CS 189/289

Today's lecture

1. Decision Trees
2. Ensemble approaches

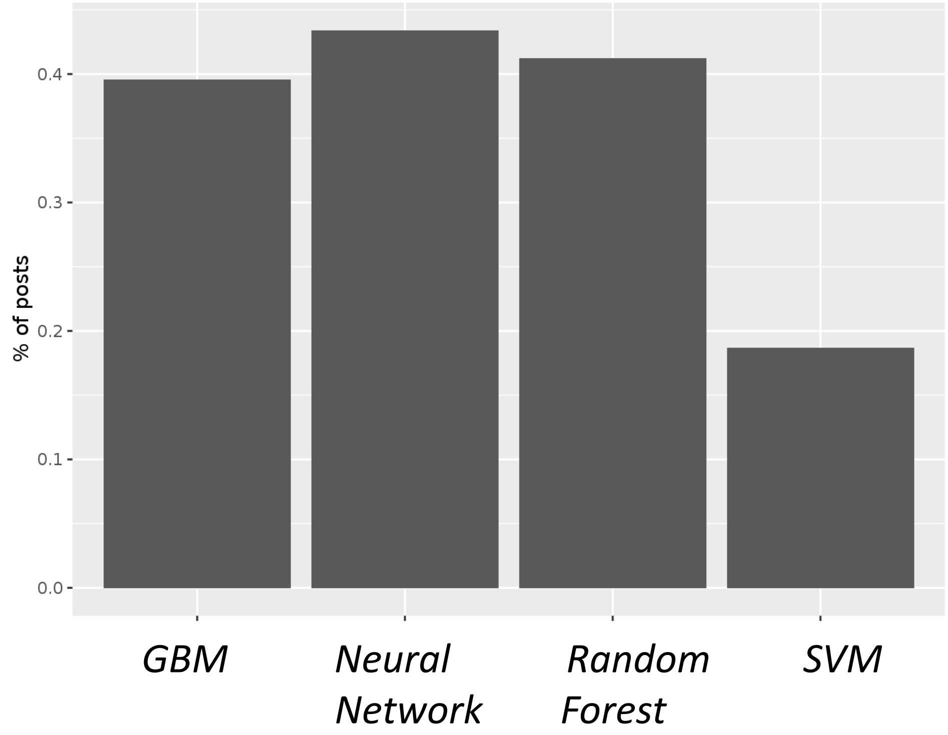
Slides based on those from David Sontag, in turn based on slides from Luke Zettlemoyer, Carlos Guestrin & Andrew Moore, and others as noted.

Classes of supervised models so far

1. Linear regression (regression)
2. Logistic regression (classification)
3. Class conditional gaussians (Gaussian Discriminant Analysis)
4. Neural networks (regression & classification)
5. Today: the “game of 20 questions” approach to modeling

Decision trees underlie state-of-the-art methods

- Competitive model classes for classification and regression rely on decision trees (DT) (*random forest* and *gradient-boosted trees*).
- Kaggle competitions 2016:
- DT-based models work well with default hyper-parameters.
- Are highly interpretable.
- NNs may generally perform better.



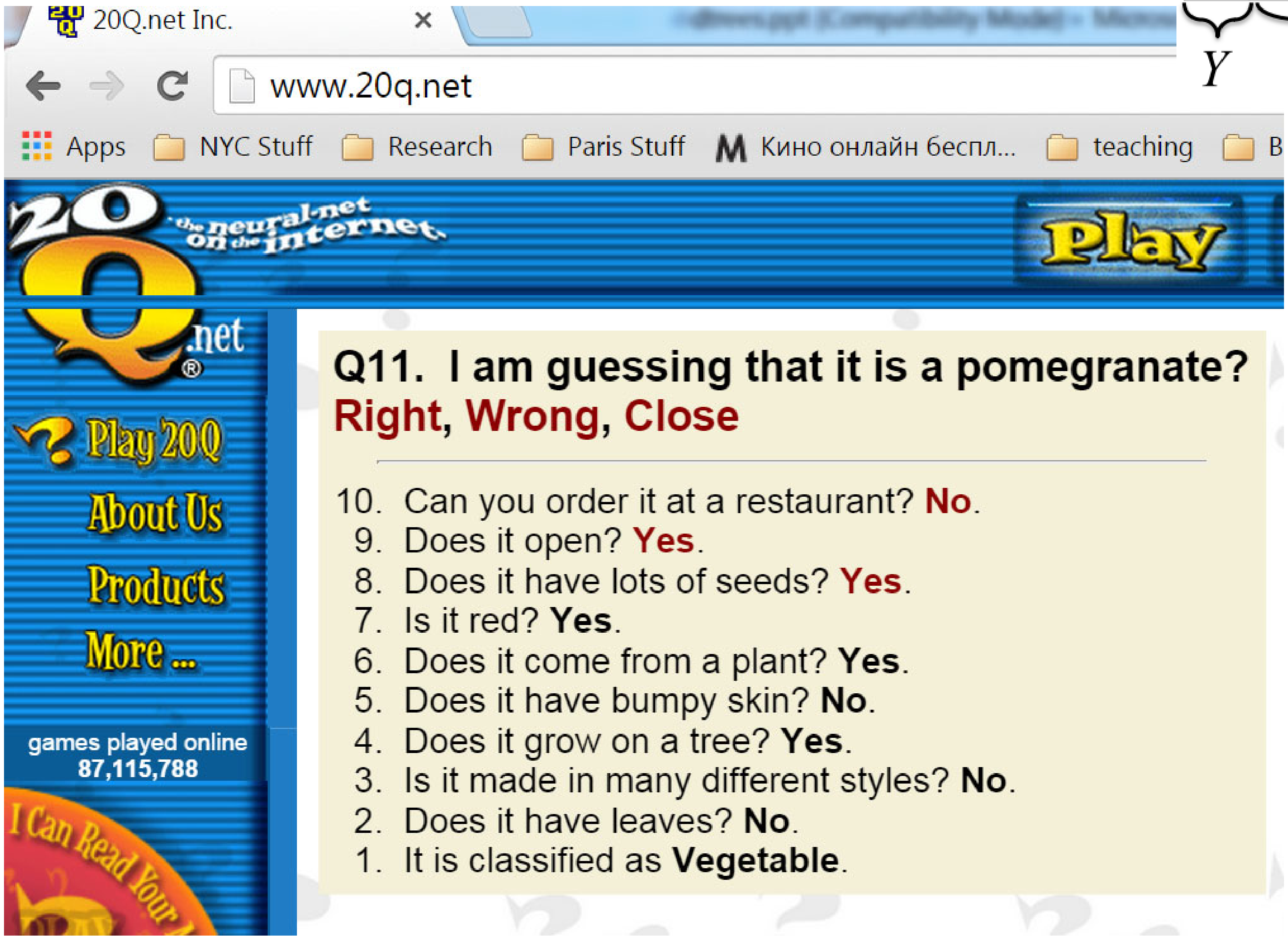
GBM= "Gradient Boosted Machine" (*ensemble* of Decision Trees)

https://www.kaggle.com/msjgriffiths/r-what-algorithms-are-most-successful-on-kaggle/report

The Game of 20 Questions

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

Y X



- How might you use something like this as a classifier?
- How would a sample get classified?
- How would you train this model?

Decision Tree running example: classify fuel efficiency

- 40 data points
- Goal: predict MPG
- Need to find: $f : X \rightarrow Y$
- Discrete data (for now)

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

$\underbrace{\text{mpg}}_{Y}$ $\underbrace{\text{cylinders} \ldots \text{maker}}_{X}$

From the UCI repository (thanks to Ross Quinlan)

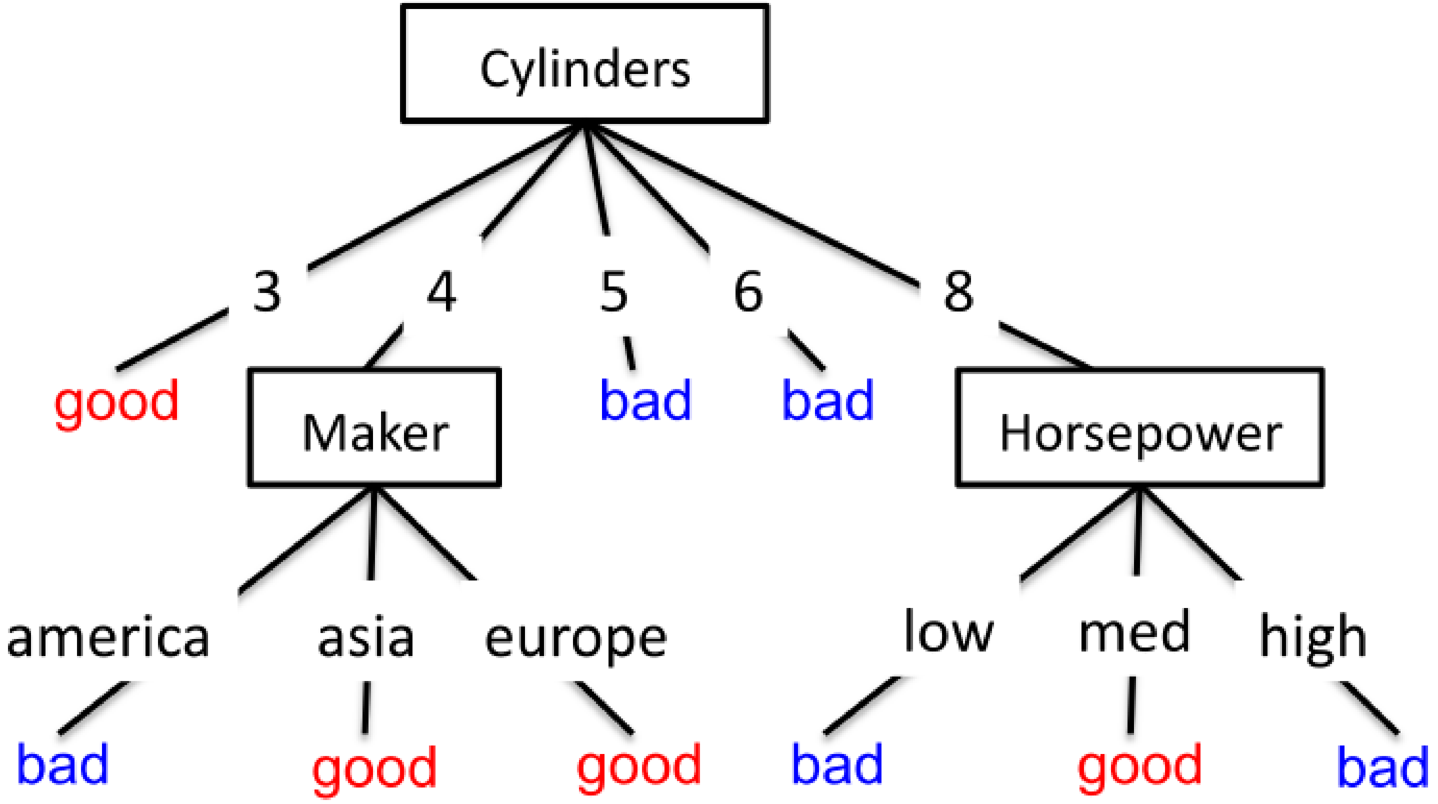
Hypotheses: decision trees $f : X \rightarrow Y$

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

Y X

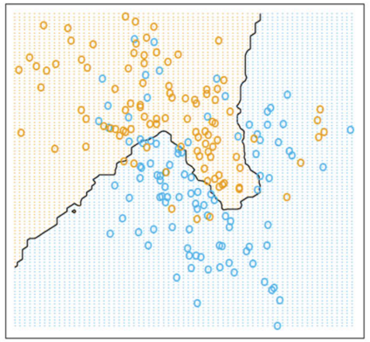
Y=fuel efficiency (mpg) $\in \{good, bad\}$

- Each internal node tests an attribute x_i
- Each branch "picks off" an 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



DT classification boundaries

What would this kind of diagram look like for a decision tree?



(recall 15-NN classifier)

- The partitions would be …axis-aligned

  i.e., no diagonals boundaries

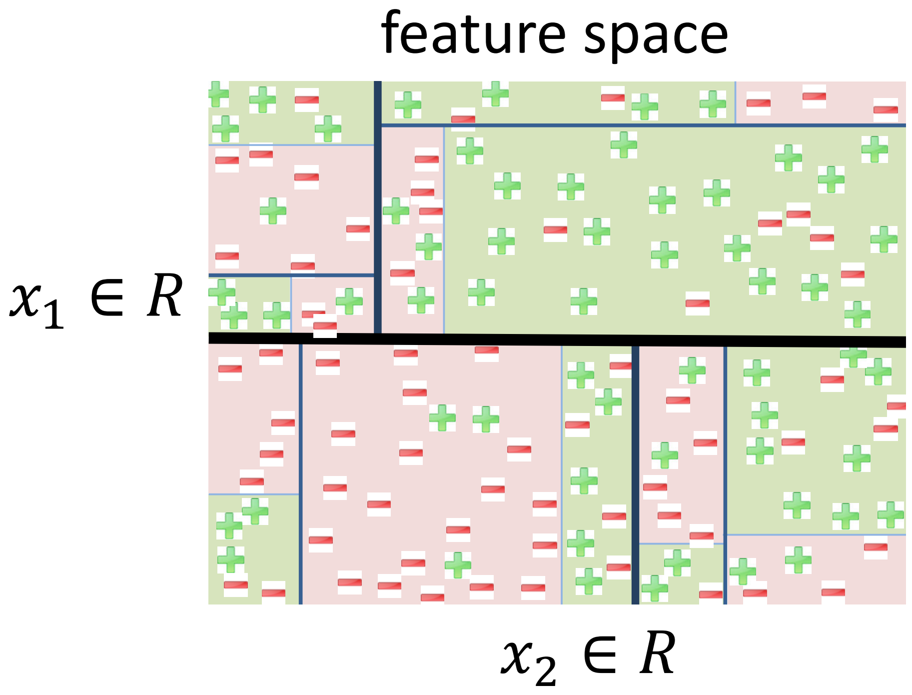
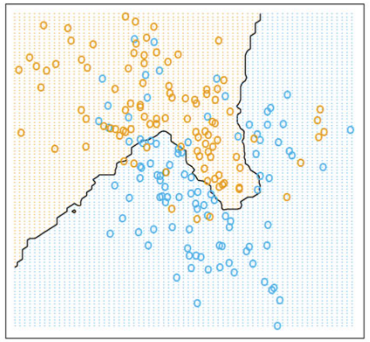


figure credit: Yisong Yue

DT classification boundaries

What would this kind of diagram look like for a decision tree?



