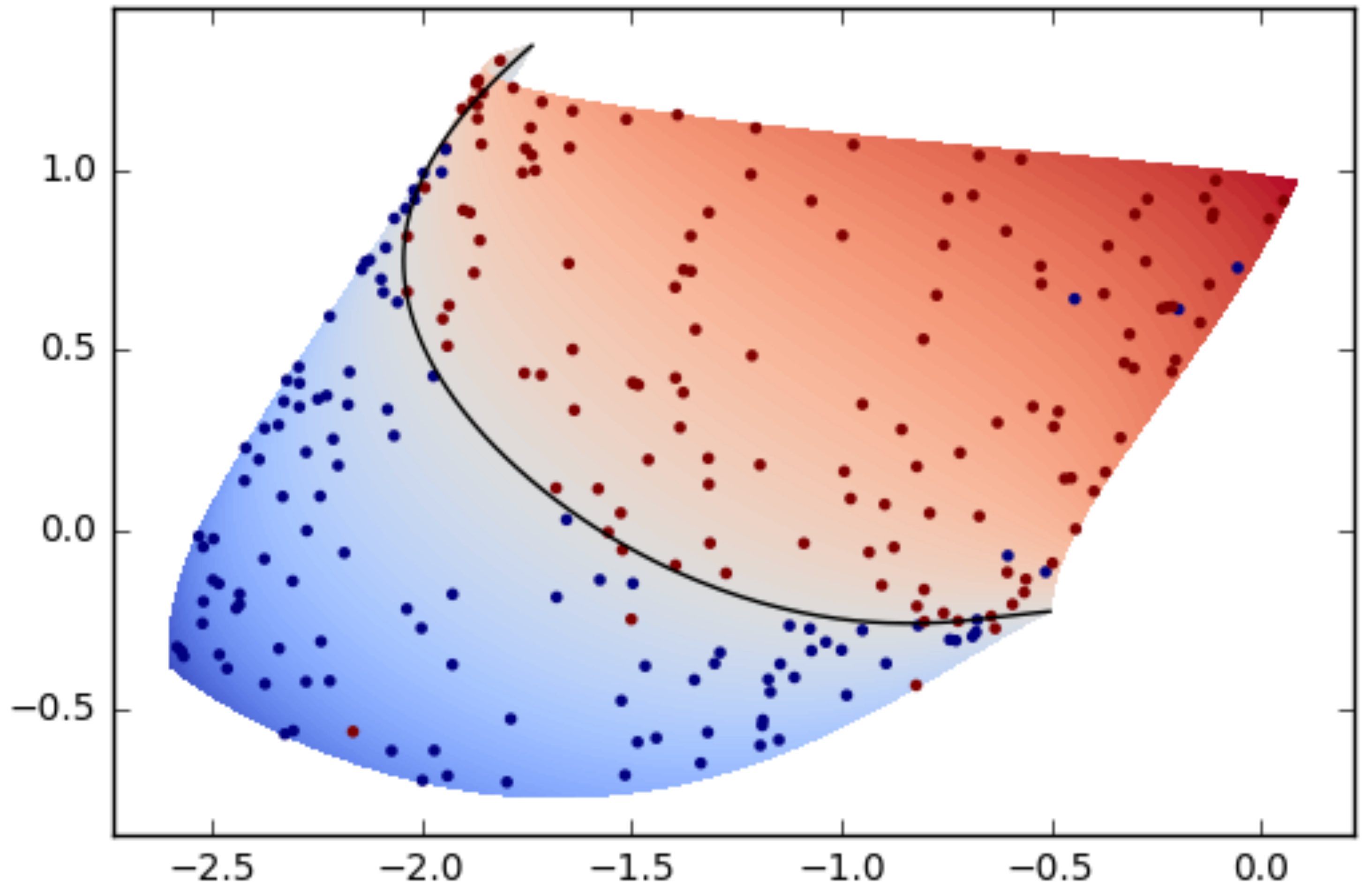
CS 273A: Machine Learning

Winter 2021

Lecture 9: Decision Trees

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine



All slides in this course adapted from Alex Ihler & Sameer Singh

Logistics

project

- Independent project this week

midterm

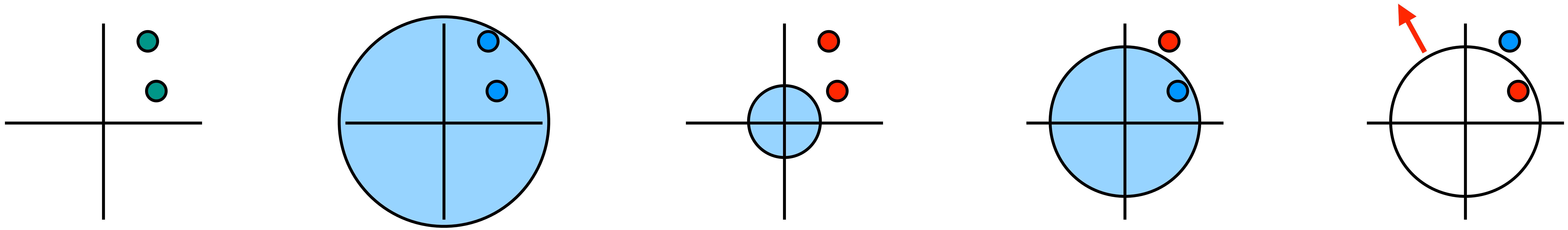
- Midterm exam next Tue, Feb 9, 2–4pm **on Canvas**
- We'll accommodate other timezones — let us know
- Review during lecture this Thu

Shattering

- Separability / realizability: there's a model that classifies all points correctly
- Shattering: the points are separable <u>regardless of their labels</u>
  - Our model class can shatter points $x^{(1)}, \ldots, x^{(h)}$

    if for <u>any</u> labeling $y^{(1)}, \ldots, y^{(h)}$

    there <u>exists</u> a model that classifies all of them correctly
- Example: can $f_\theta(x) = \text{sign}(x_1^2 + x_2^2 - \theta)$ shatter these points?

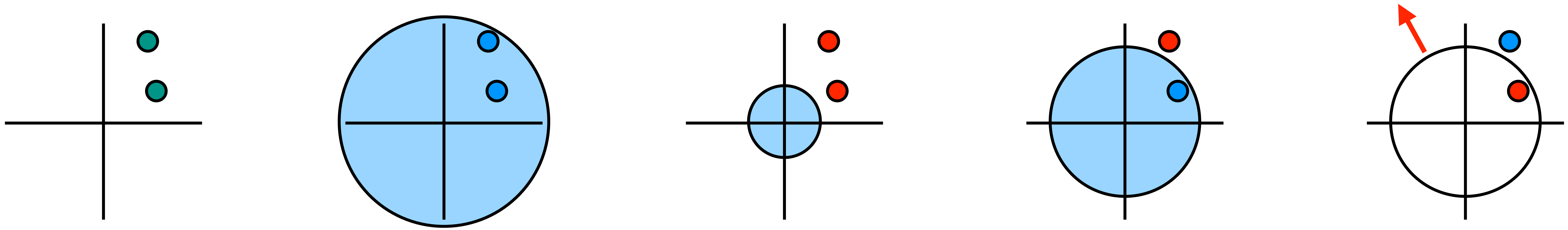


Vapnik–Chervonenkis (VC) dimension

- VC dimension: maximum number H of points that can be shattered by a class
- A game:
  - Fix a model class $f_\theta : x \to y \quad \theta \in \Theta$
  - Player 1: choose h points $x^{(1)}, \ldots, x^{(h)}$
  - Player 2: choose labels $y^{(1)}, \ldots, y^{(h)}$
  - Player 1: choose model θ
  - Are all $y^{(j)} = f_\theta(x^{(j)})$? $\implies$ Player 1 wins $\quad \exists x^{(1)}, \ldots, x^{(h)} : \ \forall y^{(1)}, \ldots, y^{(h)} : \ \exists \theta : \ \forall j : \ y^{(j)} = f_\theta(x^{(j)})$
- $h \leq H \implies$ Player 1 can win, otherwise cannot win

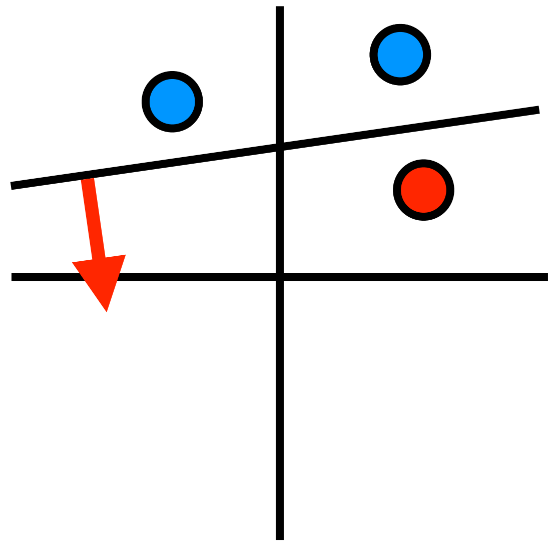
VC dimension: example (1)

- VC dimension: maximum number H of points that can be shattered by a class
- To find H, think like the winning player: 1 for $h \leq H$, 2 for $h > H$
- Example: $f_\theta(x) = \mathrm{sign}(x_1^2 + x_2^2 - \theta)$
  - We can place one point and "shatter" it
  - We can prevent shattering any two points: make the distant one blue
  - $H = 1$

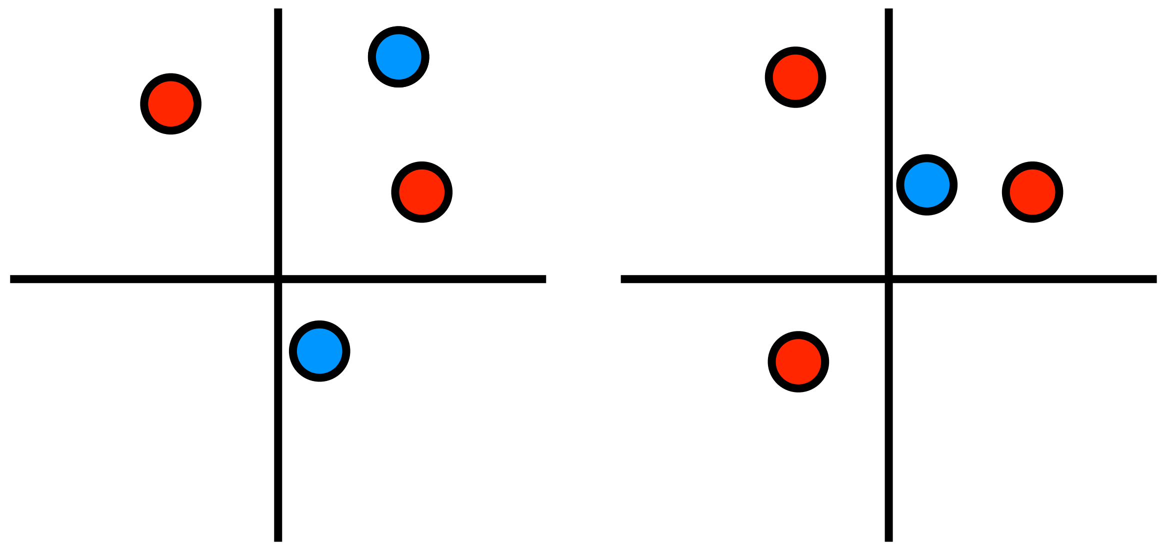


VC dimension: example (2)

- Example: $f_\theta(x) = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
  - We can place 3 points and shatter them



  - We can prevent shattering <u>any</u> 4 points:
    - If they form a convex shape, alternate labels
    - Otherwise, label differently the point in the triangle



  - $H = 3$
- Linear classifiers (perceptrons) of d features have VC-dim $d + 1$
  - But VC-dim is generally not #parameters

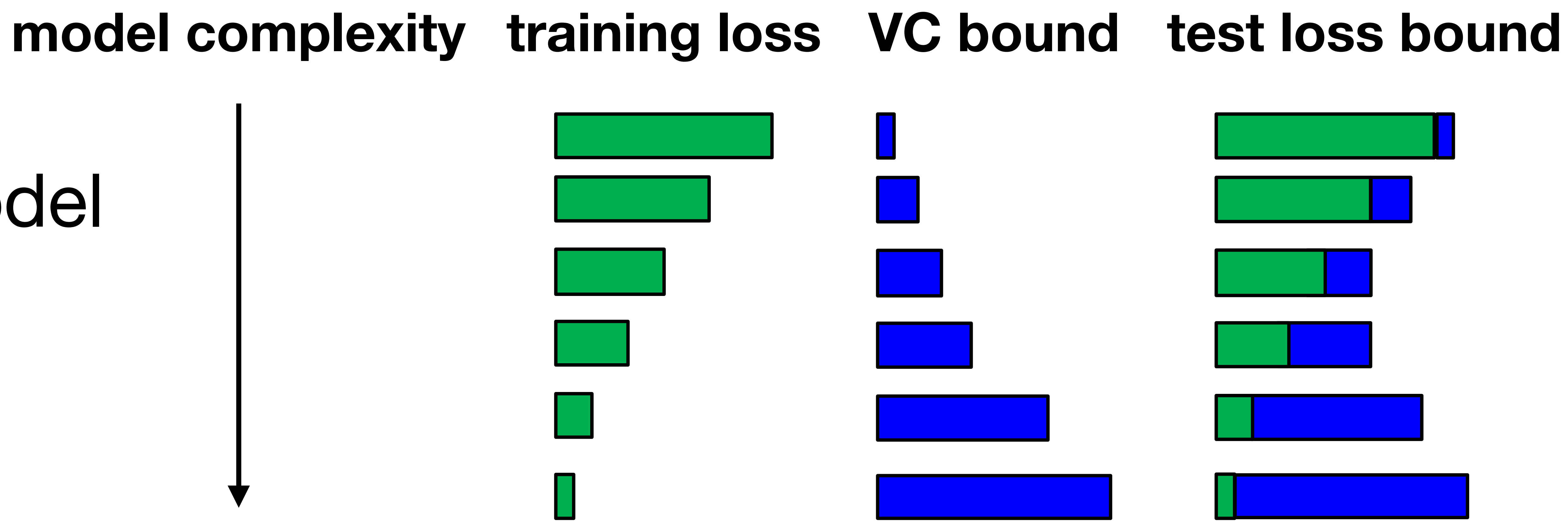
VC Generalization bound

- VC-dim of a model class can be used to bound generalization loss:
  - With probability at least $1-\eta$, we will get a "good" dataset, for which
- $$\underbrace{\text{test loss} \ - \ \text{training loss}}_{\text{generalization loss}} \leq \sqrt{\frac{H\log(2m/H)+H-\log(\eta/4)}{m}}$$
- We need larger training size m:
  - The better generalization we need
  - The more complex (higher VC-dim) our model class
  - The more likely we want to get a good training sample

