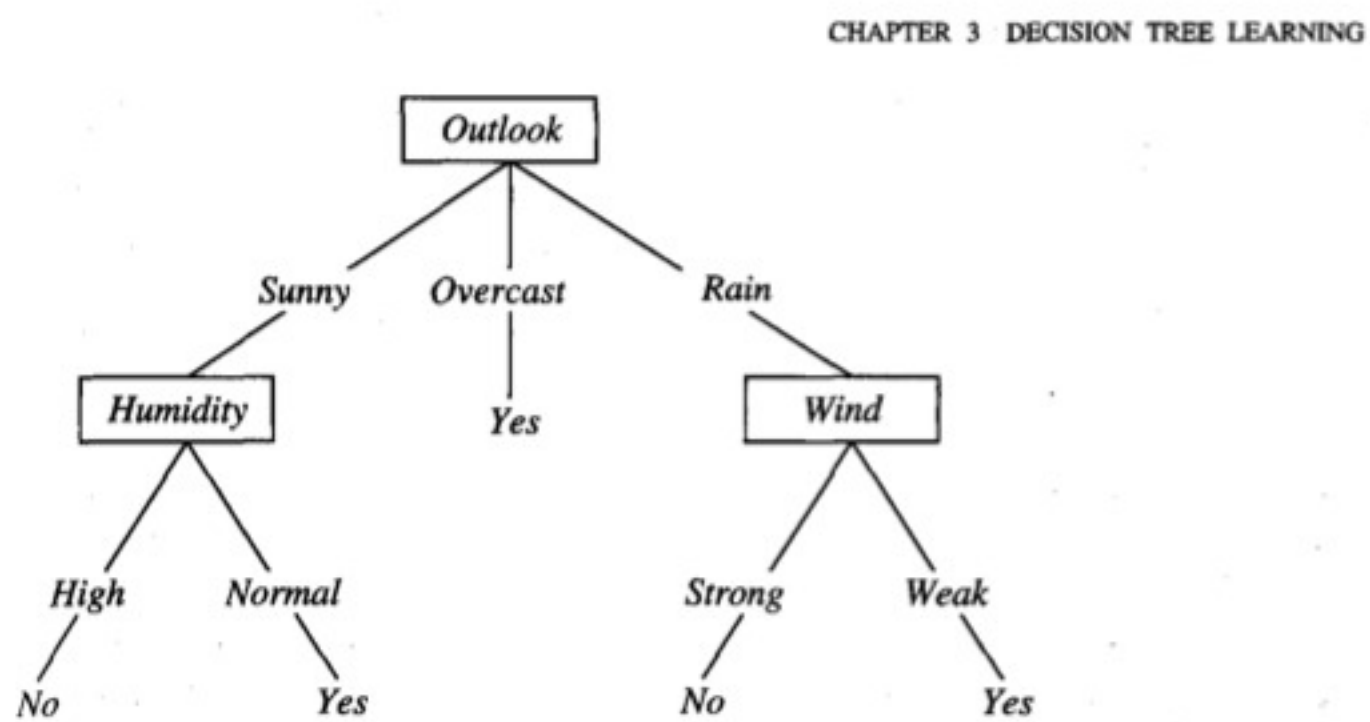
Decision Trees

6601

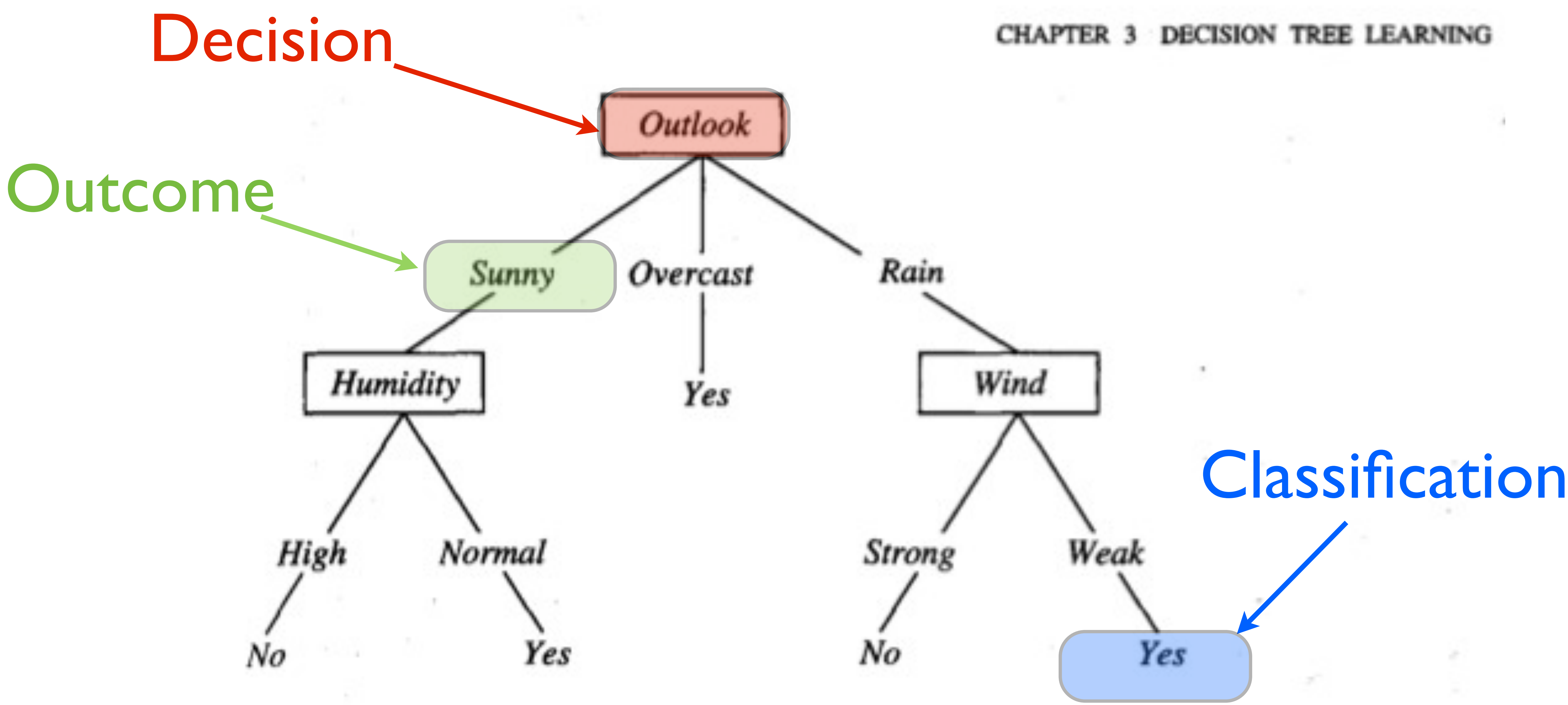


Classification

A_1	A_2	...	A_N	C
v_{11}	v_{12}	...	v_{1N}	c_1
v_{21}	v_{22}	...	v_{2N}	c_2

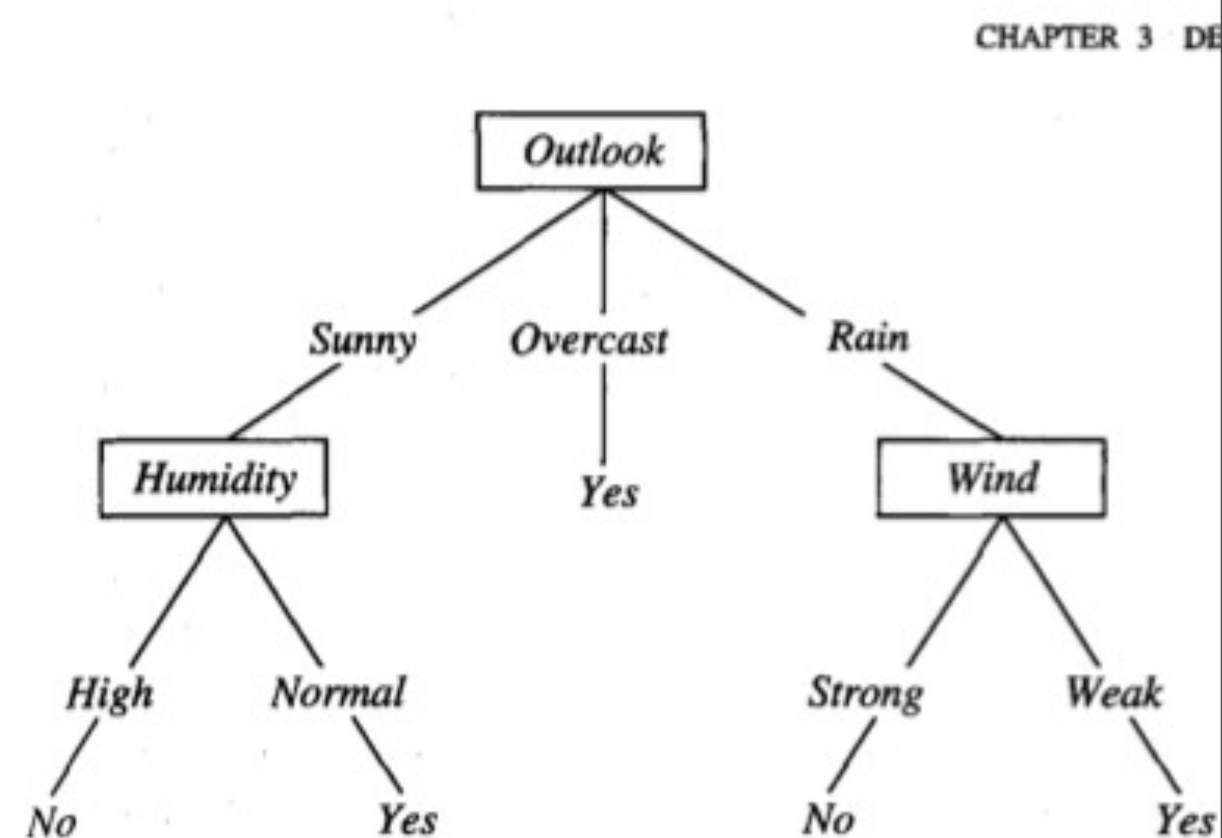
...