(recall 15-NN classifier)

- The partitions would be …axis-aligned

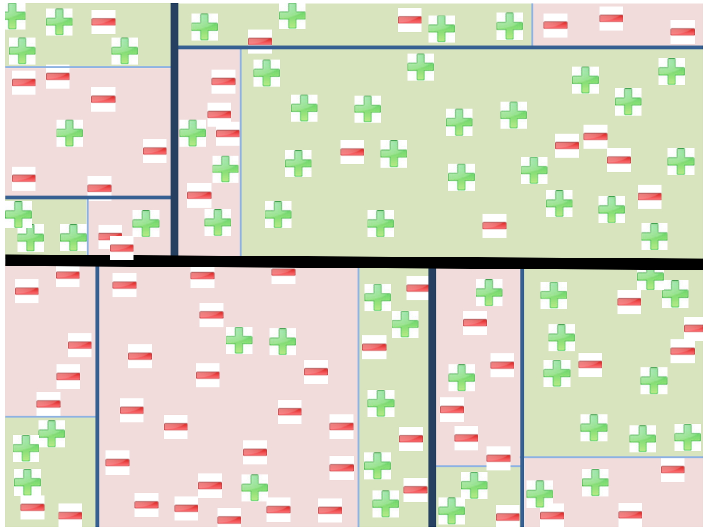
  i.e., no diagonals boundaries

- It would be a “piecewise static” function class

  i.e., each partition has a static prediction.

feature space

$x_1 \in R$



$x_2 \in R$

fitgure credit: Yisong Yue

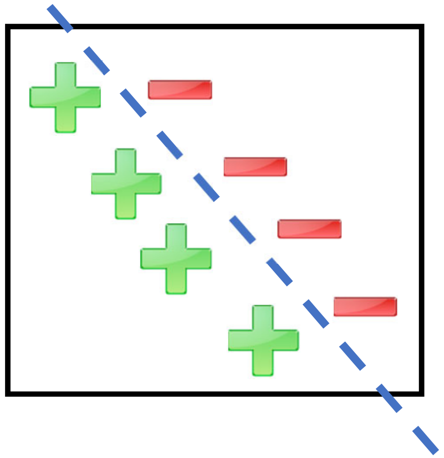
Decision Trees vs Linear Models

Decision Trees are AXIS-ALIGNED!

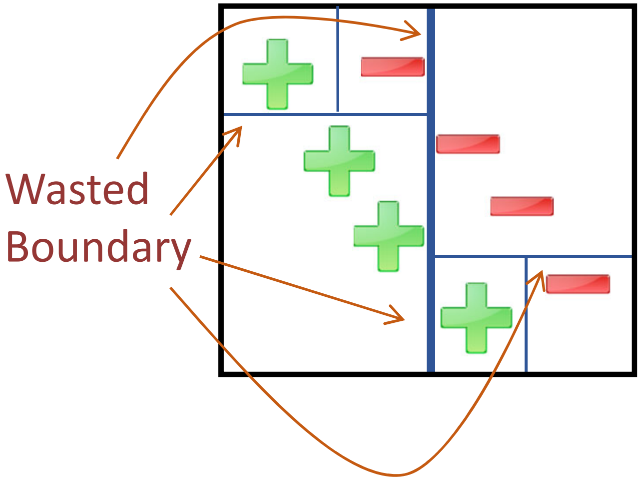
- Cannot easily model diagonal boundaries

Example:

Logistic regression can have arbitrary linear boundary



Decision Trees Require Complex Axis-Aligned Partitioning

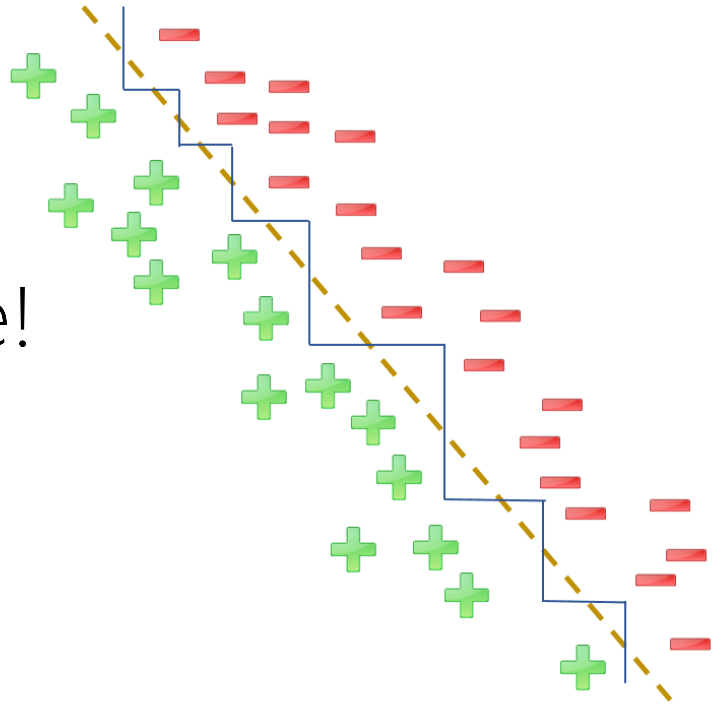


More Extreme Example

Decision Tree can waste a lot of model capacity on useless boundaries.

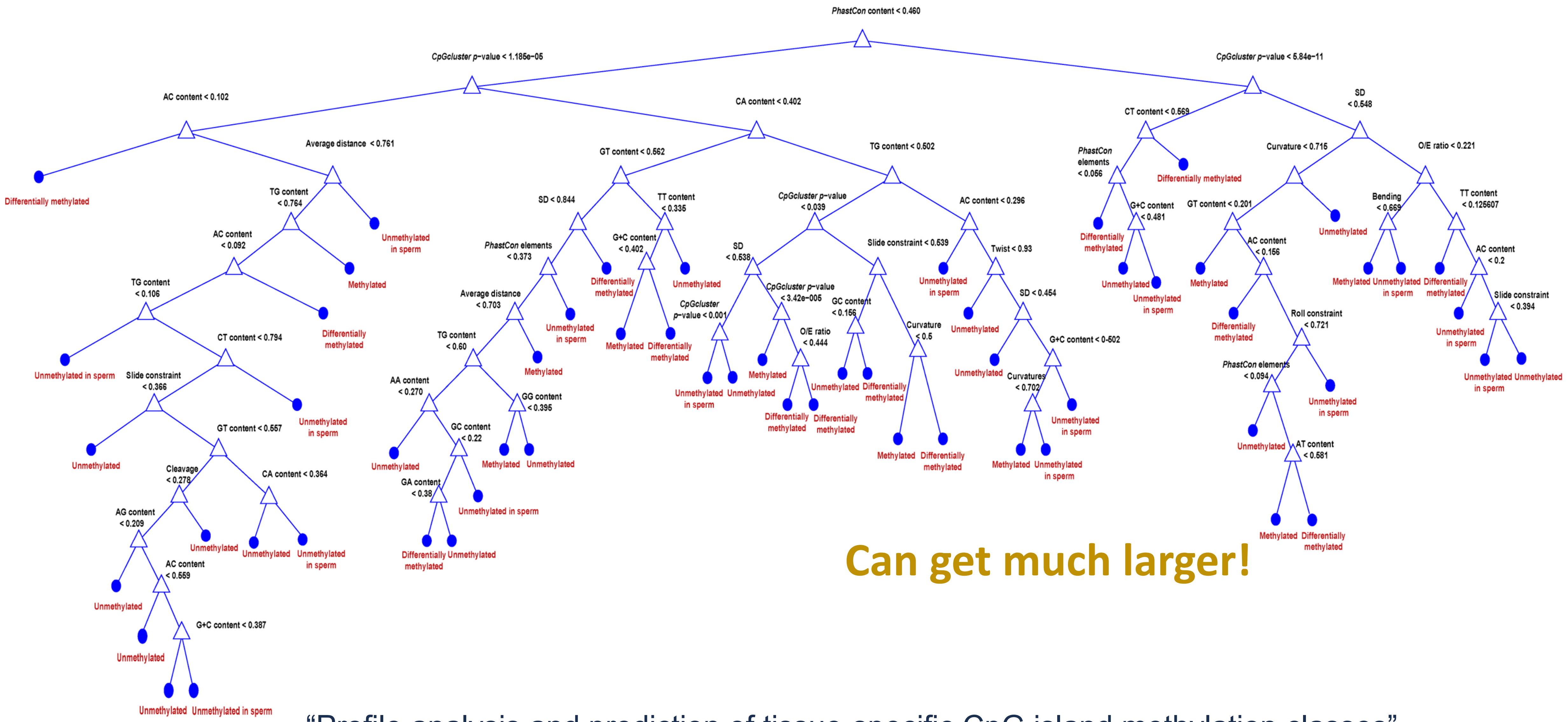
slide credit: Yisong Yue

Decision Trees vs Linear Models



- Still, Decision Trees are often more accurate!
- Their non-linearity powerful.
- Catch: requires sufficient training data.

Real Decision Trees



Can get much larger!

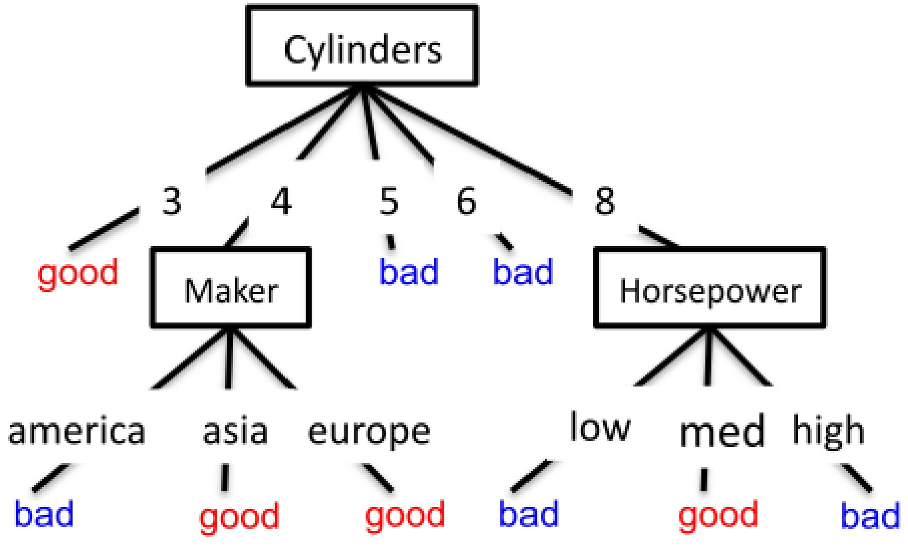
"Profile analysis and prediction of tissue-specific CpG island methylation classes"

Image Source: http://www.biomedcentral.com/1471-2105/10/116

slide credit: Yisong Yue

Model space

- How many possible models?
- 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

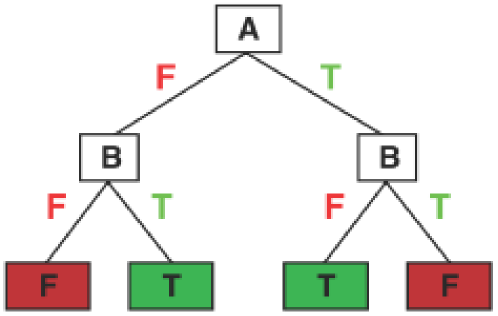
Y X

Can you think of a function that a decision tree could not represent?

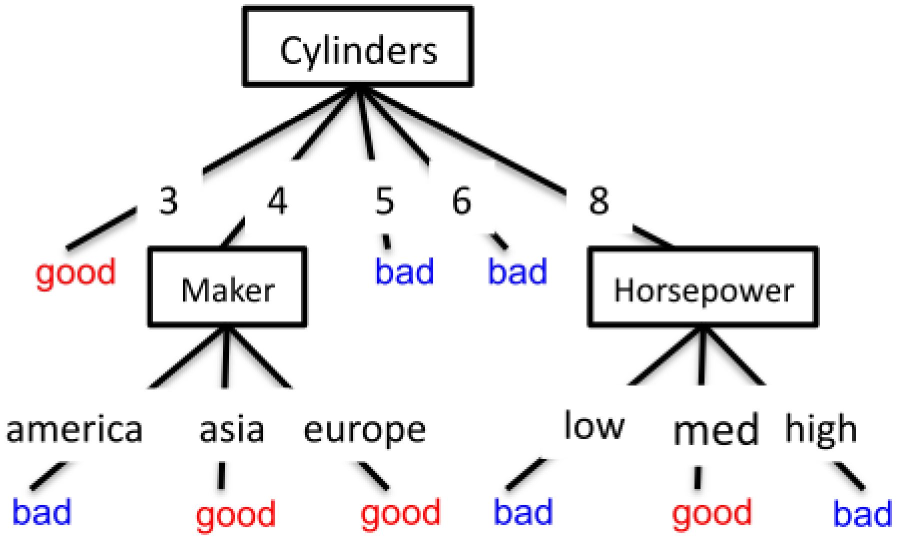
What functions can be represented?

