Supervised learning methods
Lecture 5

David Sontag
New York University

Slides adapted from Vibhav Gogate, Carlos Guestrin, Luke Zettlemoyer, and Andrew Moore

Plan for next few weeks

February 2014

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Today: nearest neighbor methods, decision trees, random forests

PS4 released tomorrow

Midterm review

March 2014

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Guest lecture: Yann LeCun on deep learning

Spring break: no class or lab!

Midterm exam: in class

Project proposals due

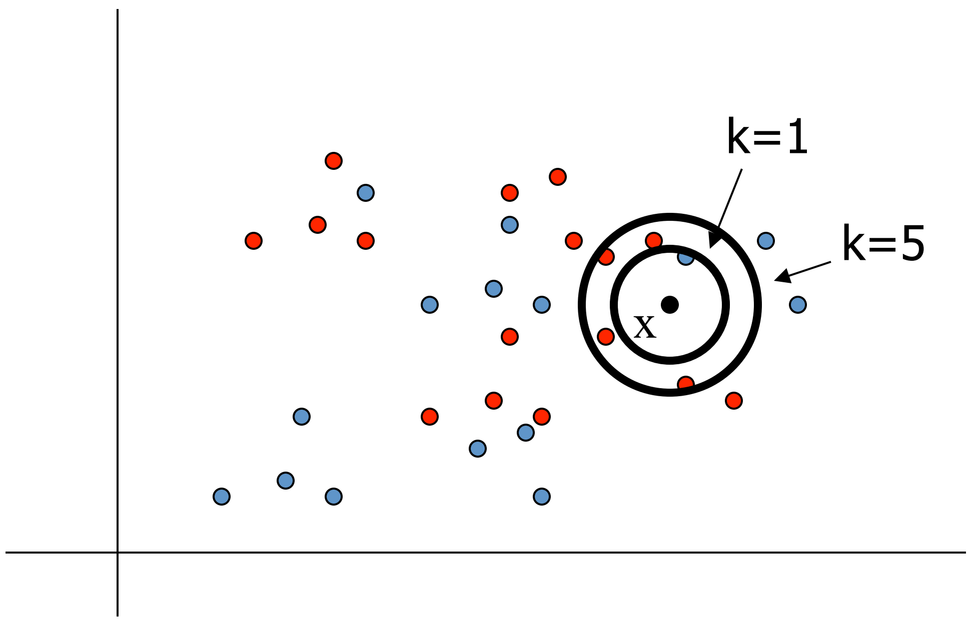
Lab: meet & chat with project advisers (registered students only)

Nearest Neighbor Algorithm

- Learning Algorithm:
  - Store training examples
- Prediction Algorithm:
  - To classify a new example **x** by finding the training example ($\mathbf{x}^i$,y^i) that is *nearest* to **x**
  - Guess the class $y = y^i$

K-Nearest Neighbor Methods

- To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class

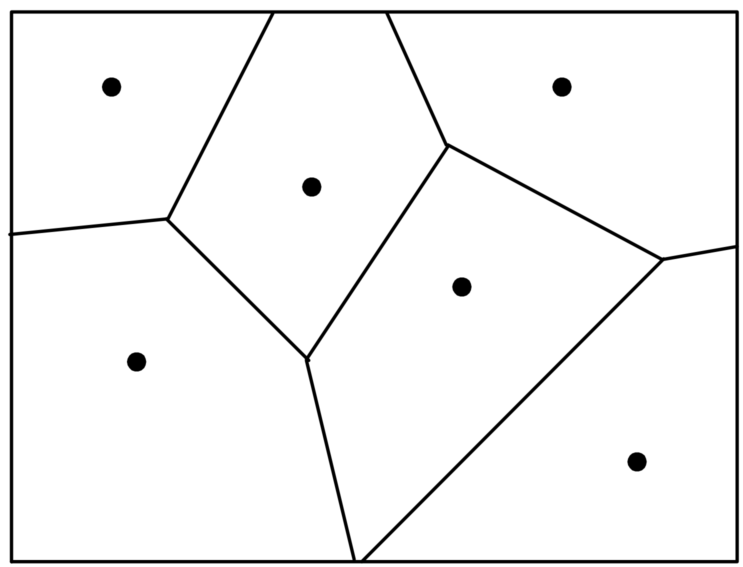


common values for k: 3, 5

Decision Boundaries

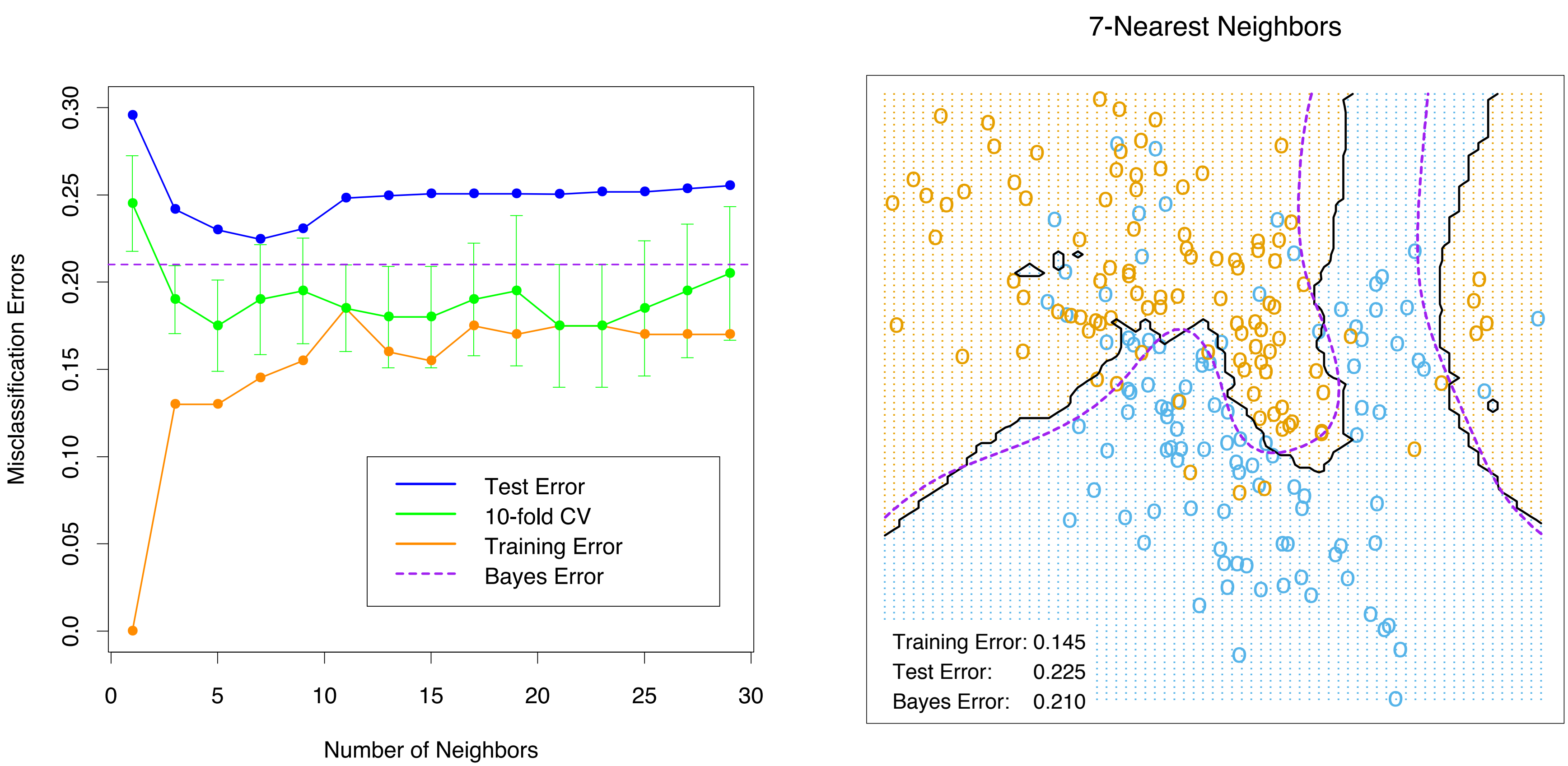
- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

1-NN Decision Surface



- The more examples that are stored, the more complex the decision boundaries can become

Example results for k-NN



[Figures from Hastie and Tibshirani, Chapter 13]

Nearest Neighbor

When to Consider

- Instance maps to points in R^n
- Less than 20 attributes per instance
- Lots of training data

Advantages

- Training is very fast
- Learn complex target functions
- Do not lose information

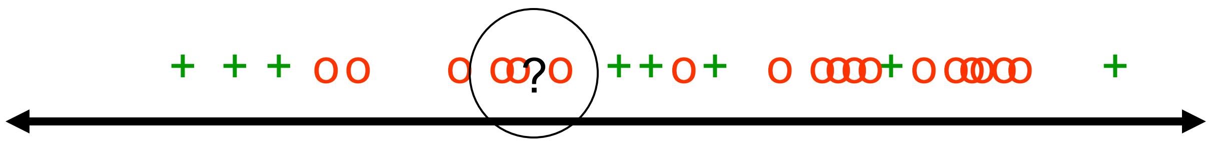
Disadvantages

- Slow at query time
- Easily fooled by irrelevant attributes

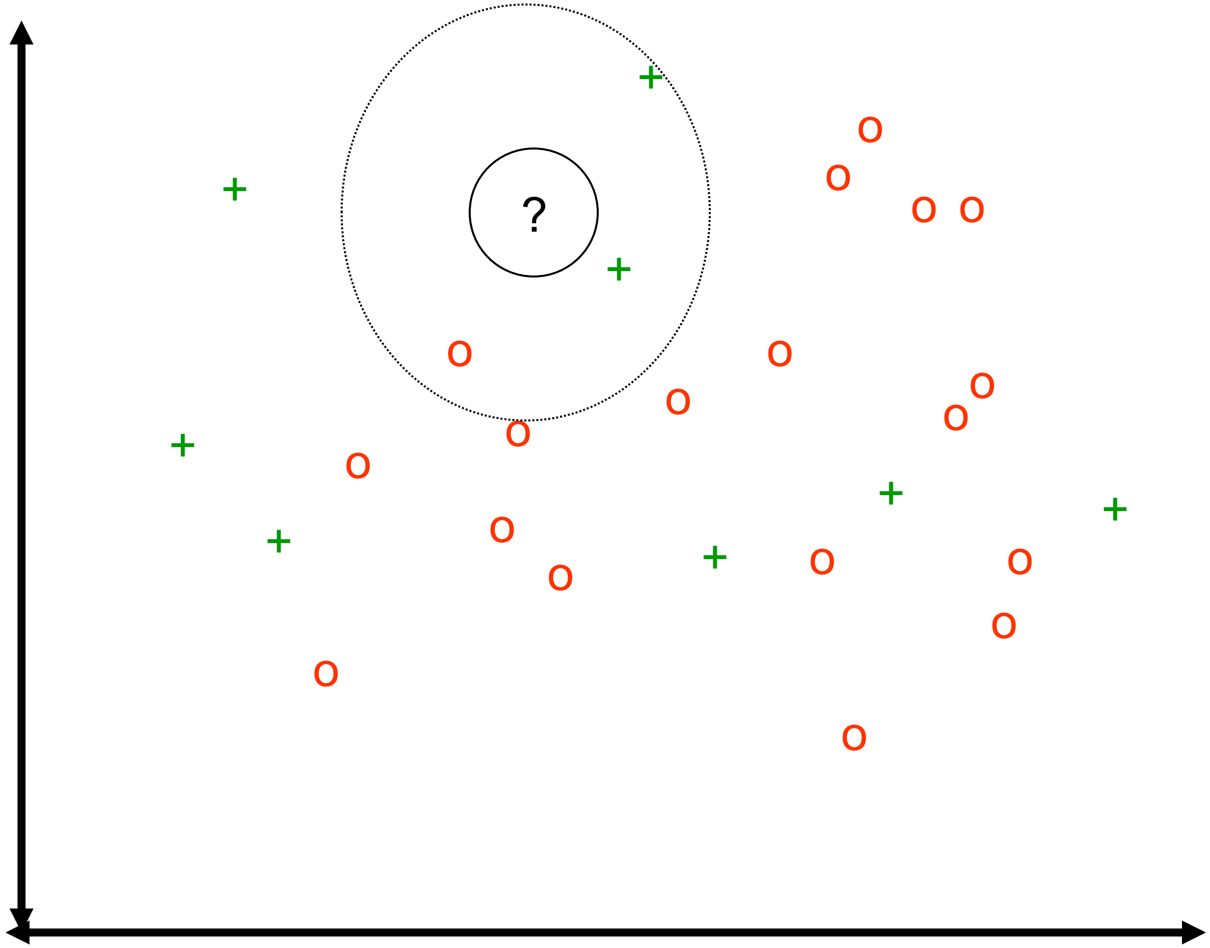
Issues

- Distance measure
  - Most common: Euclidean
- Choosing k
  - Increasing k reduces variance, increases bias
- For high-dimensional space, problem that the nearest neighbor may not be very close at all!
- Memory-based technique. Must make a pass through the data for each classification. This can be prohibitive for large data sets.

k-NN and irrelevant features



k-NN and irrelevant features



Weighted *k*-NN

- Consider the following generalization of the *k*-NN algorithm (specialized to binary classification):

$$\hat{y}(\vec{x}) \leftarrow \operatorname{sign}\left(\sum_{i=1}^{k} y_i s(\vec{x}_i, \vec{x})\right) \text{ with } s(\vec{x}_i, \vec{x}) = \frac{1}{\|\vec{x}_i - \vec{x}\|_2^2} \text{ or... } s(\vec{x}_i, \vec{x}) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}\|_2^2}{2\sigma^2}\right)$$

- Weighs the *i*'th training point's label by how far $\mathbf{x}_i$ is from $\mathbf{x}$

k-NN is similar to SVM with Gaussian kernel!

