The week ahead

- **Quiz 8:** mean is 86% and average completion time 5min 16sec
- **Assignment 3 Early bird special due 11:59pm (midnight)** → 1 complete programming question
- **Quiz 9, Friday, Oct 23th 6am until Oct 24th 11:59am (noon)**
  - Decision trees

Coming up soon

- **Assignment 3 due Mon, Oct 26th, 11:59 pm (midnight)**
- **Touch-point 2**: deliverables due Mon, Oct 30th, live-event Wed, Nov 2nd
  - Single-slide presentation outlining progress highlights and current challenges
  - Three-minute pre-recorded presentation with your progress and current challenges
- **Project midpoint report due Nov 6th 11:59pm (midnight)**
  - GitHub page with the results you have achieved utilizing unsupervised learning

CS4641B Machine Learning

Lecture 17: Decision tree

Rodrigo Borela ▸ rborelav@gatech.edu

These slides are adopted from Polo Chau, Vivek Srikumar, and Chao Zhang and Mahdi Roozbahani

Recap: Naïve Bayes

Recap: Logistic regression

Outline

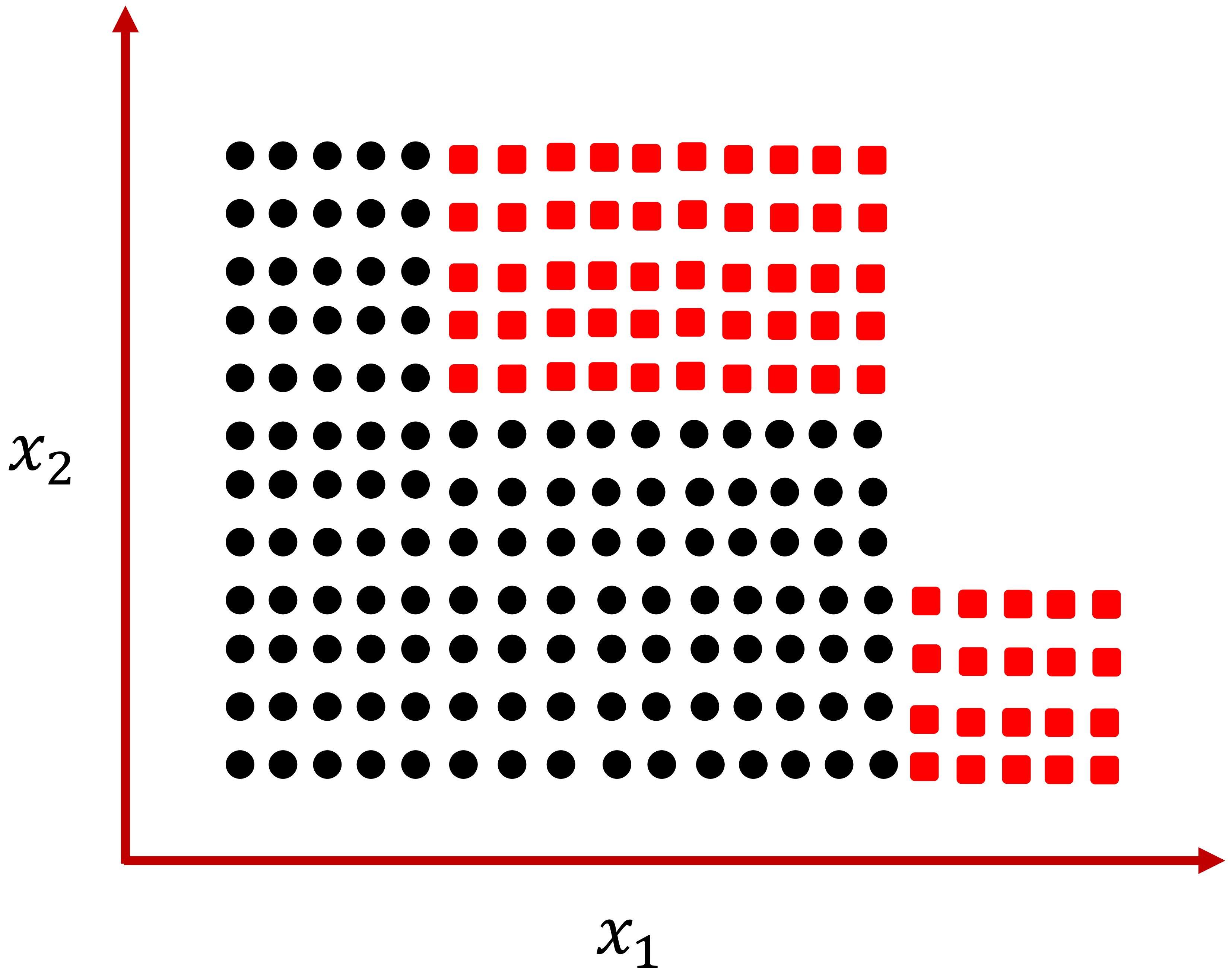
- Intuition
- Overview
- Learning a tree
- Algorithm

- *Complementary reading: Bishop PRML – Chapter 14, Section 14.4*

Outline

- Intuition
- Overview
- Learning a tree
- Algorithm

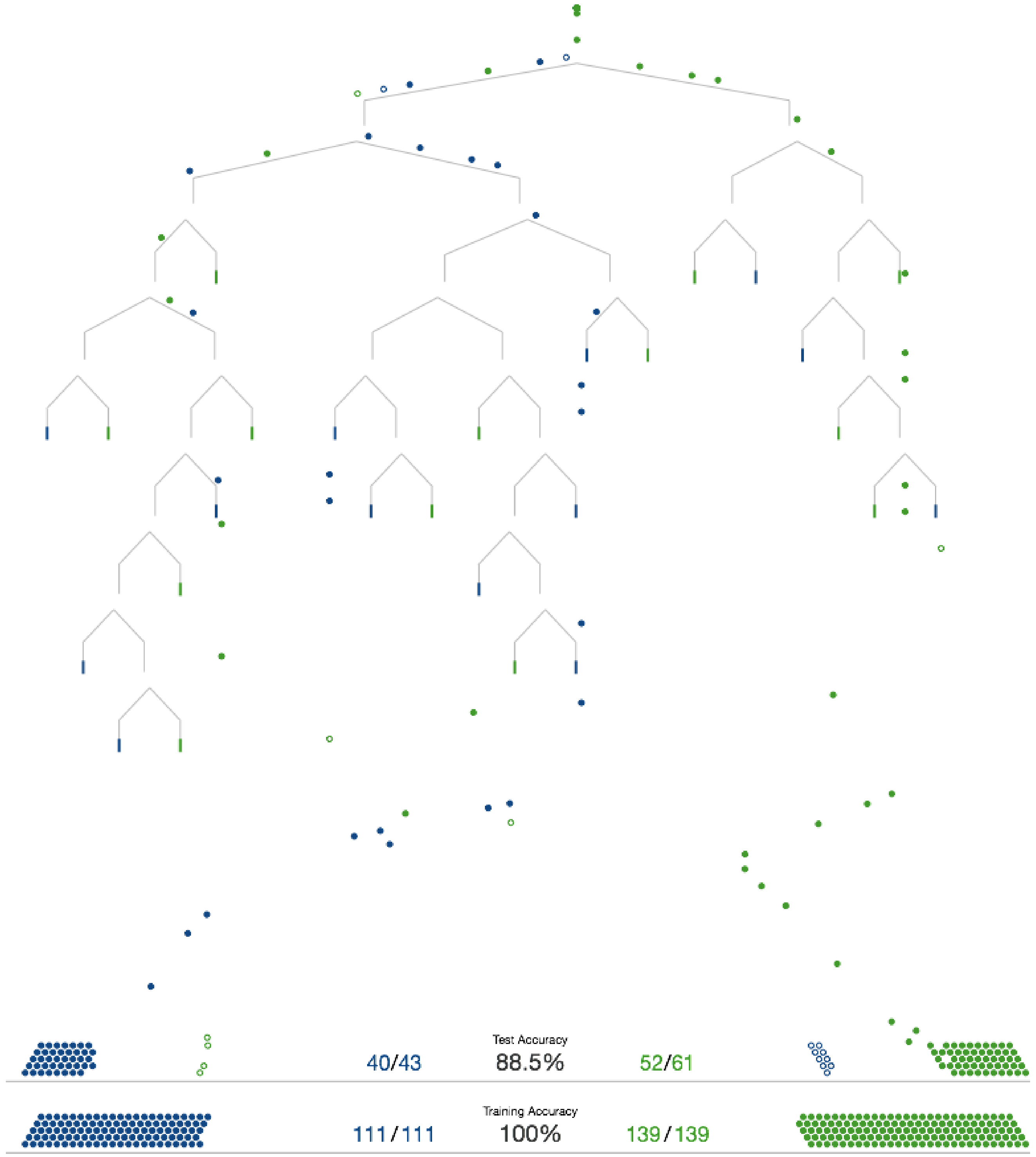
Decision trees: intuition



Outline

- Intuition
- Overview
- Learning a tree
- Algorithm

Visual introduction to decision tree



Building a tree to distinguish homes in New York from homes in San Francisco

Decision tree: example

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: Sunny, Overcast, Rainy

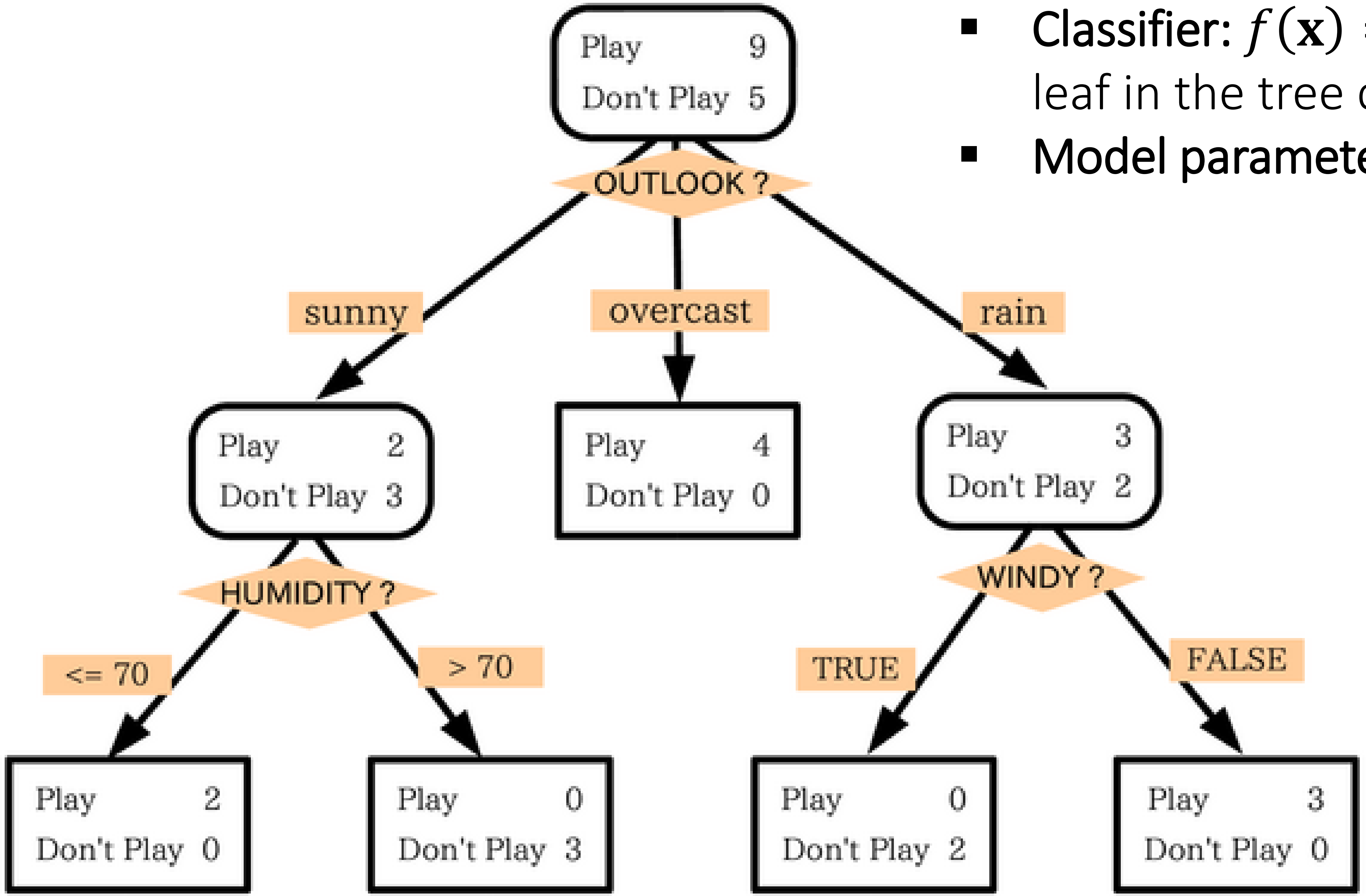
Temperature: Hot, Medium, Cool

Humidity: High, Normal, Low

Wind: Strong, Weak

Will I play tennis today?

Decision tree: example



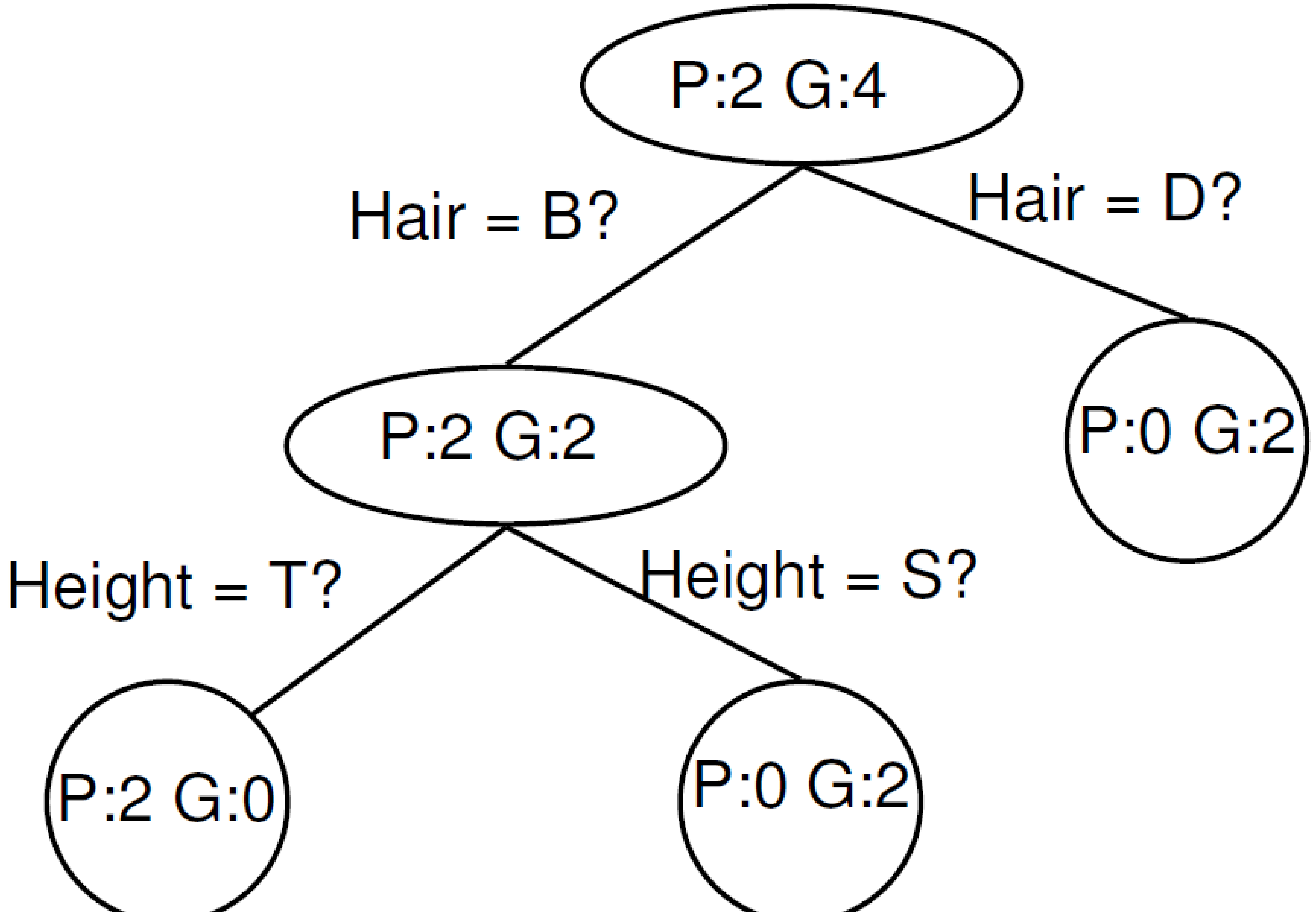
- **Classifier:** $f(\mathbf{x}) = y \rightarrow$ majority class in the leaf in the tree containing $\mathbf{x}$
- **Model parameters:** the tree structure and size

Decision trees

- Pieces:
  - Find the best attribute to split on
  - Find the best split on the chosen attribute
  - Decide on when to stop splitting

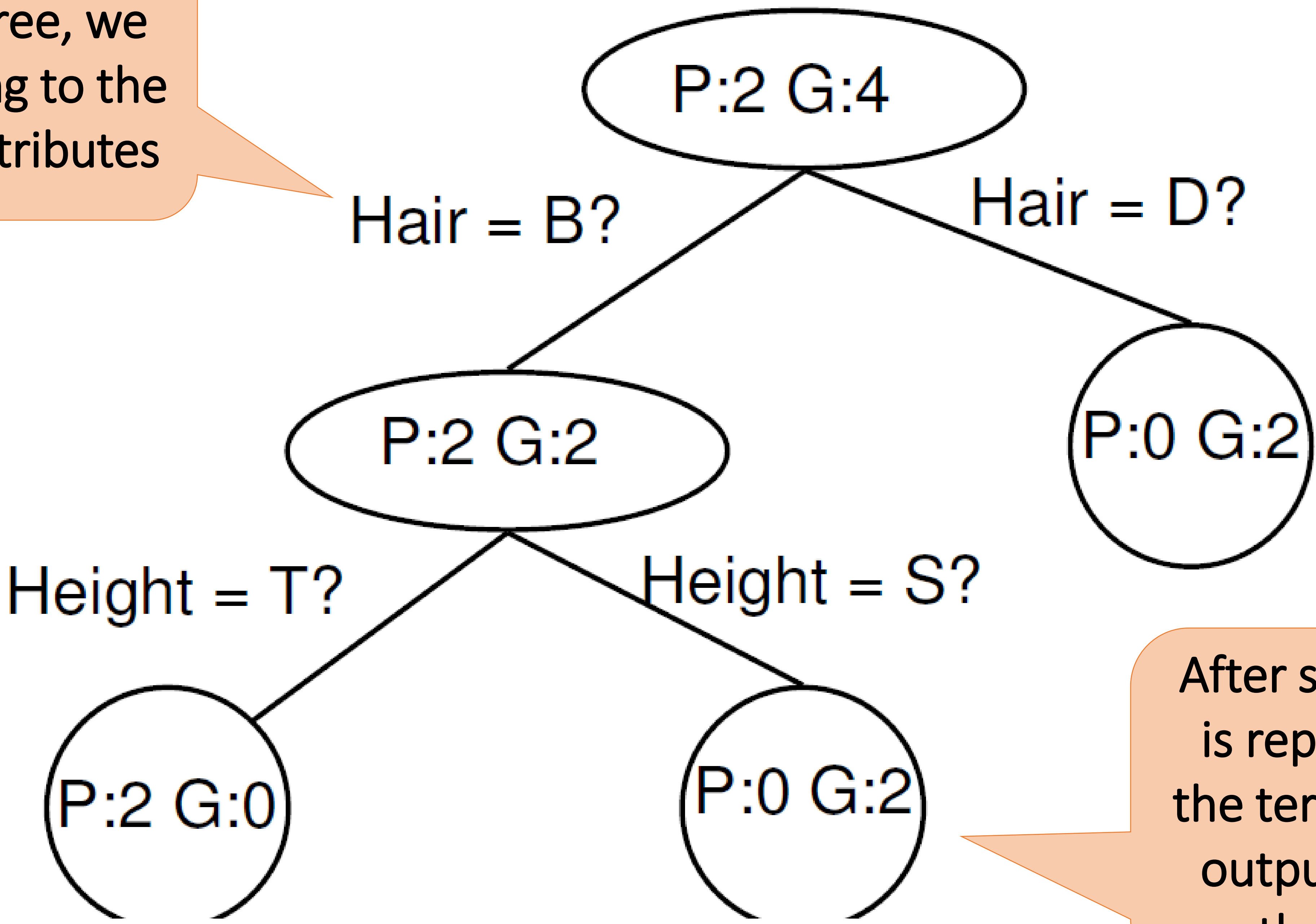
Categorical or discrete attributes

- Two variables:
  - Hair = {blond, dark}
  - Height = {tall, short}
- Label:
  - Country: {Gromland, Polvia}
- Training data:
- $\mathbf{X} = \begin{bmatrix} B & T \\ B & T \\ B & S \\ D & S \\ D & T \\ B & S \end{bmatrix}$ and $y = \begin{bmatrix} P \\ P \\ G \\ G \\ G \\ G \end{bmatrix}$