- 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



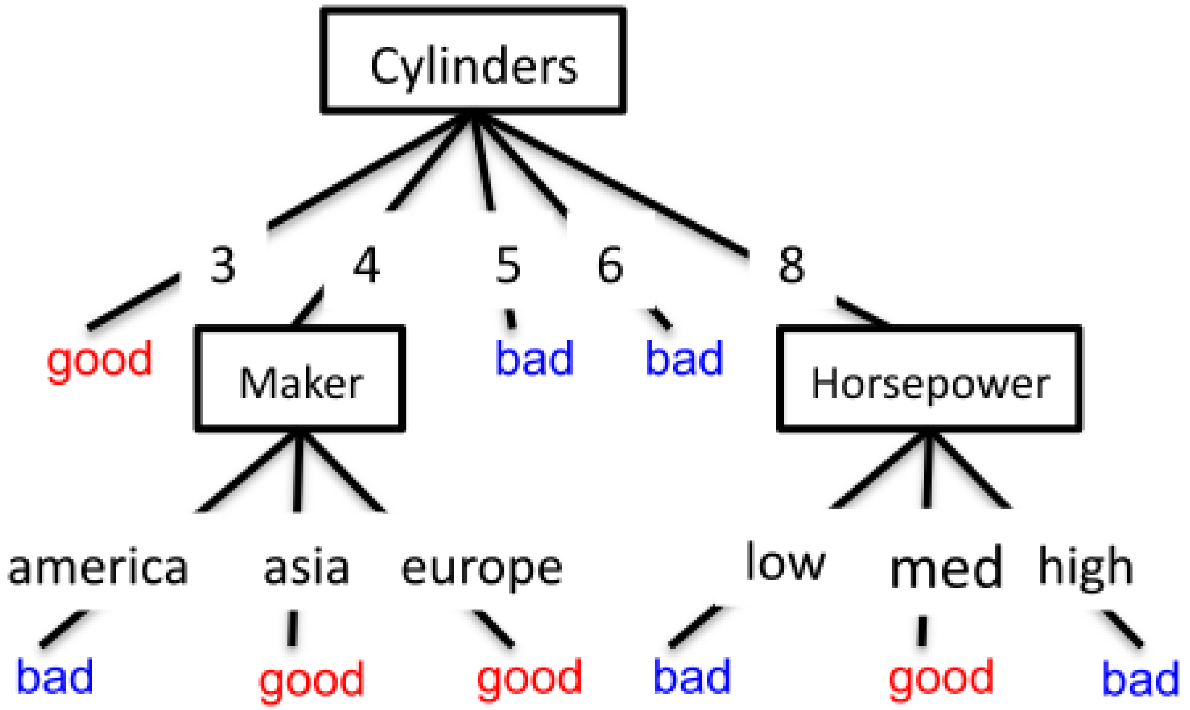
(Figure from Stuart Russell)



cyl=3 ∨ (cyl=4 ∧ (maker=asia ∨ maker=europe)) ∨ …

Hypothesis space

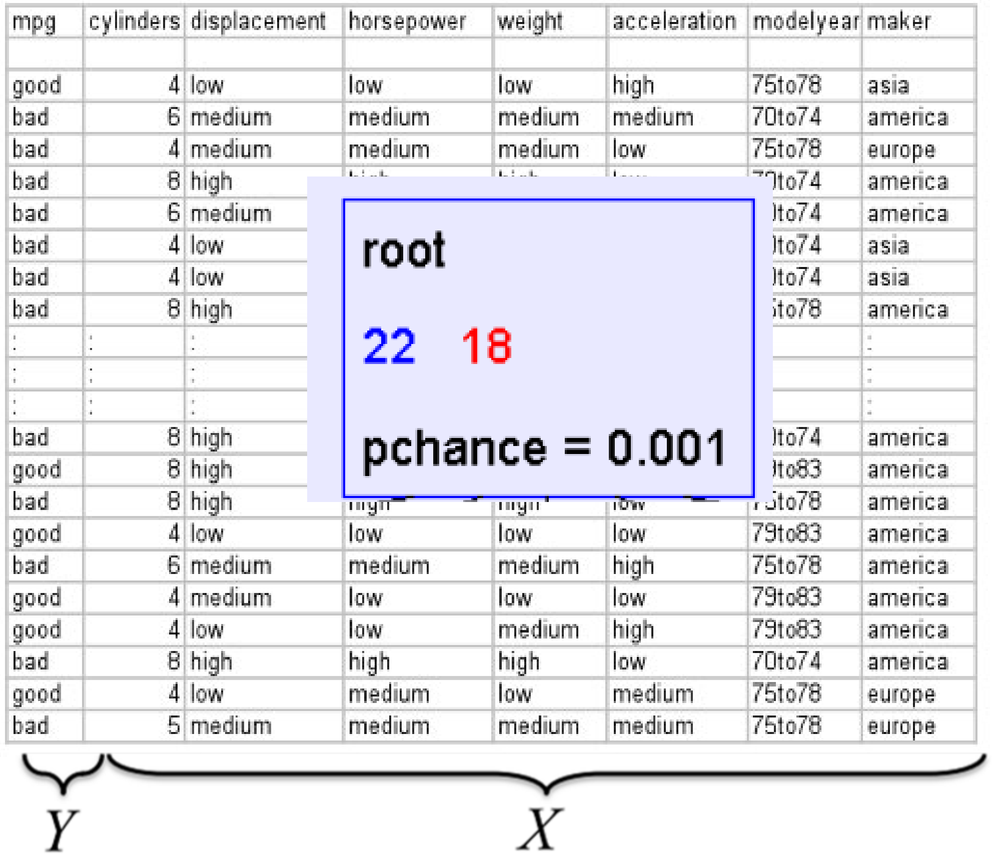
- How many possible models?
- What functions can be represented?
- How many will be consistent with a given dataset?
- How will we choose the best one?



Lets first consider how to build a tree from training data, then revisit this question.

What is the Simplest Tree?

root node only
predict majority class:
mpg=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	[illegible]	[illegible]	[illegible]	[illegible]	america
bad	6	medium	[illegible]	[illegible]	[illegible]	[illegible]	america
bad	4	low	[illegible]	[illegible]	[illegible]	[illegible]	asia
bad	4	low	[illegible]	[illegible]	[illegible]	[illegible]	asia
bad	8	high	[illegible]	[illegible]	[illegible]	[illegible]	america
:	:	:					:
:	:	:					:
:	:	:					:
bad	8	high	[illegible]	[illegible]	[illegible]	[illegible]	america
good	8	high	[illegible]	[illegible]	[illegible]	[illegible]	america
bad	8	high	[illegible]	[illegible]	[illegible]	[illegible]	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

Is this a good tree?

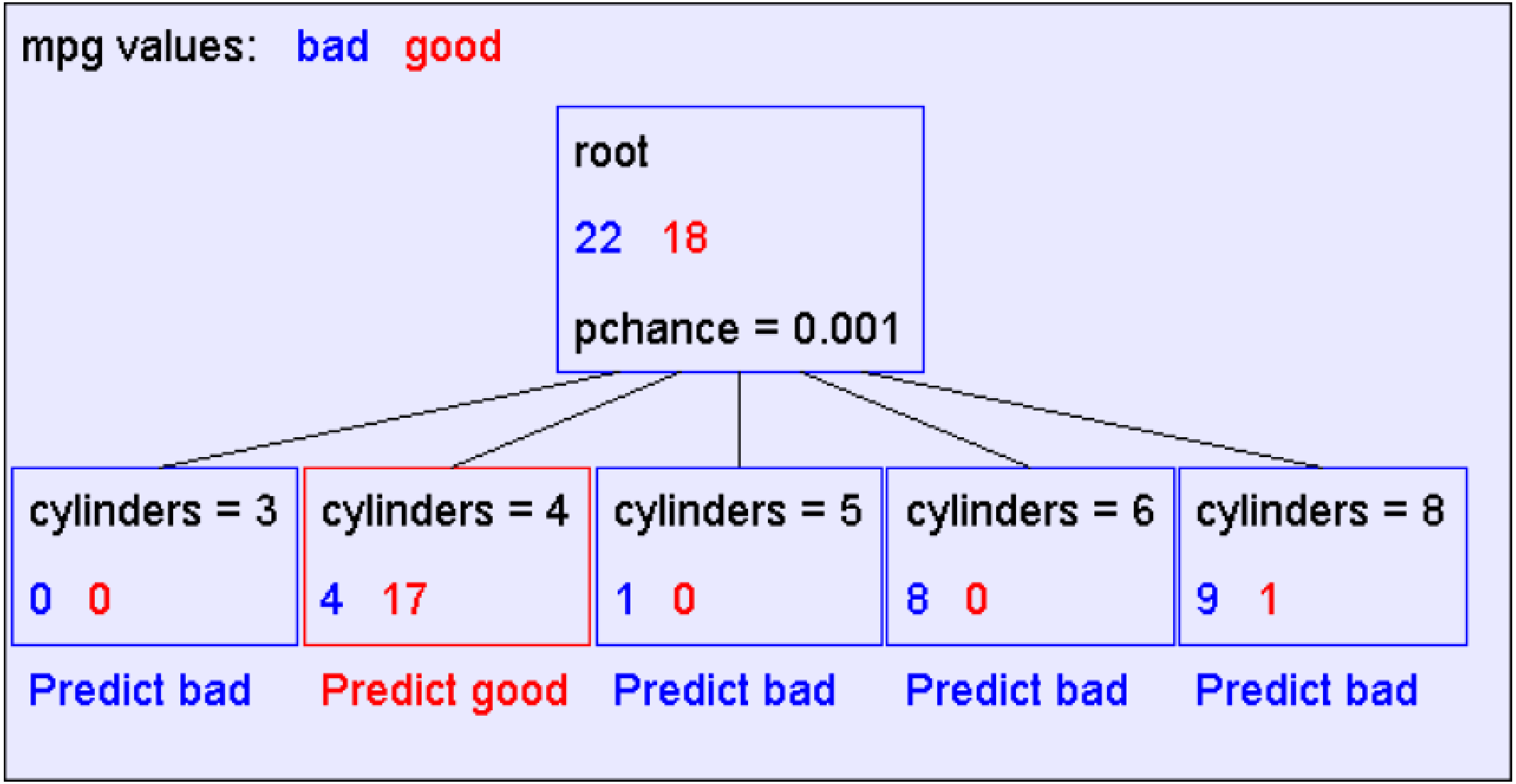
[22+, 18-]

Means:
correct on 22 examples
incorrect on 18 examples

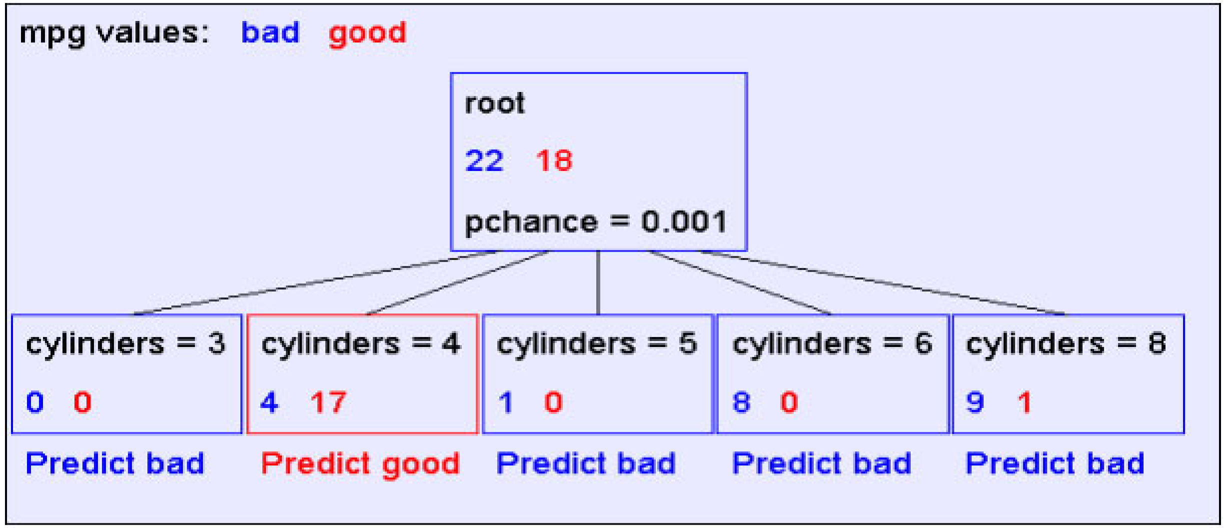
Next simplest tree: A Decision Stump (one feature splitting node)

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

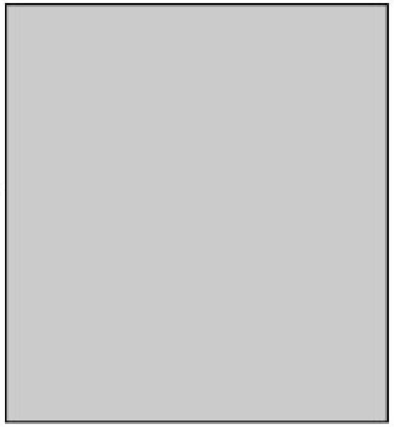
$\underbrace{\text{mpg}}_{Y}$ $\underbrace{\text{cylinders} \ldots \text{maker}}_{X}$



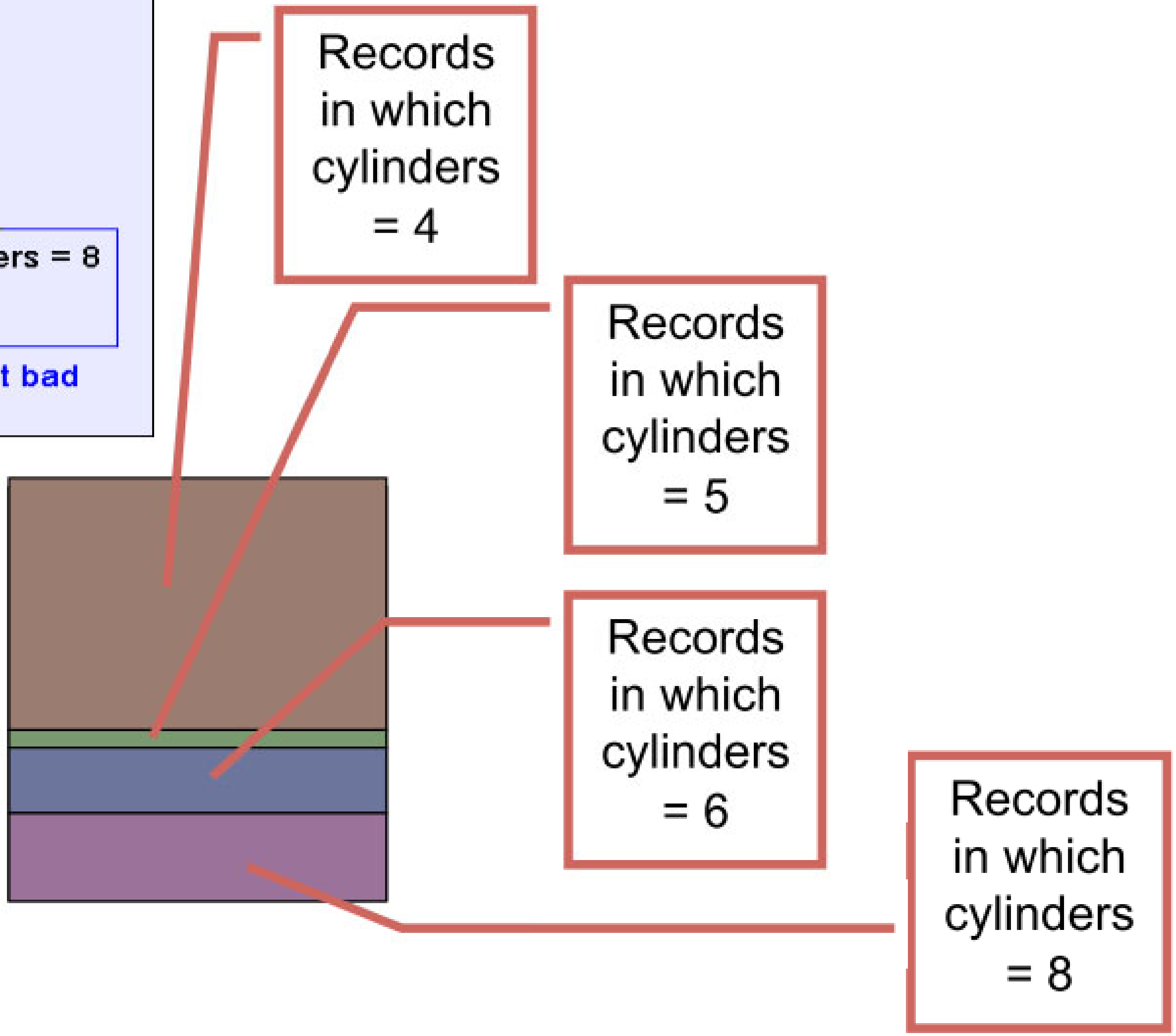
Increase complexity by recursive partitioning