Decision Tree



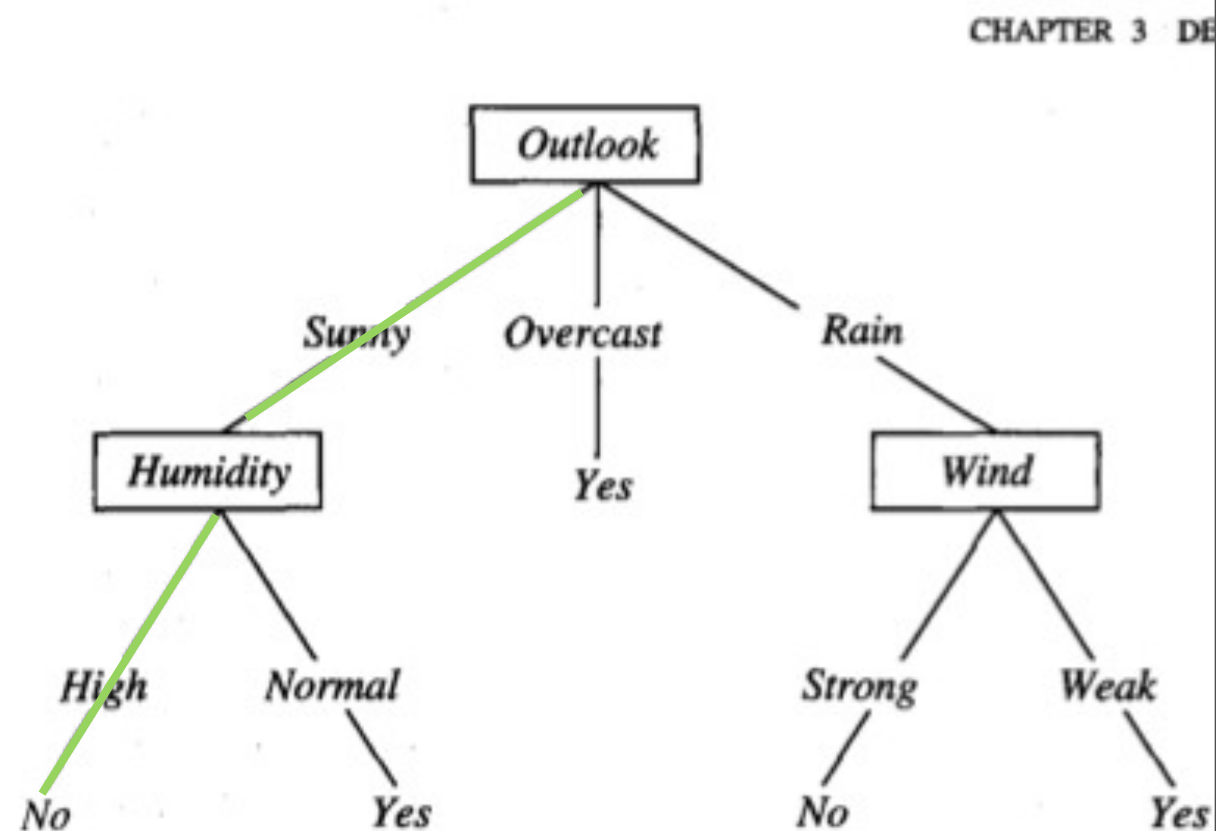
Decision Tree

H	W	O	P
H	S	S	Y/N?
N	S	O	Y/N?
H	W	R	Y/N?



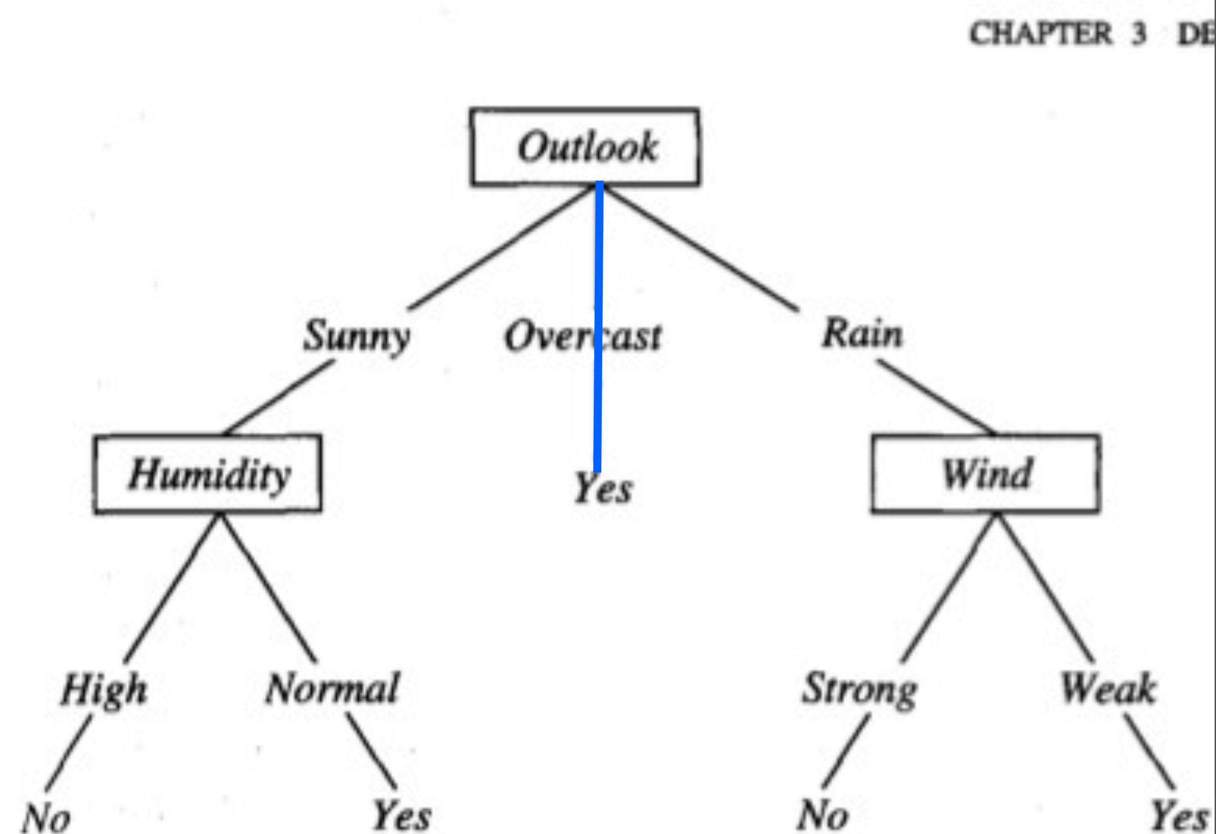
Decision Tree

H	W	O	P
H	S	S	N
N	S	O	Y/N?
H	W	R	Y/N?



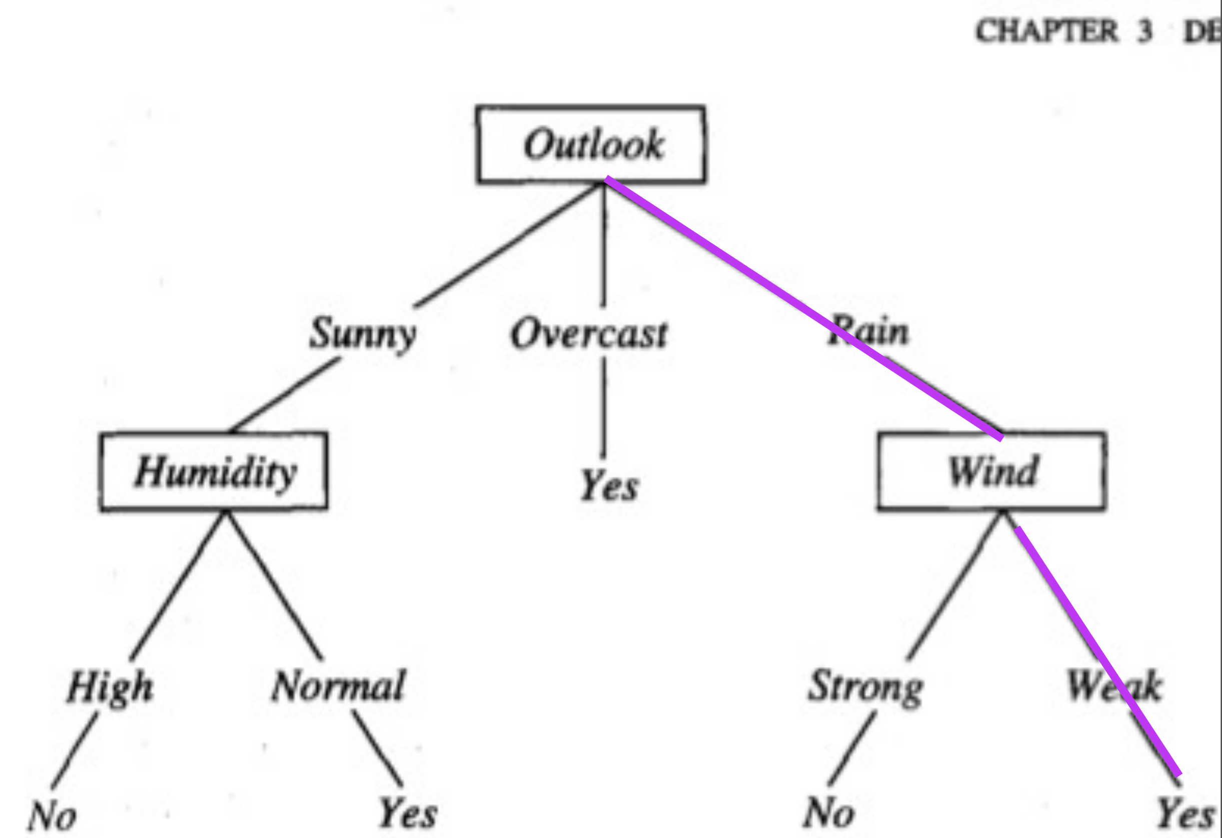
Decision Tree

H	W	O	P
H	S	S	N
N	S	O	Y
H	W	R	Y/N?