- Consider the following generalization of the k-NN algorithm (specialized to binary classification):

$$\hat{y}(\vec{x}) \leftarrow \text{sign}\left(\sum_{i=1}^{\text{N}} y_i s(\vec{x}_i, \vec{x})\right) \text{ with } s(\vec{x}_i, \vec{x}) = \frac{1}{\|\vec{x}_i - \vec{x}\|_2^2} \text{ or... } s(\vec{x}_i, \vec{x}) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}\|_2^2}{2\sigma^2}\right)$$

- Looks at *all* training points (i.e., k=N), but weighs the *i*'th training point's label by how far $\mathbf{x}_i$ is from $\mathbf{x}$

- Now compare this to classification with SVM and a Gaussian kernel:

$$\hat{y}(\vec{x}) \leftarrow \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x}_i, \vec{x})\right) \qquad K(\vec{u}, \vec{v}) = \exp\left(-\frac{||\vec{u} - \vec{v}||_2^2}{2\sigma^2}\right) \qquad 0 \leq \alpha_i \leq C$$

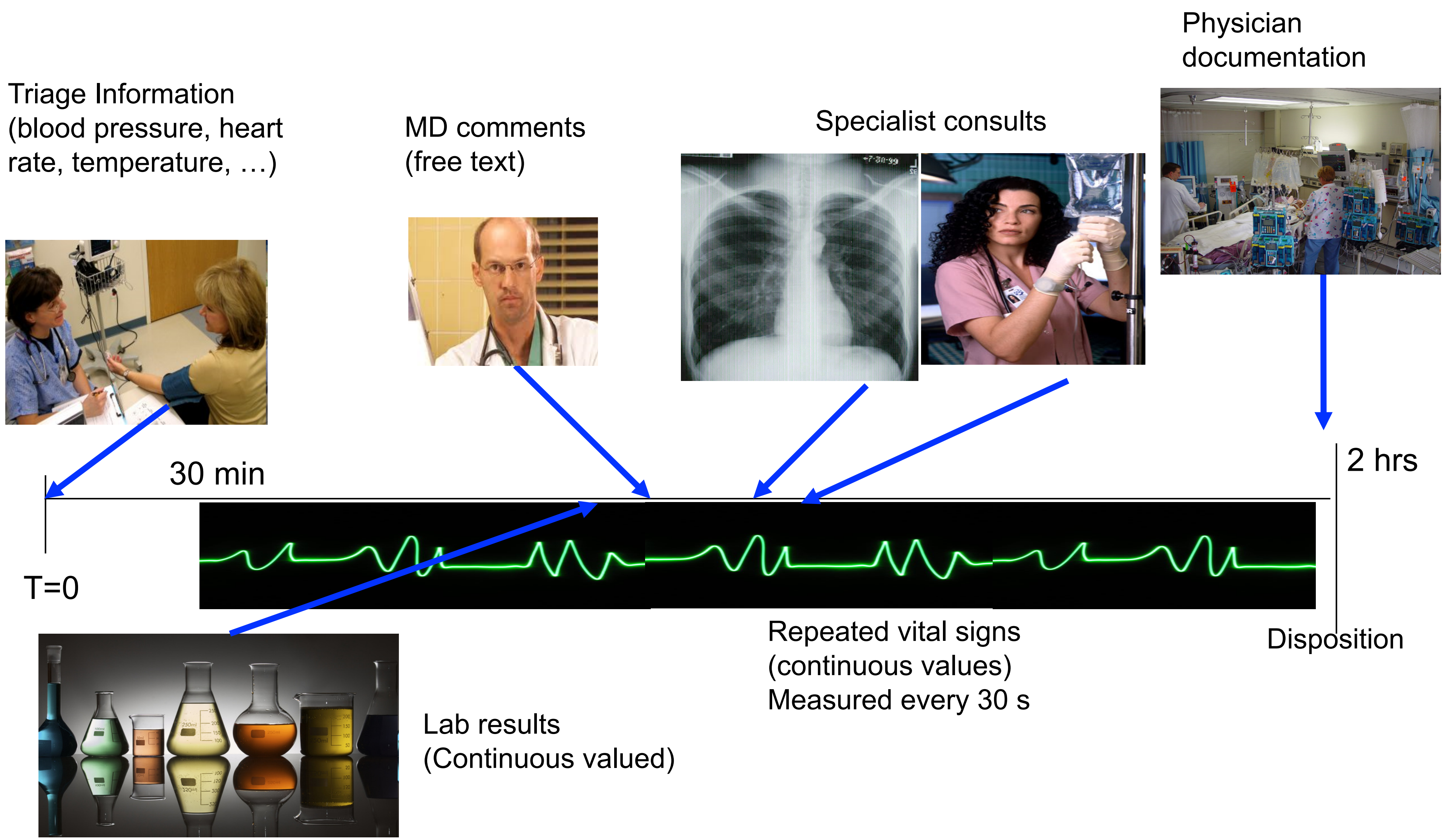
- The discriminant functions are nearly identical! The SVM has parameters α_i that can be learned

KNN Advantages

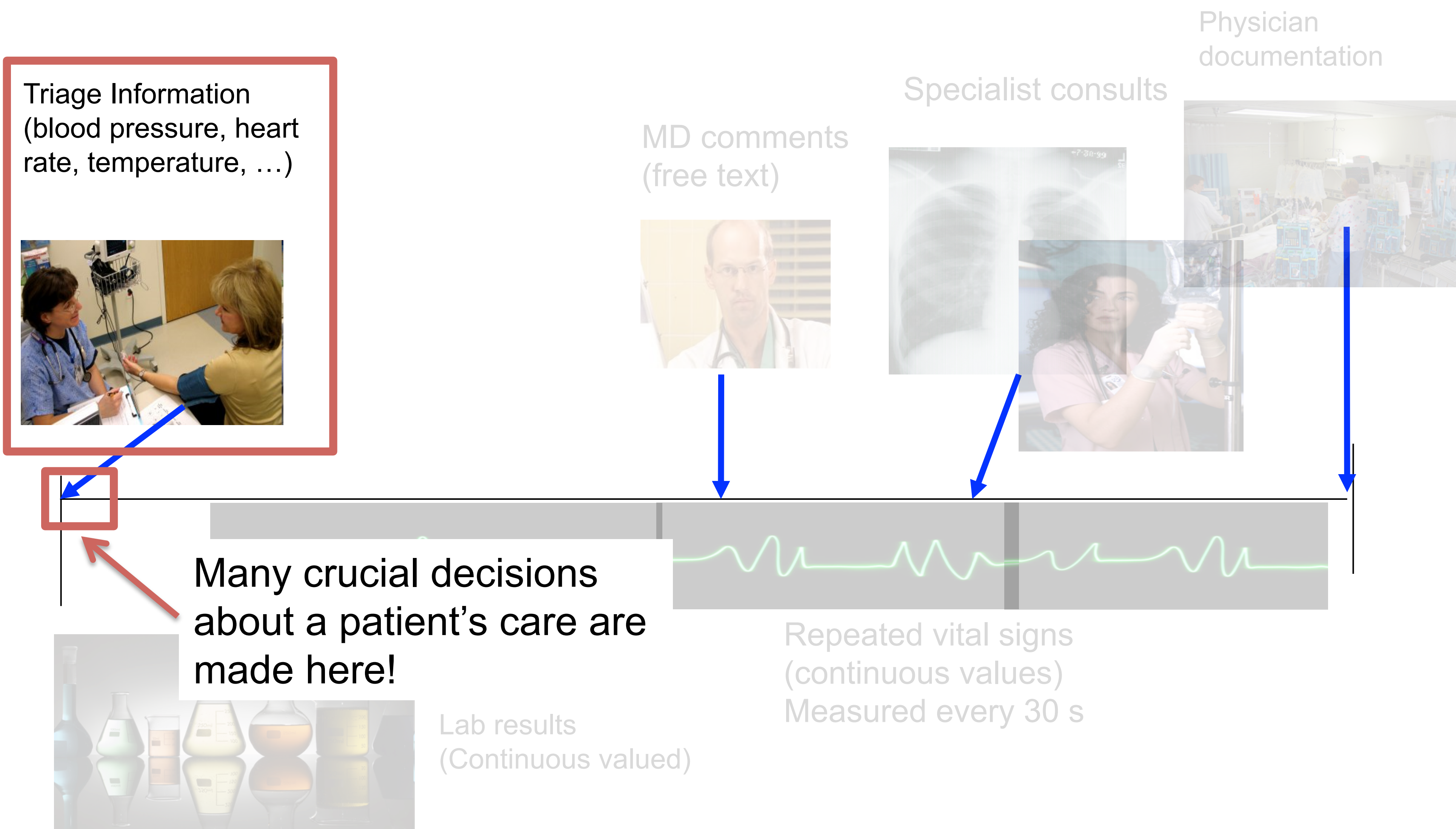
- Easy to program
- No optimization or training required
- Classification accuracy can be very good; can outperform more complex models

Decision Trees

Machine Learning in the ER

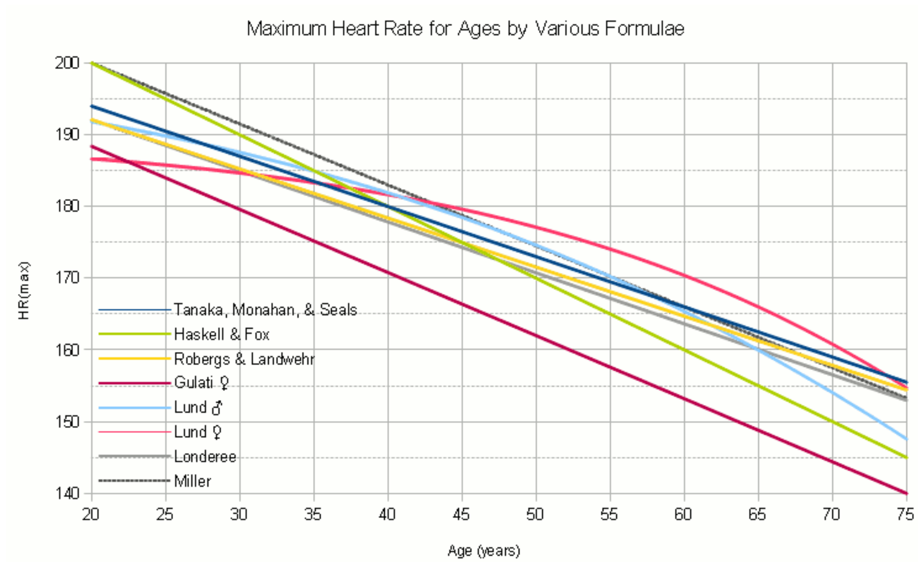


Can we predict infection?



Can we predict infection?

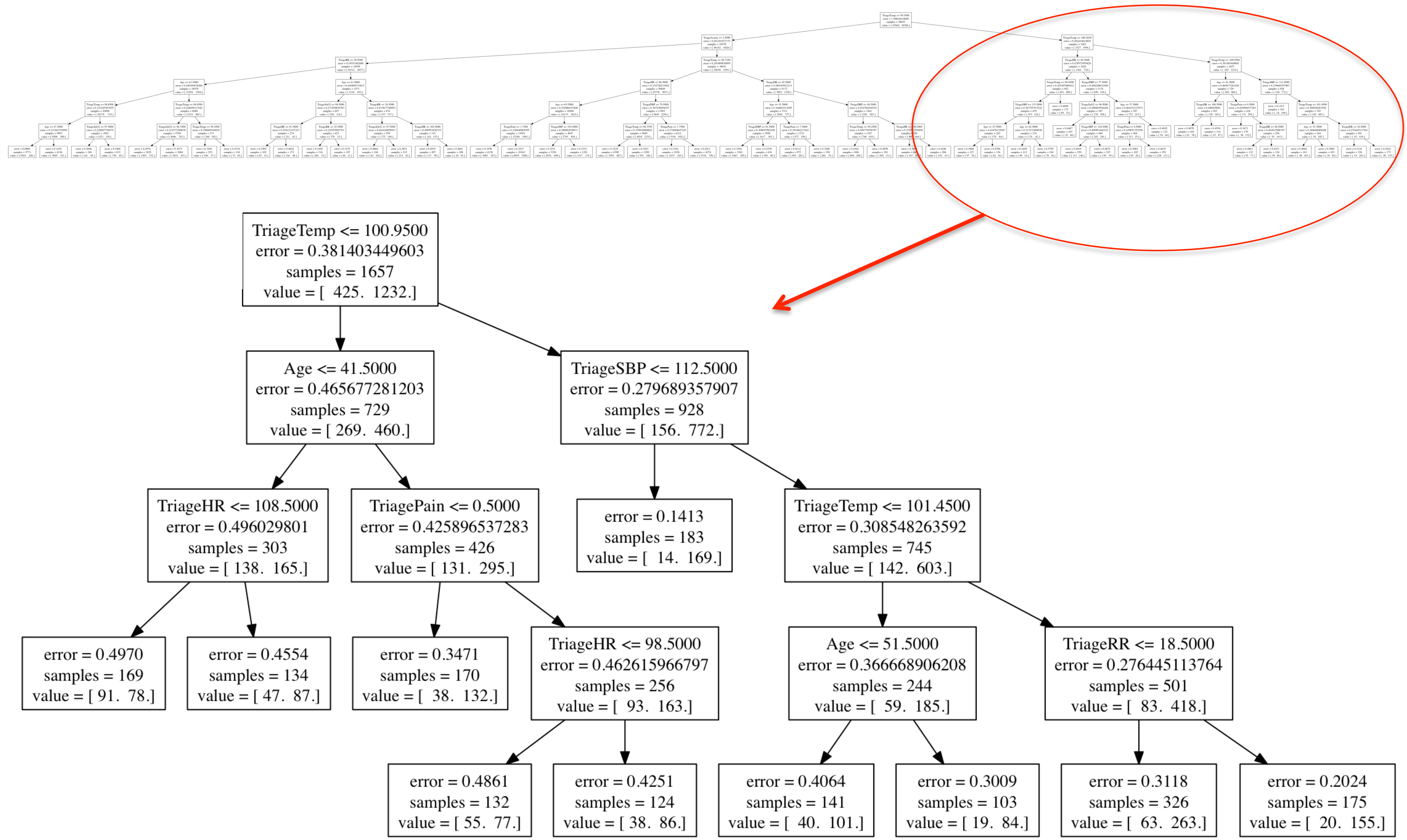
- Previous automatic approaches based on simple criteria:
  - Temperature < 96.8 °F or > 100.4 °F
  - Heart rate > 90 beats/min
  - Respiratory rate > 20 breaths/min
- Too simplified… e.g., heart rate depends on age!



Can we predict infection?

