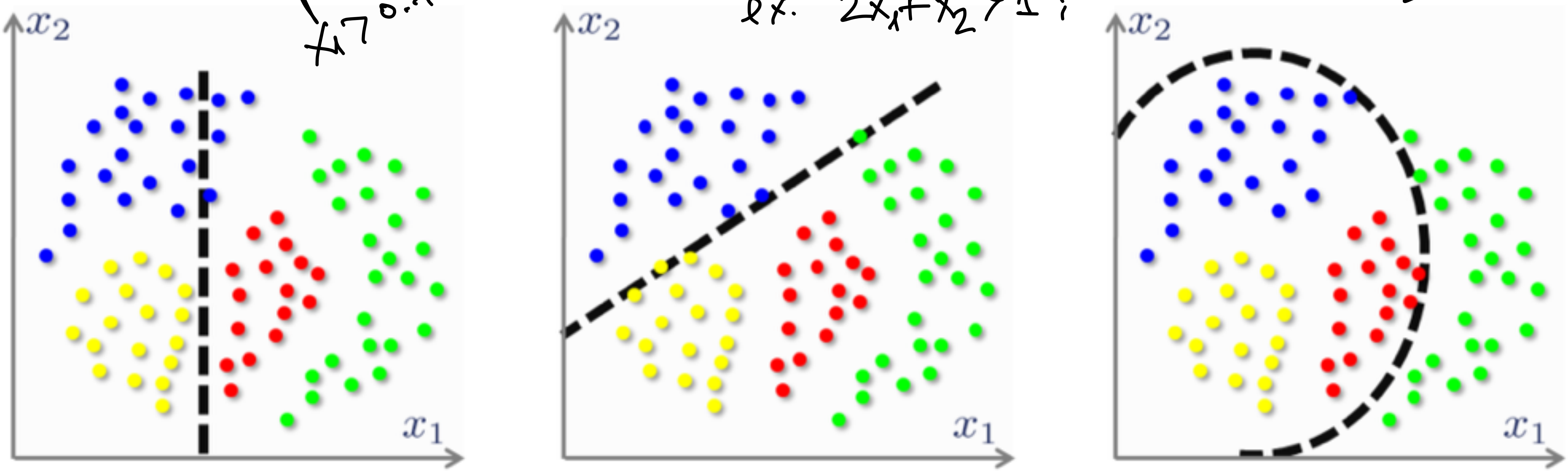
A generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, \cdots, x_d)$

Split Types

threshold split
$x_1 > 0.7$?

linear split
ex. $2x_1 + x_2 > 1$?

Kernel split
$x_1 x_2 + x_2^3 - x_1^2 < 0.2$?



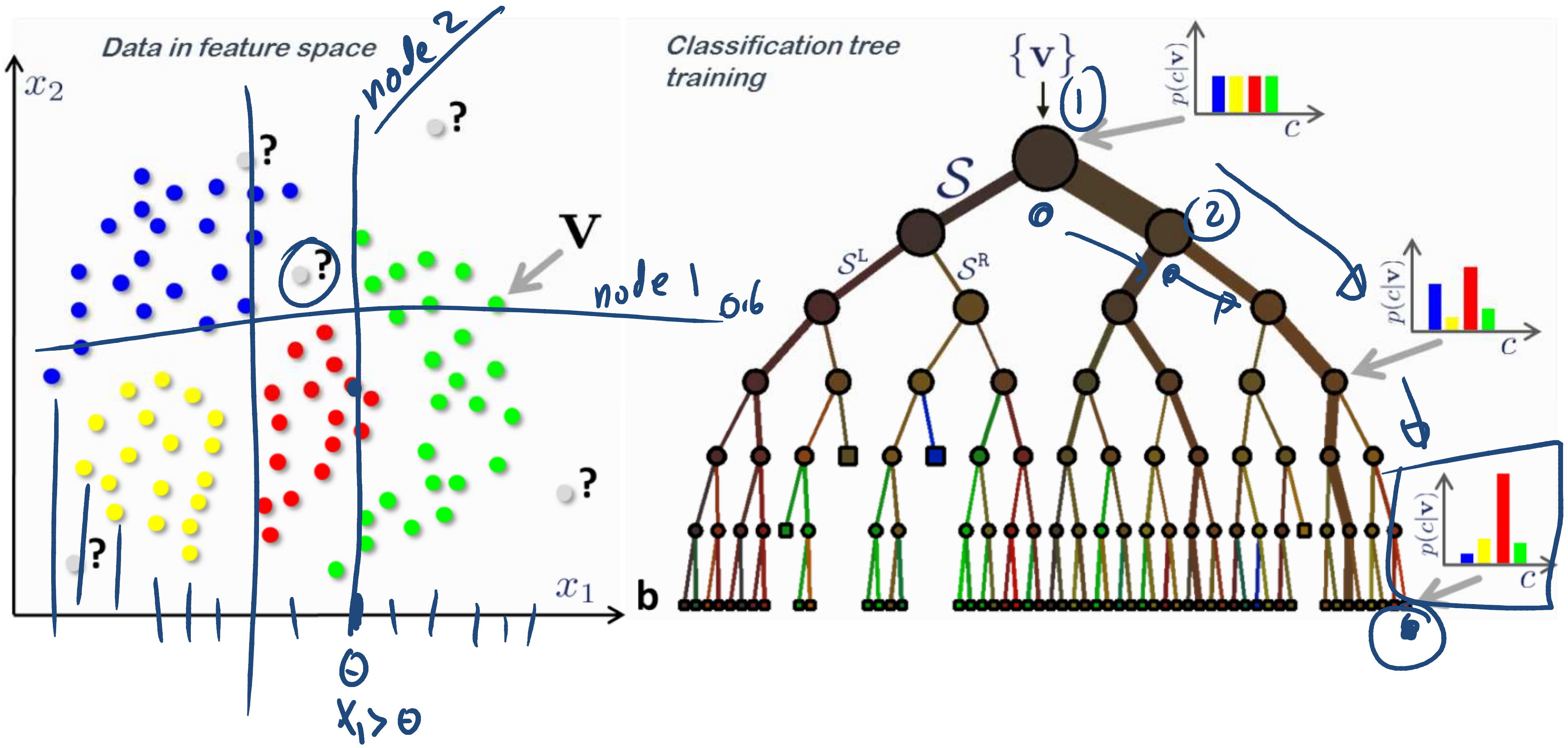
(a) (b) (c)

default

Axis-aligned Hyperplane | General oriented Hyperplane | Quadratic/Conic in 2D

Ref. — "Decision Forests" — Antonio Criminisi, Jamie Shotton, and Ender Konukoglu

Classification tree



A generic data point is denoted by a vector $\mathbf{v} = (x_1, x_2, \cdots, x_d)$

$$\mathcal{S}_j = \mathcal{S}_j^{\mathrm{L}} \cup \mathcal{S}_j^{\mathrm{R}}$$

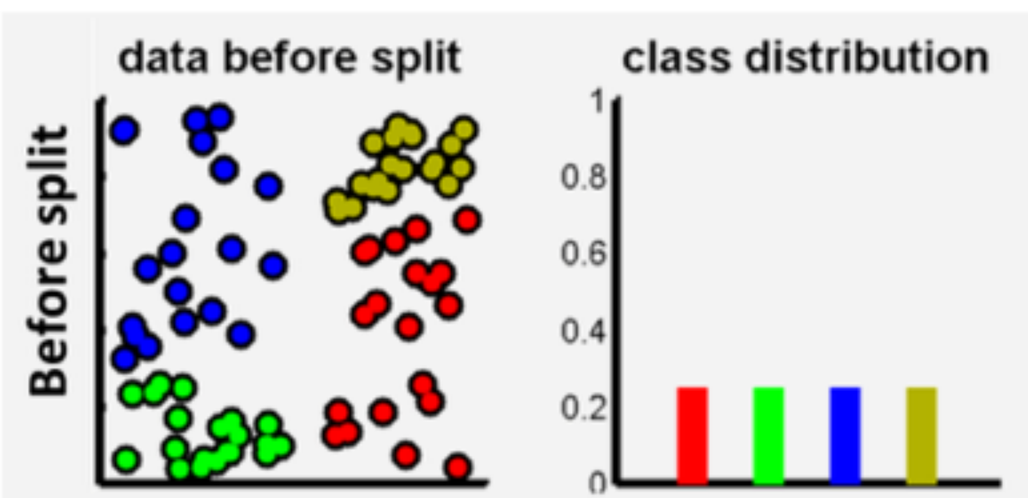
[Criminisi et al, 2011]