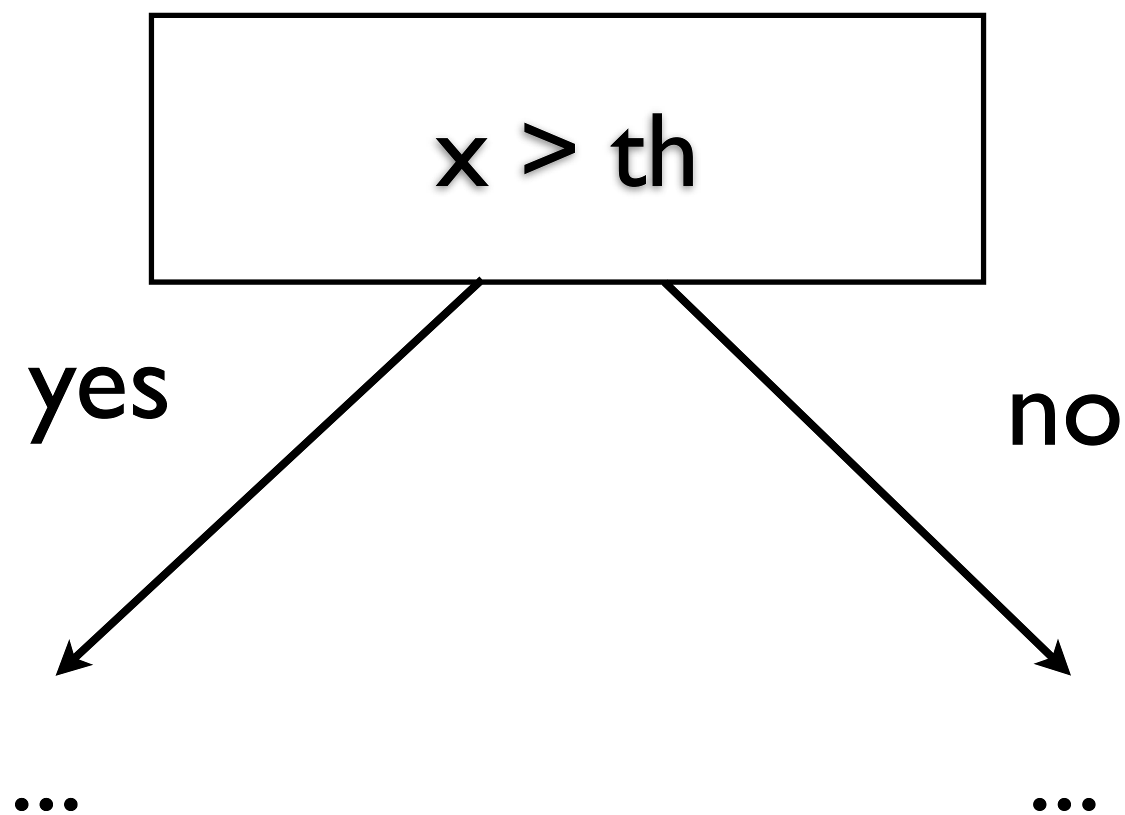


Decision Tree

H	W	O	P
H	S	S	N
N	S	O	Y
H	W	R	Y



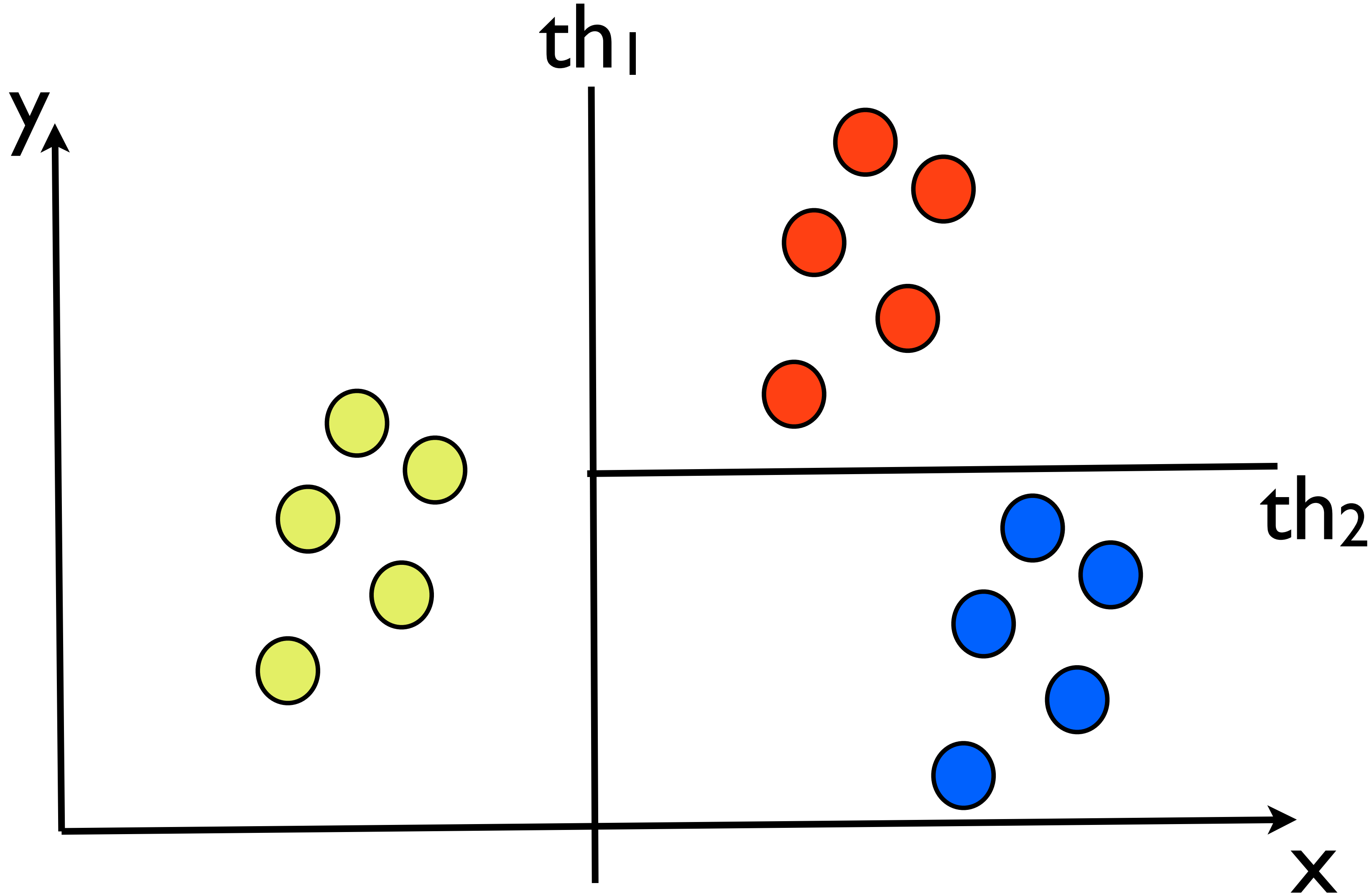
Continuos Attributes