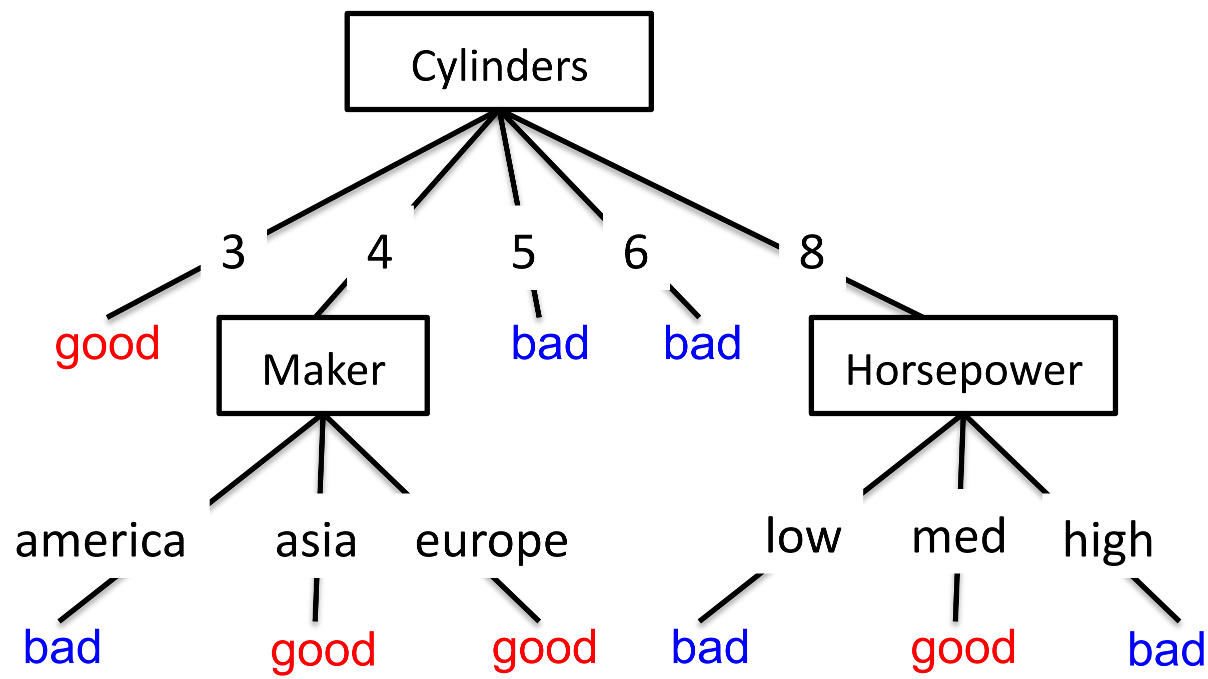
- These are the attributes we have for each patient:
  - Temperature
  - Heart rate (HR)
  - Respiratory rate (RR)
  - Age
  - Acuity and pain level
  - Diastolic and systolic blood pressure (DBP, SBP)
  - Oxygen Saturation (SaO2)
- We have these attributes + label (infection) for 200,000 patients!
- Let’s **learn** to classify infection

Predicting infection using decision trees



Hypotheses: decision trees $f : X \rightarrow Y$

- Each internal node tests an attribute x_i
- One branch for each possible attribute value $x_i = v$
- Each leaf assigns a class y
- To classify input x: traverse the tree from root to leaf, output the labeled y

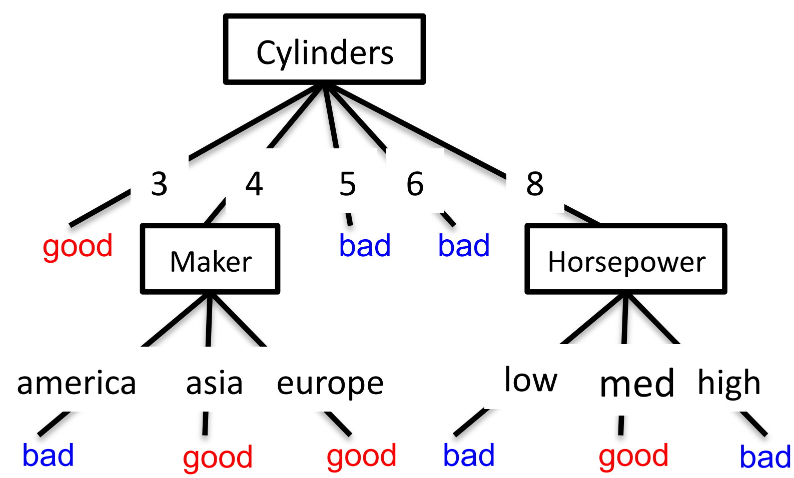


Human interpretable!

Hypothesis space

- How many possible hypotheses?
- What functions can be represented?

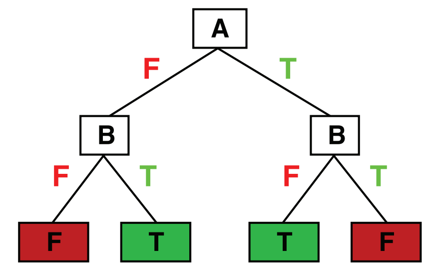
mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low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



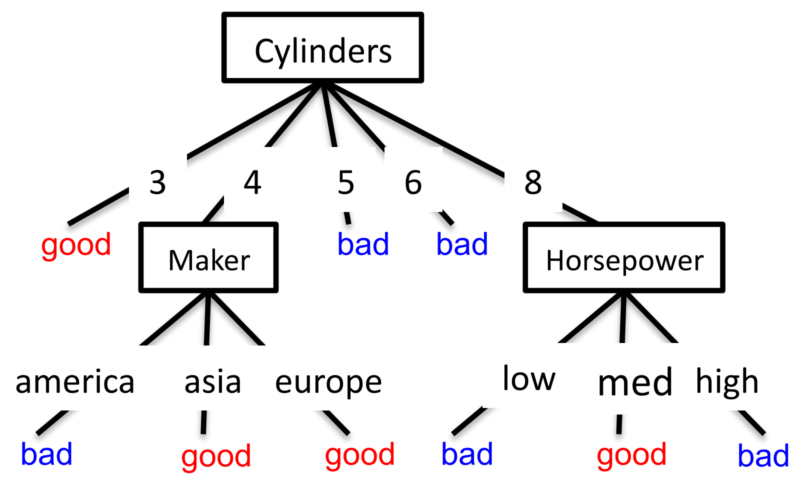
What functions can be represented?

- Decision trees can represent any function of the input attributes!
- For Boolean functions, path to leaf gives truth table row
- But, could require exponentially many nodes…

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



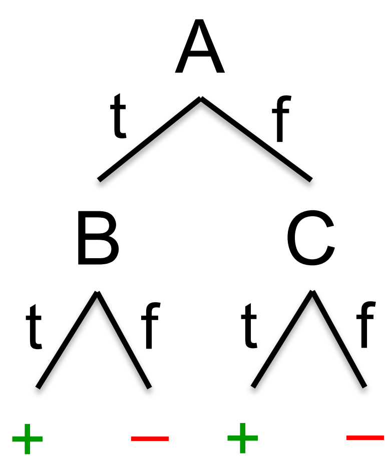
(Figure from Stuart Russell)

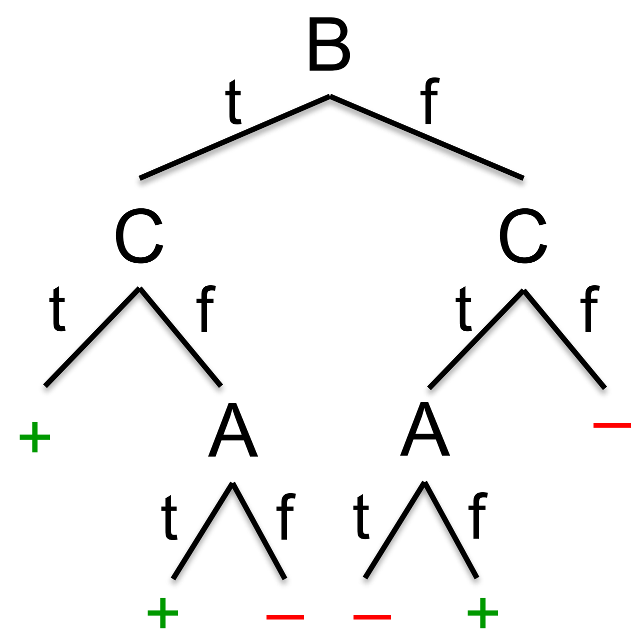


$$cyl{=}3 \lor (cyl{=}4 \land (maker{=}asia \lor maker{=}europe)) \lor \ldots$$

Are all decision trees equal?

- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., $\phi = (A \wedge B) \vee (\neg A \wedge C)$ -- ((A and B) or (not A and C))



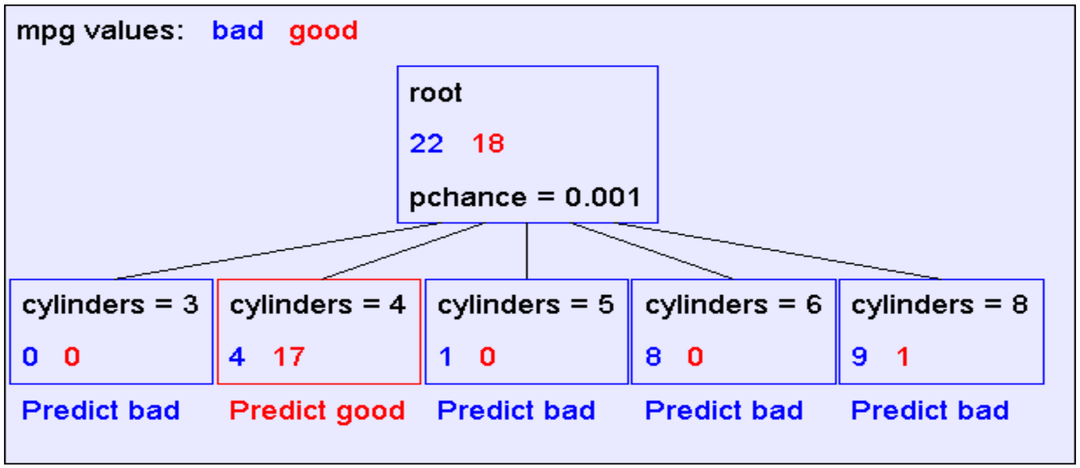


- Which tree do we prefer?

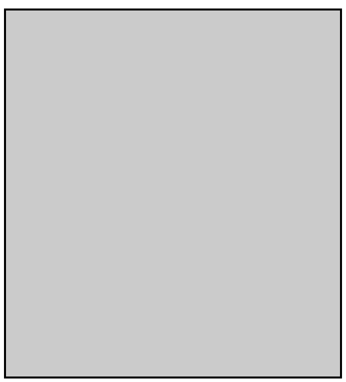
Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse

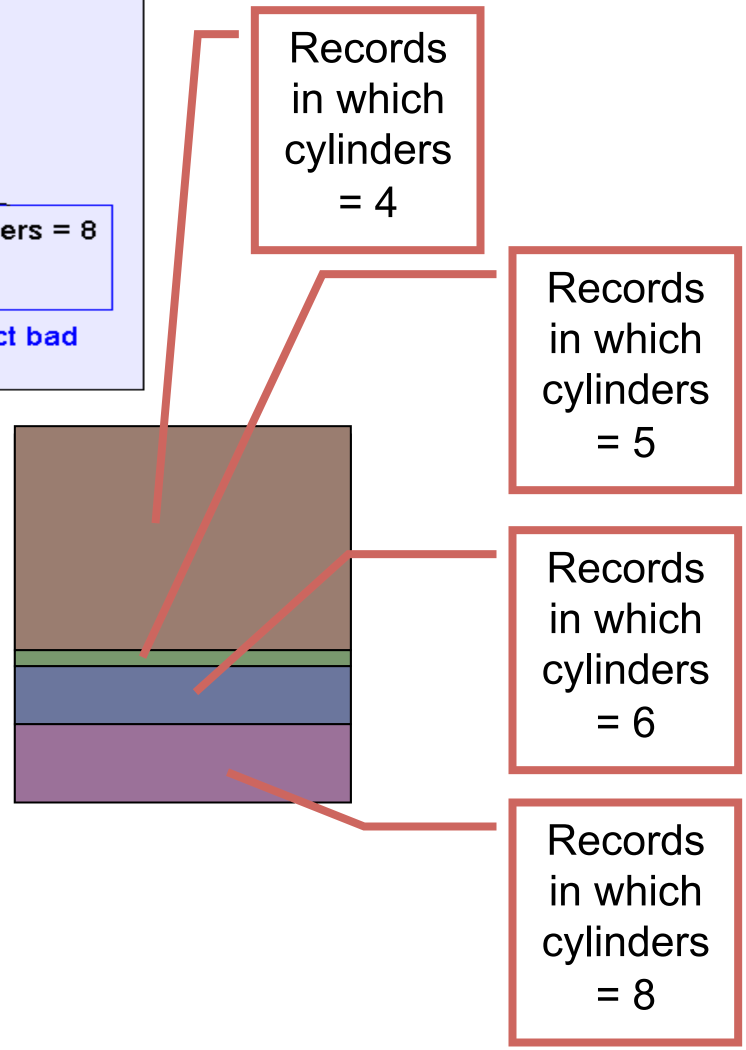
Key idea: Greedily learn trees using **recursion**



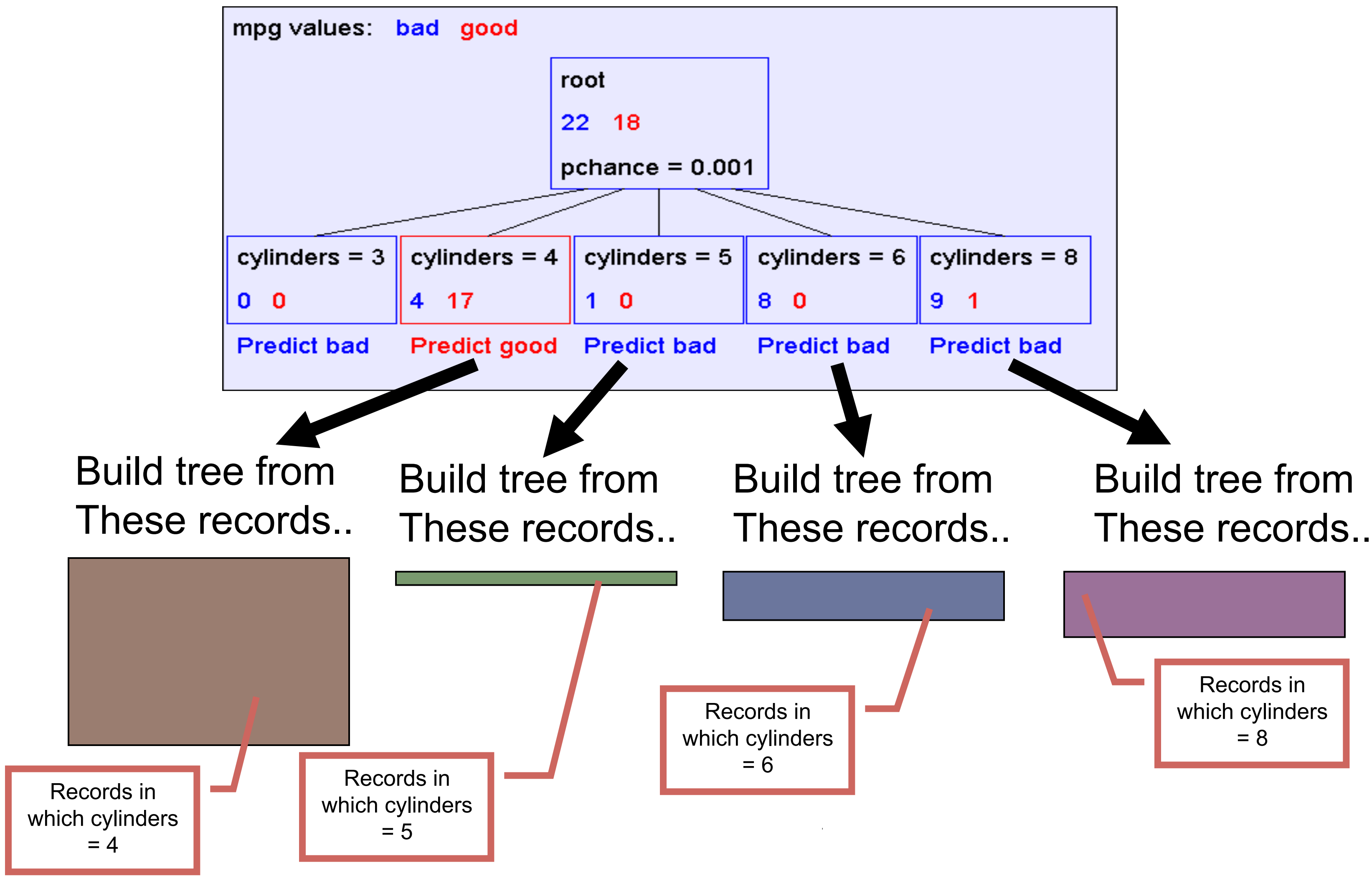
Take the Original Dataset..