- Note that the **histogram** shows the **posterior distribution** for each class:

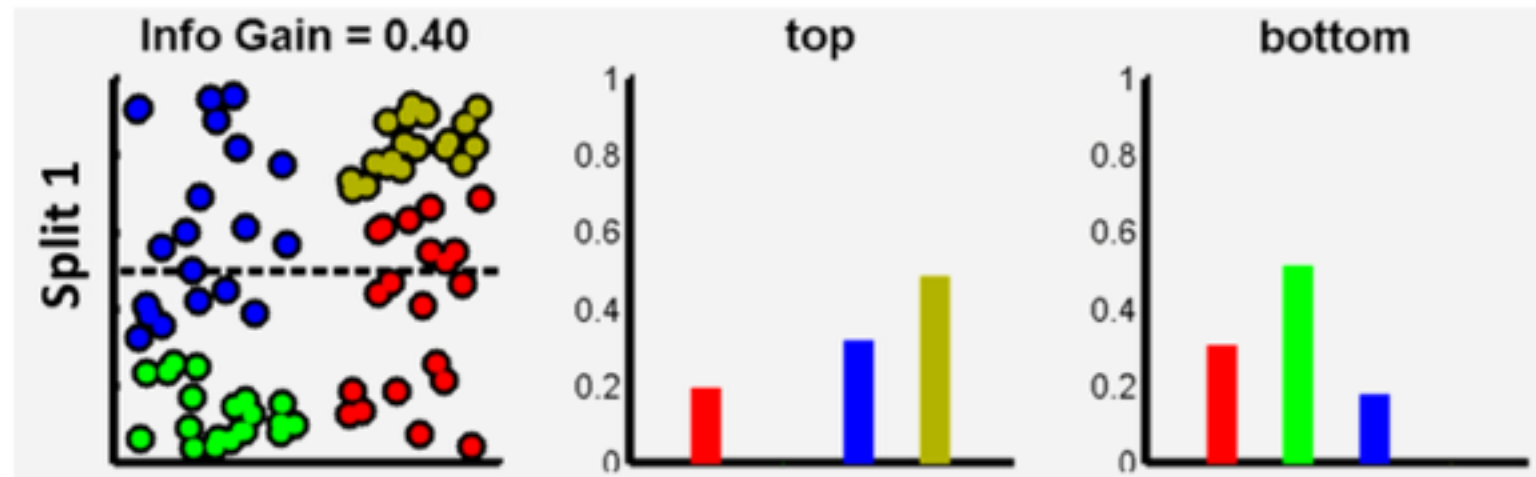
$$p(Class \,|\, Data)$$

Choosing Split

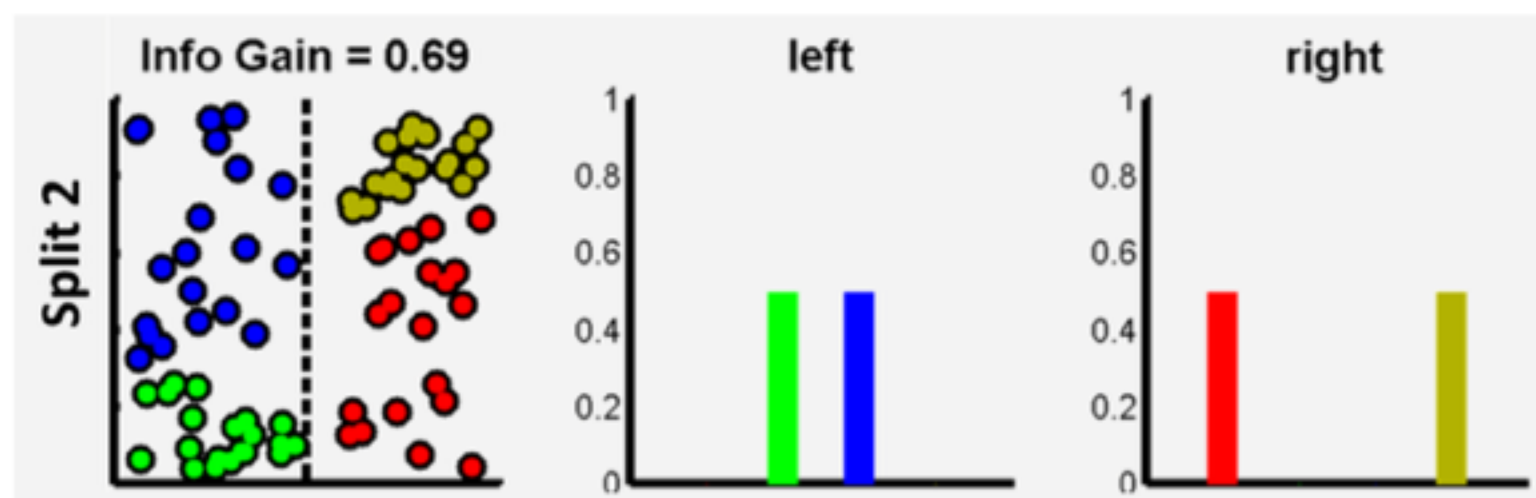
$$\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}_j \in \mathcal{T}_j} I_j$$



(a)



(b)



(c)

Ref. — “Decision Forests” — Antonio Criminisi, Jamie Shotton, and Ender Konukoglu

Expressiveness of decision trees

The tree on previous slide is a Boolean decision tree:

- ✔ the decision is a binary variable (true, false), and
- ✔ the attributes are discrete.
- ✔ It returns ally iff the input attributes satisfy one of the paths leading to an ally leaf:

$$ally \Leftrightarrow (neck = tie \wedge smile = yes) \vee (neck = \neg tie \wedge body = triangle),$$

  i.e. in general
  - ✘ $Goal \Leftrightarrow (Path_1 \vee Path_2 \vee \ldots)$, where
  - ✘ $Path$ is a conjuction of attribute-value tests, i.e.
  - ✘ the tree is equivalent to a DNF of a function.

Any function in propositional logic can be expressed as a dec. tree.

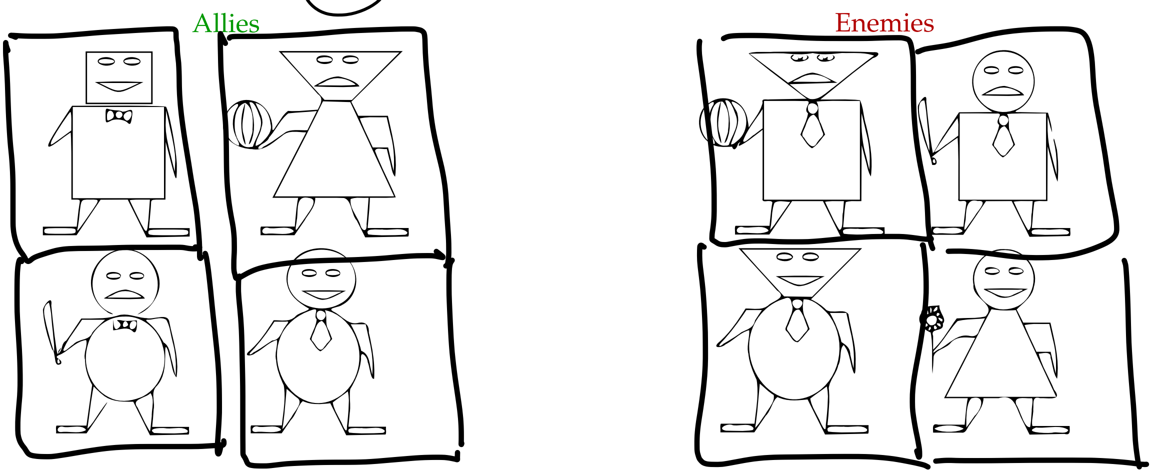
- ✔ Trees are a suitable representation for some functions and unsuitable for others.
- ✔ What is the cardinality of the set of Boolean functions of n attributes?
  - ✘ It is equal to the number of truth tables that can be created with n attributes.
  - ✘ The truth table has 2^n rows, i.e. there is 2^{2^n} different functions.
  - ✘ The set of trees is even larger; several trees represent the same function.
- ✔ We need a clever algorithm to find good hypotheses (trees) in such a large space.

