Introduction to Machine Learning (CSCI-UA 473): Fall 2021

Lecture 10: Decision Trees and Forests

Sumit Chopra
Courant Institute of Mathematical Sciences
Department of Radiology - Grossman School of Medicine
NYU

Slides derived from materials from David Sontag, Tuo Zhao, and Alexander Chouldechova

Lecture Outline

- Decision Trees
  - Motivation and intuition behind trees
  - How to decide where to branch
  - How to pick the best tree
  - The greedy learning algorithm
  - Advantages and disadvantages of trees

- Ensemble Learning
  - Bagging
    - Random Forests
  - Boosting
    - Gradient Boosted Decision Trees
    - Adaboost

Decision Trees

They have a long history in machine learning and statistical analysis

The first popular algorithm dates back to 1979

Some variants of it are still very popular as solutions to real world problems

They are intuitive to understand and easy to build

Based on the intuition on how people make decisions

When deciding on something, we consider a variety of possible factors, ask a series of questions pertaining to those factors, and follow a logical path to come to a conclusion

Decision Trees: Example

Problem definition: given a dataset of car engines predict whether the engine will have a good milage or poor milage

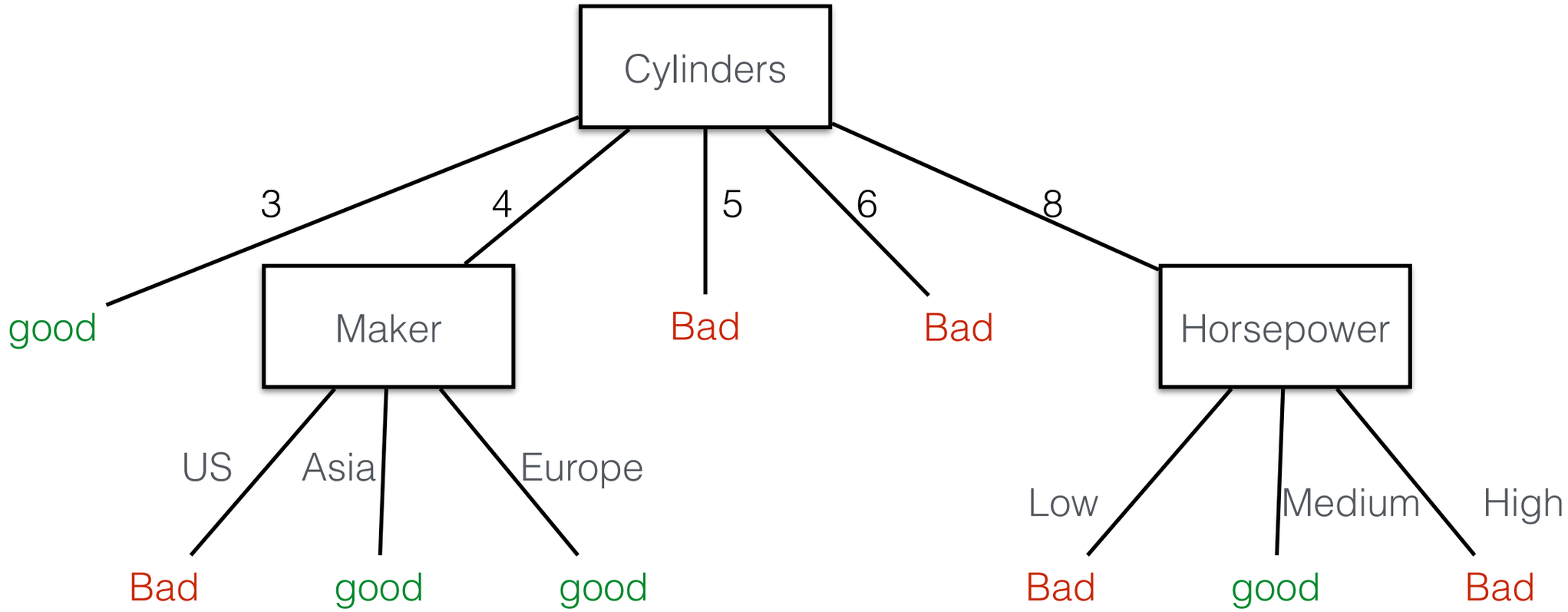
Input features: {Cylinders, Displacement, Horsepower, Weight, Acceleration, Model Year, Maker (country of origin)}

Output classes: {good milage, poor milage} (in miles per gallon)

As usual, you will be given a training set and your goal is to infer whether an engine with new input features will have poor or good milage

Decision Trees: Example

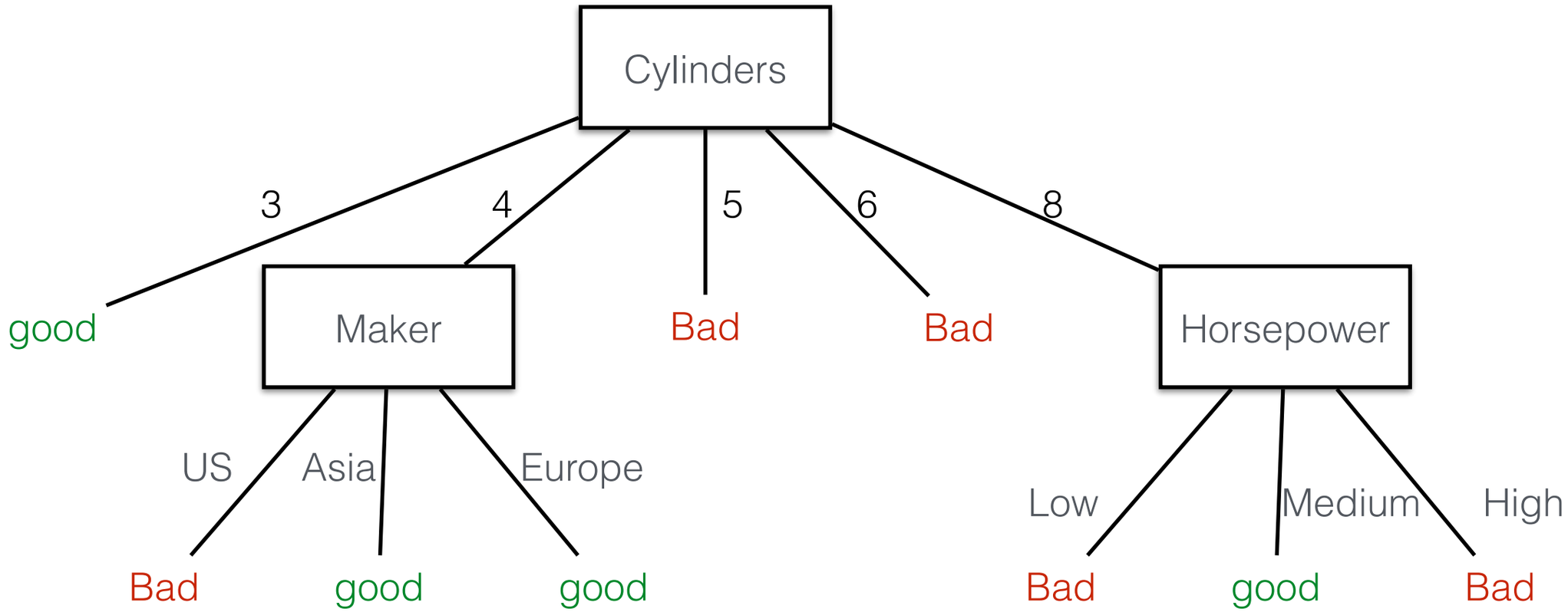
A typical decision tree would look like:



Slide taken from David Sontag

Decision Trees: Example

A typical decision tree would look like:



Nodes: each internal node tests a single attribute

Branches: we have one branch for every value of the attribute

Leaves: Each leaf node assigns a class to the examples belonging to it

To classify an input x we traverse the tree from the root to the leaves and assign the class y corresponding to the leaf node to which the input falls into

We will discuss the case of continuous attributes later

Slide taken from David Sontag

Decision Trees: Function Class

What functions can a decision tree represent?

Can represent any function of the input attributes!

They can encode any Boolean function

Proof

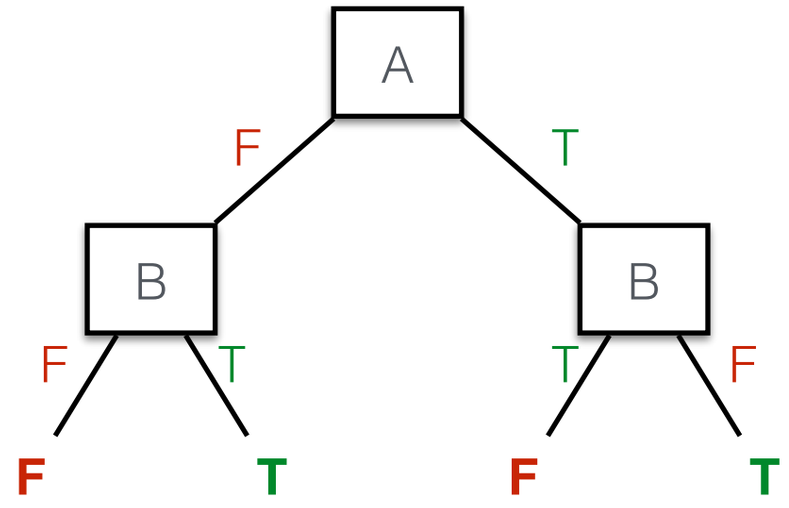
Given a truth table for a function

Construct a path in the tree for each row of the table

Given a row as input, follow that path to the desired leaf node (the output)

In order to represent functions the trees could potentially be exponentially large

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F



Slide taken from David Sontag

Decision Trees: What Makes a Good Decision Tree?

Smaller tress are better

Following the principle of Occam's razor

Prevents overfitting

A decision tree may be human readable but may not use human logic

Learning the simplest and smallest decision tree is an NP-complete problem

We resort to "greedy algorithms" to construct the trees

Greedy algorithms can usually get us to good decision trees

Overview of Greedy Algorithm

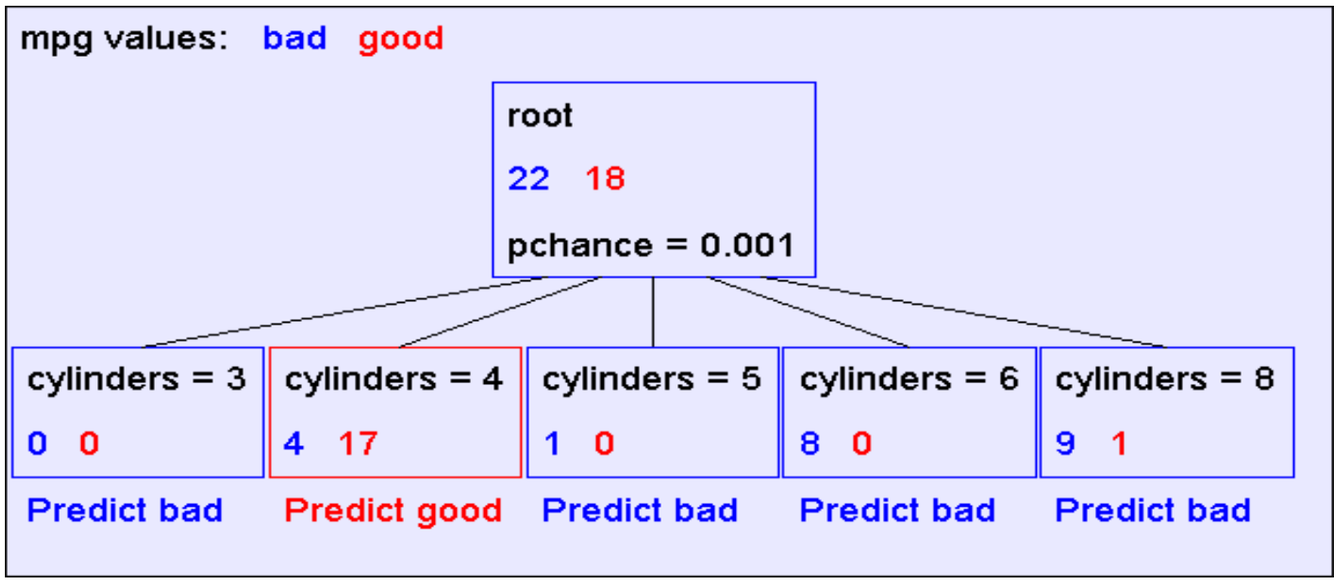
Start with an empty tree

Select the best possible variable to split on and generate child nodes: one node for every value of the variable

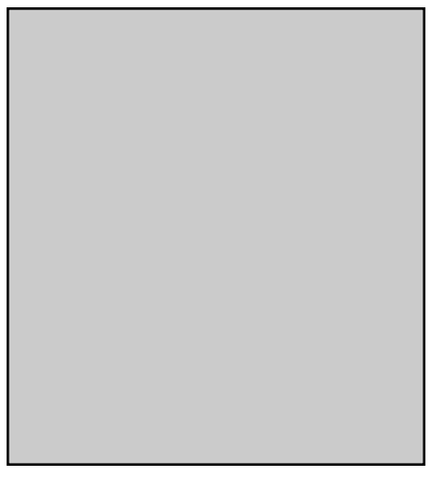
Partition samples according to their values and assign them to the appropriate child node

Recurse: repeat for each child node until all samples associated with a node are either all positive or all negative

Example of Greedy Algorithm



Take the Original Dataset..



