ML algorithm

$\boxed{\text{Decision Trees}}$

Sourav Sen Gupta

CDS 2015 | PGDBA | 6 Oct 2015

Label / task / want

Will you eat/wait?

features = attributes

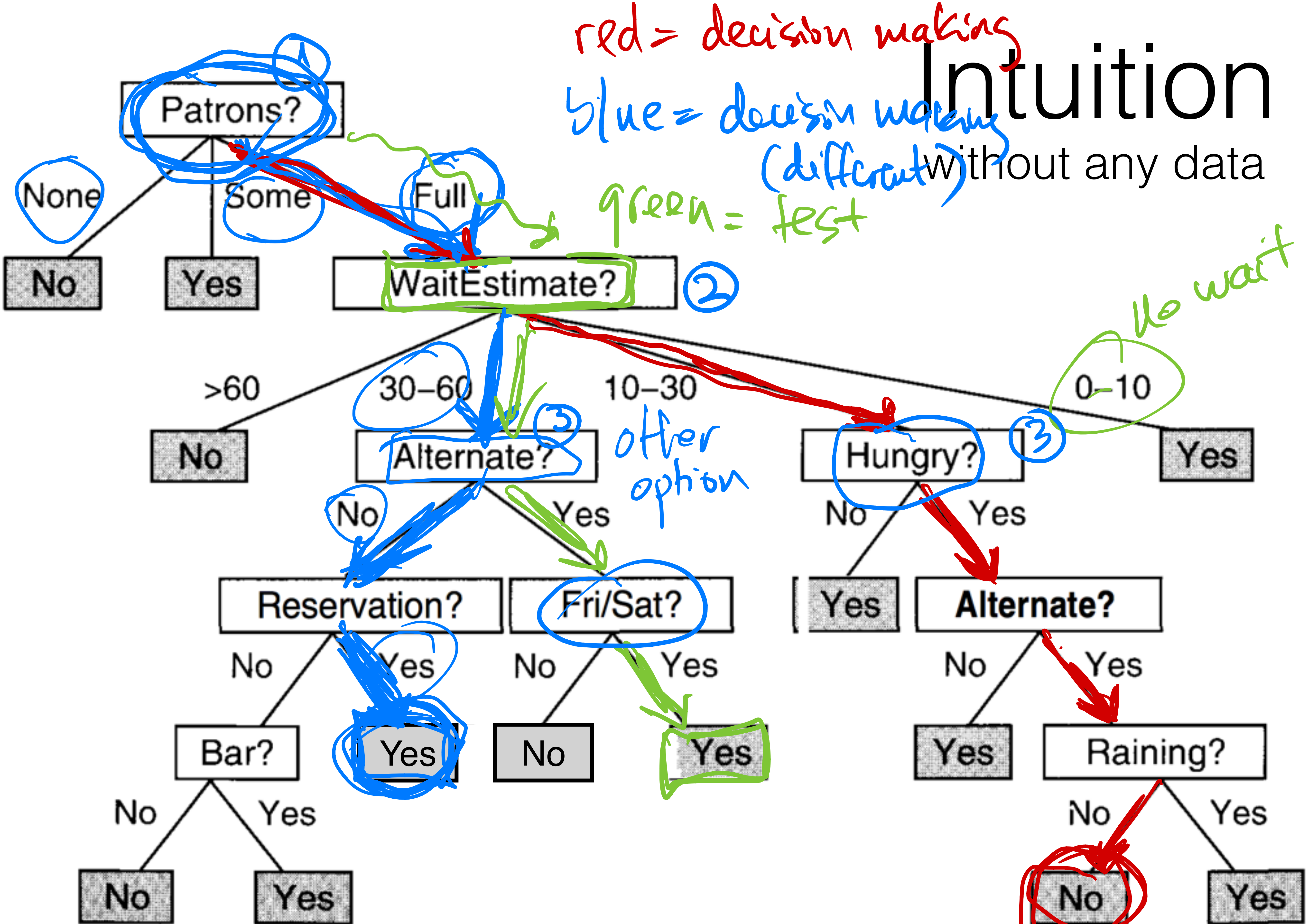
Deciding factors may be

None / Some / Full

- If there are patrons (people inside) — ~~Yes/No.~~ 3 categ
- If you are hungry already — Yes / No
- Alternative options in the vicinity — Yes / No
- The estimated time for waiting — $\boxed{\text{In minutes}}$ quantity / want
- 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 category
- If it is raining at the time — Yes/No
- The genre of cuisine — $\boxed{\text{French,}}$ Italian, Thai, Burger

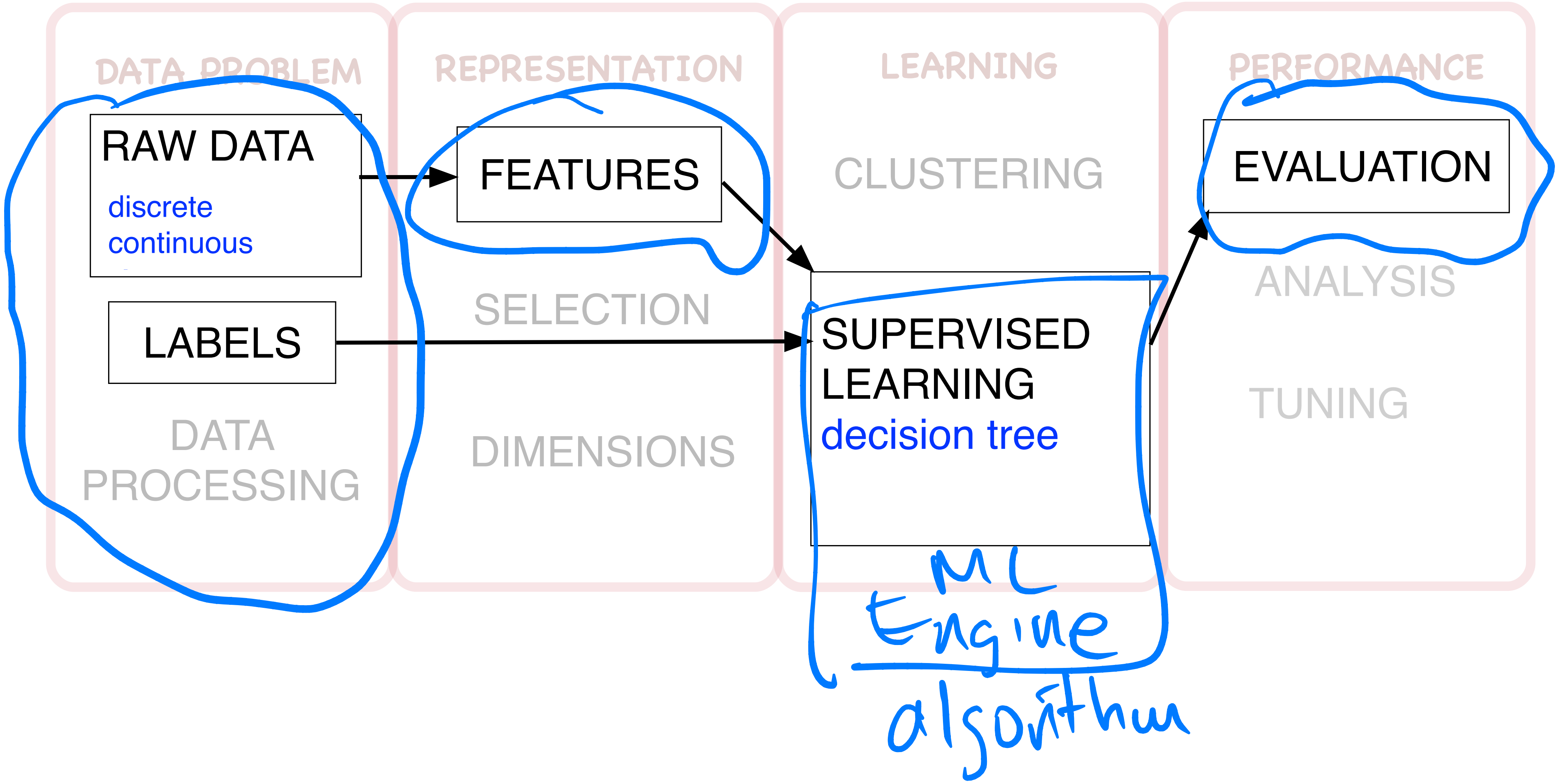
Intuition

without any data



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

ML Pipeline


DATA PROBLEM
REPRESENTATION
LEARNING
PERFORMANCE
RAW DATA
discrete
continuous
LABELS
DATA PROCESSING
FEATURES
SELECTION
DIMENSIONS
CLUSTERING
SUPERVISED LEARNING
decision tree
ML Engine
algorithm
EVALUATION
ANALYSIS
TUNING

12 examples

Training Data

6 pos
6 neg

Example	Attributes									wait	Goal
	Alt	*Bar*	*Fri* sat	*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**	pred