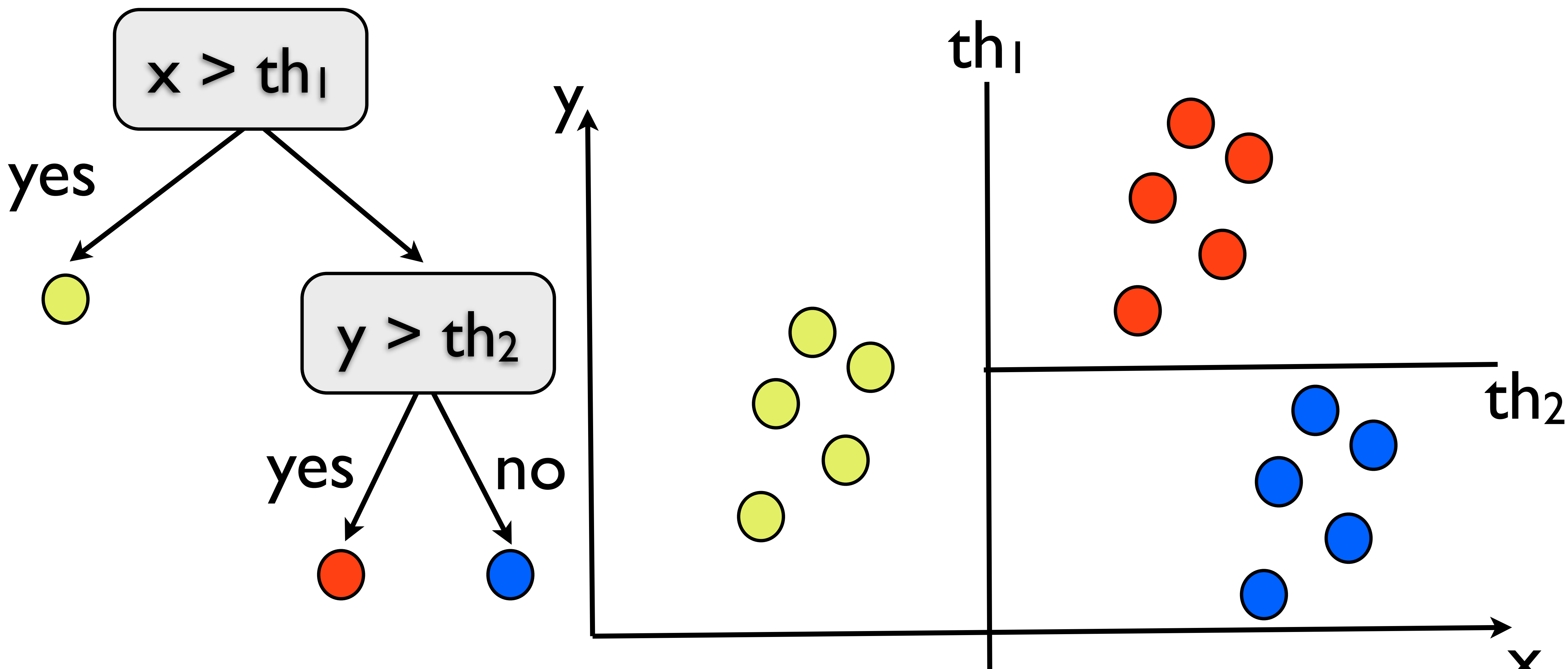
Continues Attributes

Guillotine Cut

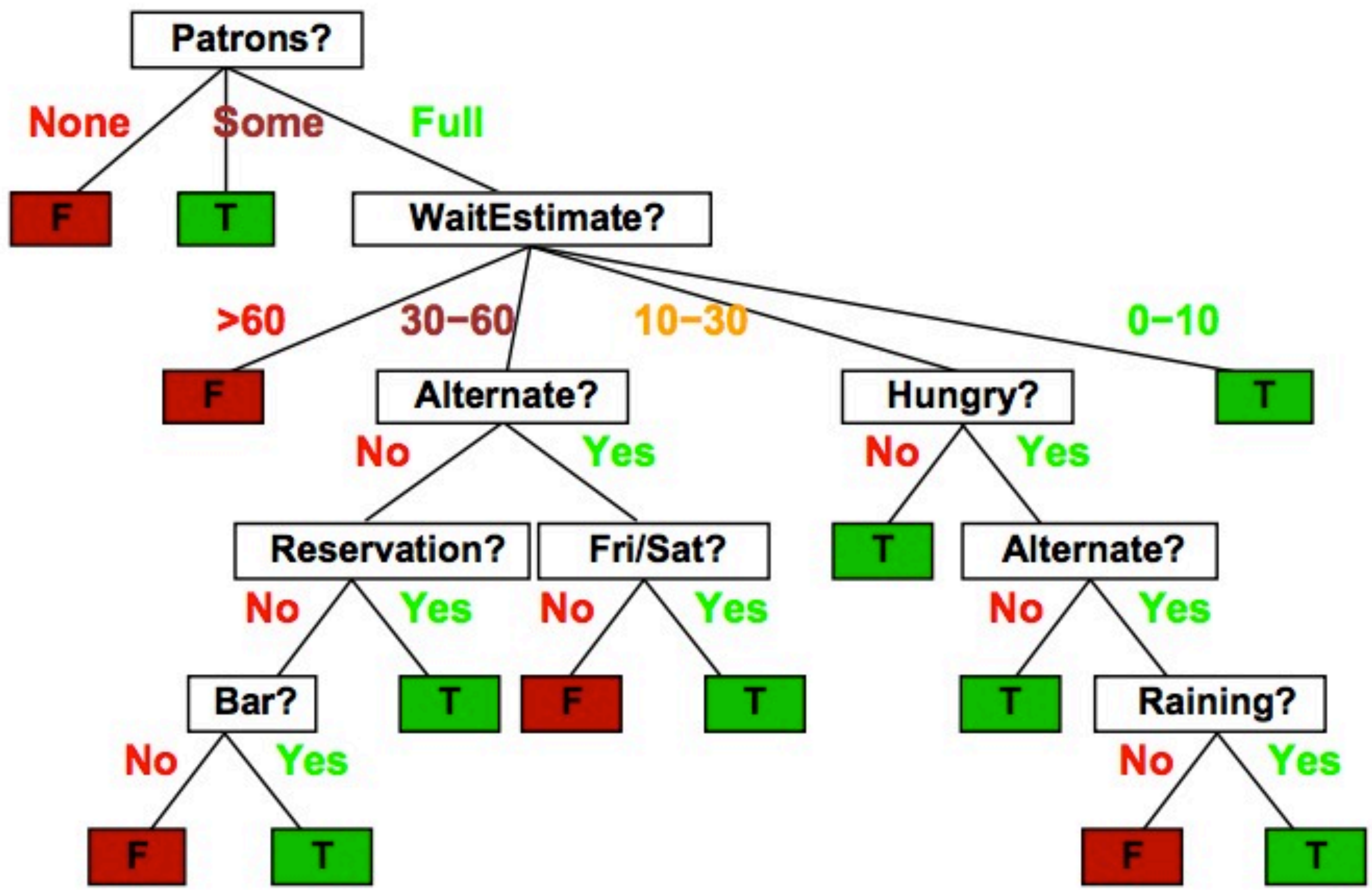
What is the tree?