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

Records in which cylinders = 4

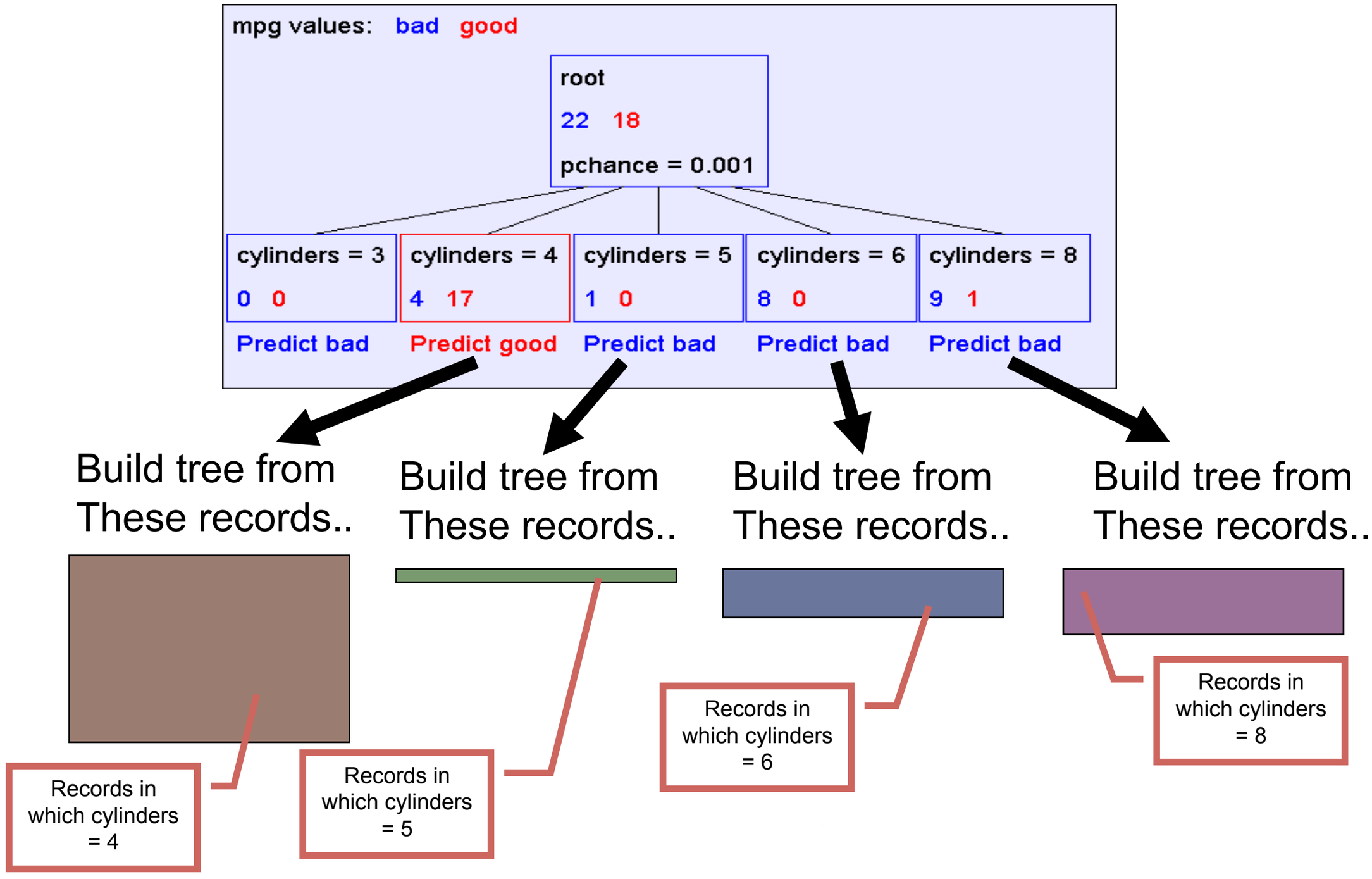
Records in which cylinders = 5

Records in which cylinders = 6

Records in which cylinders = 8

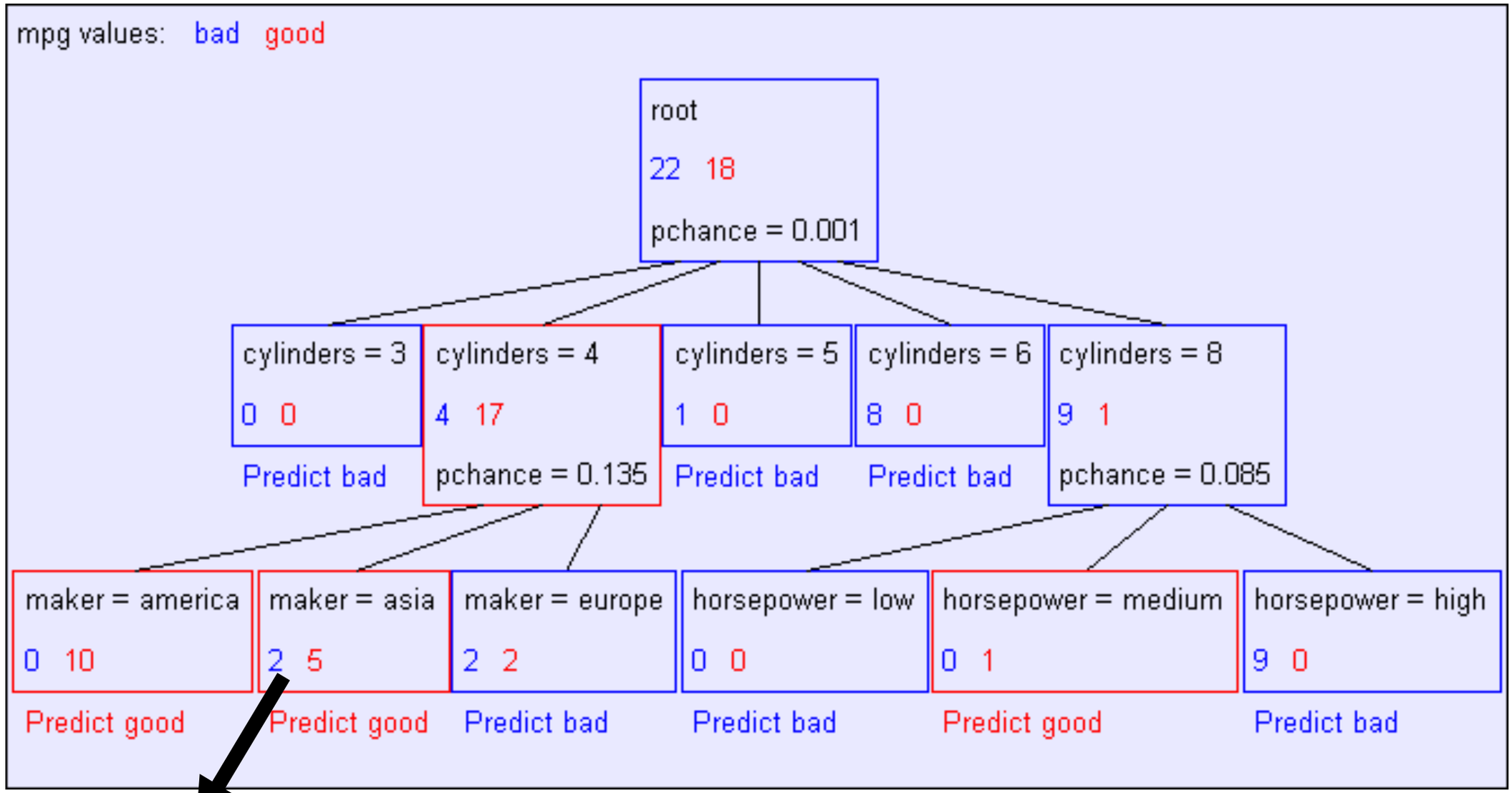
Slide taken from David Sontag

Example of Greedy Algorithm



Slide taken from David Sontag

Example of Greedy Algorithm

