Machine Learning
CSE2510 – Lecture 6.1

Non-linear classification – Decision trees

Odette Scharenborg

TU Delft

Welcome to week 6 - lecture 1

- Administrative questions?
- Recap previous lecture

- What is non-linear classification?
- XOR problem
- Decision trees
- Splitting criterion
- Stop-splitting + class assignment
- Pruning

TU Delft

Administrative questions?

Recap of the previous lecture

Implicit bias

- A preference or inclination for or against something
- Based on learned coincidences, which unknowingly affect everyday perceptions, judgement, memory, and behaviour
- Subconscious thought
- We all have it
- Might result in discrimination

TU Delft

Multiple sources of bias in ML

- Training data
- Lack of diversity in ML developers
- Implicit human biases in our culture
- Evil programmers

Debiasing ML → Fairness in ML

- Fairness is a multi-faceted concept

To improve fairness:

- Debias training data
- Use unbiased features
- Build smart algorithms

But: is difficult to ensure on both the group and individual level simultaneously

Main points

- Bias can occur at any point in a system/ organization
- It can have a technical, societal, legal, and/or educational origin

➔ Building fairness and non-discriminatory behaviour into AI models is not only a matter of technological advantage but of social responsibility

Different types of classifiers

Machine learning:

- Supervised and unsupervised classifiers
- Parametric and non-parametric classifiers
- Generative and discriminative classifiers
- New dimension: linear and non-linear classifiers

Today: supervised & non-parametric & discriminative & non-linear classification

Today's learning objectives

After practicing with the concepts of today's lecture you are able to:

- Explain the basic concepts of non-linear classification
- Explain when and why they are used
- Explain the underlying algorithm of **decision trees**

Why non-linear classification

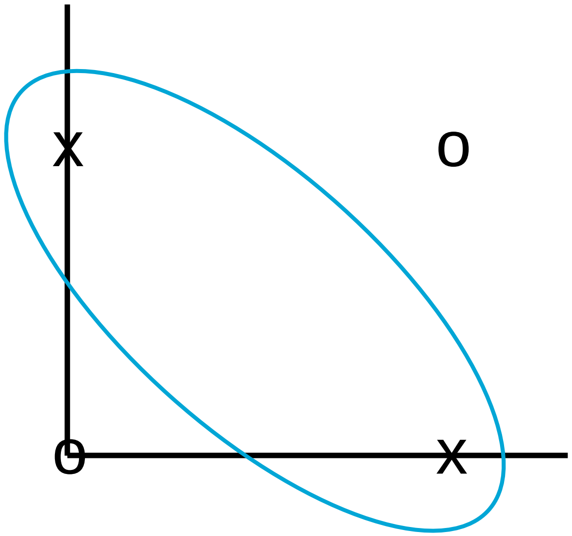
- Separate data with a decision boundary that is not a single or straight line

- Why needed?

XOR problem

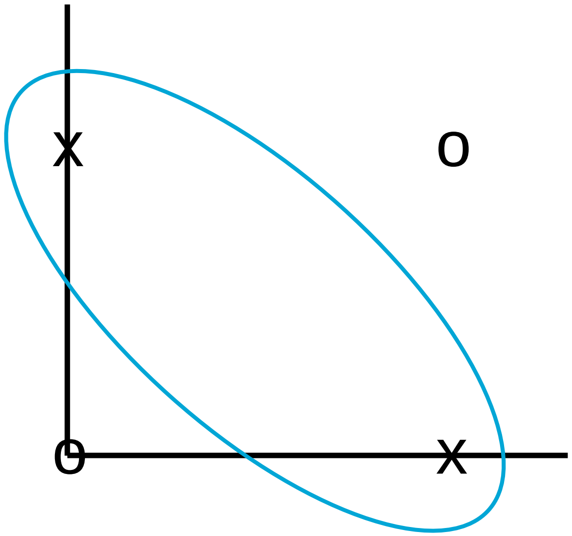
Where do you put the decision boundary to separate the two classes?

X = (0,1);(1,0)
O = (0,0);(1,1)



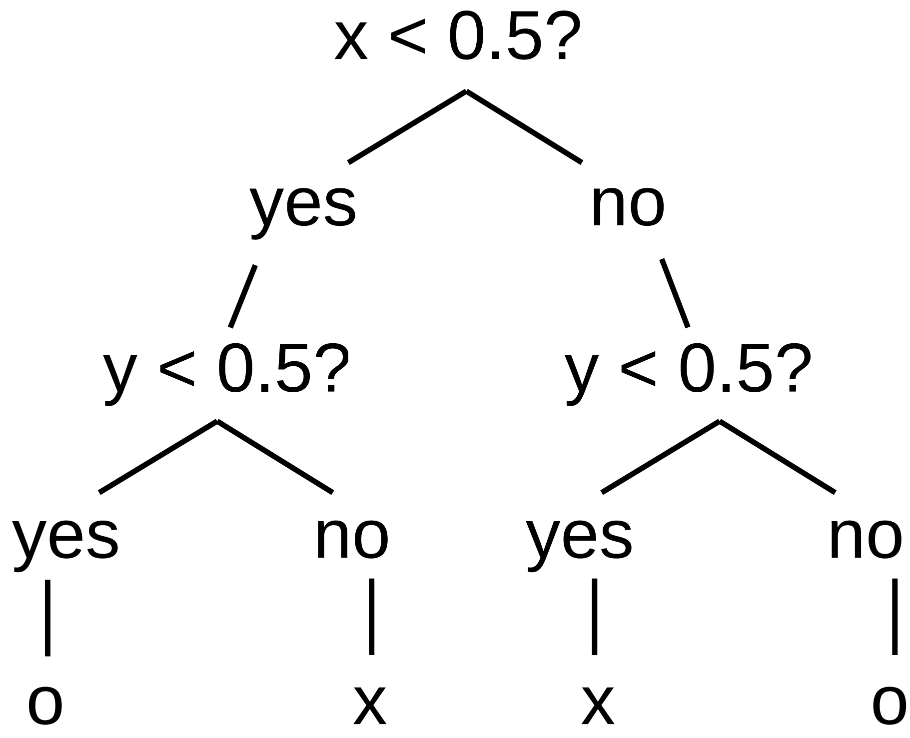
How can we separate the two classes?