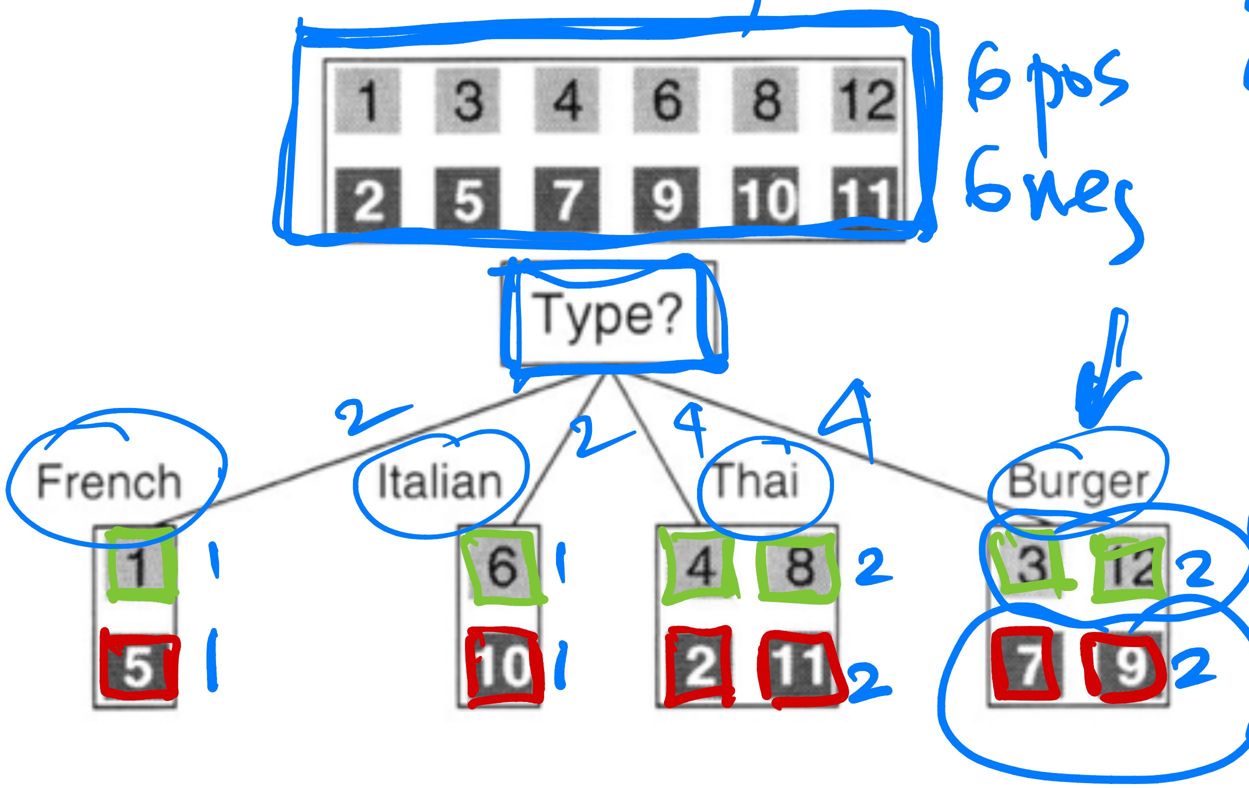
Example Split

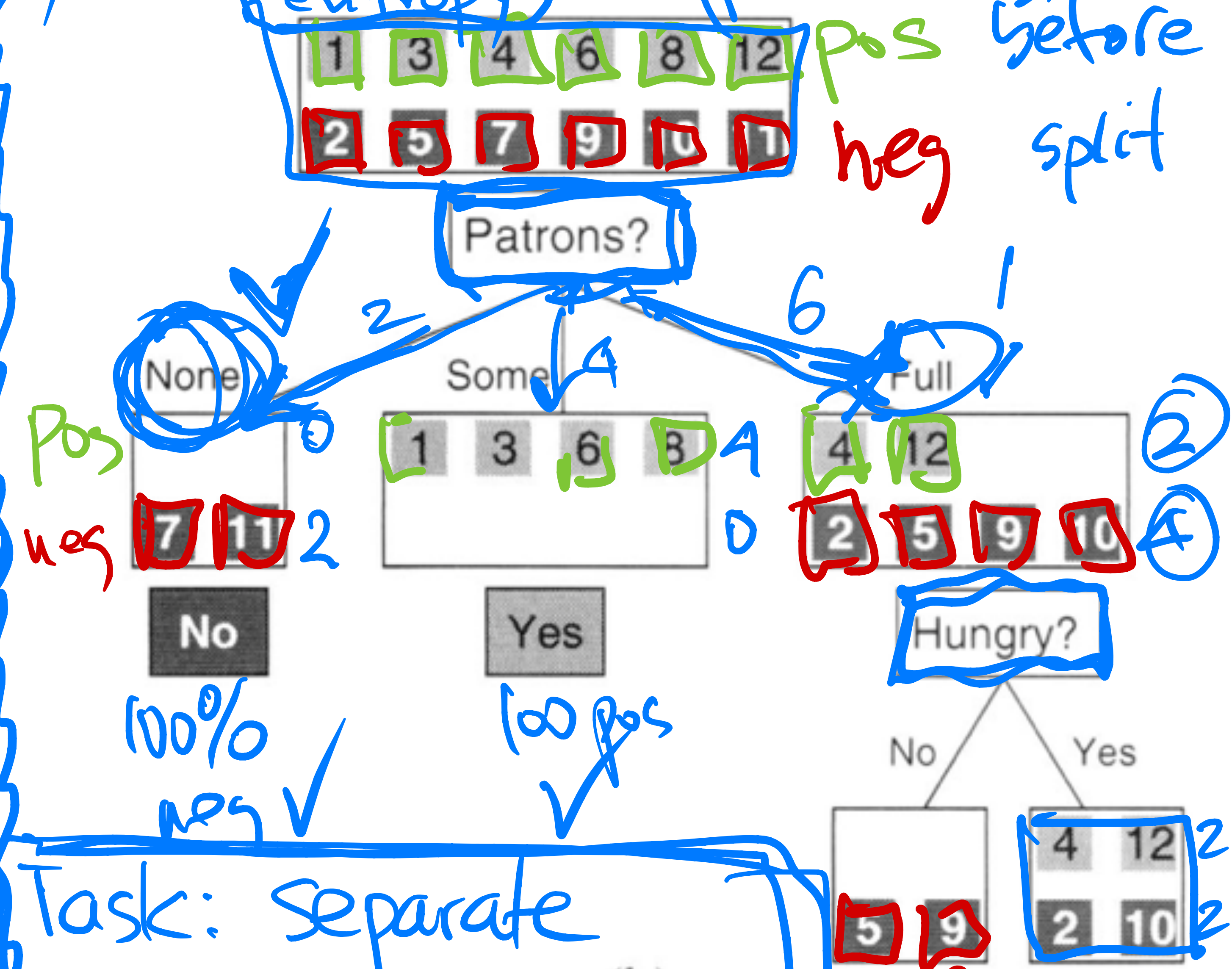
clean branch → big majority on one class (pos or neg)

high uncertainty = high entropy

6 pos 6 neg

pos before split neg





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*

Task: separate pos vs neg

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

H Entropy & ~~Information Gain~~

(distribution)

measures uncertainty

want to decrease it.

Reduction in Entropy → good

- Why a logarithm function?

$p_i = 0$ $\log(0) = NA$

$\boxed{0 \cdot \log(0) = 0}$

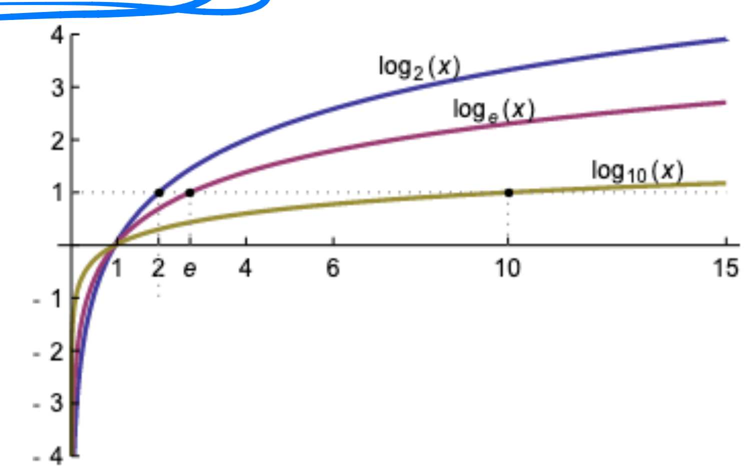
$$log(p_1 \times p_2) = log(p_1) + log(p_2)$$

$$\boxed{-\log x = \log\left(\frac{1}{x}\right)} = \log\left(x^{-1}\right)$$

- **Shannon Entropy:**

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

$$= + \sum p_i \log\left(\frac{1}{p_i}\right)$$

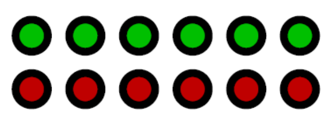


Issue: increasing number of events shrinks the probability.
Solution: use **logarithm of probability** instead and take **the average**.

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"

$p=6$
$n=6$

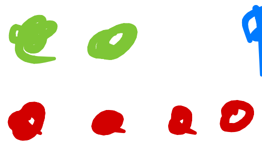


6
6 $\rightarrow$ $\frac{1}{2}$ $\frac{1}{2}$

$H = -\frac{1}{2}\log\frac{1}{2} + -\frac{1}{2}\log\frac{1}{2}$ $\rightarrow$ highest (uniform)

$= \frac{1}{2}\underbrace{\log 2}_{1} + \frac{1}{2}\underbrace{\log 2}_{1} = 1$

1 "bits"



$p=2$
$n=4$ $\Rightarrow$ $\frac{1}{3}$ $\frac{2}{3}$

$H = \frac{1}{3}\log_2(3) + \frac{2}{3}\log_2\left(\frac{3}{2}\right) < 1$

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

prob(p) prob(p) prob(n) prob(n)

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

$\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$

die uniform 6 faces T H J M W F

prob	1/6	1/6	1/6	1/6	1/6	1/6

distribution

$$H = \sum p_i \log\left(\frac{1}{p_i}\right) = \frac{1}{6}\cdot\log 6 + \frac{1}{6}\log 6 + \ldots + \frac{1}{6}\log 6$$

$$= 6\cdot\frac{1}{6}\log 6 = \boxed{\log_2(6)}$$ max H

2.58

nonuniform 6 faces

pros	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{3}{10}$	$\frac{3}{10}$

low high

valid distributio

$$H = \frac{1}{10}\log_2 10\cdot 4 + \frac{3}{10}\log_2\left(\frac{10}{3}\right)\cdot 2 = 2.37$$

skewed die

pros	$\frac{1}{1000}$,	$\frac{1}{1000}$,	$\frac{1}{1000}$,	$\frac{1}{1000}$,	$\frac{1}{1000}$)	$\frac{995}{1000}$
	F1	F2				F6

$$H = \frac{1}{1000}\log_2(1000)\cdot 5 + \frac{995}{1000}\cdot\log_2\left(\frac{1000}{995}\right) = 0.057$$

Information Gain

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

Reduction in H

$$I = H(\mathcal{S}) - \sum_{i \in \{\mathrm{L},\mathrm{R}\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

before split

branches

branch weight

after split

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

most 10,000

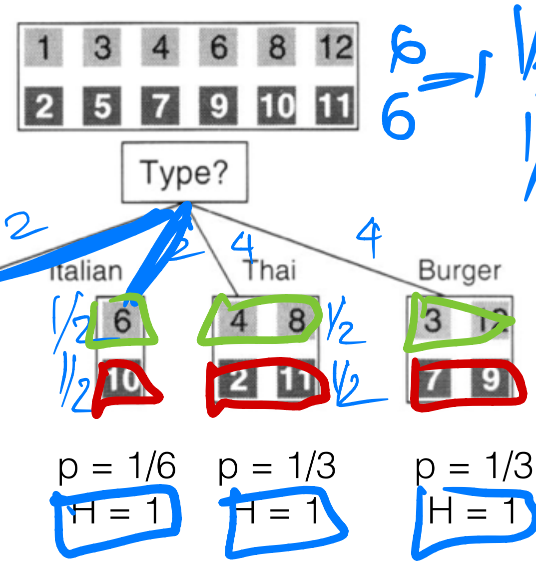
9998

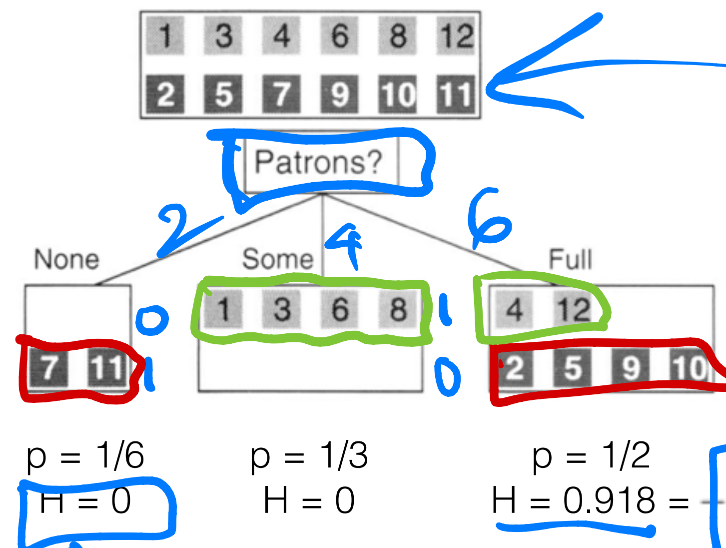
100% neg

2

100% pos

Compare Gain





before split

Parent Entropy

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

E(Child Entropy)

weighted

$$\frac{2}{12}\times 1 + \frac{2}{12}\times 1 + \frac{4}{12}\times 1 + \frac{4}{12}\times 1 = 1$$

Gain = diff(H) = 0

$$\underbrace{0\cdot\log\left(\tfrac{1}{0}\right)}_{0} + \underbrace{1\cdot\log\left(\tfrac{1}{1}\right)}_{0}$$

Parent Entropy

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

E(Child Entropy)

weighted

$$\frac{2}{12}\times 0 + \frac{4}{12}\times 0 + \frac{6}{12}\times 0.918 = 0.459$$

Gain = 1 − .459 = .541

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

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:

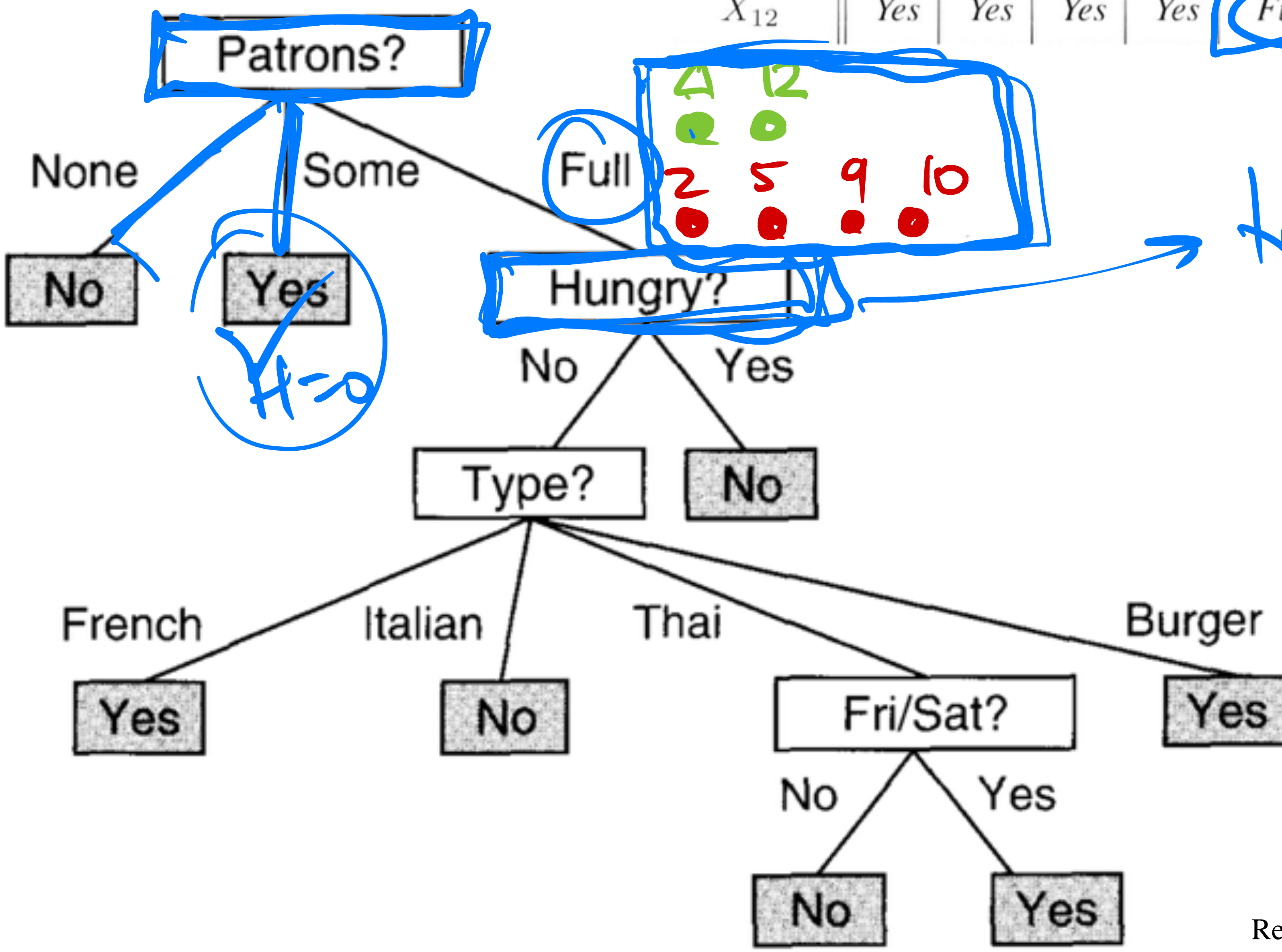
$$I(A) = H\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - EH(A)$$

before after

= **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*



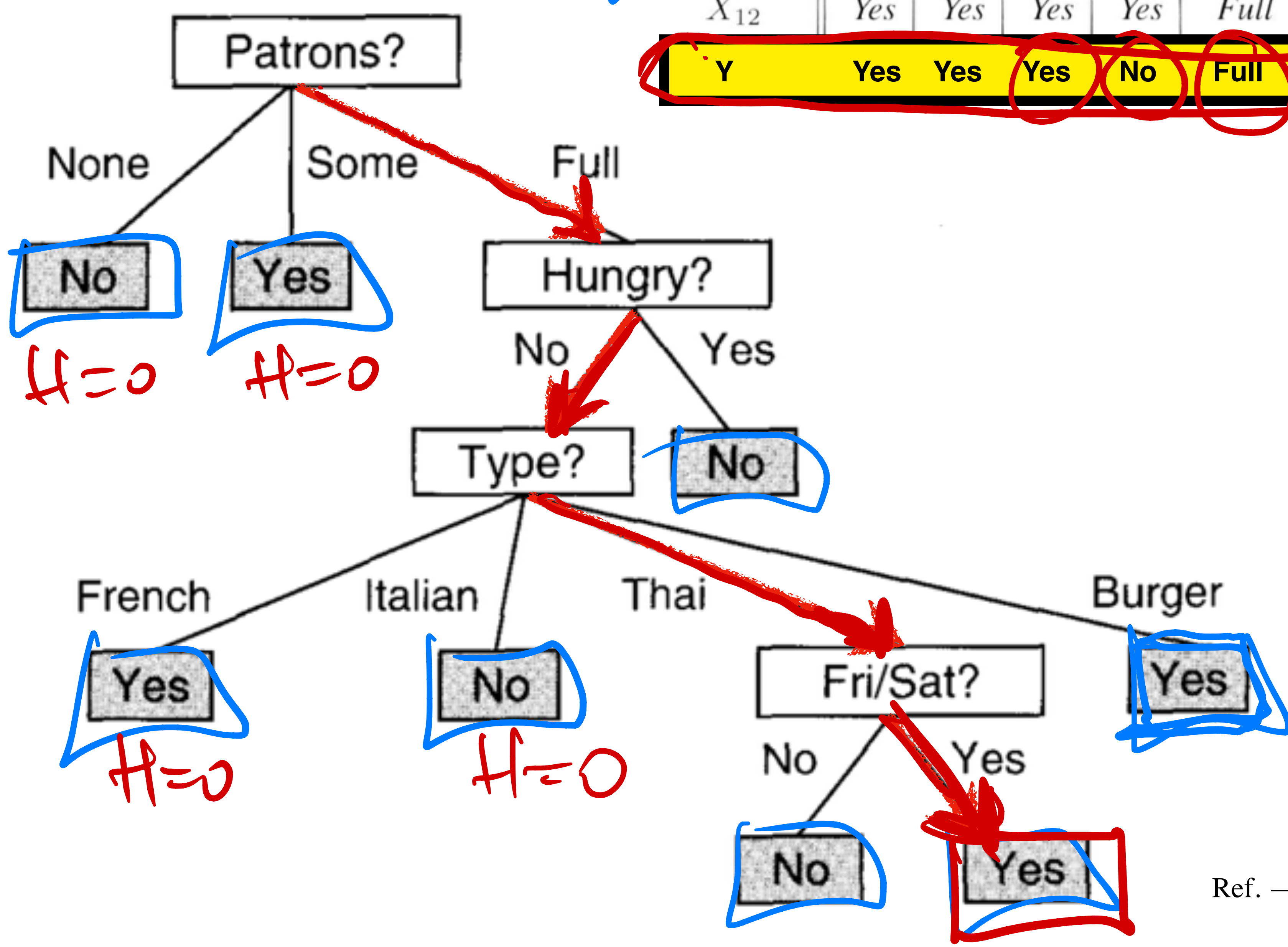
try all splits, pick the best (highest IG = reduction in entropy) for current data (4, 12 / 2, 5, 9, 10) on branch "Full"

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

12

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	?

predict label task



H=0.1 → pos (few)
→ neg (many)
majority

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

Classification Tree

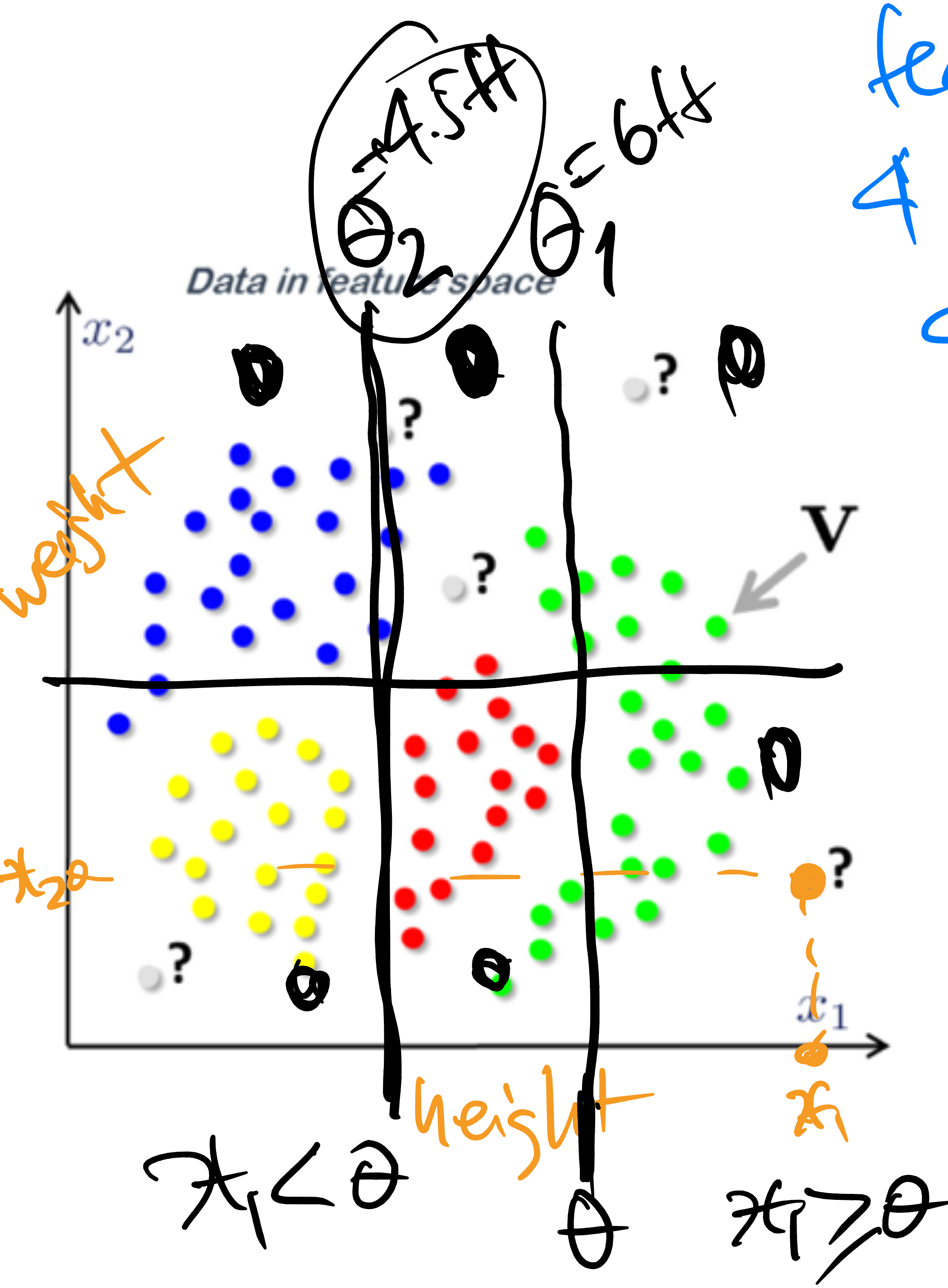
features = coordinates (x_1, x_2) → continuous

4 labels = B, G, R, Y

splits = thresholds (binary)