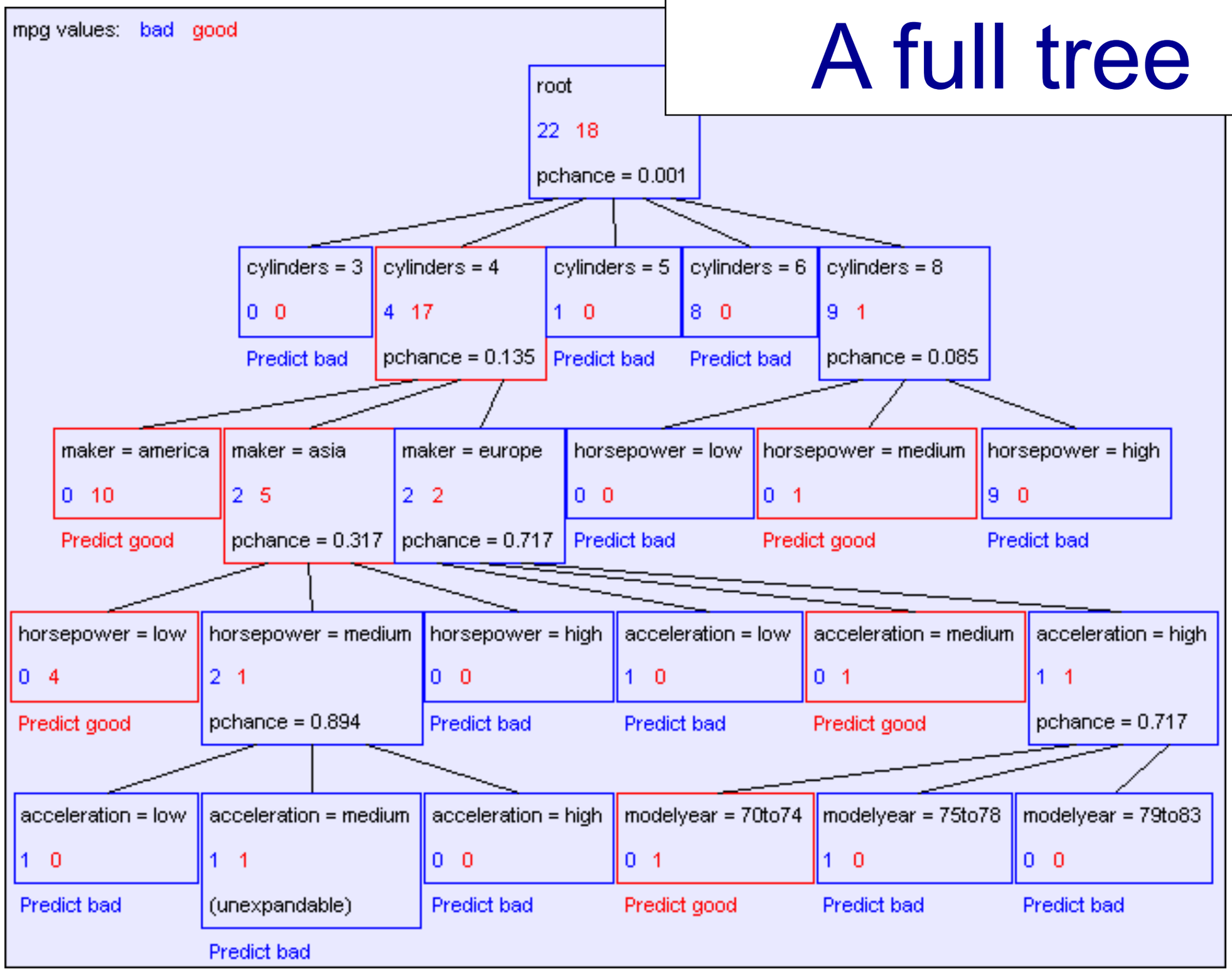


Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

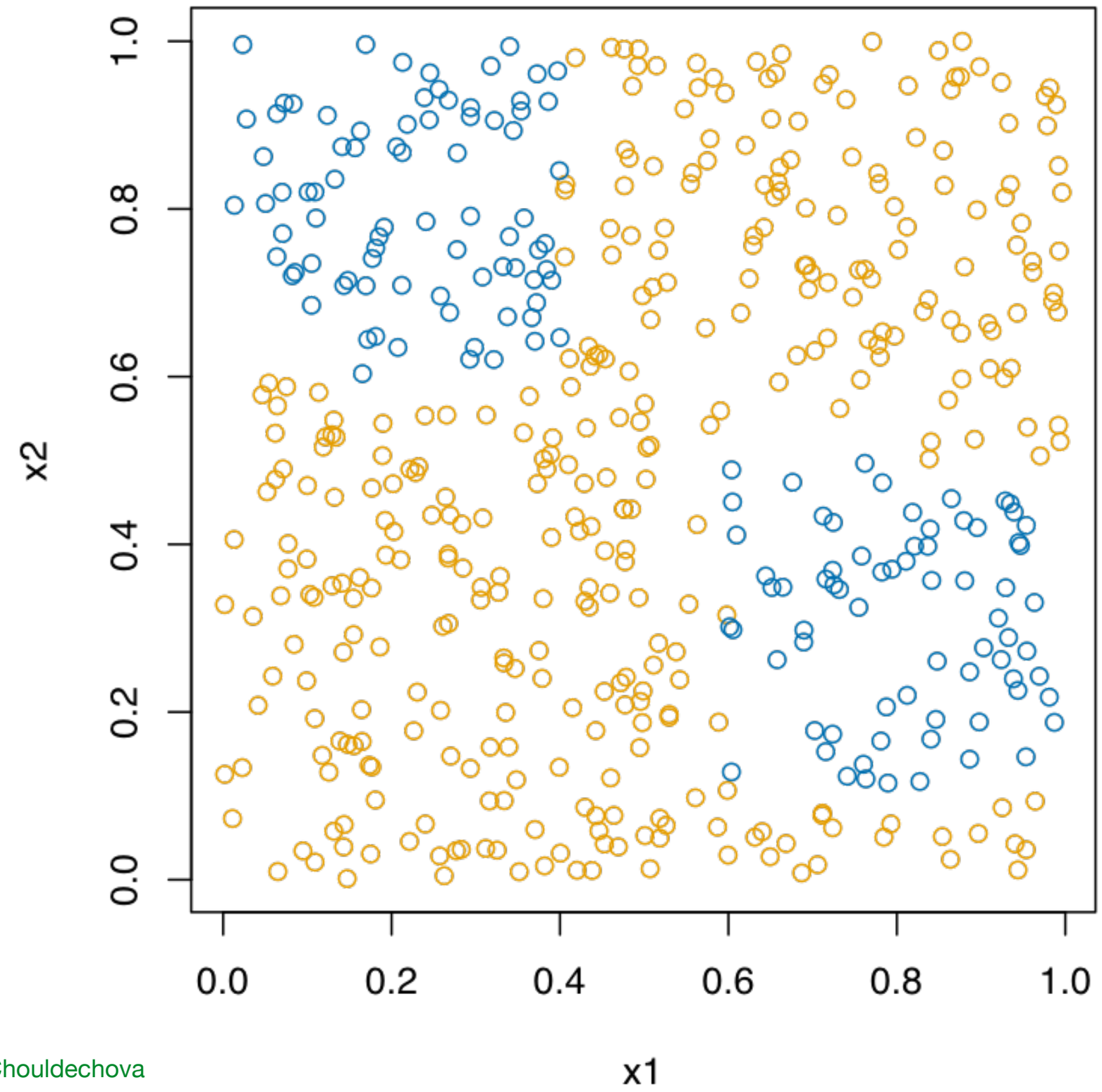
Slide taken from David Sontag

Example of Greedy Algorithm



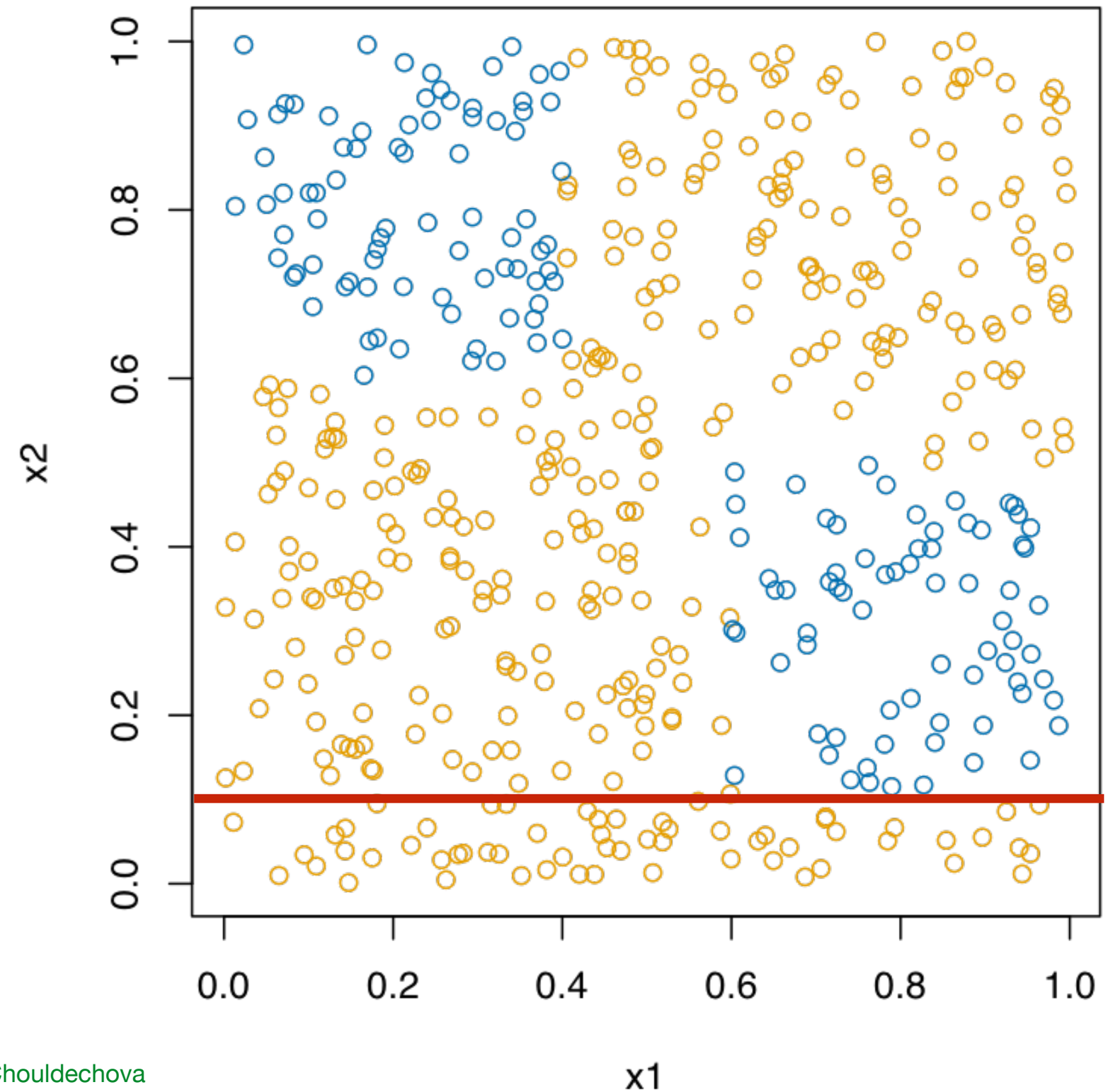
Slide taken from David Sontag

Another view of Decision Tree Greedy Algorithm



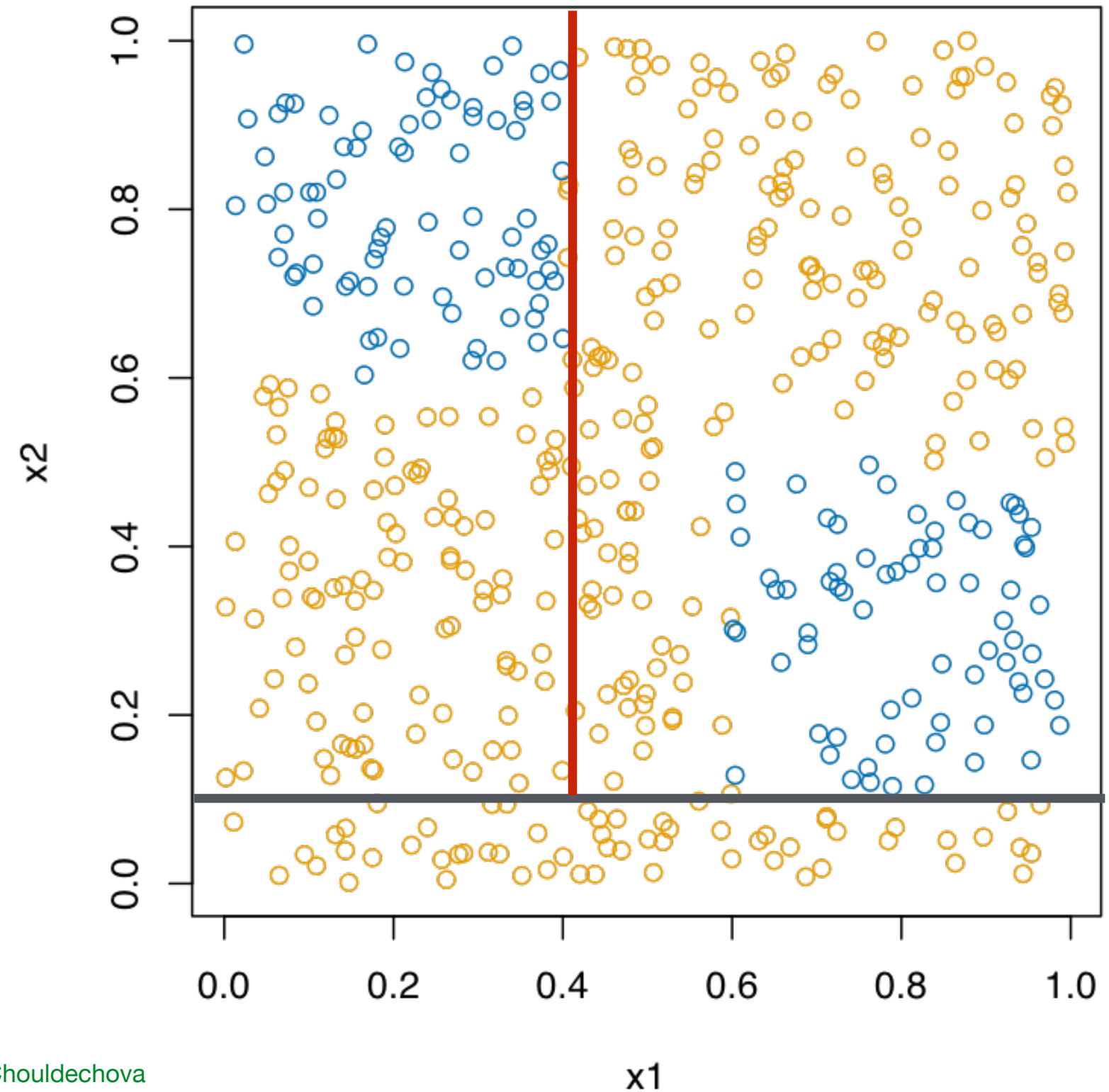
Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