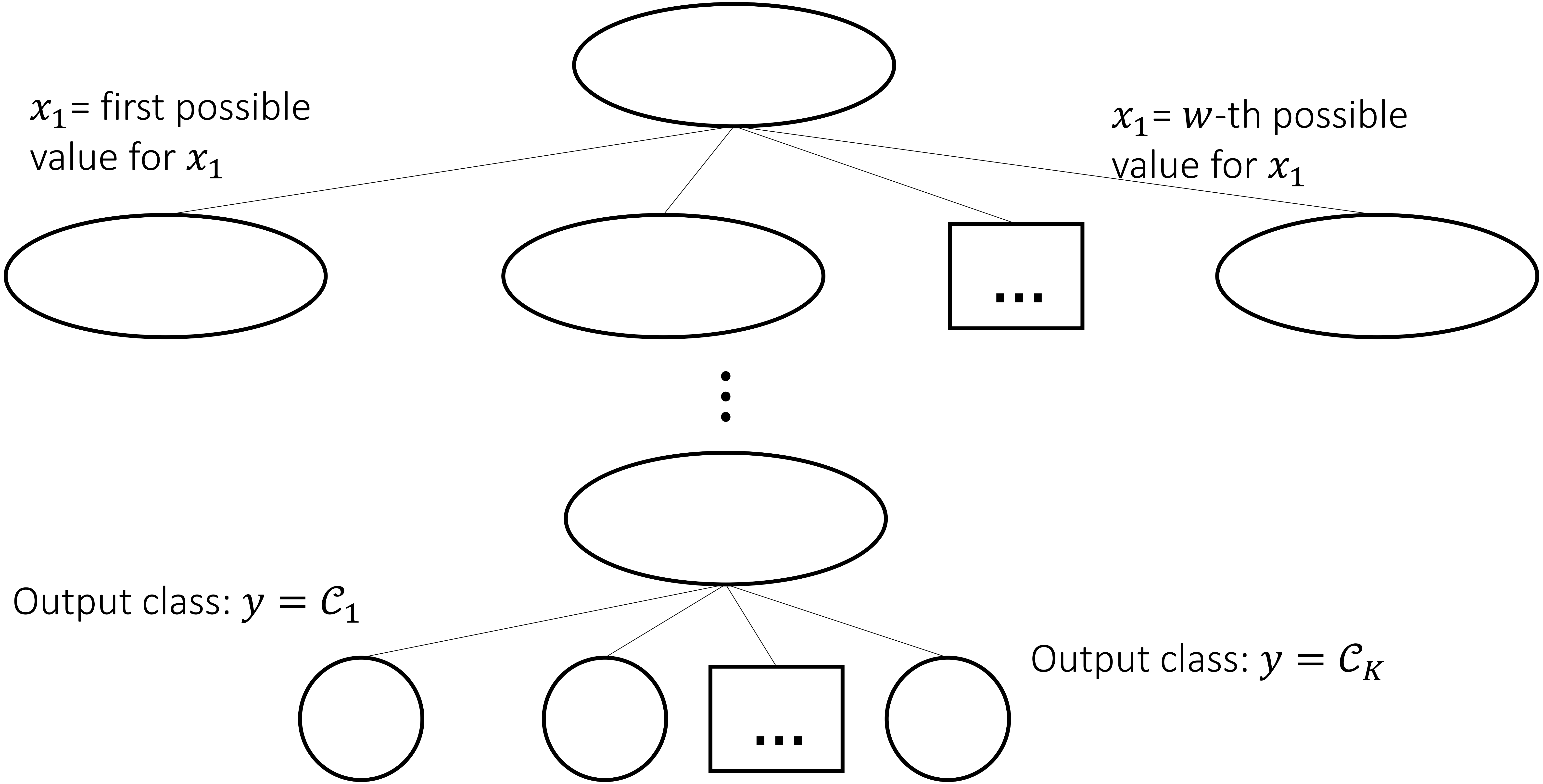
Categorical or discrete attributes

At each level of the tree, we split the data according to the value of one of the attributes

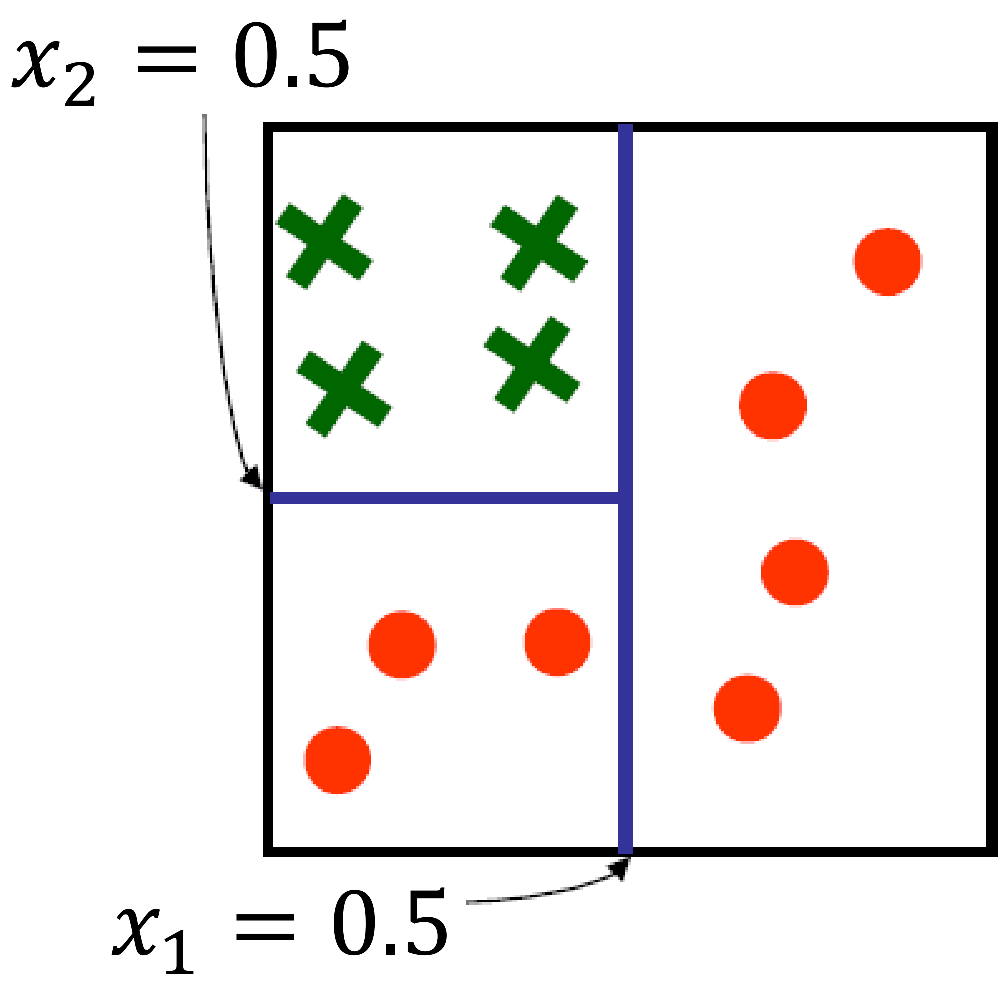


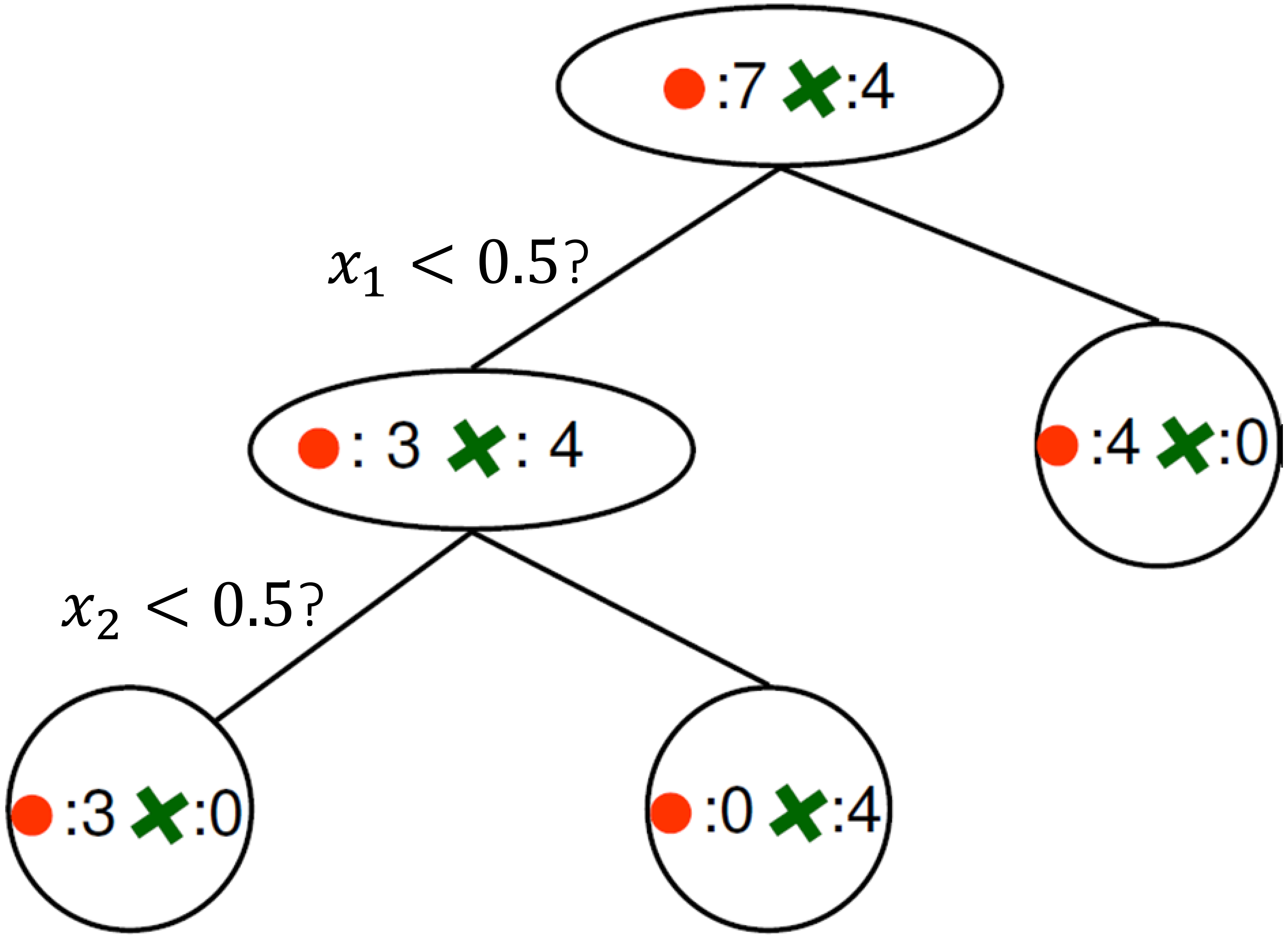
After sufficient splits, only one class is represented in the node. This is the terminal leaf of the tree, and we output the corresponding class to that node. (In this case, "G")

General decision tree: discrete attributes

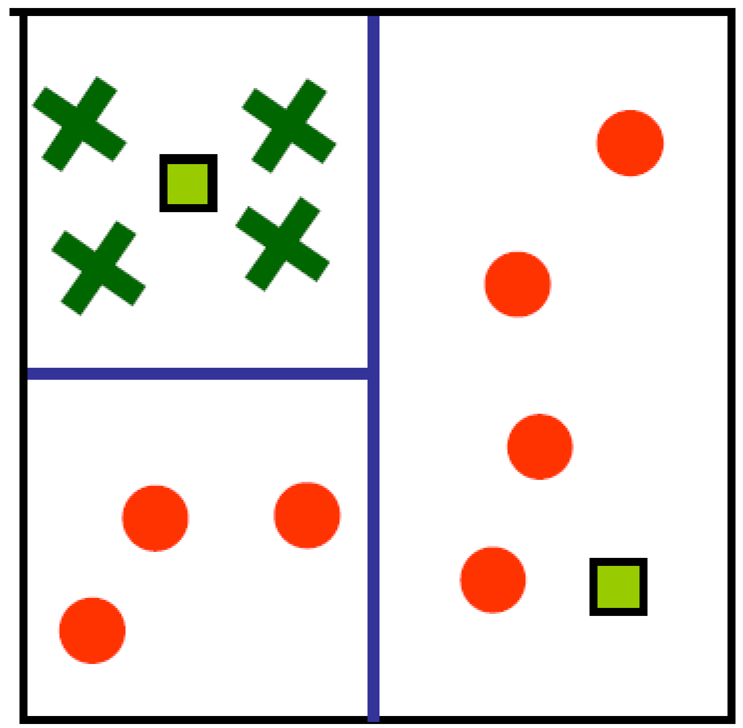


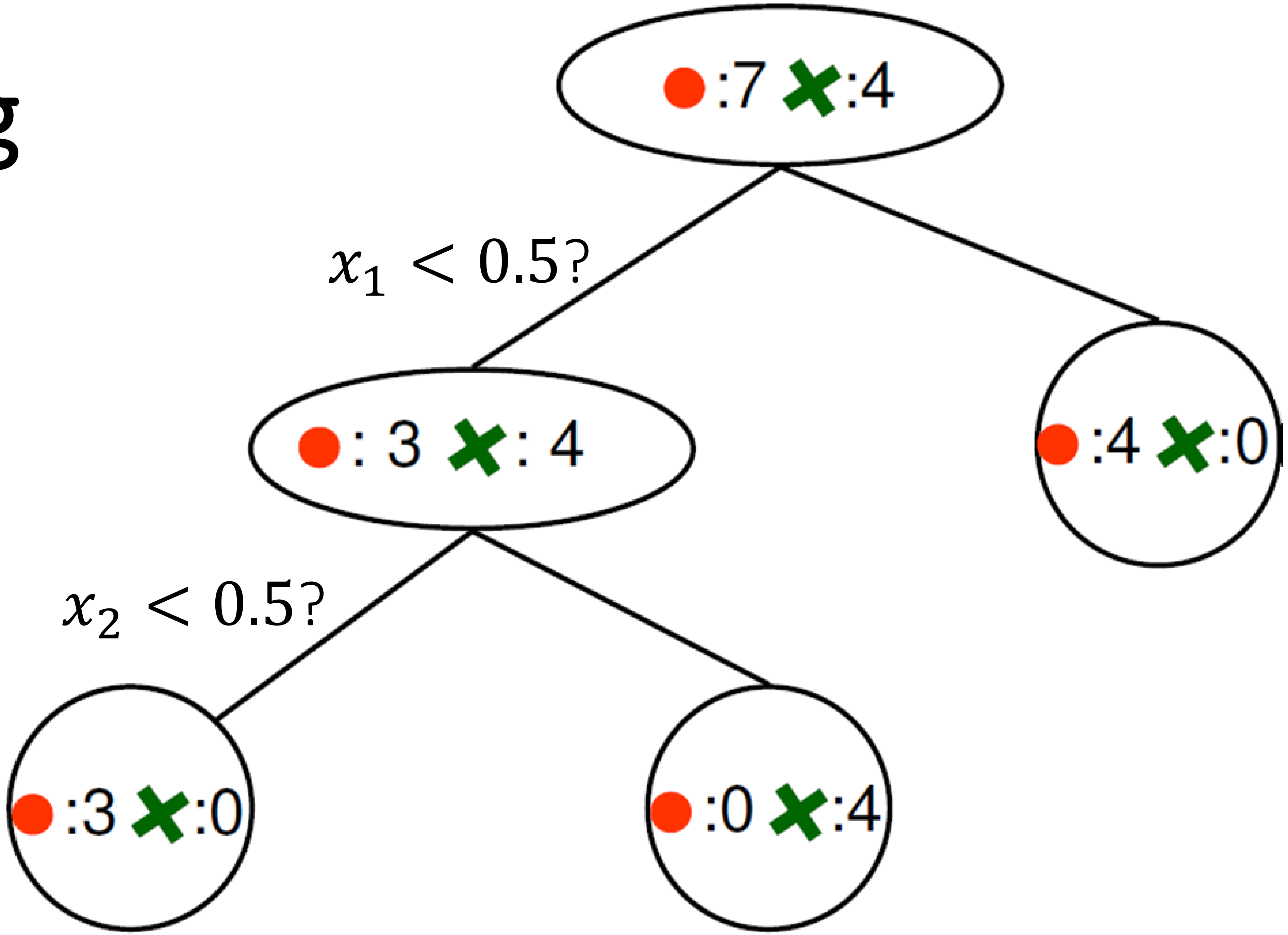
General decision tree: continuous attributes