Learning a Decision Tree 6 / 29

A computer game

labels

Example 1:
Can you distinguish between allies and enemies after seeing a few of them?

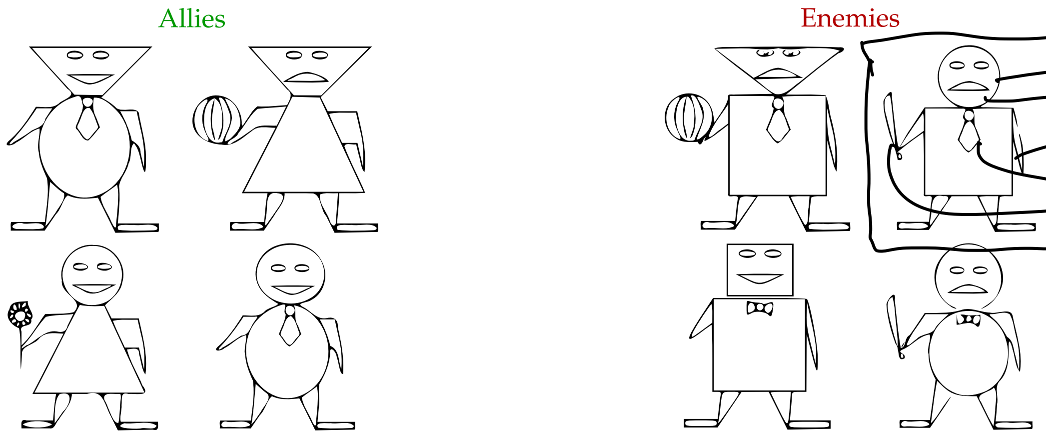


Hint: concentrate on the shapes of heads and bodies.
Answer: Seems like allies have the same shape of their head and body.
How would you represent this by a decision tree? (Relation among attributes.)
How do you know that you are right?

features = ?

A computer game

Example 2:
Some robots changed their attitudes:



circle
no smile
square
tie
sword

No obvious simple rule.
How to build a decision tree discriminating the 2 robot classes?

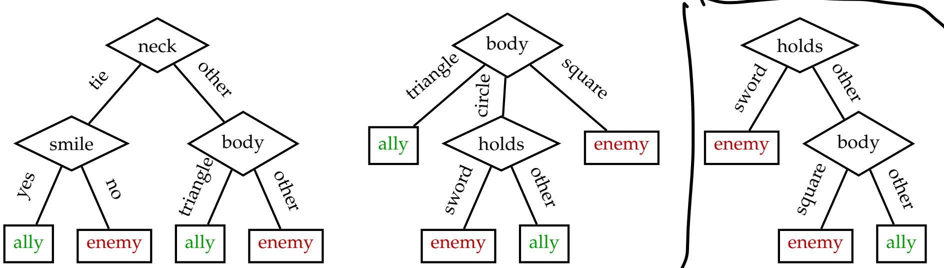
Alternative hypotheses

Example 2: Attribute description:

head	body	smile	neck	holds	class
triangle	circle	yes	tie	nothing	ally
triangle	triangle	no	nothing	ball	ally
circle	triangle	yes	nothing	flower	ally
circle	circle	yes	tie	nothing	ally
triangle	square	no	tie	ball	enemy
circle	square	no	tie	sword	enemy
square	square	yes	bow	nothing	enemy
circle	circle	no	bow	sword	enemy

best?
DT

Alternative hypotheses (suggested by an oracle for now): Which of the trees is the best (right) one?



How to choose the best tree?

We want a tree that is

- ✔ **consistent** with the data,
- ✔ is as **small** as possible, and
- ✔ which also **works for new data**.

Consistent with data?

- ✔ All 3 trees are consistent.

Small?

- ✔ The right-hand side one is the simplest one:

	left	middle	right
depth	2	2	2
leaves	4	4	3
conditions	3	2	2

Will it work for new data?

- ✔ We have no idea!
- ✔ We need a set of new testing data (different data from the same source).

Learning a Decision Tree

It is an intractable problem to find **the smallest consistent tree** among $> 2^{2^n}$ trees.
We can find approximate solution: **a small (but not the smallest) consistent tree**.

Top-Down Induction of Decision Trees (TDIDT):

- ✔ A greedy divide-and-conquer strategy.
- ✔ Progress:
  1. Test the most important attribute.
  2. Divide the data set using the attribute values.
  3. For each subset, build an independent tree (recursion).
- ✔ "Most important attribute": attribute that makes the most difference to the classification.
- ✔ All paths in the tree will be short, the tree will be shallow.

Attribute importance

head	body	smile	neck	holds	class
triangle	circle	yes	tie	nothing	ally
triangle	triangle	no	nothing	ball	ally
circle	triangle	yes	nothing	flower	ally
circle	circle	yes	tie	nothing	ally
triangle	square	no	tie	ball	enemy
circle	square	no	tie	sword	enemy
square	square	yes	bow	nothing	enemy
circle	circle	no	bow	sword	enemy
triangle: 2:1	triangle: 2:0	yes: 3:1	tie: 2:2	ball: 1:1	
circle: 2:2	circle: 2:1	no: 1:3	bow: 0:2	sword: 0:2	
square: 0:1	square: 0:3		nothing: 2:0	flower: 1:0	
				nothing: 2:1	

A perfect attribute divides the examples into sets each of which contain only a single class. (Do you remember the simply created perfect attribute from Example 1?)

A useless attribute divides the examples into sets each of which contains the same distribution of classes as the set before splitting.

None of the above attributes is perfect or useless. Some are more useful than others.

Choosing the test attribute

Information gain:

- Formalization of the terms "useless", "perfect", "more useful".
- Based on entropy, a measure of the uncertainty of a random variable V with possible values v_i:

$$H(V) = -\sum_i p(v_i) \log_2 p(v_i)$$

- Entropy of the target class C measured on a data set S (a finite-sample estimate of the true entropy):

$$H(C, S) = -\sum_i p(c_i) \log_2 p(c_i),$$

  where $p(c_i) = \frac{N_S(c_i)}{|S|}$, and $N_S(c_i)$ is the number of examples in S that belong to class c_i.

- The entropy of the target class C **remaining in the data set** S **after splitting** into subsets S_k using values of attribute A (weighted average of the entropies in individual subsets):

$$H(C, S, A) = \sum_k p(S_k) H(C, S_k), \qquad \text{where } p(S_k) = \frac{|S_k|}{|S|}$$

- The information gain of attribute A for a data set S is

$$Gain(A, S) = H(C, S) - H(C, S, A).$$

  Choose the attribute with the highest information gain, i.e. the attribute with the lowest $H(C, S, A)$.