X = (0,1);(1,0)
O = (0,0);(1,1)



Decision trees

- X = (0,1);(1,0)
- O = (0,0);(1,1)



Form of the decision boundary

- Linear classifier:
  - 2D: a line
  - Higher dimensions: a hyperplane

- Nonlinear classifier:
  - Locally, it can be linear
  - In general, has a complex shape

Different non-linear classifiers

- Decision trees (today)
- Multi-layer perceptrons (Friday)

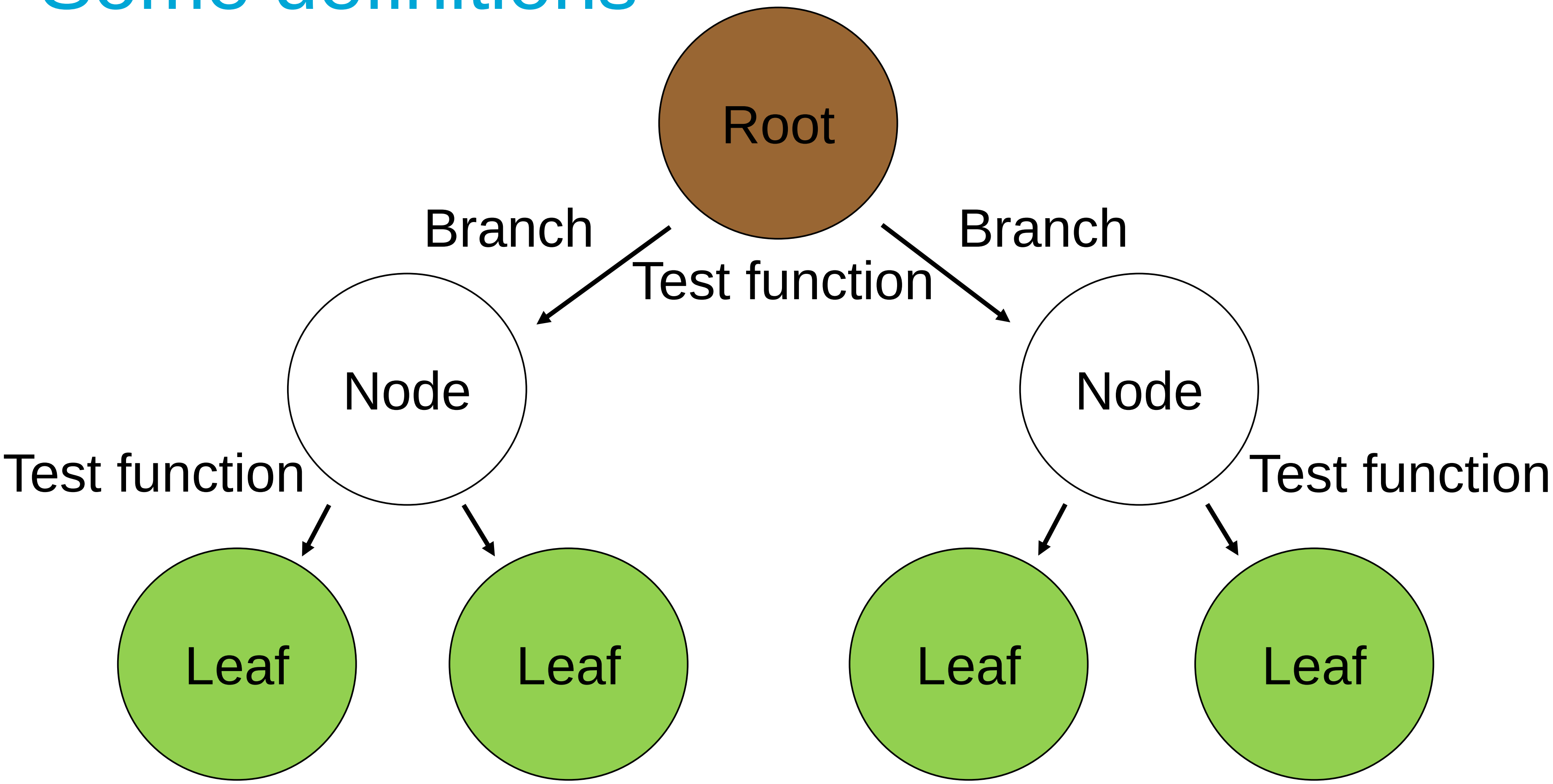
Decision trees

Largely based on slides from Victor Lavrenko, 2011

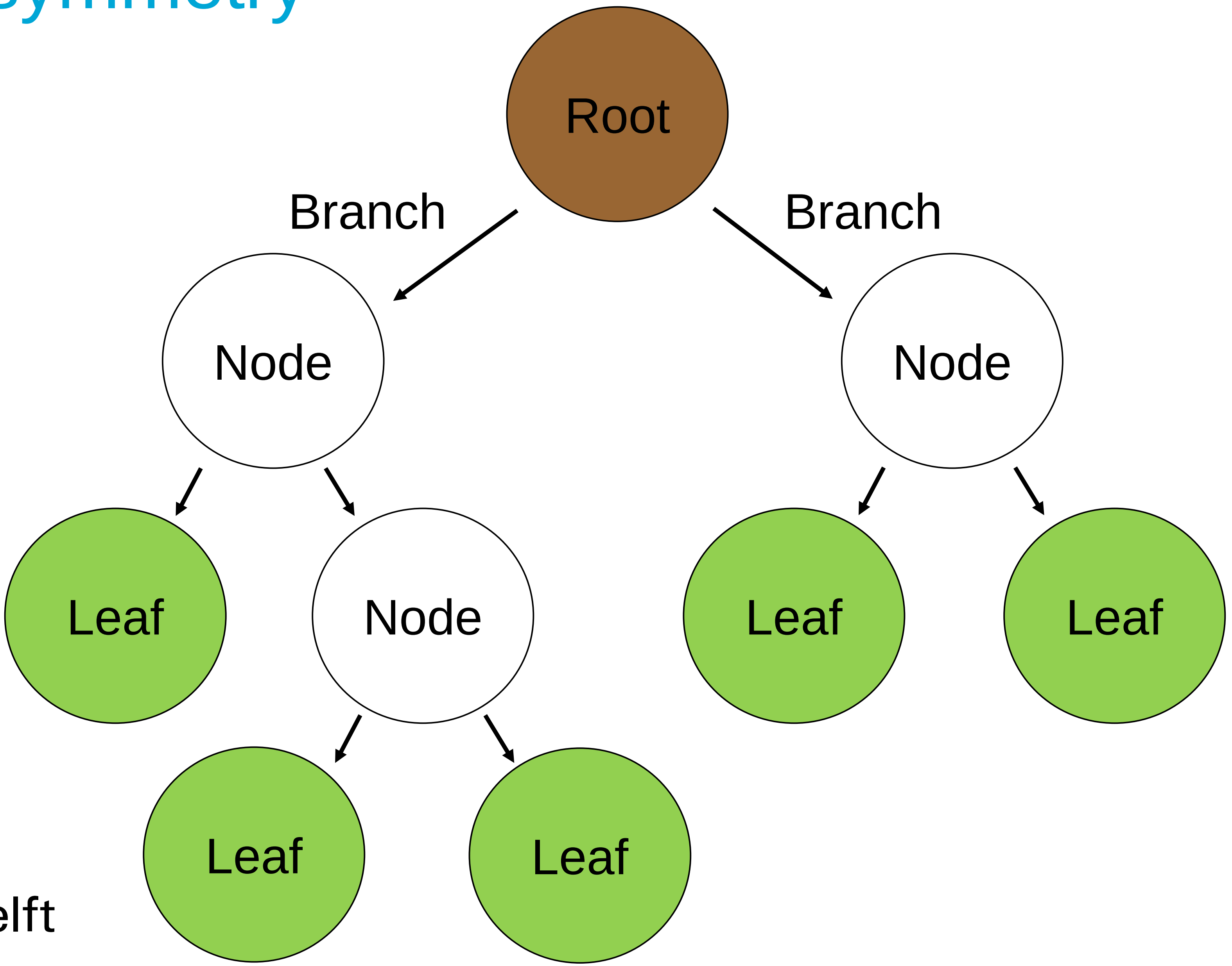
Introduction to Applied Machine Learning