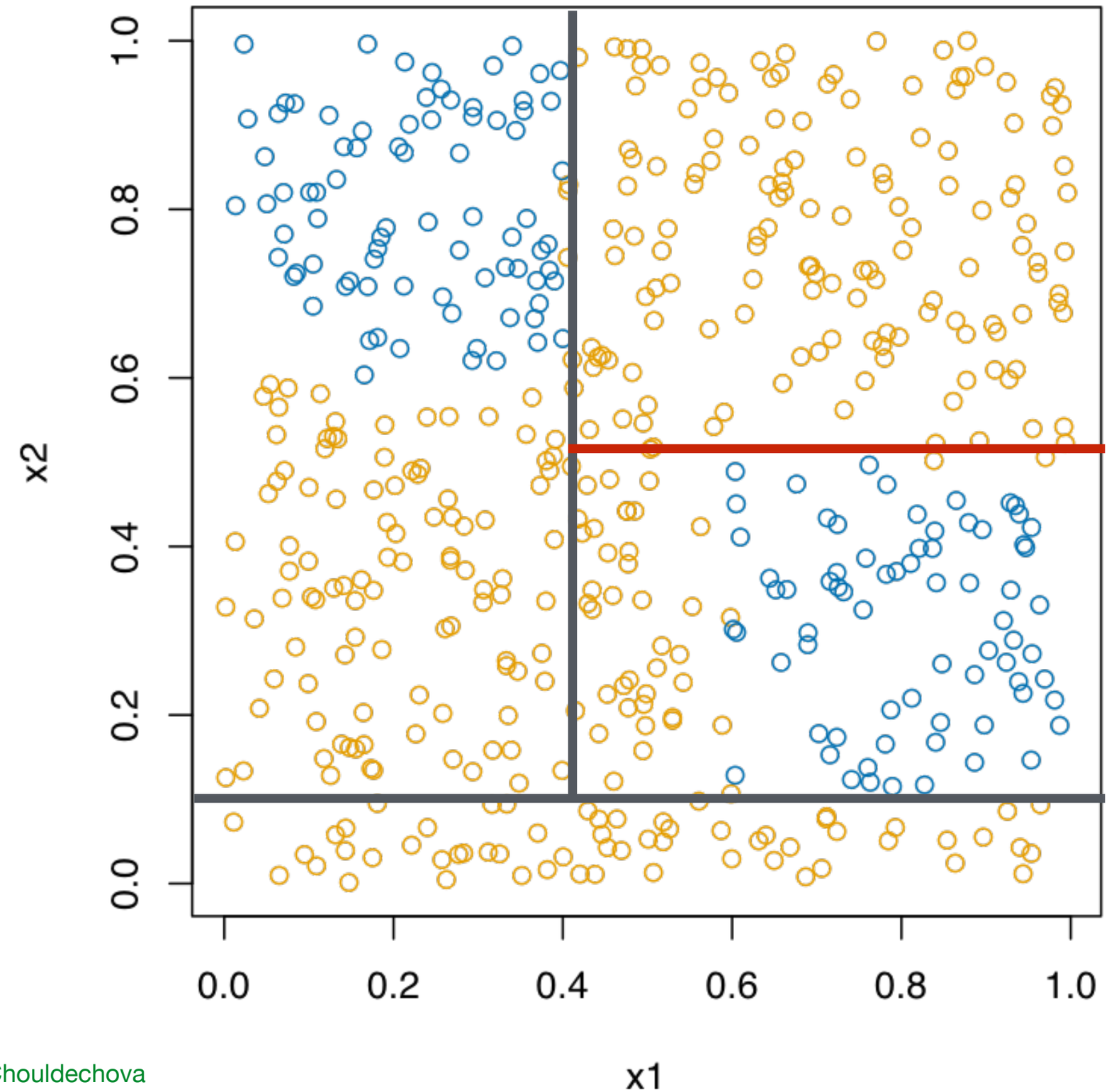
Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



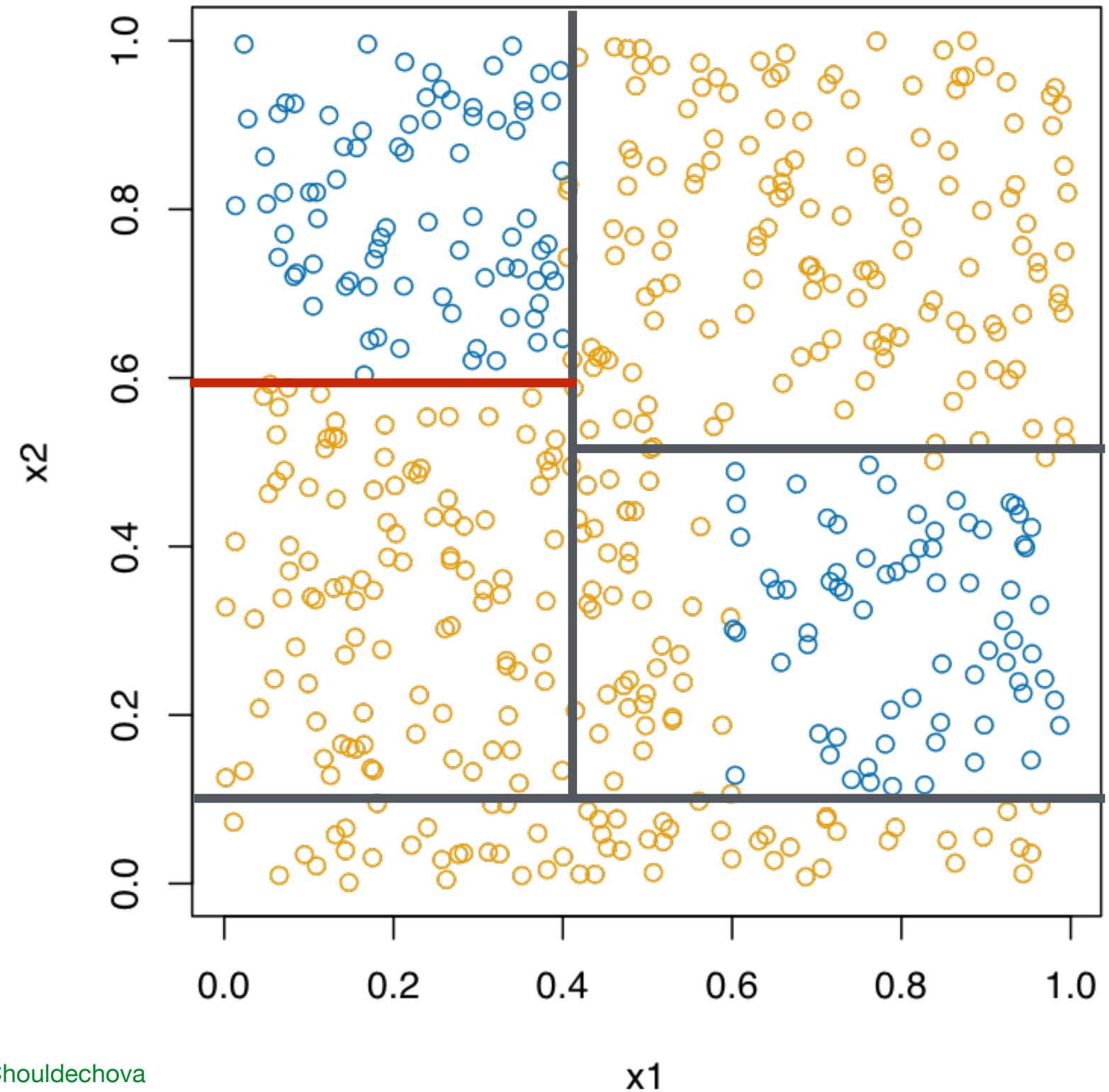
Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



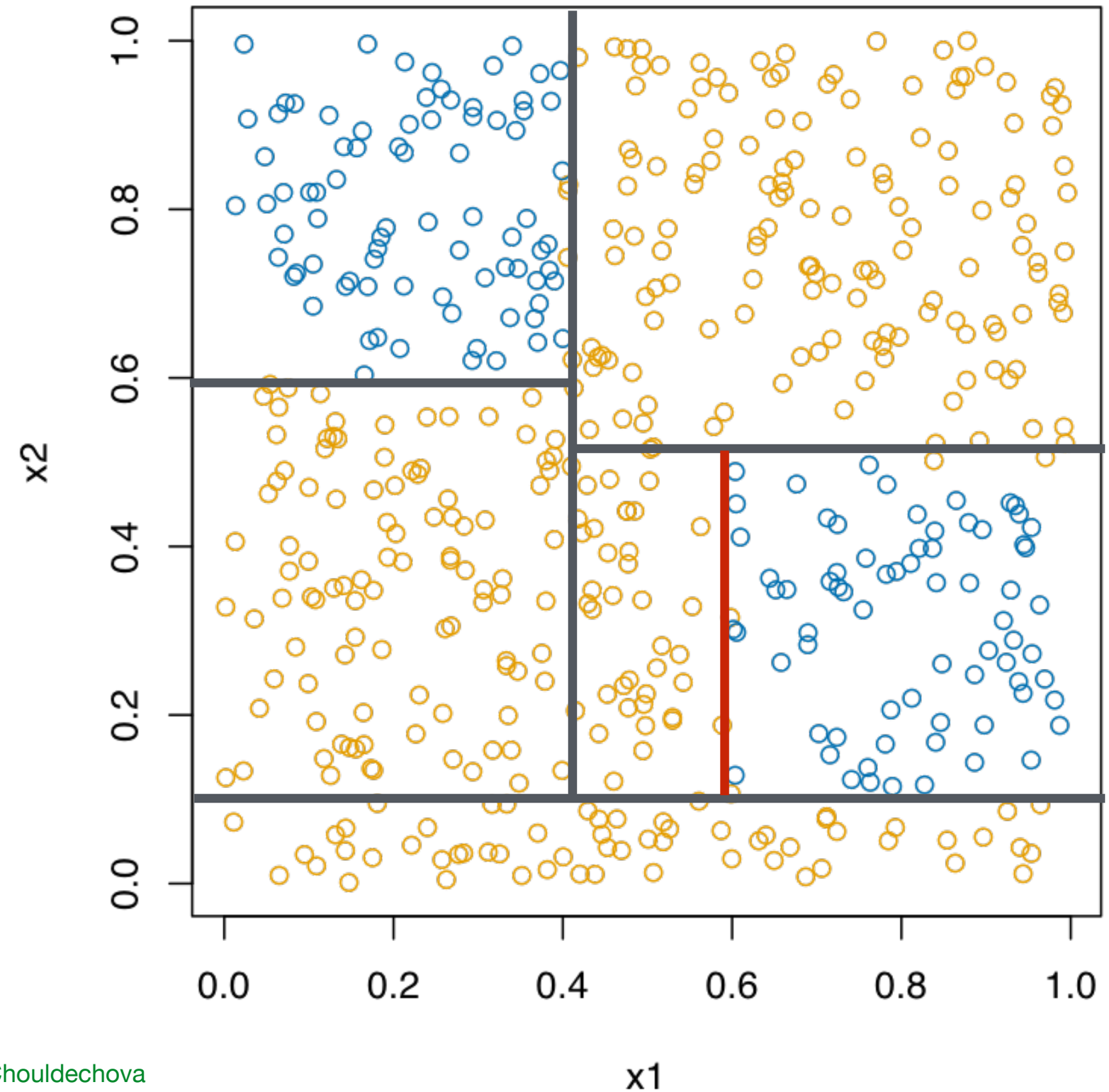
Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



Slide taken from Alexandra Chouldechova

Overview of Greedy Algorithm

What variable to split on?