Choosing the test attribute (special case: binary classification)

- For a Boolean random variable V which is true with probability q, we can define:

$$H_B(q) = -q \log_2 q - (1-q) \log_2 (1-q)$$

- Entropy of the target class C measured on a data set S with N_p positive and N_n negative examples:

$$H(C,S) = H_B\left(\frac{N_p}{N_p + N_n}\right) = H_B\left(\frac{N_p}{|S|}\right)$$

Choosing the test attribute (example)

head		body		smile		neck		holds	
triangle:	2:1	triangle:	2:0	yes:	3:1	tie:	2:2	ball:	1:1
circle:	2:2	circle:	2:1	no:	1:3	bow:	0:2	sword:	0:2
square:	0:1	square:	0:3			nothing:	2:0	flower:	1:0
								nothing:	2:1

head:
$p(S_{\text{head=tri}}) = \frac{3}{8}$; $H(C, S_{\text{head=tri}}) = H_B\left(\frac{2}{2+1}\right) = 0.92$
$p(S_{\text{head=cir}}) = \frac{4}{8}$; $H(C, S_{\text{head=cir}}) = H_B\left(\frac{2}{2+2}\right) = 1$
$p(S_{\text{head=sq}}) = \frac{1}{8}$; $H(C, S_{\text{head=sq}}) = H_B\left(\frac{0}{0+1}\right) = 0$
$H(C, S, head) = \frac{3}{8} \cdot 0.92 + \frac{4}{8} \cdot 1 + \frac{1}{8} \cdot 0 = 0.84$
$Gain(head, S) = 1 - 0.84 = 0.16$

body:
$p(S_{\text{body=tri}}) = \frac{2}{8}$; $H(C, S_{\text{body=tri}}) = H_B\left(\frac{2}{2+0}\right) = 0$
$p(S_{\text{body=cir}}) = \frac{3}{8}$; $H(C, S_{\text{body=cir}}) = H_B\left(\frac{2}{2+1}\right) = 0.92$
$p(S_{\text{body=sq}}) = \frac{3}{8}$; $H(C, S_{\text{body=sq}}) = H_B\left(\frac{0}{0+3}\right) = 0$
$H(C, S, body) = \frac{2}{8} \cdot 0 + \frac{3}{8} \cdot 0.92 + \frac{3}{8} \cdot 0 = 0.35$
$Gain(body, S) = 1 - 0.35 = 0.65$

smile:
$p(S_{\text{smile=yes}}) = \frac{4}{8}$; $H(C, S_{\text{yes}}) = H_B\left(\frac{3}{3+1}\right) = 0.81$
$p(S_{\text{smile=no}}) = \frac{4}{8}$; $H(C, S_{\text{no}}) = H_B\left(\frac{1}{1+3}\right) = 0.81$
$H(C, S, smile) = \frac{4}{8} \cdot 0.81 + \frac{4}{8} \cdot 0.81 + \frac{3}{8} \cdot 0 = 0.81$
$Gain(smile, S) = 1 - 0.81 = 0.19$

neck:
$p(S_{\text{neck=tie}}) = \frac{4}{8}$; $H(C, S_{\text{neck=tie}}) = H_B\left(\frac{2}{2+2}\right) = 1$
$p(S_{\text{neck=bow}}) = \frac{2}{8}$; $H(C, S_{\text{neck=bow}}) = H_B\left(\frac{0}{0+2}\right) = 0$
$p(S_{\text{neck=no}}) = \frac{2}{8}$; $H(C, S_{\text{neck=no}}) = H_B\left(\frac{2}{2+0}\right) = 0$
$H(C, S, neck) = \frac{4}{8} \cdot 1 + \frac{2}{8} \cdot 0 + \frac{2}{8} \cdot 0 = 0.5$
$Gain(neck, S) = 1 - 0.5 = 0.5$

holds:
$p(S_{\text{holds=ball}}) = \frac{2}{8}$; $H(C, S_{\text{holds=ball}}) = H_B\left(\frac{1}{1+1}\right) = 1$
$p(S_{\text{holds=swo}}) = \frac{2}{8}$; $H(C, S_{\text{holds=swo}}) = H_B\left(\frac{0}{0+2}\right) = 0$
$p(S_{\text{holds=flo}}) = \frac{1}{8}$; $H(C, S_{\text{holds=flo}}) = H_B\left(\frac{1}{1+0}\right) = 0$
$p(S_{\text{holds=no}}) = \frac{3}{8}$; $H(C, S_{\text{holds=no}}) = H_B\left(\frac{2}{2+1}\right) = 0.92$
$H(C, S, holds) = \frac{2}{8} \cdot 1 + \frac{2}{8} \cdot 0 + \frac{1}{8} \cdot 0 + \frac{3}{8} \cdot 0.92 = 0.6$
$Gain(holds, S) = 1 - 0.6 = 0.4$

The **body** attribute

- brings us the largest information gain, thus
- it shall be chosen for the first test in the tree!

Entropy gain toy example

At each split we are going to choose the feature that gives the highest information gain.

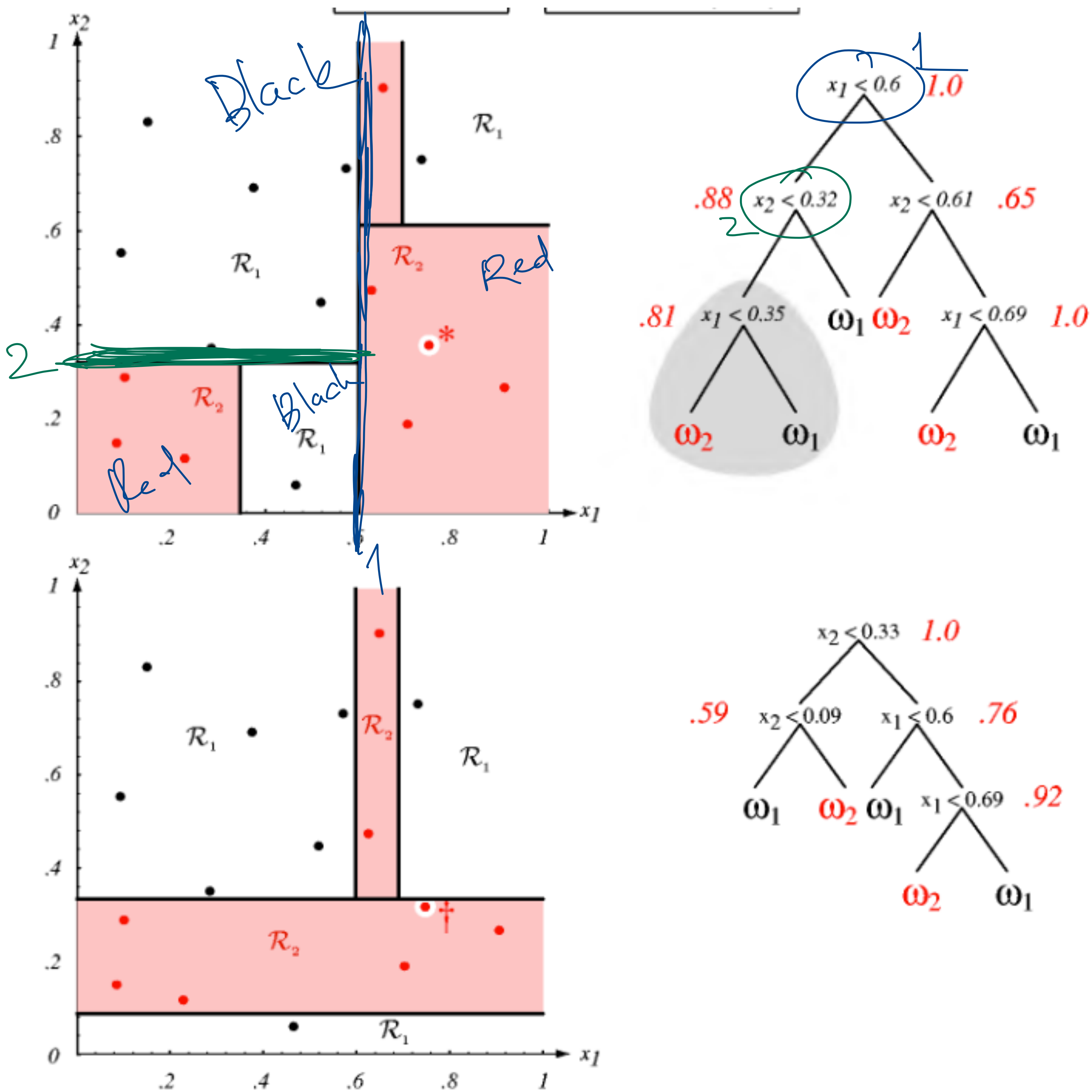
X^1	X^2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

Figure 6: 2 possible features to split by

$$H(Y|X^1) = \frac{1}{2}H(Y|X^1 = T) + \frac{1}{2}H(Y|X^1 = F) = 0 + \frac{1}{2}(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}) \approx .405$$
$$IG(X^1) = H(Y) - H(Y|X^1) = .954 - .405 = .549$$

$$H(Y|X^2) = \frac{1}{2}H(Y|X^2 = T) + \frac{1}{2}H(Y|X^2 = F) = \frac{1}{2}(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}) + \frac{1}{2}(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) \approx .905$$
$$IG(X^2) = H(Y) - H(Y|X^2) = .954 - .905 = .049$$