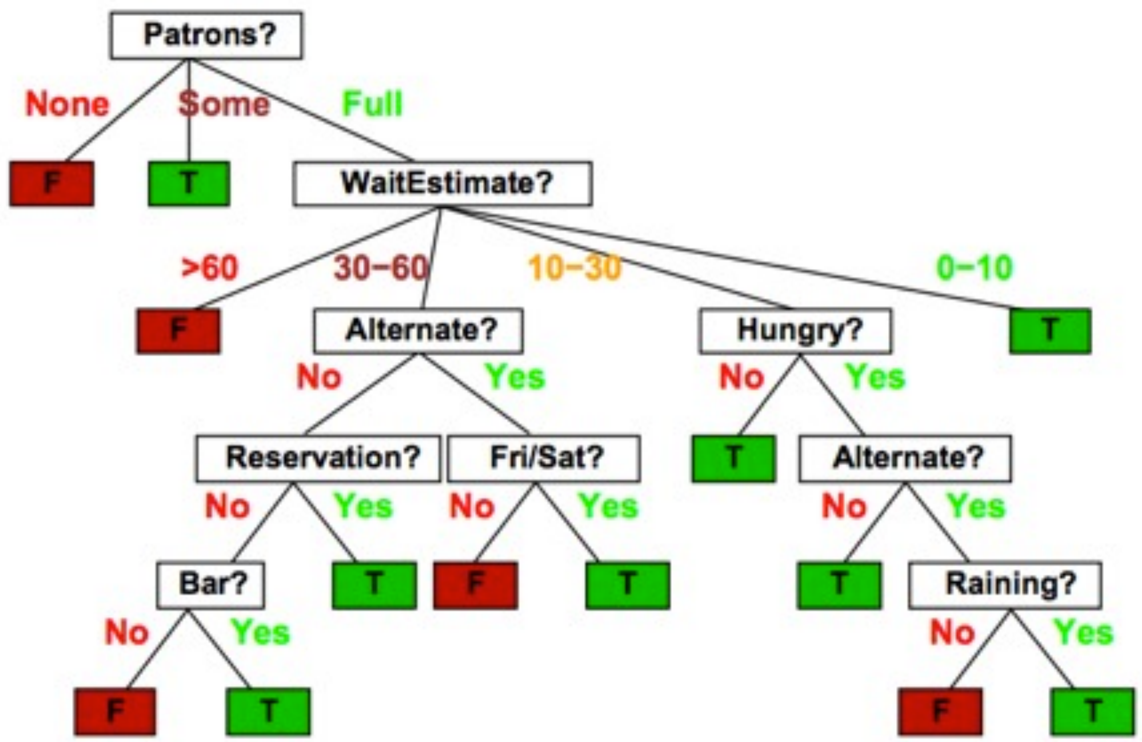
Continues Attributes

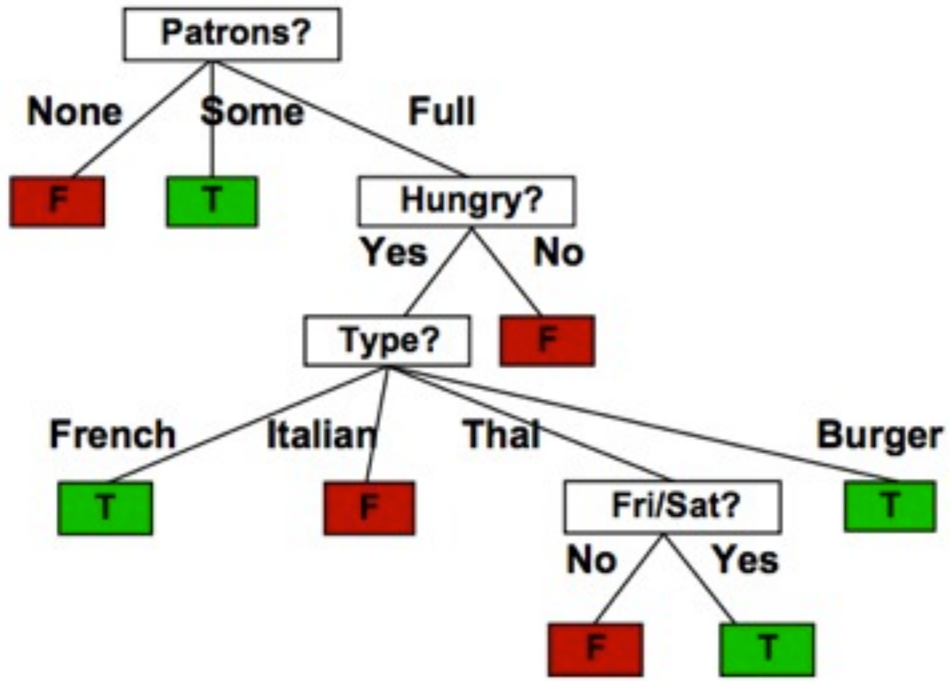


Decision Tree



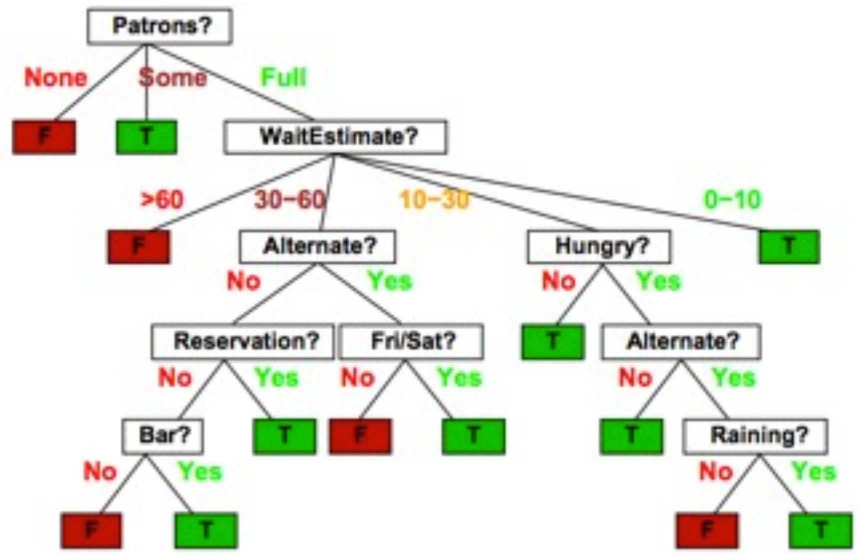
Decision Tree



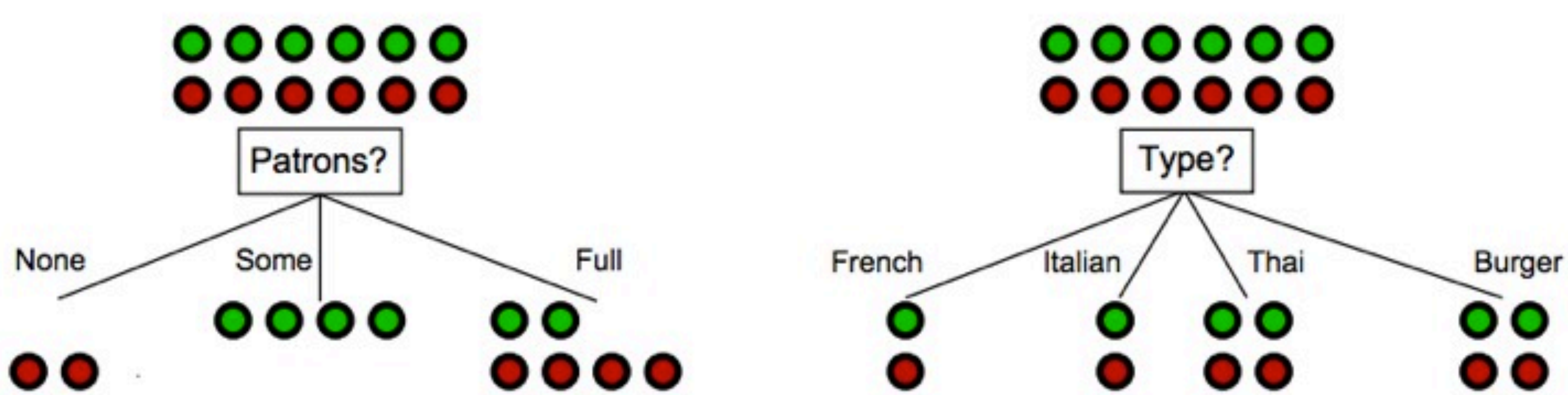


Which Tree is better if the classification accuracy is the same ?

Decision Tree Learning

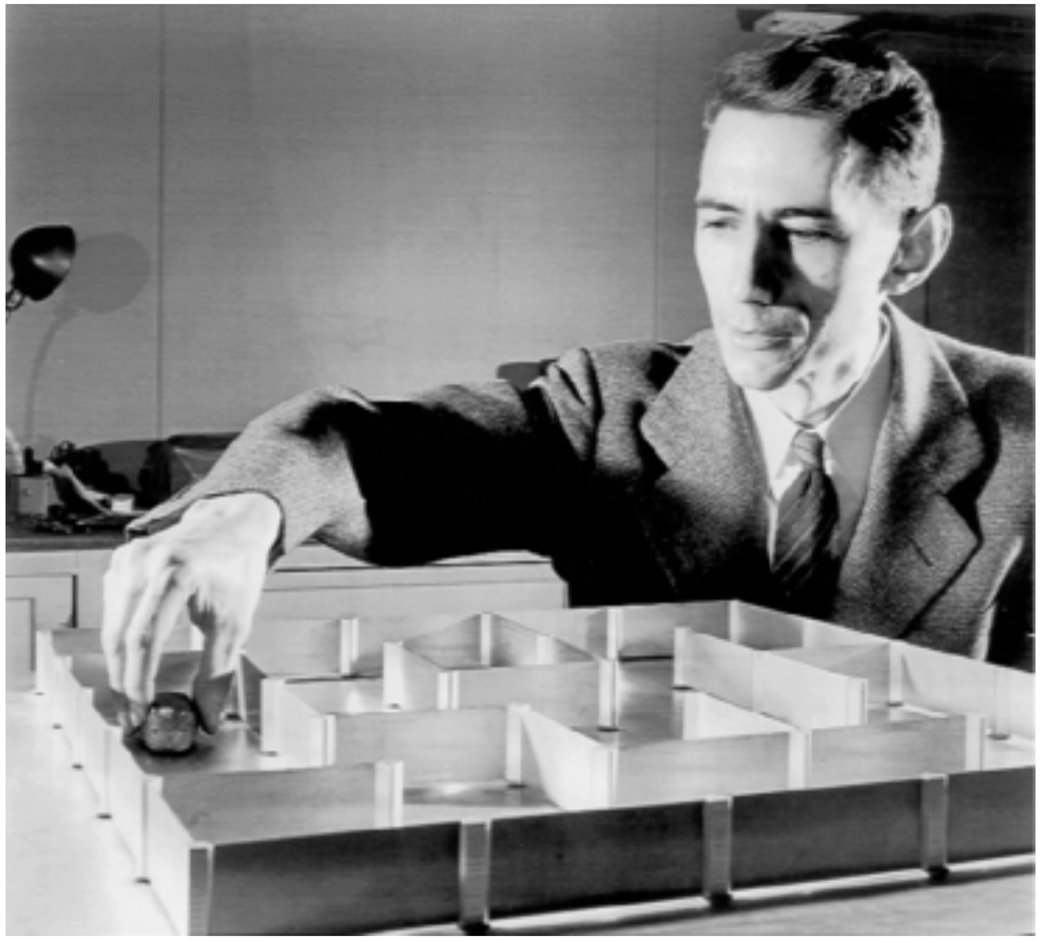


Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

Entropy



Claude Shannon

Measure Of
Uncertainty / Unpredictability
in a Random Variable

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

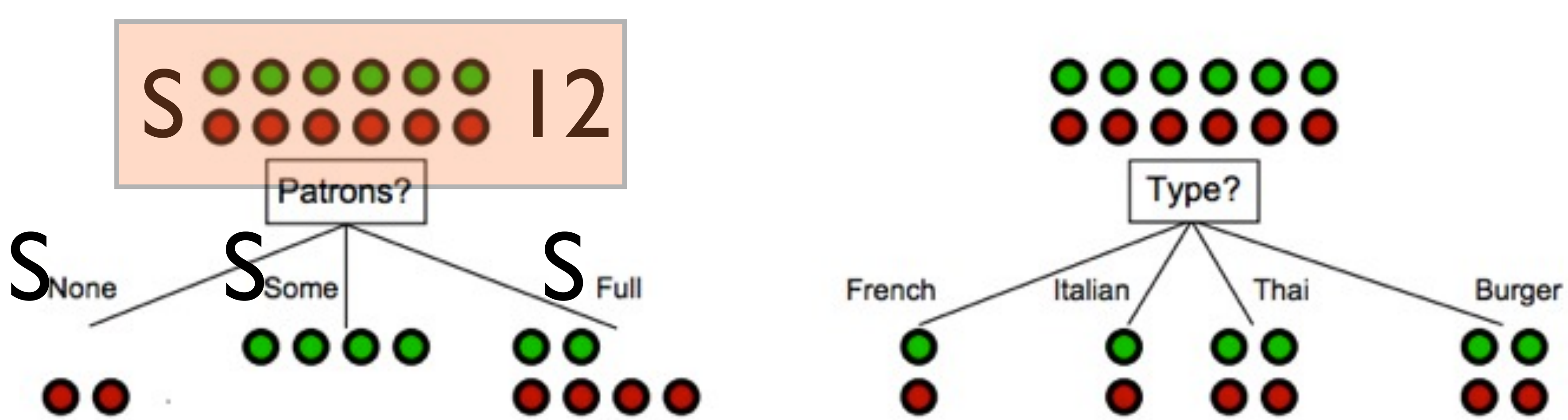
Quantifies Information in a Message

Entropy

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

$$(-\frac{6}{12} * log_2 \frac{6}{12}) - (-\frac{6}{12} * log_2 \frac{6}{12}) = 1$$