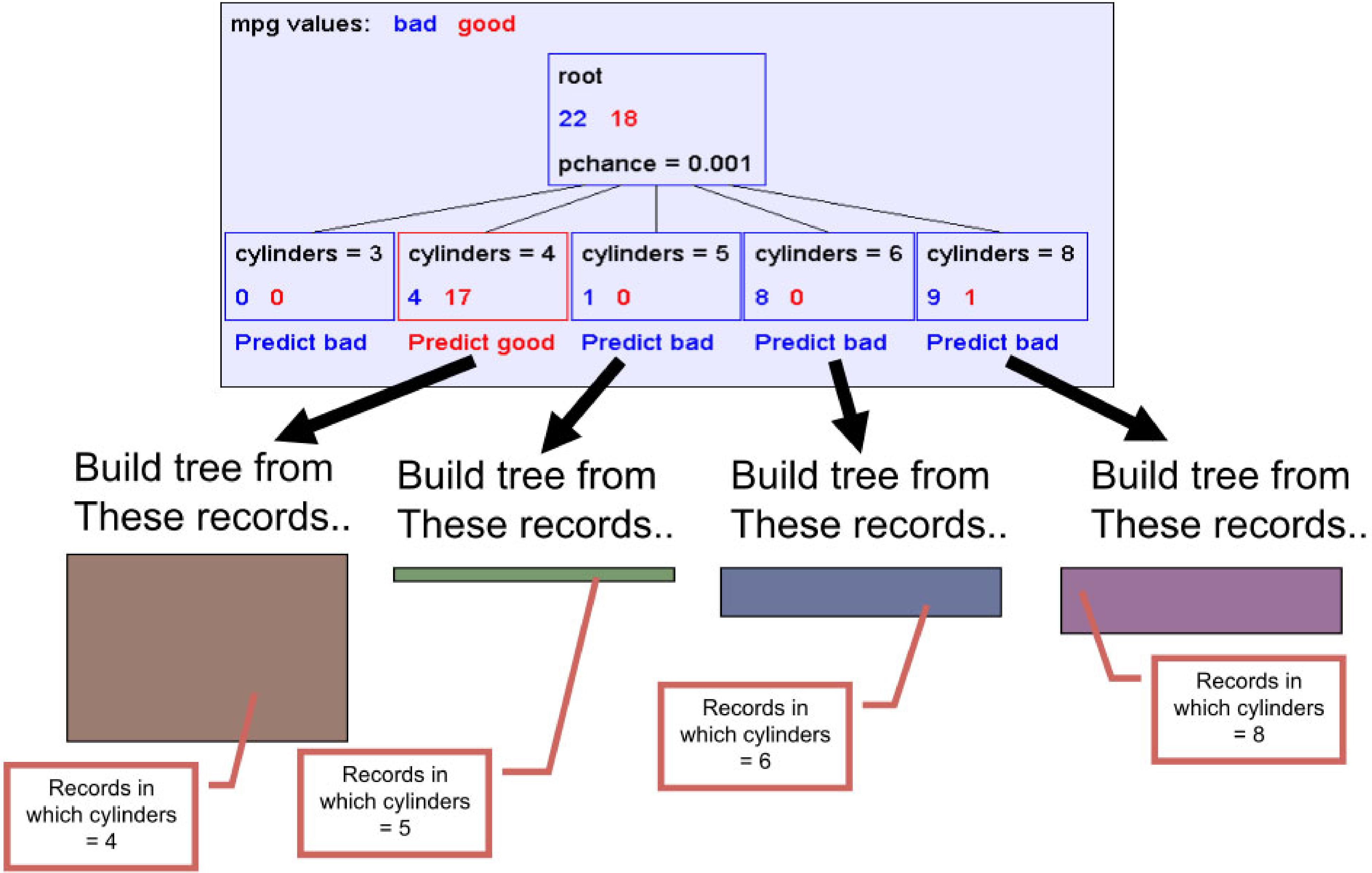
Take the Original Dataset..



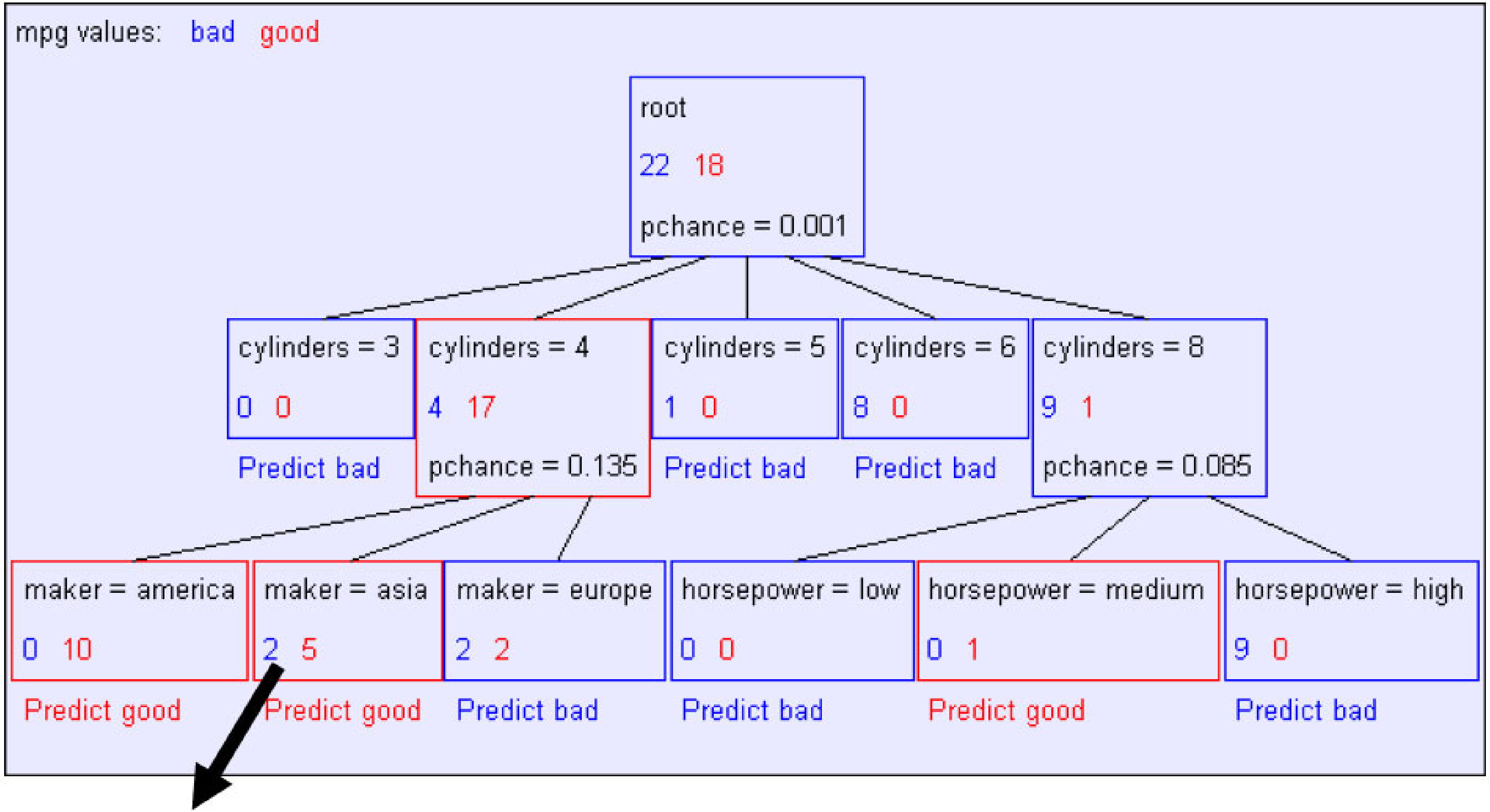
And partition it according to the value of the attribute we split on



Increase complexity by recursive partitioning

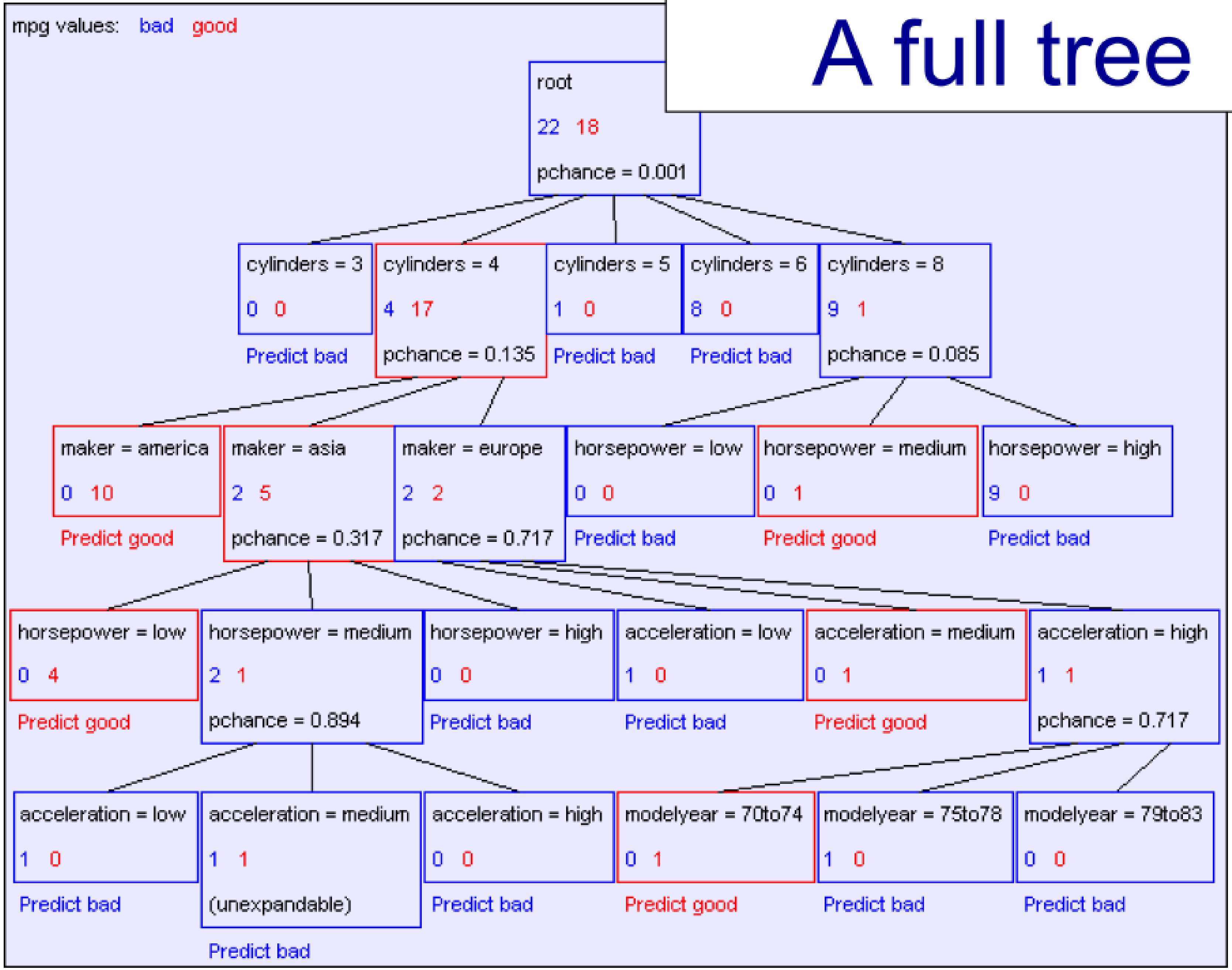


Now have second level of tree



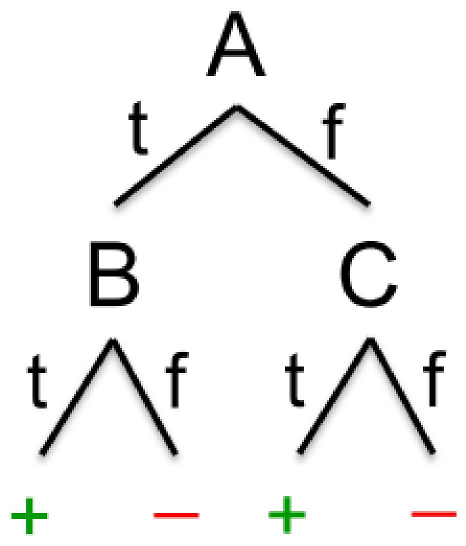
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

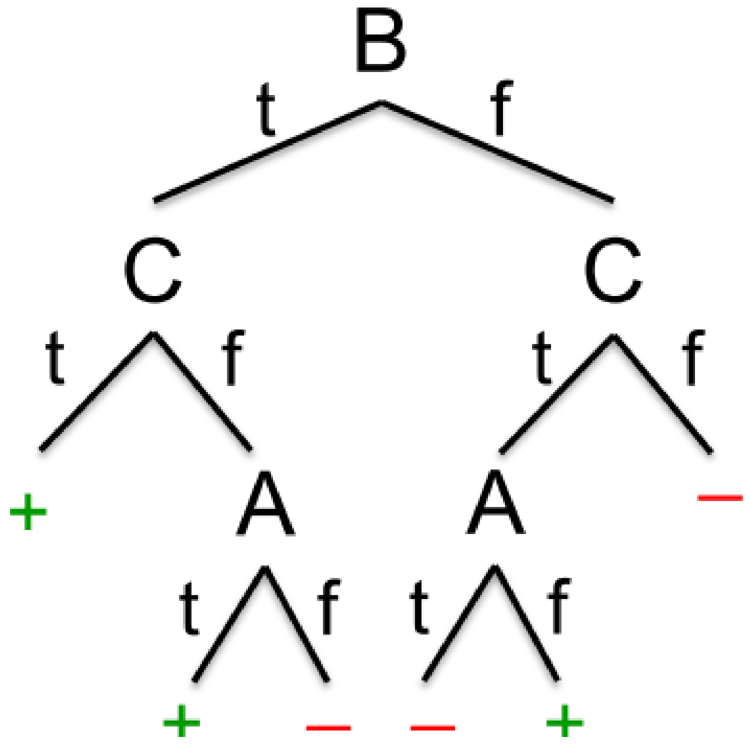
A full tree



- Each leaf has only one example, or is “unexpandable”.
- i.e., the value for any unused features is constant.