University of Edinburgh, UK

Some definitions



Asymmetry



Decision trees

- Split the training data into unique regions *sequentially*
- Structure of the tree is not predefined
- Structure grows depending on the complexity and structure of the training data
- At every node, a decision is made which splits the training data in smaller subsets

Predict if John will play tennis

- Divide & conquer:
  - split into subsets
  - are they pure? (all yes or all no)
  - if yes: stop
  - if not: repeat
- See which subset new data falls into

Training examples: **9 yes / 5 no**

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

New data:

Day	Outlook	Humidity	Wind	Play
D15	Rain	High	Weak	?

9 yes / 5 no

Outlook

Overcast

Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

4 yes / 0 no

pure subset

Sunny

Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

2 yes / 3 no

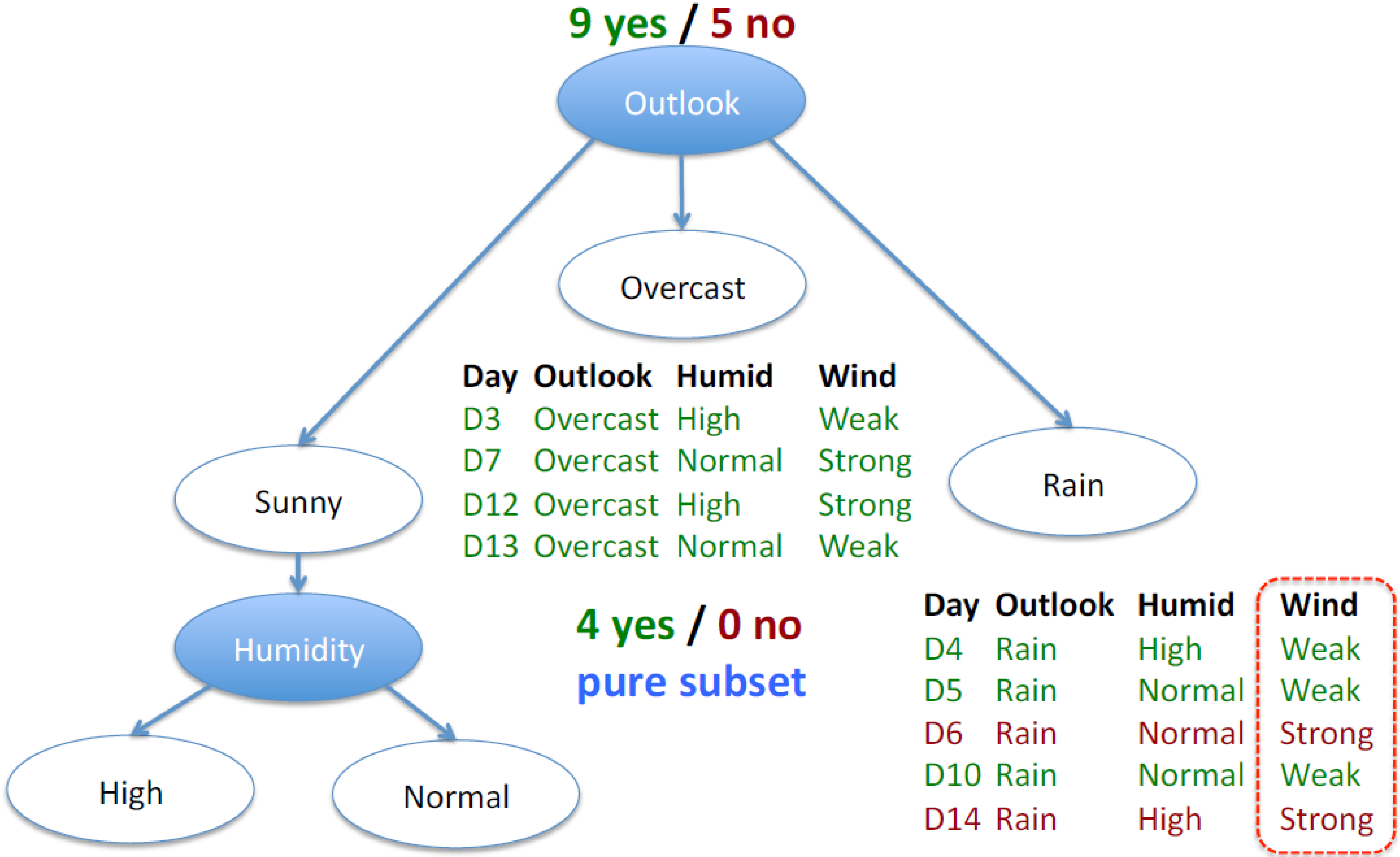
split further

Rain

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

3 yes / 2 no

split further



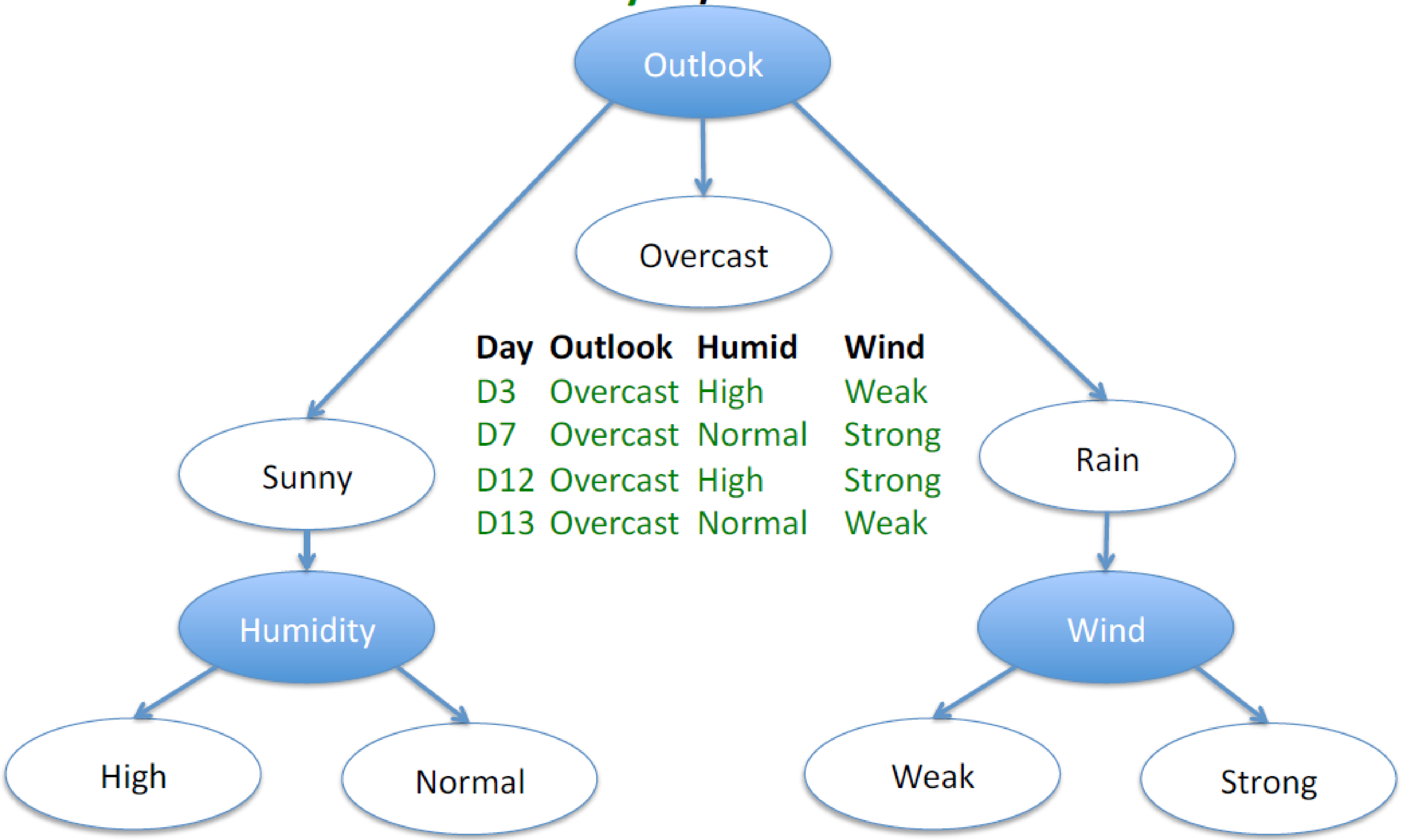
Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

Day	Humid	Wind
D1	High	Weak
D2	High	Strong
D8	High	Weak

Day	Humid	Wind
D9	Normal	Weak
D11	Normal	Strong

9 yes / 5 no



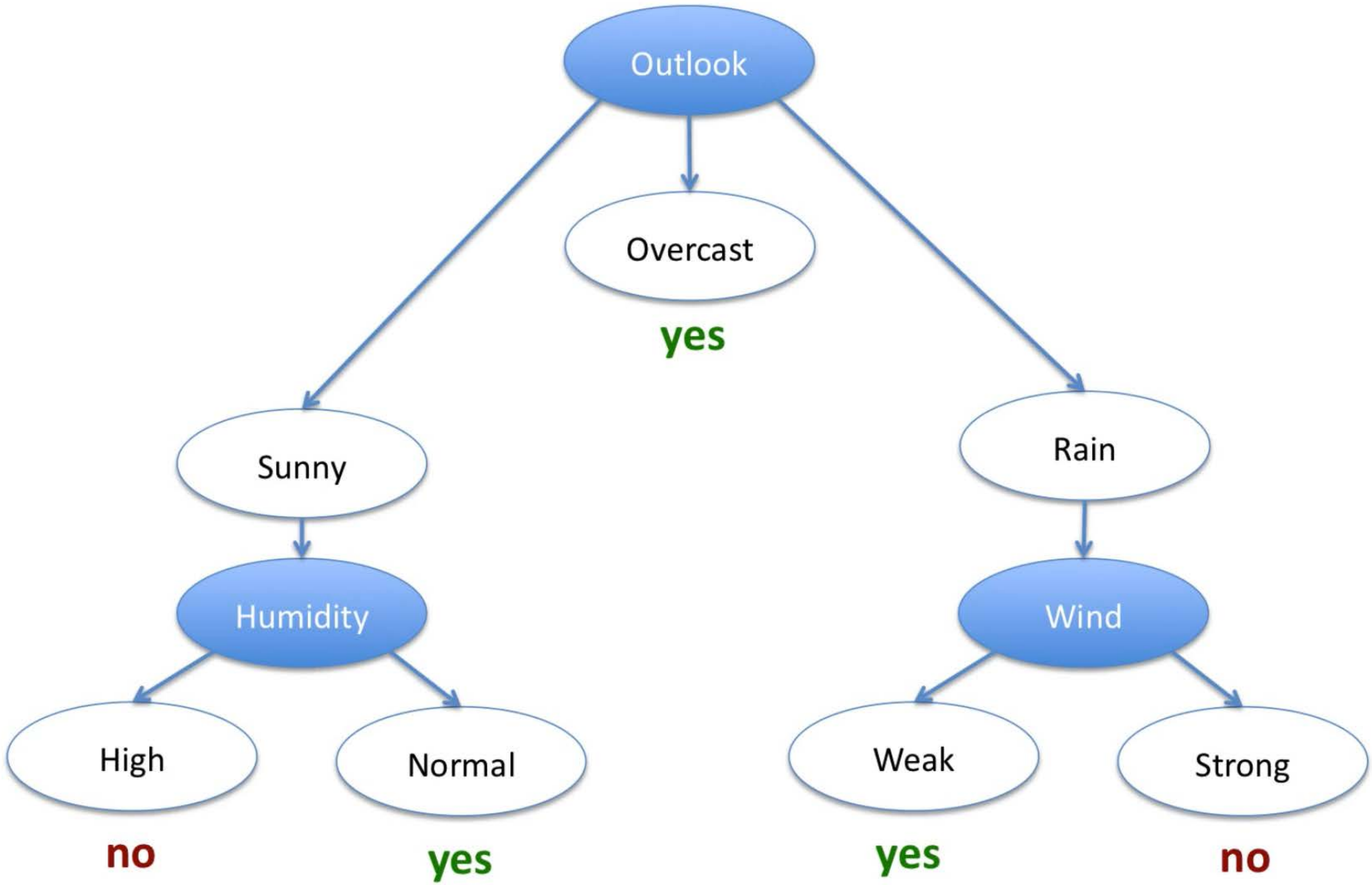
Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