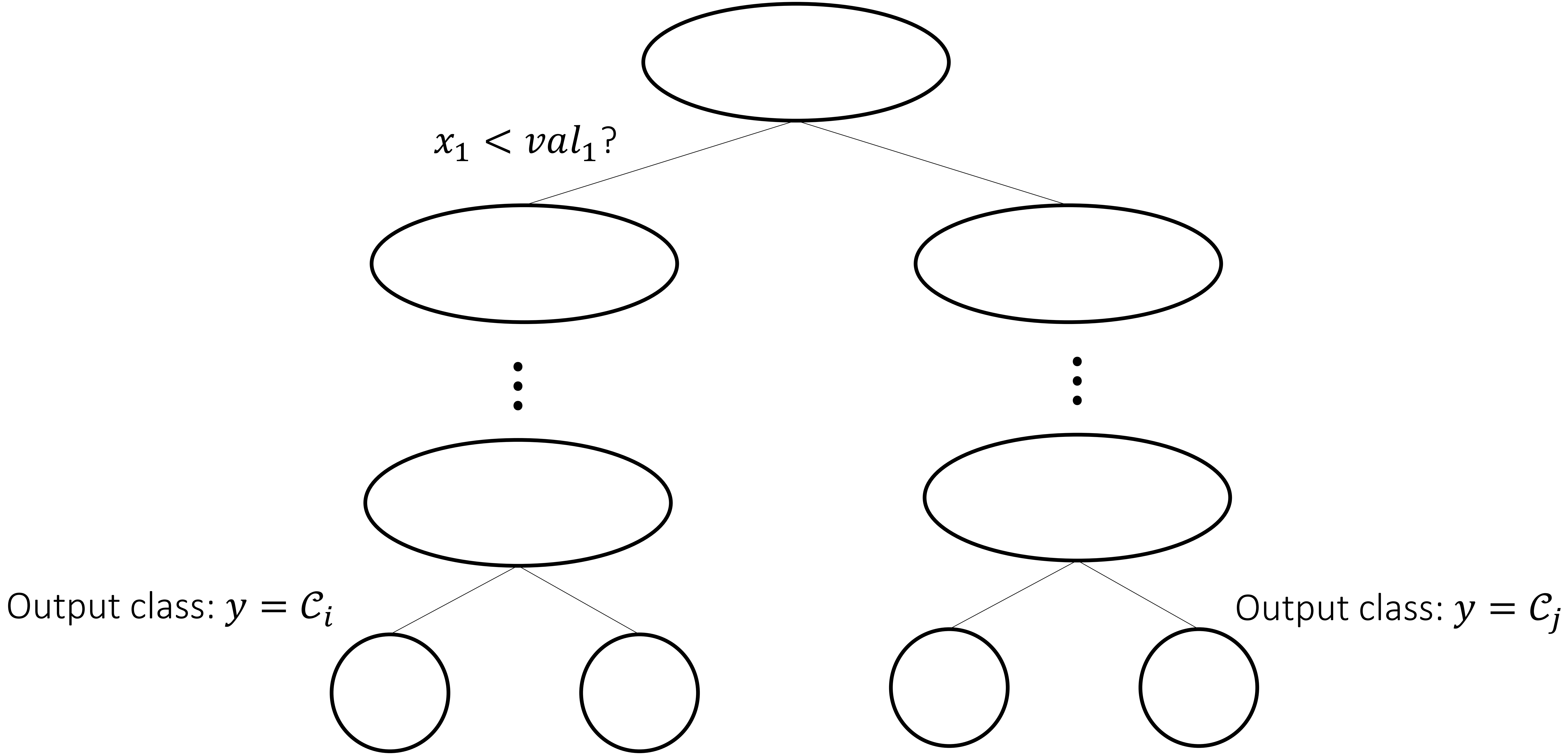
Decision tree: testing





- The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:
  - $\mathbf{x_A} = [0.2 \quad 0.8]^\mathrm{T}$ is classified as
  - $\mathbf{x_B} = [0.8 \quad 0.2]^\mathrm{T}$ is classified as

General decision tree: continuous attributes



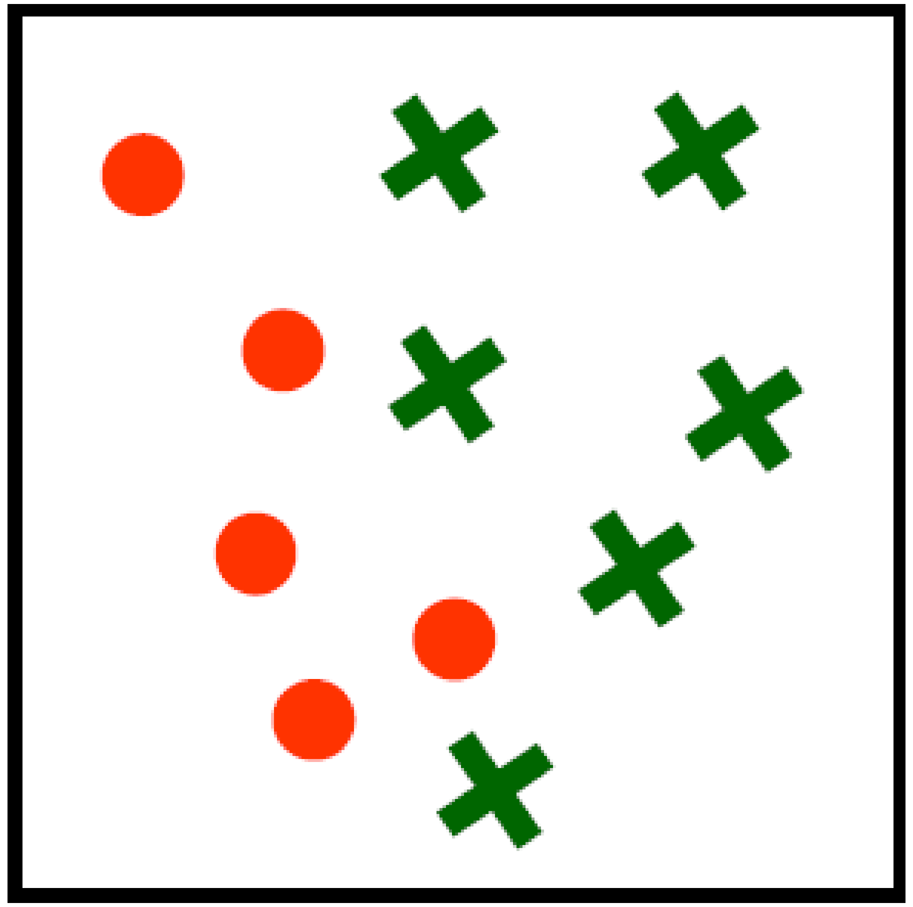
Outline

- Intuition
- Overview
- Learning a tree
- Algorithm

Important questions

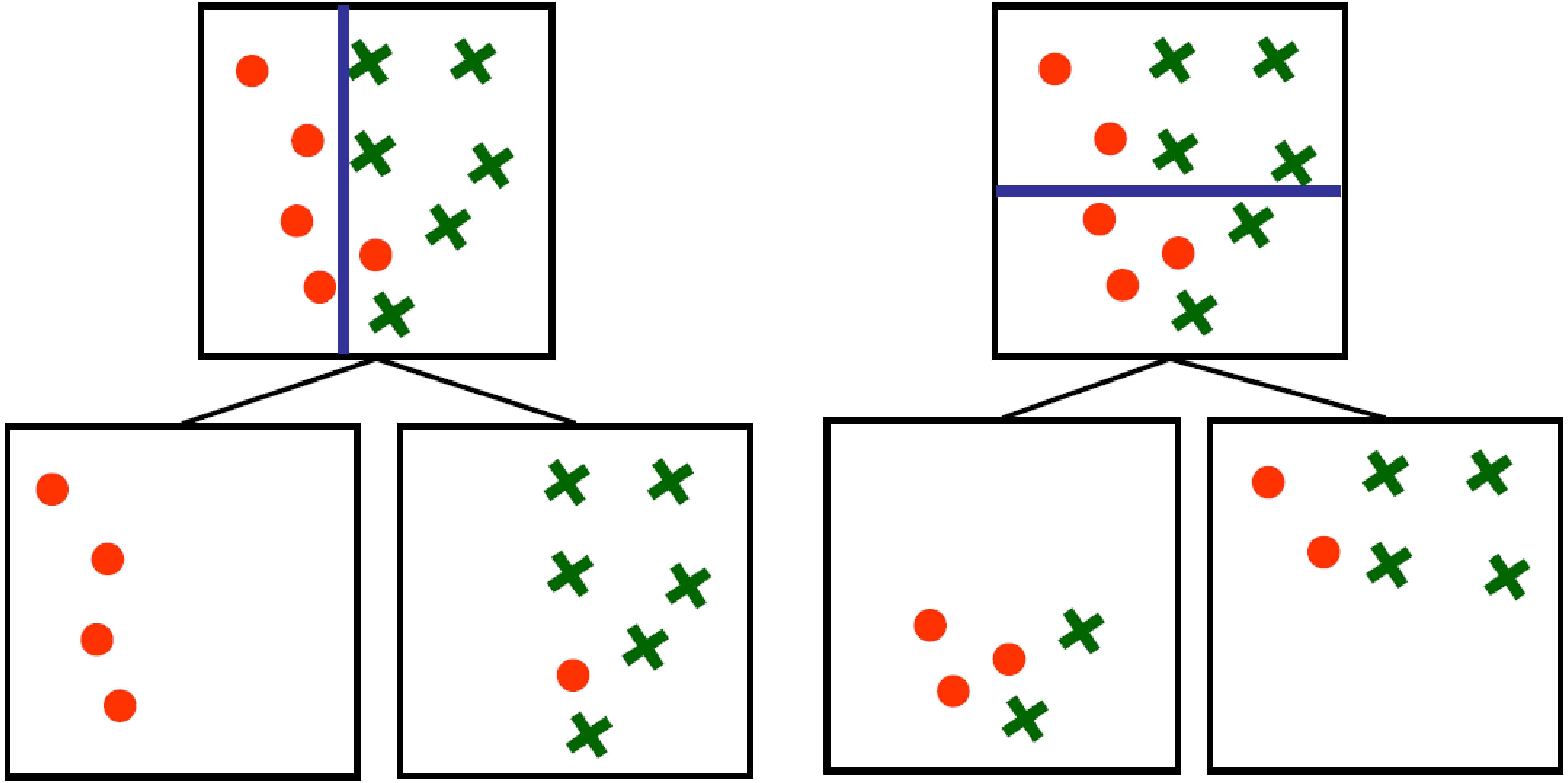
- How to choose the attribute and value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

Choosing the attribute and value to split on at each tree level

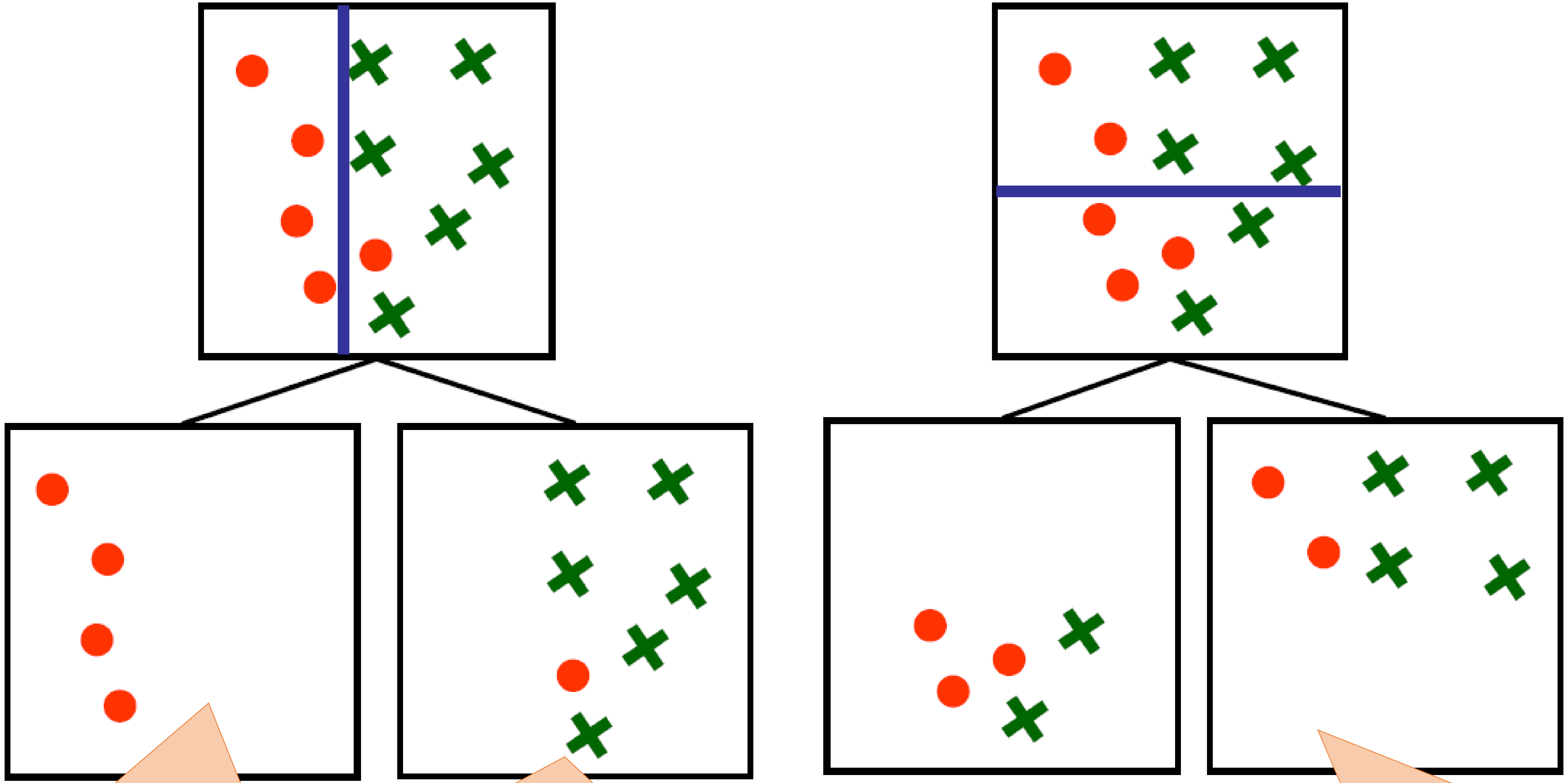


- Two classes (red circles/green crosses)
- Two attributes: x_1 and x_2
- 11 points in training data
- **Goal:** construct a decision tree such that the leaf nodes predict correctly the class for all the training examples

Choosing the attribute and value to split on at each tree level



Choosing the attribute and value to split on at each tree level

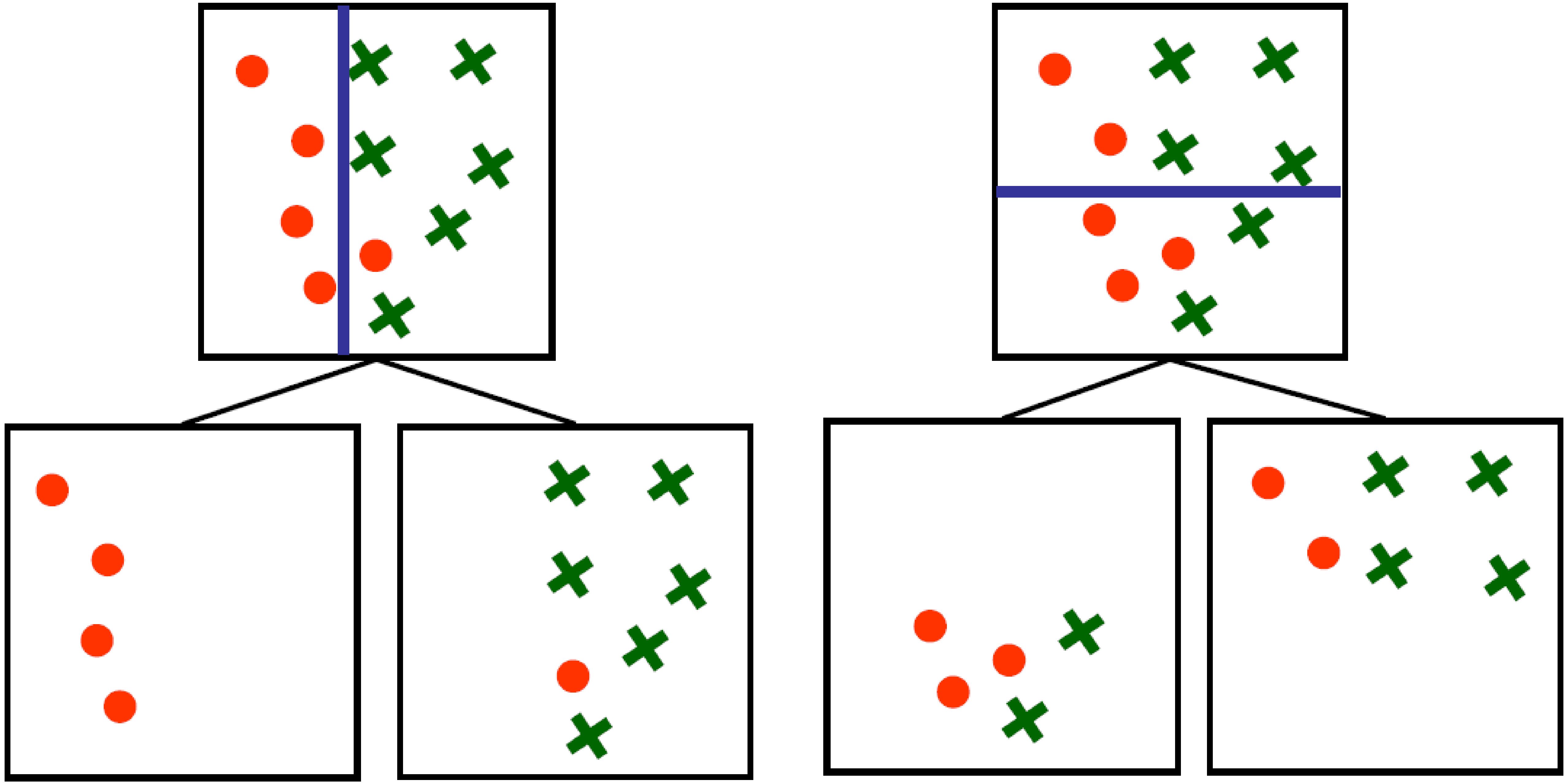


This node is pure because there is only one class left, therefore no ambiguity in the class label

This node is almost pure because there is little ambiguity in the class label

These nodes contain a mixture of classes, not disambiguating between classes

Choosing the attribute and value to split on at each tree level



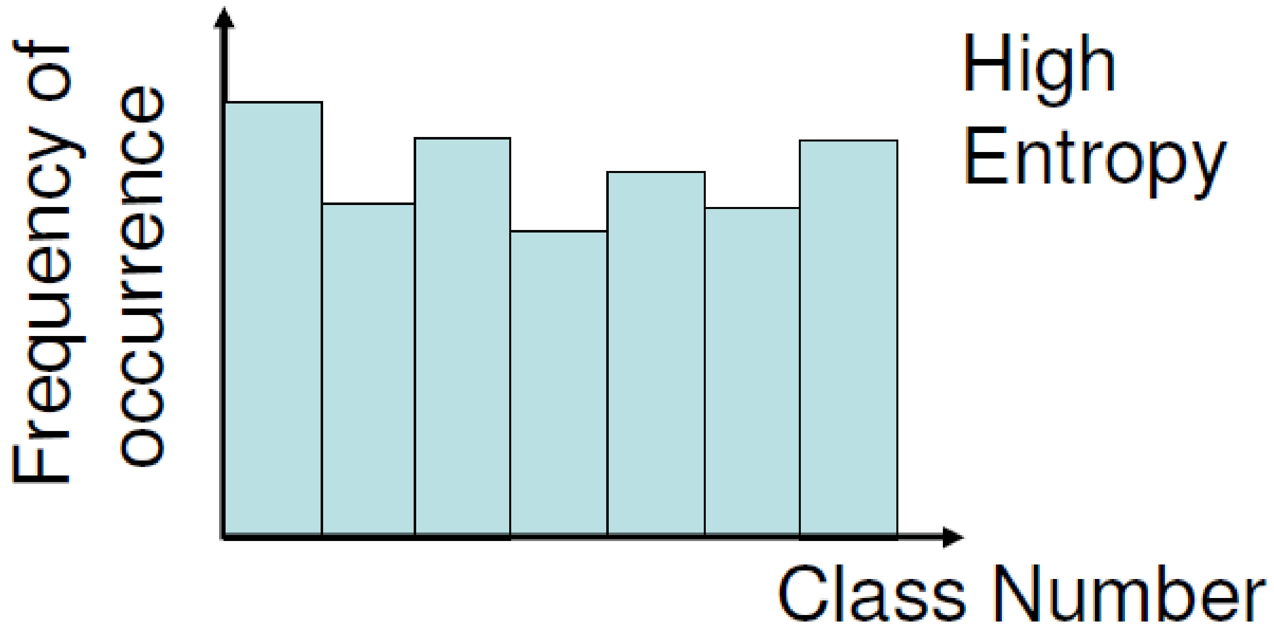
Find the most compact tree that classifies the training data correctly (Occam's razor).
To do that we want to find the split choices that will get us the fastest to pure nodes