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





Which tree do we prefer?

“Occam’s razor”

- “Nunquam ponenda est pluralitis sin necesitate”

- “Entities should not be multiplied beyond necessity’

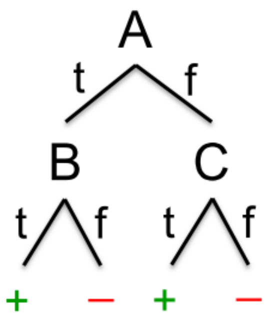
English Franciscan friar William of Ockham (c. 1287–1347)

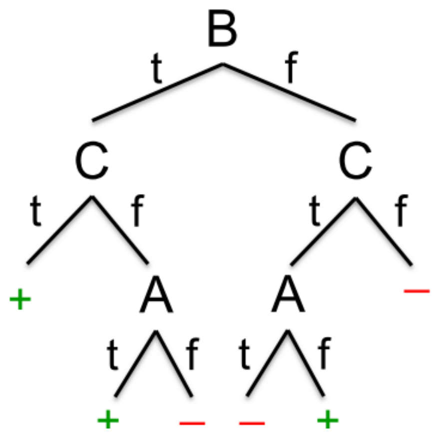
- “when you have two competing theories that make exactly the same predictions, the simpler one is the better”

https://www.biostat.wisc.edu/~craven/cs760/lectures/decision-trees.pdf

Why is Occam's razor a reasonable heuristic for decision tree learning?

- there are fewer short models (i.e. small trees) than long ones
- a short model is unlikely to fit the training data well by chance
- a long model is more likely to fit the training data well coincidentally





https://www.biostat.wisc.edu/~craven/cs760/lectures/decision-trees.pdf

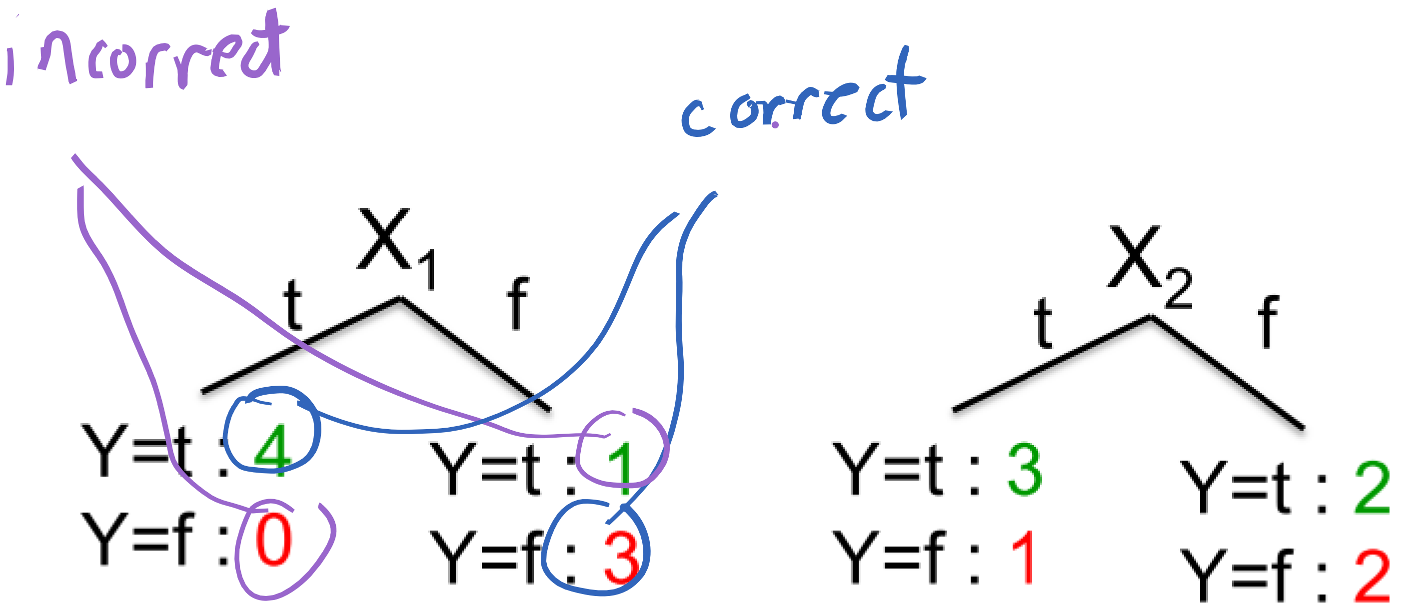
Learning simplest decision tree is hard

- 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

but what does this mean?

How to pick a good feature to split on?

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 $p(Y)$.
- Goal: find distributions where we're not surprised to see the label.
- i.e. those with low entropy.

Entropy: a measure of expected surprise

Think about a flipping a coin once, and how surprised you would be at observing a head.

$p(head) = 0.5$

$p(head) = 0$

$p(head) = 0.01$

$p(head) = 1$

Entropy: a measure of expected surprise

- The “surprise” of observing that a discrete random variable (RV) Y takes on value $\boldsymbol{k}$ is:

$$log\,\frac{1}{P(Y=k)} = -\log(P(Y=k))$$

- As $P(Y=k) \rightarrow \mathbf{0}$, the surprise of observing $\boldsymbol{k}$ approaches $\boldsymbol{\infty}$.
- As $P(Y=k) \rightarrow \mathbf{1}$, the surprise of observing $\boldsymbol{k}$ approaches $\mathbf{0}$.
- The entropy of the distribution of $\boldsymbol{Y}$ is the *expected surprise*:

$$H(Y) \equiv E_Y[-\log P(Y=k)] = \sum_k P(Y=k)\log P(Y=k)$$

Entropy example: flipping a coin

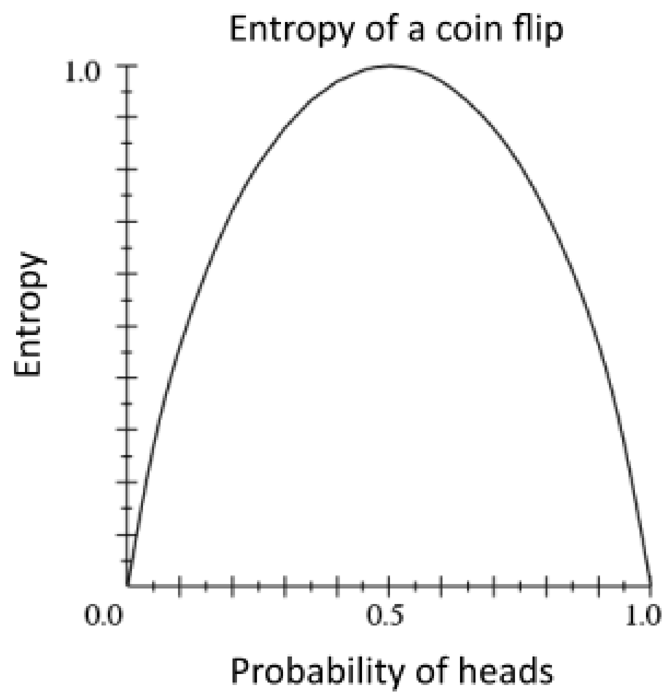
$$H(Y) = -\sum_{i=1}^{\kappa} P(Y = y_i) \log_2 P(Y = y_i)$$

P(Y=t) = 5/6

P(Y=f) = 1/6

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

$= 0.65$



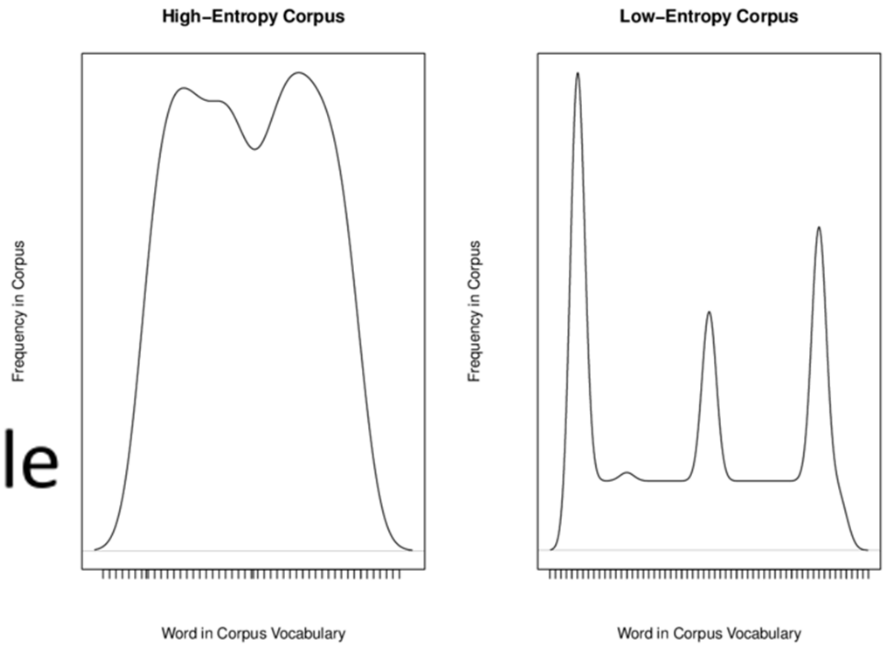
Entropy of a random variable Y:

"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 lows and highs
- Values sampled from it are more predictable



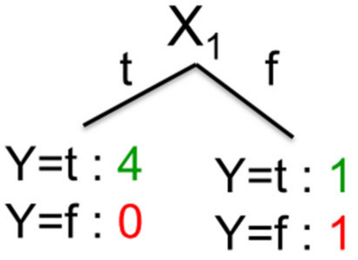
https://www.researchgate.net/figure/Hypothetical-distributions-of-term-frequency-in-high-and-low-entropy-corpora_fig1_305417514

(Slide from Vibhav Gogate)

Entropy for choosing feature to split next

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

- Cannot control entropy of RV Y (label).
- Can control the entropy of $p(Y|X)$, i.e. after splitting on feature(s).
- Goal: recursively reduce the *conditional entropy* at each node split until expected surprise at leaf nodes=0.



Conditional entropy (for a tree split)

weighted average of entropy on each side of the split of feature into $x_j < v$ *and* $x_j \geq v$

$$H(Y|X_{j,v}) := P(X_{j,v}=1)H(Y|X_{j,v}=1) + P(X_{j,v}=0)H(Y|X_{j,v}=0)$$

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

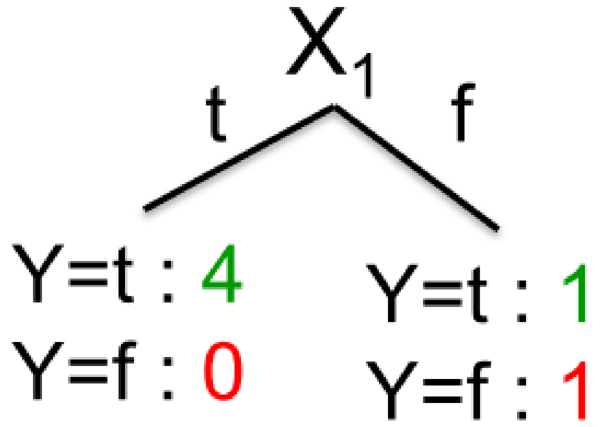
Conditional entropy (for a tree split)

weighted average of entropy on each side of the split of feature into $x_j < v$ *and* $x_j \geq v$

$$H(Y|X_{j,v}) := P(X_{j,v} = 1)H(Y|X_{j,v} = 1) + P(X_{j,v} = 0)H(Y|X_{j,v} = 0)$$

Example:

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



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

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

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

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

$\qquad\qquad - 2/6\,(1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

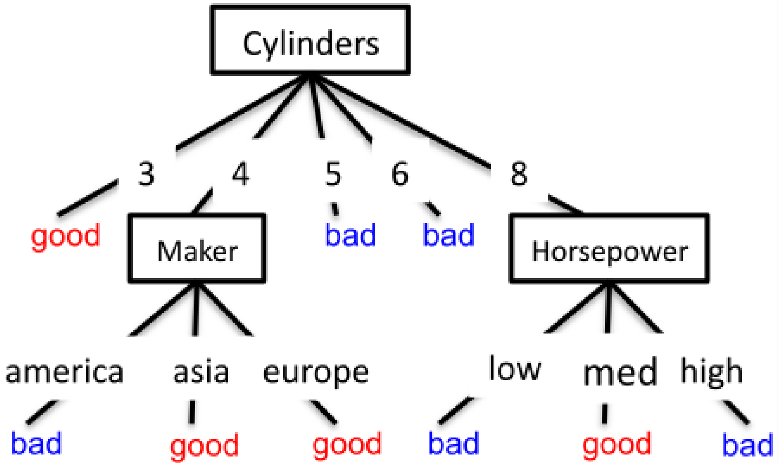
$\qquad = 2/6$

Conditional entropy (for a tree split)

weighted average of entropy on each side of the split of feature into $x_j < v$ and $x_j \geq v$

$$H(Y|X_{j,v}) := P(X_{j,v}=1)H(Y|X_{j,v}=1) + P(X_{j,v}=0)H(Y|X_{j,v}=0)$$

Goal: at each recursion, find the feature (and split) which minimizes the conditional entropy.