Day	Humid	Wind
D1	High	Weak
D2	High	Strong
D8	High	Weak

Day	Humid	Wind
D9	Normal	Weak
D11	Normal	Strong

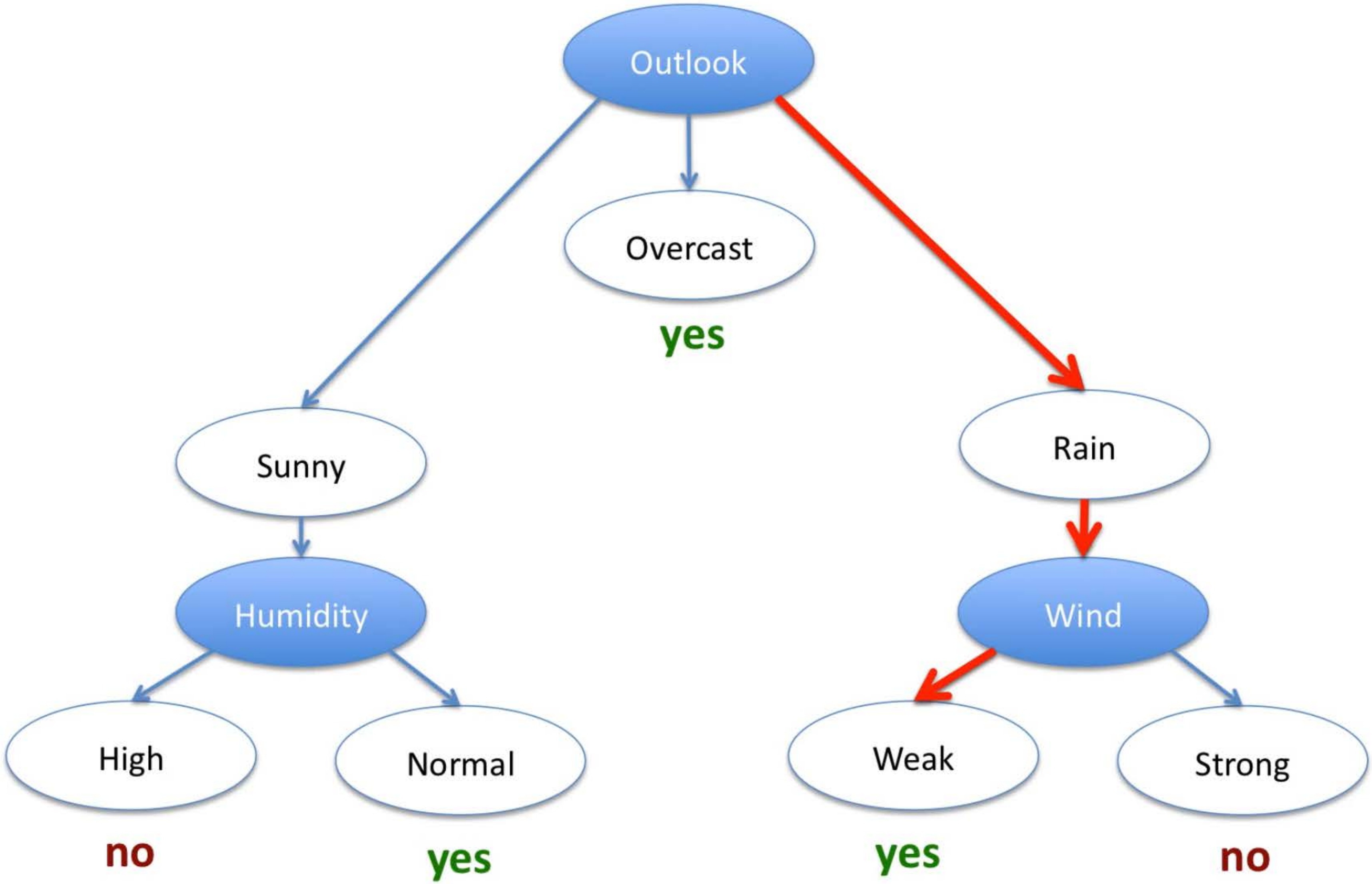
Day	Humid	Wind
D4	High	Weak
D5	Normal	Weak
D10	Normal	Weak

Day	Humid	Wind
D6	Normal	Strong
D14	High	Strong



New data:

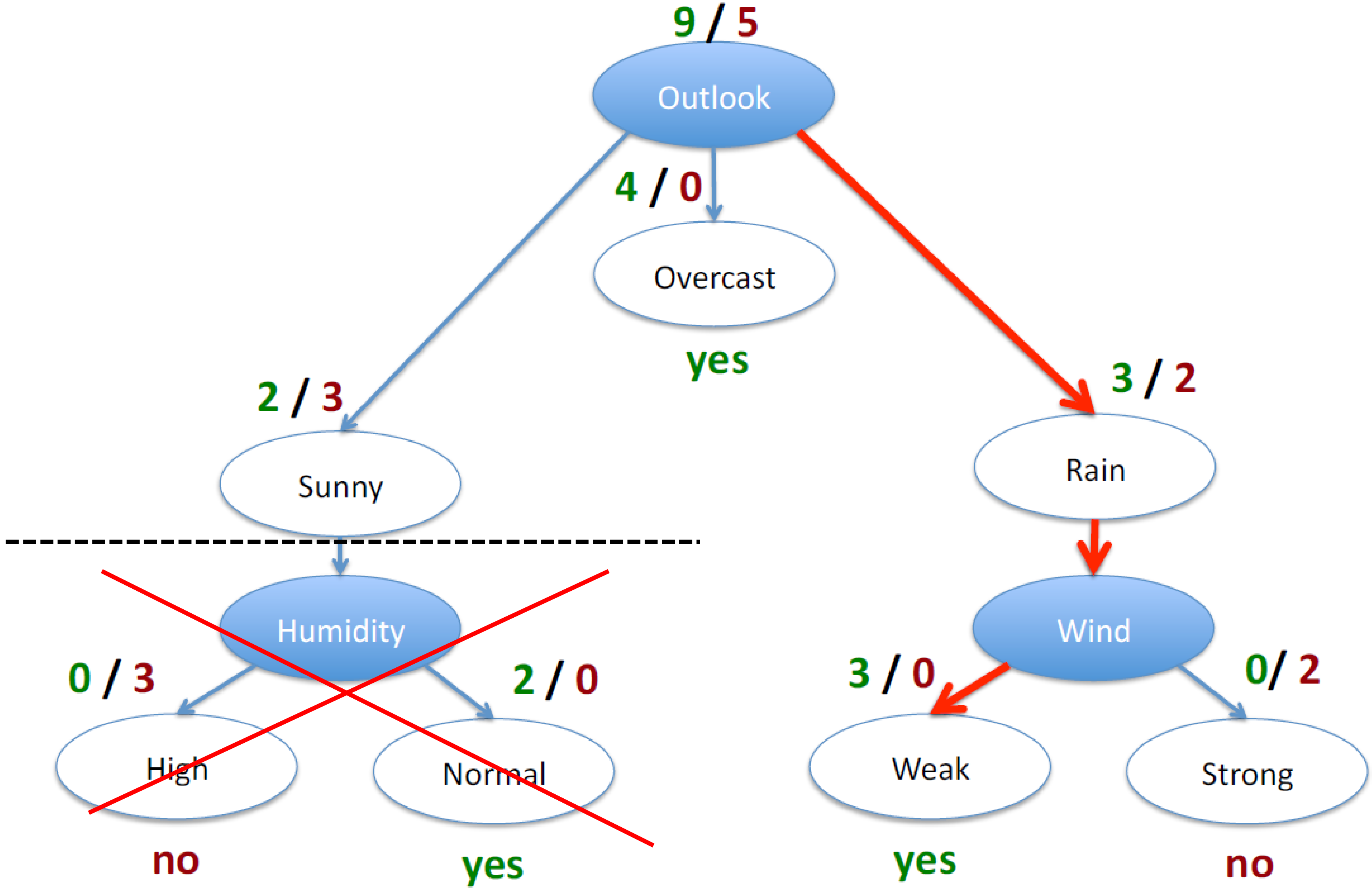
Day	Outlook	Humid	Wind
D15	Rain	High	Weak