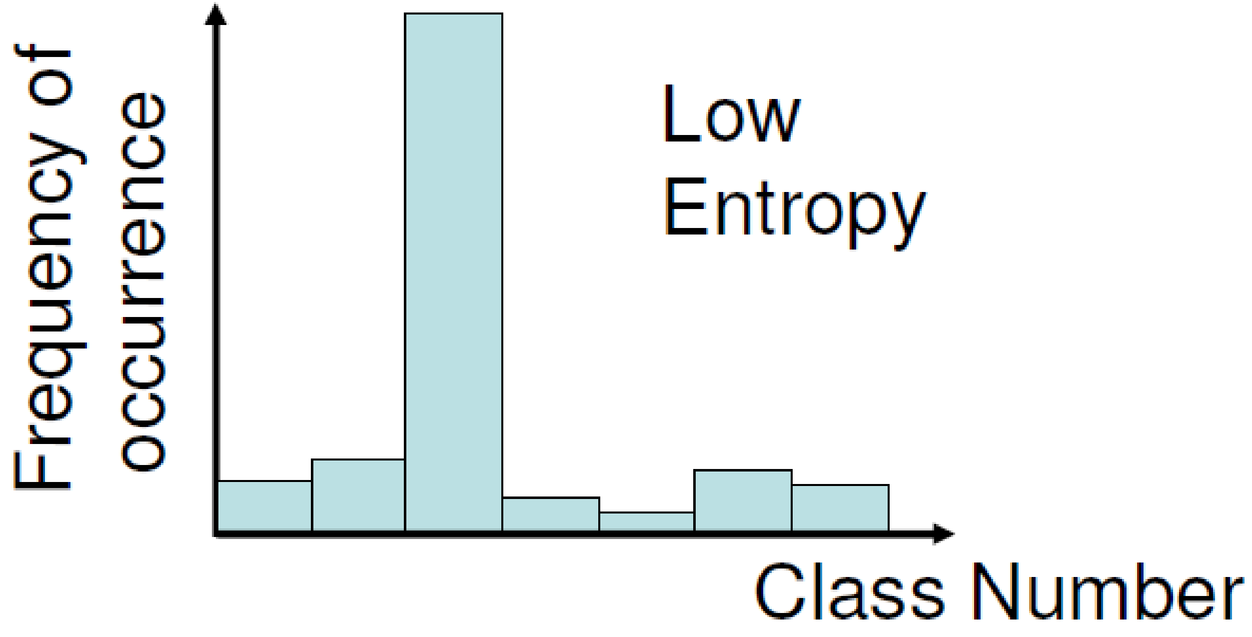
Entropy

- In general, the average number of bits necessary to encode K values is the entropy:

$$H(x) = -\sum_{k=1}^{K} p(x=k)\log_2 p(x=k) = \sum_{k=1}^{K} p(x=k)\log_2 \frac{1}{p(x=k)}$$

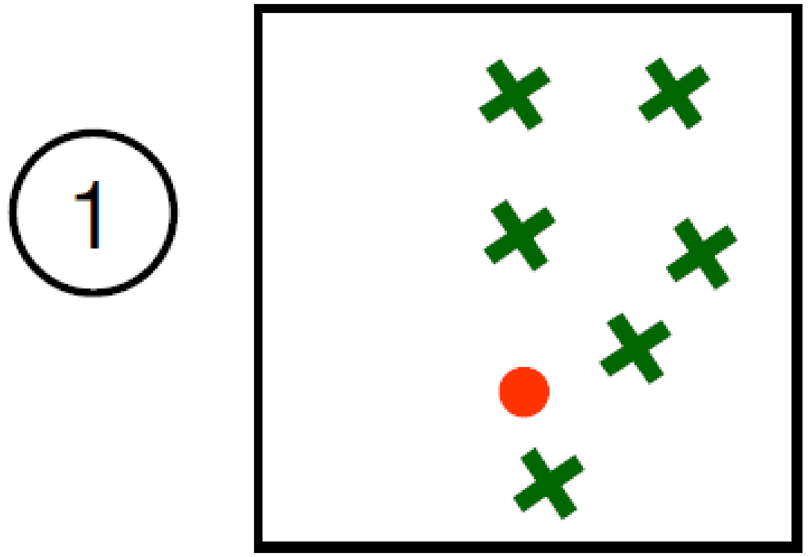
- High entropy → all the classes are (nearly) equally likely
- Low entropy → a few classes are likely; most of the classes are rarely observed





The entropy captures the degree of purity of the distribution

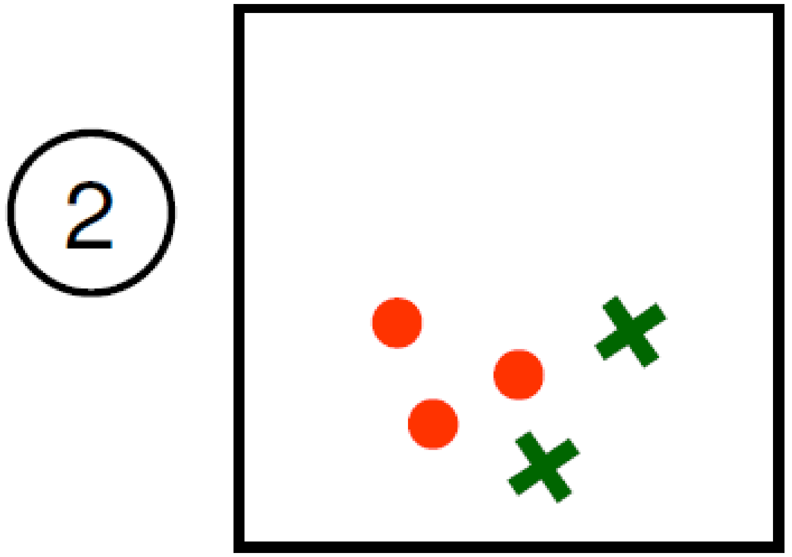
Entropy calculation: example



$N_1 = 1;\ N_2 = 6$

$$p_1 = \frac{N_1}{(N_1 + N_2)} = \frac{1}{7}$$

$$p_2 = \frac{N_2}{(N_1 + N_2)} = \frac{6}{7}$$



$N_1 = 3;\ N_2 = 2$

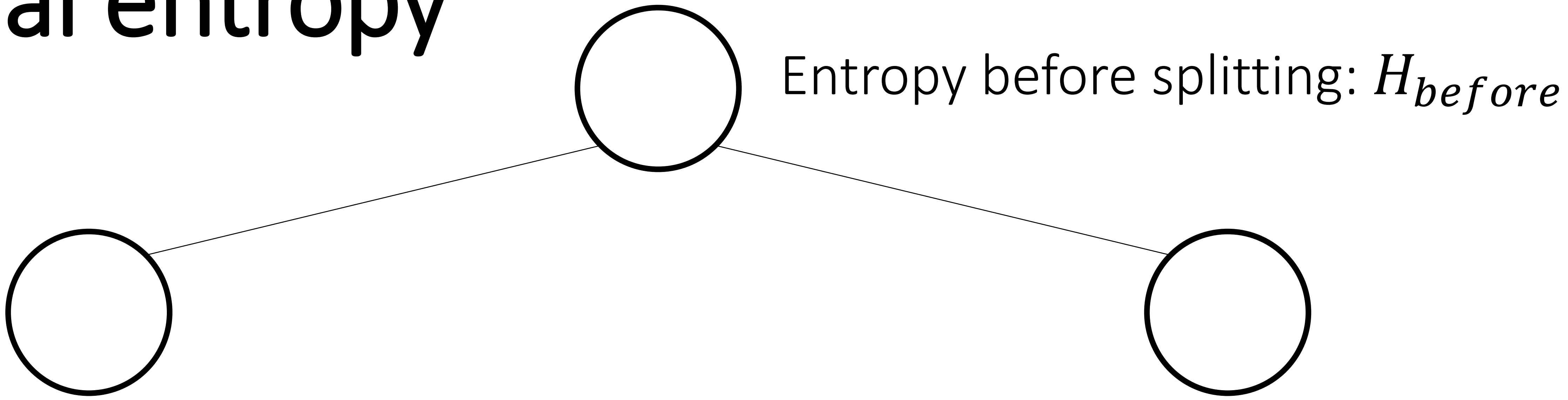
$$p_1 = \frac{N_1}{(N_1 + N_2)} = \frac{3}{5}$$

$$p_2 = \frac{N_2}{(N_1 + N_2)} = \frac{2}{5}$$

$$H_{node\ 1}(x) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 = 0.59 \qquad H_{node\ 2}(x) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 = 0.97$$

$H_{node\ 1}(x) < H_{node\ 2}(x) \rightarrow$ Node 2 less pure than Node 1

Conditional entropy



After splitting, a fraction p_L of the data goes to the left node, which has entropy H_L

After splitting, a fraction p_R of the data goes to the left node, which has entropy H_R

The average conditional entropy after splitting is:

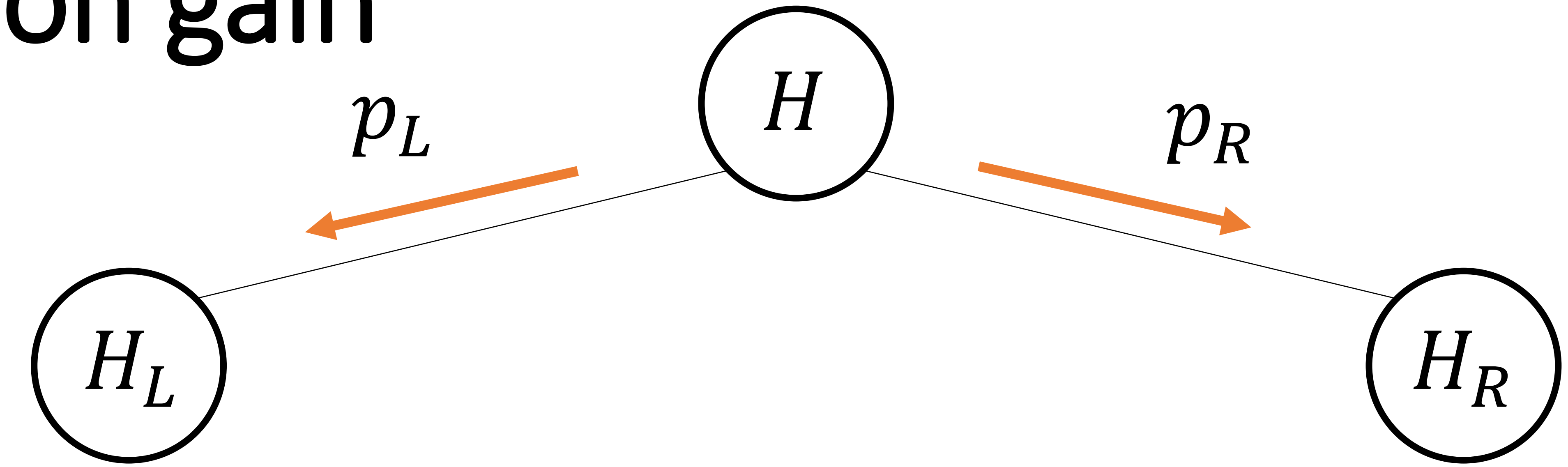
$$H_{after} = p_L \times H_L + p_R \times H_R$$

$$H(Y|x_d) = p_{x_d<val} H(Y|x_d < val) + p_{x_d \geq split} H(Y|x_d \geq val)$$

Probability that a random input is directed to the left node

Entropy of left node

Information gain



- We want nodes as pure as possible:
  - We want to reduce the entropy as much as possible
  - We want to maximize the difference between the entropy of the parent node and the expected entropy of the children

- Maximize information gain:

$$IG = H - (p_L \times H_L + p_R \times H_R)$$

- **Information gain (IG)** = amount by which the ambiguity is decreased by splitting the node on a specific value

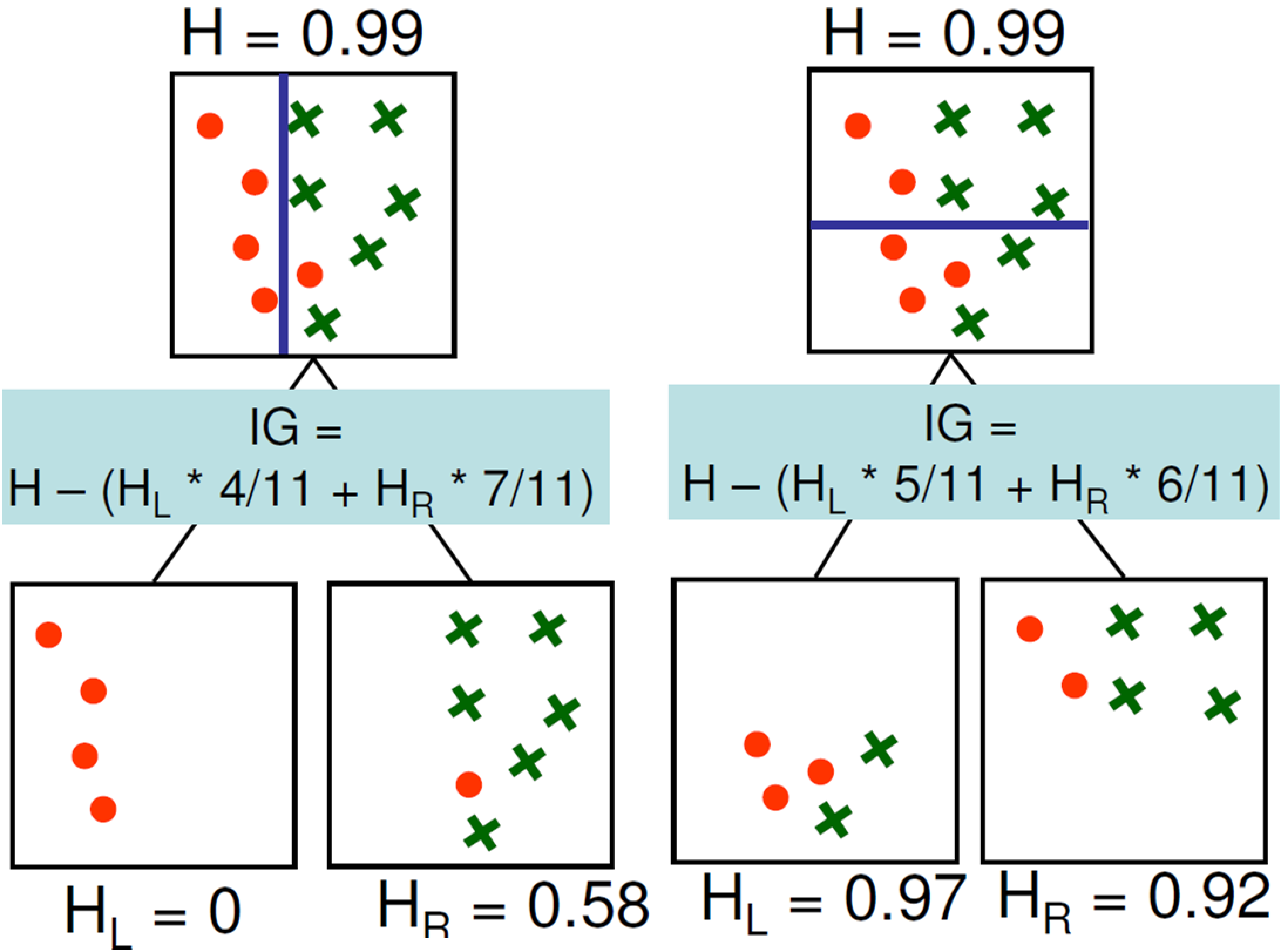
Reconnecting to information theory

- Entropy: $H(Y)$ = entropy of the distribution of classes at a node

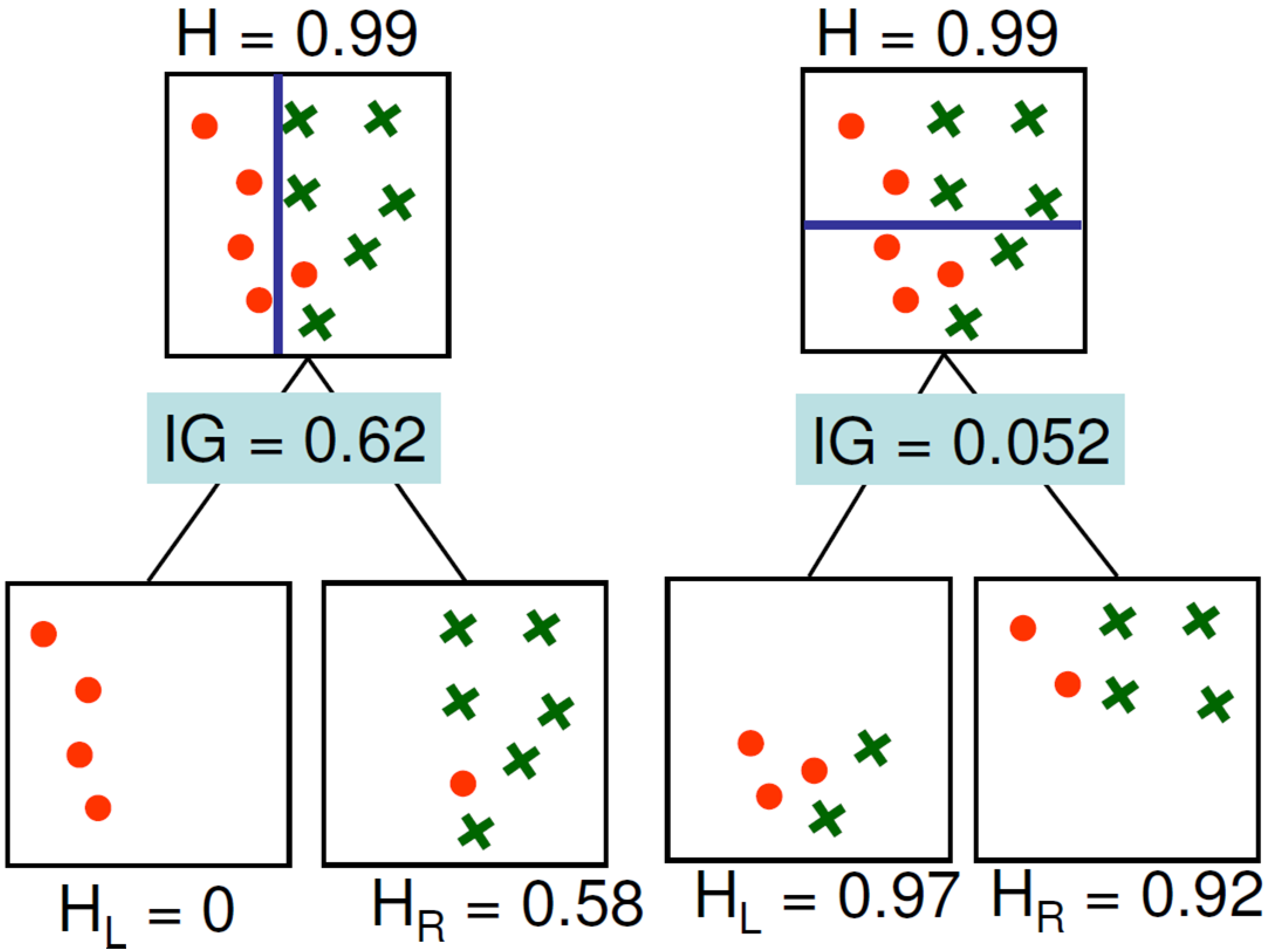
- Conditional entropy:
  - Discrete: $H(Y|x_d) =$ entropy after splitting with respect to variable d
  - Continuous: $H(Y|x_d, val) =$ entropy after splitting with respect to variable d with threshold val