(height, θ) split

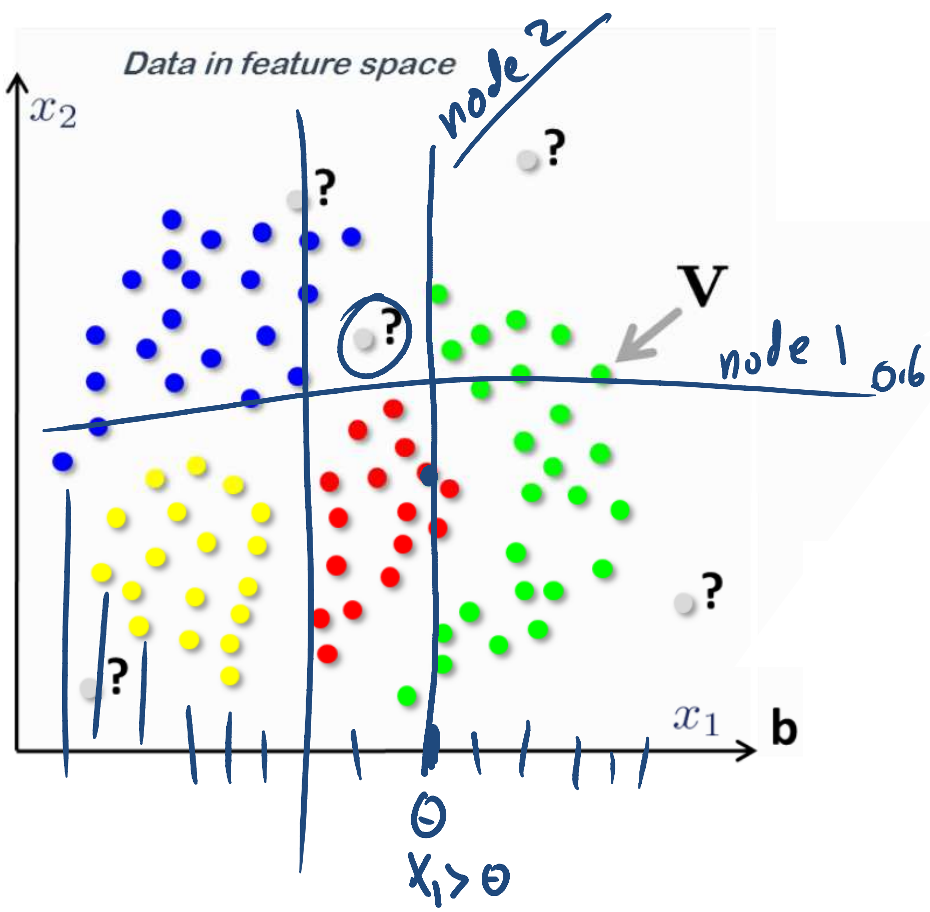
$x_1 < \theta$ ↙ ↘ $x_1 \geq \theta$



θ = ? trial and error

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

Classification tree



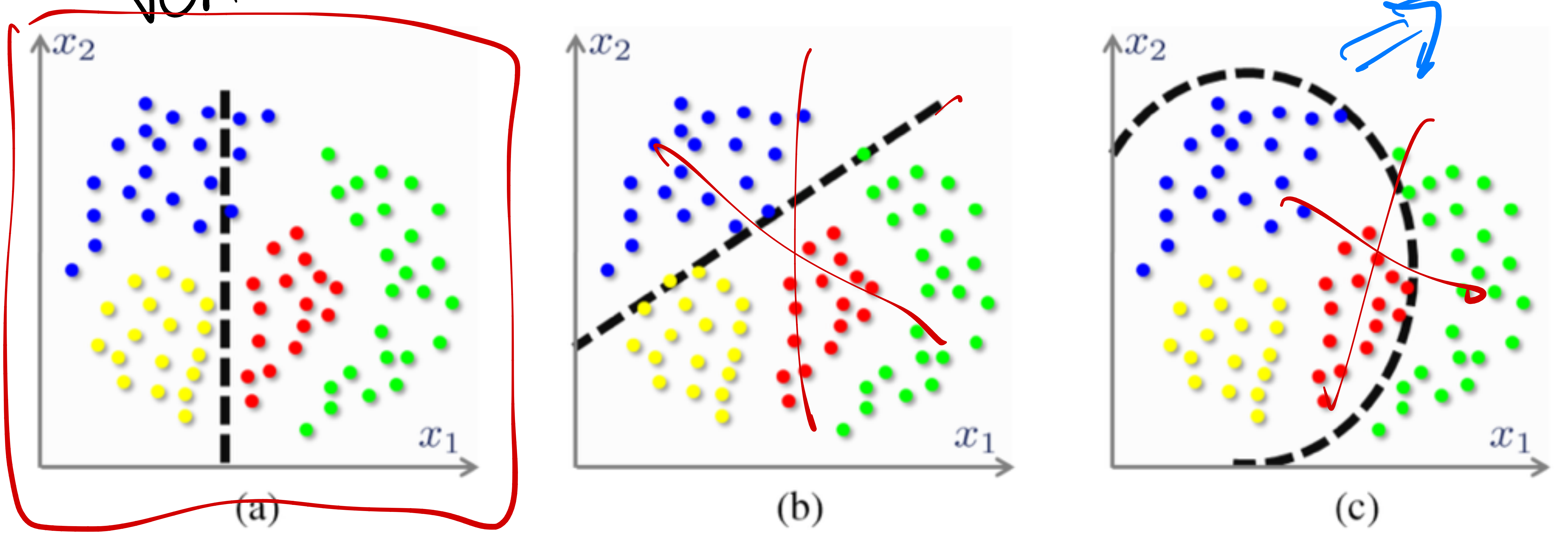
- How to deal with ***continuous features***?
  - Create the splits **randomly**
  - Compute **information gain** for each split
  - Choose the one with **maximum gain**

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

Split Types

horizontal
vertical only

Kernels



$\boxed{\text{Axis-aligned}}$ Hyperplane

General oriented Hyperplane

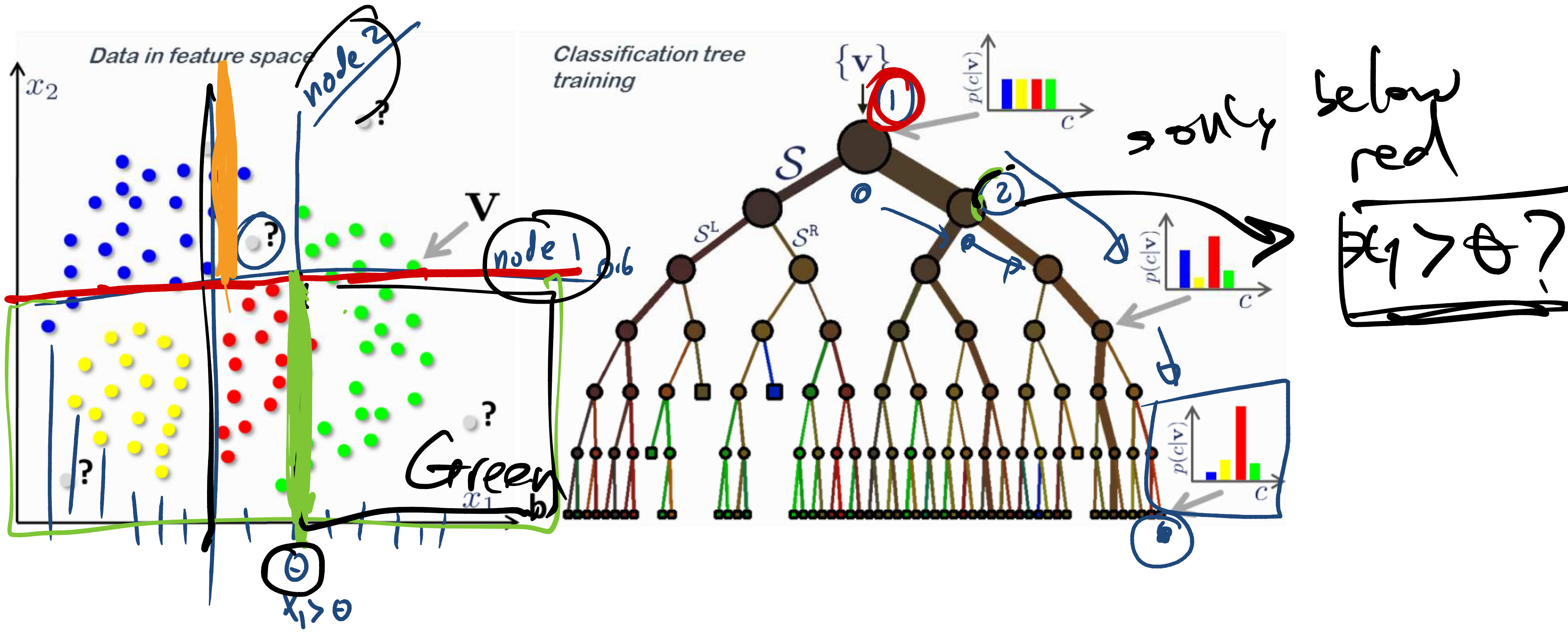
Quadratic/Conic in 2D

avg multiple trees (boosting)

Forests

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

Classification tree



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

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

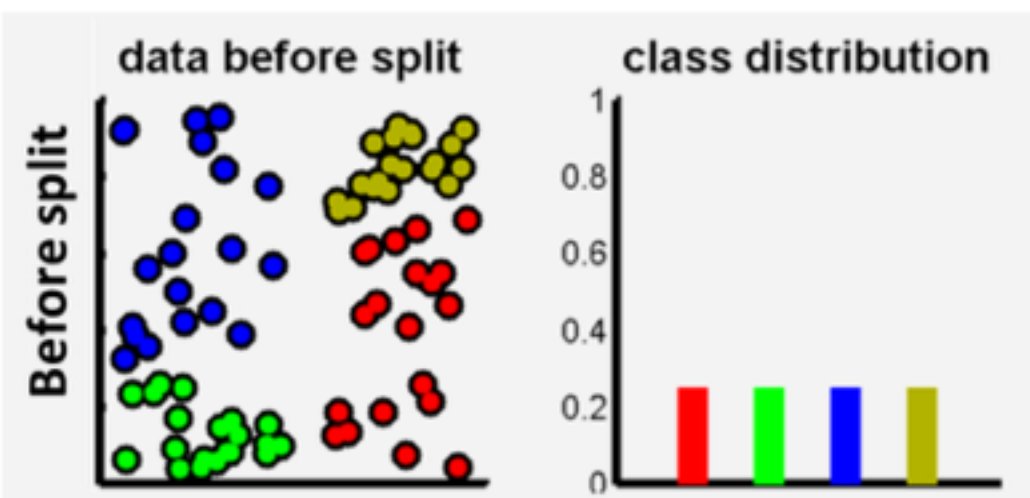
[Criminisi et al, 2011]