New data:

Day	Outlook	Humid	Wind	
D15	Rain	High	Weak	→ Yes

TU Delft



New data:

Day	Outlook	Humid	Wind
D15	Rain	High	Weak

ID3 algorithm

- Split (node, {examples}):
  1. A ← the best attribute for splitting the {examples}
  2. Decision attribute for this node ← A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. If examples perfectly classified: STOP
     else: iterate over new child nodes
     Split (child_node, {subset of examples})
- Ross Quinlan (ID3: 1986), (C4.5: 1993)
- Breimanetal (CaRT: 1984) from statistics

ID3 algorithm

- Split (node, {examples}):
  1. A ← the $\boxed{\text{best attribute}}$ for splitting the {examples}
  2. Decision attribute for this node ← A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. If examples perfectly classified: STOP
     else: iterate over new child nodes
     Split (child_node, {subset of examples})
- Ross Quinlan (ID3: 1986), (C4.5: 1993)
- Breimanetal (CaRT: 1984) from statistics

How to find the best attribute

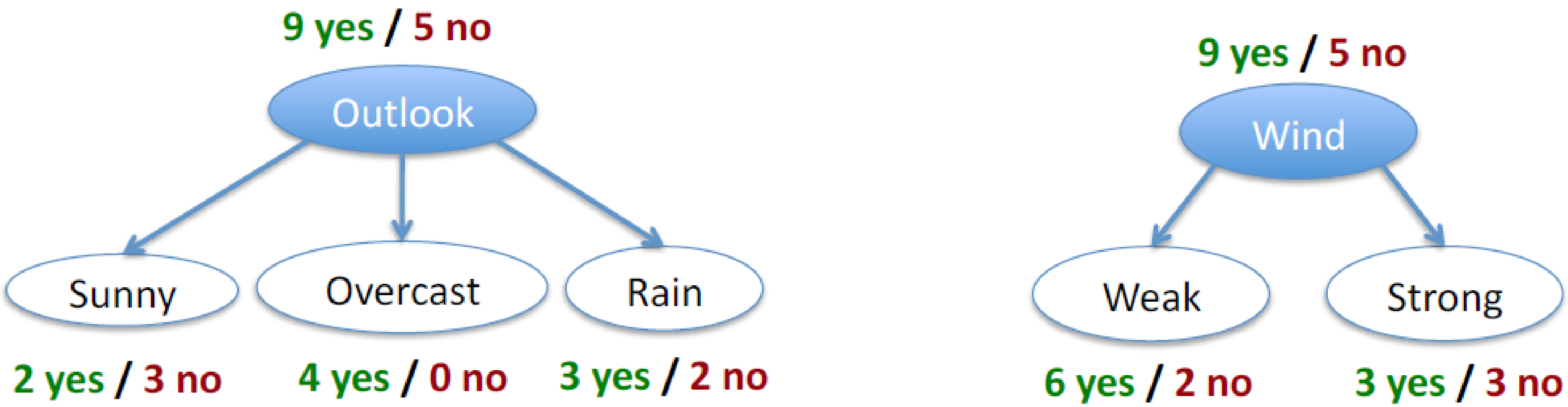
- What attribute to split on?

- Splitting criterion

- Stop-splitting

➔ Done? Assign label to class
    - E.g., majority rule

Which attribute to split on?



- Want to measure "purity" of the split
  - more certain about Yes/No after the split
    - pure set (4 yes / 0 no) => completely certain (100%)
    - impure (3 yes / 3 no) => completely uncertain (50%)
  - can't use P("yes" | set):
    - must be symmetric: 4 yes / 0 no as pure as 0 yes / 4 no

TU Delft

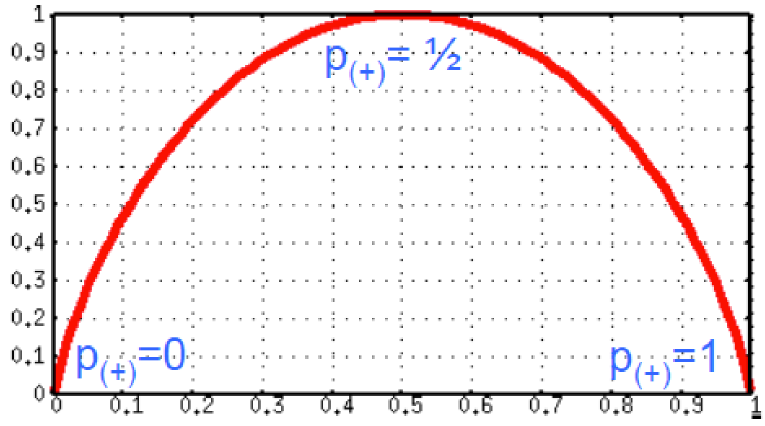
Splitting criterion: Entropy

- Entropy: $H(S) = - p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$ bits
  - S … subset of training examples
  - $p_{(+)}$ / $p_{(-)}$ … % of positive / negative examples in S
- Interpretation: assume item X belongs to S
  - how many bits need to tell if X positive or negative
- impure (3 yes / 3 no):

$$H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1 \text{ bits}$$

- pure set (4 yes / 0 no):

$$H(S) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0 \text{ bits}$$



TUDelft

Stop-splitting: Information gain

- Want many items in pure sets
- Expected drop in entropy after split:

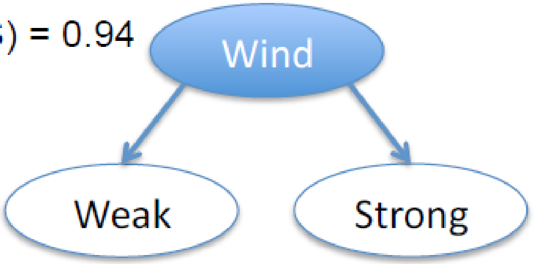
$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