- Information gain:
  - Discrete: $IG(Y|x_d) = H(Y) - H(Y|x_d) =$ mutual information between class and variable d
  - Continuous:$IG(Y|x_d, val) = H(Y) - H(Y|x_d, val) =$ mutual information between class and variable d with threshold val

Information gain: example

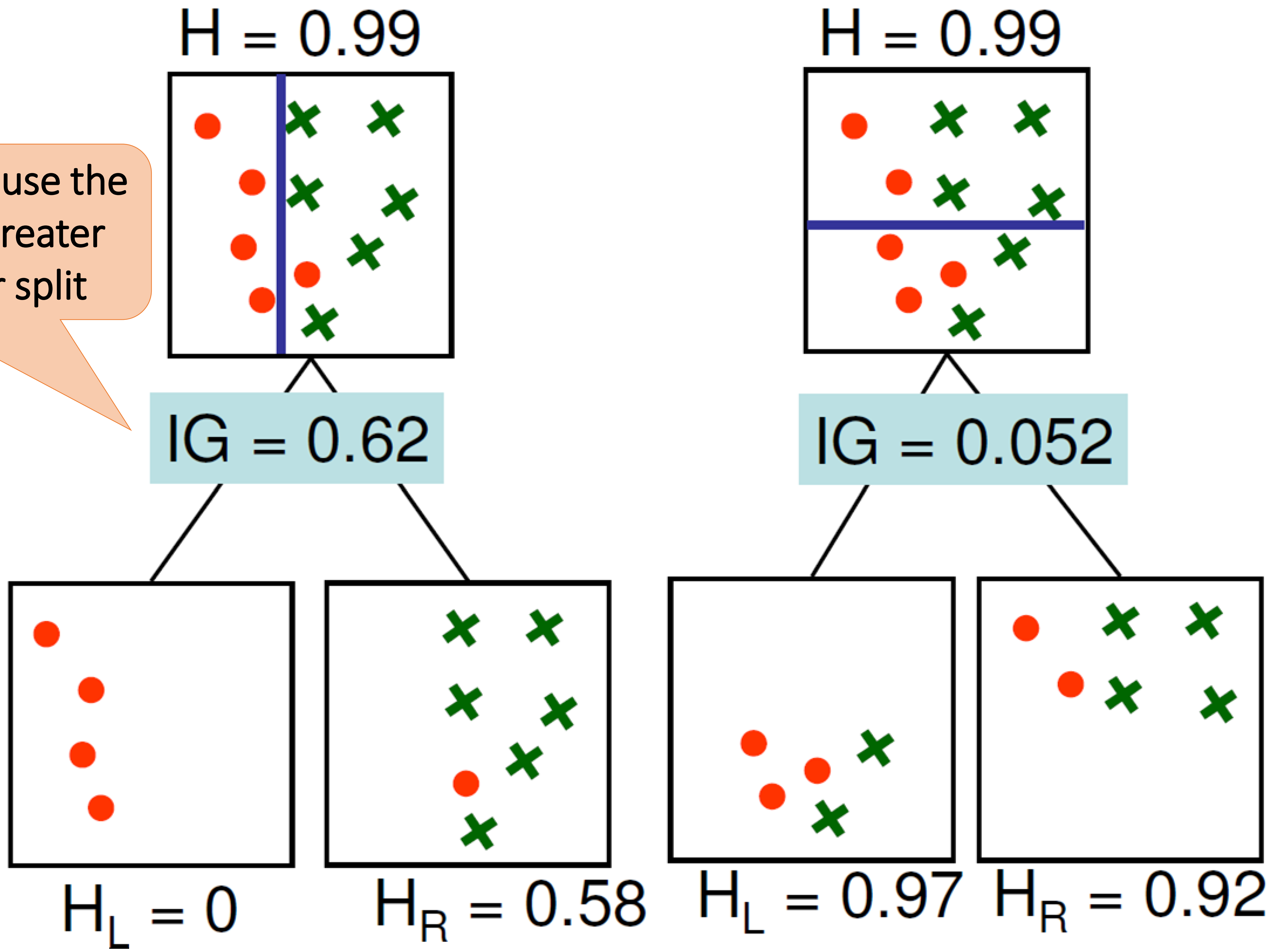


Information gain: example

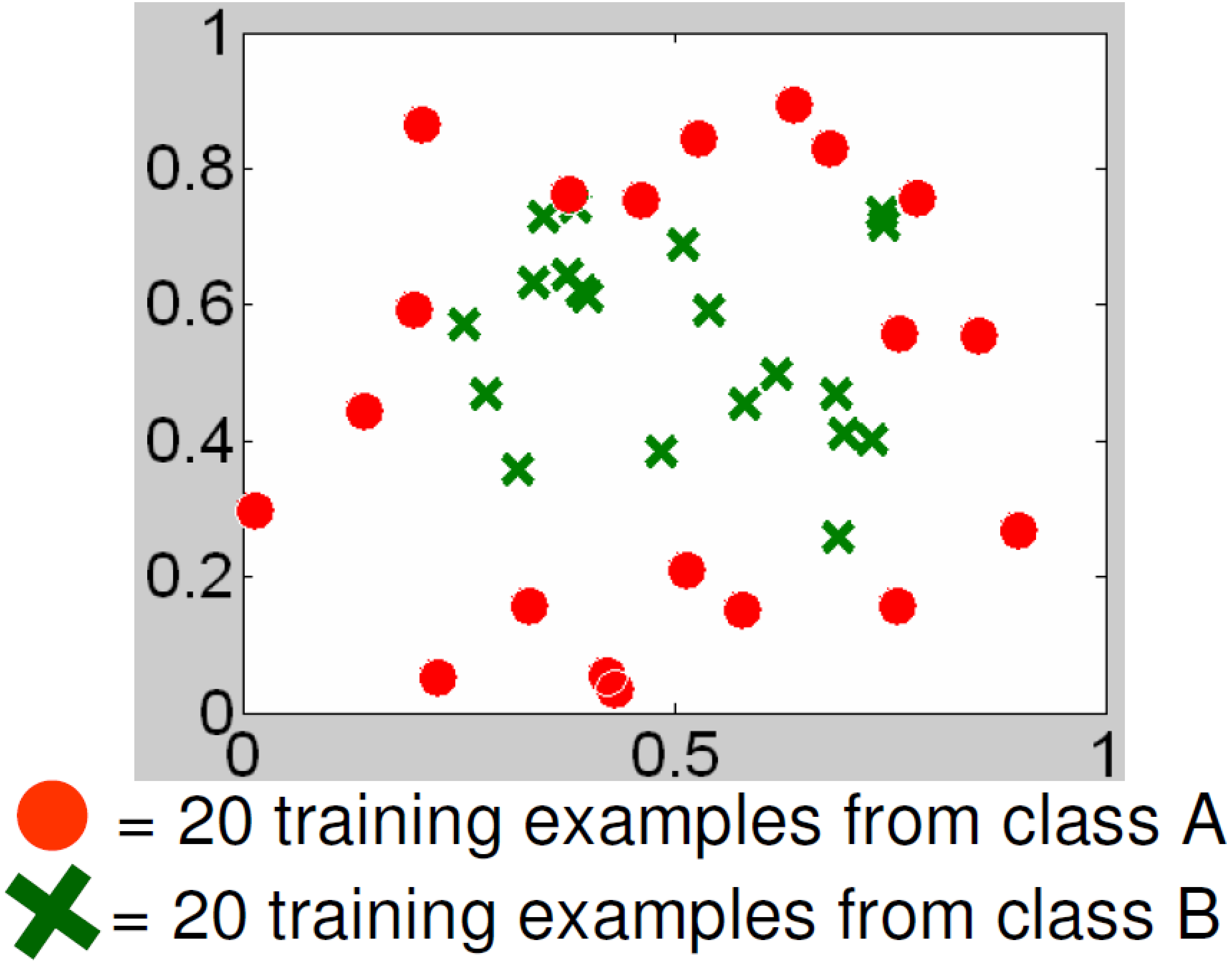


Information gain: example

Choose this split because the information gain is greater than with the other split

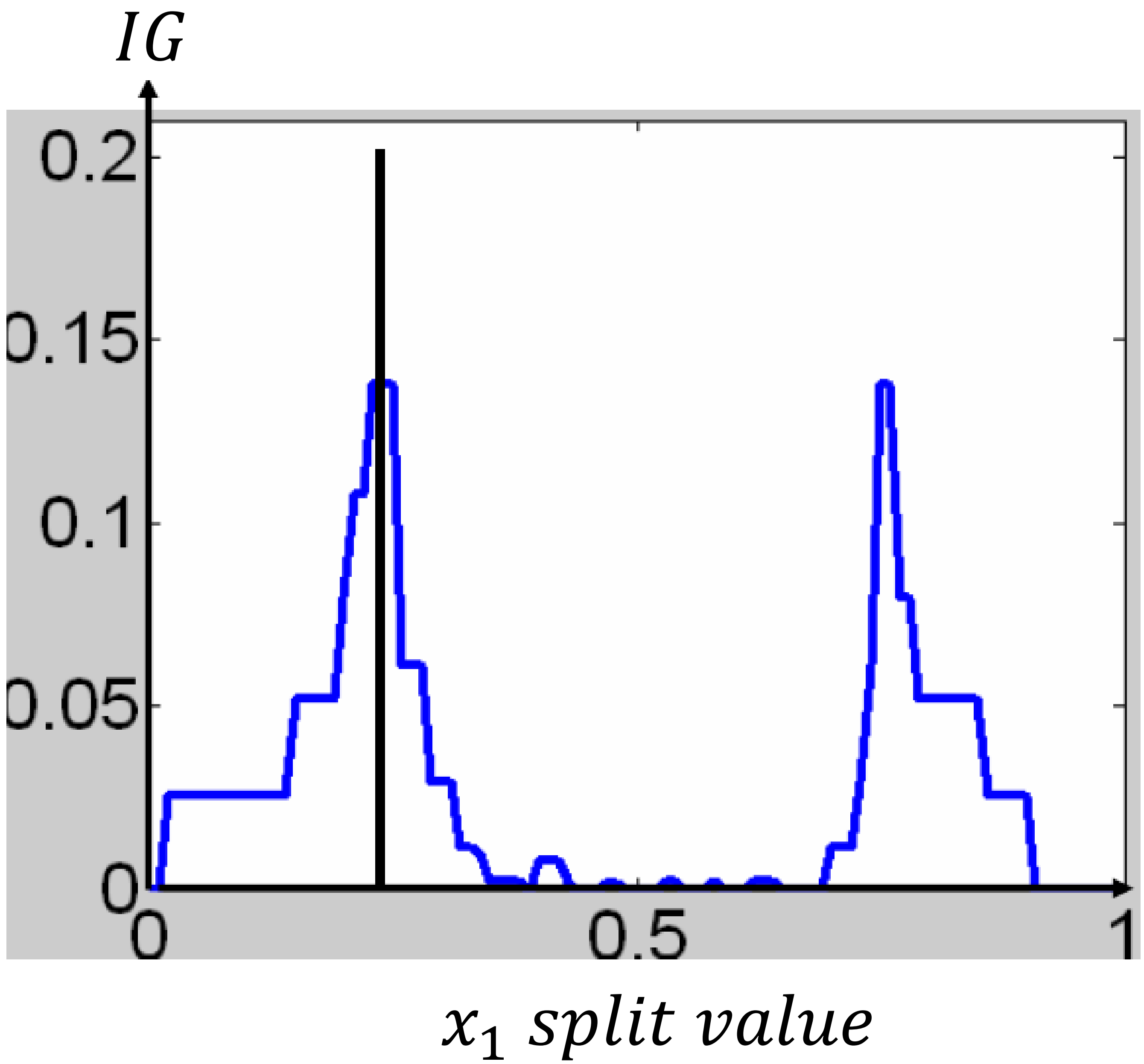


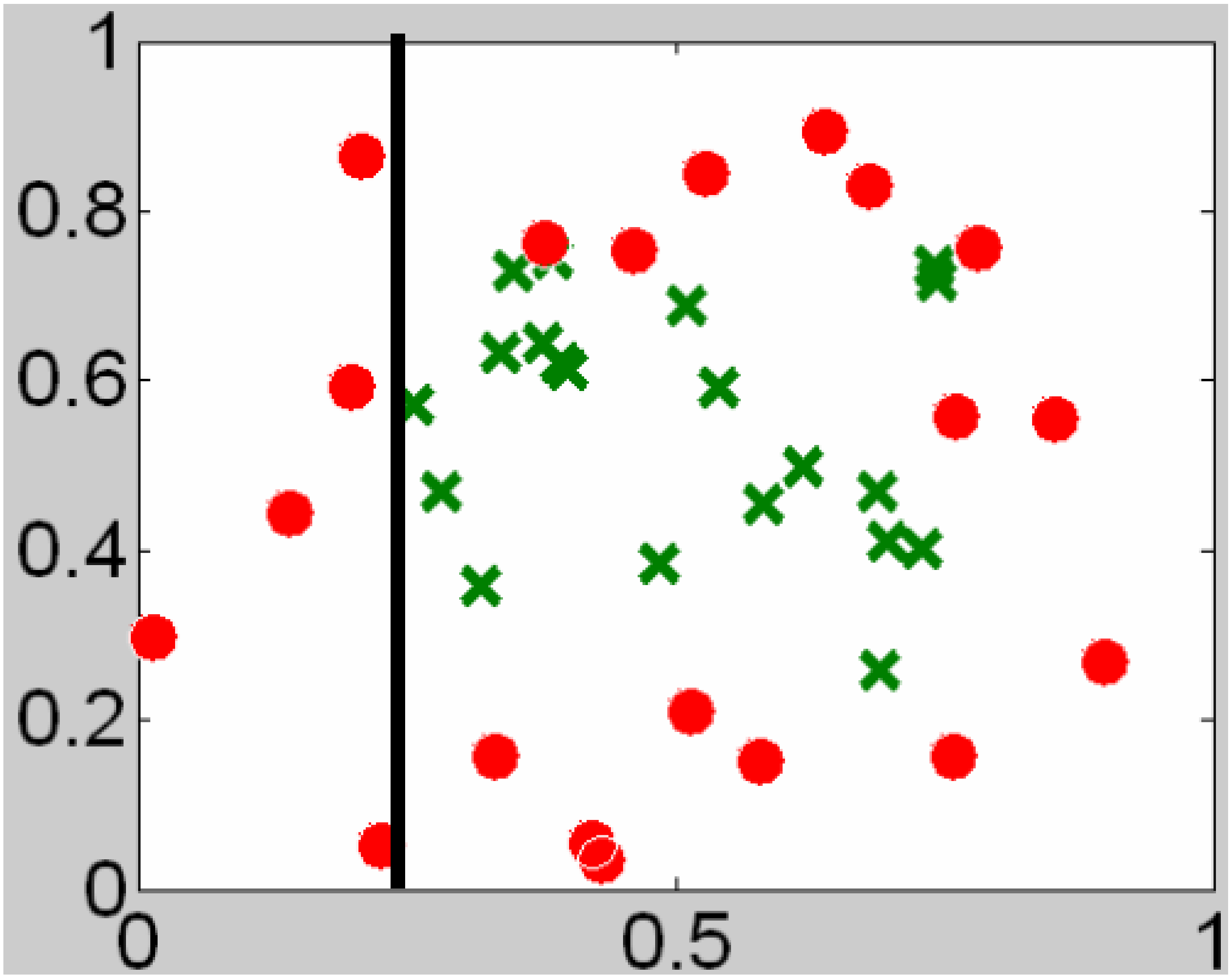
Complete example



Features: x_1 (horizontal) and x_2 (vertical)