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

$\qquad\qquad - 2/6\ (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$

$\qquad = 2/6$

T	T	T
T	F	T
F	T	T
F	F	F

Equivalently: maximize the *information gain*

(also called the *mutual information*)

constant

$$\text{maximize } I(X_{j,v};Y) := H(Y) - H(Y|X_{j,v})$$

“Learning” Decision Trees (aka building from data)

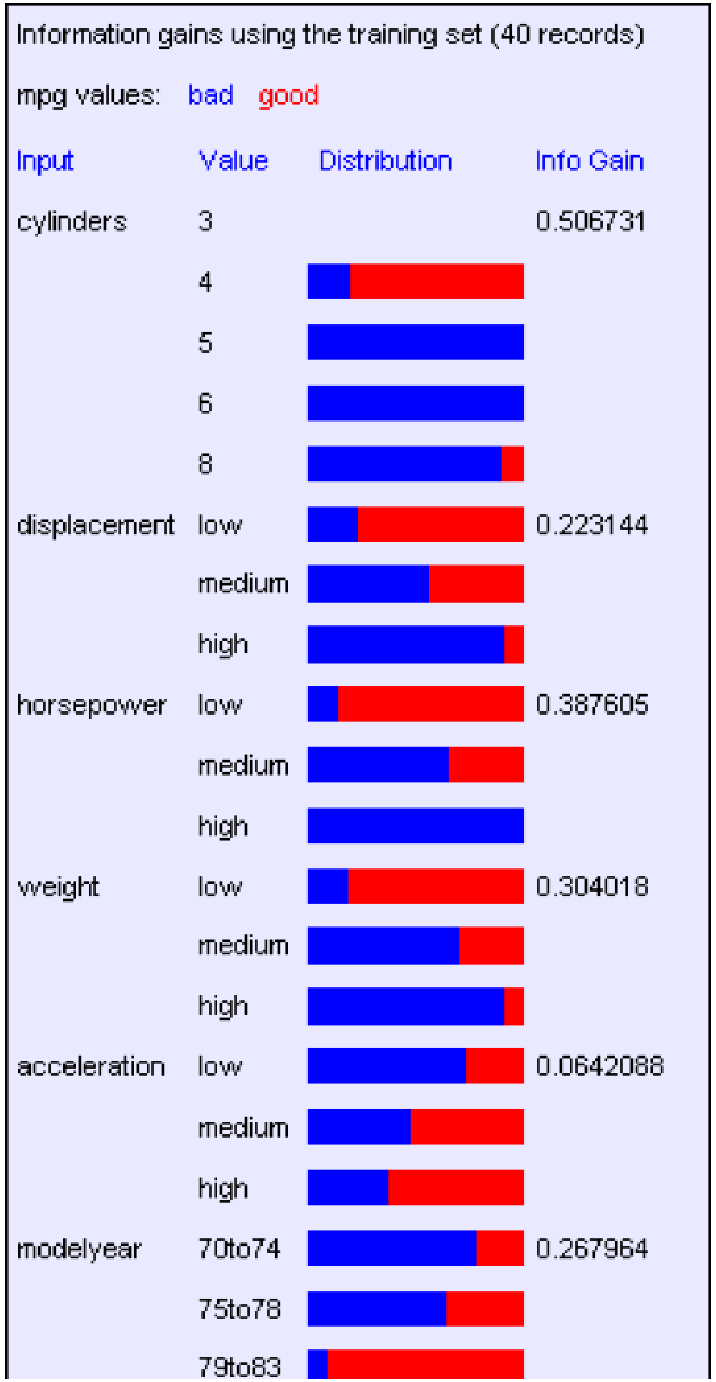
- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute:

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

- Recurse

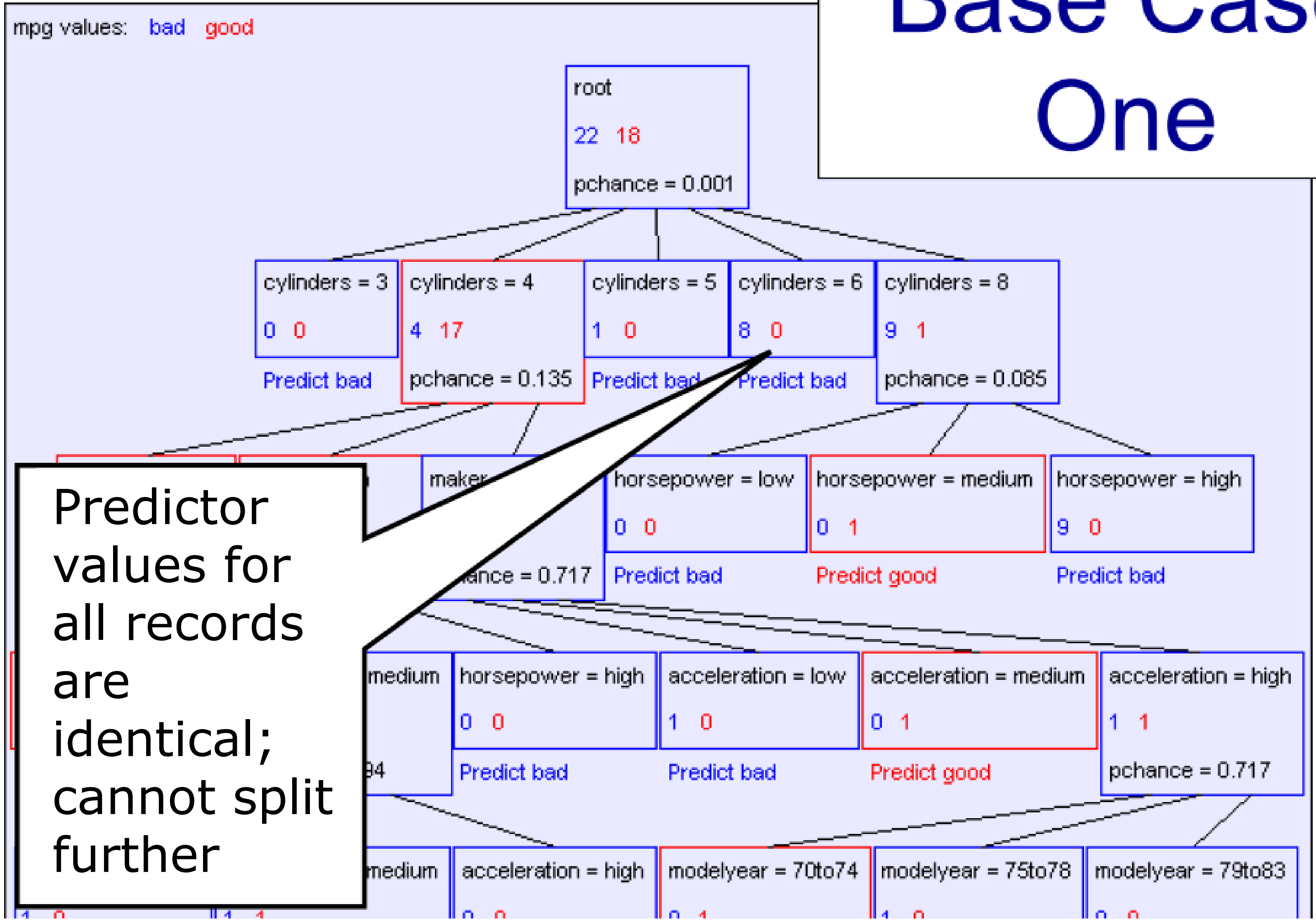
Back to the “mpg” example

- Compute information gain for every possible split at every node.
- Categorical features: split into each category, or could do one vs. rest.
- Real-valued features: pick a threshold to split on; try different thresholds.



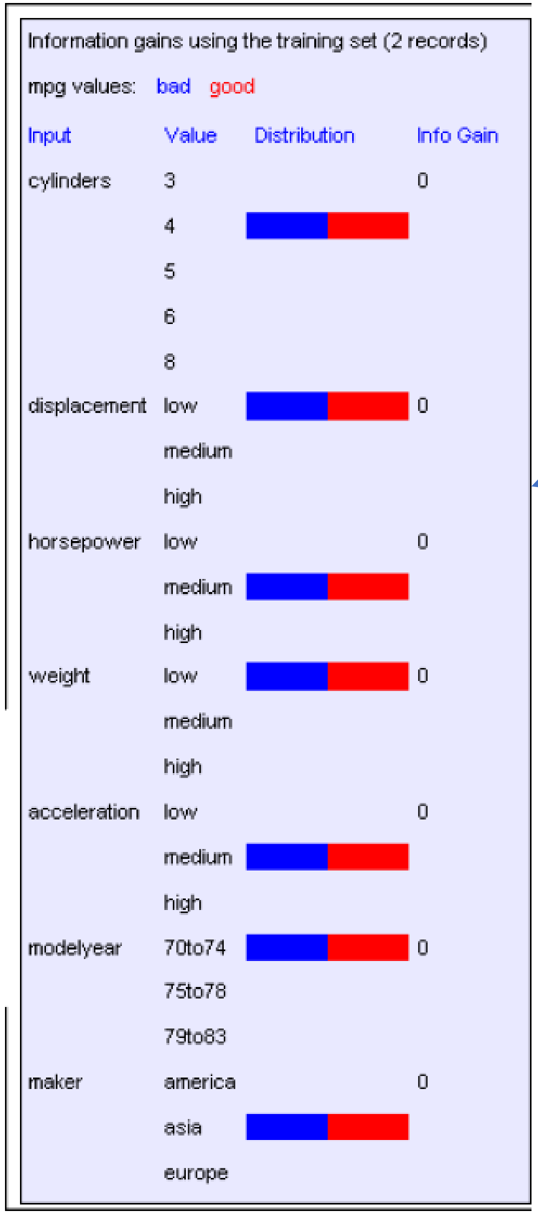
When to stop recursing on a node?

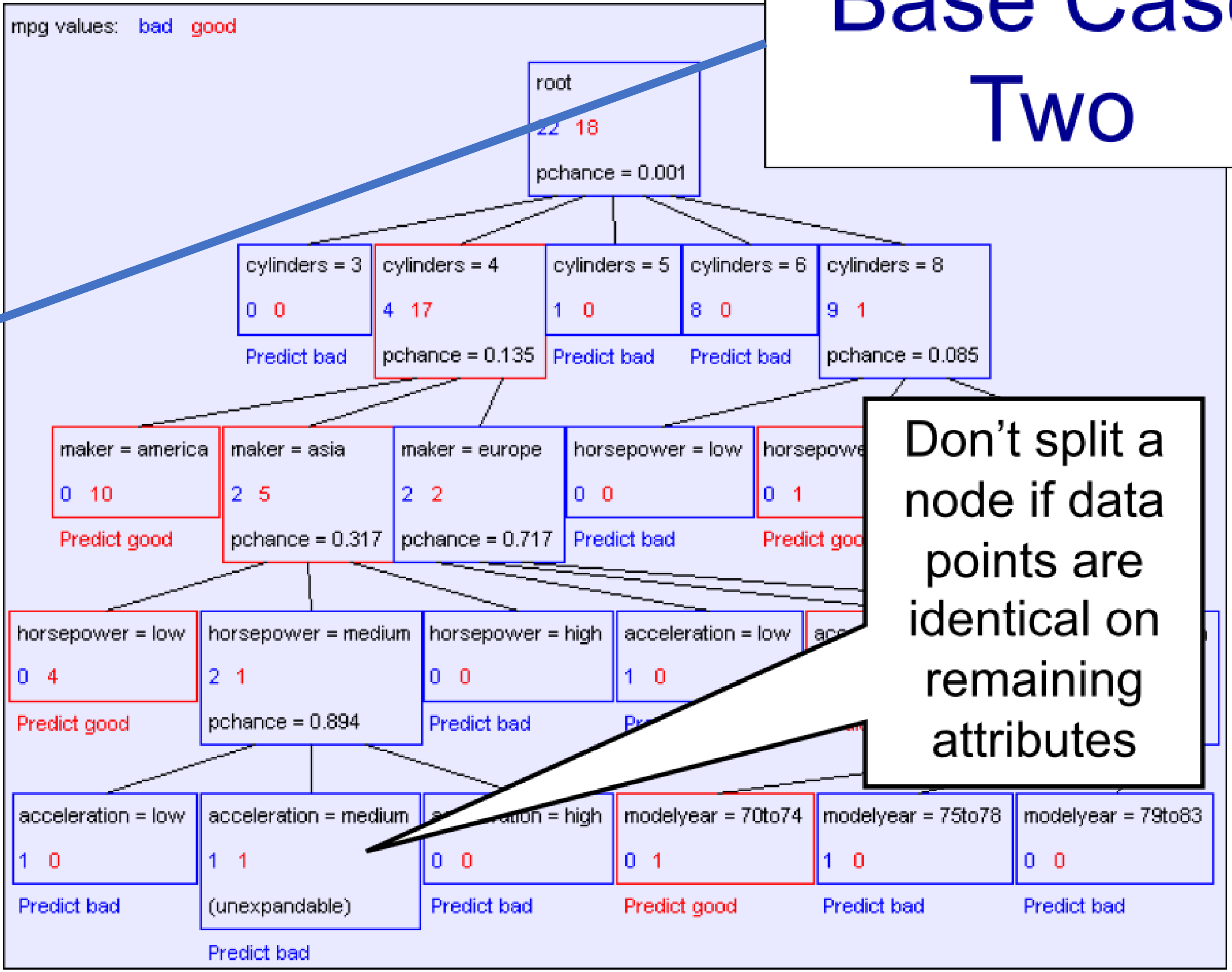
Base Case One



Predictor values for all records are identical; cannot split further

When to stop recursing on a node?





Base Case Two

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

When to stop recursing on a node?

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 zero information gain then don't recurse

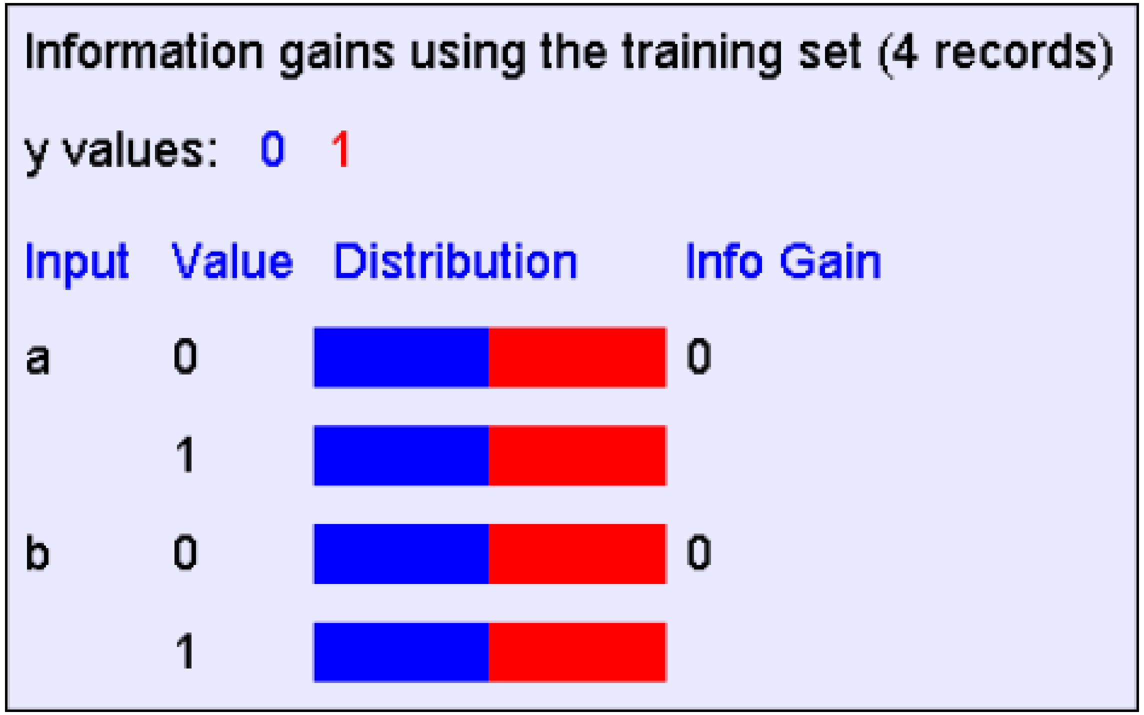
• *Is this a good idea?*

The problem with Base Case 3 for stopping

We are using a greedy heuristic.