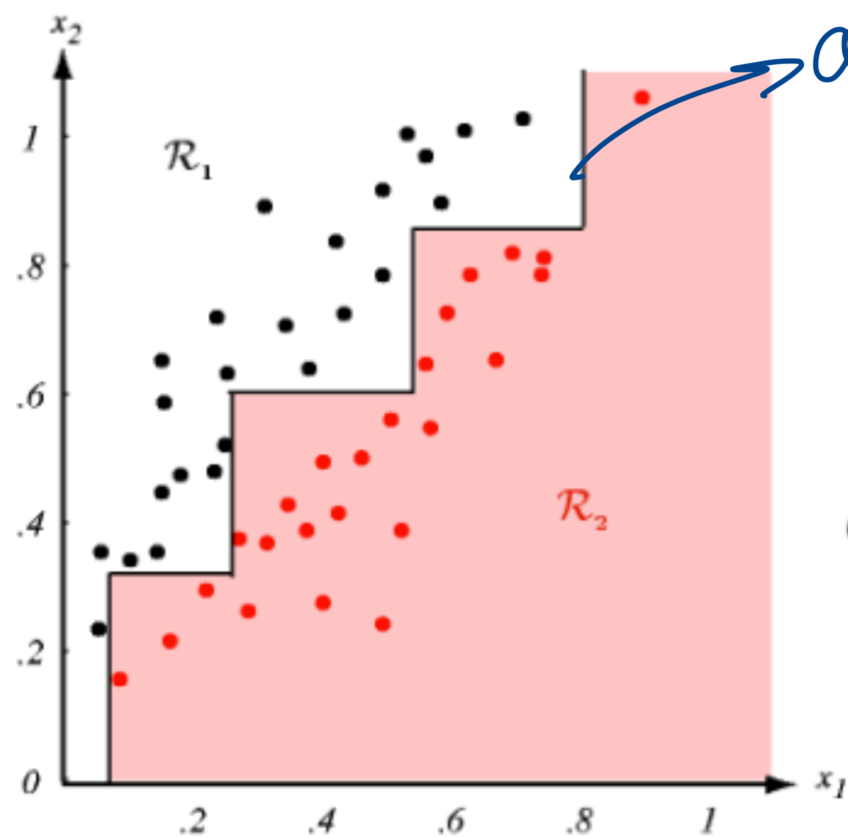
Data Partition Rules



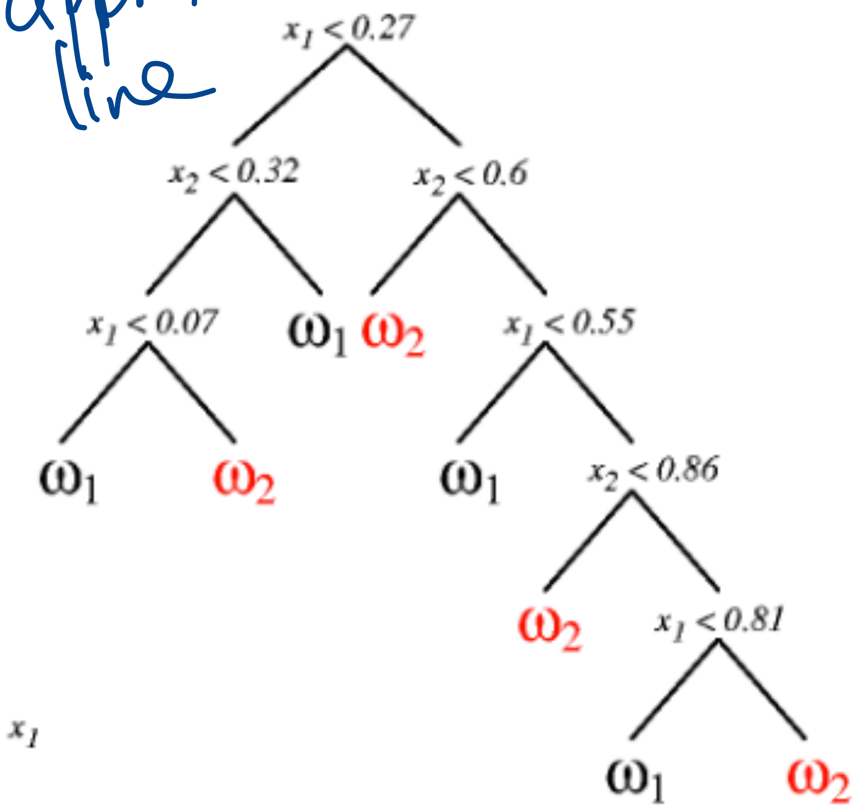
x_1 = word count

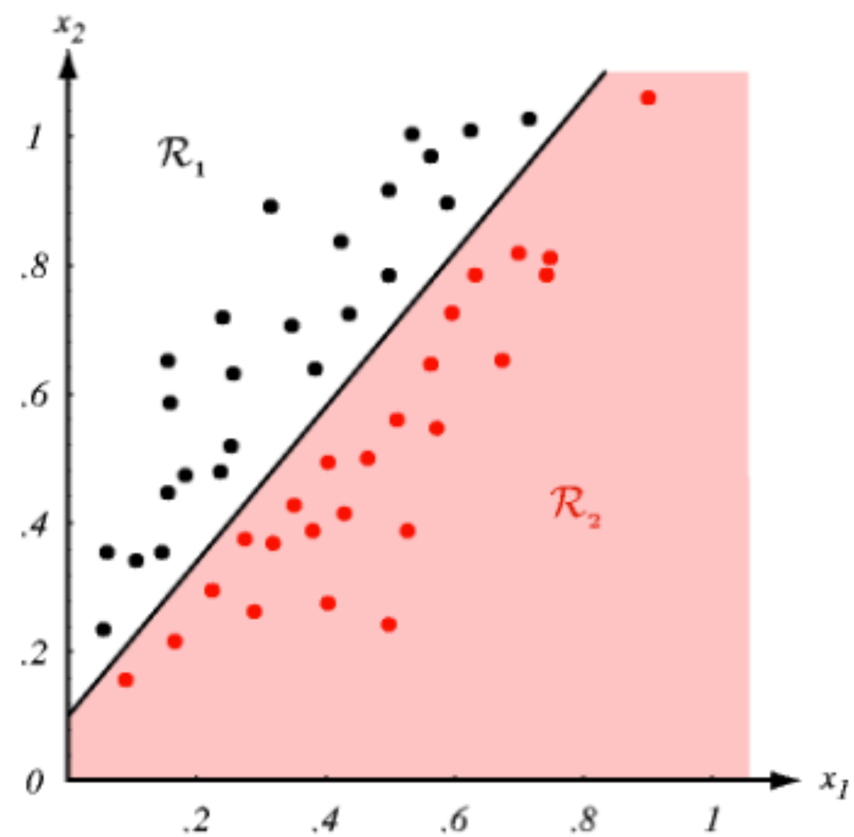
- x_1,x_2 = data features
- Each path in the tree corresponds to a region
- Deeper paths correspond to smaller regions