Start with an empty tree

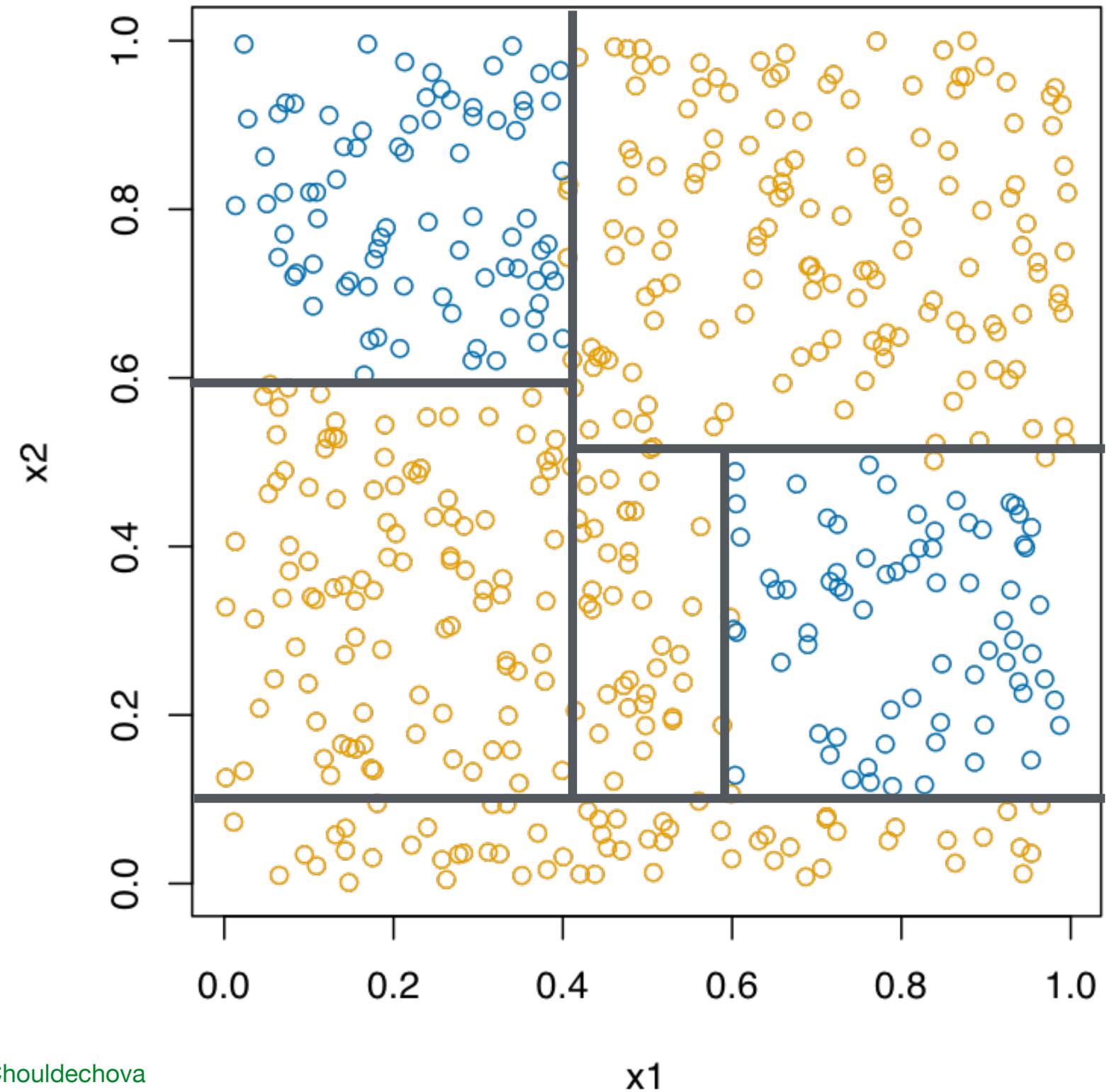
Select the best possible variable to split on and generate child nodes: one node for every value of the variable

Partition samples according to their values and assign them to the appropriate child node

Recurse: repeat for each child node until all samples associated with a node are either all positive or all negative

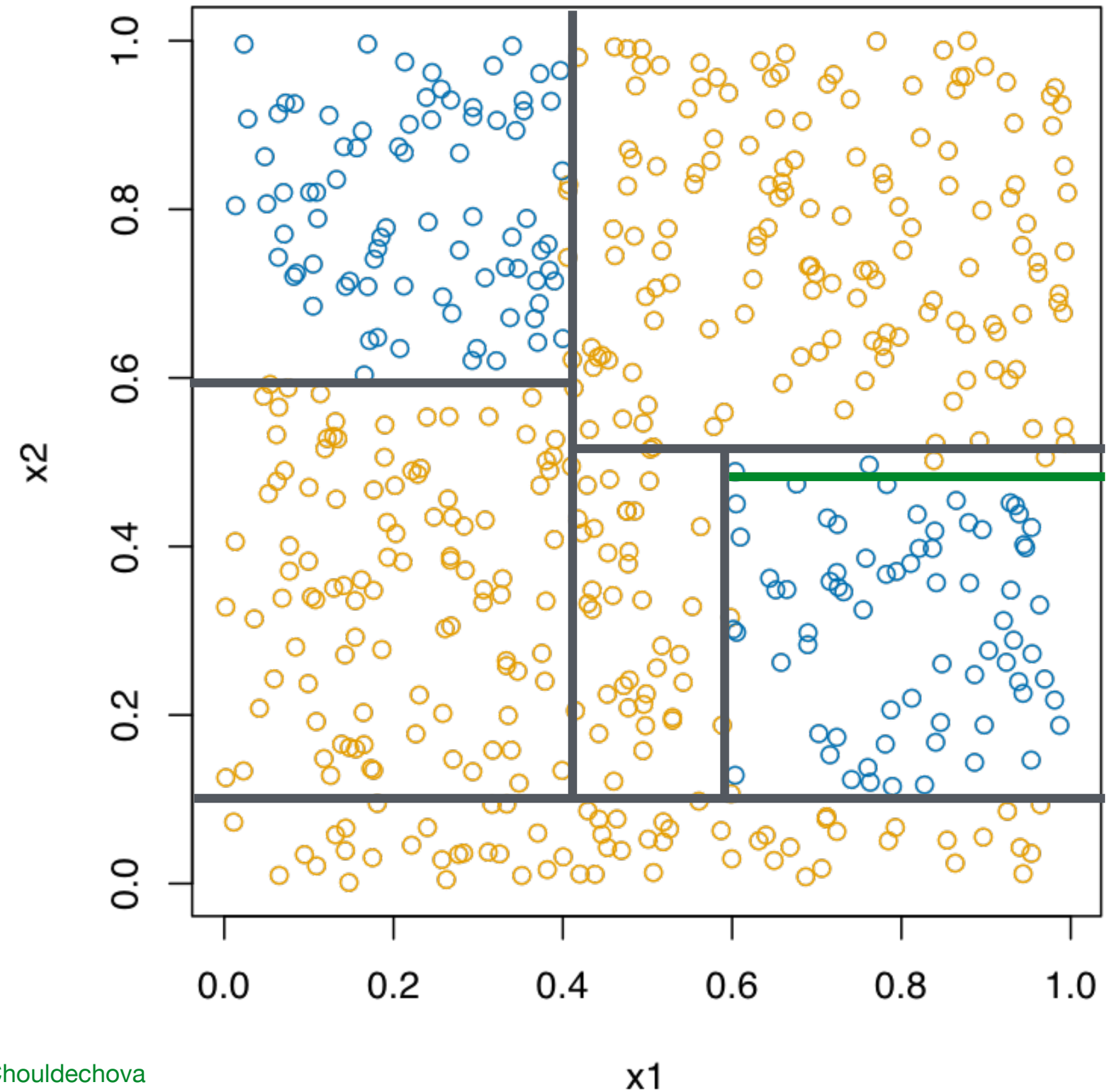
When to stop?

Another view of Decision Tree Greedy Algorithm



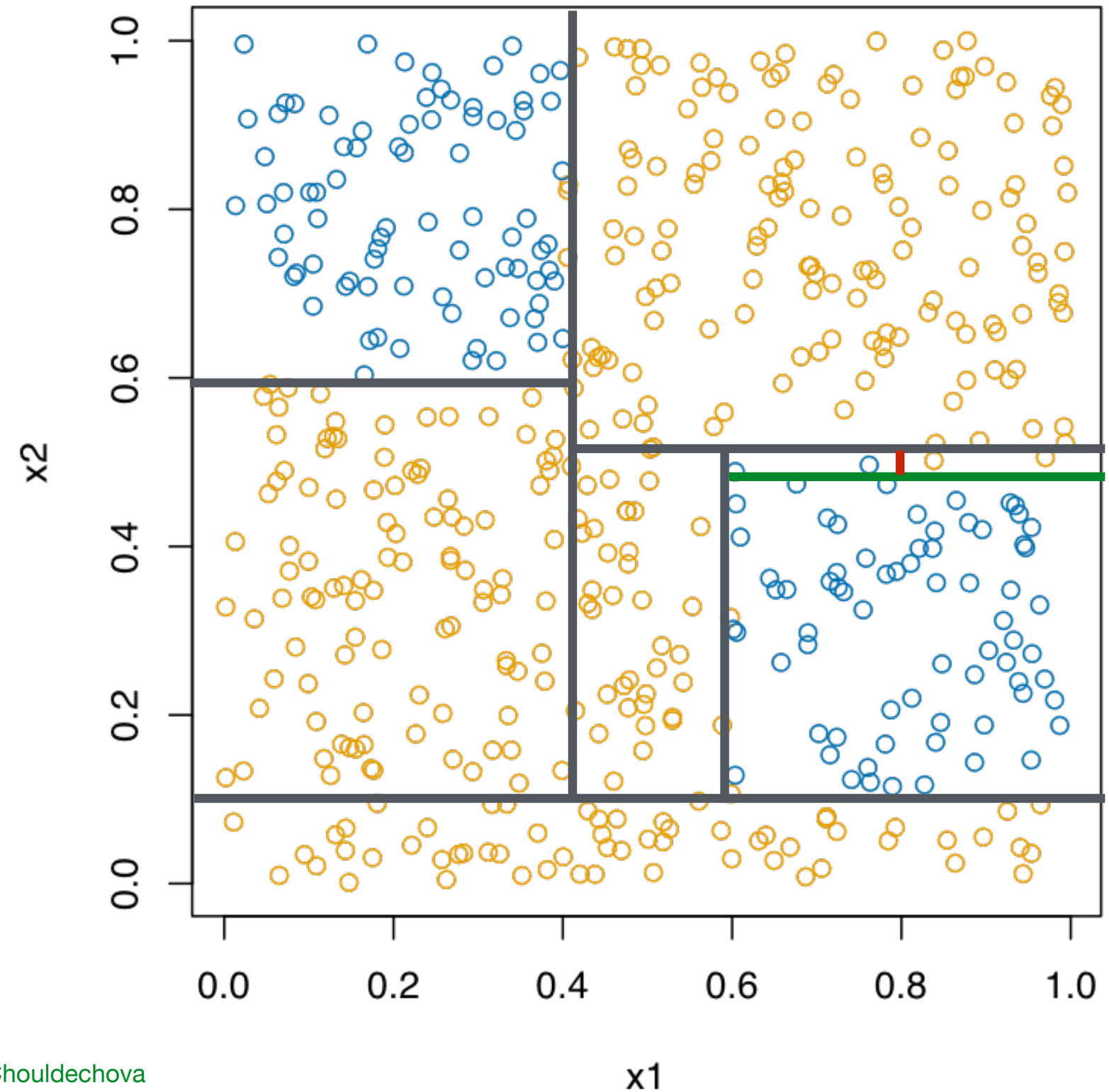
Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



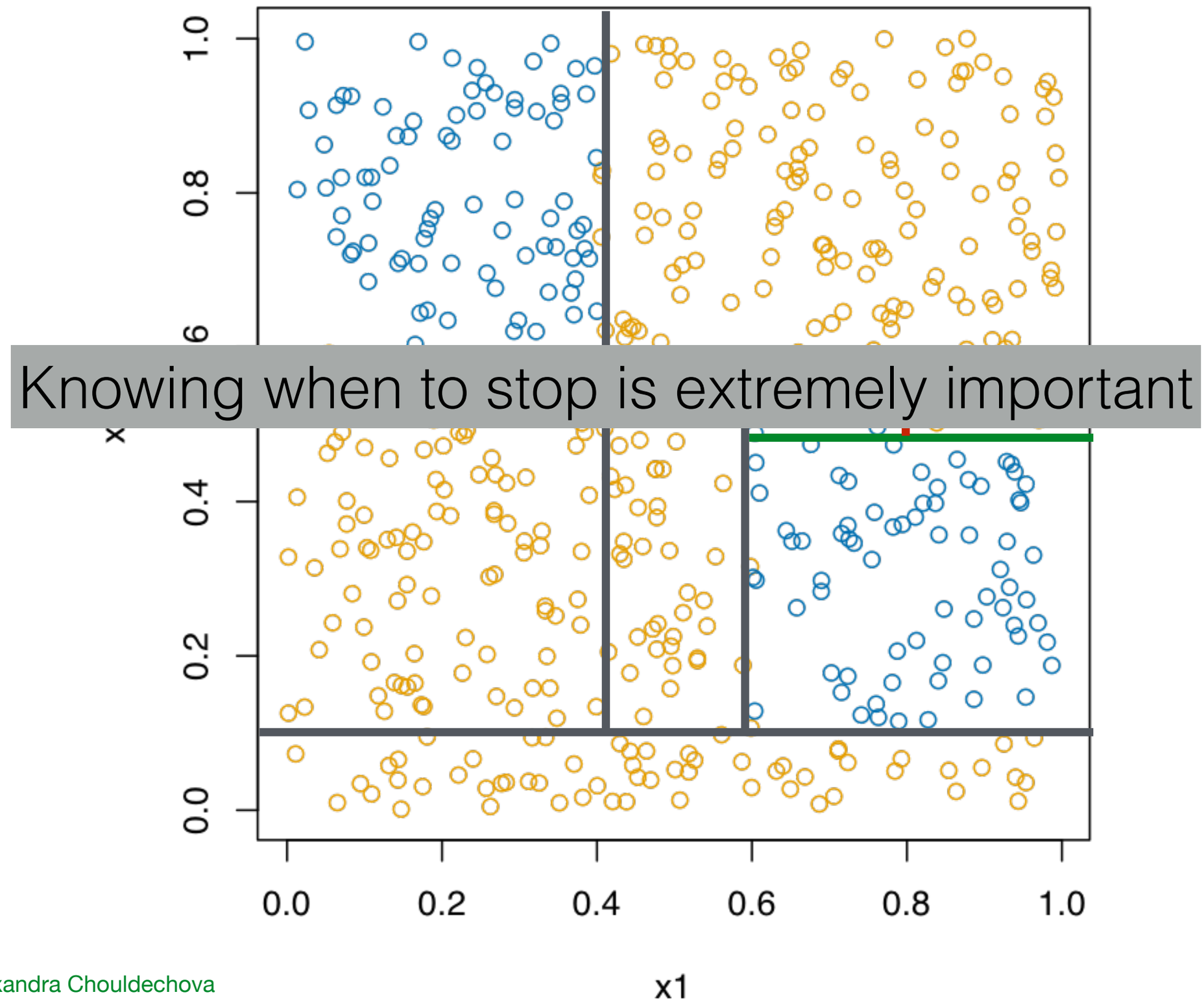
Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