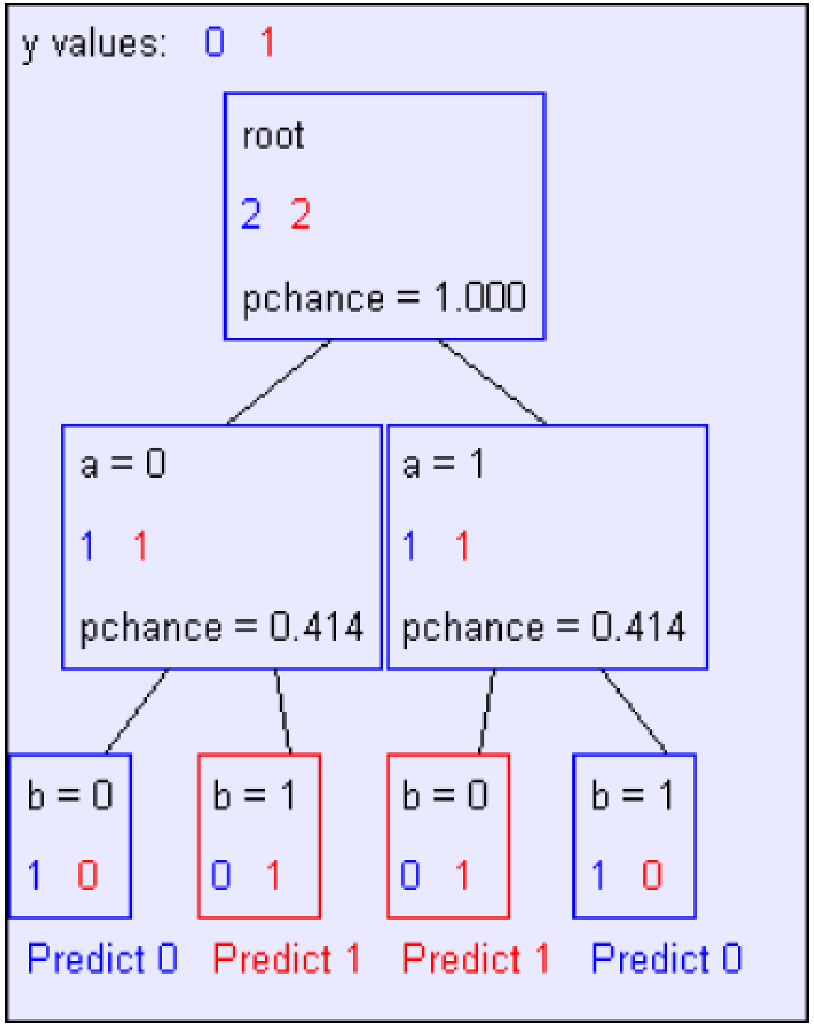
$y = a \text{ XOR } b$

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



If we omit Base Case 3 for stopping:

The resulting decision tree:



y = a XOR b

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

Instead, perform *pruning* after building the tree using:

1. Statistical test
2. Hold out performance.

Summary: building Decision Trees

X Y

BuildTree(*DataSet*,*Output*)

- If all output values are the same in *DataSet*, return a leaf node that says "predict this unique output" BASE CASE 1
- If all input values are the same, return a leaf node that says "predict the majority output" BASE CASE 2
- Else find attribute *X* with highest Info Gain
- Suppose *X* has n_X distinct values (i.e. X has arity n_X).
  - Create a non-leaf node with n_X children.
  - The *i*'th child should be built by calling

    BuildTree(DS_i,*Output*)

    Where DS_i contains the records in DataSet where X =

- What fundamental flaw does this have?
- It will keep going until the data are *perfectly* split/labelled.

This algorithm will always overfit the data

Naive decision trees have no learning bias

- Training set error is always zero!
  - (If there is no label noise)
- Must introduce some bias towards simpler trees

How to regularize tree-building

- Limit on depth of the tree.
- Min # data points at a node.
- Backward-greedy pruning (greedily remove node & descendants that most improves validation performance).
- Early stopping according to any number of criteria, such as a statistical test (but then subject to the problem of Base Case 3).
- Take an *ensemble* of small trees.

Decision Tree for regression

- Prediction can be mean of those in the leaf "bucket".
- Or linear regression within a leaf bucket.
- Now choose nodes to split on by minimizing the sum of variances after a split:

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

CS 189/289

Today's lecture

1. Decision Trees
2. Ensemble approaches

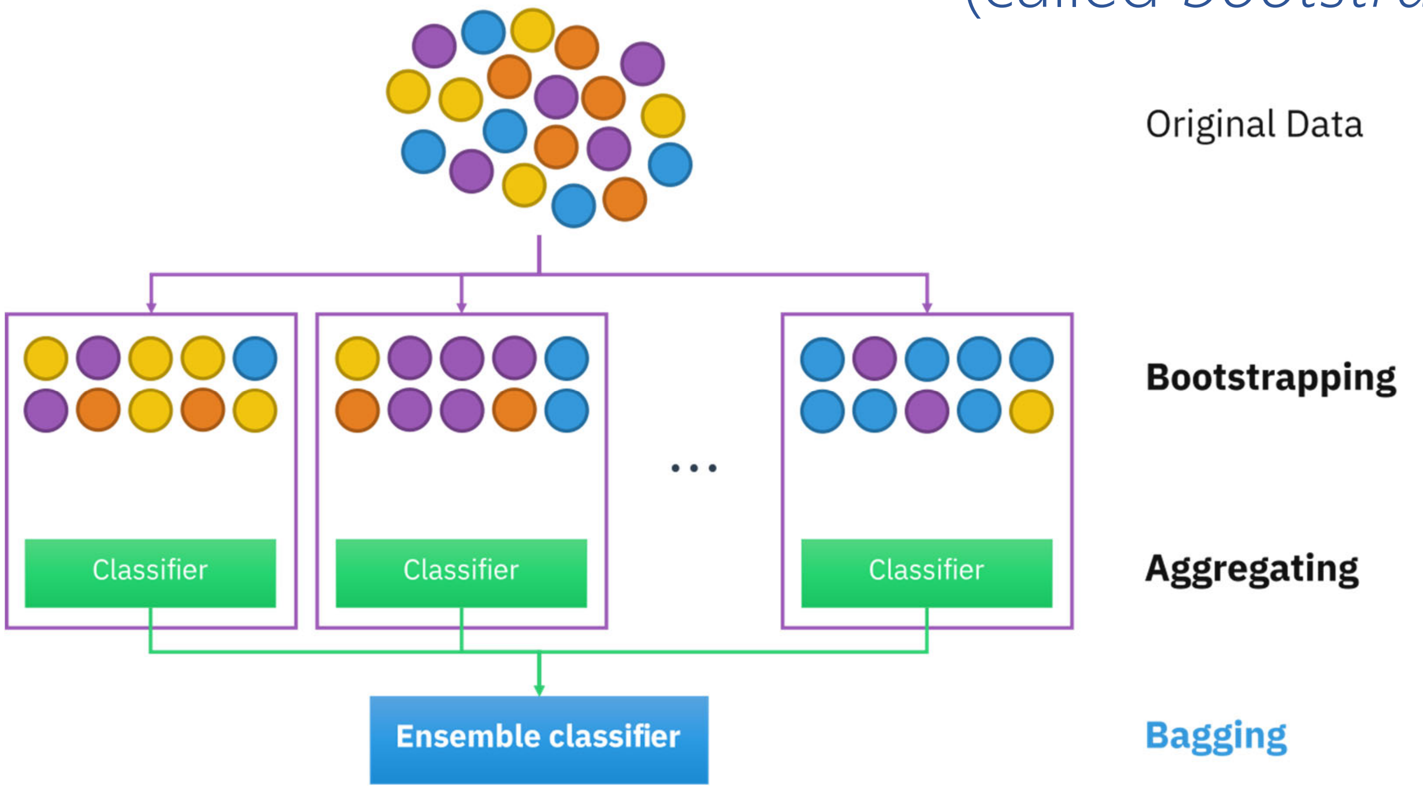
Bolstering Decision Trees

1. Decision trees can easily overfit if we don't regularize considerably.
2. Slightly different samples can lead to very different trees (high variance models).
3. If we average several randomized trees, we tend to do better: *Random Forests*.
4. Instantiation of broader, commonly used idea of *ensembling* models.

Creating *ensemble* of DTs

1. Train $\boldsymbol{M}$ models using $\boldsymbol{M}$ samples of size $\boldsymbol{n'}$ with replacement, from $\boldsymbol{n}$ total data points.

(called *bootstrap replicates* of the data)



These trees will be weakly correlated.

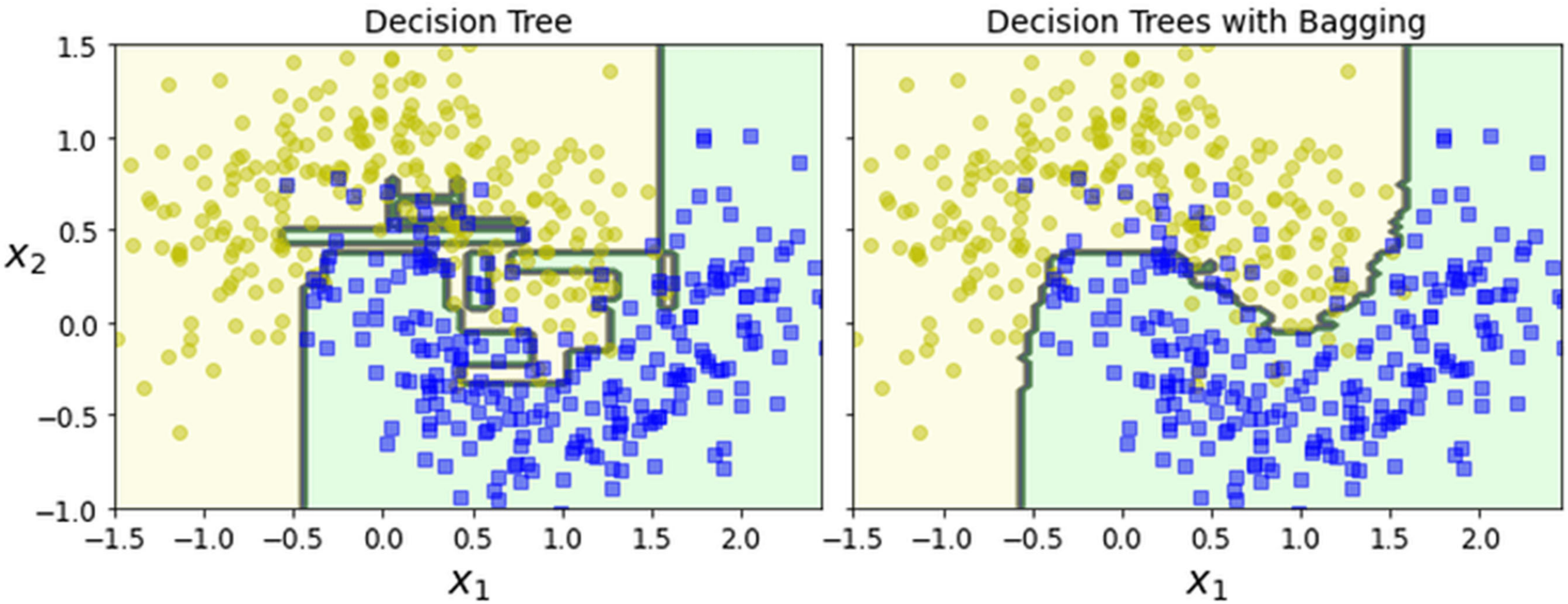
https://upload.wikimedia.org/wikipedia/commons/thumb/c/c8/Ensemble_Bagging.svg/1280px-Ensemble_Bagging.svg.png

Creating *ensemble* of DTs

1. Train $\boldsymbol{M}$ models using $\boldsymbol{M}$ samples of size $\boldsymbol{n'}$ with replacement, from $\boldsymbol{n}$ total data points.
2. When building these $\boldsymbol{M}$ DTs, allow use of only a random subset of the features (each time you recurse to create a new node), e.g. 2/3 of features.
3. 1 + 2 + averaging $\boldsymbol{M}$ predictions is called a *Random Forest*.
4. 1 + averaging is called *Bagging* (“bootstrap aggregation”)

(These DTs are not uncorrelated, but randomization helps to make their errors uncorrelated, which provides the advantage in model averaging approaches.