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

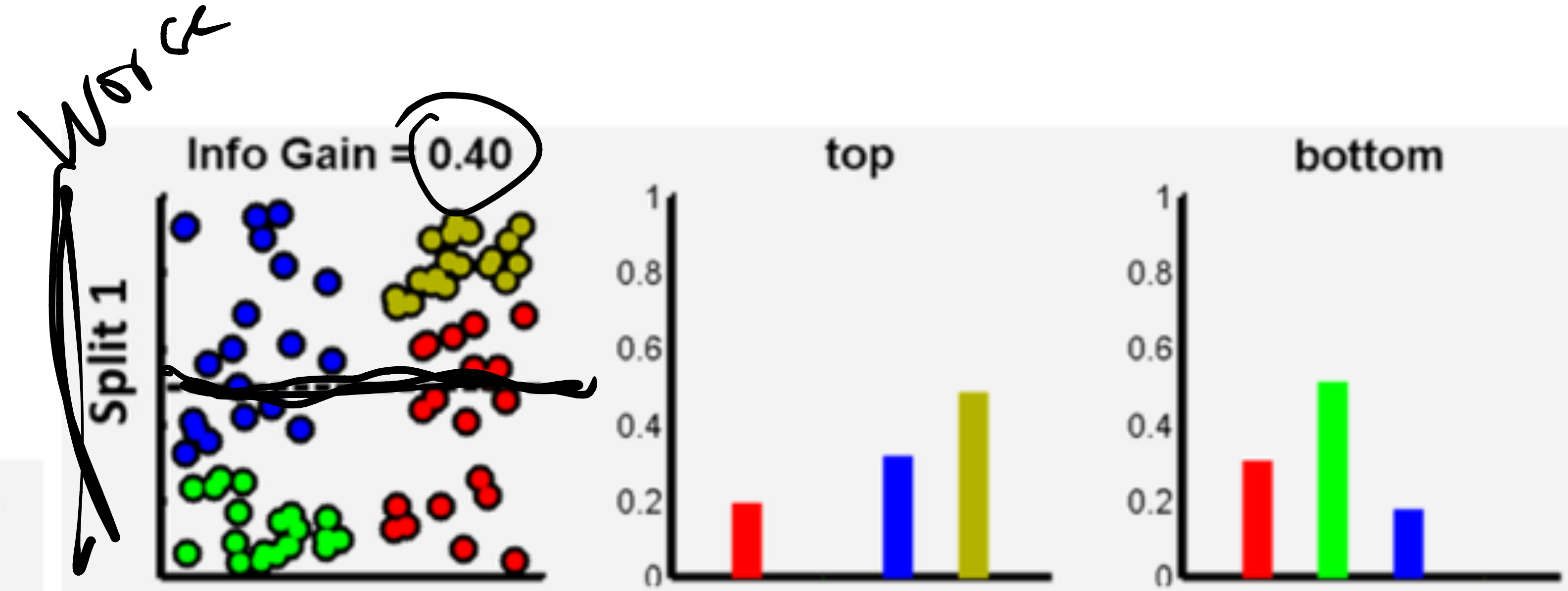
$$p(Class \,|\, Data)$$

Choosing Split

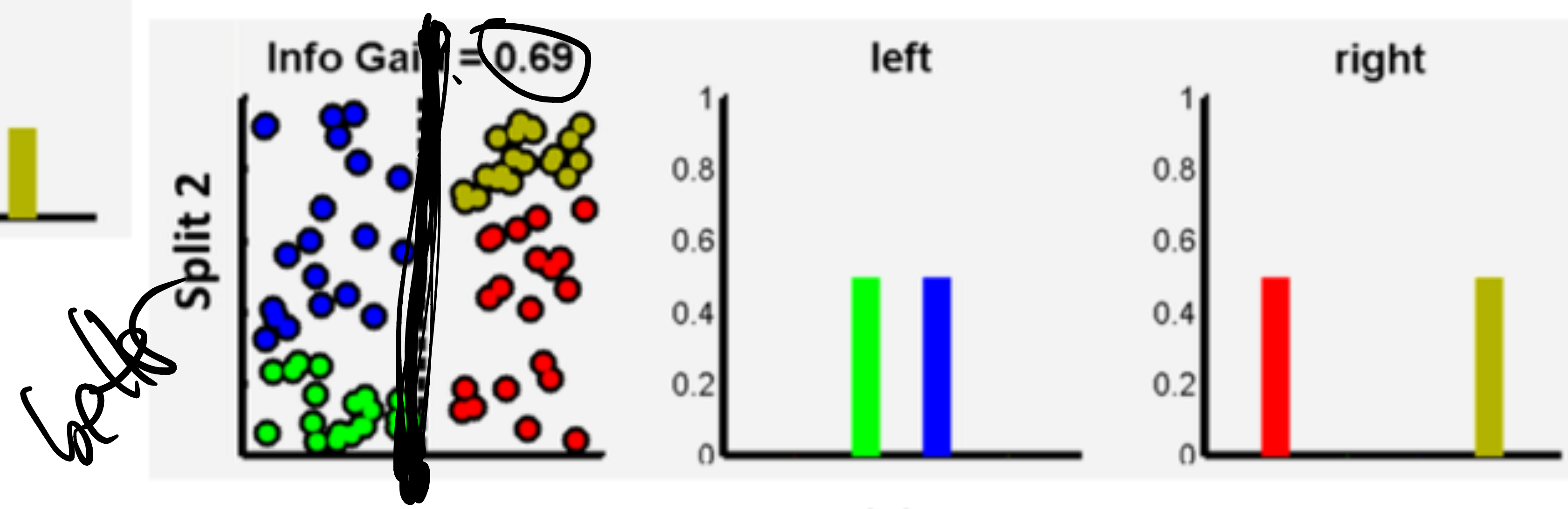
$$\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}_j \in \mathcal{T}_j} I_j$$



(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.

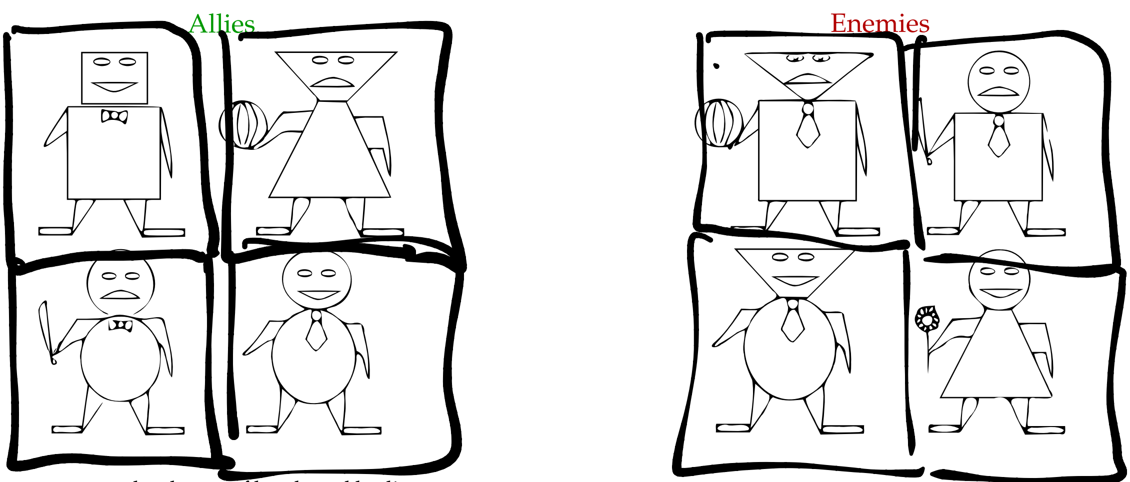
- ✔ Trees are a suitable representation for some functions and unsuitable for others.
- ✔ What is the cardinality of the set of Boolean functions of n attributes?
  - ✘ It is equal to the number of truth tables that can be created with n attributes.
  - ✘ The truth table has 2^n rows, i.e. there is 2^{2^n} 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

A computer game

features?

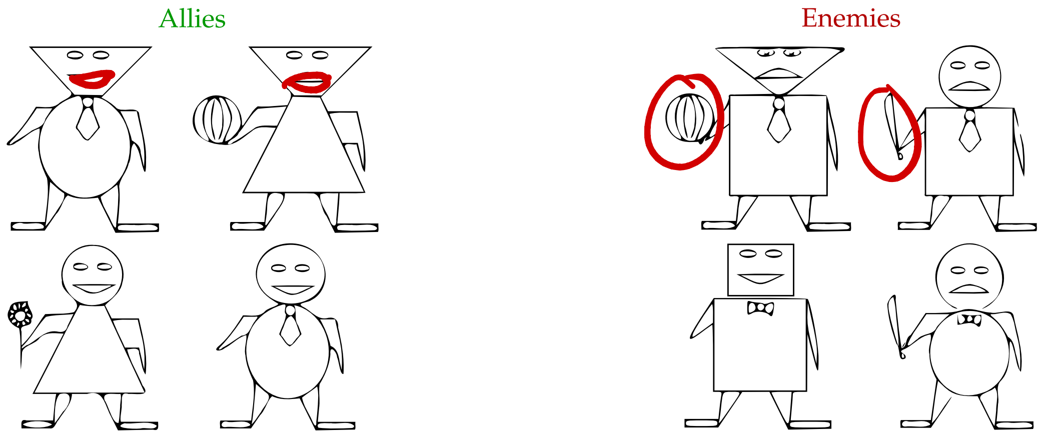
Example 1:
Can you distinguish between allies and enemies after seeing a few of them?



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?

A computer game

Example 2:
Some robots changed their attitudes:



4 pos

4 neg

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

Alternative hypotheses

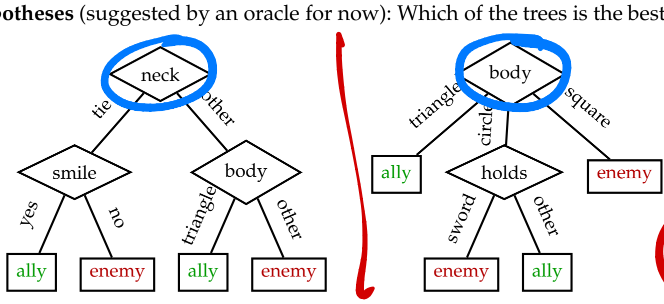
Example 2: Attribute description:

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

train

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

better



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

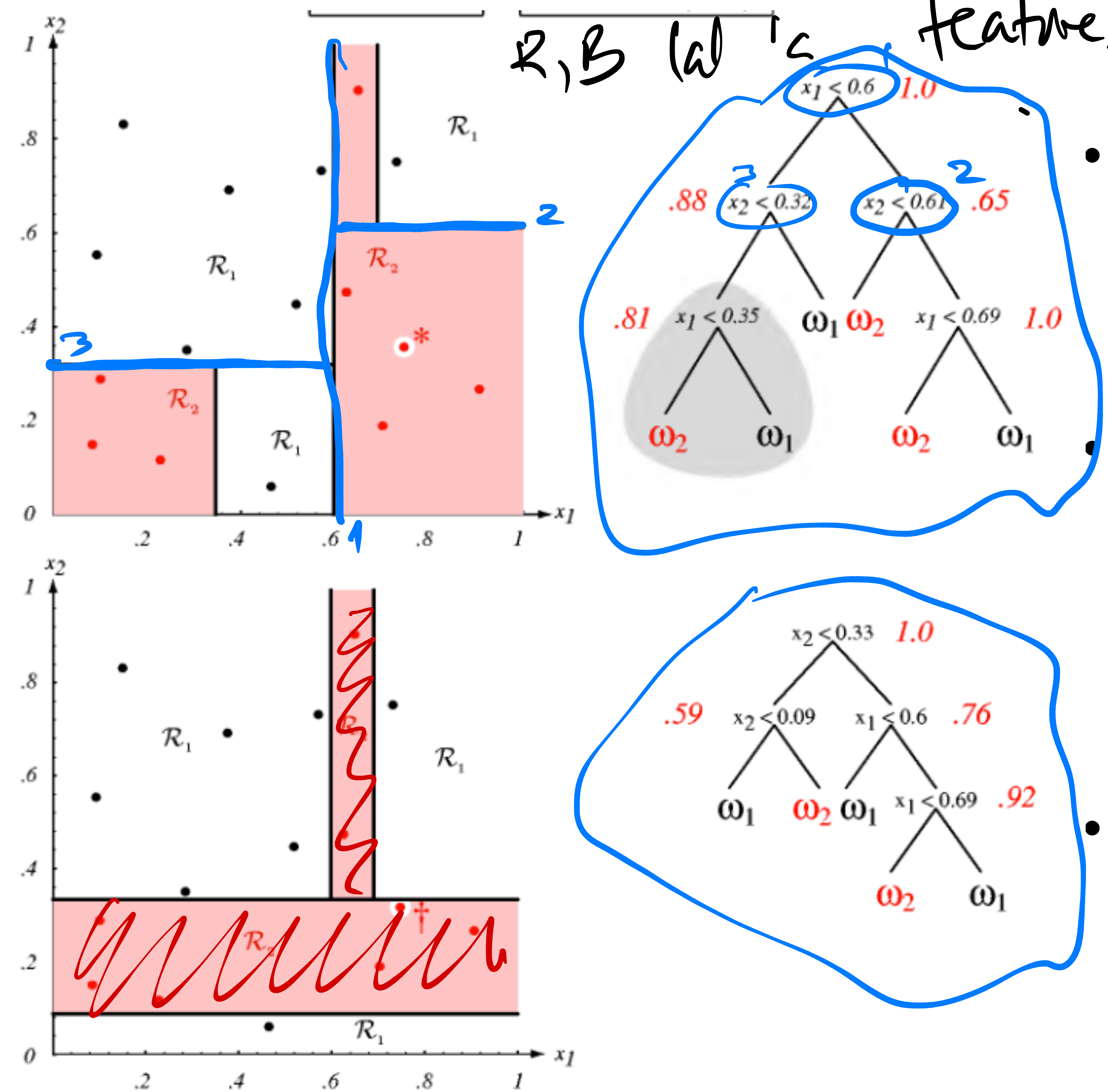
Figure 6: 2 possible features to split by