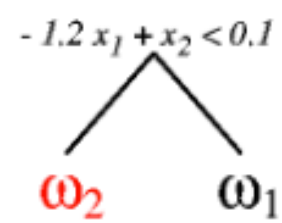
Data Partition Rules



approx of line

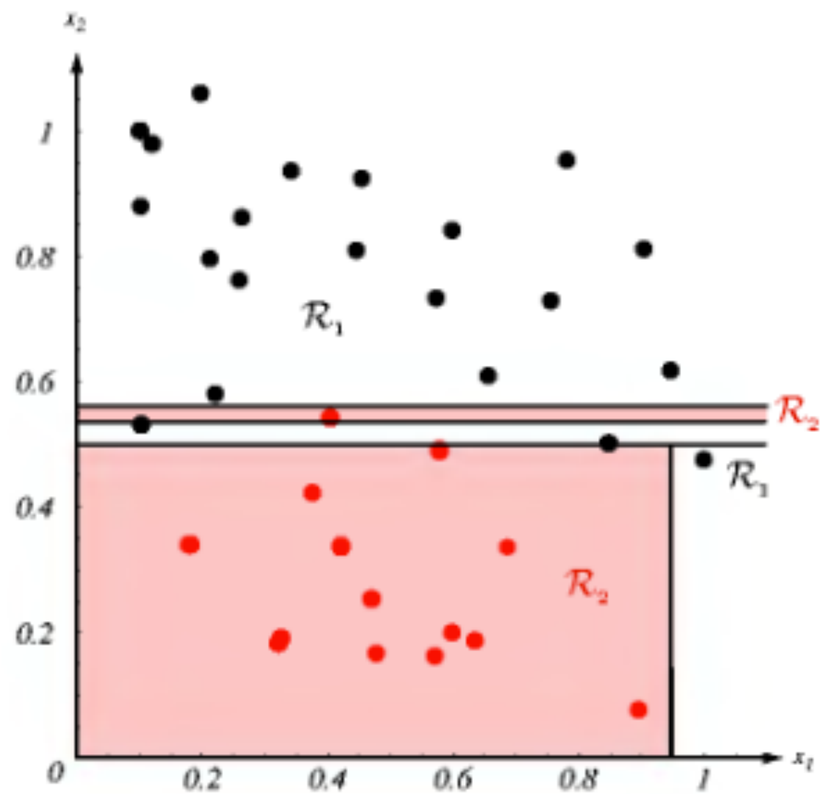


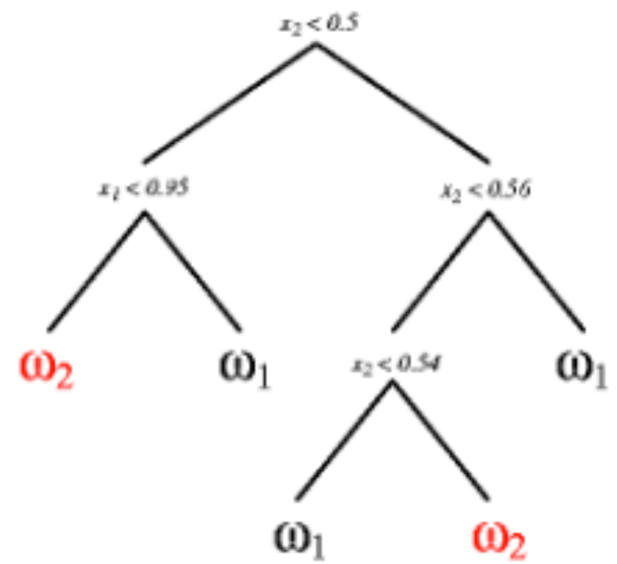


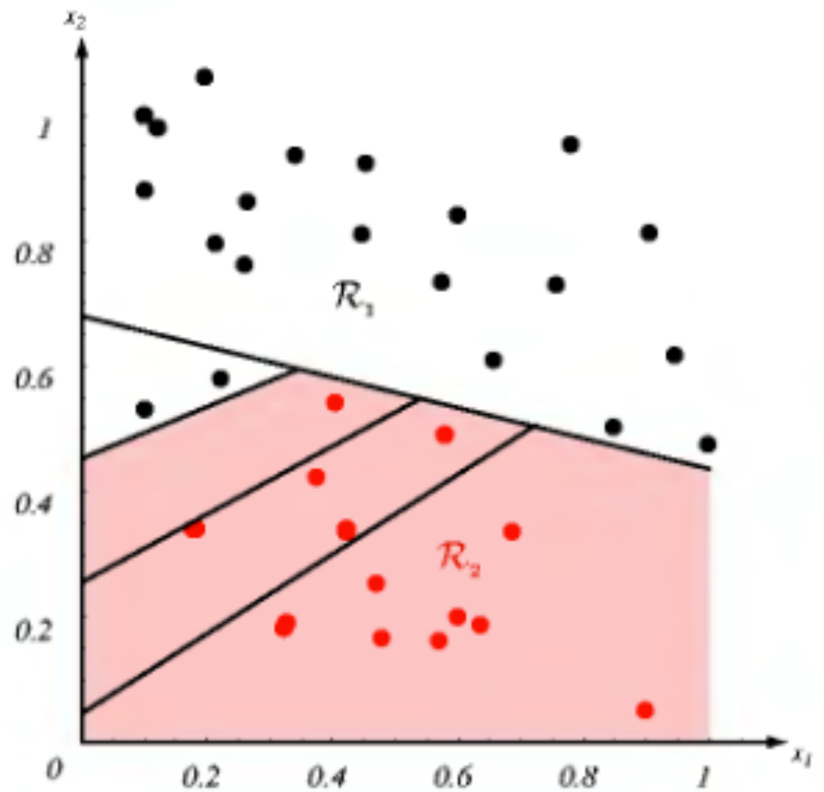


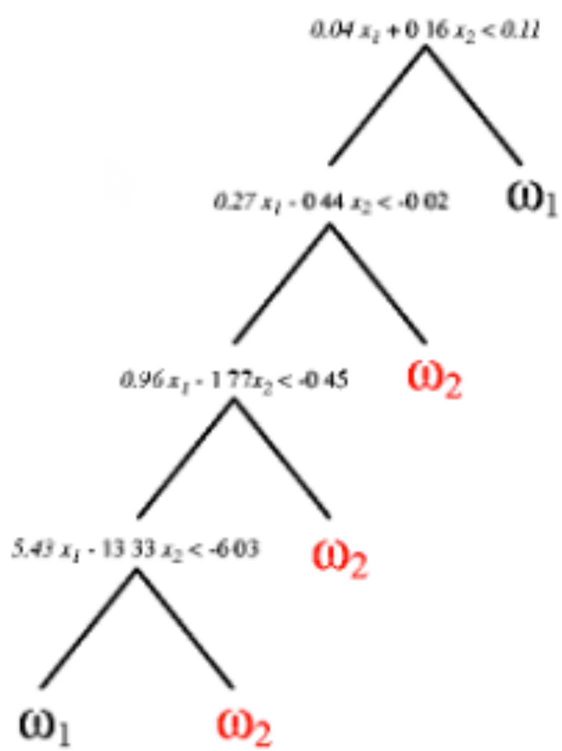
Regression is better

Data Partition Rules