V … possible values of A
S … set of examples {X}
S_V … subset where $X_A = V$

- Mutual Information
  - between attribute A and class labels of S

$-\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$ **9 yes / 5 no**

H(S) = 0.94



6 yes / 2 no | **3 yes / 3 no**

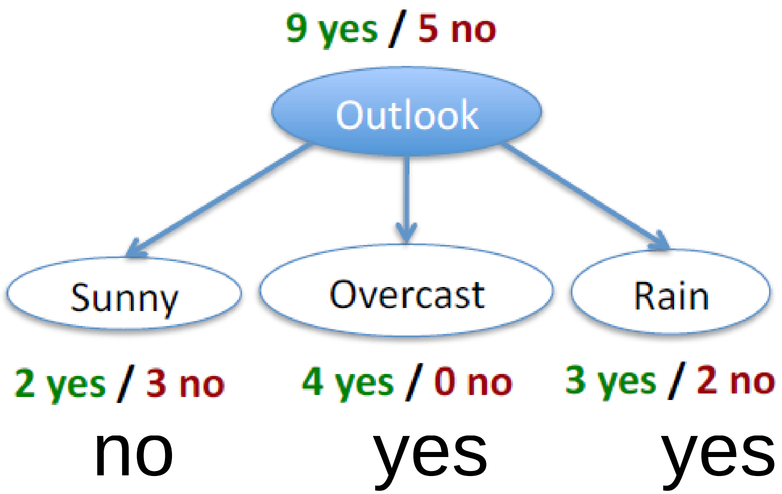
$-\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8}$ | $-\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6}$

$H(S_{weak}) = 0.81$ | $H(S_{strong}) = 1.0$

Gain (S, Wind)
$= H(S) - {}^{8}/_{14}\, H(S_{weak}) - {}^{6}/_{14}\, H(S_{weak})$
$= 0.94 - {}^{8}/_{14} * 0.81 - {}^{6}/_{14} * 1.0$
$= 0.049$

Class assignment

- Rule needed to assign each leaf to a class



Procedure

- Test all attributes to find the split that gives the highest information gain
- Repeat process for the child nodes
- Until all nodes are leafs

Overfitting in decision trees

- Can always classify training examples perfectly
  - keep splitting until each node contains 1 example
  - singleton = pure
- Doesn't work on new data

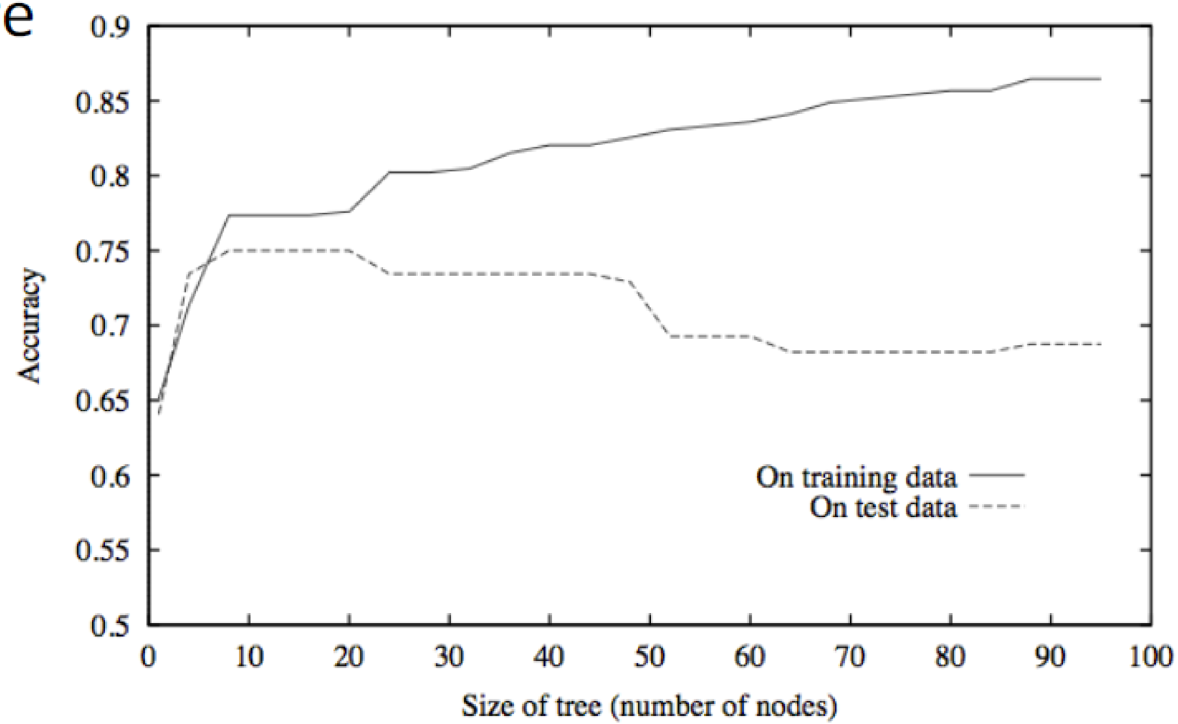
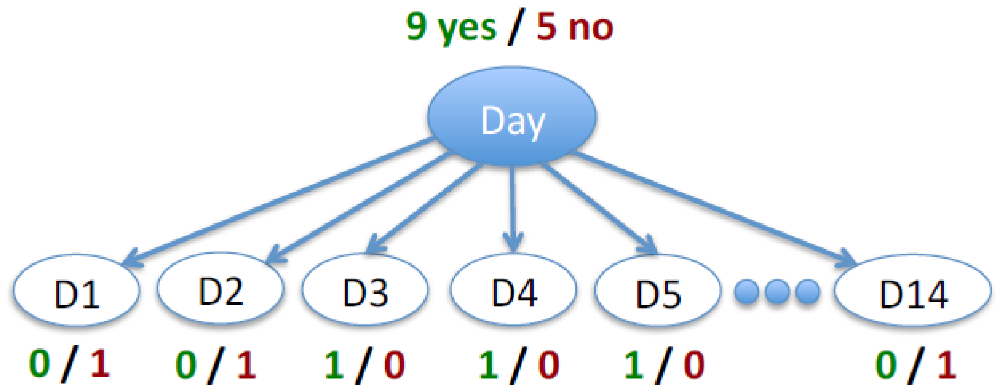


Figure credit: Tom Mitchell, 1997

Avoid overfitting: pruning

- Stop splitting when not statistically significant
- Grow, then post-prune
  - based on validation set
- Sub-tree replacement pruning (WF 6.1)
  - for each node:
    - pretend remove node + all children from the tree
    - measure performance on validation set
  - remove node that results in greatest improvement
  - repeat until further pruning is harmful

Problems with information gain



all subsets perfectly pure => optimal split

- Biased towards attributes with many values
- Won't work for new data: D15 Rain High Weak
- Use GainRatio:

$$SplitEntropy(S,A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}$$

A … candidate attribute
V … possible values of A
S … set of examples {X}
S_V … subset where $X_A = V$