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

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

Data Partition Rules

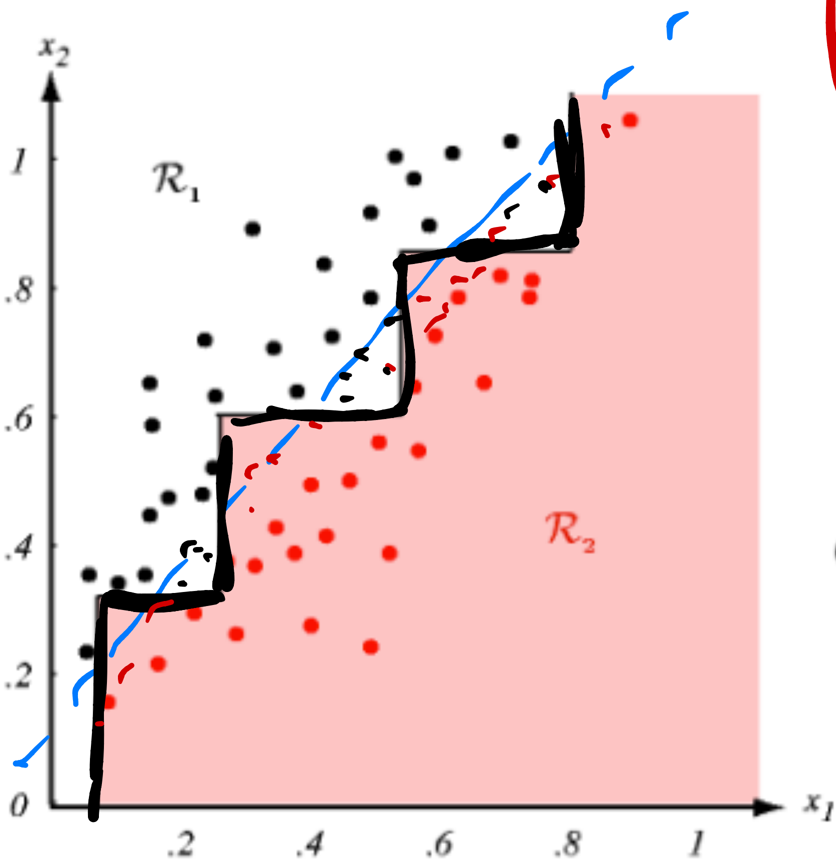
features = coordinates x_1 x_2

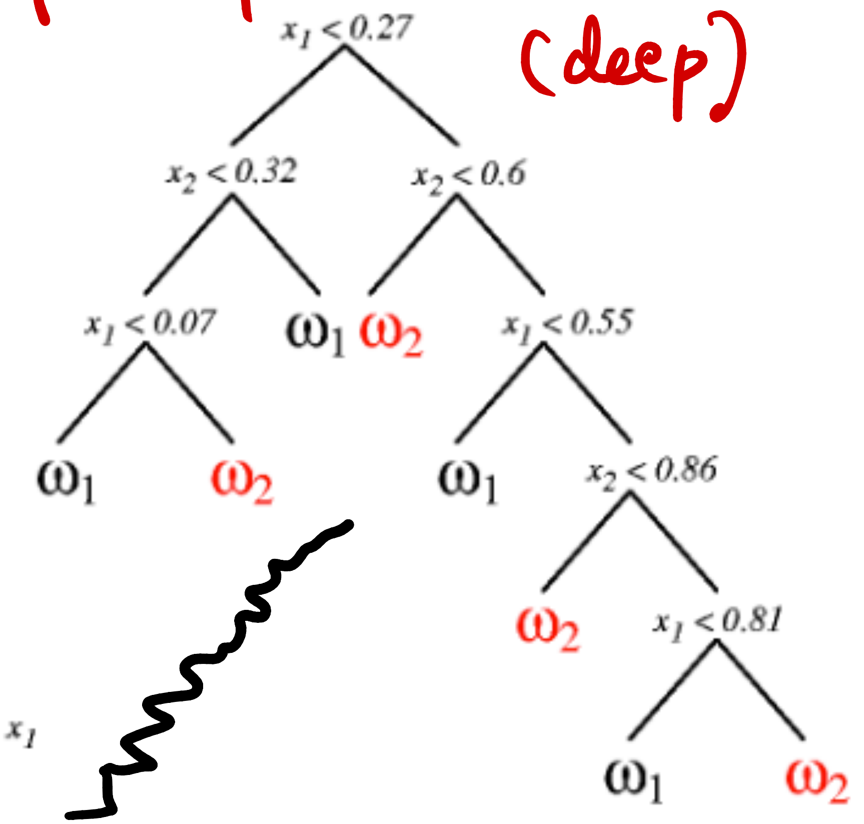


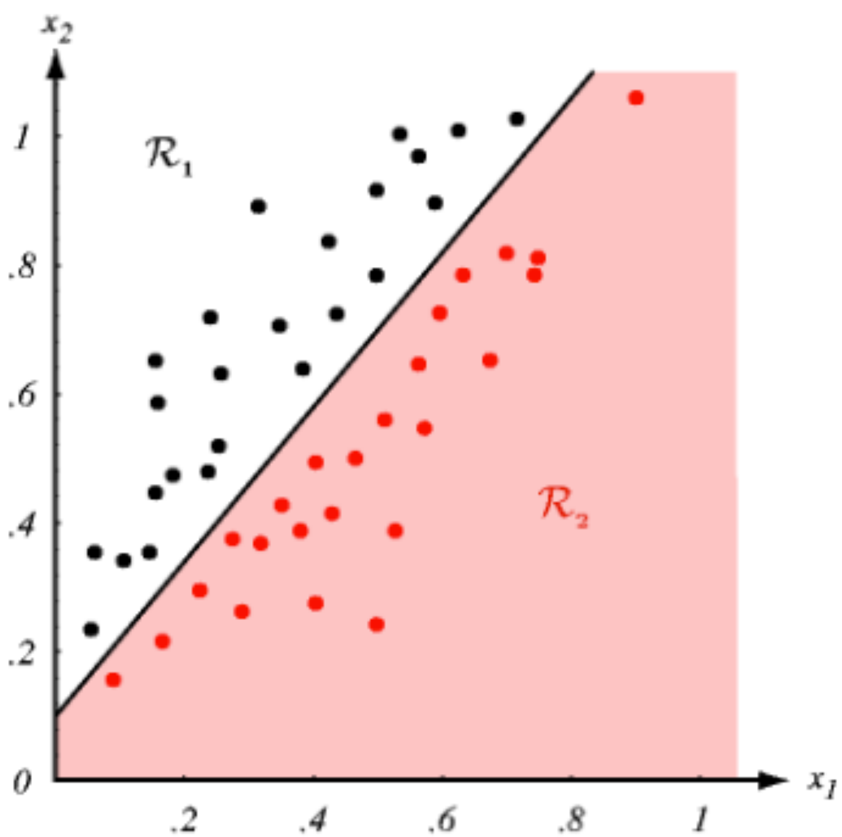
- x_1,x_2 = data features
- Each path in the tree corresponds to a region
- Deeper paths correspond to smaller regions

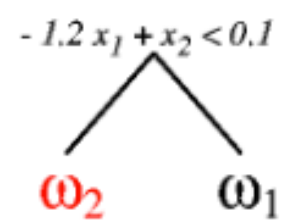
Data Partition Rules

| — splits ⇒ DT hard time (deep)