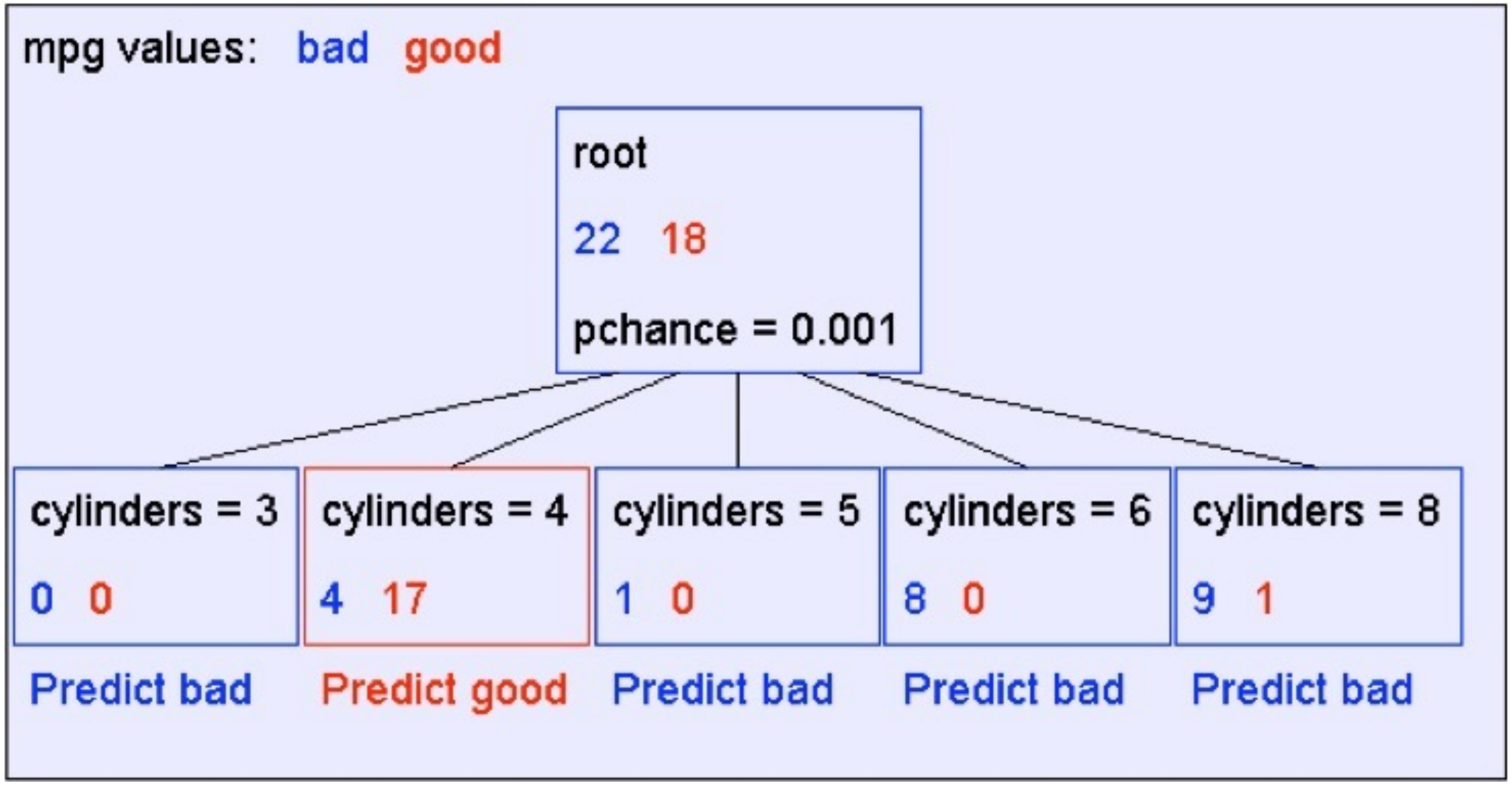
Walkthrough Decision Tree Example

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

40 Records

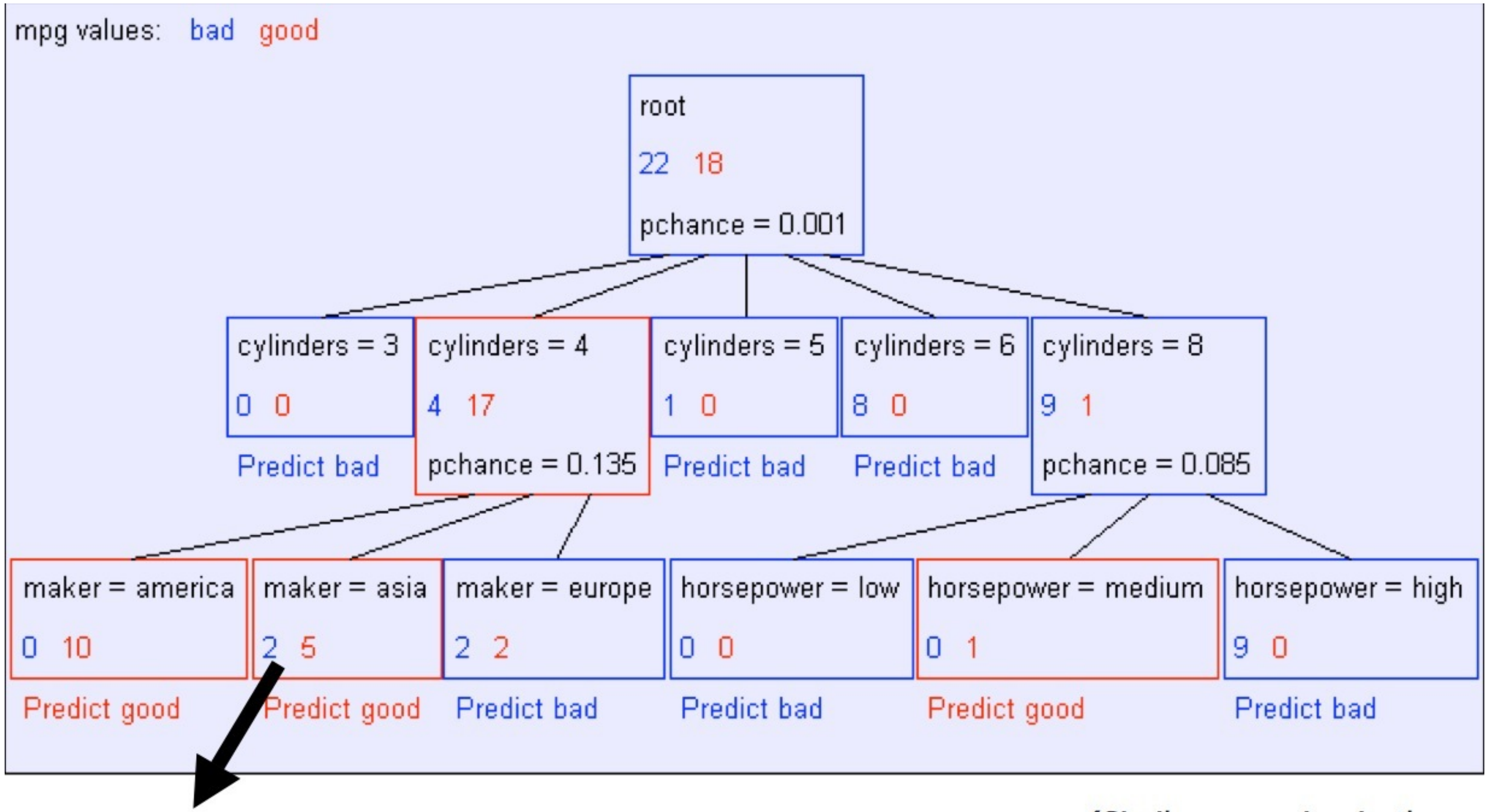
- Data (matrix) example : automobiles
- Target : mpg $\in$ {good, bad} - 2 class /binary problem