Model selection with VC-dim

- Using validation / cross-validation:
  - Estimate loss on held out set
  - Use validation loss to select model

- Using VC dimension:
  - Use generalization bound to select model
  - Structural Risk Minimization (SRM)
  - Bound not tight, must too conservative



Today's lecture

Decision Trees

Learning Decision Trees

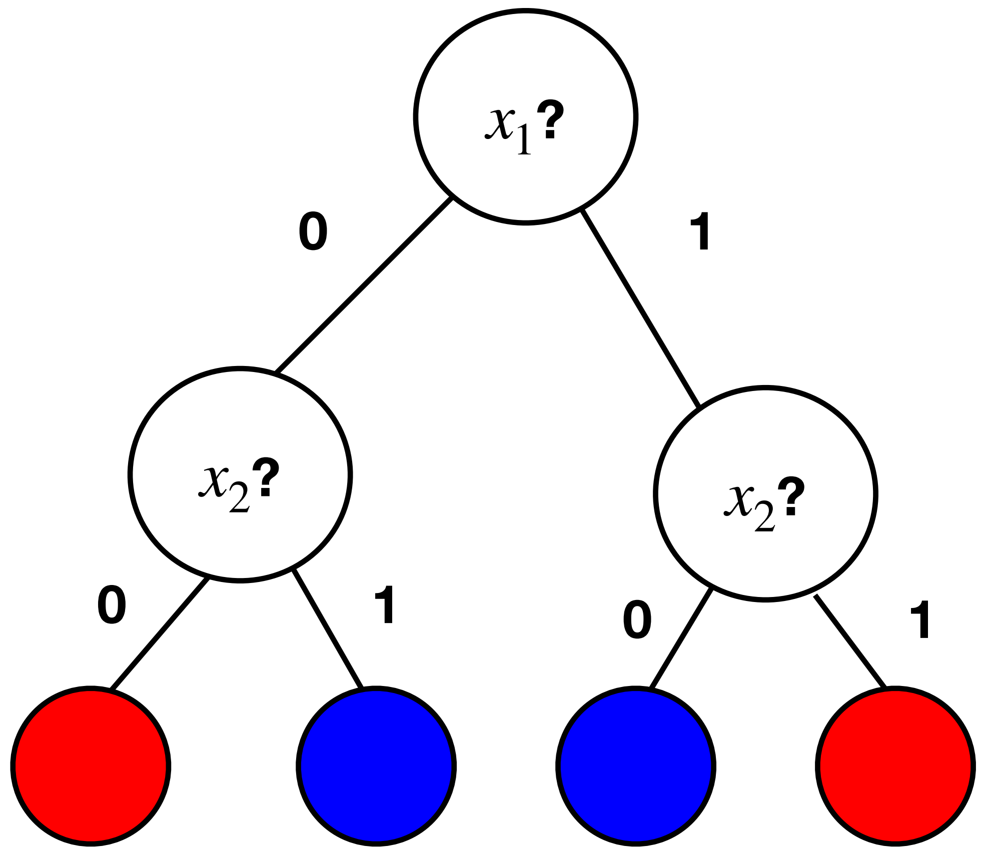
Complexity of Decision Trees

Decision Trees

- Decision Tree = nested if–then–else statements
  - Assume discrete features
- Structure:
  - Internal nodes: check feature, branch on value
  - Leaf nodes: output prediction
- Parameters:
  - Internal node features
  - Leaf outputs

XOR

x_1	x_2	y
0	0	-1
0	1	1
1	0	1
1	1	-1




```
if x1:            # branch on feature at root
  if x2:          # if true, branch or right child feature
    return -1     #   if right child true, return -1
  else:
    return +1     #   if right child false, return +1
else:
  if x2:          # if root false, branch on left child
    return +1     #   if left child true, return +1
  else:
    return -1     #   if left child false, return -1
```

Toy example: car MPG

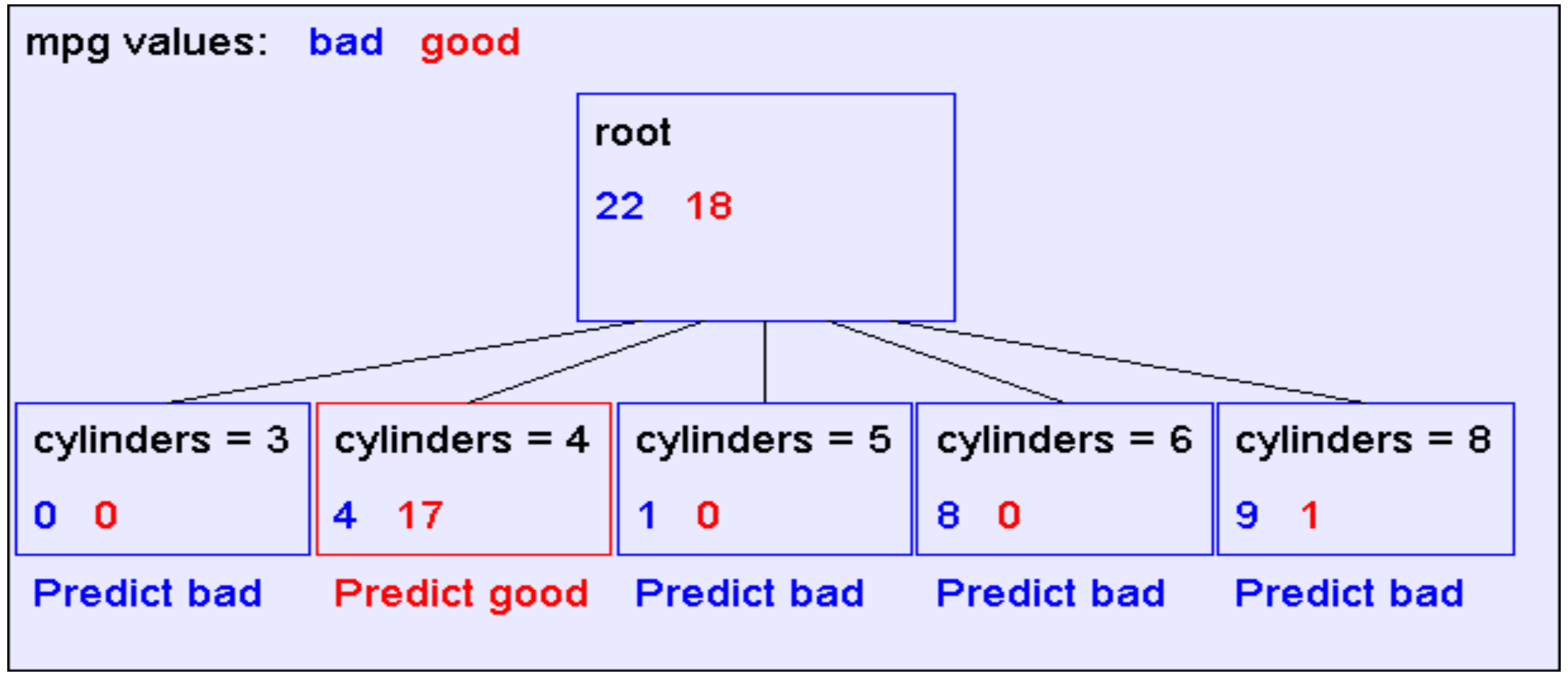
- Predict car gas usage (MPG = miles-per-gallon)
  - Discretize features and predictions (good / bad)

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

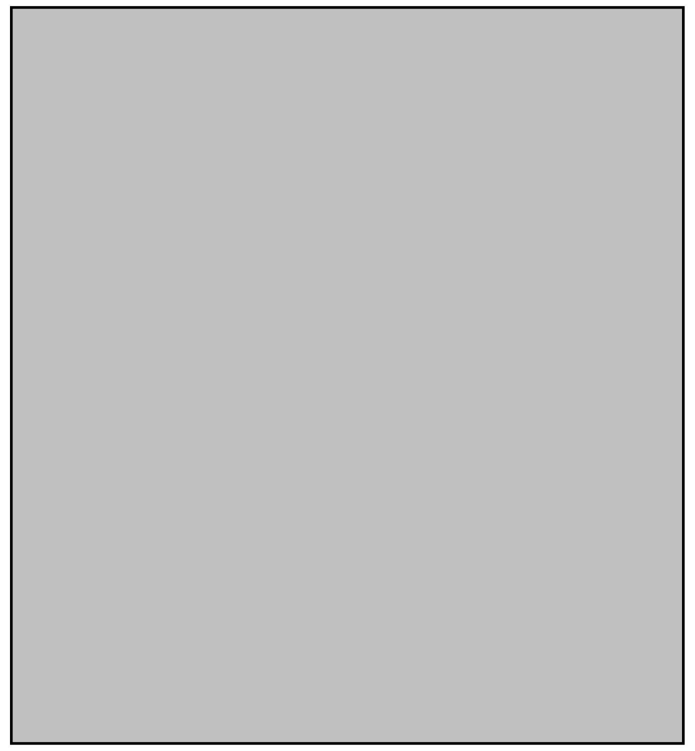
y $\underbrace{\qquad\qquad\qquad}_{x}$

40 training examples

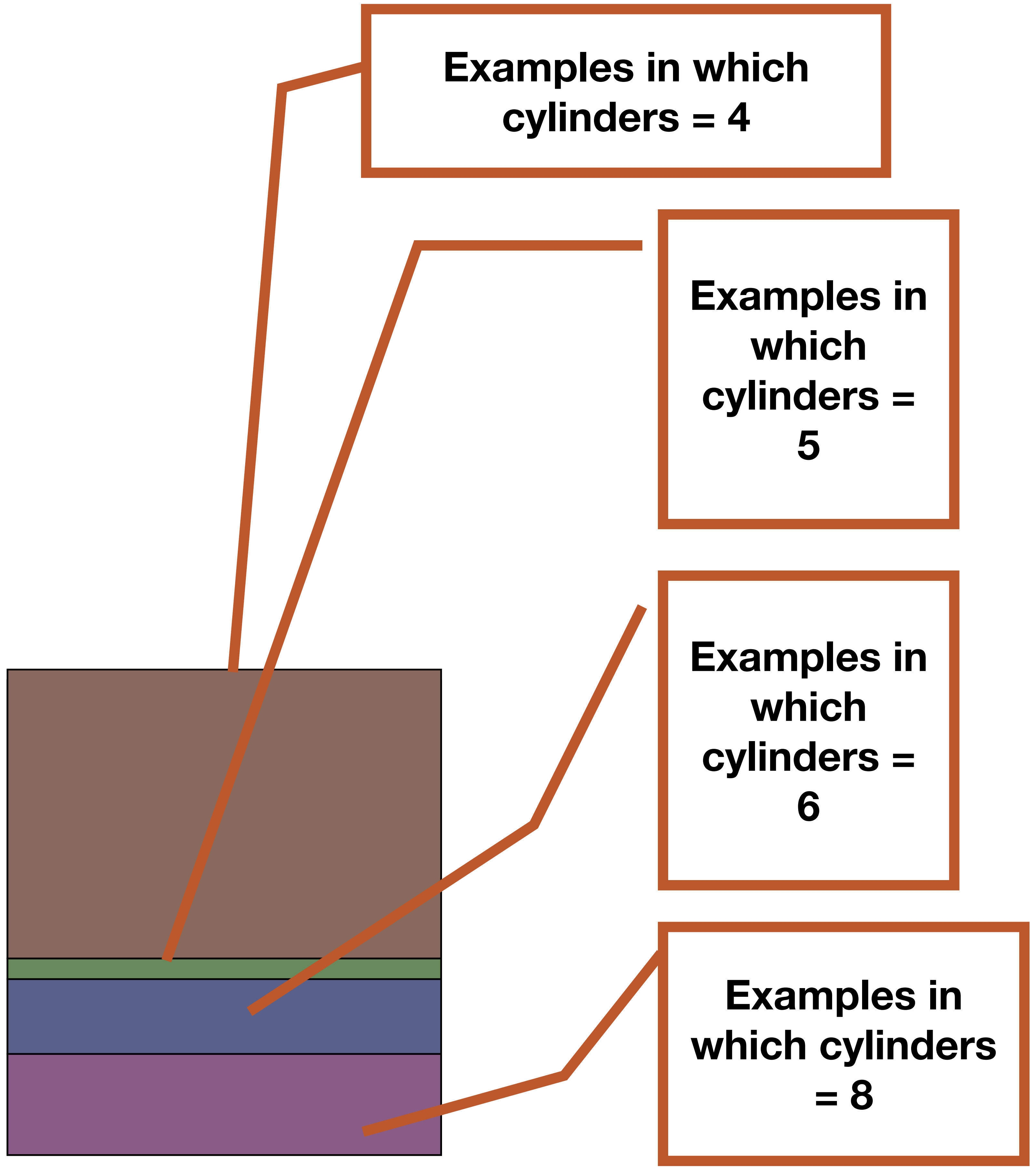
Decision Stump (1-rule)