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

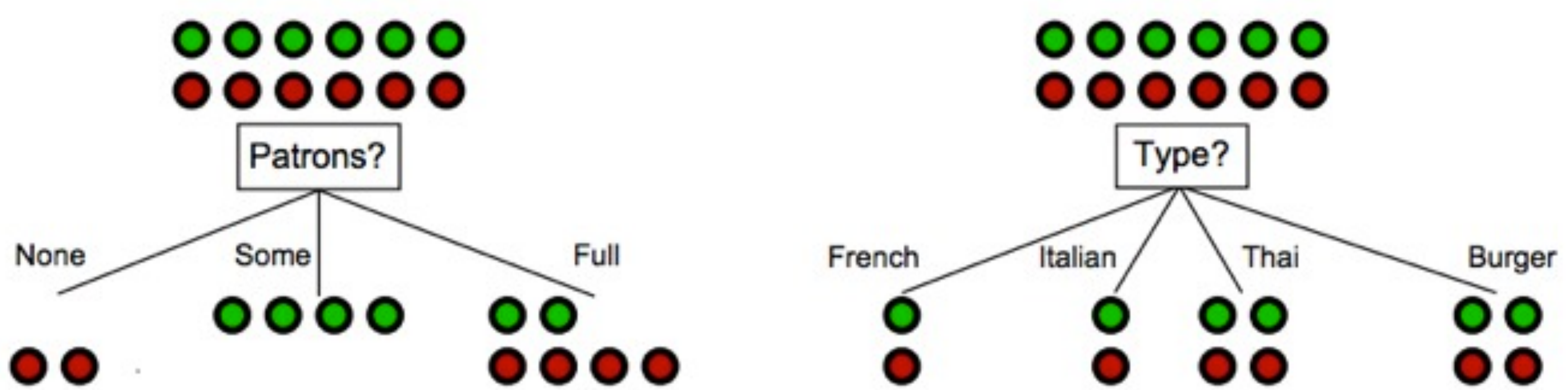


Patrons? is a better choice—gives **information** about the classification

Information Gain

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

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



Patrons? is a better choice—gives **information** about the classification

Calculate

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

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

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Result

$$Gain(S, Outlook) = 0.246$$

$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$

Decision Tree Learning

function DTL(*examples, attributes, default*) **returns** a decision tree

  if *examples* is empty **then return** *default*
  else if all *examples* have the same classification **then return** the classification
  else if *attributes* is empty **then return** MODE(*examples*)
  else
    best ← CHOOSE-ATTRIBUTE(*attributes, examples*)
    tree ← a new decision tree with root test *best*
    for each value v_i of *best* **do**
      $examples_i$ ← {elements of *examples* with $best = v_i$}
      subtree ← DTL($examples_i$, *attributes* − *best*, MODE(*examples*))
      add a branch to *tree* with label v_i and subtree *subtree*
    return *tree*

Greedy Algorithm

- In search terms: A greedy algorithm with the Information Gain as a Heuristic
- Could we do better?

Example

What is the best?

```
@relation 'gatech_admission'
@attribute 'recommendation' {'strong', 'weak'}
@attribute 'gpa' real
@attribute 'gre_math' real
@attribute 'gre_verbal' real
@attribute 'admitted' {'yes','no'}

@data
'strong', 4,   800, 800, 'yes'
'weak', 3.4, 600, 500, 'yes'
'strong', 3.6, 800, 550, 'yes'
'strong', 3,   700, 650, 'yes'
'weak',   3.2, 800, 800, 'yes'
'strong', 4,   550, 500, 'yes'
'strong', 3.7, 700, 750, 'yes'
'weak',   4,   800, 200, 'yes'
'strong', 4,   200, 800, 'yes'
'strong', 3.4, 600, 500, 'yes'
'strong', 3.6, 800, 550, 'yes'
'weak', 3,   700, 650, 'yes'
'strong', 4,   550, 500, 'yes'
'strong', 3.7, 700, 750, 'yes'
'weak',   2.8, 800, 800, 'no'
'weak',   4,   200, 200, 'no'
```

```
'strong', 2,   500, 200, 'no'
'strong', 3.5, 200, 800, 'no'
'weak',   2,   800, 800, 'no'
'weak',   1.7, 100, 100, 'no'
'weak',   3.7, 50,  0, 'no'
'weak',   2.8, 100, 100, 'no'
'weak',   4,   200, 200, 'no'
'strong',   2,   100, 100, 'no'
'weak',   1.7, 100, 100, 'no'
'weak',   3.7, 50,  0, 'no'
'weak',   2.8, 100, 800, 'no'
'weak',   4,   200, 200, 'no'
'strong', 2,   500, 200, 'no'
'strong', 3.5, 200, 800, 'no'
```

```
'weak',   2,   800, 800, 'no'
'weak',   1.7, 100, 100, 'no'
'strong',   3.7, 50,  0, 'no'
'weak',   2.8, 100, 800, 'no'
'weak',   4,   200, 200, 'no'
'strong', 2,   500, 200, 'no'
'strong', 3.5, 200, 800, 'no'
'weak',   2,   800, 800, 'no'
'weak',   1.7, 100, 100, 'no'
'weak',   3.7, 50,  0, 'no'
```

Weka Demo

Use Case: Mobile Text Entry

Rollon (E ,W)

Rolloff (E ,W)

Q 1 W 2 E 3 R (T) Y _ U – I + O @ P
* A 4 S 5 D 6 F / G : H ; J ' K " L del
alt 7 Z 8 X 9 C ? V ! B , N . M $
aA 0 space sym aA