Decision Tree Split



- Split by feature “cylinders”, using feature values for branches

Decision Tree Splits



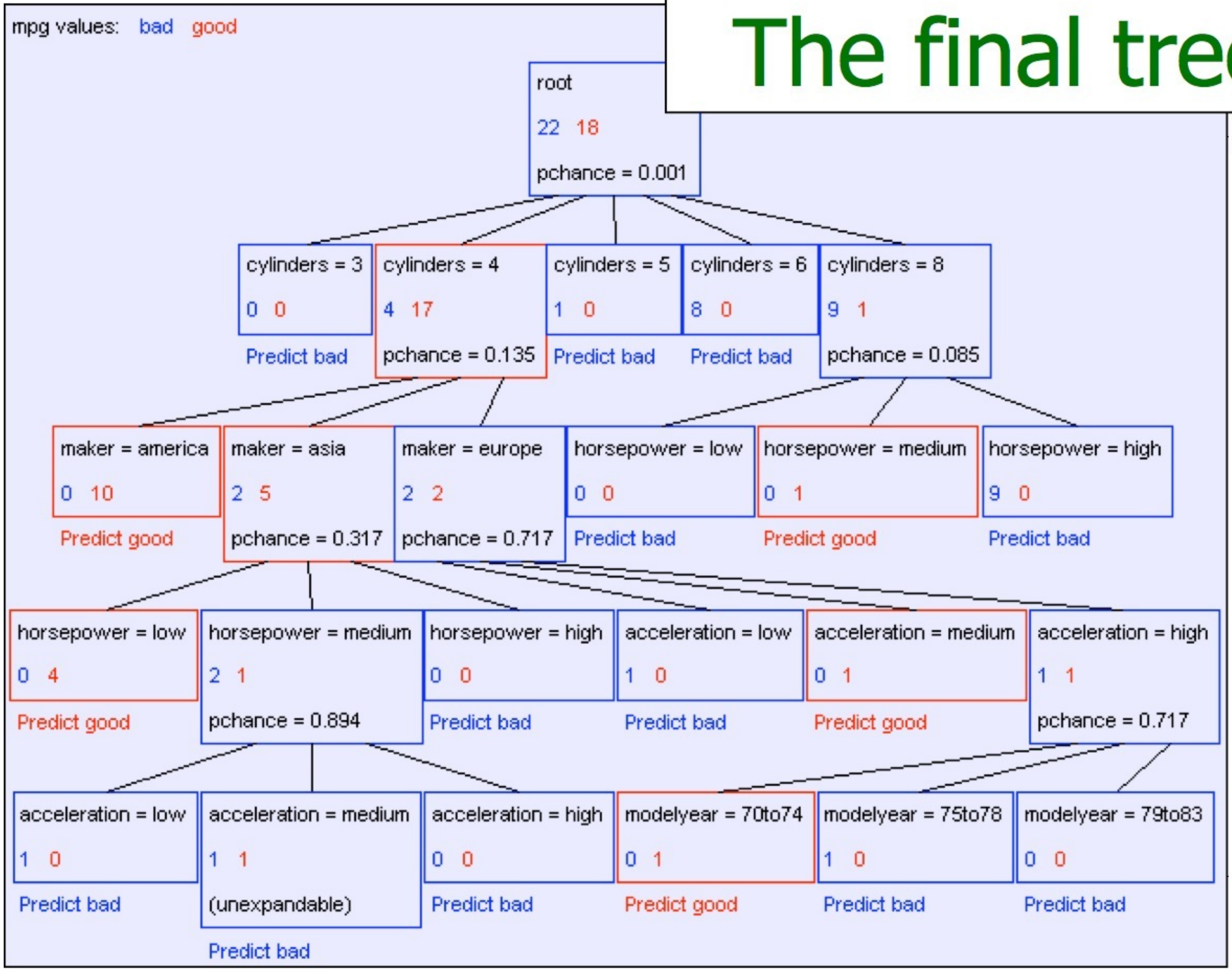
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

- each terminal leaf is labeled by majority (at that leaf). This leaf-label is used for prediction.

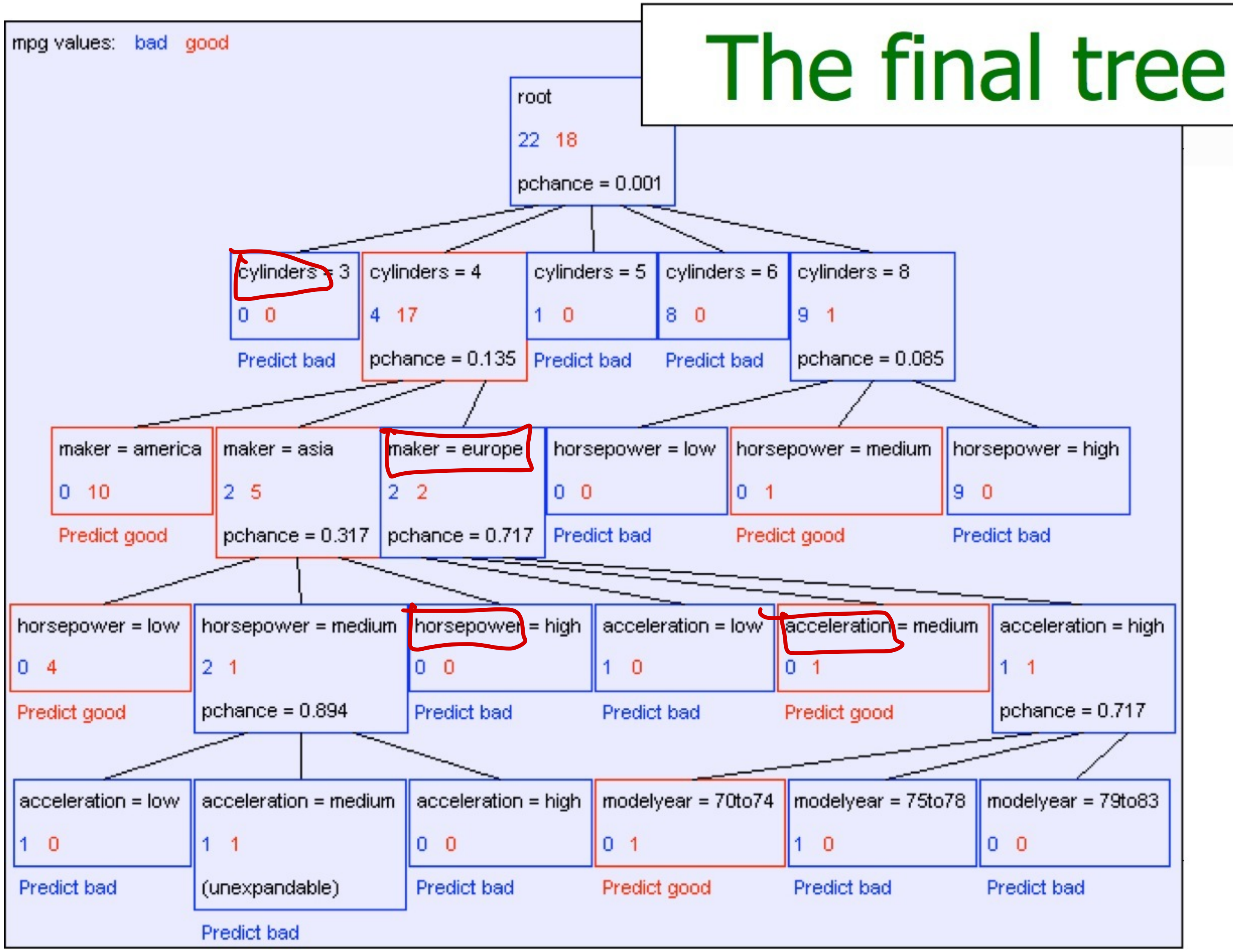
Decision Tree Splits

The final tree



Prediction with a tree

- testpoint:
  - cylinder=4
  - maker=asia
  - horsepower=low
  - weight=low
  - displacement=medium
  - modelyear=75to78



Regression Tree

Variance / square error instead of Information Gain

- same tree structure, split criteria
- assume $\boxed{\text{numerical labels}}$
- for each terminal node compute the node label (predicted value) and the mean square error

Estimate a predicted value per tree node

$$g_m = \frac{\sum_{t \in \chi_m} y_t}{|\chi_m|}$$

Calculate mean square error

$$E_m = \frac{\sum_{t \in \chi_m} (y_t - g_m)^2}{|\chi_m|}$$

Goal: Reduction in variance

- choose a split criteria to minimize the weighted error at children nodes

Regression Tree

labels: 1, 2, 2, 3, 10, 12, 14, 15

$$g = \frac{1+2+2+3+10+12+14+15}{8} = 7.37$$

$$Error = \sum_i (label_i - g)^2 = 247.87$$

labels: 1, 2, 2, 3

$$g = \frac{1+2+2+3}{4} = 2$$

$$Error = \sum_i (label_i - g)^2 = 2$$

labels: 10, 12, 14, 15

$$g = \frac{10+12+14+15}{4} = 12.75$$

$$Error = \sum_i (label_i - g)^2 = 14.75$$

sq loss instead of H

- choose a split criteria to minimize the weighted or total error at children nodes
  - in the example total error after the split is 14.75 + 2=16.75

Prediction with a tree

- for each test datapoint $x=(x^1,x^2,\ldots,x^d)$ follow the corresponding path to reach a terminal node n

- predict the value/label associated with node n

Overfitting

- decision trees can overfit quite badly
  - in fact they are designed to do so due to high complexity of the produced model
  - if a decision tree training error doesn't approach zero, it means that data is inconsistent
  -
- some ideas to prevent overfitting:
  - create more than one tree, each using a different subset of features; average/vote predictions
  - do not split nodes in the tree that have very few datapoints (for example less than 10)
  - only split if the improvement is massive

Pruning

- done also to prevent overfitting
- construct a full decision tree
- then walk back from the leaves and decide to "merge" overfitting nodes
  - when split complexity overwhelms the gain obtained by the spit