Slide taken from Alexandra Chouldechova

Another view of Decision Tree Greedy Algorithm



Knowing when to stop is extremely important

Slide taken from Alexandra Chouldechova

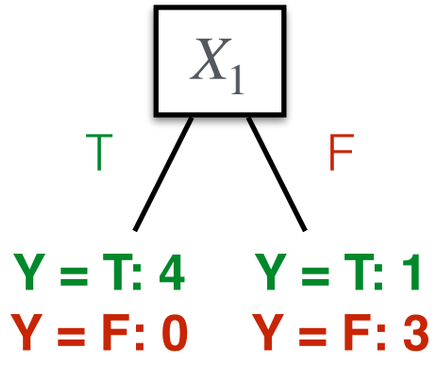
How to Choose the Variable to Split?

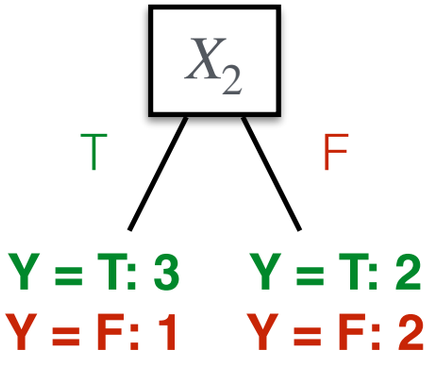
A split is good if it provides us more certainty about the classification after the split

Example of a really good fit: if after the split all positive examples go to one node and the negative examples to the other node

A really bad fit: gives us a uniform distribution over the classes

What about the distributions in between?





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

Select the variable that is most informative about the labels

Slide taken from David Sontag

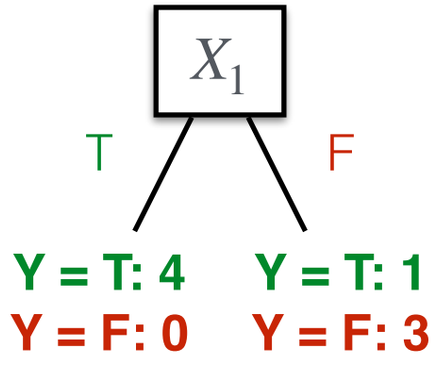
How to Choose the Variable to Split?

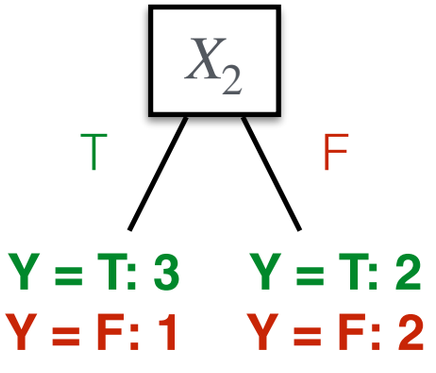
A split is good if it provides us more certainty about the classification after the split

Example of a really good fit: if after the split all positive examples go to one node and the negative examples to the other node

A really bad fit: gives us a uniform distribution over the classes

What about the distributions in between?





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

Select the variable that is most informative about the labels

All of this can be formalized using concepts from Information Theory

Entropy

Conditional Entropy

Information Gain

Slide taken from David Sontag

Entropy

Entropy $H(X)$ of a random variable X is defined as

$$H(X) = -\sum_{i=1}^{k} P(X = x_i) \cdot \log_2 P(X = x_i)$$

More uncertainty implies more entropy

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

High Entropy

X is from a uniform like distribution

Histogram is flat

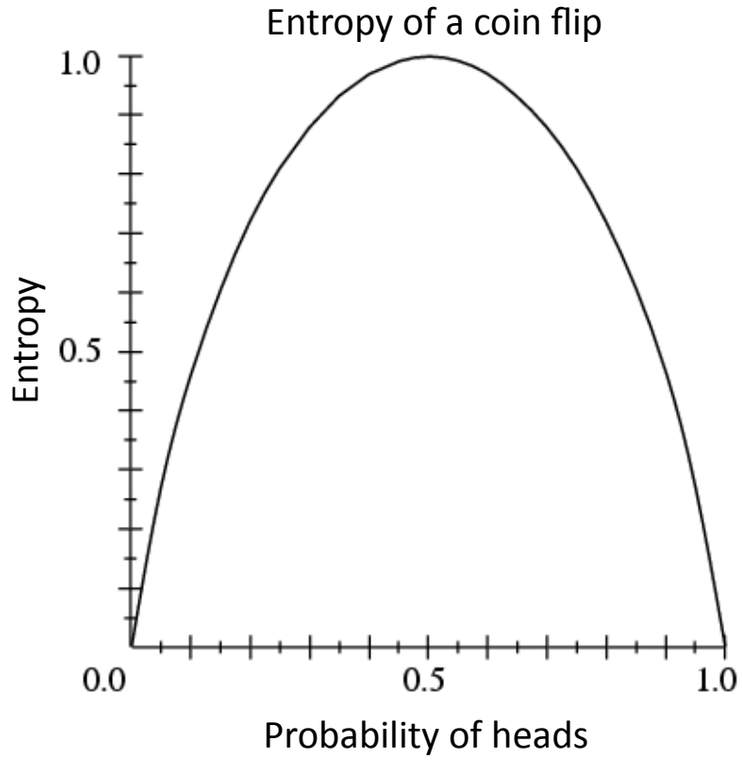
Values sampled from it are less predictable

Low Entropy

X is from a distribution which has peaks and valleys

Histogram has many lows and highs

Values sampled from it are more predictable



Slide taken from David Sontag

Entropy

Entropy $H(X)$ of a random variable X is defined as

$$H(X) = -\sum_{i=1}^{k} P(X = x_i) \cdot \log_2 P(X = x_i)$$

Example

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

$$P(Y = T) = 5/6$$

$$P(Y = F) = 1/6$$

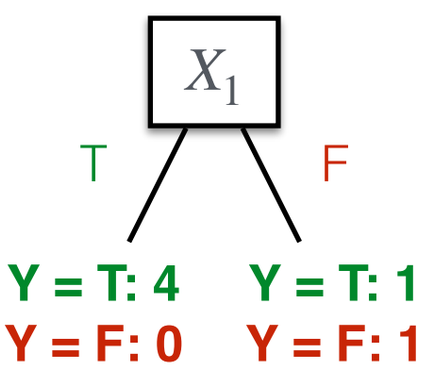
$$H(Y) = -5/6 \log_2(5/6) - 1/6 \log_2(1/6)$$
$$= 0.65$$

Slide taken from David Sontag

Conditional Entropy

Conditional entropy $H(Y|X)$ of a random variable Y conditioned on the random variable X is defined as

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



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

$$P(X = T) = 4/6$$

$$P(X = F) = 2/6$$

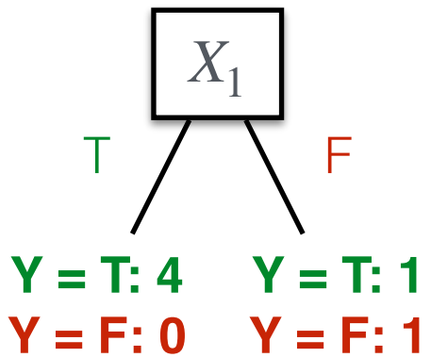
$$H(Y) = -4/6 \cdot (1 \log_2 1 + 0 \log_2 0) - 2/6 \cdot (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$$
$$= 0.33$$

Slide taken from David Sontag

Information Gain

The Information Gain $IG(Y)$ is defines as the decrease in the entropy after splitting

$$IG(Y) = H(Y) - H(Y|X)$$



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

$$IG(X) = 0.65 - 0.33$$

We prefer to split when the information gain is greater than zero

Ideally we will choose a variable that maximizes the information gain

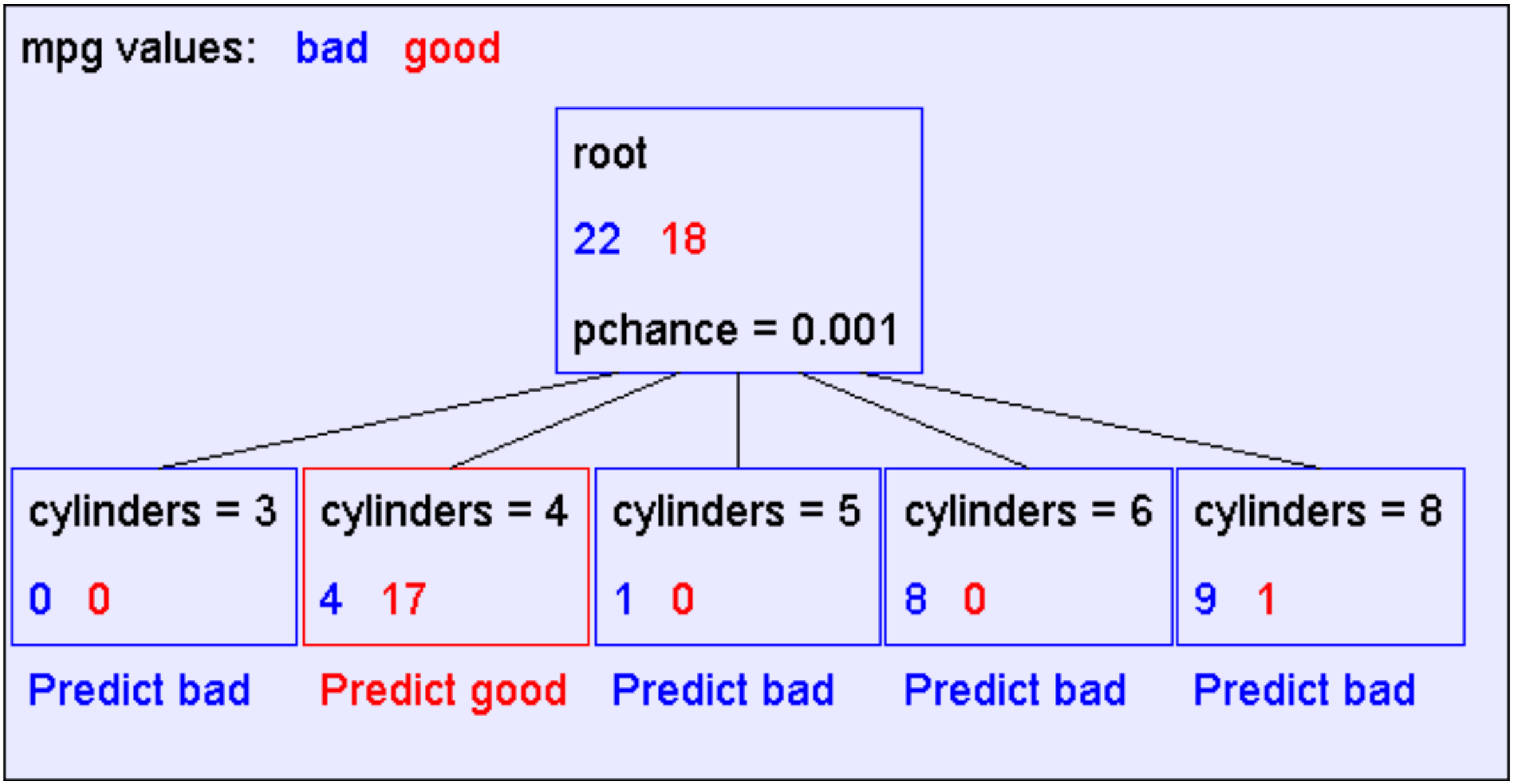
$$\arg\max_i IG(X_i) = \arg\max_i H(Y) - H(Y|X_i)$$

Slide taken from David Sontag

When to Stop?