Complete example

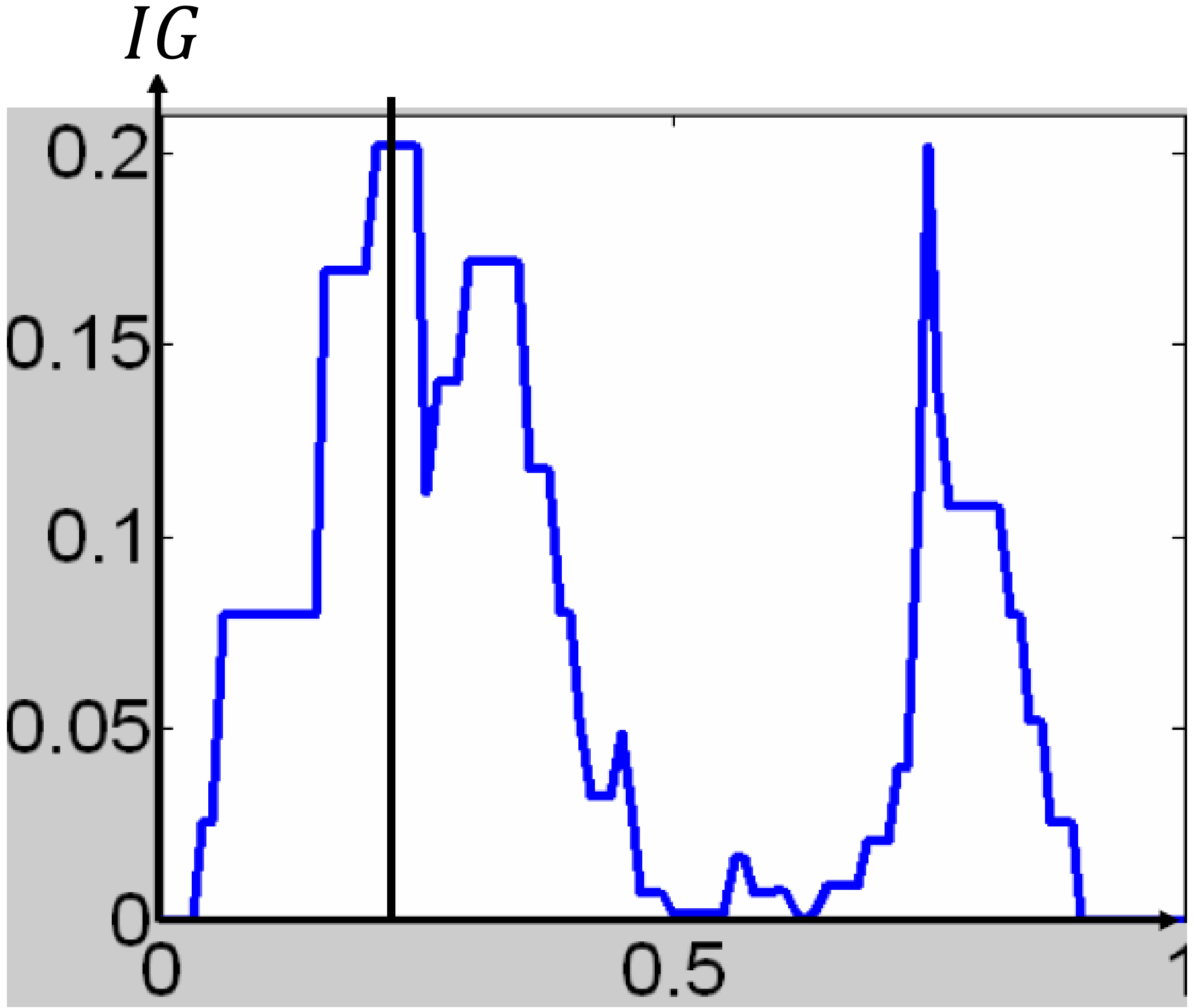




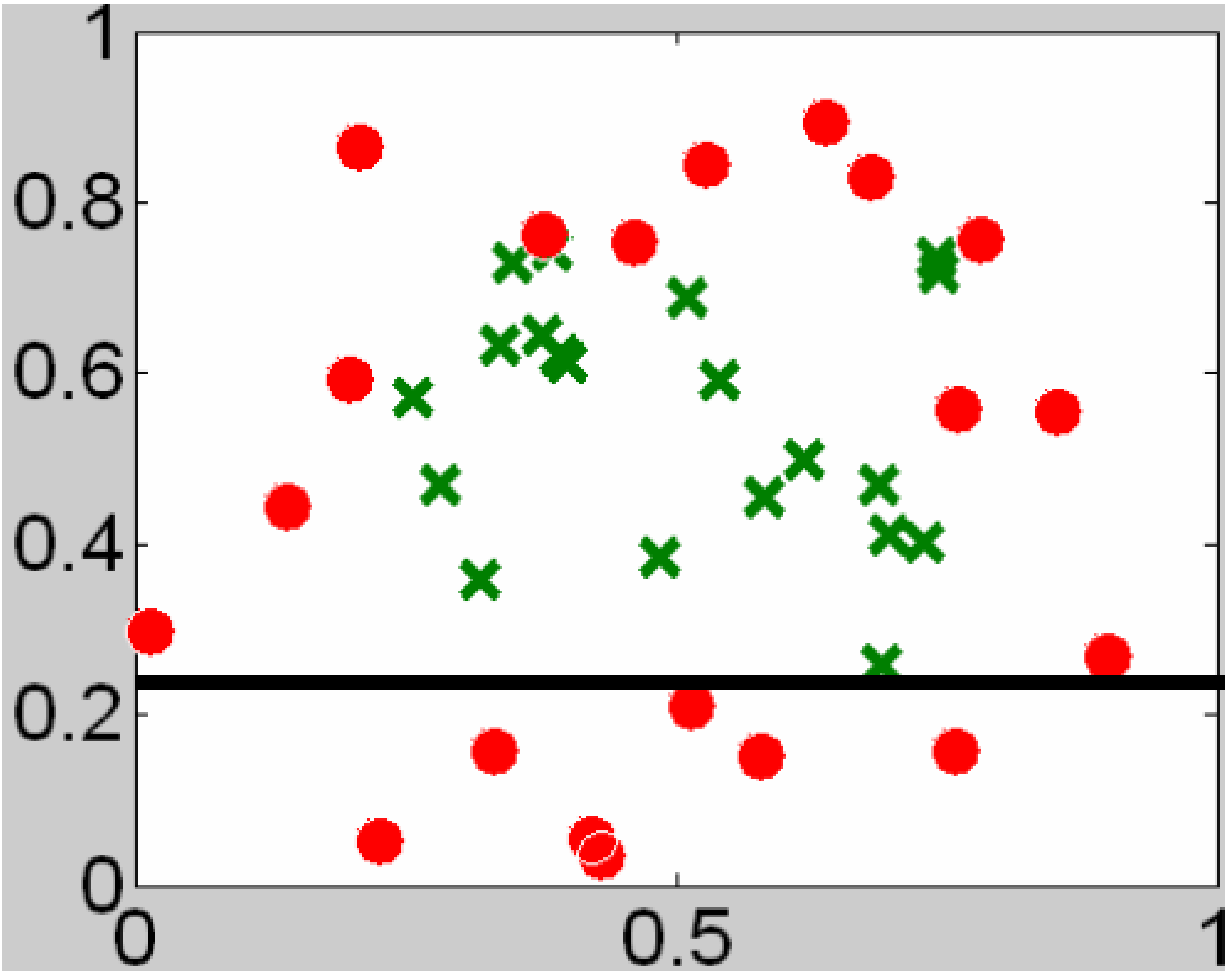
Best split value → maximum information gain for feature x_1:

$x_1 = 0.24$ with $IG = 0.138$

Complete example



x_2 *split value*



Best split value → maximum information gain for feature x_2:

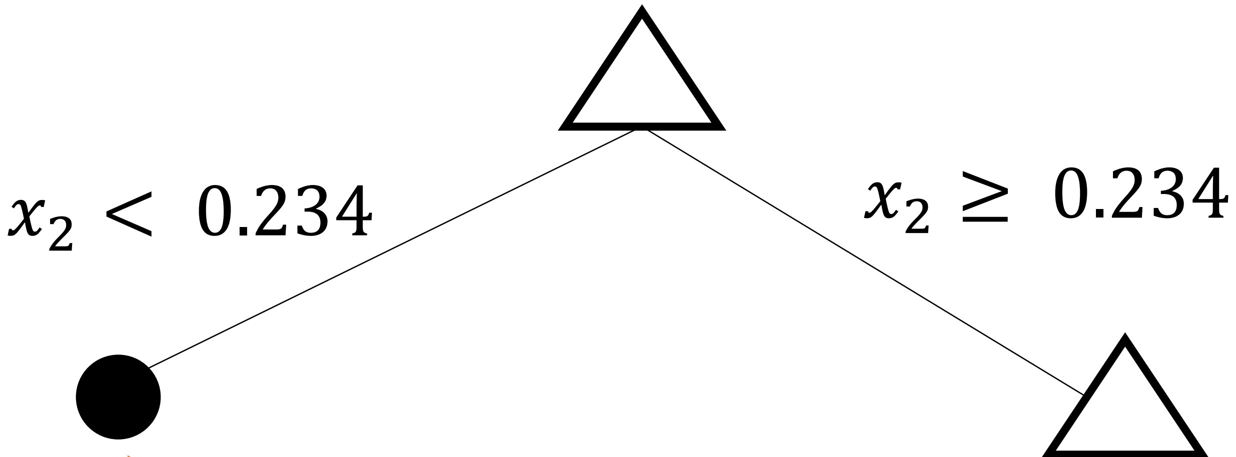
$x_2 = 0.234$ with $IG \ = \ 0.202$

Complete example

Best x_1 split: $x_1 = 0.24$ with $IG = 0.138$

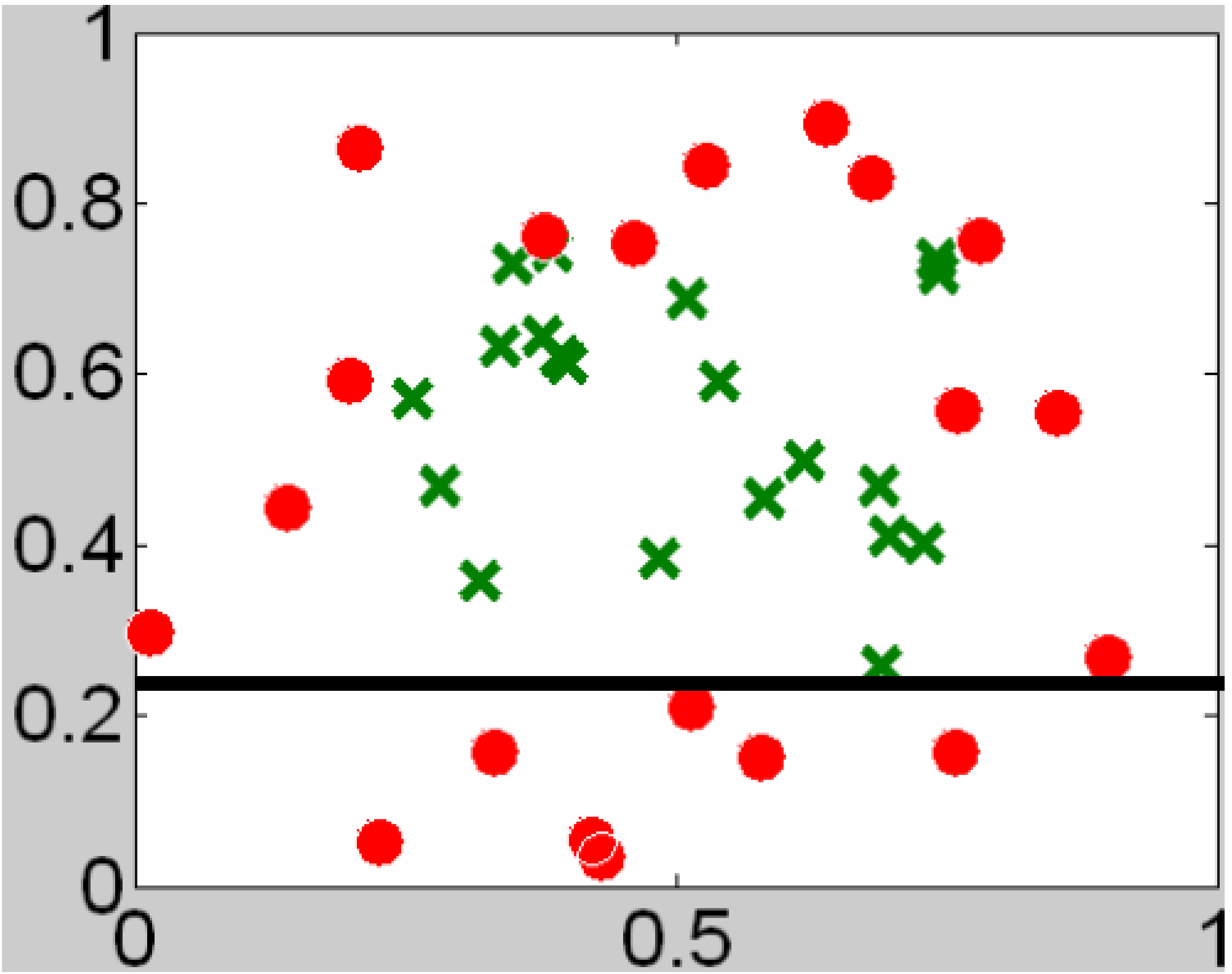
Best x_2 split: $x_2 = 0.234$ with $IG = 0.202$

Split: $x_2 = 0.234$



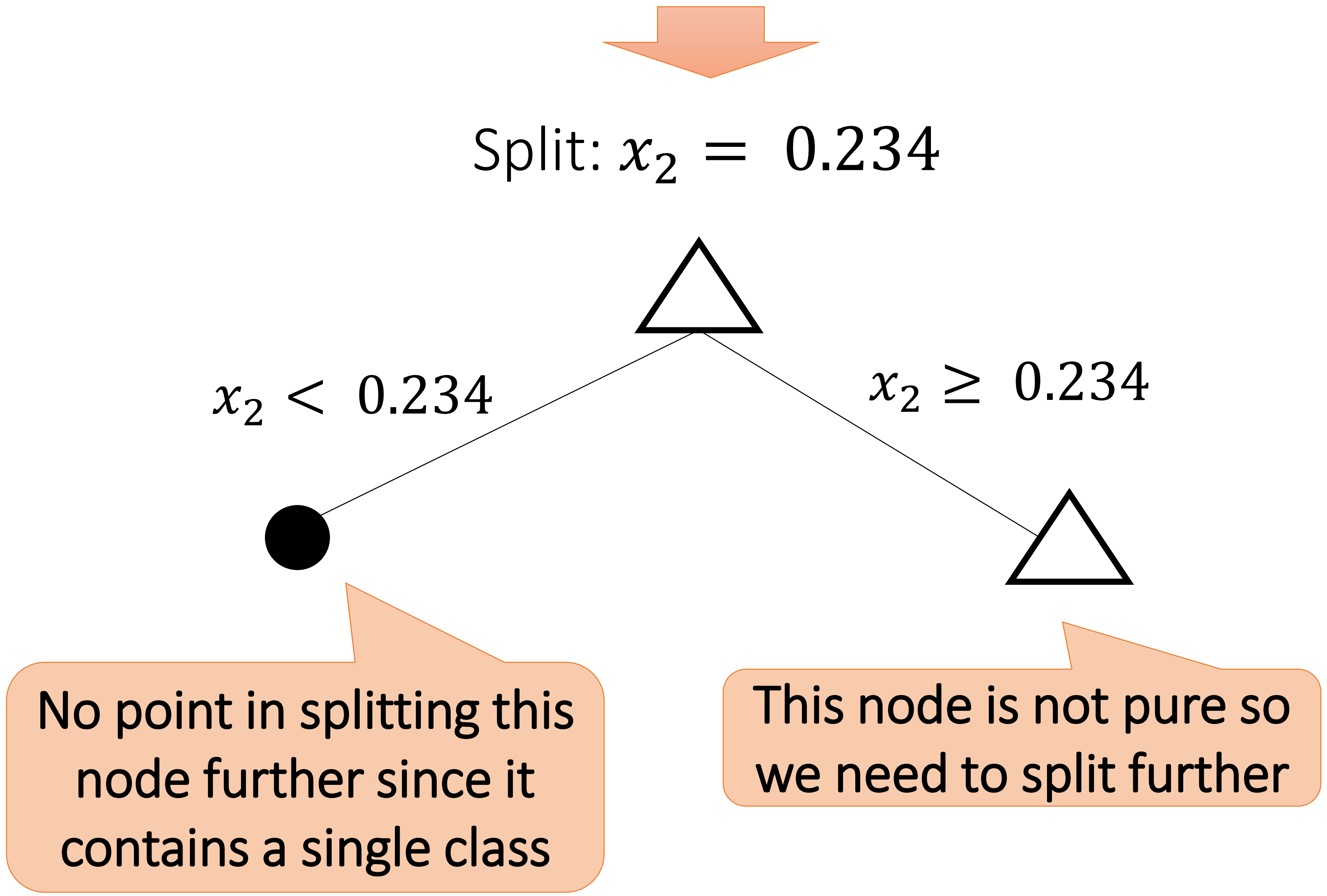
Total data points = 7
Class distribution:
7 (A) and 0 (B)

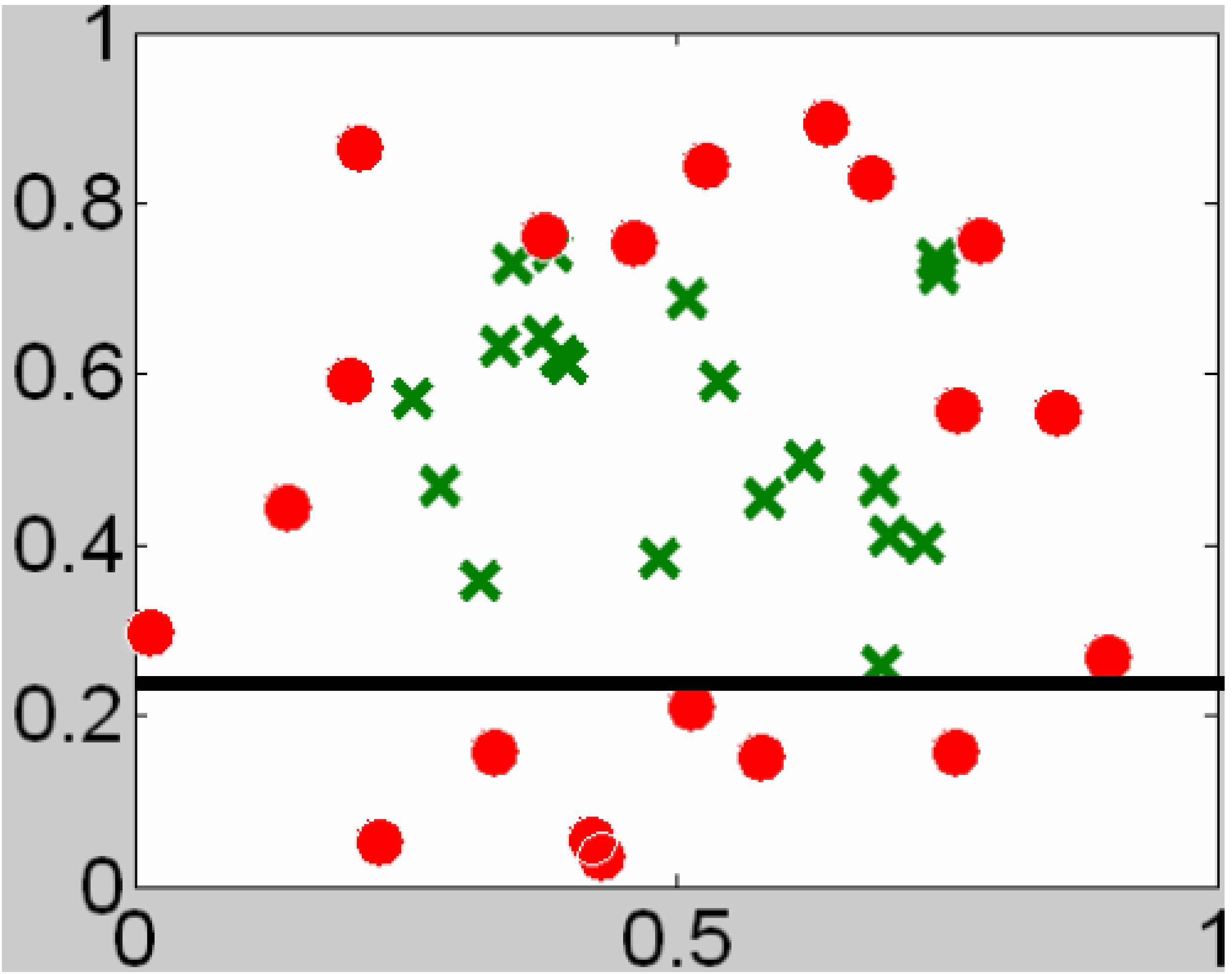
Total data points = 33
Class distribution:
13 (A) and 20 (B)