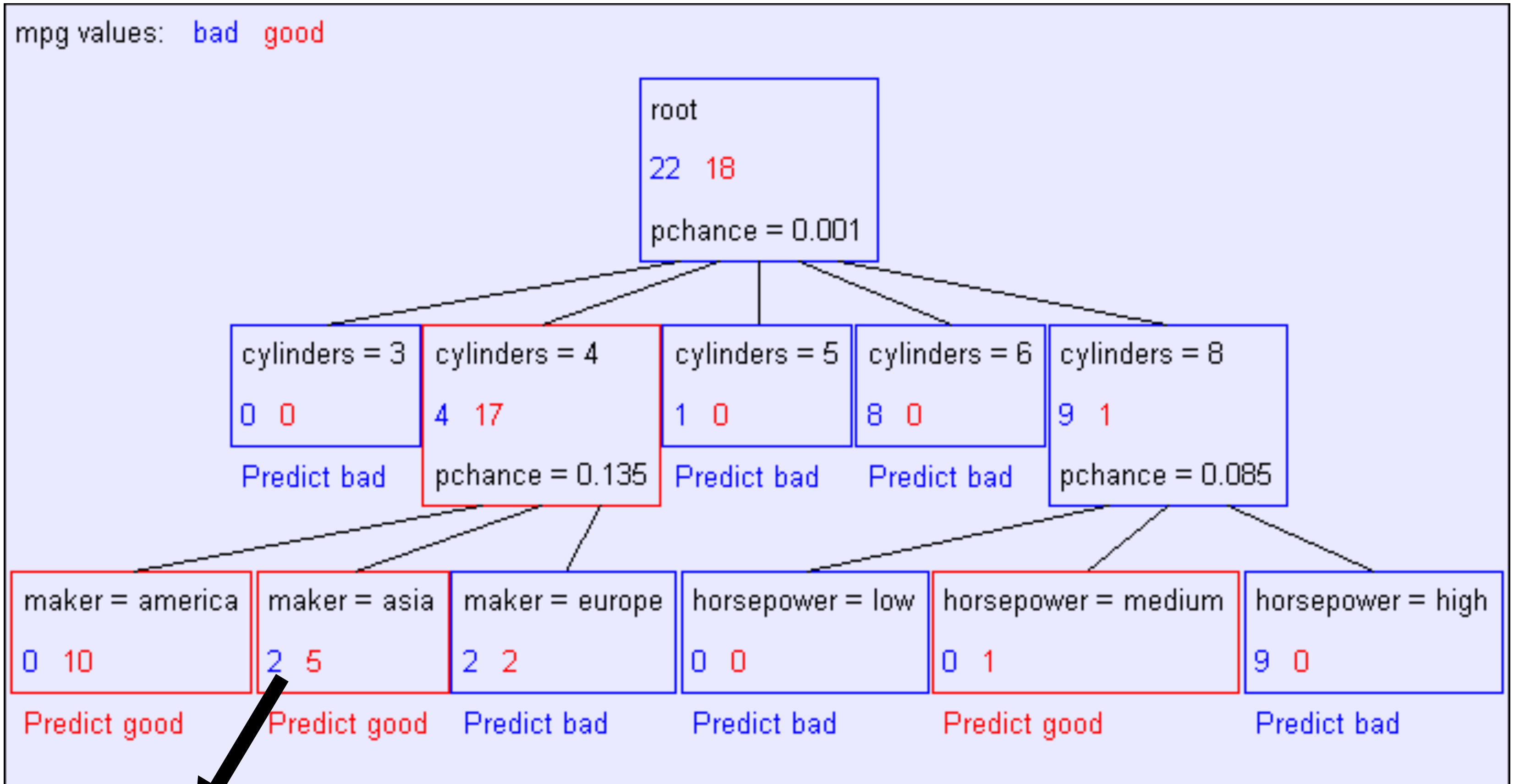
And partition it according to the value of the attribute we split on



Recursive Step



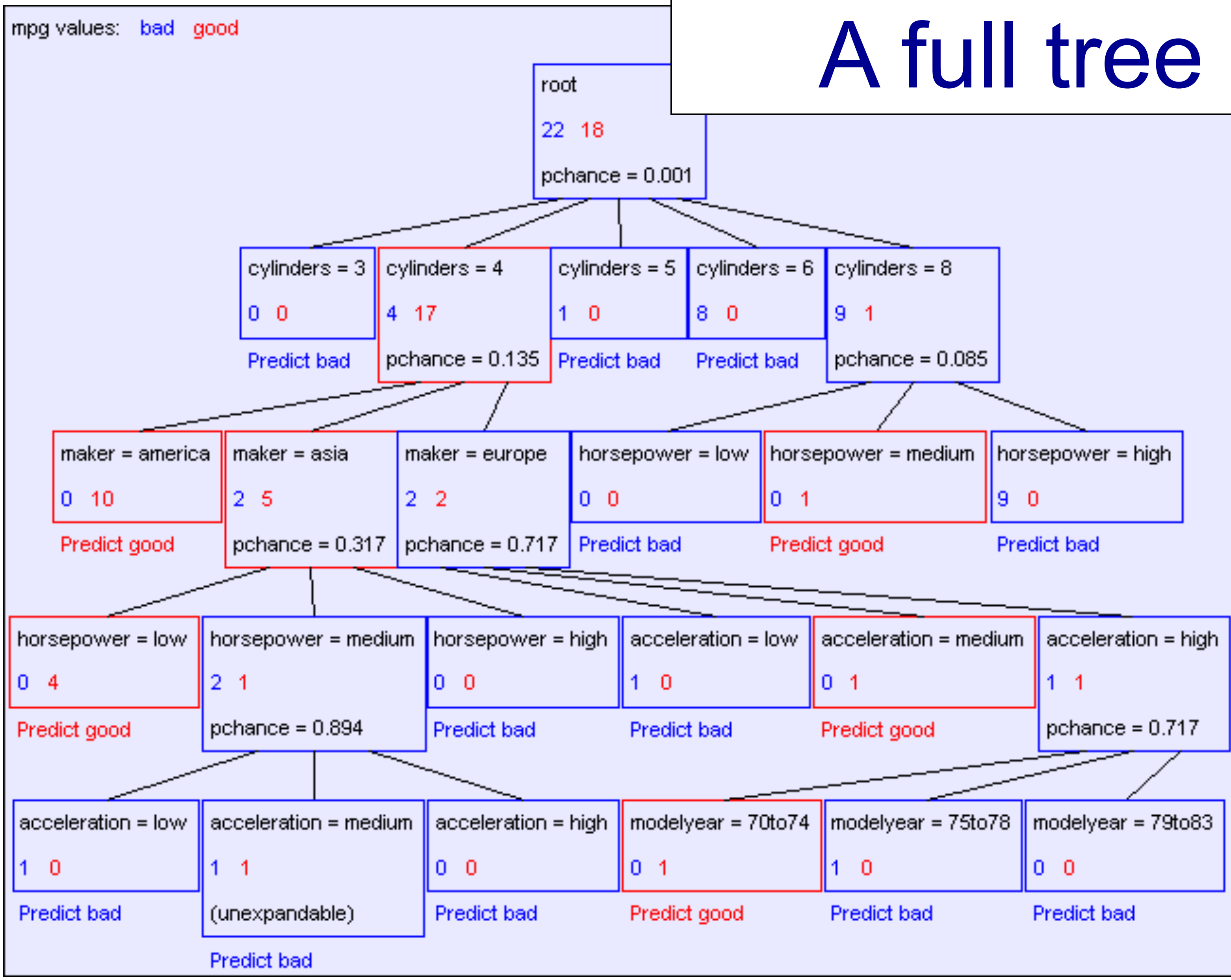
Second level of tree



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)

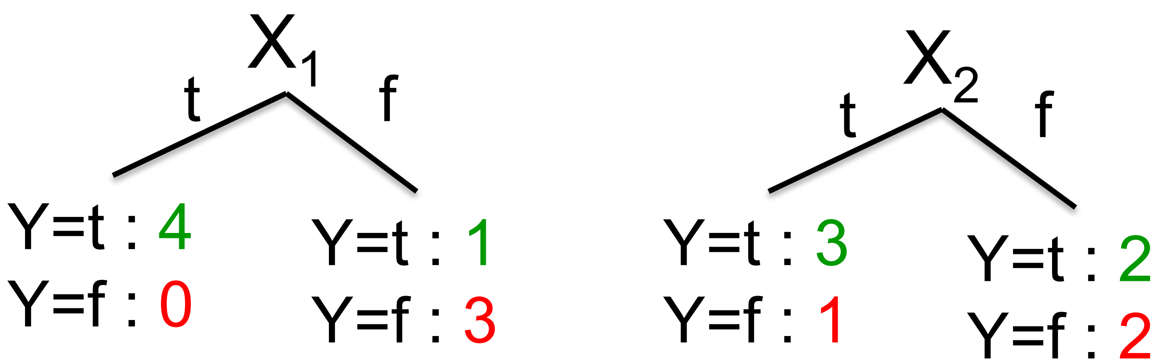
A full tree



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

Splitting: choosing a good attribute

Would we prefer to split on X_1 or X_2?



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

Idea: use counts at leaves to define probability distributions, so we can measure uncertainty!

Measuring uncertainty

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution bad
  - What about distributions in between?

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

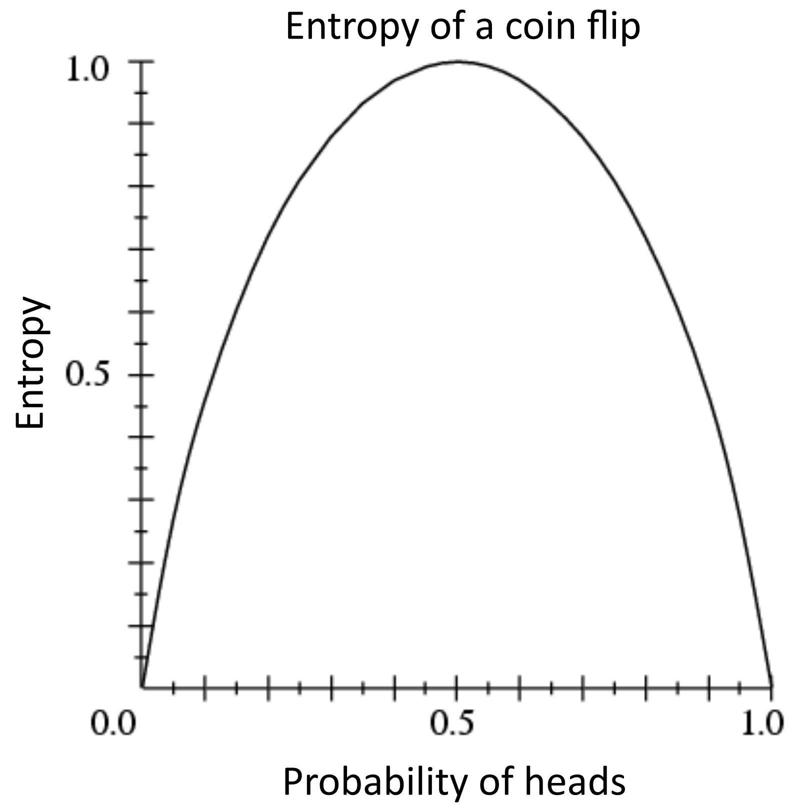
Entropy

Entropy $H(Y)$ of a random variable Y

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation:
$H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)

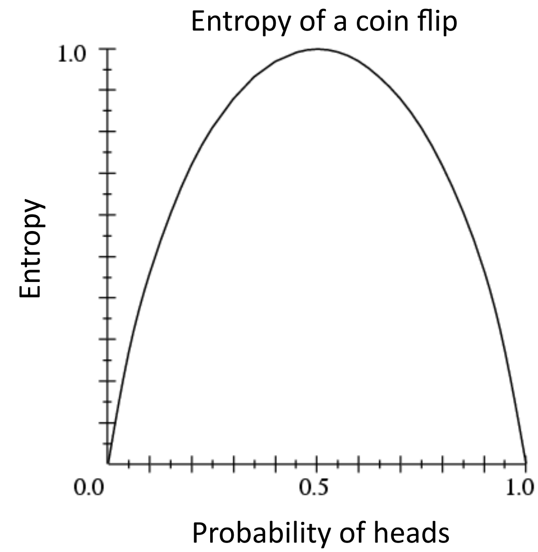


High, Low Entropy

- “High Entropy”
  - Y is from a uniform like distribution
  - Flat histogram
  - Values sampled from it are less predictable
- “Low Entropy”
  - Y is from a varied (peaks and valleys) distribution
  - Histogram has many lows and highs
  - Values sampled from it are more predictable

(Slide from Vibhav Gogate)

Entropy Example



$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

P(Y=t) = 5/6

P(Y=f) = 1/6

$H(Y) = -\ 5/6 \log_2 5/6 - 1/6 \log_2 1/6$

$= 0.65$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Conditional Entropy

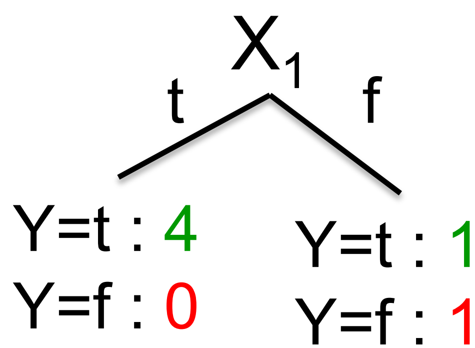
Conditional Entropy $H(Y|X)$ of a random variable Y conditioned on a random variable X