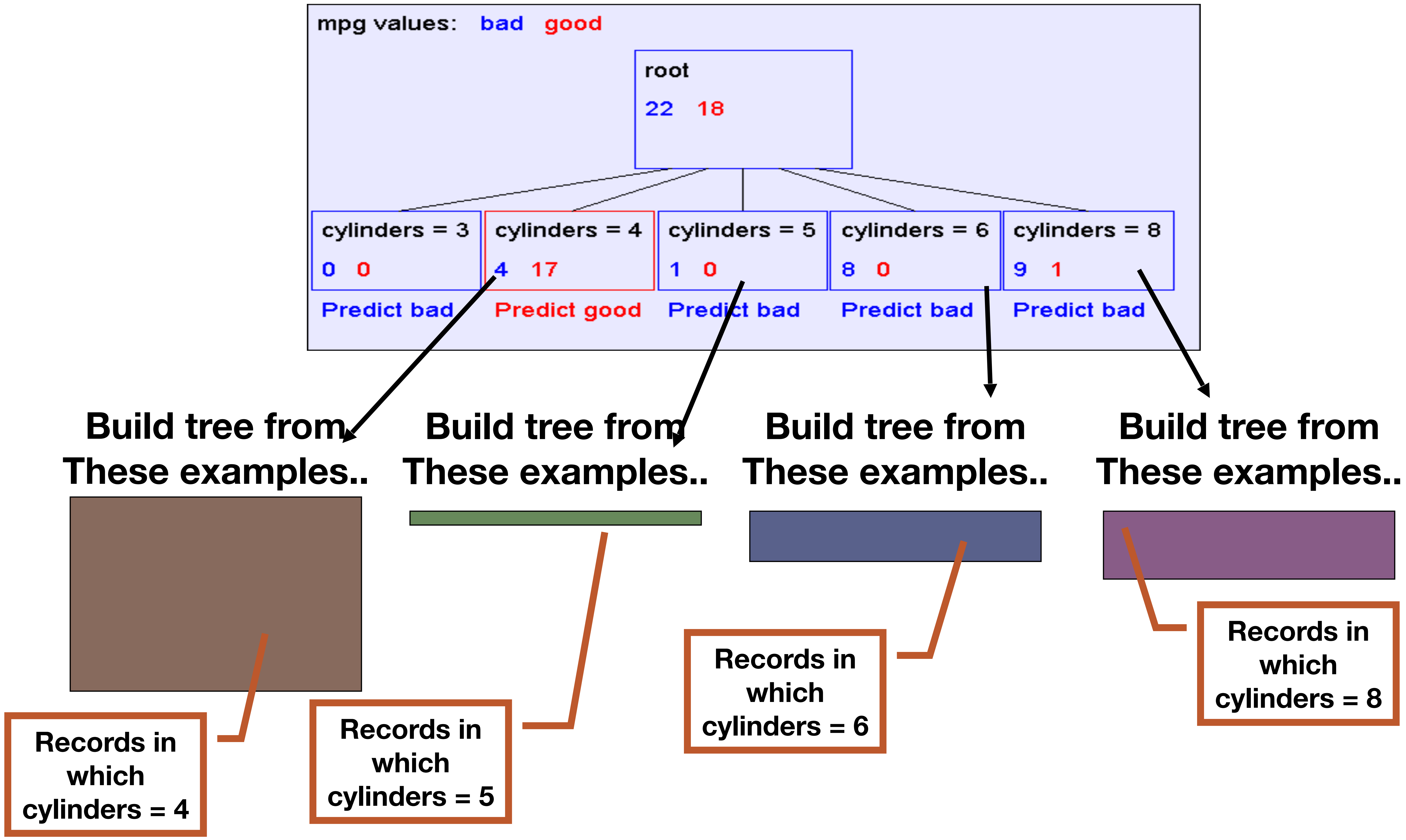
Take the Original Dataset..



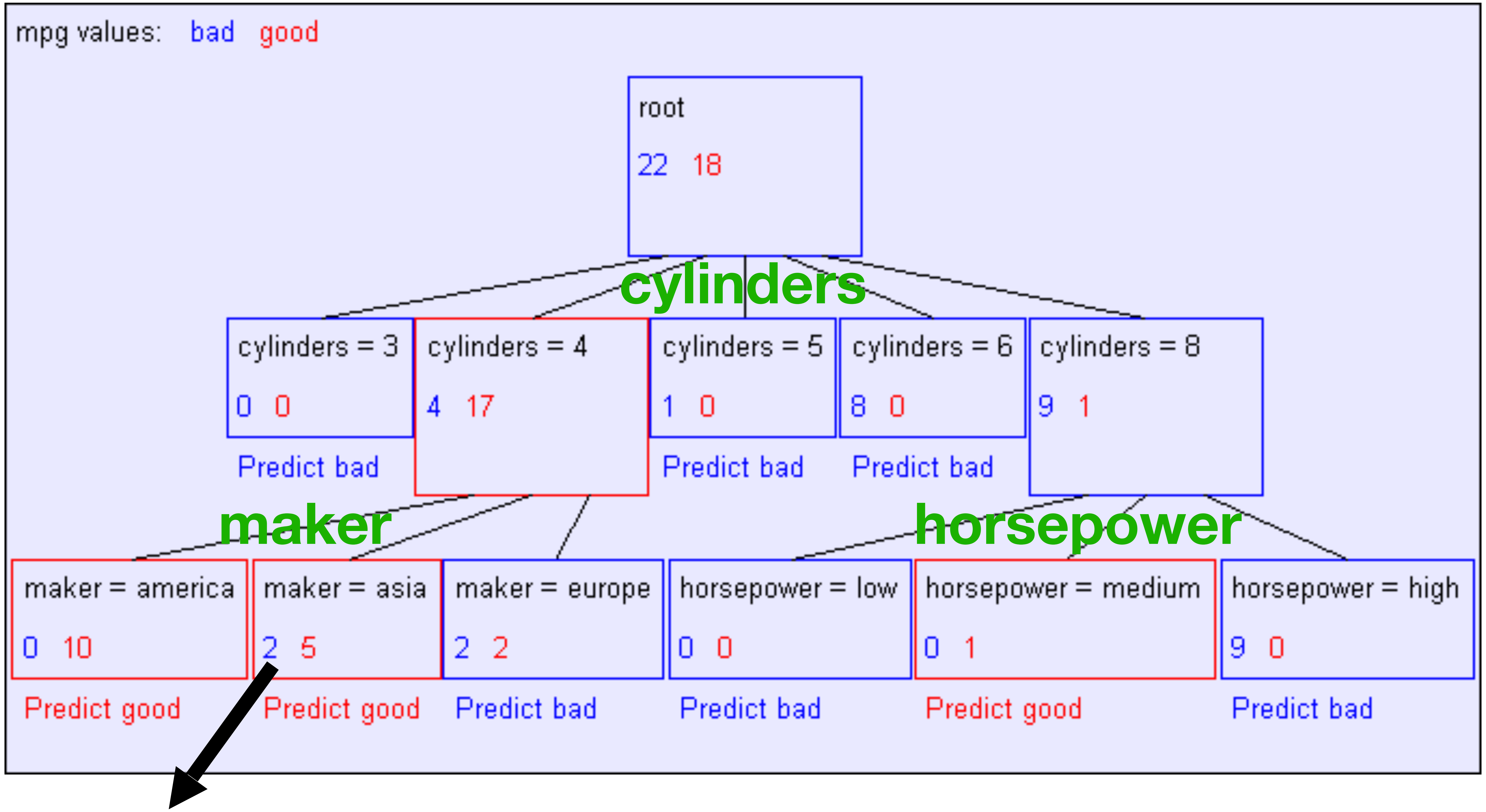
And partition it according to the value of the attribute we split on



Recursion Step



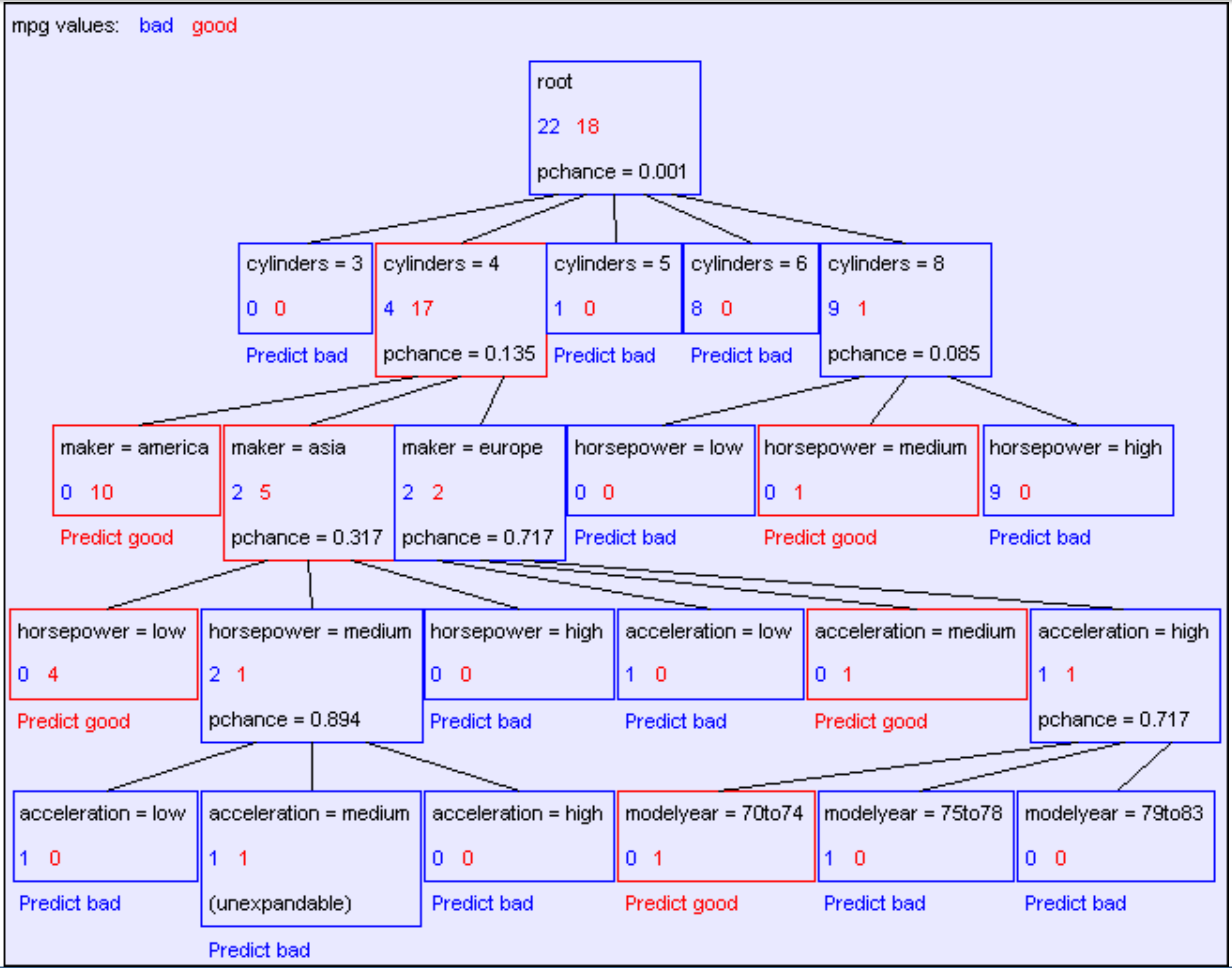
Second level of tree



Recursively build sub-tree from 7 records with cylinders = 4 and maker = Asia

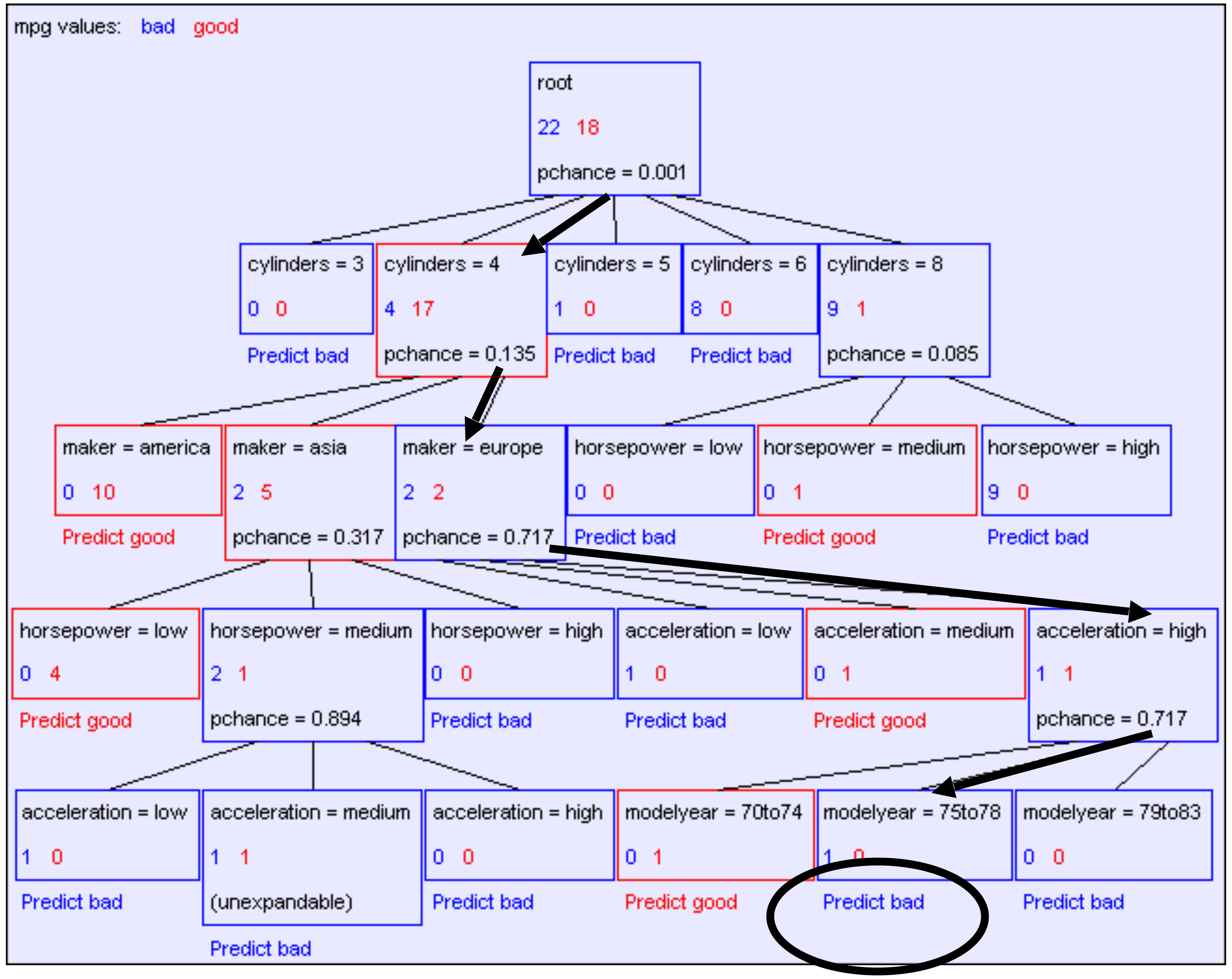
(Similar recursion in the other cases)