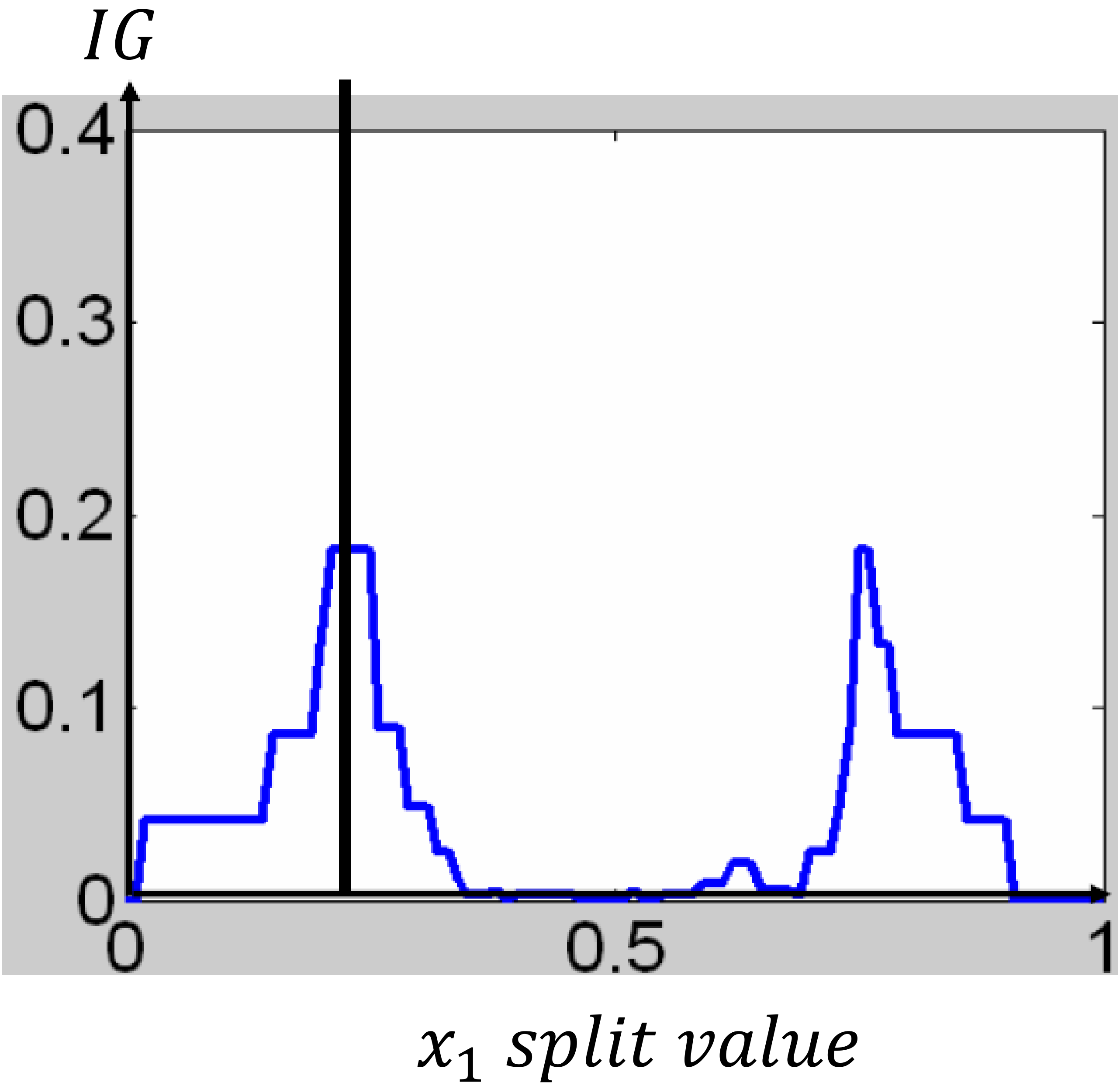
Complete example

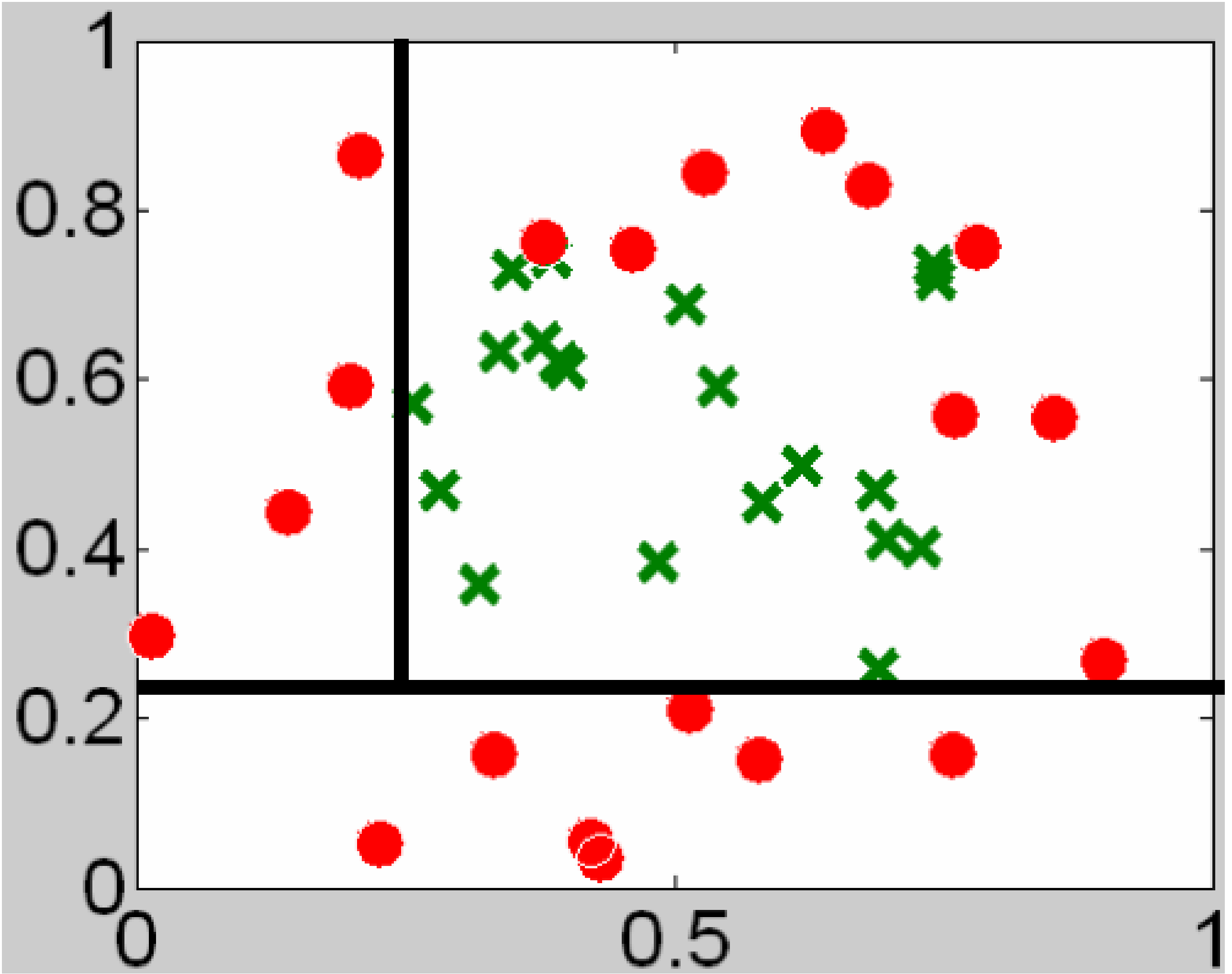
Best x_1 split: $x_1 = 0.24$ with $IG = 0.138$
Best x_2 split: $x_2 = 0.234$ with $IG = 0.202$





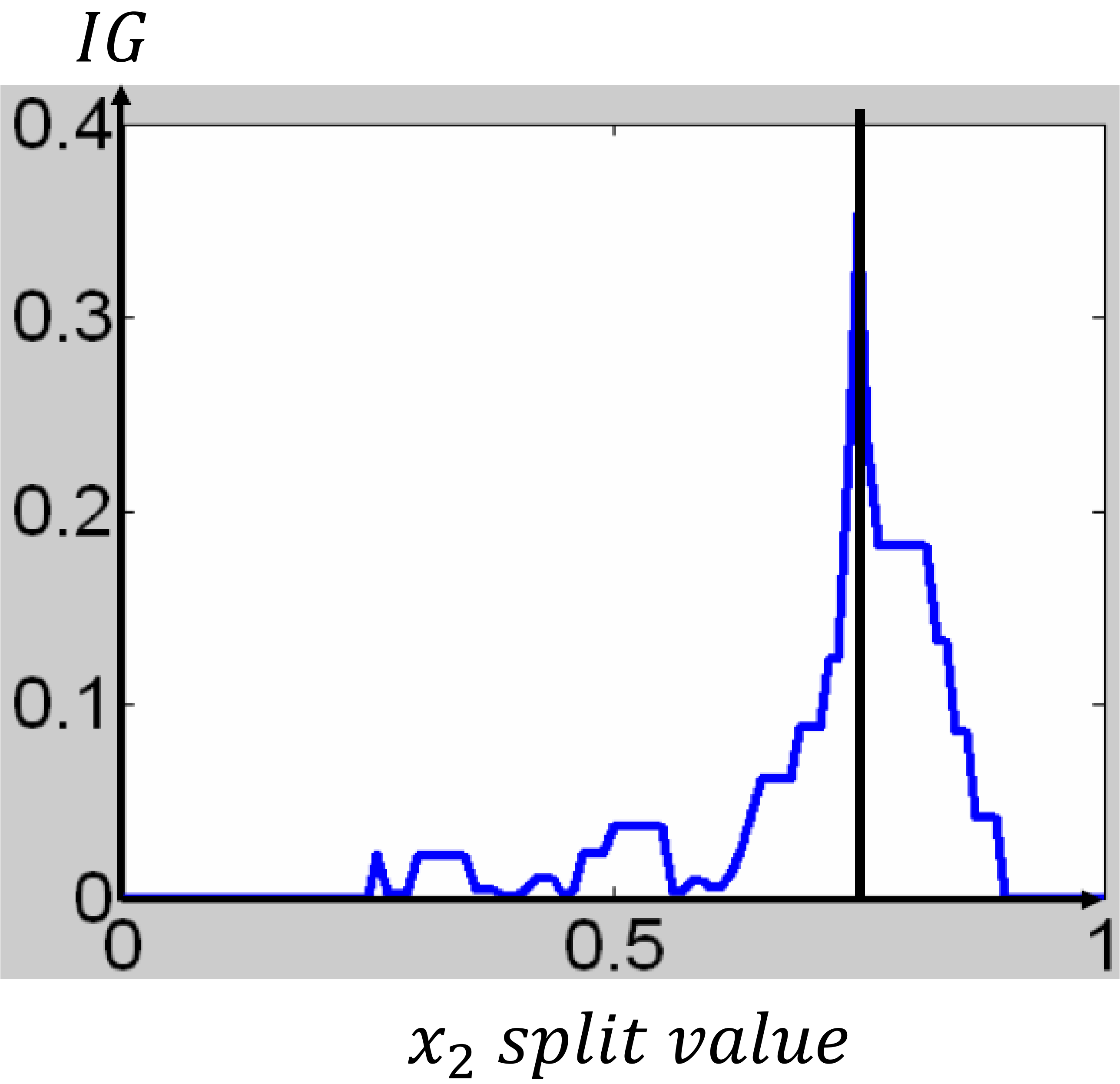
Complete example

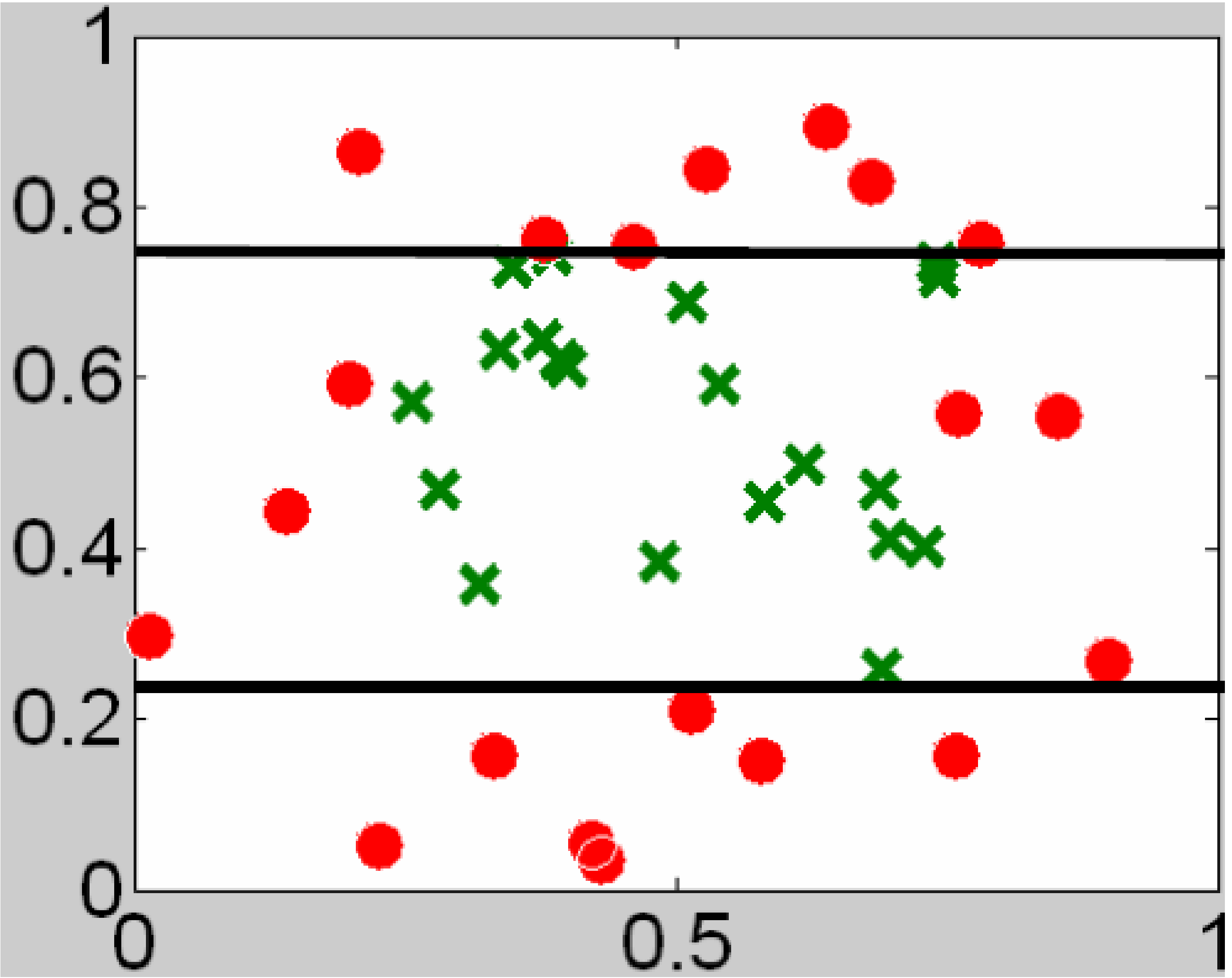




Best split value → maximum information gain for feature x_1:
$x_1 = 0.22$ with $IG = 0.182$

Complete example



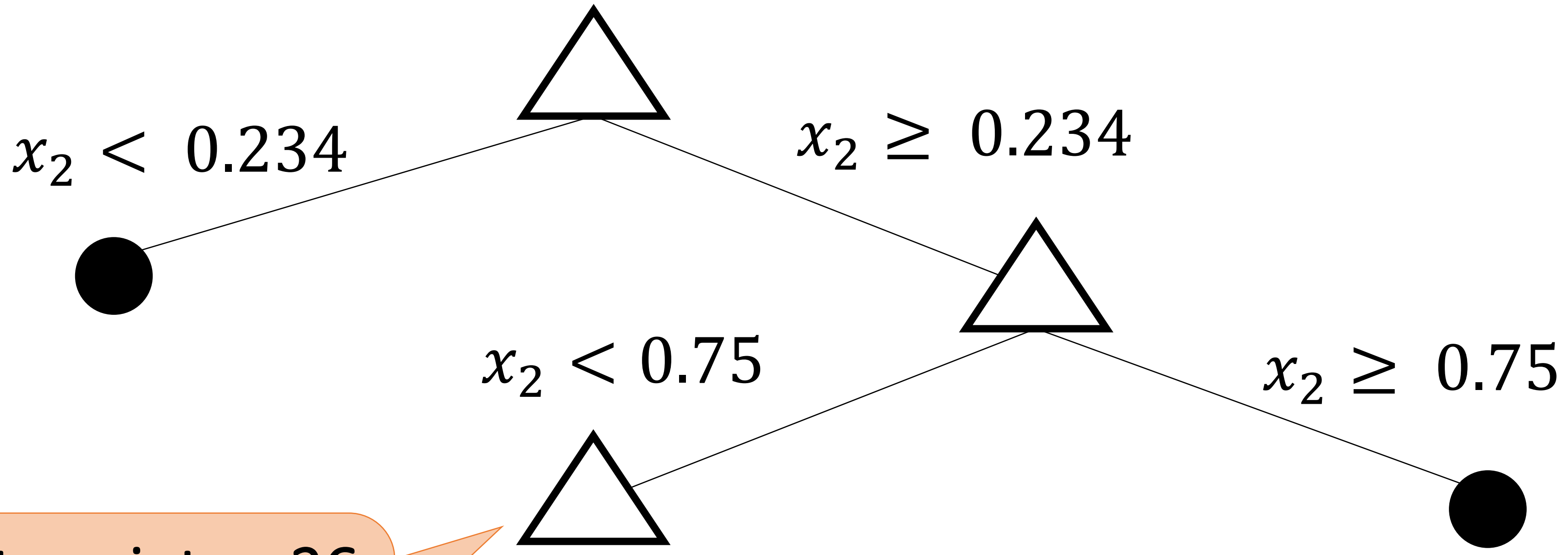


Best split value → maximum information gain for feature x_2:
$x_2 = 0.75$ with $IG = 0.356$

Complete example

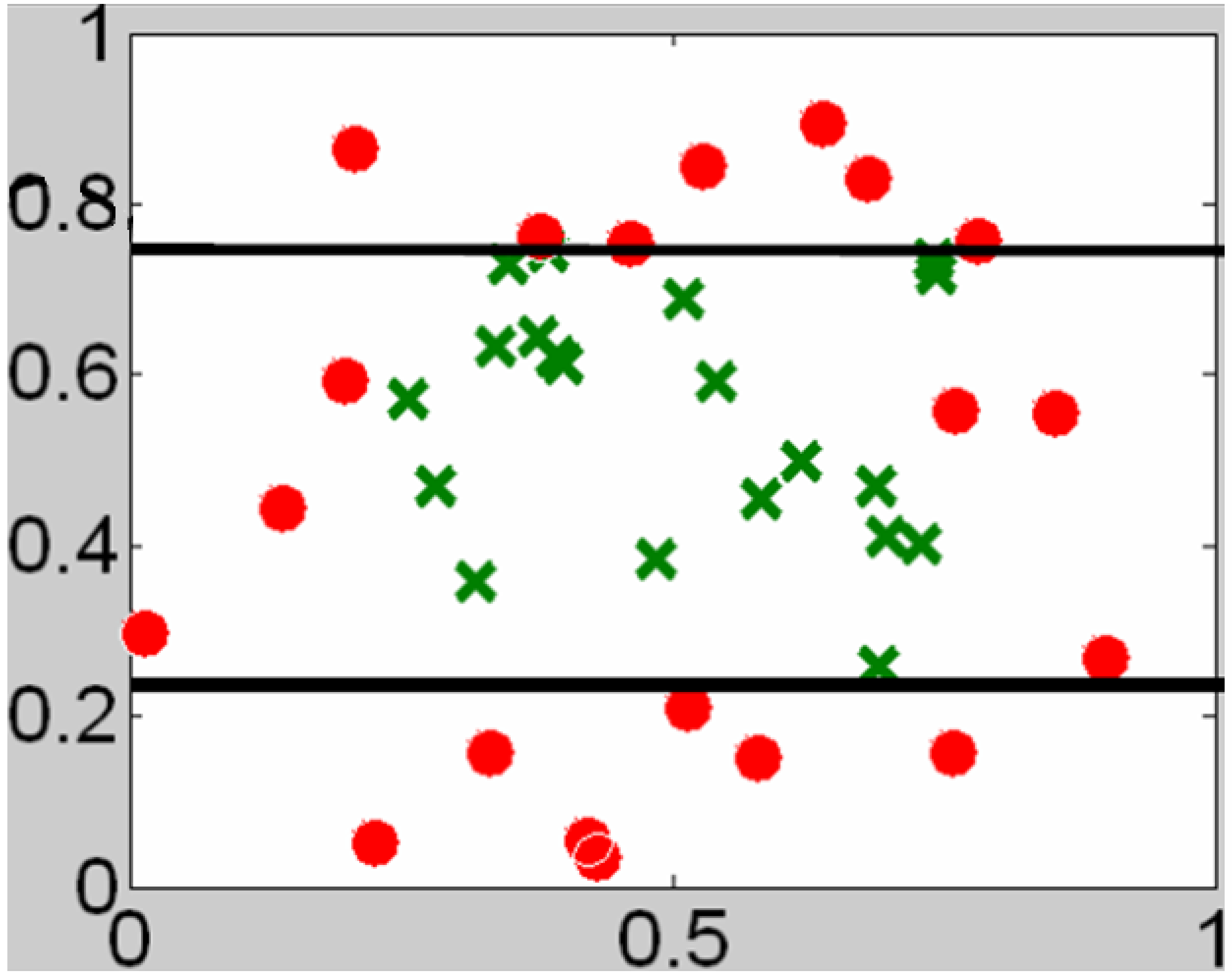
Best x_1 split: $x_1 = 0.22$ with $IG = 0.182$
Best x_2 split: $x_2 = 0.75$ with $IG = 0.353$