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

Example:

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$$H(Y|X_1) = -\,4/6\,(1 \log_2 1 + 0 \log_2 0)$$
$$-\,2/6\,(1/2 \log_2 1/2 + 1/2 \log_2 1/2)$$
$$= 2/6$$

Information gain

- Decrease in entropy (uncertainty) after splitting

$$IG(X) = H(Y) - H(Y \mid X)$$

In our running example:

$IG(X_1) = H(Y) – H(Y|X_1)$
$= 0.65 – 0.33$

$IG(X_1) > 0 \rightarrow$ we prefer the split!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

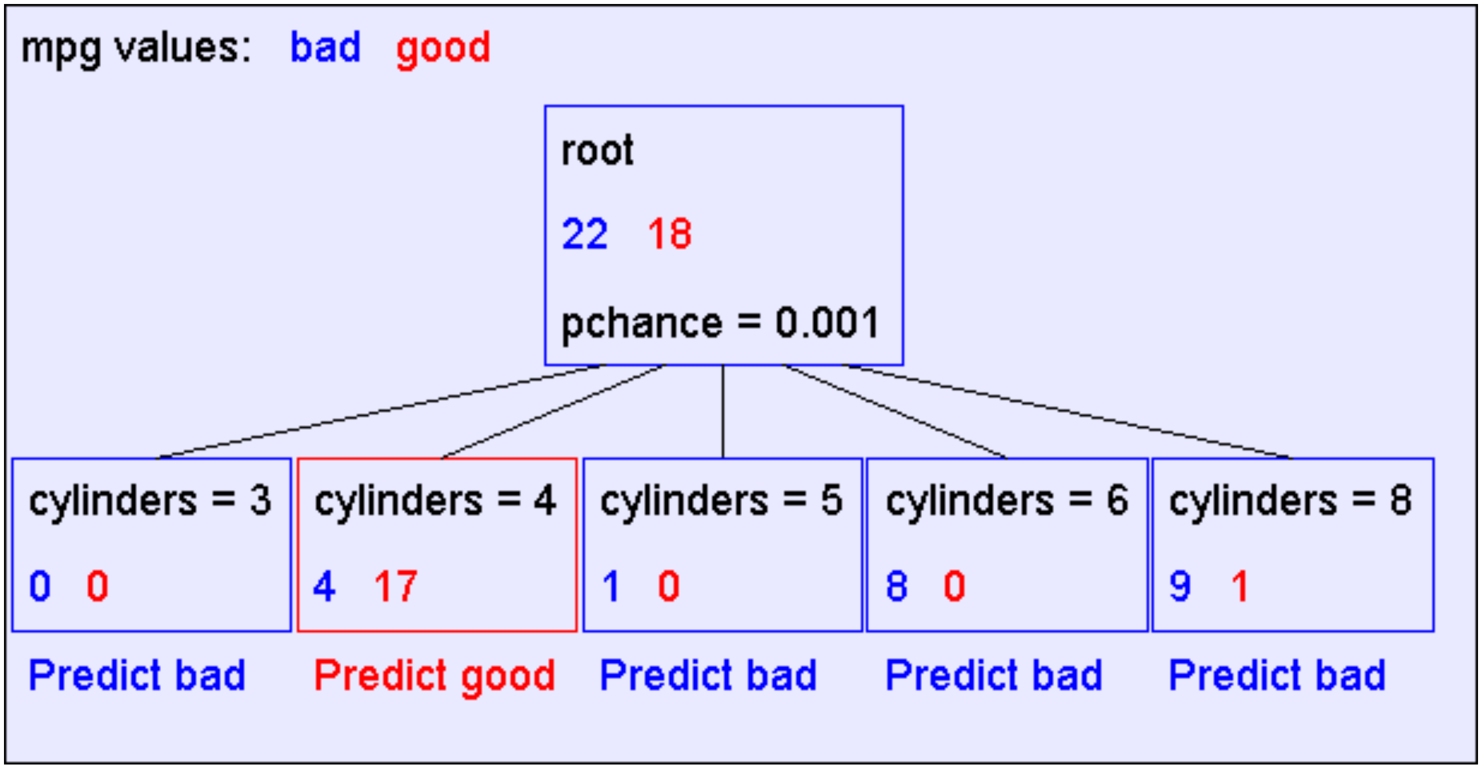
Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y \mid X_i)$$

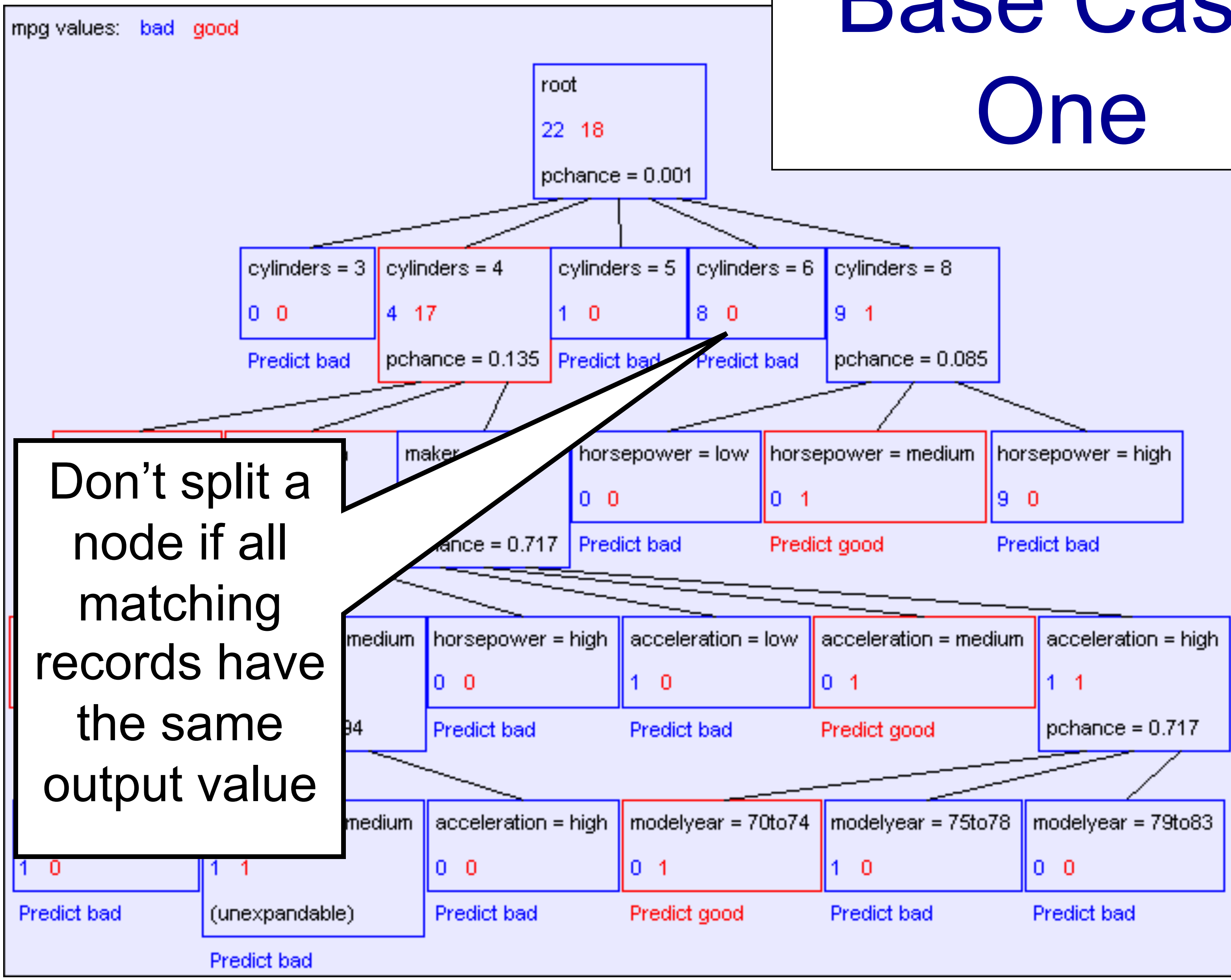
- Recurse

When to stop?



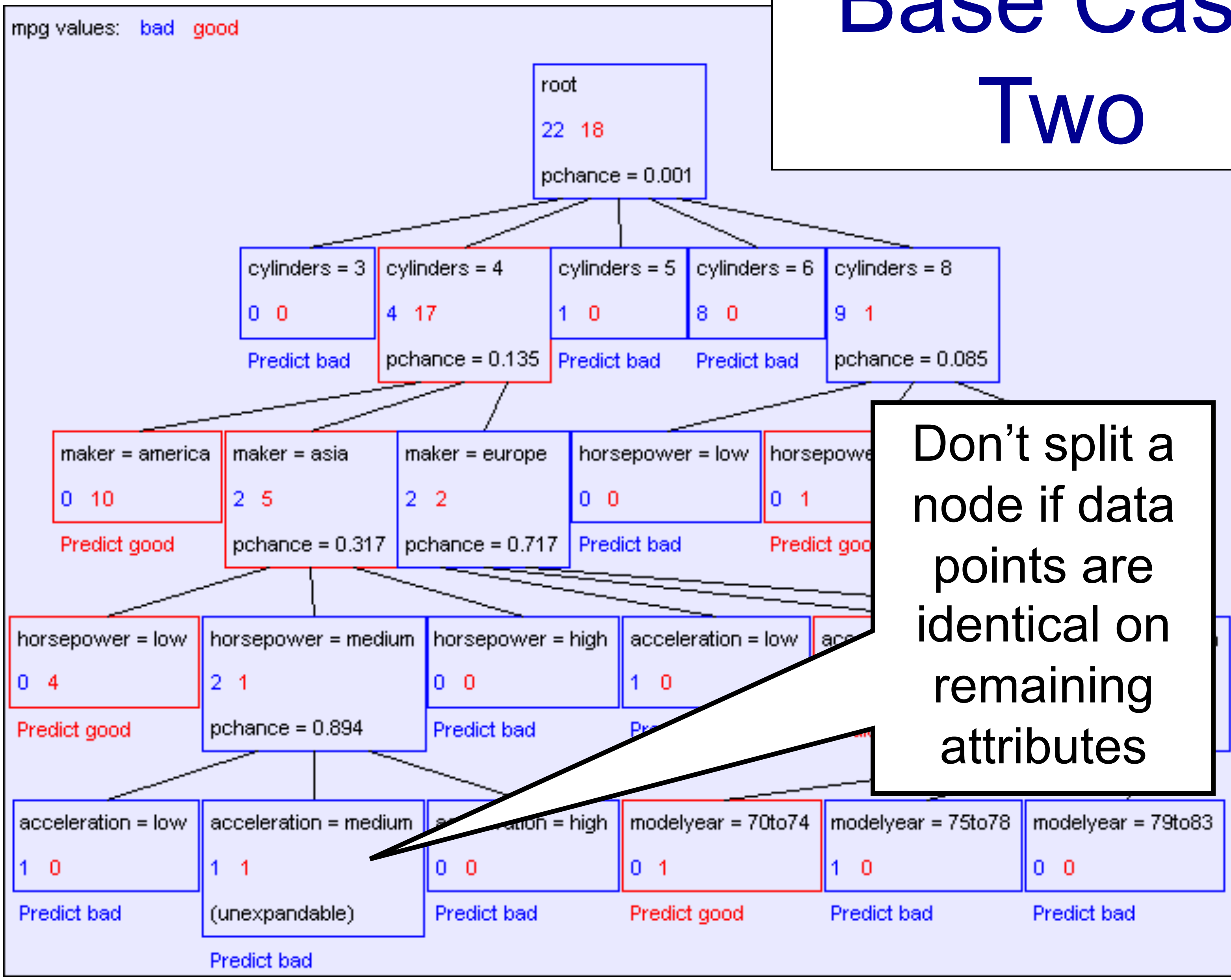
First split looks good! But, when do we stop?

Base Case One



Don't split a node if all matching records have the same output value

Base Case Two



Don't split a node if data points are identical on remaining attributes

Base Cases: An idea

- Base Case One: If all records in current data subset have the same output then don’t recurse
- Base Case Two: If all records have exactly the same set of input attributes then don’t recurse

Proposed Base Case 3:
If all attributes have small information gain then don’t recurse

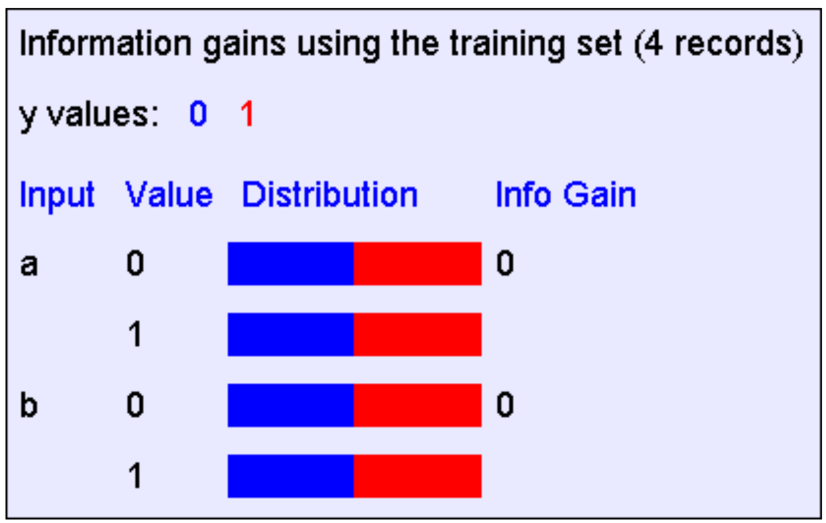
- *This is not a good idea*

The problem with proposed case 3

$y = a \text{ XOR } b$

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

The information gains:



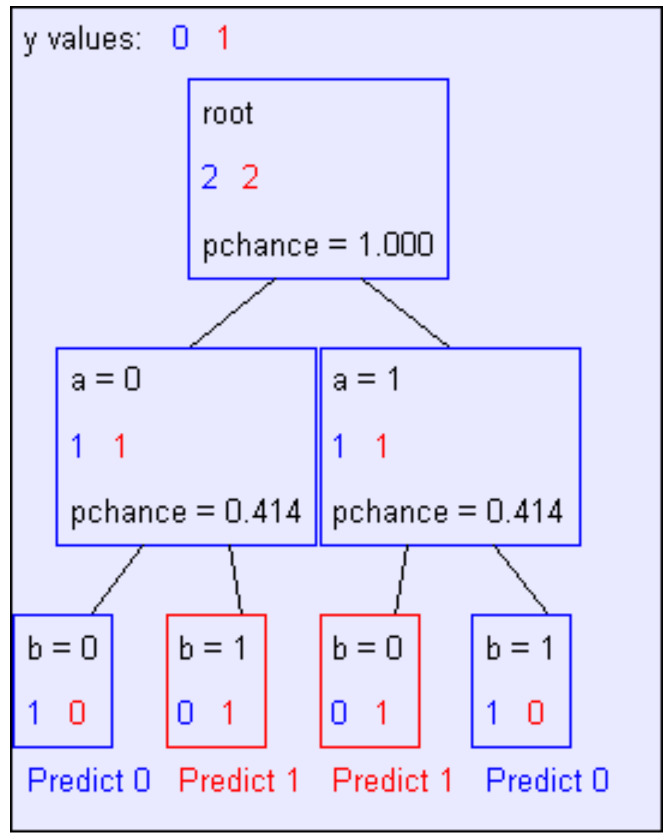
If we omit proposed case 3:

y = a XOR b

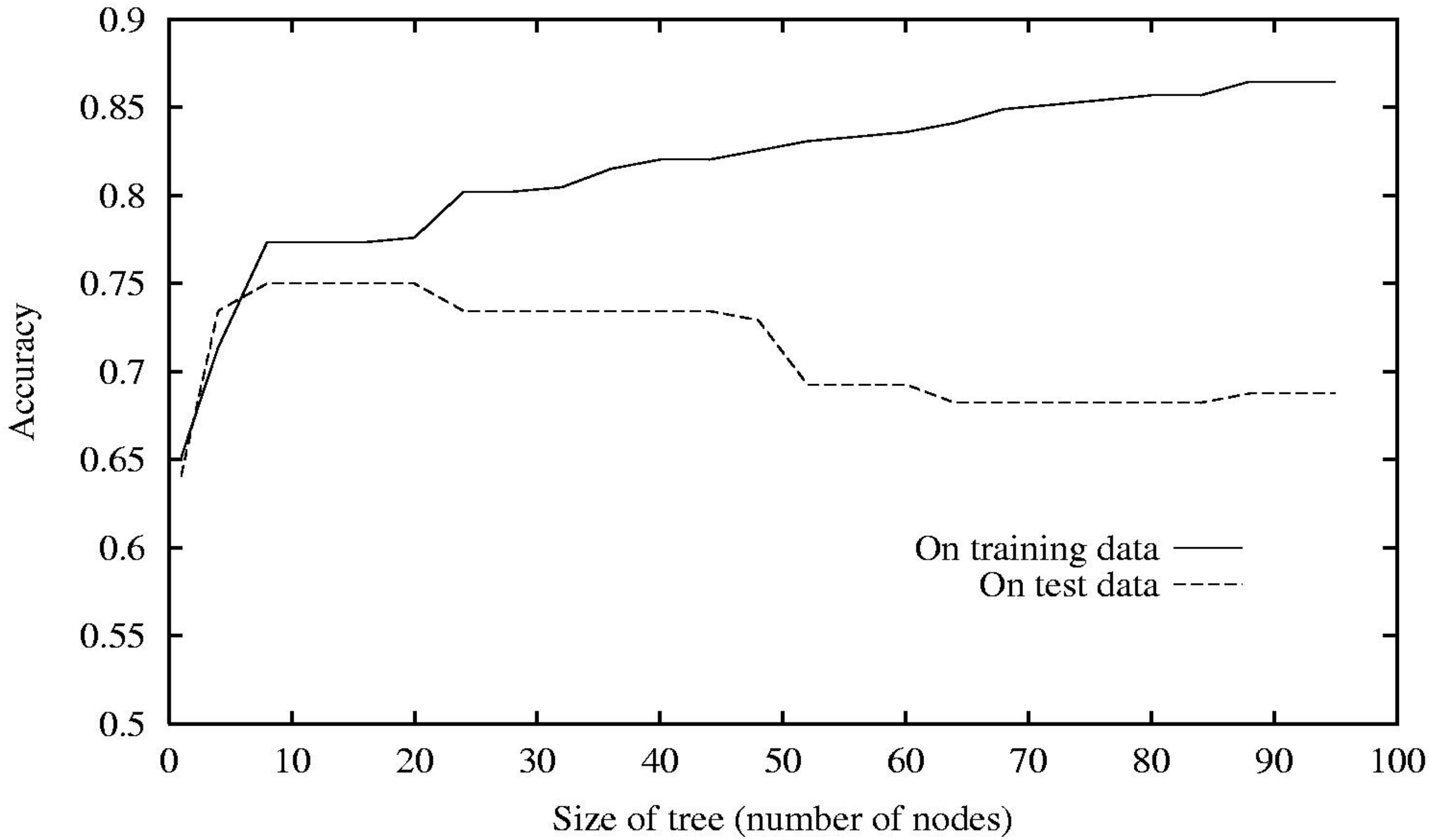
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Instead, perform **pruning** after building a tree

The resulting decision tree:



Decision trees will overfit



Decision trees will overfit

- Standard decision trees have no learning bias
  - Training set error is always zero!
    - (If there is no label noise)
  - Lots of variance
  - Must introduce some bias towards simpler trees
- Many strategies for picking simpler trees
  - Fixed depth
  - Fixed number of leaves
- Random forests

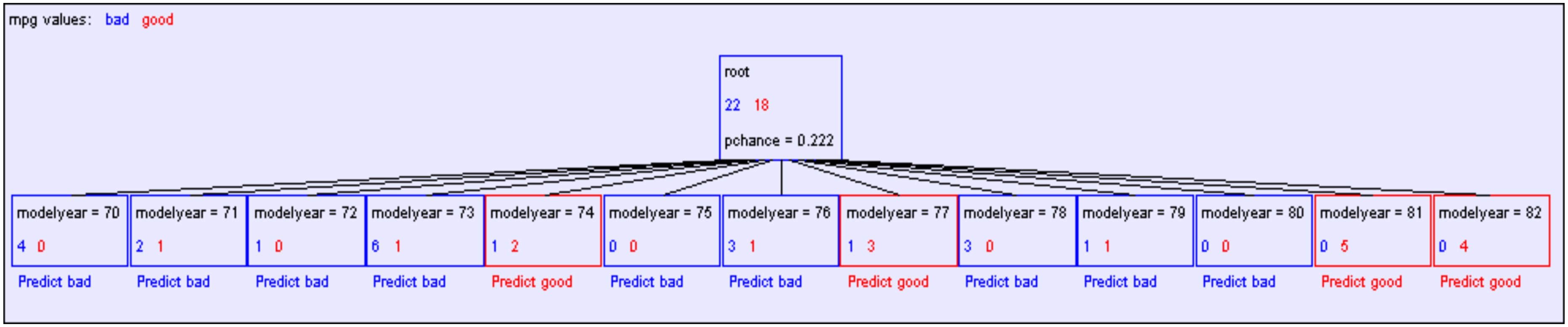
Real-Valued inputs

What should we do if some of the inputs are real-valued?

Infinite number of possible split values!!!

mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
good	4	120	79	2625	18.6	82	america
bad	8	455	225	4425	10	70	america
good	4	107	86	2464	15.5	76	europe
bad	5	131	103	2830	15.9	78	europe

“One branch for each numeric value” idea:

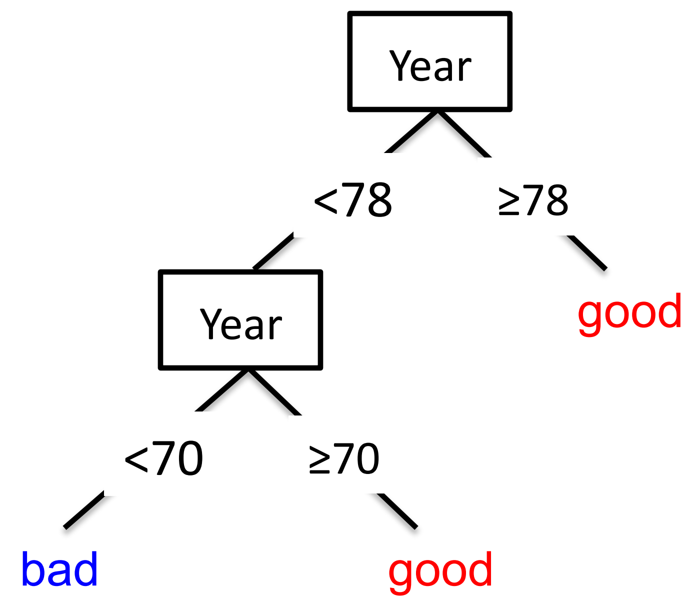


Hopeless: hypothesis with such a high branching factor will shatter *any* dataset and overfit

Threshold splits

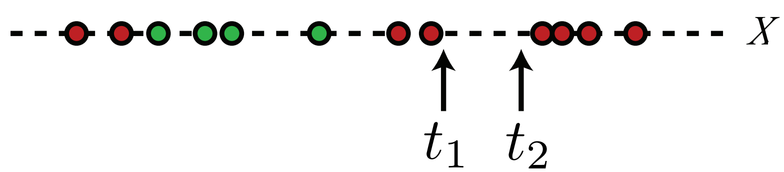
- Binary tree: split on attribute X at value t
  - One branch: X < t
  - Other branch: X ≥ t

- Requires small change
  - Allow repeated splits on same variable **along a path**

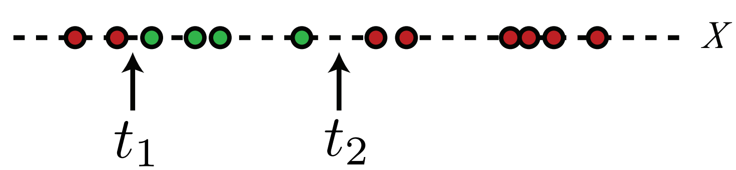


The set of possible thresholds

- Binary tree, split on attribute X
  - One branch: X < t
  - Other branch: X ≥ t
- Search through possible values of *t*
  - Seems hard!!!
- But only a finite number of *t*'s are important:



  - Sort data according to X into $\{x_1,\ldots,x_m\}$
  - Consider split points of the form $x_i + (x_{i+1} - x_i)/2$
  - Morever, only splits between examples of different classes matter!

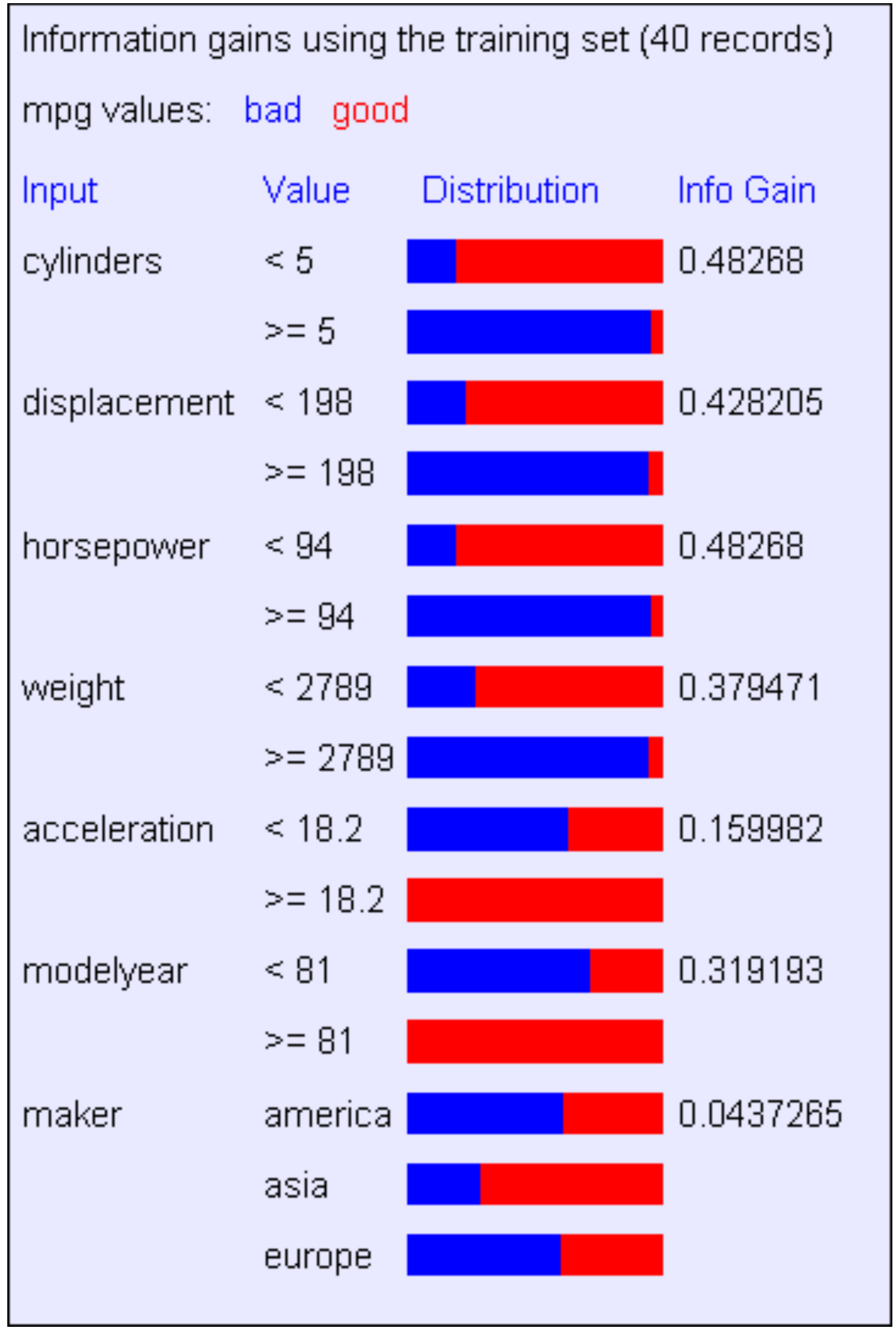


(Figures from Stuart Russell)