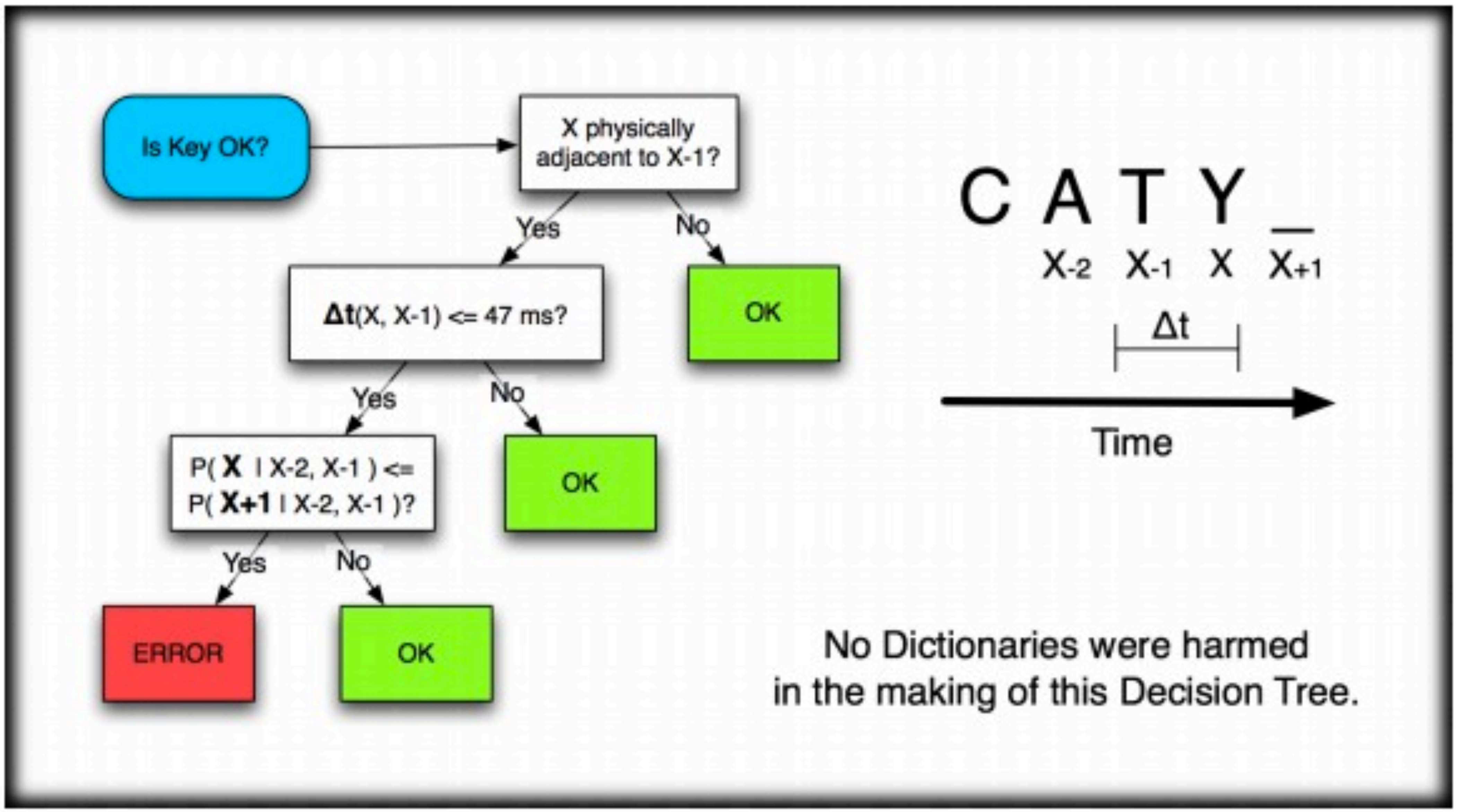
Use Case: Fat Thumbs

```
prob ≤ 0
| prevcuradjacent_nom = False
| | curfutadjacent_nom = False
| | | dropprobdiff1abs ≤ 0.000934:  repeat (104.0/45.0)
| | | dropprobdiff1abs > 0.000934
| | | | futneighborprob ≤ 0:  nonobo (552.0/11.0)
| | | | futneighborprob > 0
| | | | | neighborprob ≤ 0.011765:  nonobo (143.0/21.0)
| | | | | neighborprob > 0.011765
| | | | | | ud_sub1 ≤ -140:  nonobo (172.0/80.0)
| | | | | | ud_sub1 > -140:  obosubstitute (527.0/109.0)
| | curfutadjacent_nom = True:  rollon (323.0/73.0)
| prevcuradjacent_nom = True:  rolloff (555.0/46.0)
prob > 0
| dt_ud_0_p1 ≤ 124
| | dropprobdiffsign ≤ -1:  rolloff (155.0/36.0)
| | dropprobdiffsign > -1:  nonerror (111.0/49.0)
```

```
| | | | | | neighborprob ≤ 0.003185:  nonerror (142.0/35.0)
| | | | | | neighborprob > 0.003185
| | | | | | | average_dd_2 ≤ -88
| | | | | | | | letterfreq ≤ 0.04853:  obosubstitute (106.0/35.0)
| | | | | | | | letterfreq > 0.04853:  nonobo (142.0/69.0)
| | | | | | | average_dd_2 > -88
| | | | | | | | neighborprobdiff ≤ 0.009091:  nonerror (181.0/65.0)
| | | | | | | | neighborprobdiff > 0.009091:  obosubstitute (229.0/82.0)
| | | | curfutadjacent_nom = True:  nonerror (129.0/62.0)
| | | futprob > 0:  nonerror (94376.0/1715.0)
```

```
| dt_ud_0_p1 > 124
| | dt_ud_1_0 ≤ 144
| | | dropprobgain ≤ 0.006861:  nonerror (522.0/62.0)
| | | dropprobgain > 0.006861
| | | | neighborprob1diff ≤ -0.016393:  nonerror (185.0/92.0)
| | | | neighborprob1diff > -0.016393:  rollon (381.0/126.0)
| | dt_ud_1_0 > 144
| | | futprob ≤ 0
| | | | curfutadjacent_nom = False
| | | | | futneighborprob ≤ 0
| | | | | | futnowdiff ≤ -176:  nonobo (141.0/54.0)
| | | | | | futnowdiff > -176:  nonerror (894.0/201.0)
| | | | | futneighborprob > 0
| | | | | | neighborprob ≤ 0.003185:  nonerror (142.0/35.0)
| | | | | | neighborprob > 0.003185
| | | | | | | average_dd_2 ≤ -88
| | | | | | | | letterfreq ≤ 0.04853:  obosubstitute (106.0/35.0)
| | | | | | | | letterfreq > 0.04853:  nonobo (142.0/69.0)
| | | | | | | average_dd_2 > -88
```

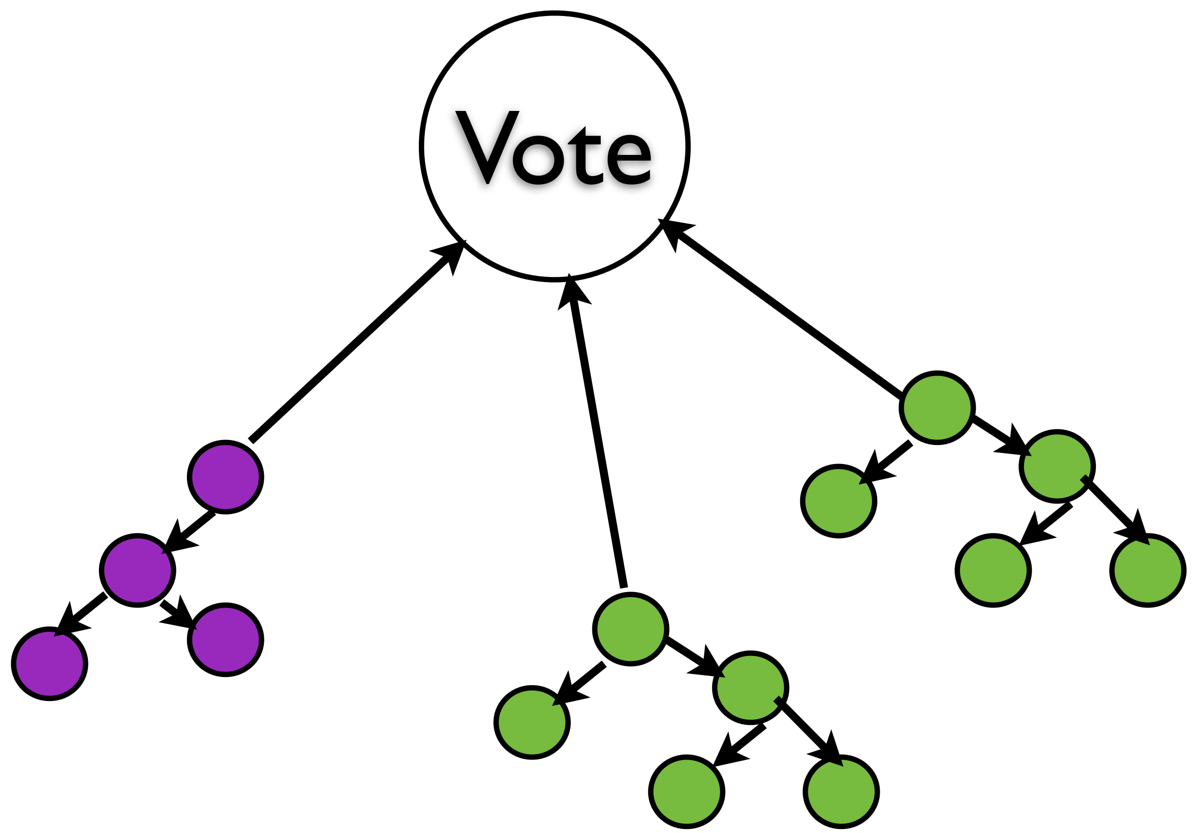
Use Case: Fat Thumbs



What are the drawbacks?

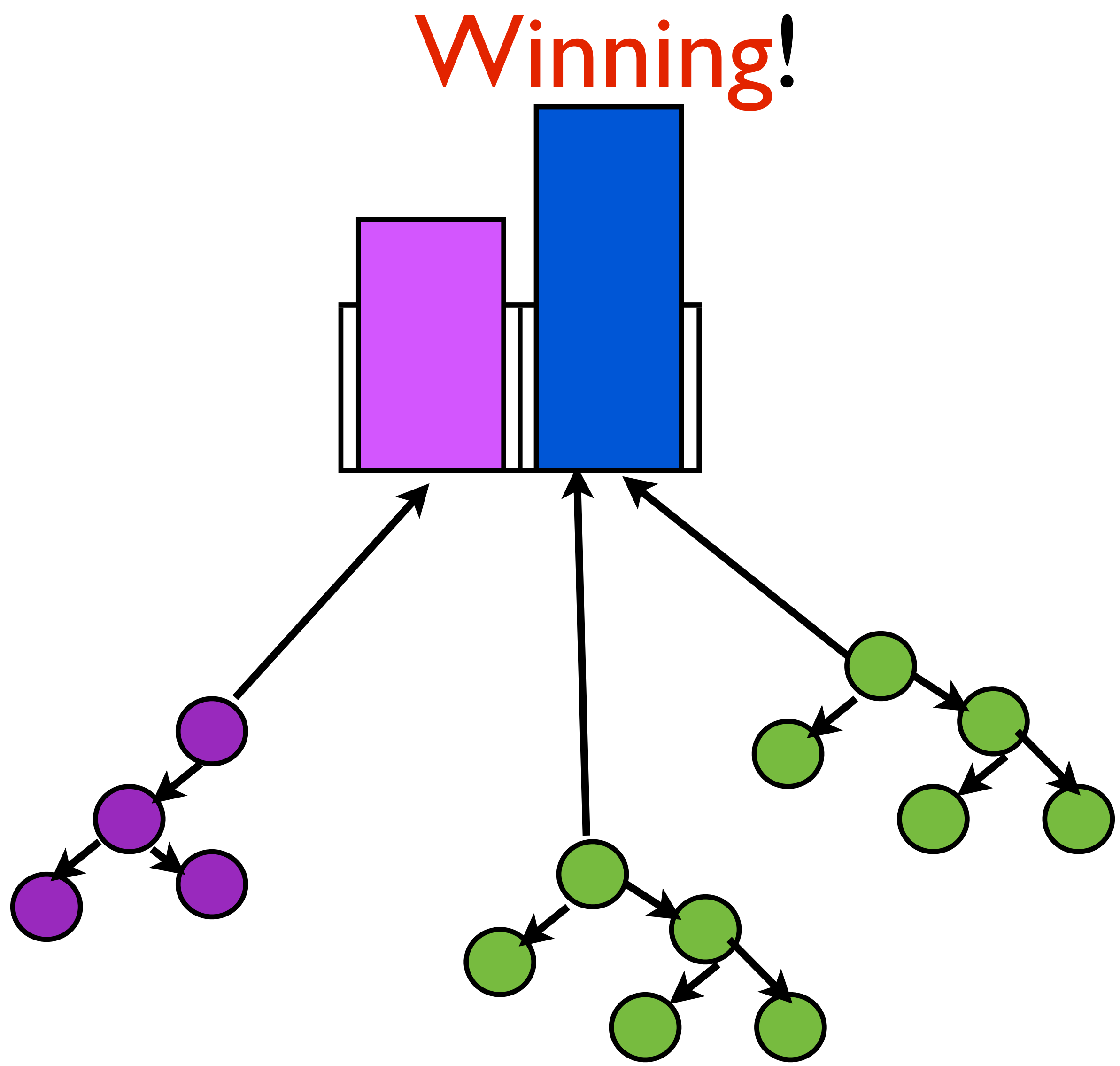
Decision Trees are known to over fit the data

Ensemble Learning



Mixture of Experts: Have we seen one before?