The first split looks good but when do we stop splitting?

Remember we are biased towards finding the simple decision trees

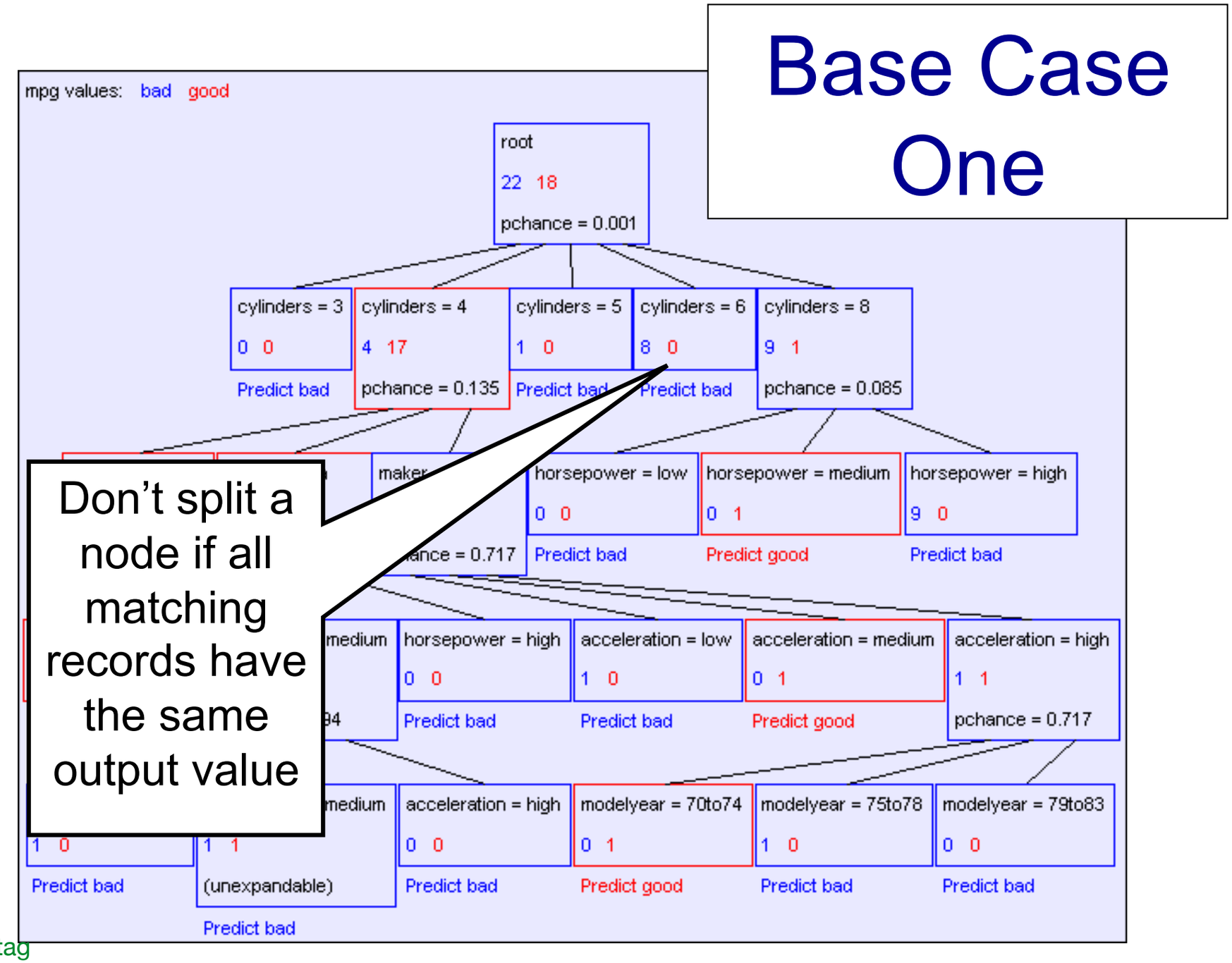


Slide taken from David Sontag

When to Stop?

The first split looks good but when do we stop splitting?

Remember we are biased towards finding the simple decision trees

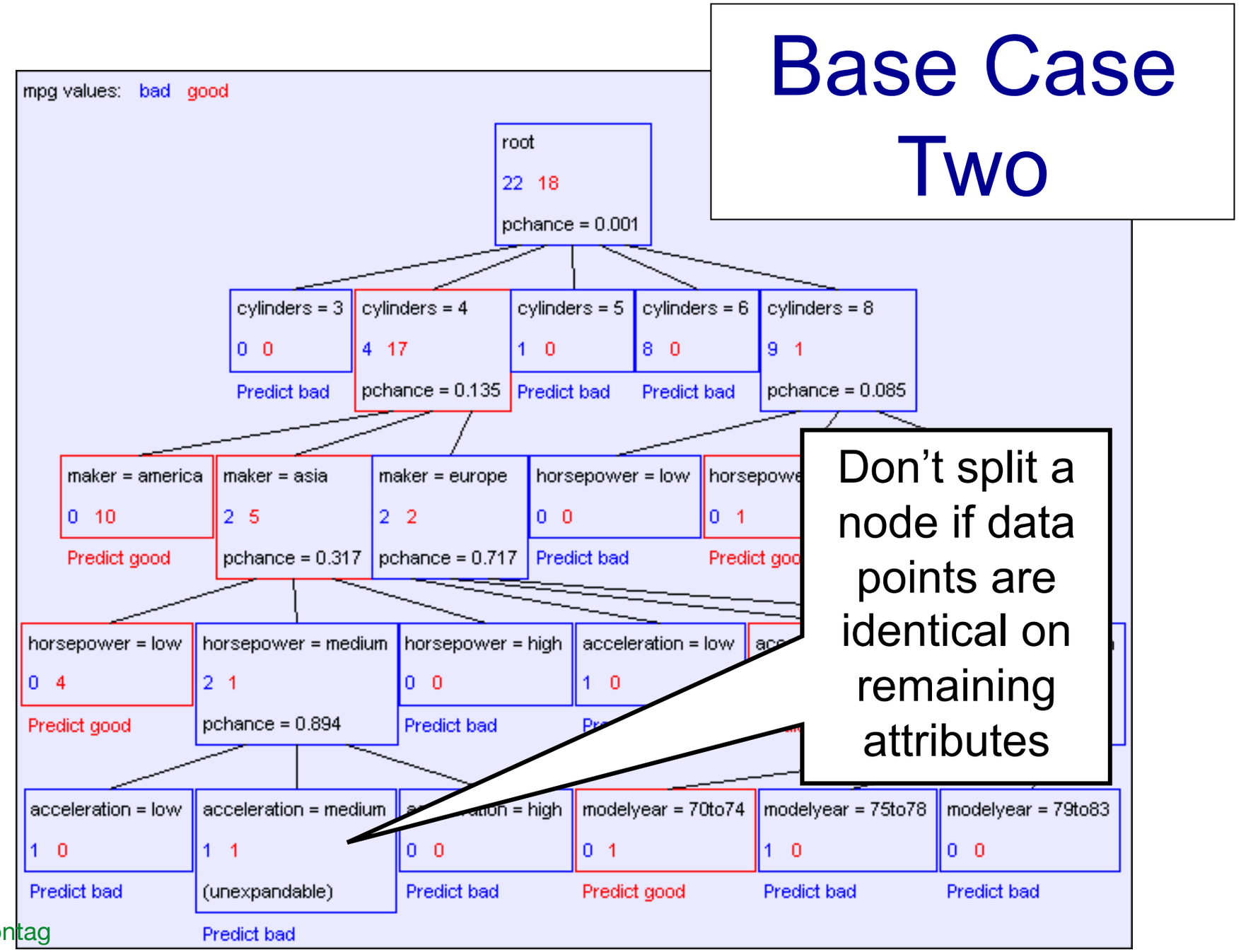


Slide taken from David Sontag

When to Stop?

The first split looks good but when do we stop splitting?

Remember we are biased towards finding the simple decision trees



Slide taken from David Sontag

When to Stop?

The first split looks good but when do we stop splitting?

Remember we are biased towards finding the simple decision trees

How about the case when there is no information gain: $IG(Y) = 0$?

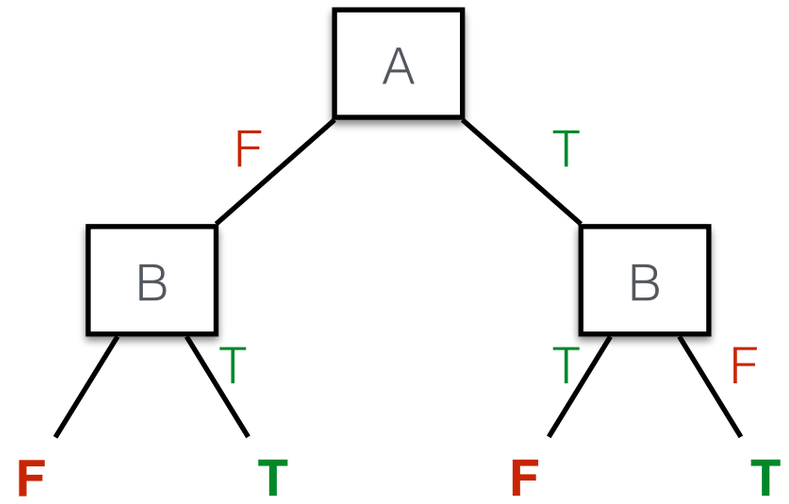
When to Stop?

The first split looks good but when do we stop splitting?

Remember we are biased towards finding the simple decision trees

How about the case when there is no information gain: $IG(Y) = 0$?

A	B	Y
F	F	F
F	T	T
T	F	T
T	T	F



$$IG(Y)_A = H(Y) - H(Y|A) = 0$$
$$IG(Y)_B = H(Y) - H(Y|B) = 0$$

Sometimes we need a little exploration

The general strategy is to over-split and then prune

Slide taken from David Sontag

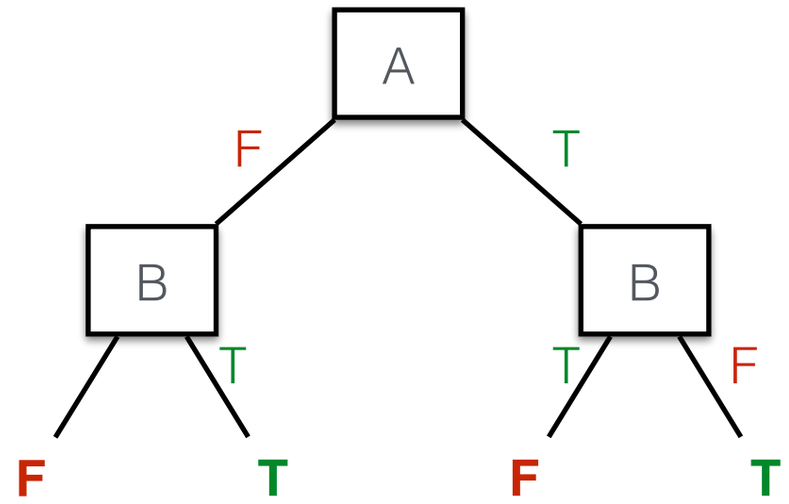
When to Stop?

The first split looks good but when do we stop splitting?

Remember we are biased towards finding the simple decision trees

How about the case when there is no information gain: $IG(Y) = 0$?