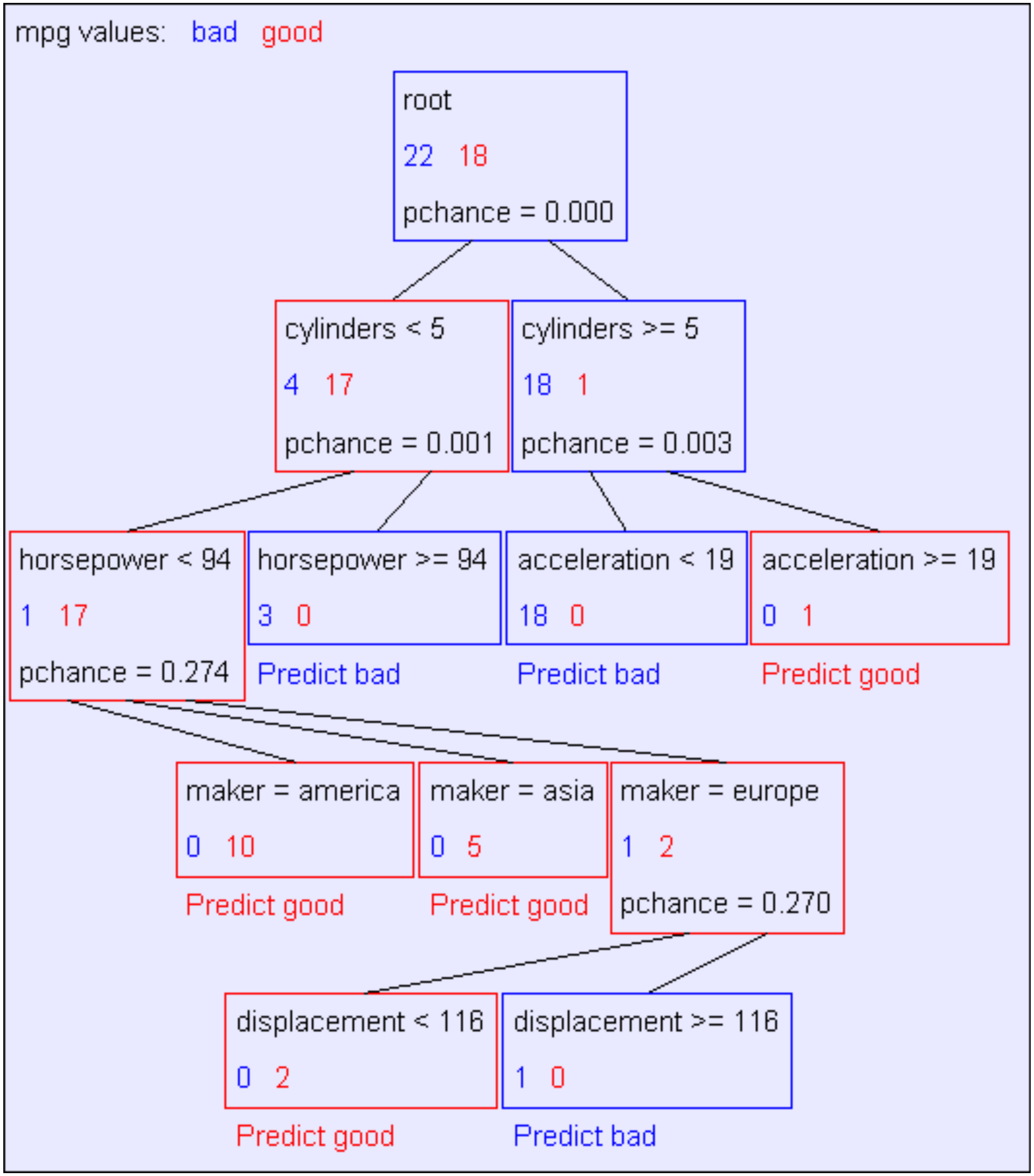
Picking the best threshold

- Suppose *X* is real valued with threshold *t*
- Want **IG(Y | X:t)**, the information gain for Y when testing if *X* is greater than or less than *t*
- Define:
  - $H(Y|X{:}t) = p(X < t)\, H(Y|X < t) + p(X >= t)\, H(Y|X >= t)$
  - $IG(Y|X{:}t) = H(Y) - H(Y|X{:}t)$
  - $IG^*(Y|X) = \max_t IG(Y|X{:}t)$
- Use: $IG^*(Y|X)$ for continuous variables

Example with MPG



Example tree for our continuous dataset



What you need to know about decision trees

- Decision trees are one of the most popular ML tools
  - Easy to understand, implement, and use
  - Computationally cheap (to solve heuristically)
- Information gain to select attributes (ID3, C4.5,…)
- Presented for classification, can be used for regression and density estimation too
- Decision trees will overfit!!!
  - Must use tricks to find “simple trees”, e.g.,
    - Fixed depth/Early stopping
    - Pruning
  - Or, use ensembles of different trees (random forests)

Ensemble learning

Slides adapted from Navneet Goyal, Tan, Steinbach, Kumar, Vibhav Gogate

Ensemble methods

Machine learning competition with a $1 million prize

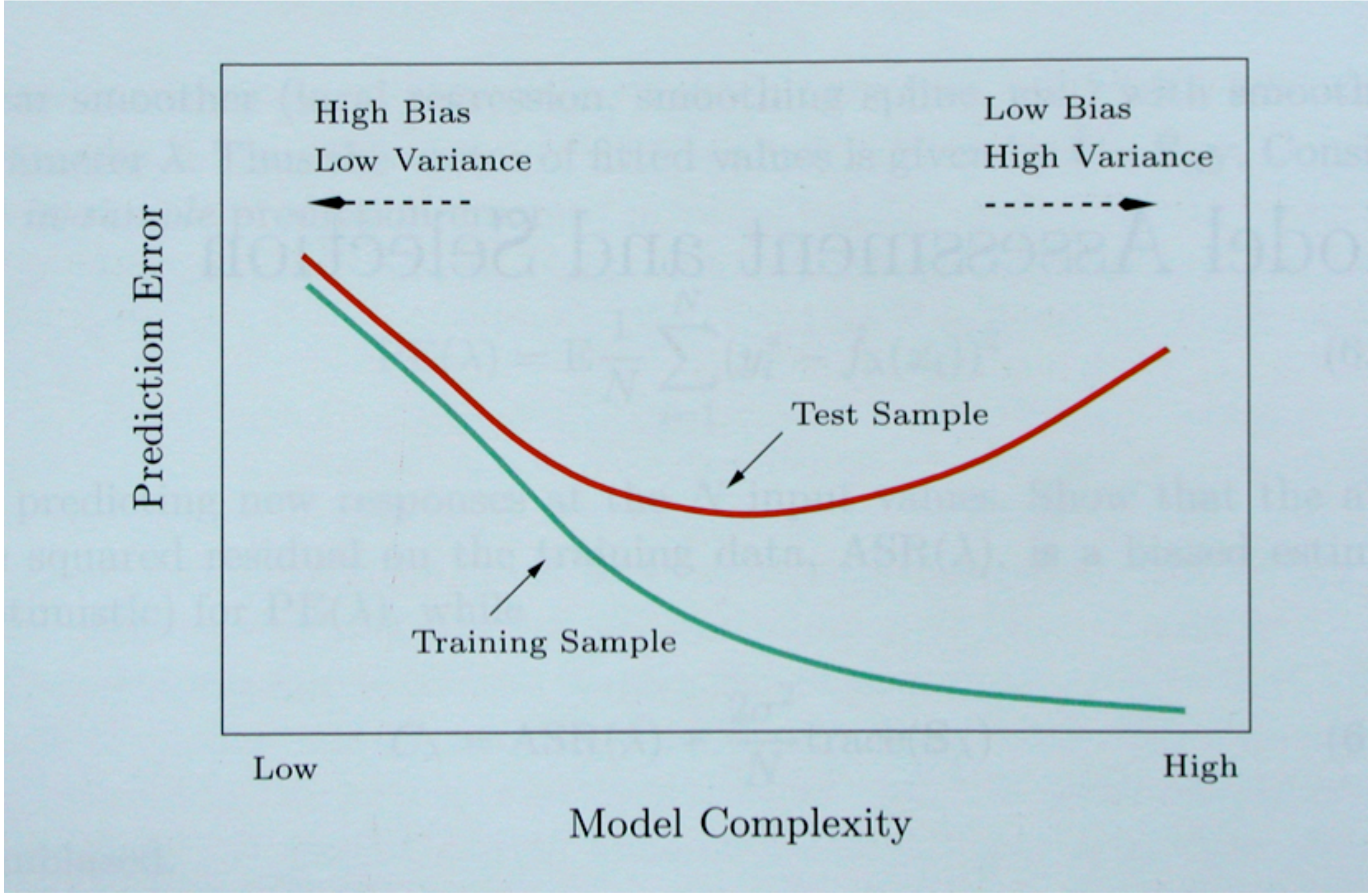
Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
Grand Prize - RMSE <= 0.8563				
3	Grand Prize Team	0.8571	9.91	2009-07-24 13:07:49
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-25 20:05:52
5	Vandelay Industries !	0.8579	9.83	2009-07-26 02:49:53
6	PragmaticTheory	0.8582	9.80	2009-07-12 15:09:53
7	BellKor in BigChaos	0.8590	9.71	2009-07-26 12:57:25
8	Dace_	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-26 18:02:08
10	BellKor	0.8612	9.48	2009-07-26 17:19:11
11	BigChaos	0.8613	9.47	2009-06-23 23:06:52
12	Feeds2	0.8613	9.47	2009-07-24 20:06:46
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8633	9.26	2009-07-21 02:04:40
14	Gravity	0.8634	9.25	2009-07-26 15:58:34
15	Ces	0.8642	9.17	2009-07-25 17:42:38
16	Invisible Ideas	0.8644	9.14	2009-07-20 03:26:12
17	Just a guy in a garage	0.8650	9.08	2009-07-22 14:10:42
18	Craig Carmichael	0.8656	9.02	2009-07-25 16:00:54
19	J Dennis Su	0.8658	9.00	2009-03-11 09:41:54
20	acmehill	0.8659	8.99	2009-04-16 06:29:35
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
Cinematch score on quiz subset - RMSE = 0.9514				

	users →					
movies ↓	1		?	3	5	?
	?	1				2
		4		4	5	?

Bias/Variance Tradeoff



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Reduce Variance Without Increasing Bias

- Averaging reduces variance:

$$Var(\overline{X}) = \frac{Var(X)}{N}$$

(when predictions are **independent**)

Average models to reduce model variance

One problem:

only one training set

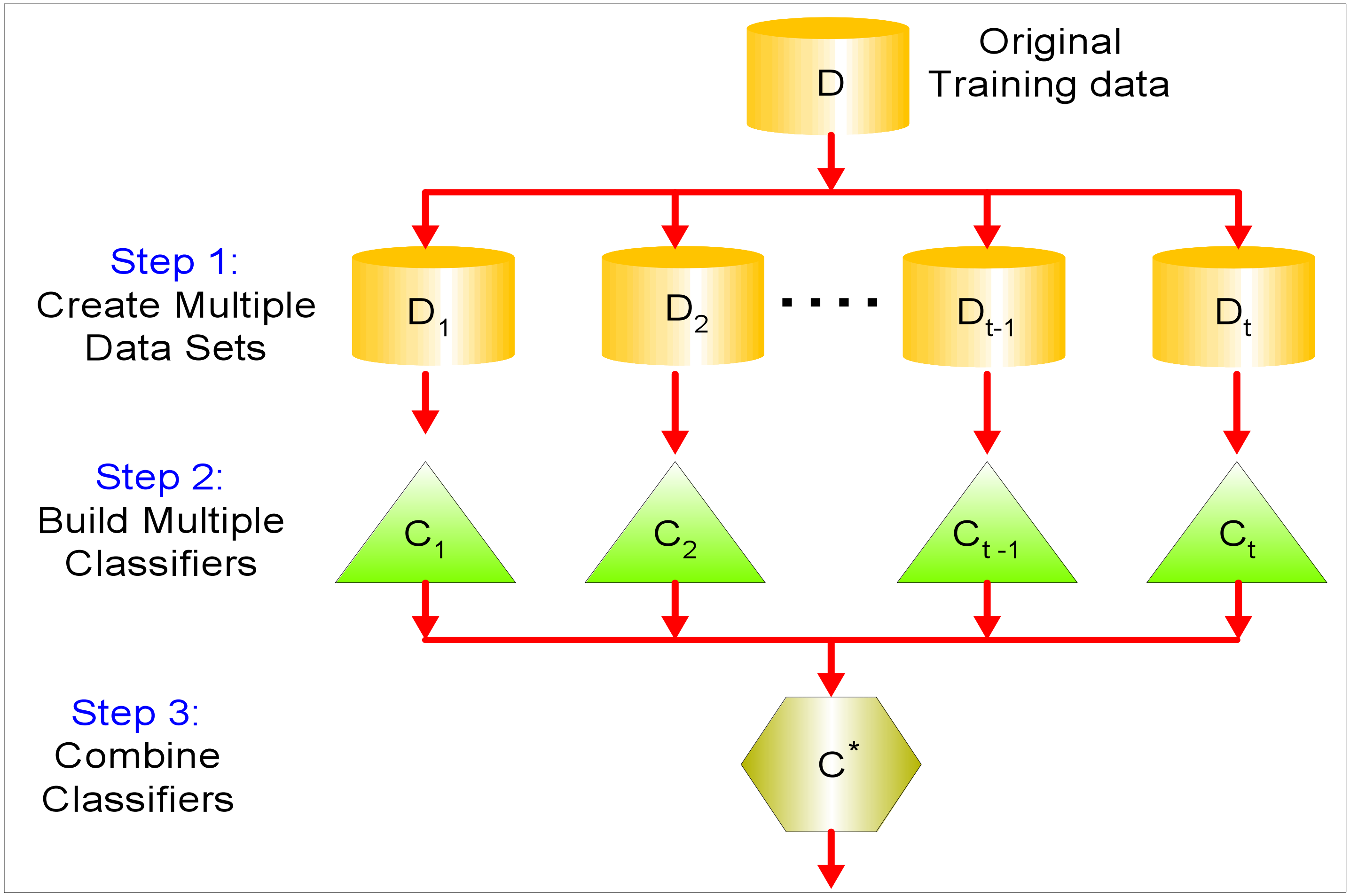
where do multiple models come from?

Bagging: Bootstrap Aggregation

- Leo Breiman (1994)
- Take repeated bootstrap samples from training set *D*
- *Bootstrap sampling*: Given set *D* containing *N* training examples, create *D*' by drawing *N* examples at random with replacement from *D*.

- Bagging:
  - Create k bootstrap samples D_1 … D_k.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote / average.