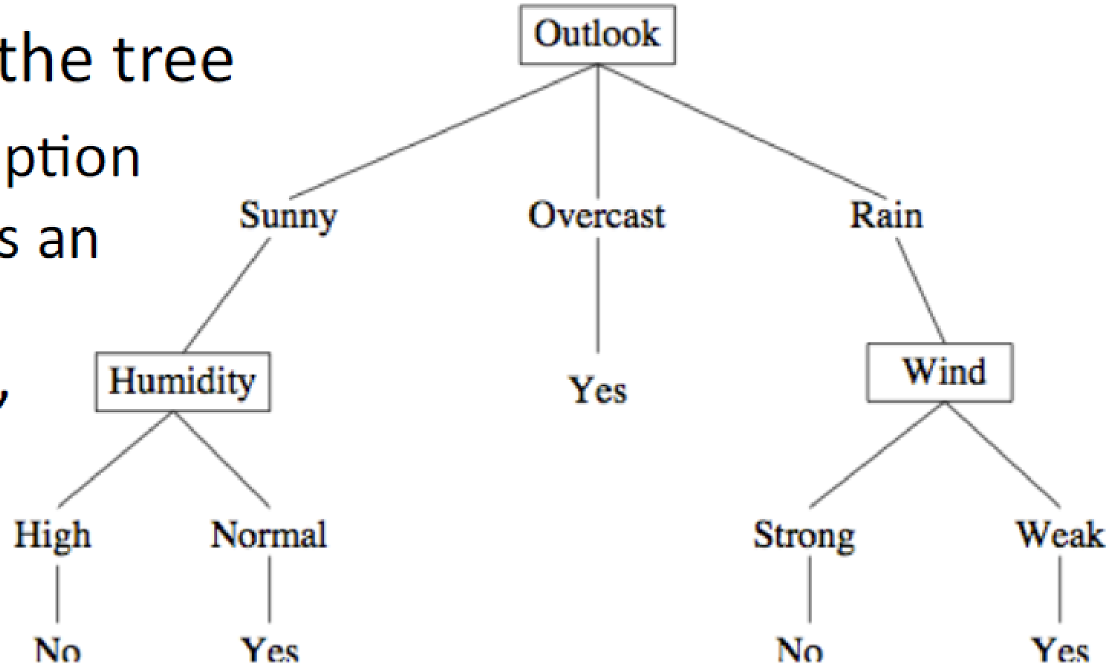
$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

penalizes attributes with many values

TU Delft

Trees are interpretable

- Read rules off the tree
  - concise description of what makes an item positive
- No “black box”
  - important for users



Rule: (Outlook = Overcast) $\vee$
(Outlook = Rain $\wedge$ Wind = Weak) $\vee$
(Outlook = Sunny $\wedge$ Humidity = Normal)

Figure credit: Tom Mitchell, 1997

Continuous attributes

- Dealing with continuous-valued attributes:
  - create a split: (Temperature > 72.3) = True,False
- Threshold can be optimized

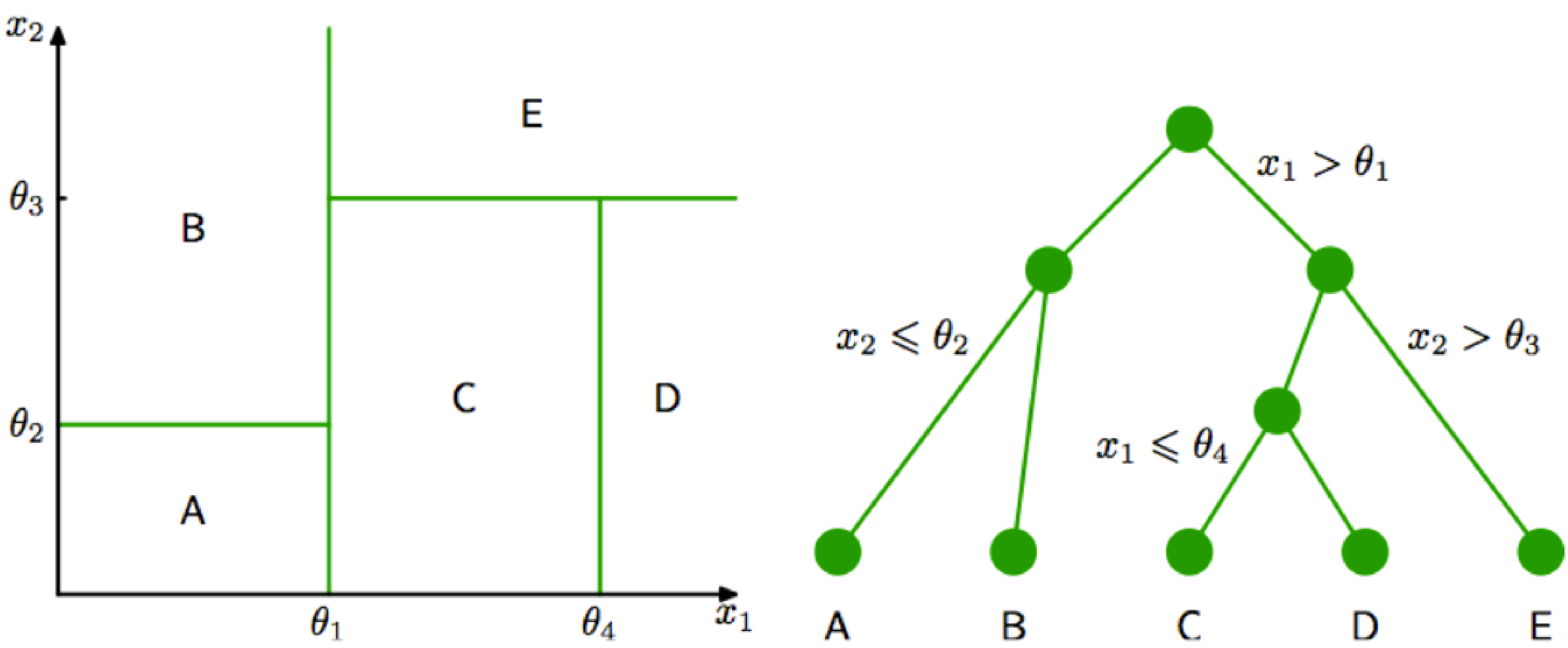


Figure credit: Chris Bishop, PRML

TUDelft

Summary: decision trees

- If the data cannot be split with a single, straight decision boundary, non-linear classifiers can be used

Decision trees

- Grow from the root down
  - Greedily selects next best attribute
- Searches a complete hypothesis space
  - Prefers smaller trees, high gain at the root
- Overfitting addressed by post-pruning
  - Prune models, while accuracy ⇧ on validation set

TU Delft