Random Forests

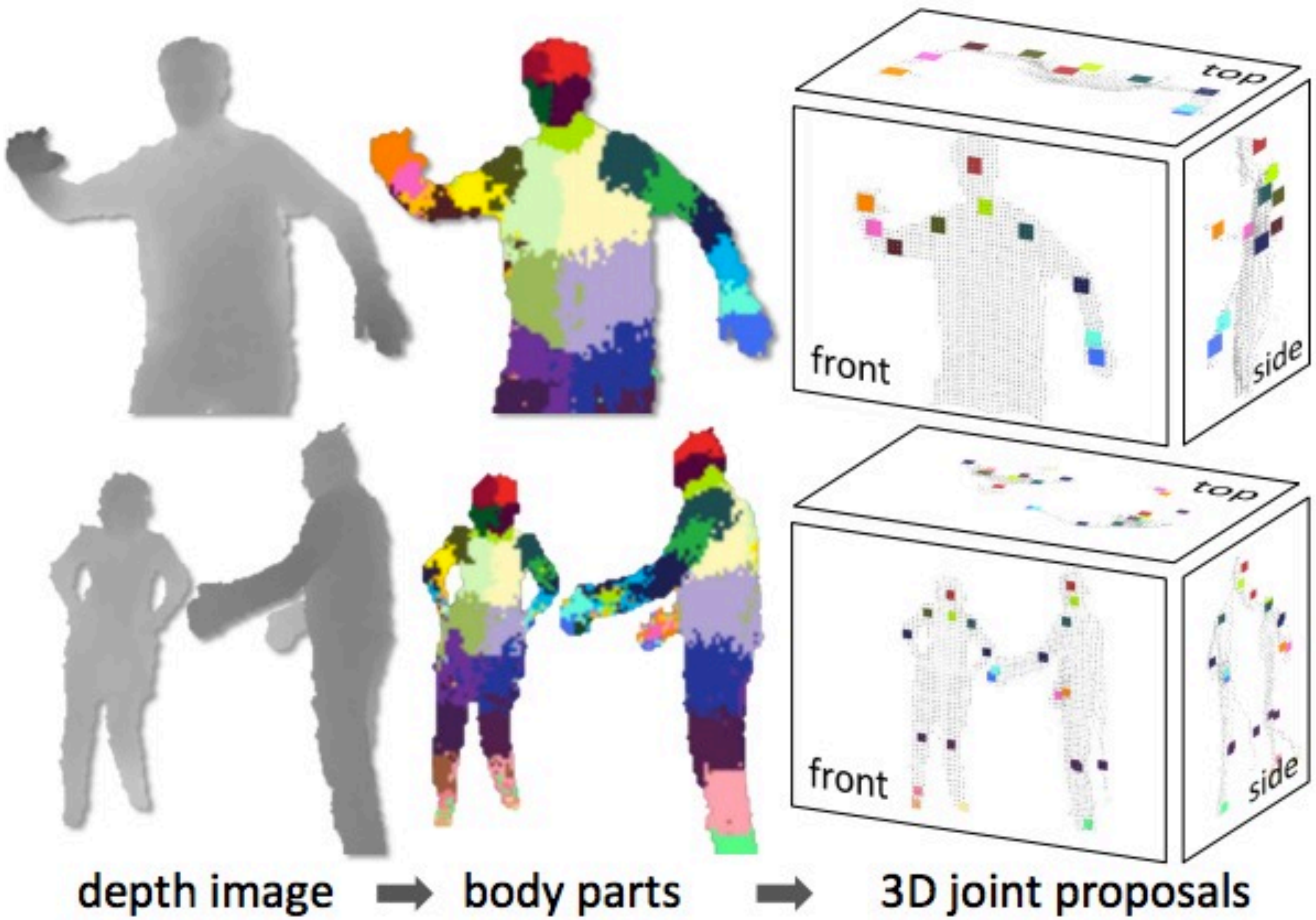


Random Forests

Bagging: Bootstrap AGGregation

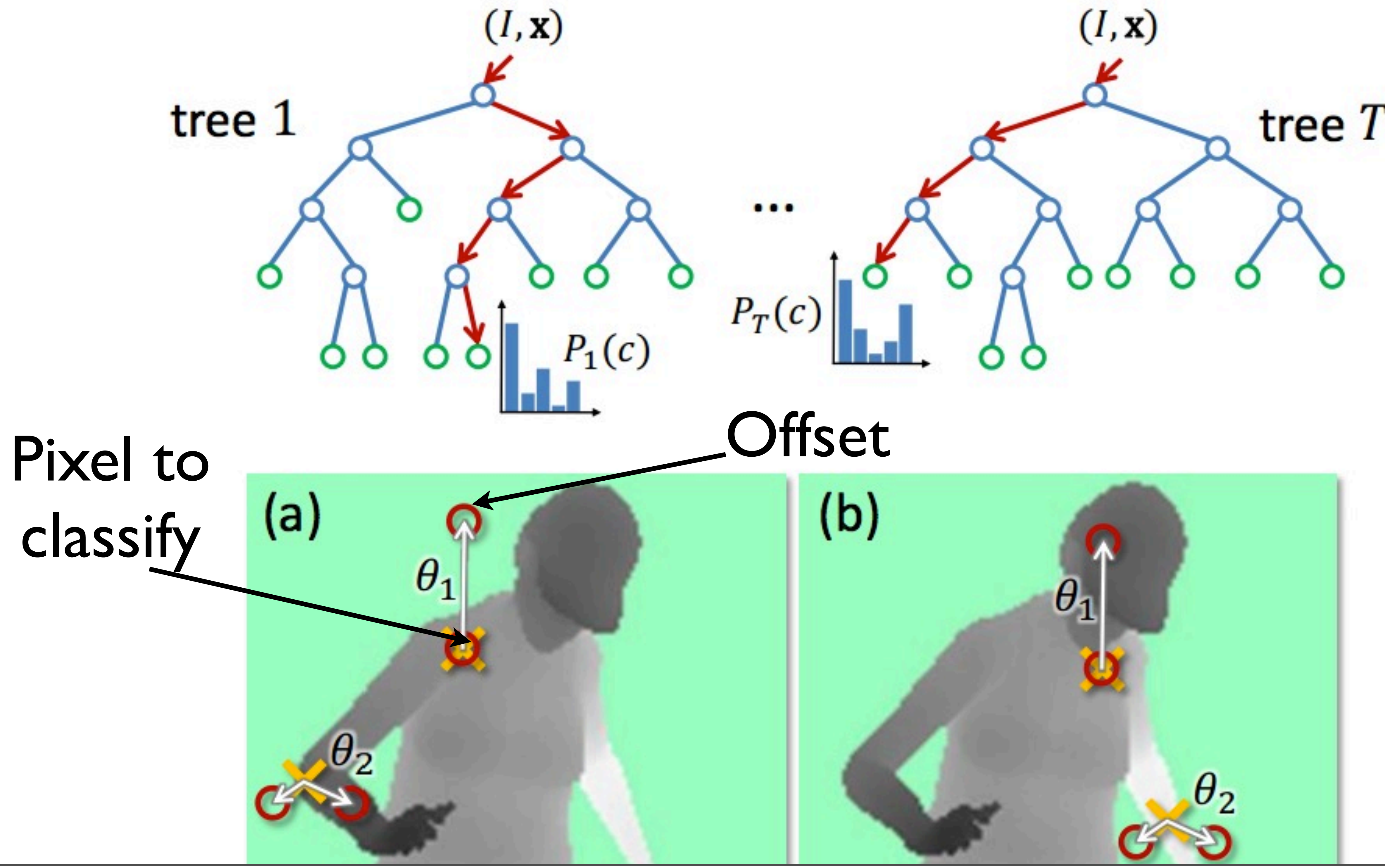
- INPUT: Data Set of size N with M dimensions
- 1) SAMPLE n times from Data
- 2) SAMPLE m times from Attributes
- 3) LEARN TREE on sampled Data and Attributes
- REPEAT UNTIL k trees

Use Case: Kinect



Use Case: Kinect

Use Case: Kinect



Use Case: Kinect

Training 3 trees to depth 20 from 1 million images takes about a day on a 1000 core cluster