General Idea



Example of Bagging

- Sampling with replacement

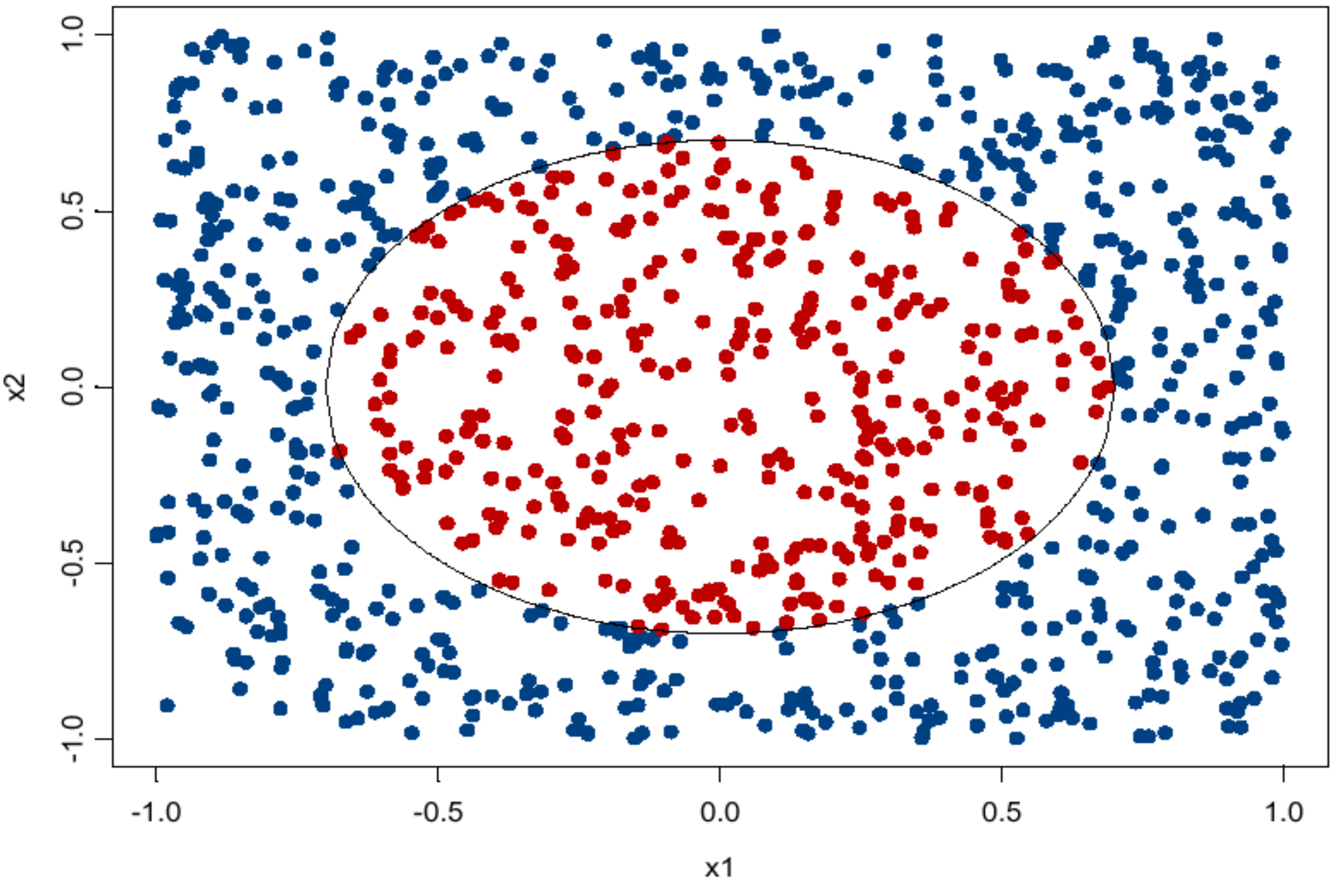
Training Data

Data ID

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

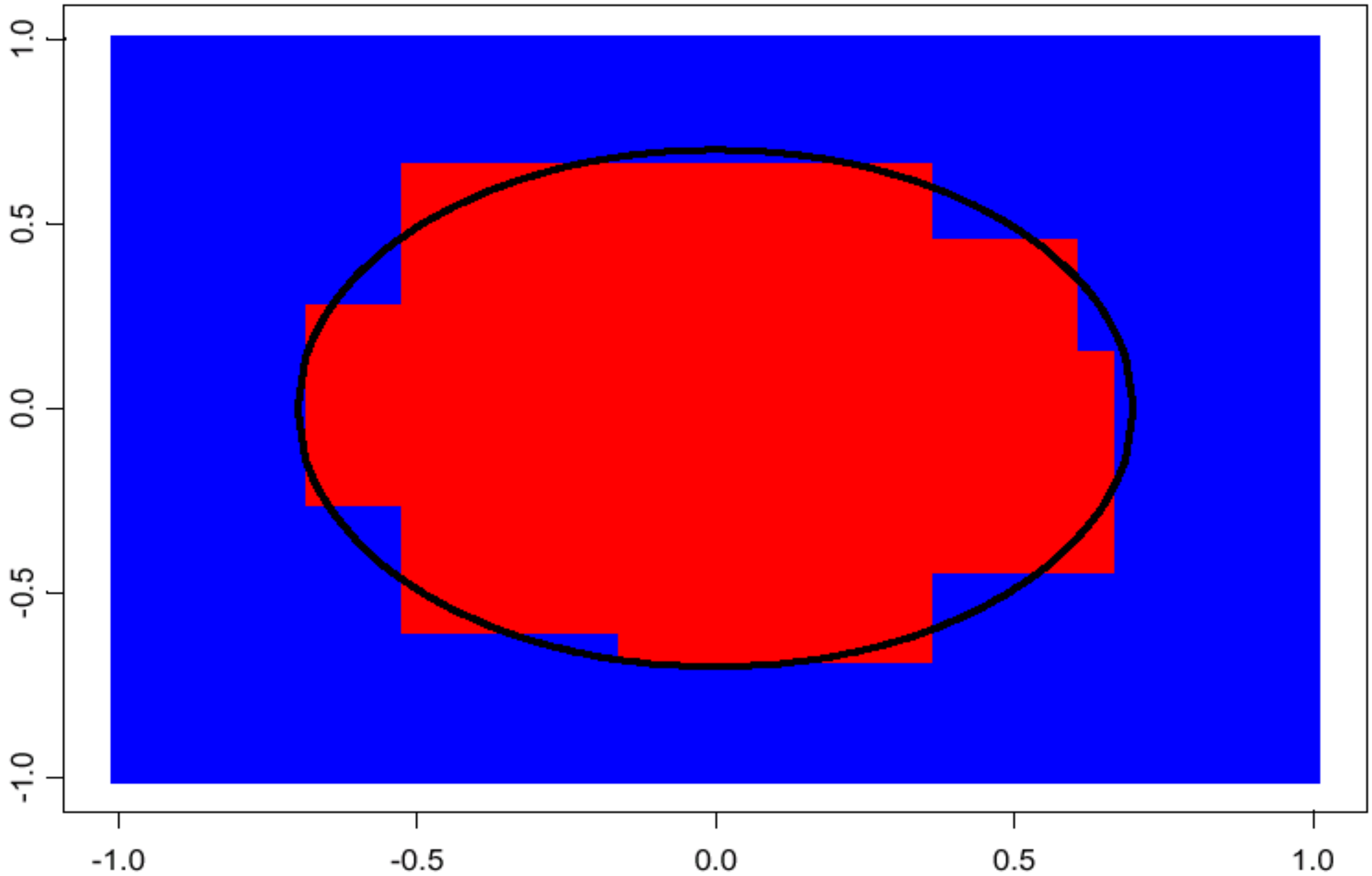
- Build classifier on each bootstrap sample
- Each data point has probability $(1 – 1/n)^n$ of being selected as test data
- Training data = $1- (1 – 1/n)^n$ of the original data

Bagging Example

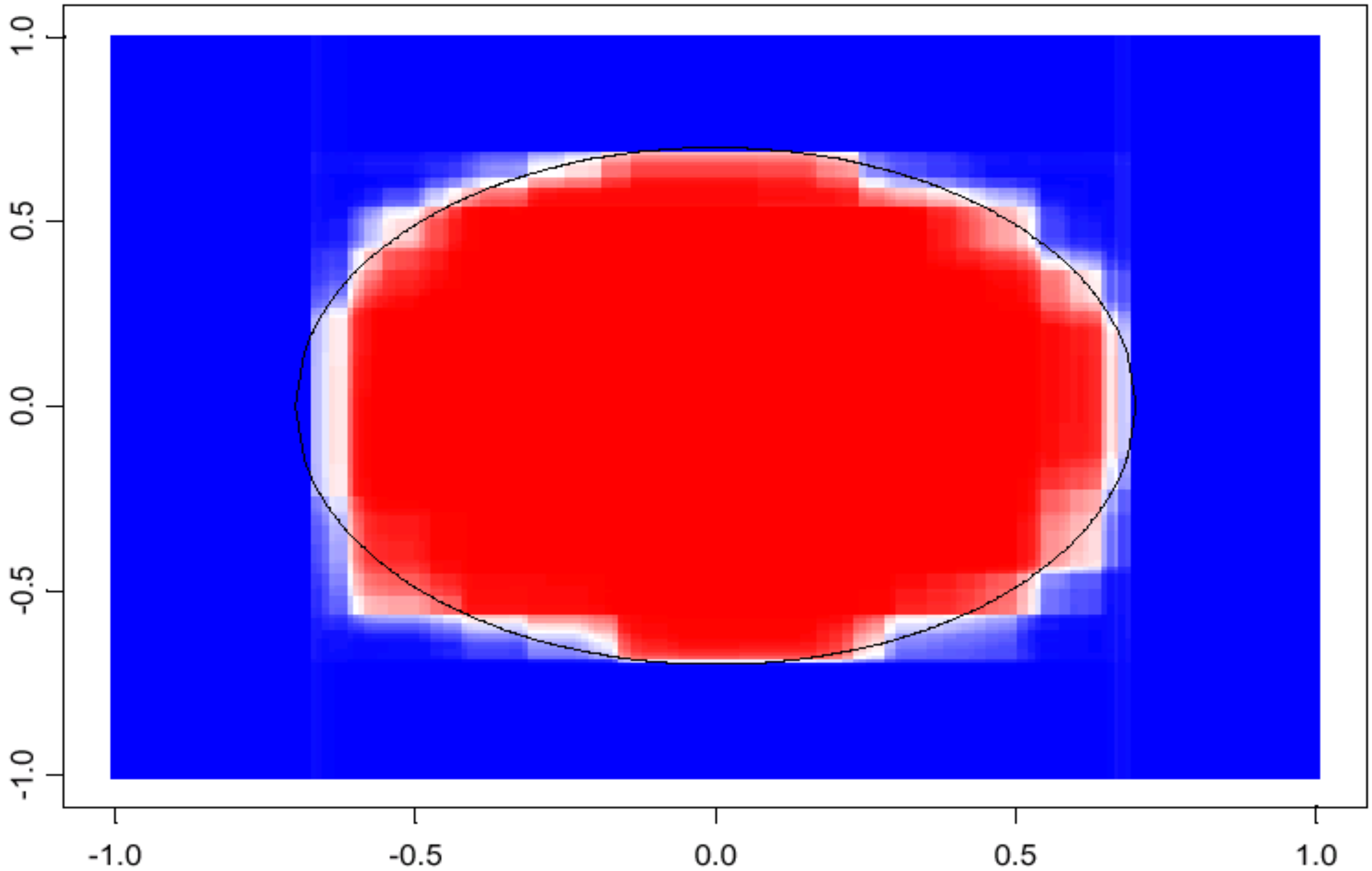


decision tree learning algorithm; very similar to ID3

CART decision boundary



100 bagged trees



shades of blue/red indicate strength of vote for particular classification

Random Forests

- Ensemble method specifically designed for decision tree classifiers

- Introduce two sources of randomness: "Bagging" and "Random input vectors"
  - Bagging method: each tree is grown using a bootstrap sample of training data
  - Random vector method: At each node, best split is chosen from a random sample of *m* attributes instead of all attributes

Random Forests

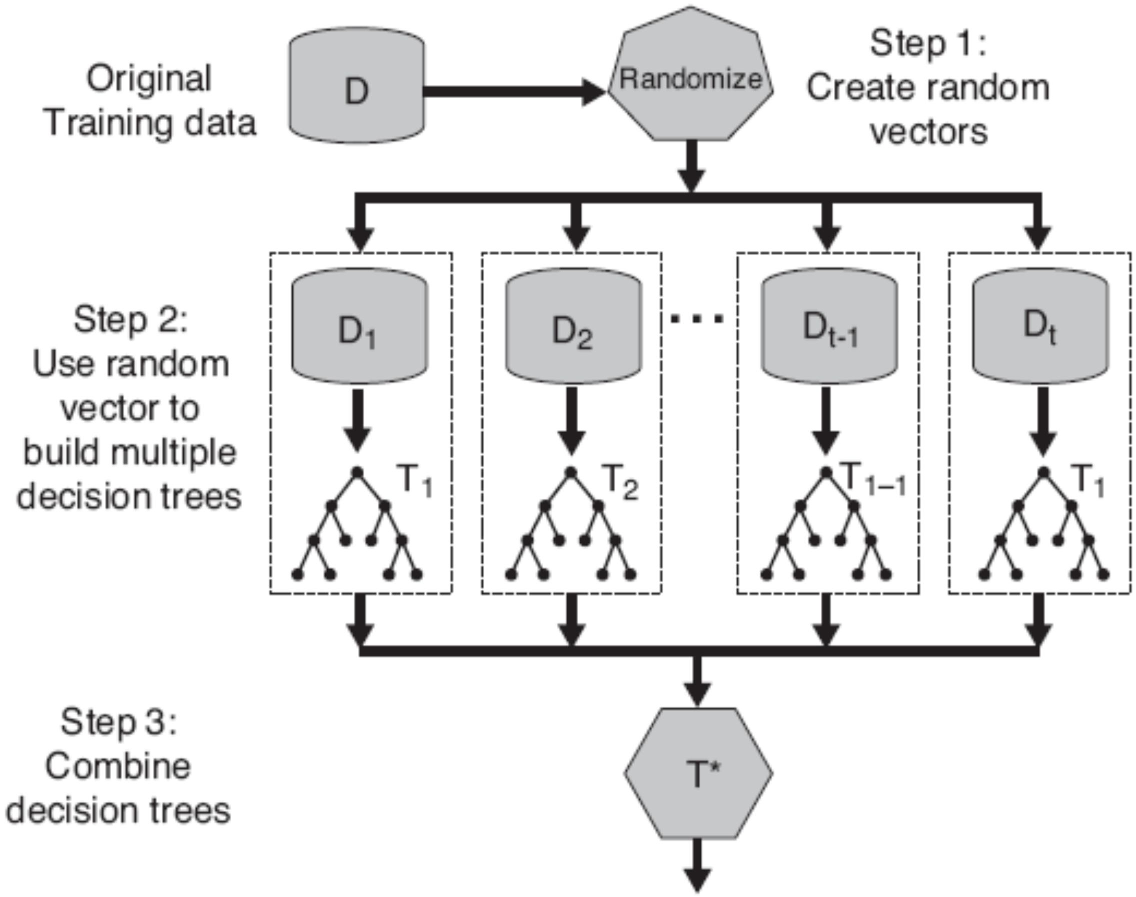


Figure 5.40. Random forests.

Random Forests Algorithm

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.