A	B	Y
F	F	F
F	T	T
T	F	T
T	T	F



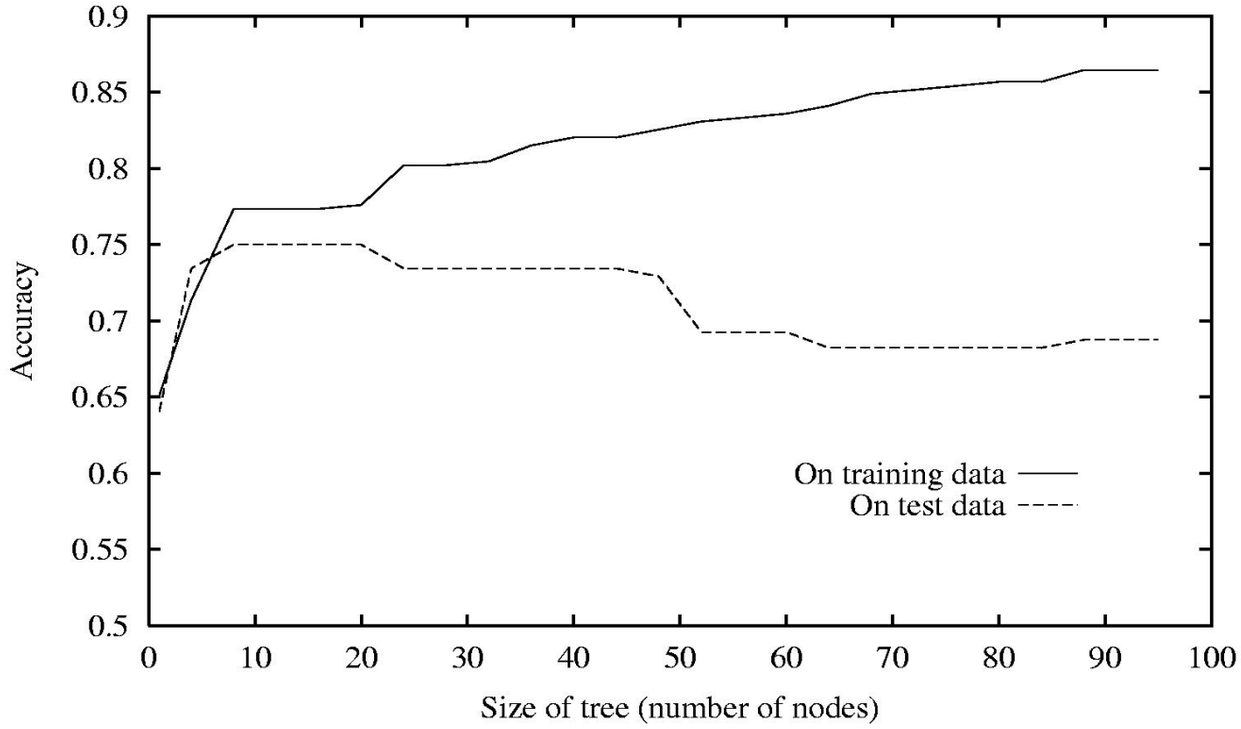
$$IG(Y)_A = H(Y) - H(Y|A) = 0$$
$$IG(Y)_B = H(Y) - H(Y|B) = 0$$

Sometimes we need a little exploration

The general strategy is to over-split and then prune

Slide taken from David Sontag

Choosing a Best Decision Tree



Decision trees will overfit!

Standard decision trees have no learning bias
Training error can always go to zero

They do have a lot of variance
Slight change in the training set can generate a completely different tree

Hence we must introduce some bias towards simpler trees

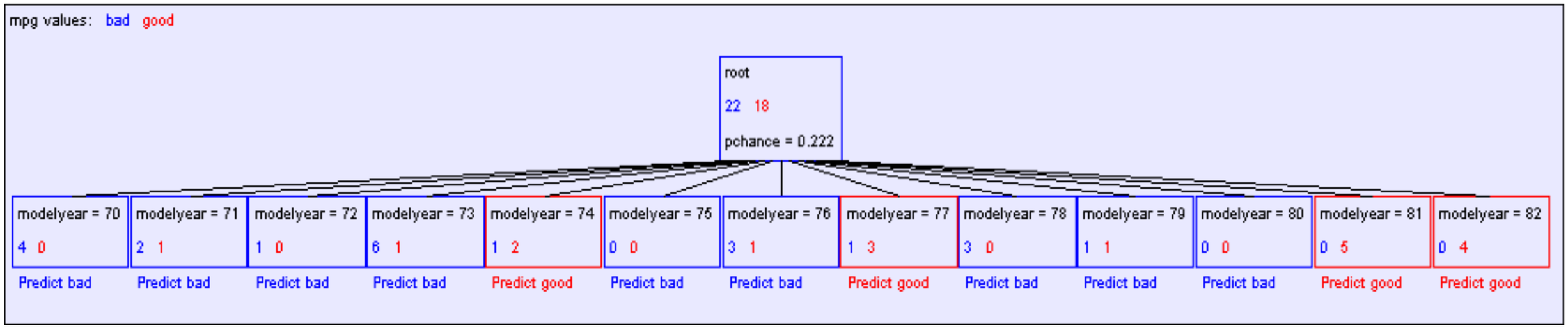
Many techniques exist for picking simpler trees
Upper bounding the depth of the tree
Lower bounding the number of samples per leaf node
Ensembling (Random Forests)

Slide taken from David Sontag

Handling Continuous Variables

Remember a branch from a node corresponds to a single value of the variable being used to split

How will we handle the case when the variable is continuous or discrete variables with large number of possibilities?



This is a bad idea as this high branching factor will overfit any data!

Slide taken from David Sontag

Handling Continuous Variables

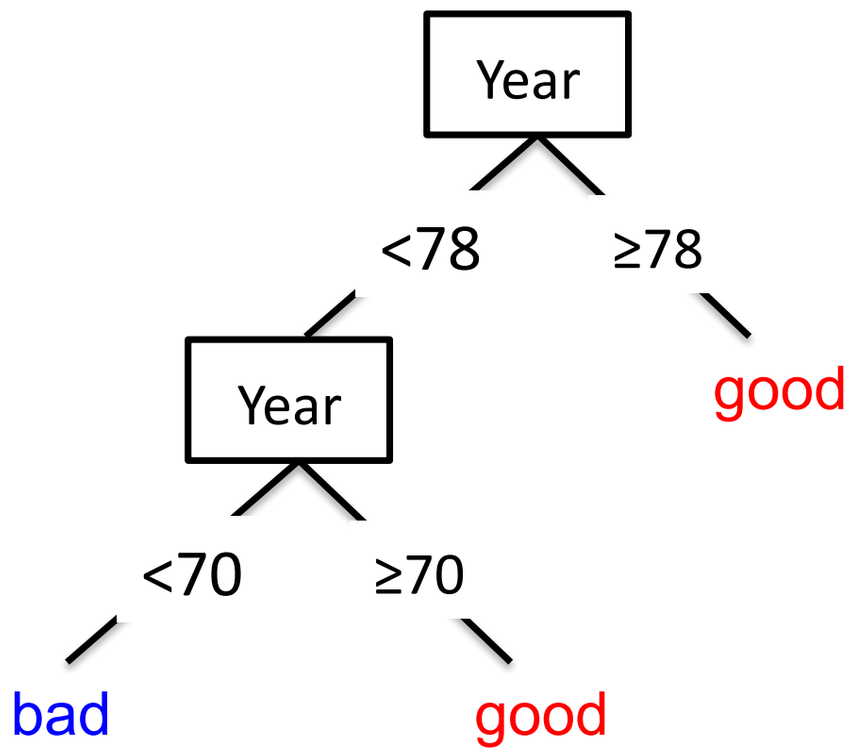
Decide on a threshold t and split based on the value of the variable X compared to the threshold

Branch 1: $X < t$

Branch 2: $X \geq t$

If we want a finer grained thresholding we can make a slight modification

Allow repeated splits on the same variable along a path with different thresholds



Slide taken from David Sontag

Handling Continuous Variables

Decide on a threshold t and split based on the value of the variable X compared to the threshold

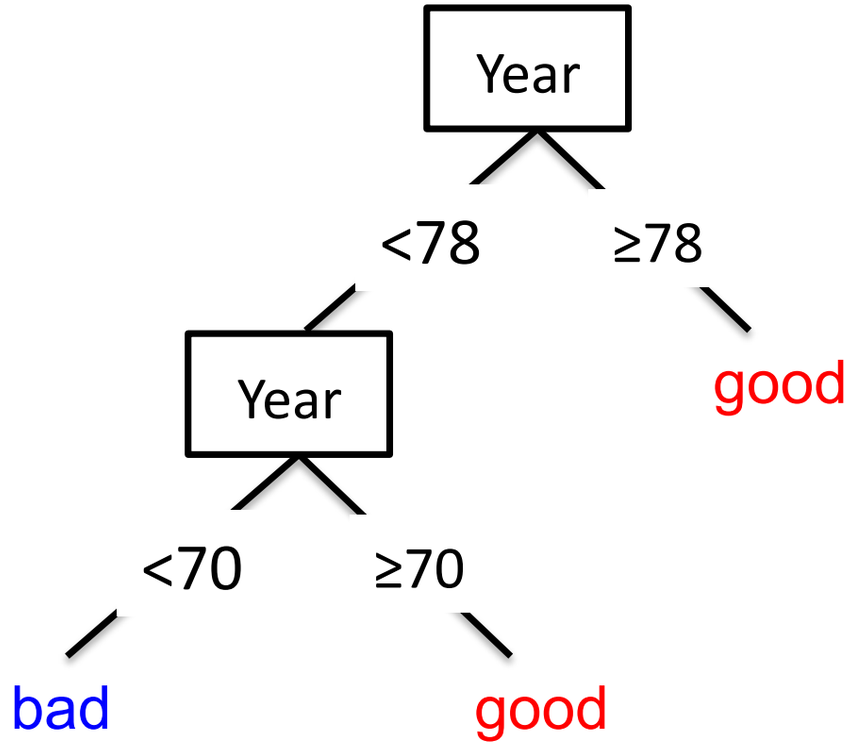
Branch 1: $X < t$

Branch 2: $X \geq t$

Decision Stumps

If we want a finer grained thresholding we can make a slight modification

Allow repeated splits on the same variable along a path with different thresholds



Slide taken from David Sontag

Handling Continuous Variables

Decide on a threshold t and split based on the value of the variable X compared to the threshold

Branch 1: $X < t$ Branch 2: $X \geq t$

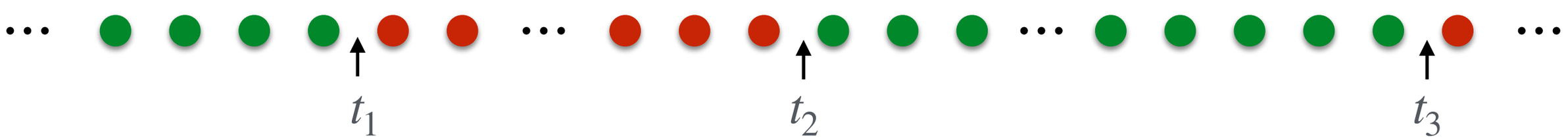
Searching through all possible values to get the best threshold seems hard!

But there's a way around it

Sort data according to X into $\{X_1, X_2, \ldots, X_m\}$

Consider split points of the form $X_i + (X_{i+1} - X_i)/2$

Moreover only the split points between examples of different classes matter. Hence the number of potential candidates are way smaller than the entire range



Slide taken from David Sontag

Handling Continuous Variables

Let X be a real valued variable on which to potentially split on with the threshold t

We need to compute $IG\left(Y \mid X \lessgtr t\right)$: the information gain for Y when testing if X is greater than or less than t

$$H\left(Y \mid X \lessgtr Y\right) = P(X < t) \cdot H(Y \mid X < t) + P(X \geq t) \cdot H(Y \mid X \geq t)$$

$$IG\left(Y \mid X \lessgtr t\right) = H(Y) - H\left(Y \mid X \lessgtr t\right)$$

$$IG^*\left(Y \mid X\right) = \max_t IG\left(Y \mid X \lessgtr t\right)$$

Use $IG^*\left(Y \mid X\right)$ for continuous variables

Decision Tree Takeaways

Decision trees are quite popular in the machine learning community, especially in practice

Easy to understand, implement and use

Computationally they are cheap

While they are human readable they may not necessarily use human logic

What variable to select on is chosen based on information gain

They can also be used for regression purposes: typically take the average of examples in the bin

Decision trees will overfit!

Need to use tricks/heuristics to avoid overfitting (limit depth, threshold number of points in a leaf bin etc)

Use ensemble of trees (Random Forests)

Ensemble Learning

Ensemble Learning

Used with the motivation to reduce variance in models

Take multiple models trained on the same task and perform an average over predictions

More diverse the models the better it is

The idea is independent of the underlying model types

Its like a trick used in most ML solutions to real world problems

Particularly suited for decision tree classifiers

Bagging (Bootstrap Aggregating)

Boosting

Bayesian model averaging

End of Lecture 10