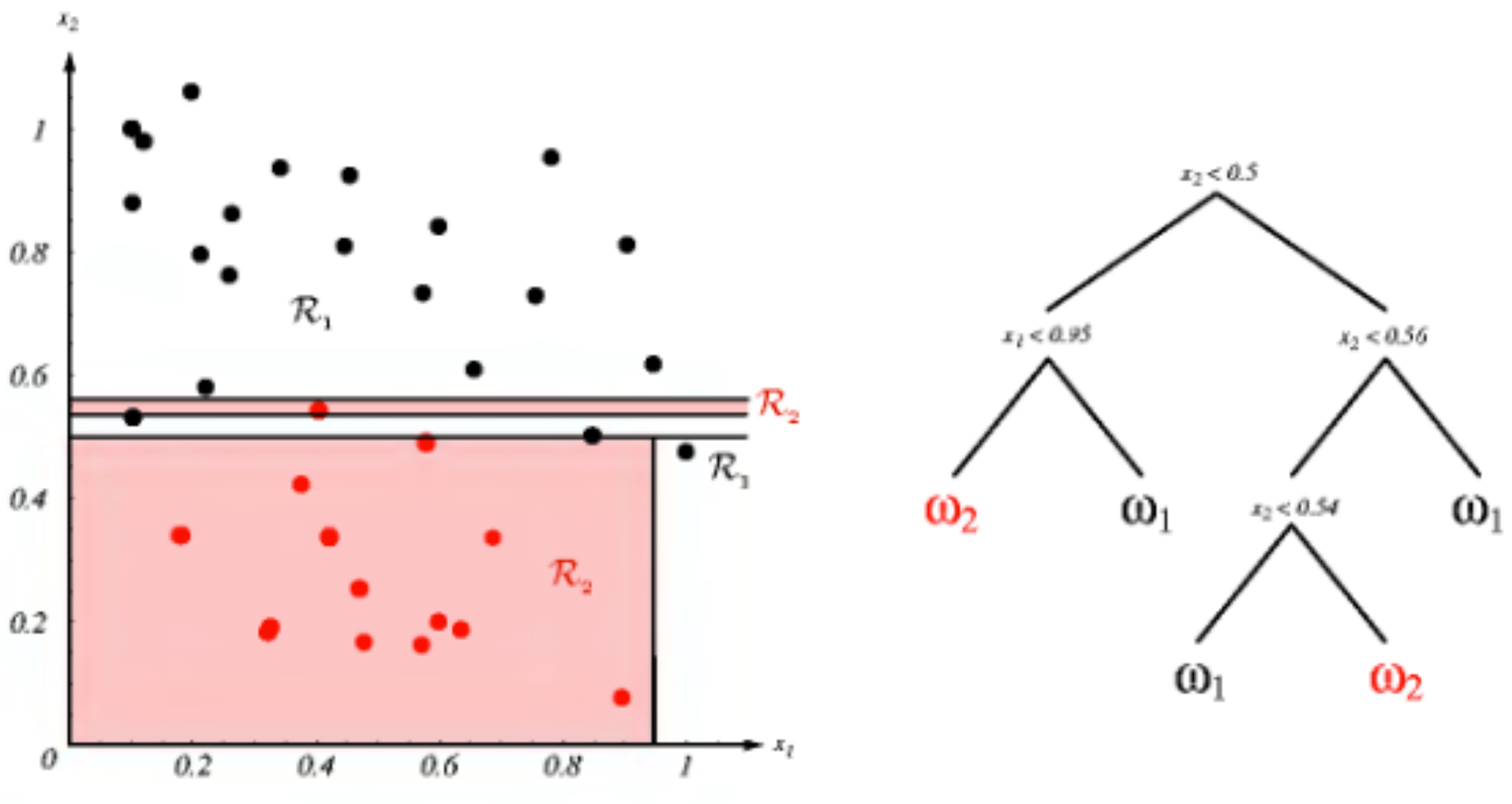


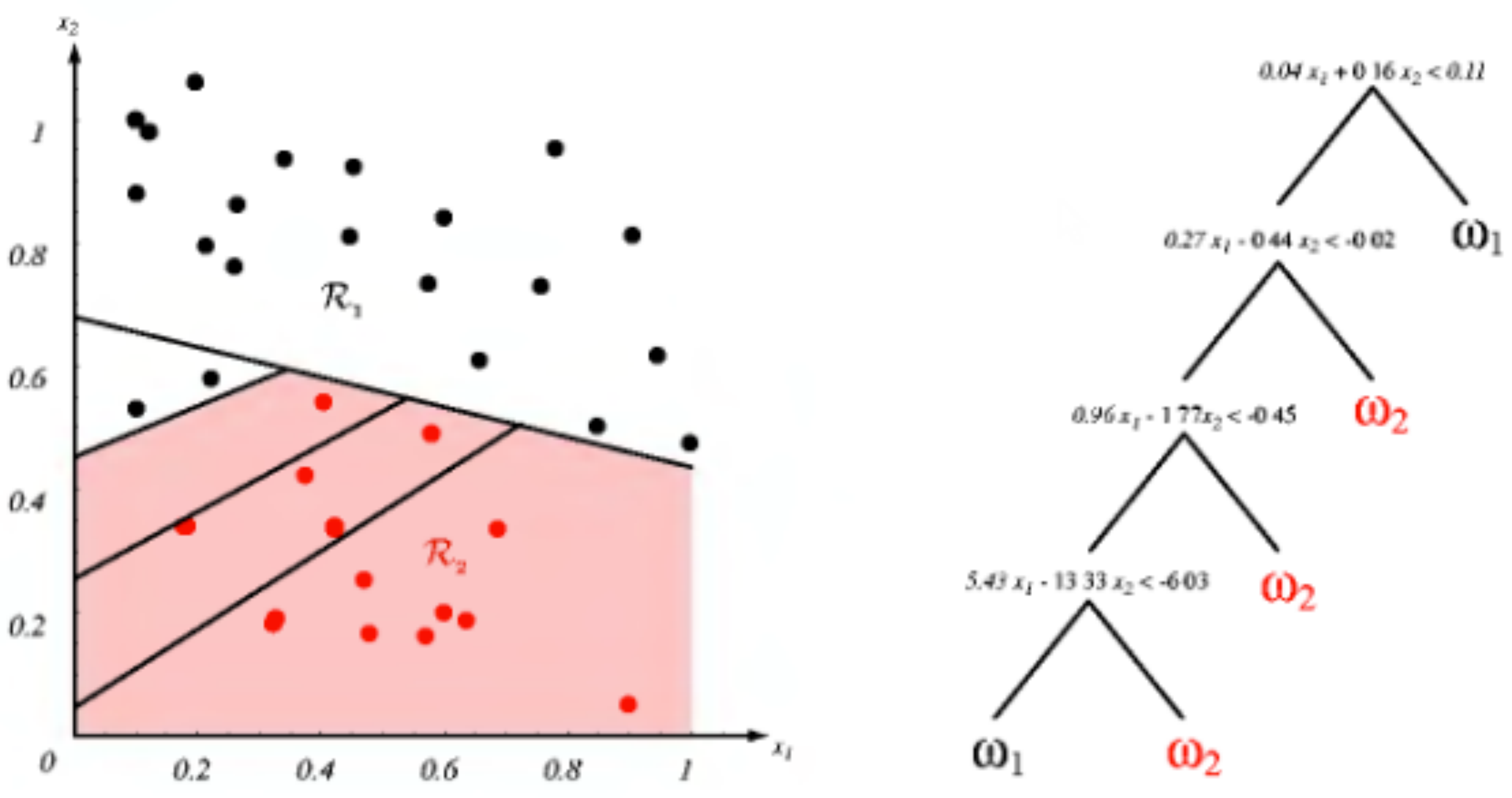






Data Partition Rules





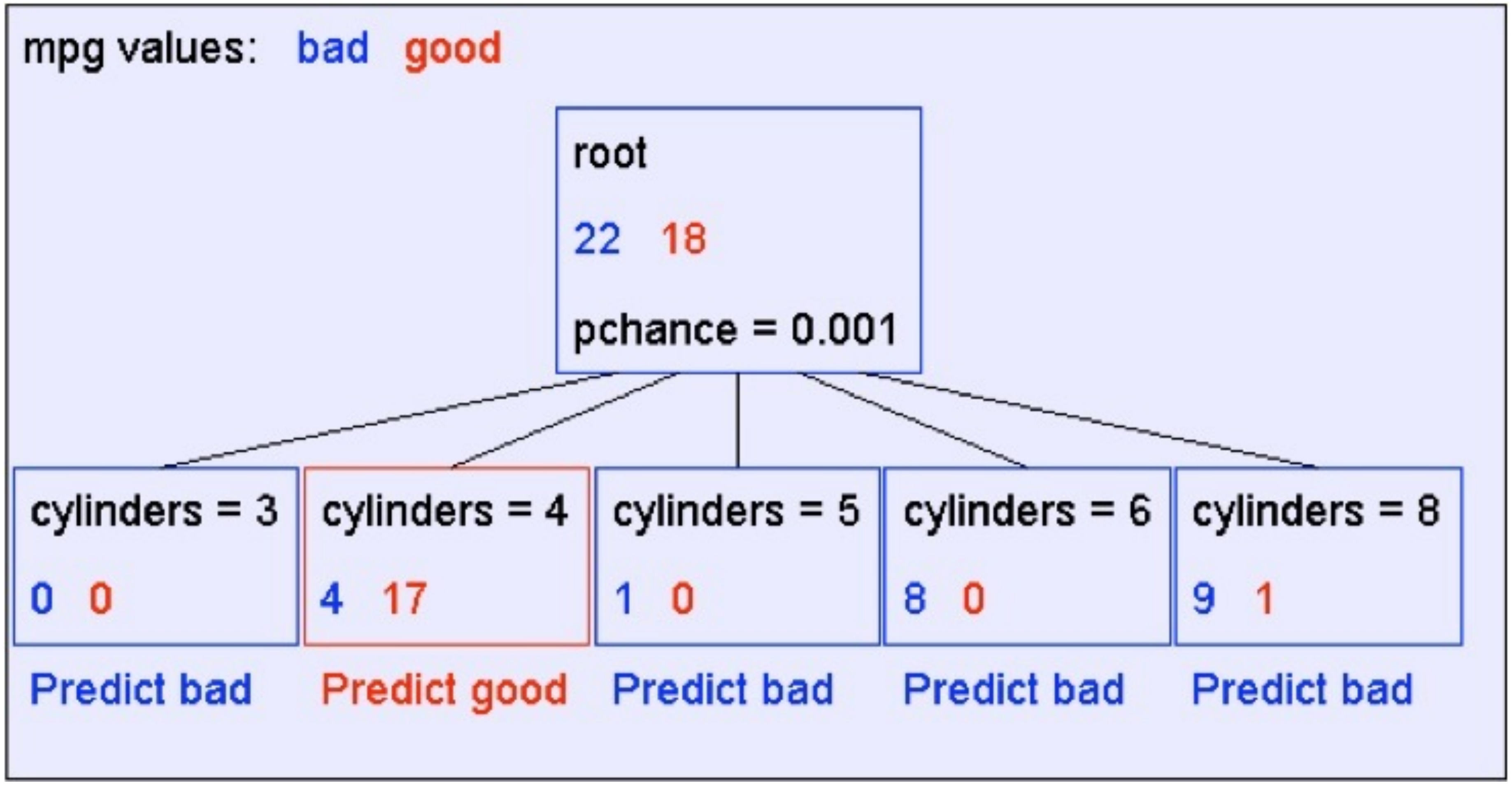
Walkthrough Decision Tree Example

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

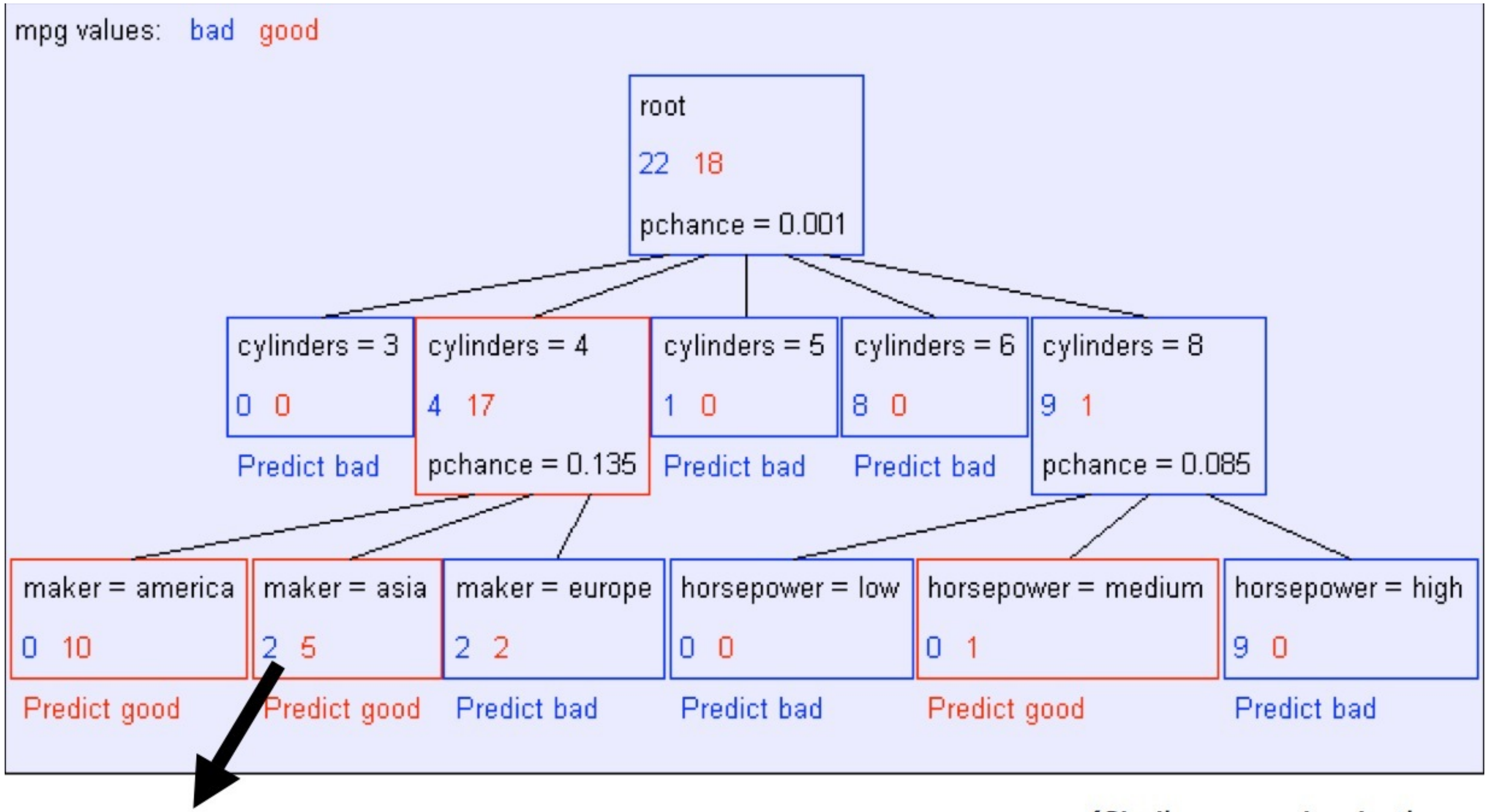
- Data (matrix) example : automobiles
- Target : mpg $\in$ {good, bad} - 2 class /binary problem