Final tree


mpg values: bad good
root
22 18
pchance = 0.001
cylinders = 3
0 0
Predict bad
cylinders = 4
4 17
pchance = 0.135
cylinders = 5
1 0
Predict bad
cylinders = 6
8 0
Predict bad
cylinders = 8
9 1
pchance = 0.085
maker = america
0 10
Predict good
maker = asia
2 5
pchance = 0.317
maker = europe
2 2
pchance = 0.717
horsepower = low
0 0
Predict bad
horsepower = medium
0 1
Predict good
horsepower = high
9 0
Predict bad
horsepower = low
0 4
Predict good
horsepower = medium
2 1
pchance = 0.894
horsepower = high
0 0
Predict bad
acceleration = low
1 0
Predict bad
acceleration = medium
0 1
Predict good
acceleration = high
1 1
pchance = 0.717
acceleration = low
1 0
Predict bad
acceleration = medium
1 1
(unexpandable)
Predict bad
acceleration = high
0 0
Predict bad
modelyear = 70to74
0 1
Predict good
modelyear = 75to78
1 0
Predict bad
modelyear = 79to83
0 0
Predict bad

Classification of a new example

- What to predict on x with
  - cylinders = 4
  - maker = Europe
  - acceleration = high
  - year = 76 → 75–78



Today's lecture

Decision Trees

Learning Decision Trees

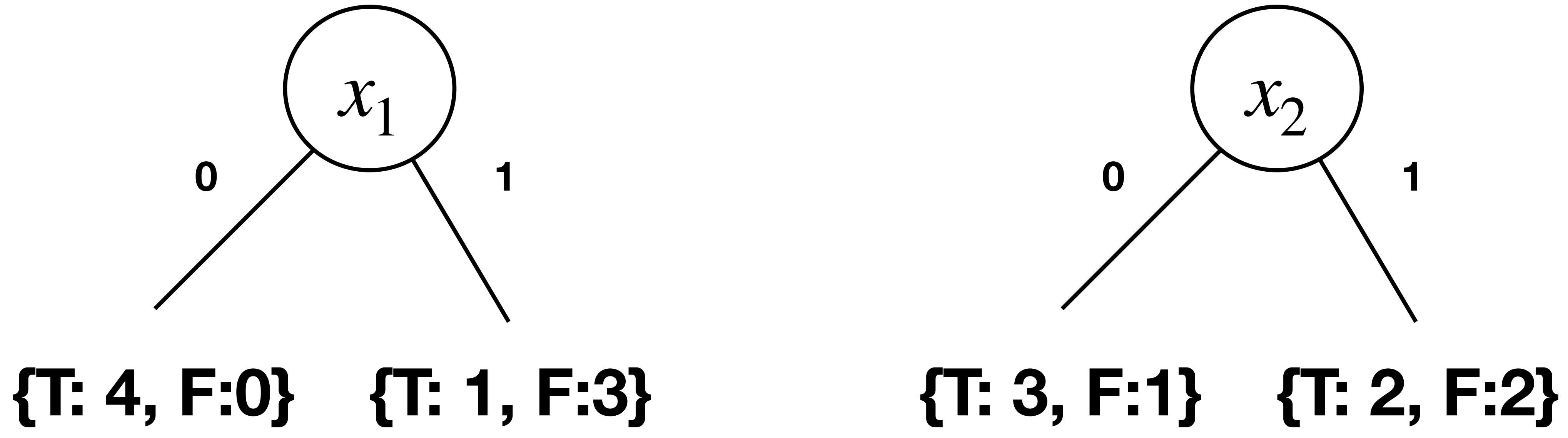
Complexity of Decision Trees

Learning Decision Trees is hard

- Many trees represent the same decision function
  - Some are smaller → more efficient, less complexity
- Finding the smallest Decision Tree is an NP-complete problem [Hyafil & Rivest '76]
- Greedy heuristic:
  - Start from empty decision tree
  - Split on "best" feature x_i: label root, split data to children
  - Repeat for each sub-tree, until no more features (or all data have same y)
  - Label leaf with majority y

Which split is best?

- "To grow a tree, start with a stump"
  - Select its feature
  - But which of $x_1, \ldots, x_n$?



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

- Solve this, and greedy recursion gives entire tree

Measuring uncertainty

- Good splits reduce uncertain about y
  - Consider a branch of the tree: $b = (x_i = v_i, x_j = v_j, \ldots)$
  - Best distribution $p_{\mathscr{D}}(y \mid b)$: deterministic (all true or all false)
  - Worst distribution $p_{\mathscr{D}}(y \mid b)$: uniform

P(Y=A) = 1/2	P(Y=B) = 1/4	P(Y=C) = 1/8	P(Y=D) = 1/8

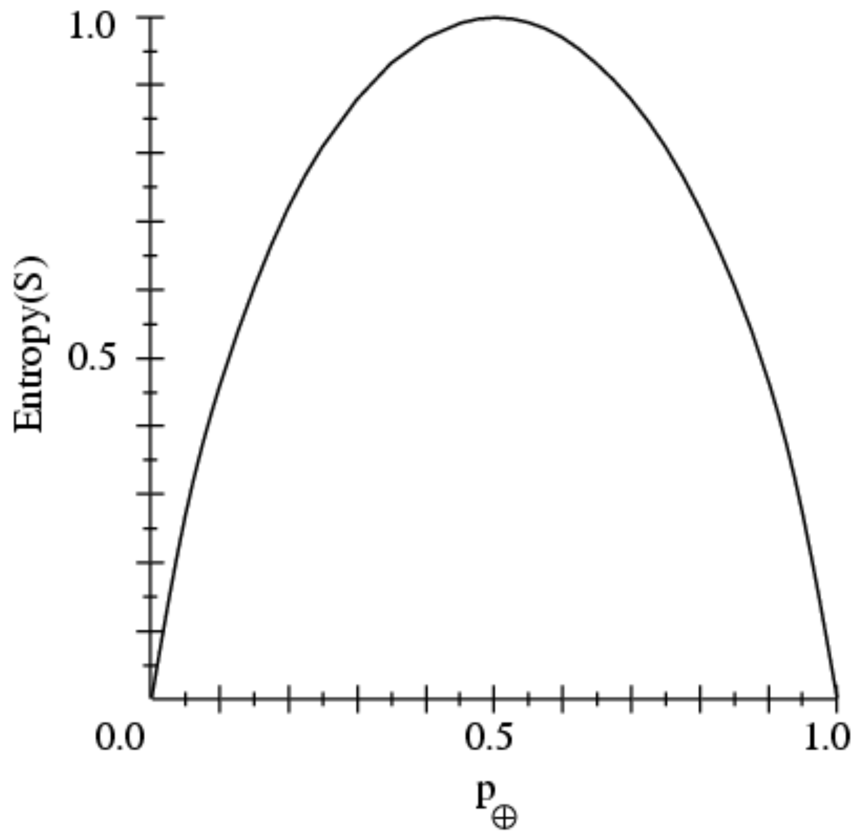
P(Y=A) = 1/4	P(Y=B) = 1/4	P(Y=C) = 1/4	P(Y=D) = 1/4

Entropy

- How surprised are we to see $y = c$? Surprisal = $s_y(c) = -\log p(y=c)$
  - If $\log$ is in base 2 (divide by $\log 2$): number of bits, on average, to efficiently encode $y = c$
- Entropy $\mathbb{H}[y]$ of a random variable y:

$$\mathbb{H}[y] = \mathbb{E}[s_y(c)] = -\sum_c p(y=c)\log p(y=c)$$

- More uncertainty $\implies$ more surprisal (on average) $\implies$ more entropy
- Example: binary variable $y \sim \mathrm{Bernoulli}(p)$
  - $\mathbb{H}[y] = -p\log p - (1-p)\log(1-p)$



Entropy reduction

- Select feature that most decreases uncertainty

- Entropy of y in branch b (before the next split):

$$\mathbb{H}[y|b] = -\sum_c p(y=c|b)\log p(y=c|b)$$

$$= -\frac{5}{8}\log\frac{5}{8} - \frac{3}{8}\log\frac{3}{8} = 0.66$$

- Entropy after splitting by x_1:

$$\mathbb{H}[y|b,x_1] = \mathbb{E}_{x_1|b}[\mathbb{H}[y|b,x_1]] = -\sum_v p(x_1=v|b)\sum_c p(y=c|b,x_1=v)\log p(y=c|b,x_1=v)$$

$$= -\frac{4}{8}(\frac{4}{4}\log\frac{4}{4} + \frac{0}{4}\log\frac{0}{4}) - \frac{4}{8}(\frac{1}{4}\log\frac{1}{4} + \frac{3}{4}\log\frac{3}{4}) = 0.28$$

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

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

- Information gain = reduction in entropy from conditioning y on x_1
  - The amount of new information that x_1 has on y

$$\mathbb{I}[x_1; y \mid b] = \mathbb{H}[y \mid b] - \mathbb{H}[y \mid b, x_1] = 0.66 - 0.28 = 0.38$$

$$\mathbb{I}[x_2; y \mid b] = 0.66 - 0.63 = 0.03$$

select x_1 for Decision Tree

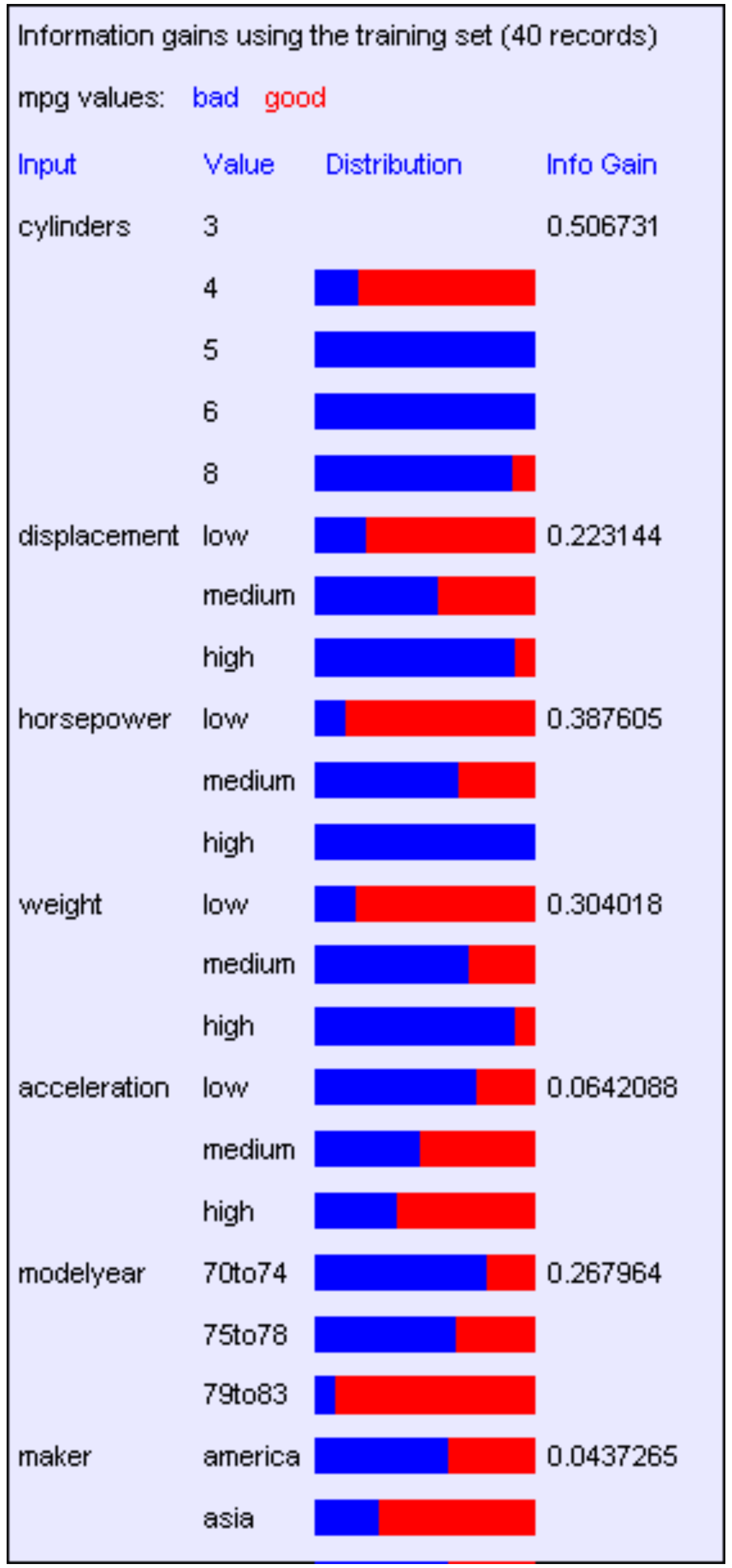
- Information gain is always non-negative
  - By convexity of the entropy