Empirical Comparison of Decision Boundaries



https://thecleverprogrammer.com/2020/07/31/bagging-and-pasting-in-machine-learning/

Other forms of ensemble modelling

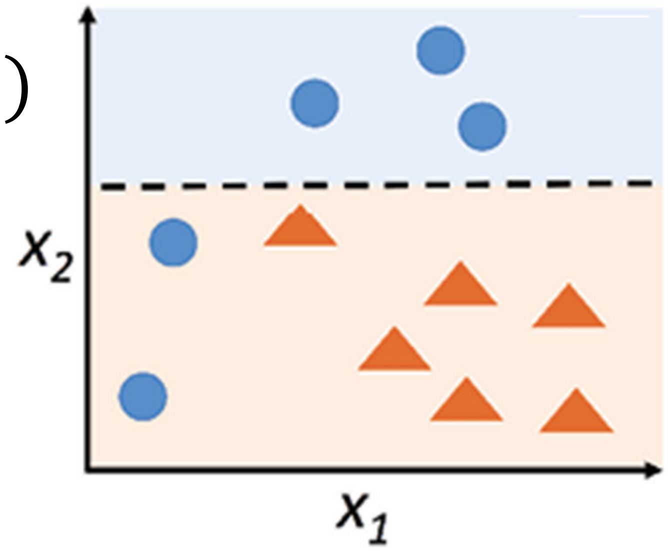
- So far we have considered *averaging* several models together to reduce the variance, $y = \frac{1}{m}\sum_{m=1}^{M} G_m(x)$.
- Why not consider a weighted average,
  $y = \sum_{m=1}^{M} \alpha_m G_m(x)$?
- Given a set of trained models of any kind (e.g. neural network, decision tree, …), $\{G_m(x)\}$, if we then optimize a loss for the weights $\{\alpha_m\}$, this is called stacking.
- What about jointly optimizing $\{G_m(x)\}$ and $\{\alpha_m\}$?
- Difficult optimization problem, but…

Boosting

- Too hard to jointly optimize $\{G_m(x)\}$ and $\{\alpha_m\}$ in
  $$y = \sum_{m=1}^{M} \alpha_m G_m(x) \, .$$
- Instead, lets use a greedy approximation wherein we sequentially train the next $G_m(x)$ conditioned on all previous learned base models $\{G_{m-1}, G_{m-2}, \ldots, G_1\}$, and their corresponding weights, $\{\alpha_{m-1}, \alpha_{m-2}, \ldots, \alpha_1\}$.
- The intuition is that at each iteration, we will re-weight the training points to focus on those that are not correctly classified.

Boosting at an intuitive level (decision stump)

$G_1(x)$

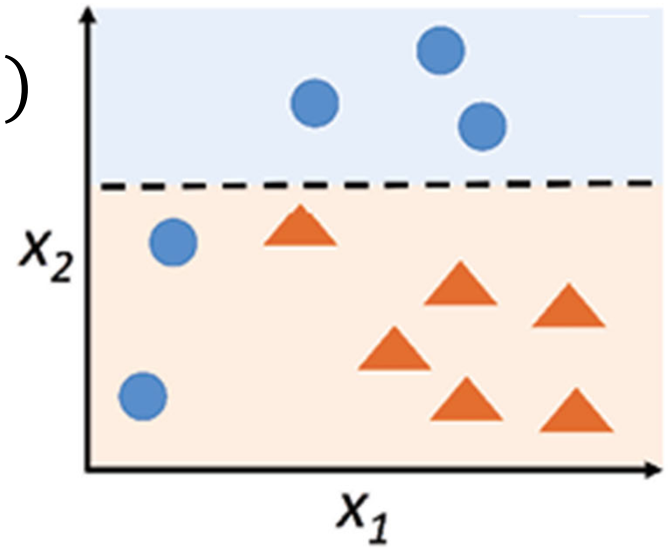


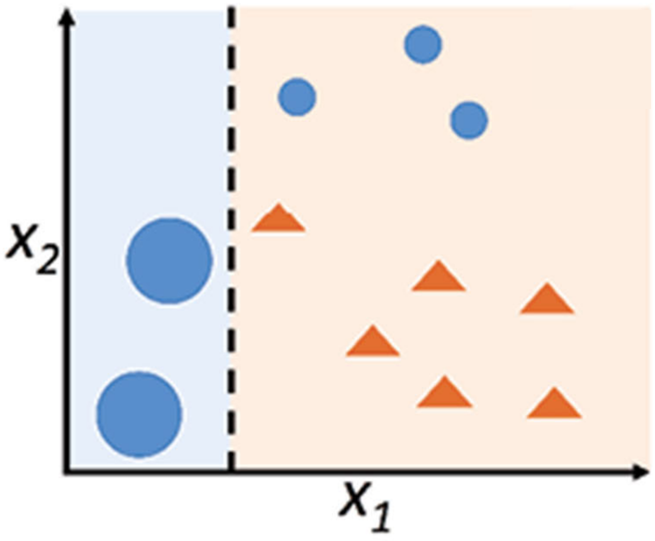
$$G(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

https://vitalflux.com/adaboost-algorithm-explained-with-python-example/

Boosting at an intuitive level (decision stump)

$G_1(x)$





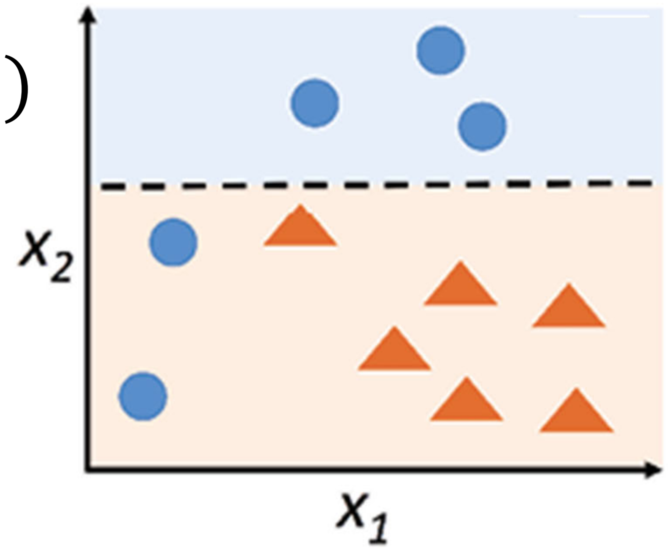
$G_2(x)$

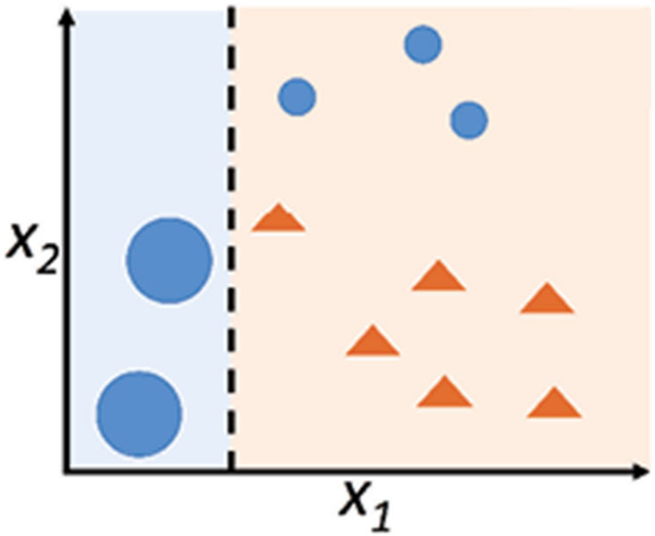
$$G(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

https://vitalflux.com/adaboost-algorithm-explained-with-python-example/

Boosting at an intuitive level (decision stump)

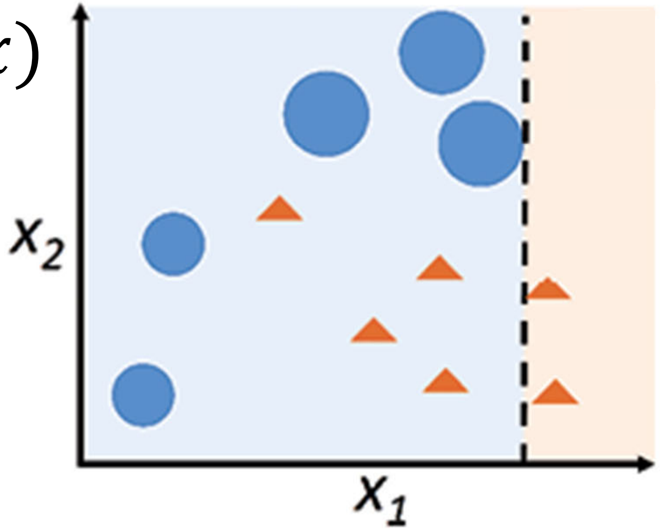
$G_1(x)$





$G_2(x)$

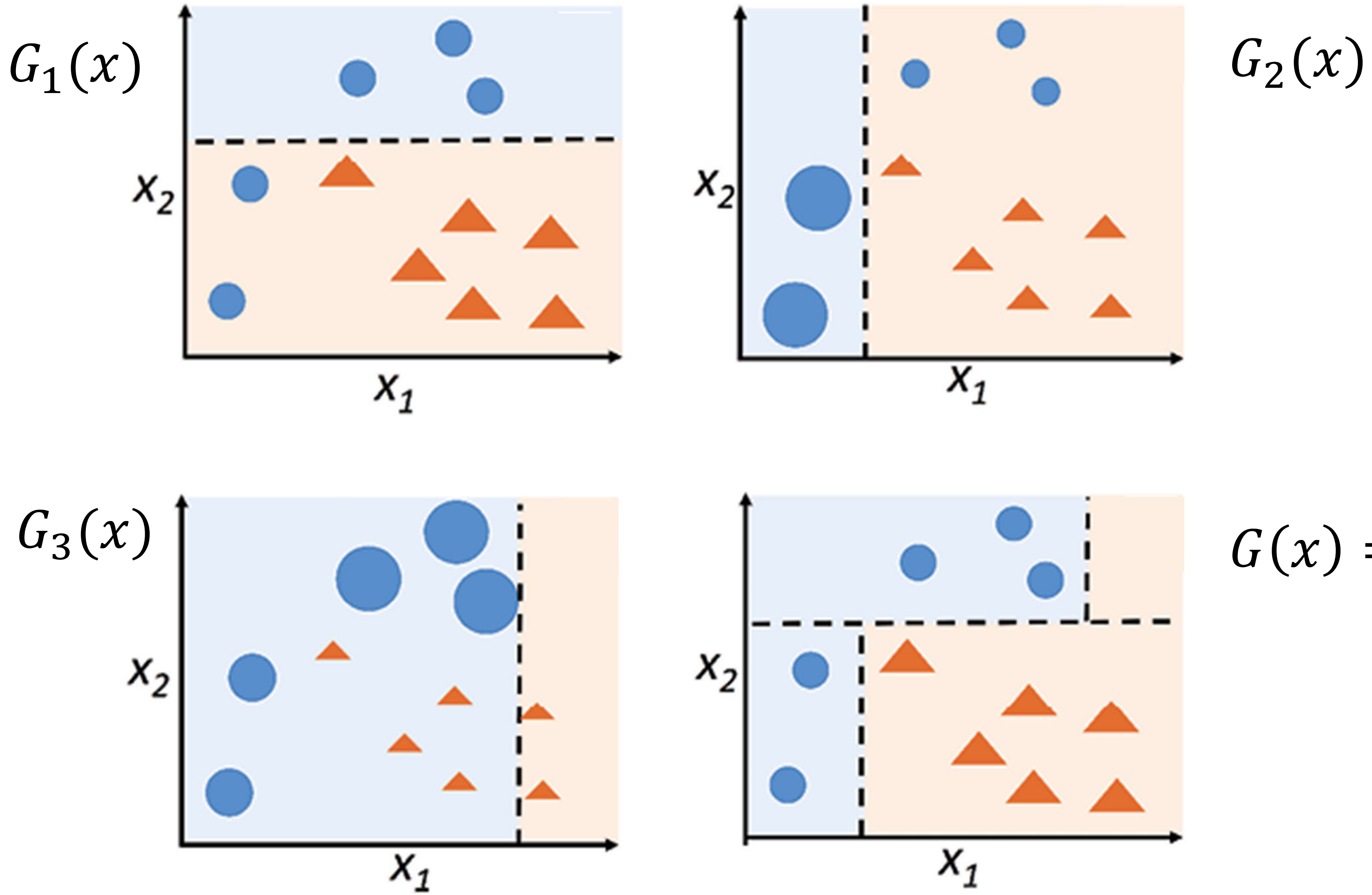
$G_3(x)$



$$G(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

https://vitalflux.com/adaboost-algorithm-explained-with-python-example/

Boosting at an intuitive level (decision stump)

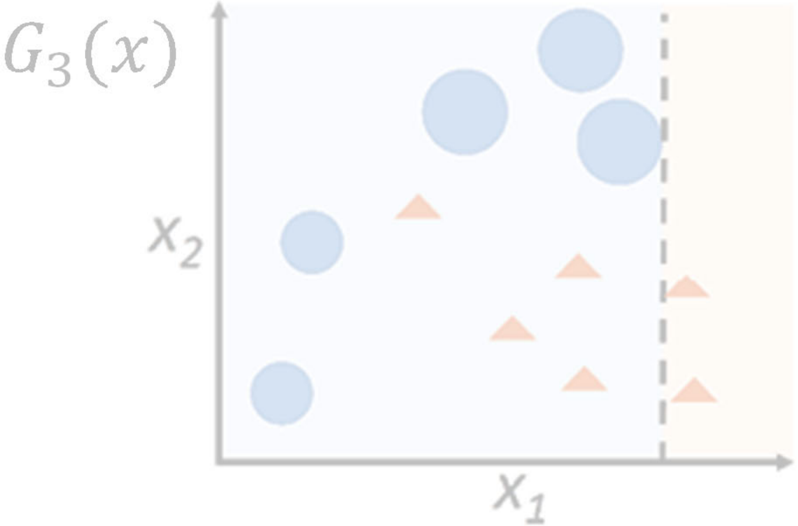


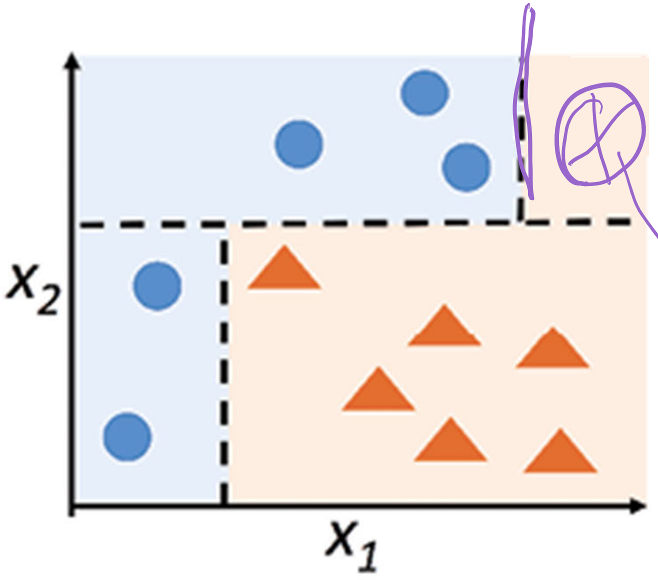
$$G(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

https://vitalflux.com/adaboost-algorithm-explained-with-python-example/

Boosting at an intuitive level (decision stump)

- Final boosted model looks a lot like a decision tree.
- Can you spot anything that makes you think a decision tree could not have come up with this?
- (Hint: try to reconstruct what a DT algorithm would do)





$$G(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

DT would not split here because no data (already 100% pure)

https://vitalflux.com/adaboost-algorithm-explained-with-python-example/