Decision Tree Split



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

Decision Tree Splits

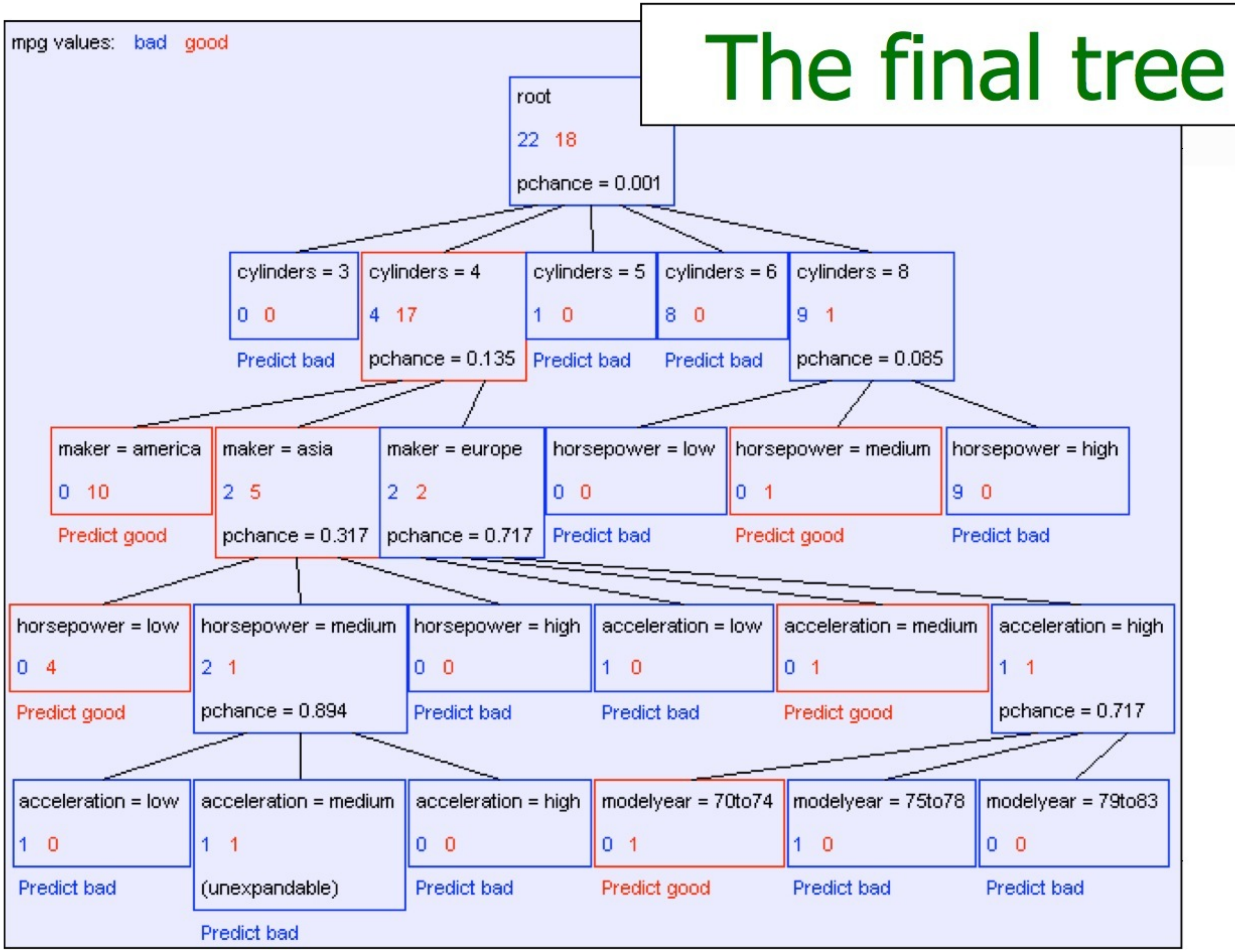


Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

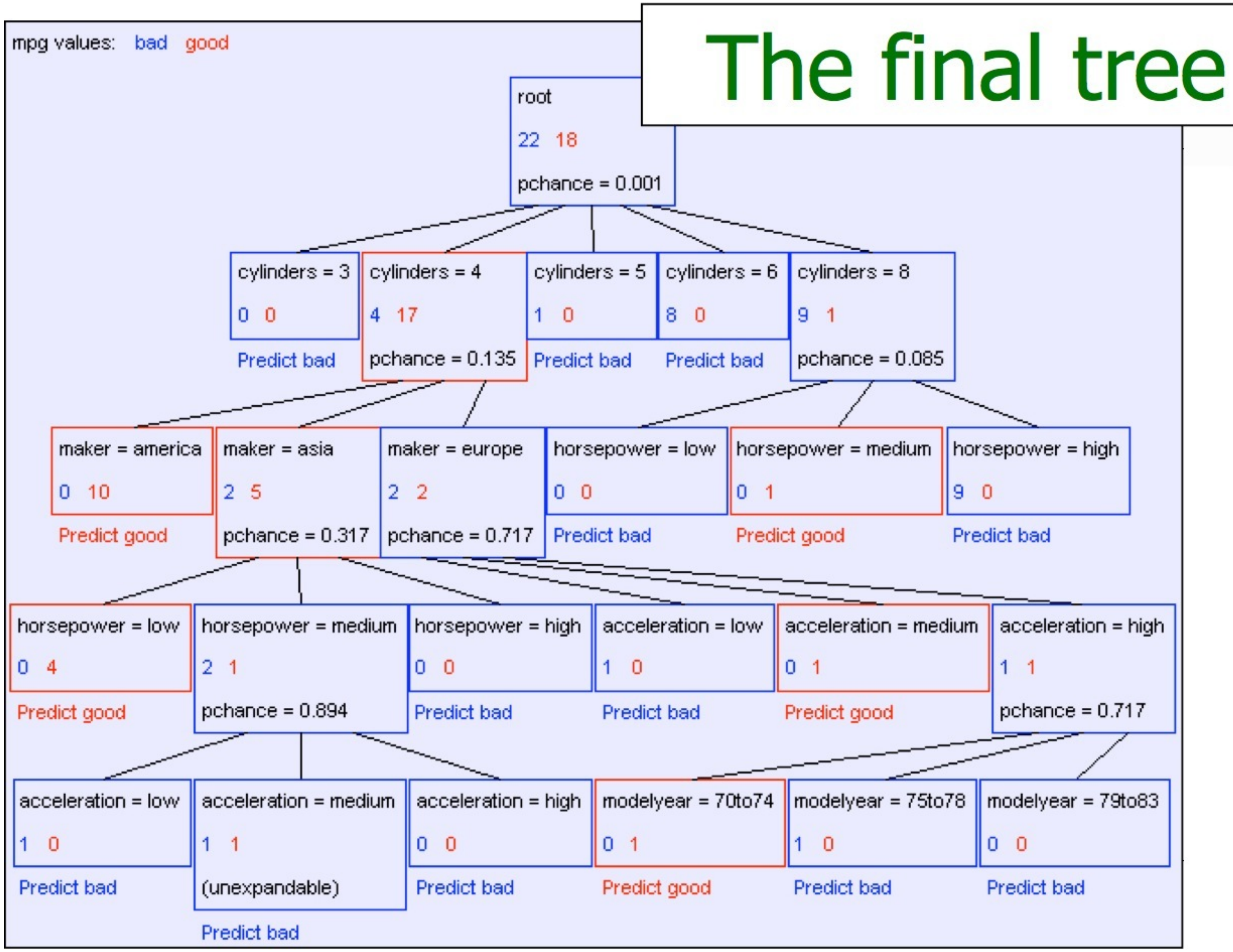
- each terminal leaf is labeled by majority (at that leaf). This leaf-label is used for prediction.

Decision Tree Splits



Prediction with a tree

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



Regression Tree

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

Estimate a predicted value per tree node

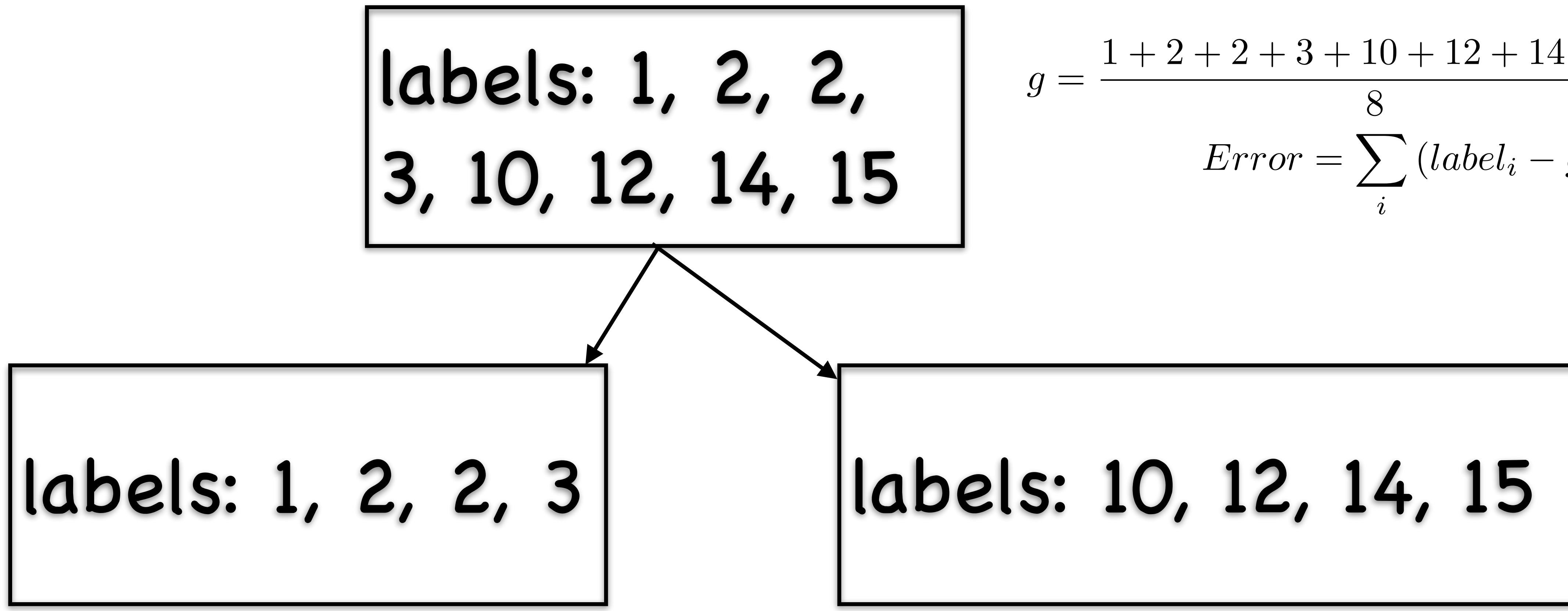
$$g_m = \frac{\sum_{t \in \chi_m} y_t}{|\chi_m|}$$

Calculate mean square error

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

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

Regression Tree



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

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

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

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

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

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

- 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