Learning Decision Trees

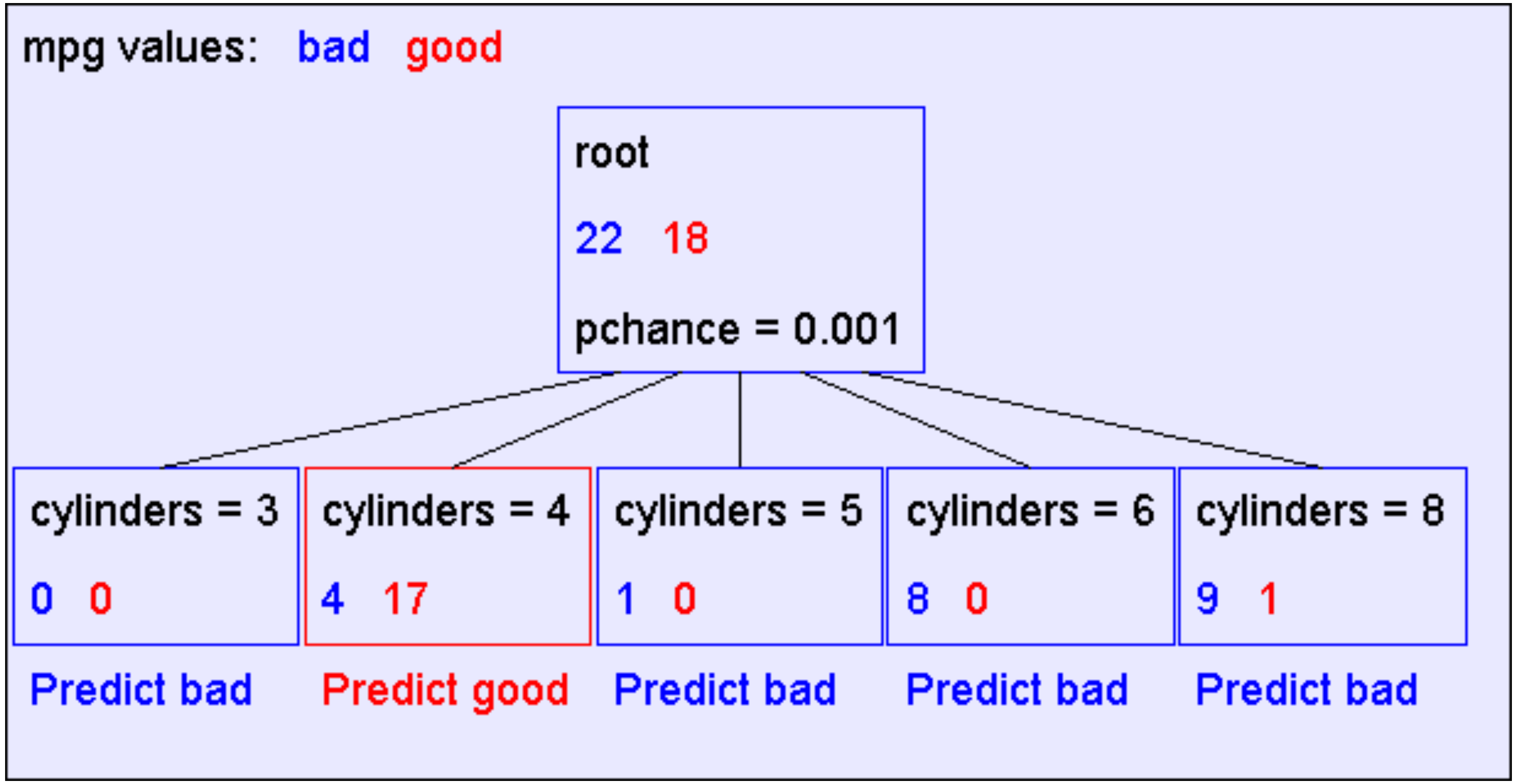
- Start from empty decision tree
- Split on max-info-gain feature x_i
  - $\arg\max_i \mathbb{I}[x_i; y \mid b] = \arg\max_i \mathbb{H}[y \mid b] - \mathbb{H}[y \mid b, x_i]$
- Repeat for each sub-tree, until:
  - Entropy = 0 (all y are the same)
  - No more features
  - Information gain very small?
- Label leaf with majority y

Maximizing information gain

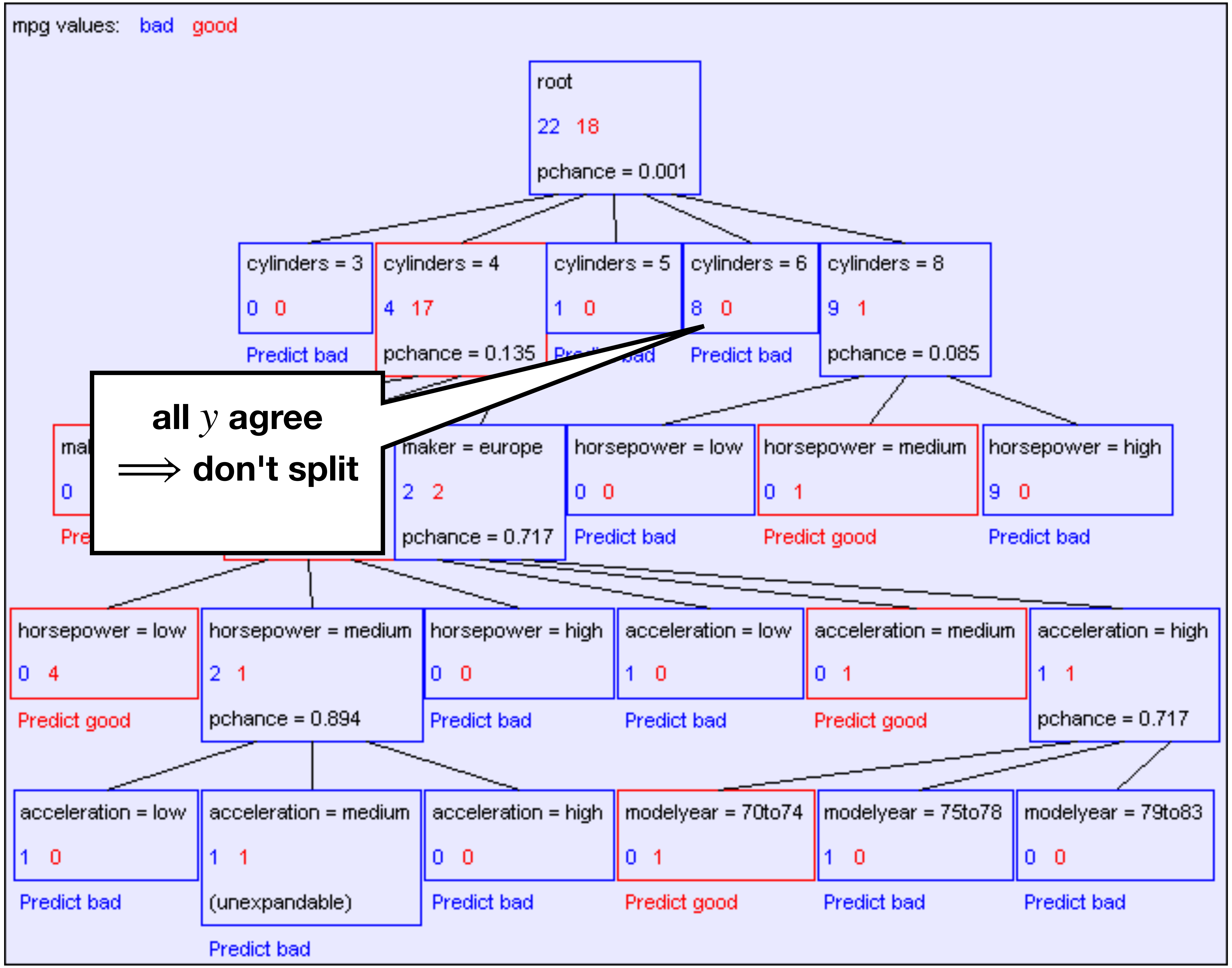
- 7 features are candidates for first split:
  - Cylinders, displacement, horsepower, …
- Each split involves a finite number of feature values
  - Data subset for each value has both blue and red examples
  - We want low (weighted) average entropies in these subsets
    - Cylinders seems a good split
    - Acceleration seems a bad split
- Information gain quantifies this intuition



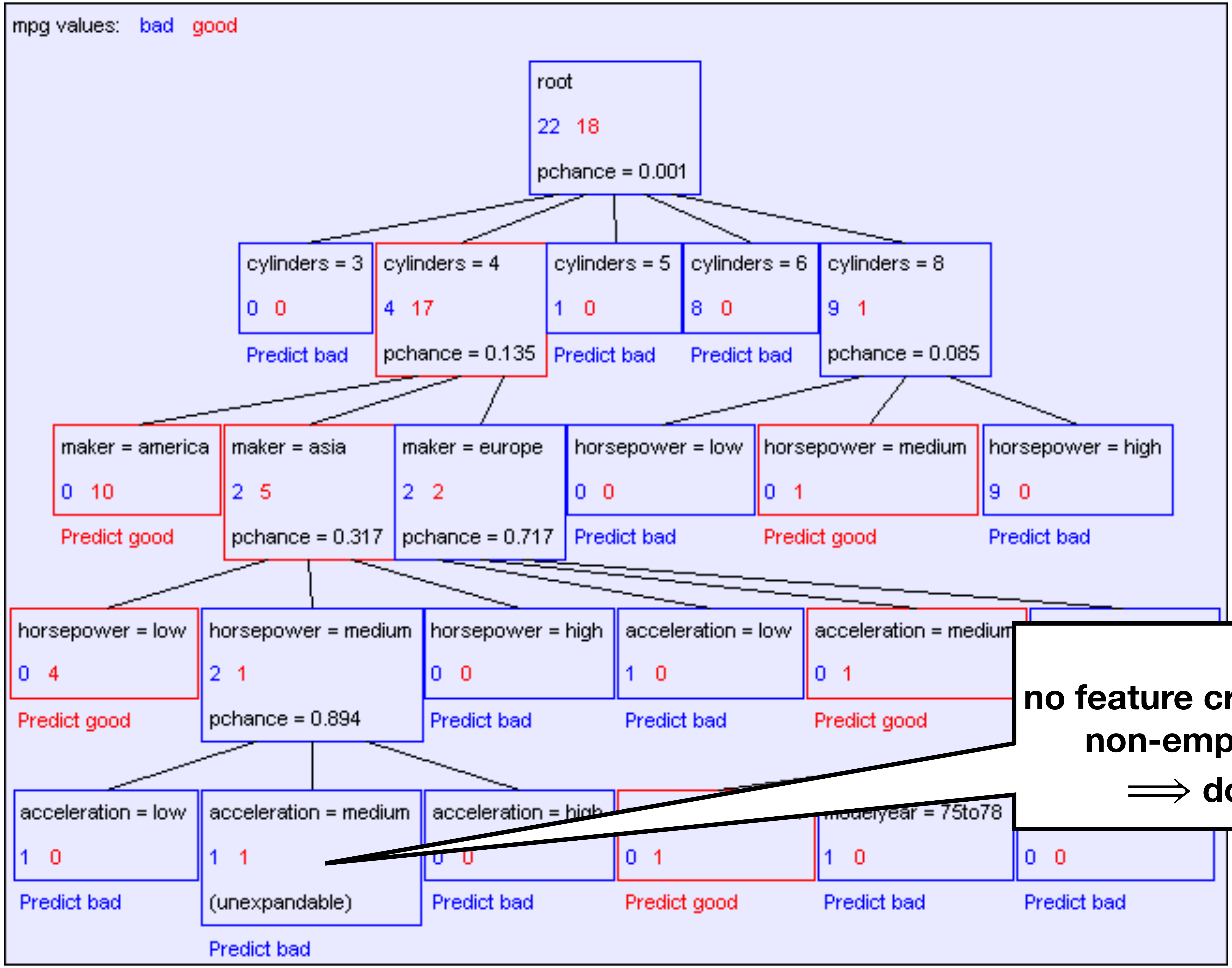
Growing a tree: stopping criteria



Case 1: stop on consensus

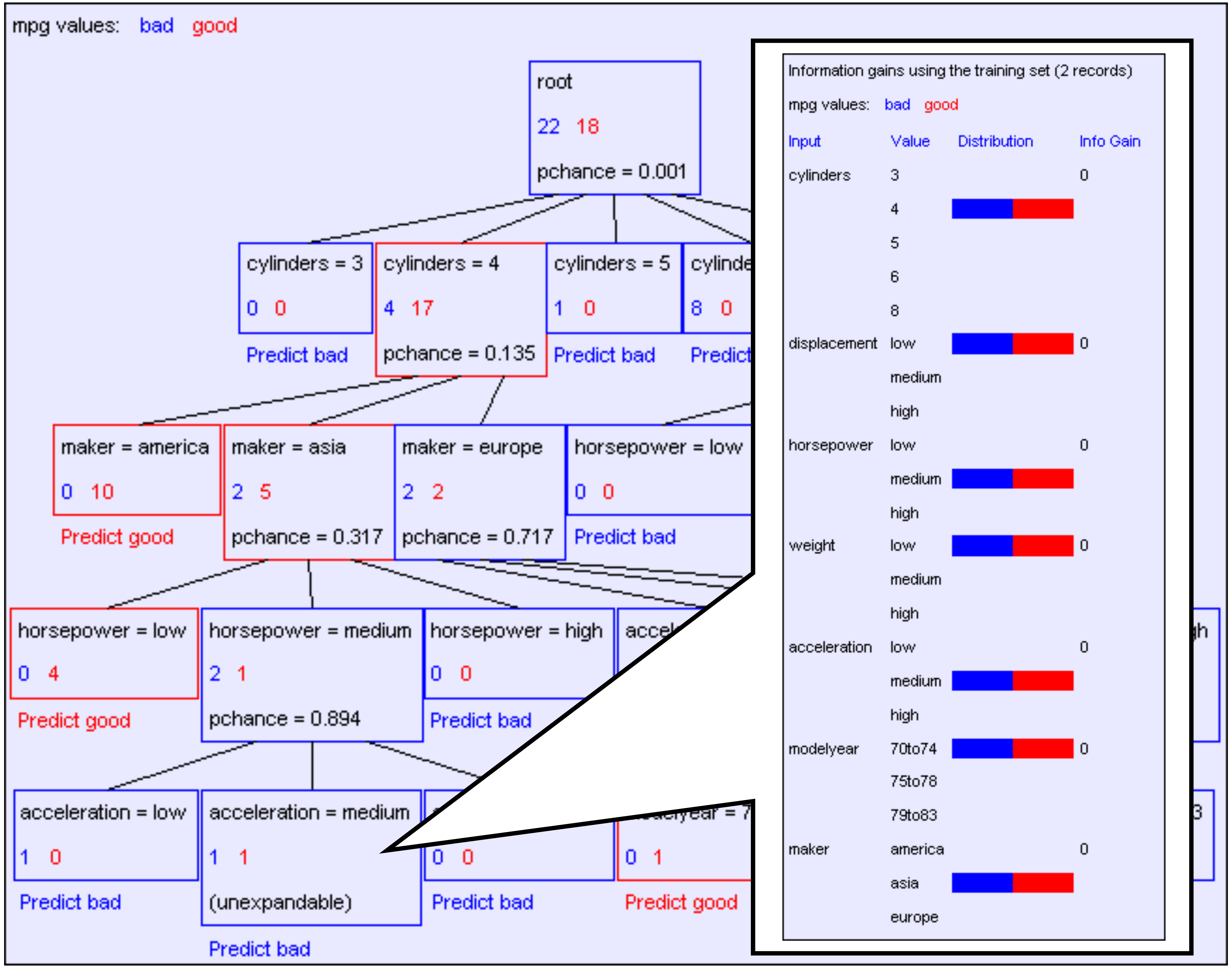


Case 2: stop on no useful features



no feature creates multiple non-empty children
$\Longrightarrow$ don't split

Case 2: stop on no useful features



Stopping criteria

- Stopping criterion 1: consensus = all data points in the branch agree on y
- Stopping criterion 2: no useful features = all data points agree on x
- Consider stopping criterion 3: no feature gives positive information gain
  - Is this always good?

XOR

x_1	x_2	y
0	0	-1
0	1	1
1	0	1
1	1	-1

$$\mathbb{H}[y] = \mathbb{H}(\tfrac{1}{2}) = 1$$

$$\mathbb{H}[y\,|\,x_1] = \tfrac{1}{2}\mathbb{H}(\tfrac{1}{2})+\tfrac{1}{2}\mathbb{H}(\tfrac{1}{2}) = 1$$

$$\mathbb{H}[y\,|\,x_2] = \tfrac{1}{2}\mathbb{H}(\tfrac{1}{2})+\tfrac{1}{2}\mathbb{H}(\tfrac{1}{2}) = 1$$

no information gain

but consider splitting by both!

Today's lecture

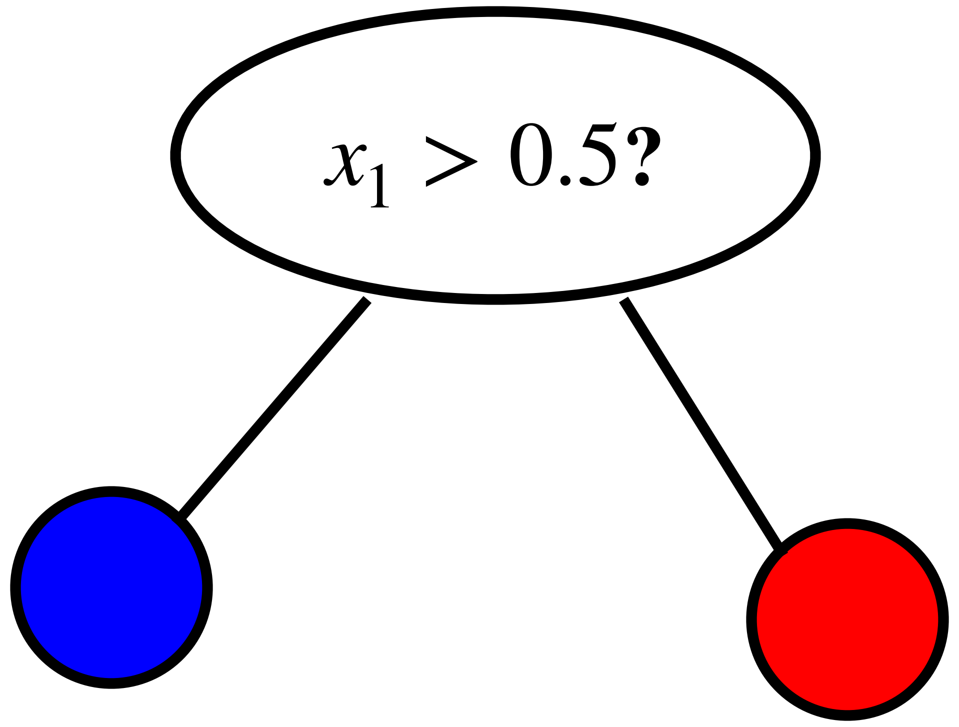
Decision Trees

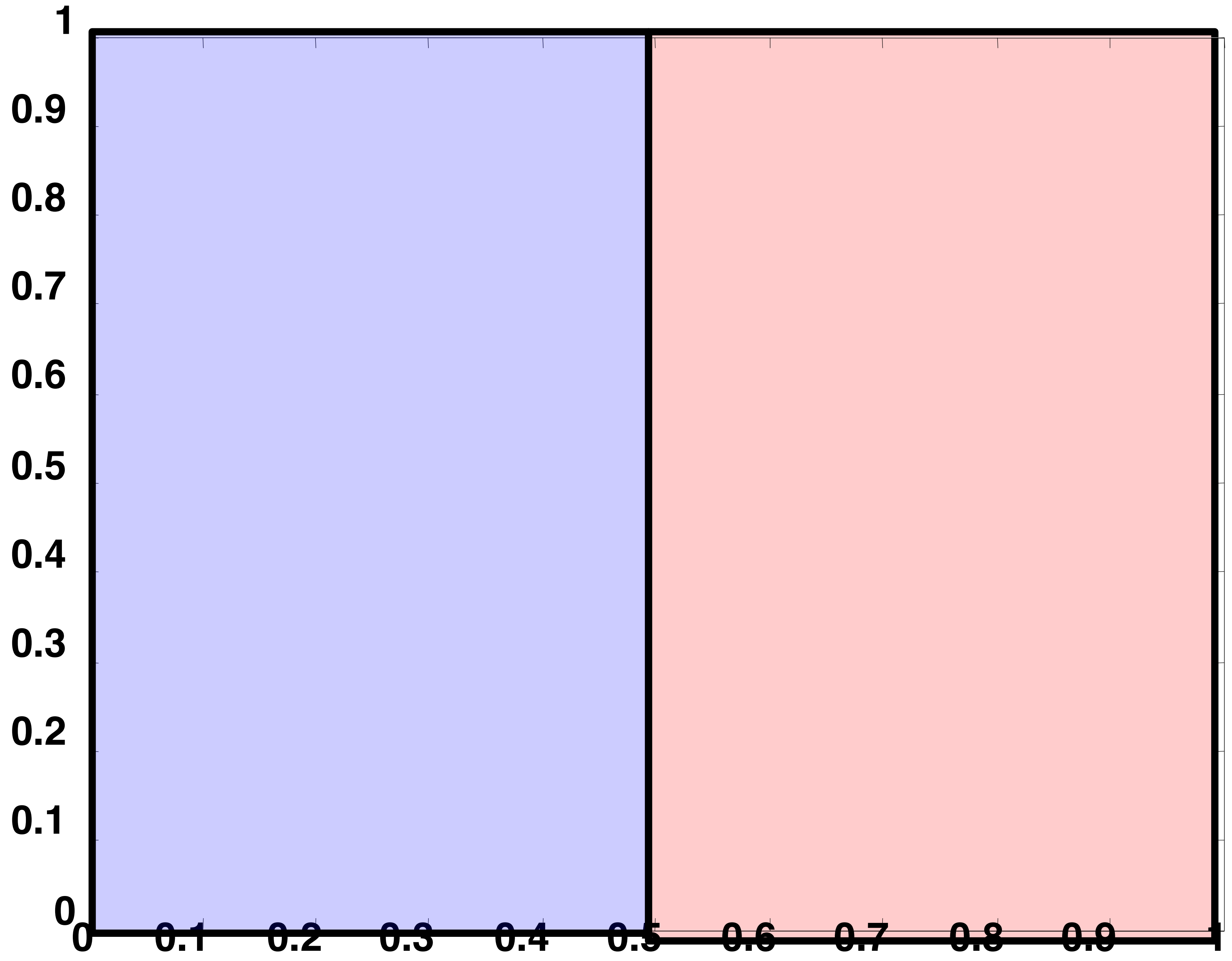
Learning Decision Trees

Complexity of Decision Trees

Continuous features

- How can we split on continuous features?
  - Add binary features $T(x_i - c) = \delta[x_i > c]$
- Decision Stump:

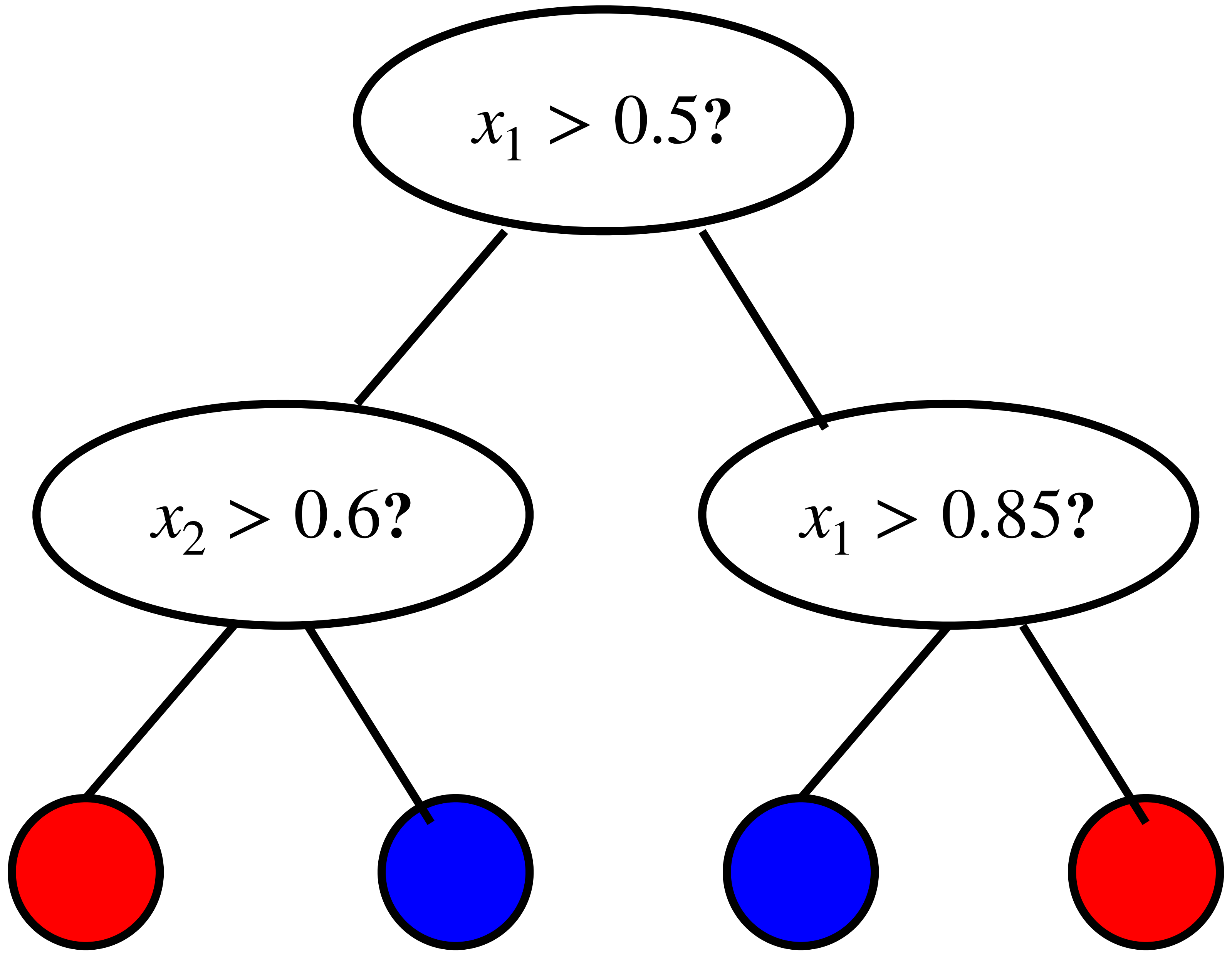


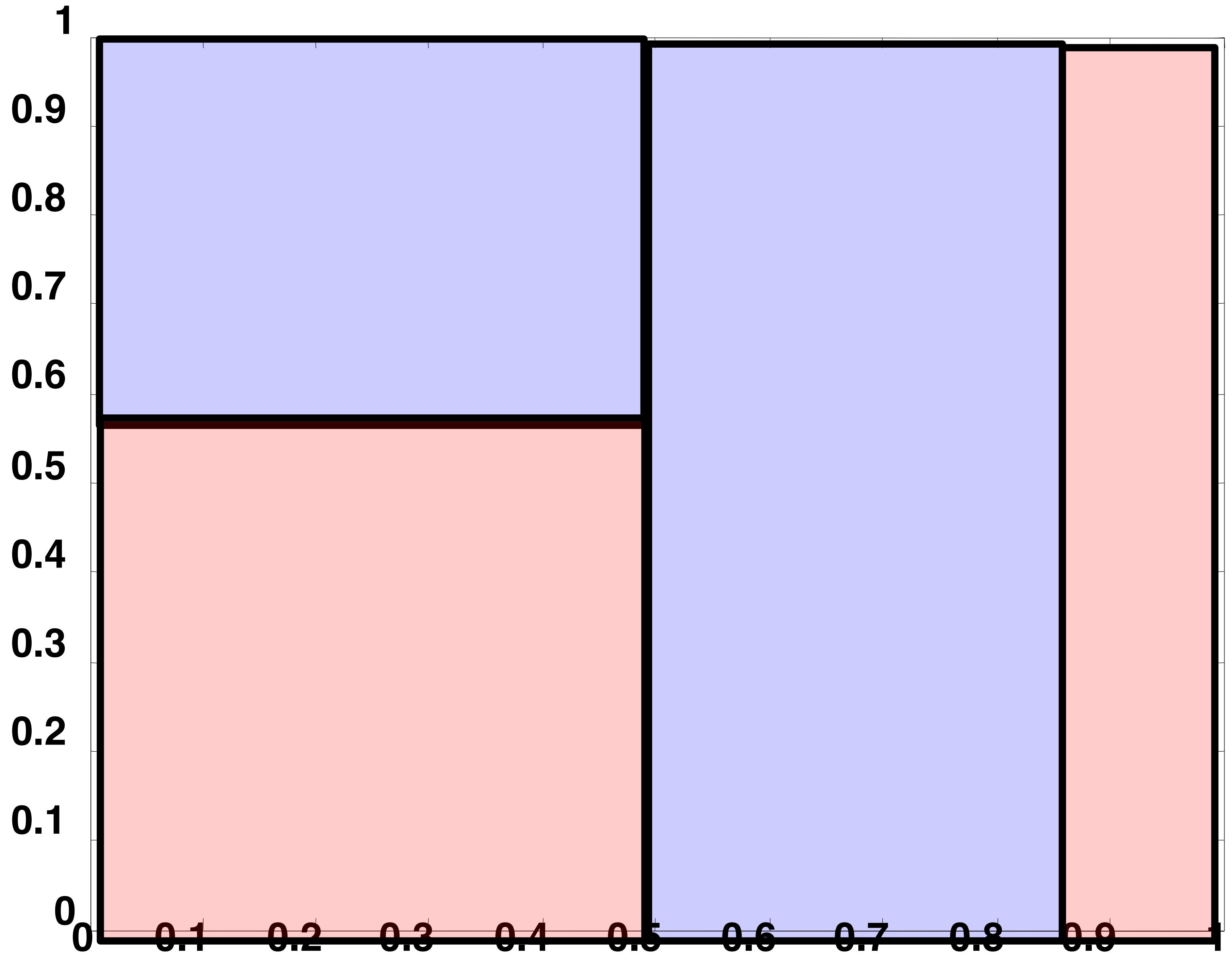


  - Simpler than linear classifier

Decision trees & complexity

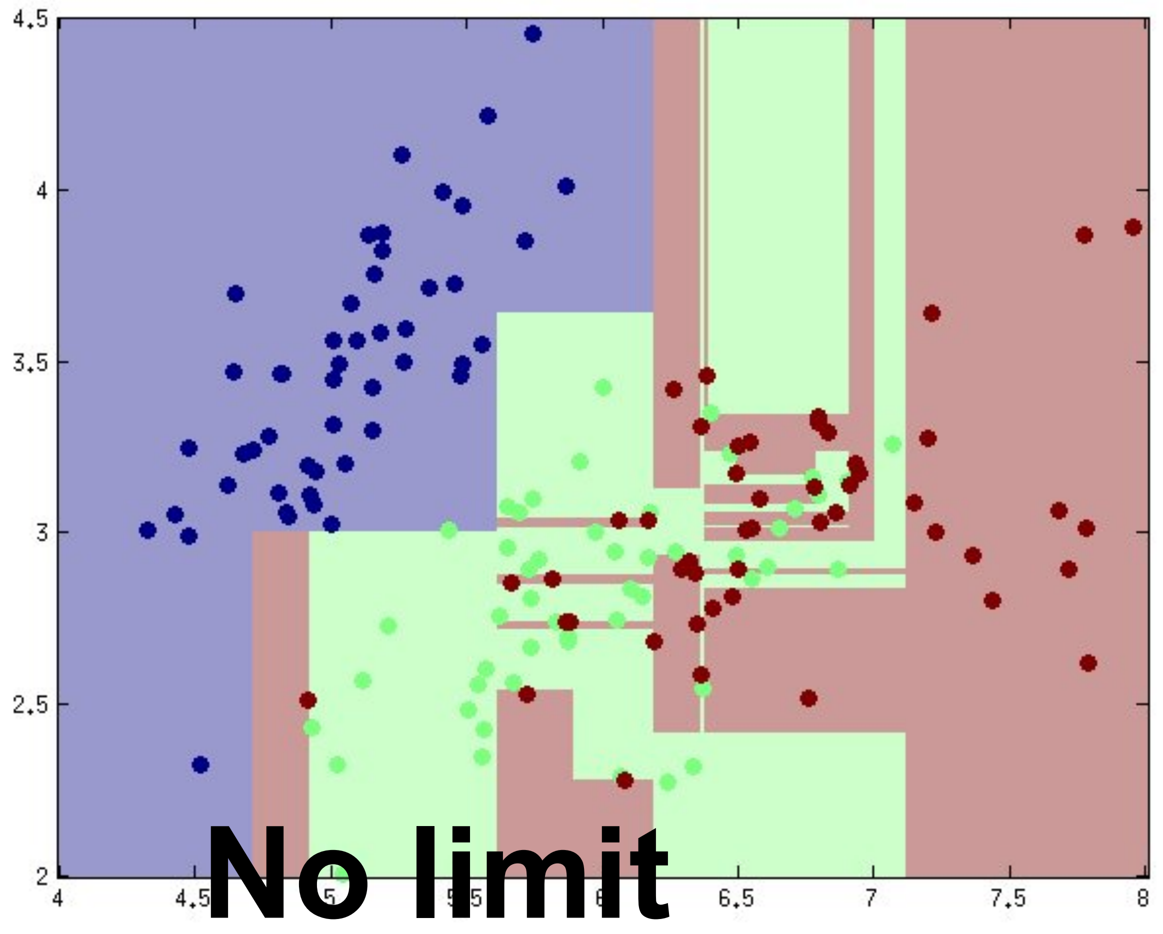
- Complexity of class grows with depth
  - More splits allow finer-grained partitioning

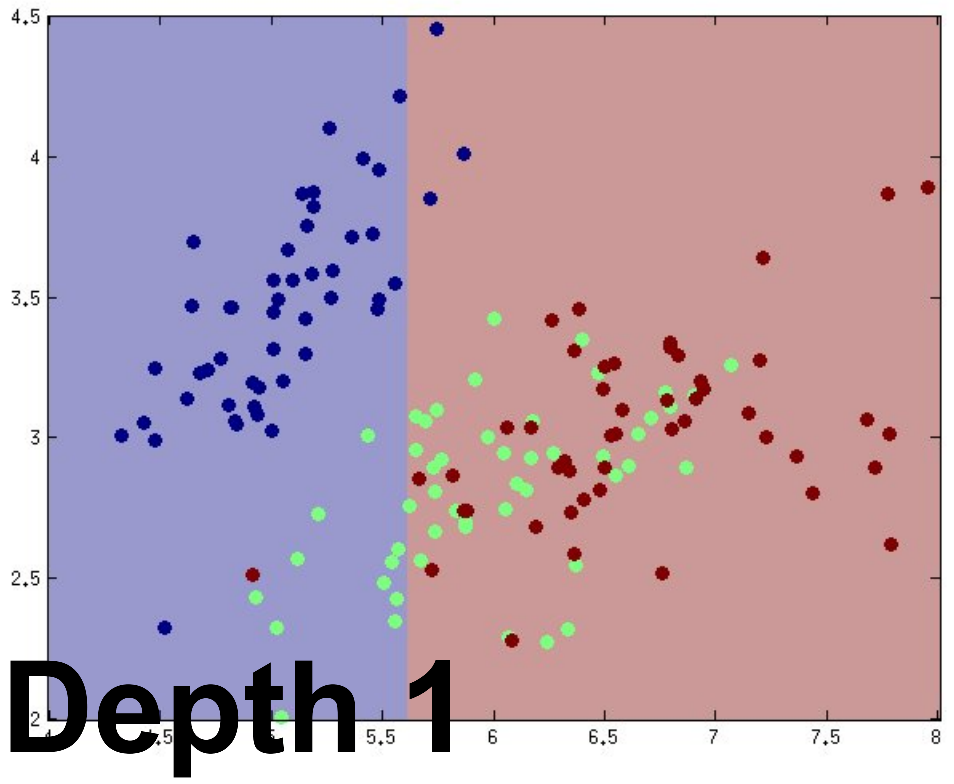


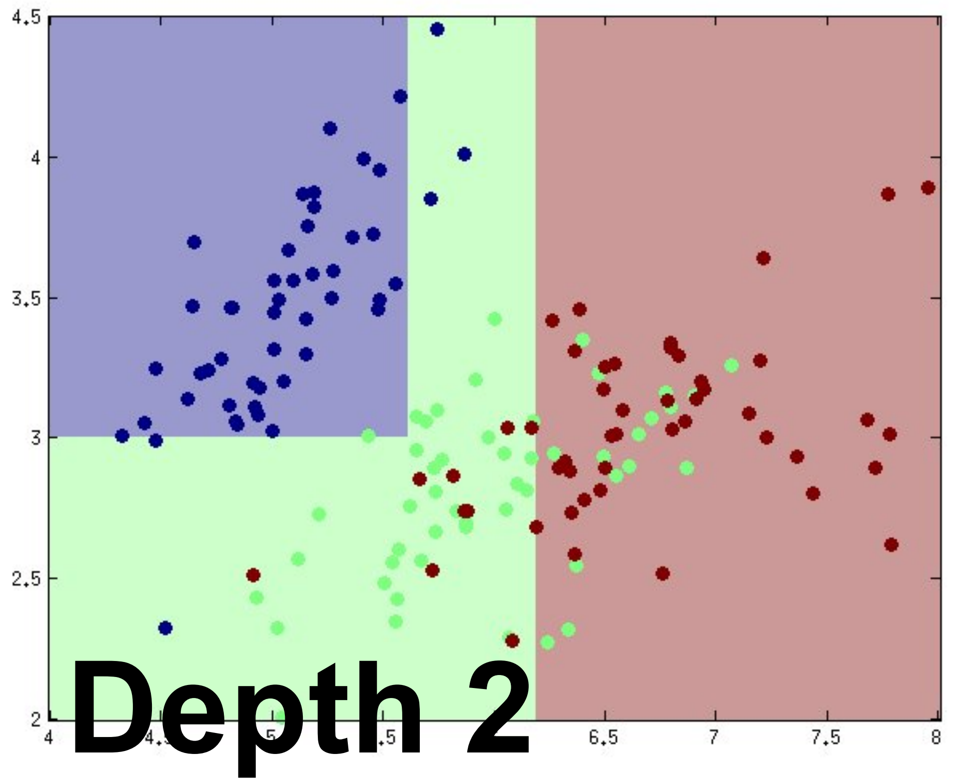


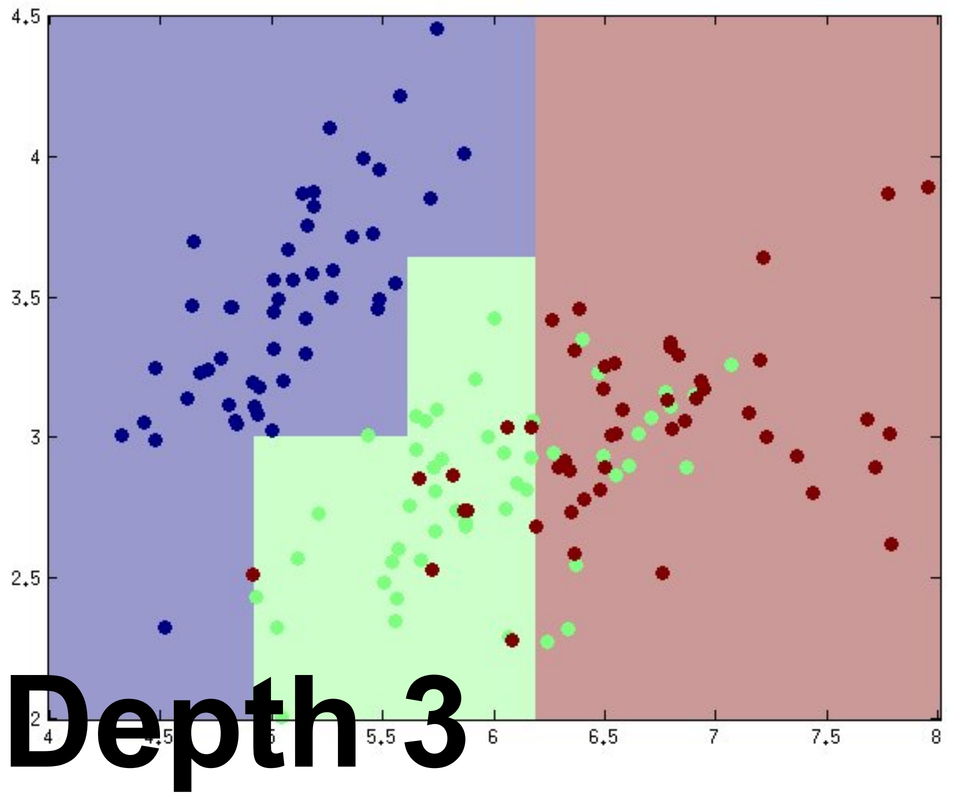
  - up to 2^d regions = leaves

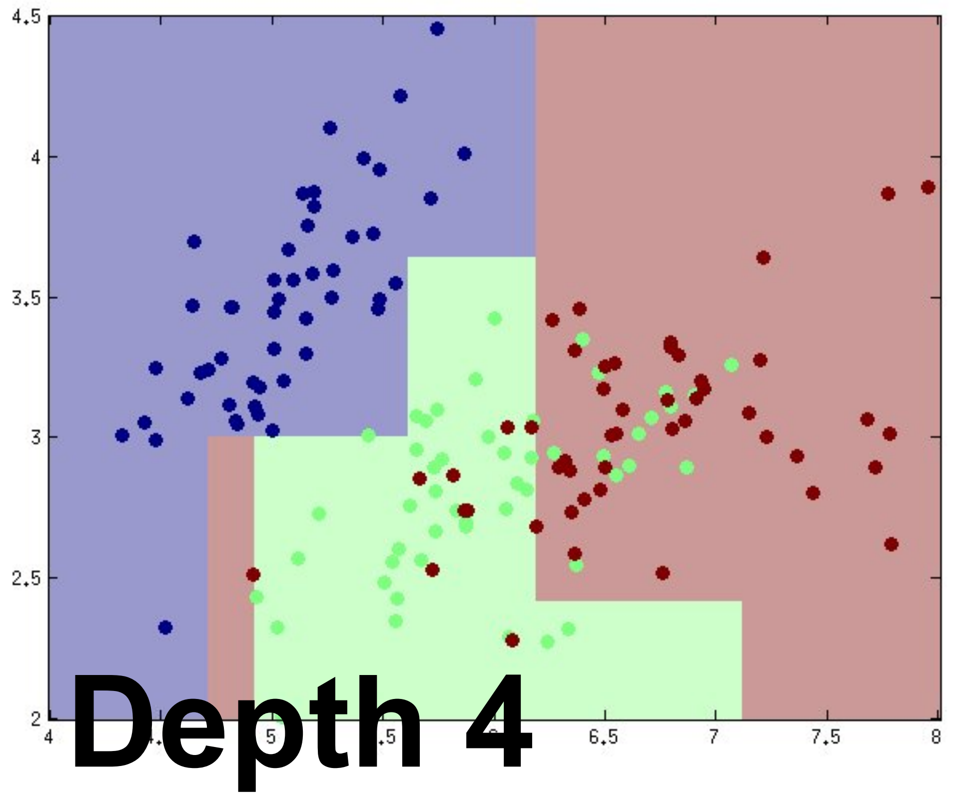
Controlling complexity

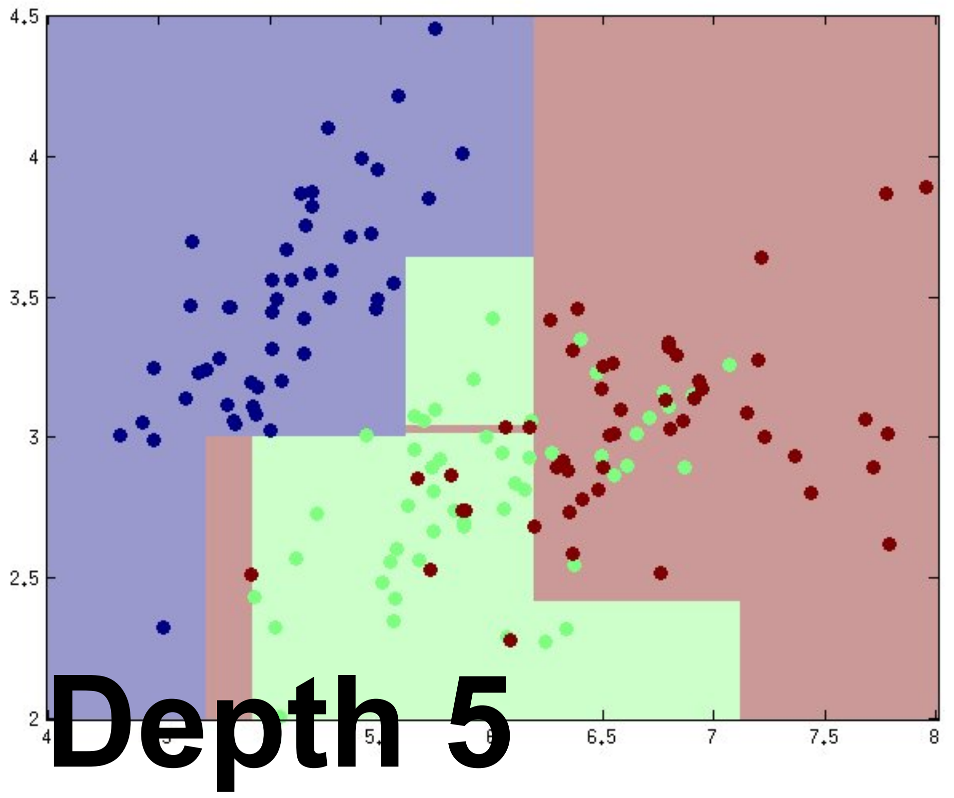












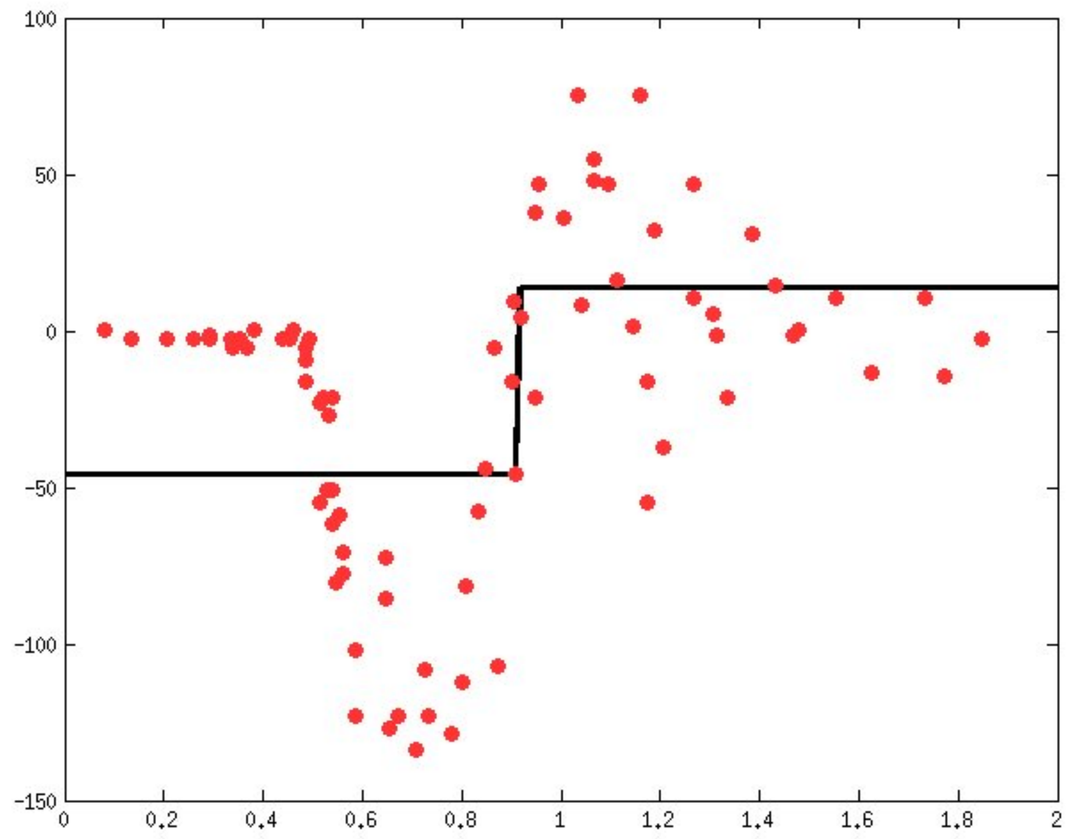
Decision trees will overfit

- Standard decision trees have no inductive bias
  - Training error is always 0, if there is no label noise = $x \to y$ is 1-to-1
  - Danger: high variance! Will overfit if left unstopped!
  - Must bias towards simpler trees
- Many strategies for picking simpler trees:
  - Fixed depth
  - Fixed number of leaves
  - Or something smarter…

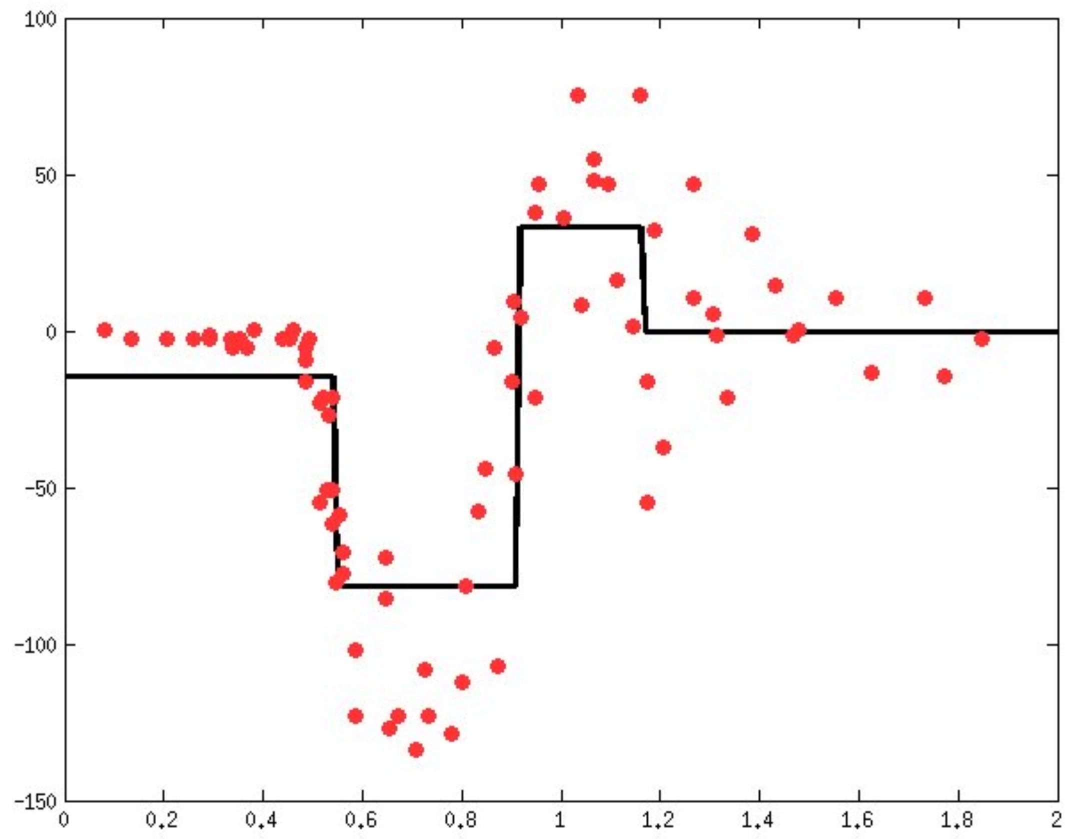
Decision trees for regression

- How to make a prediction of continuous y?
  - Average value of y in a leaf node
- How to compute information gain?
  - Need model of y distribution at node; e.g., Gaussian

depth 1 $\Longrightarrow$ 2 regions



depth 2 $\Longrightarrow$ 4 regions



Recap

- Decision trees
  - Flexible functional form
  - At each node, pick a feature (for continuous: also pick threshold)
  - At leaves, predict a value
- Learning decision trees
  - Score all splits, maximize information gain
  - Apply stopping criteria
- Complexity depends on depth
  - Decision Stumps (aka 1-rule): simpler than linear classifiers

Logistics

project

- Independent project this week

midterm

- Midterm exam next Tue, Feb 9, 2–4pm **on Canvas**
- We'll accommodate other timezones — let us know
- Review during lecture this Thu