Split: $x_2 = 0.75$



Total data points = 26
Class distribution:
6 (A) and 20 (B)

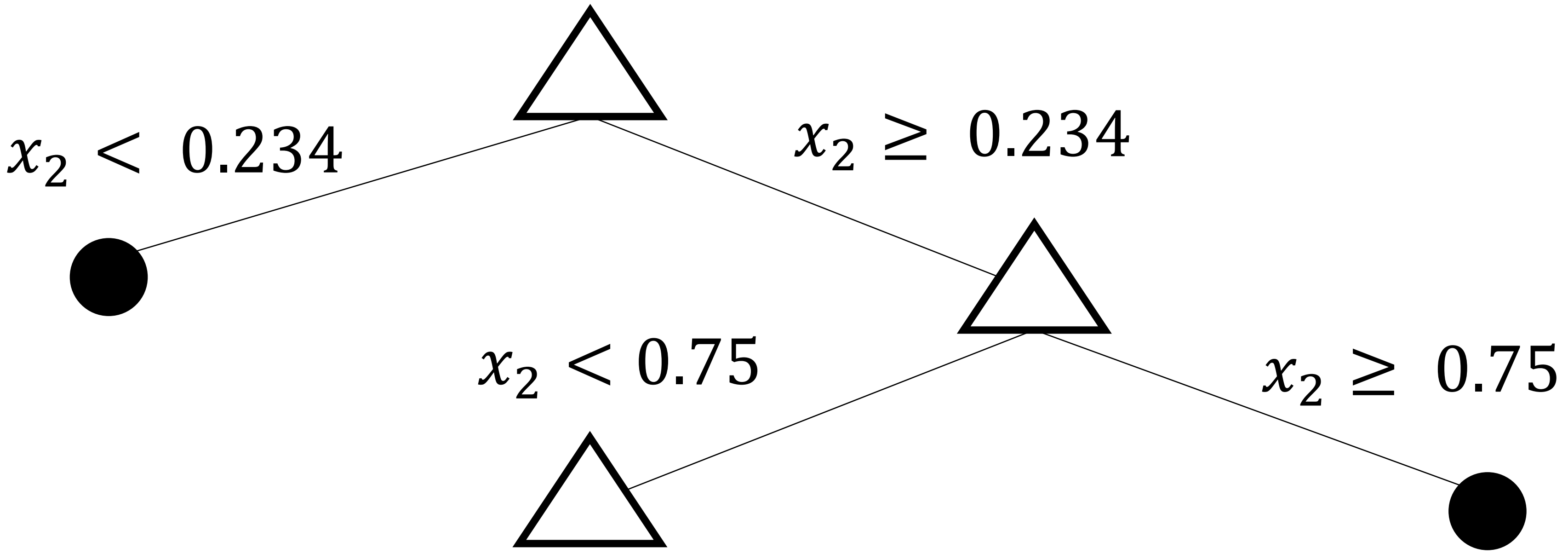
Total data points = 7
Class distribution:
7 (A) and 0 (B)



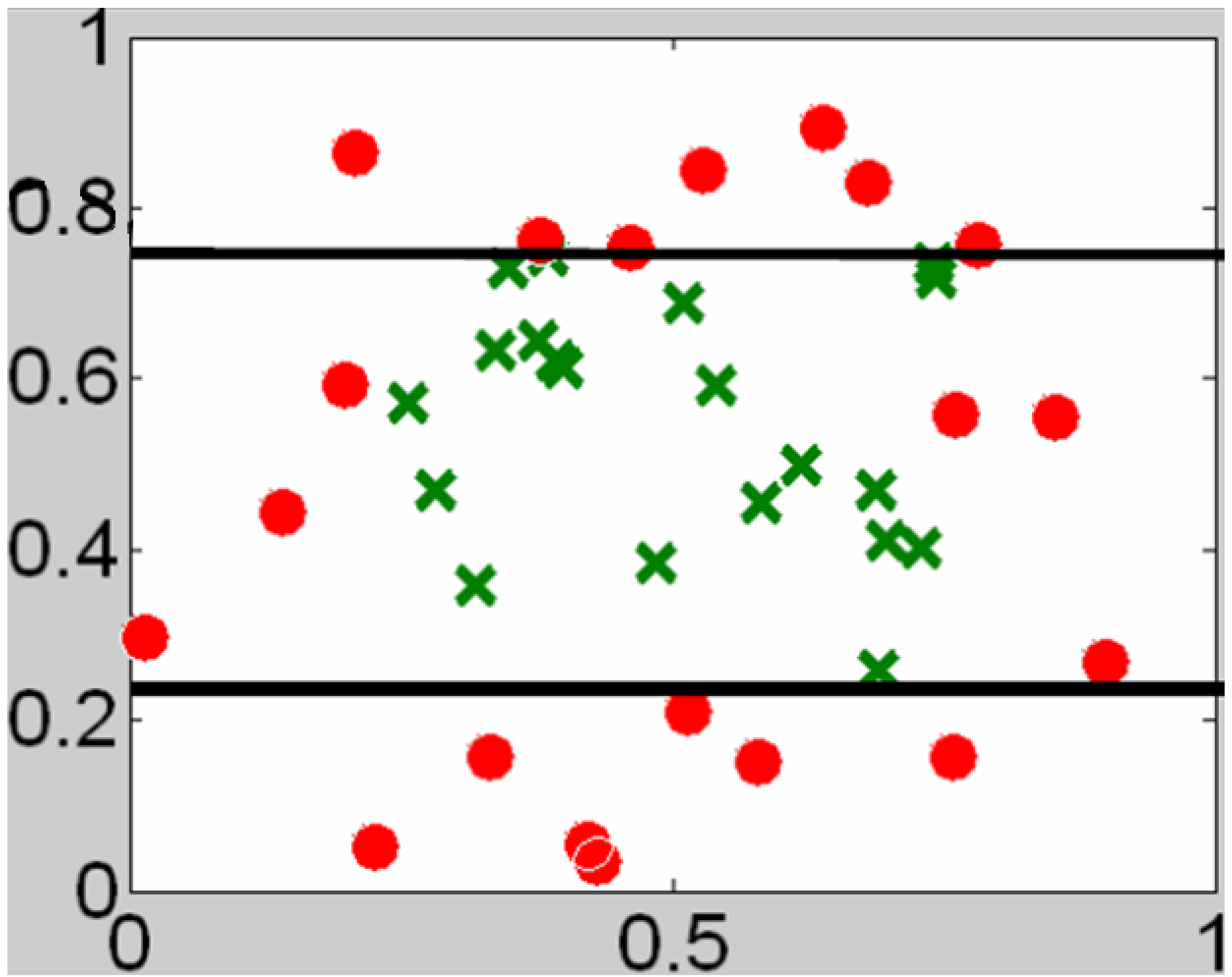
Complete example

Best x_1 split: $x_1 = 0.22$ with $IG = 0.182$
Best x_2 split: $x_2 = 0.75$ with $IG = 0.353$

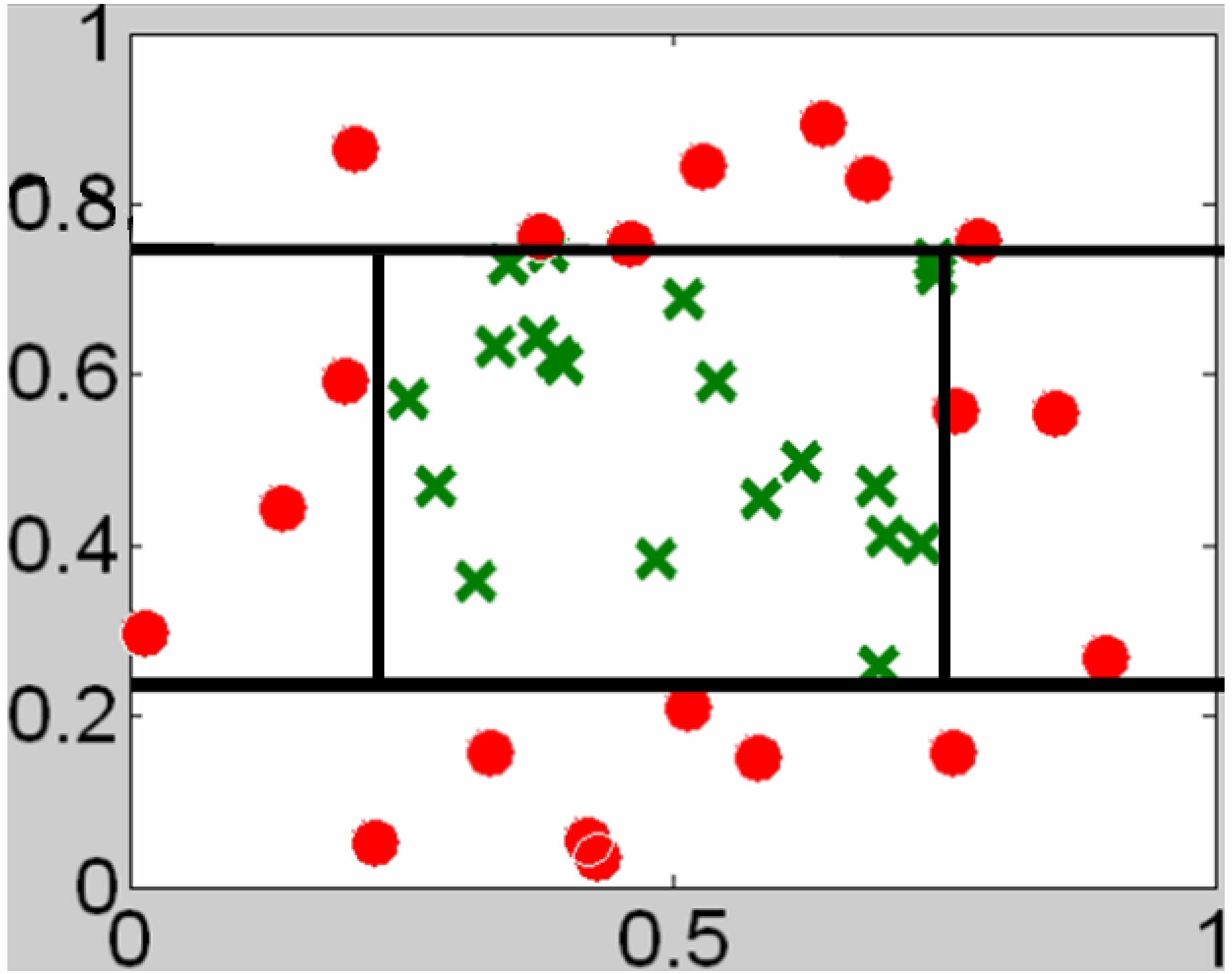
Split: $x_2 = 0.75$

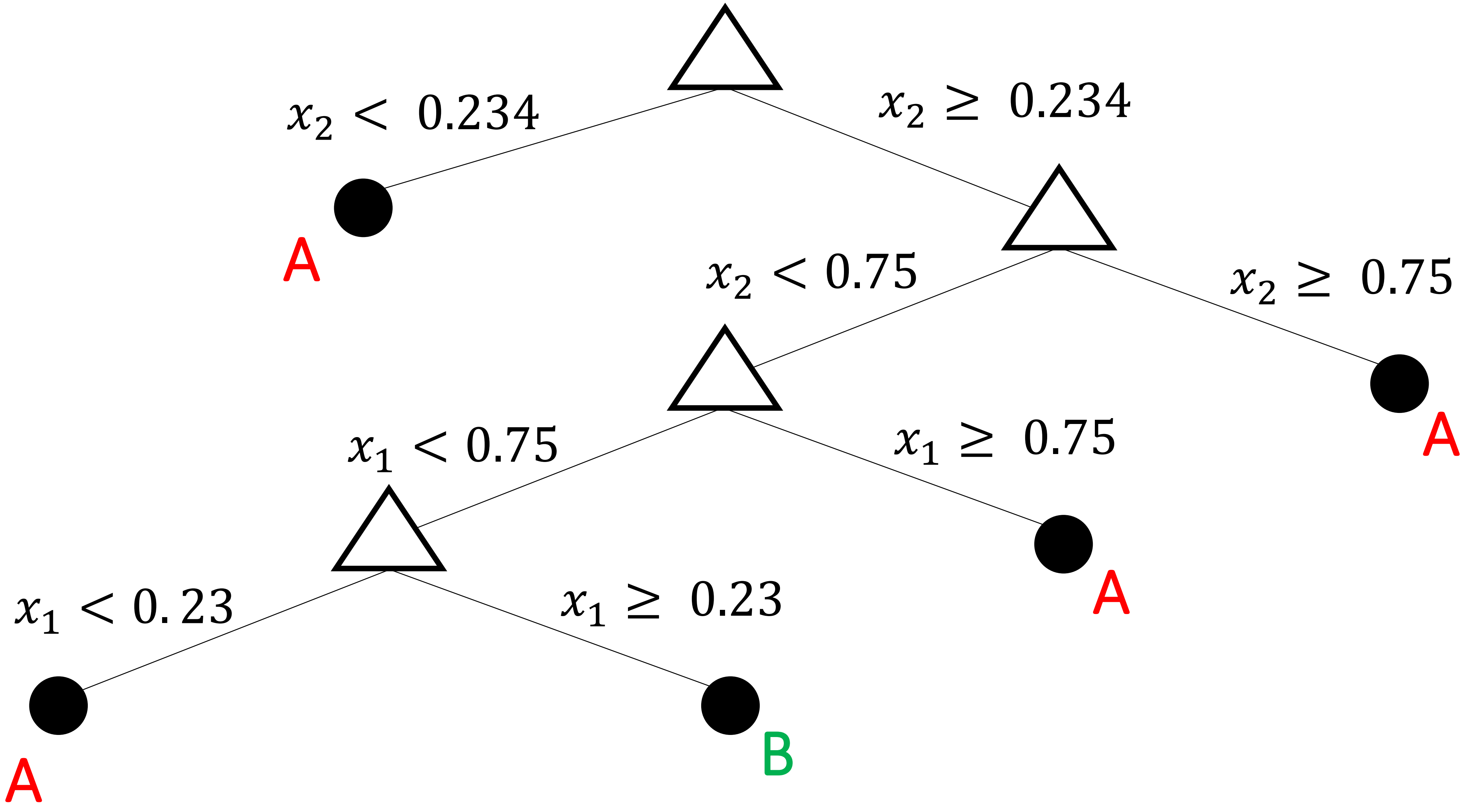


No point in splitting this node further since it contains a single class



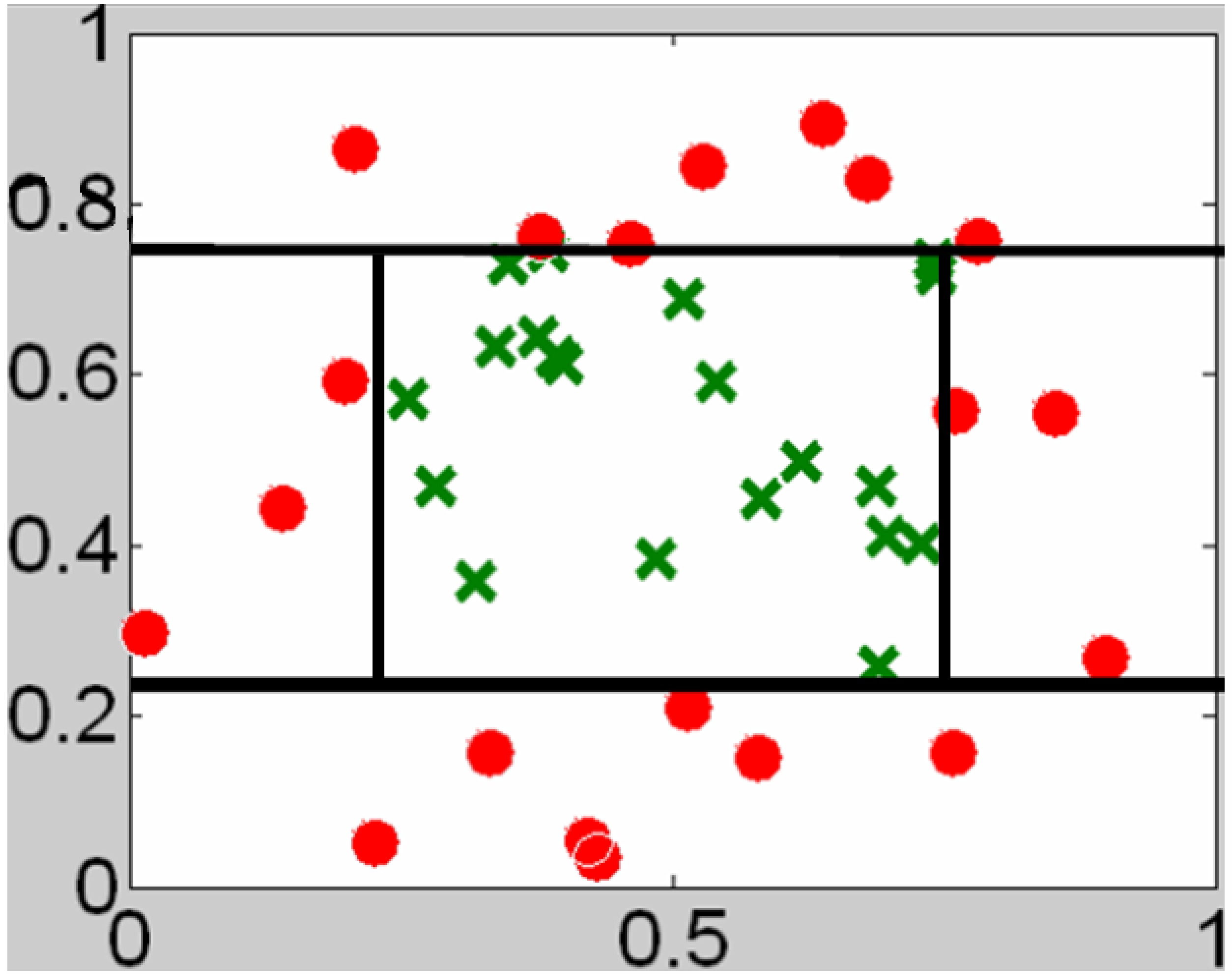
Final decision tree

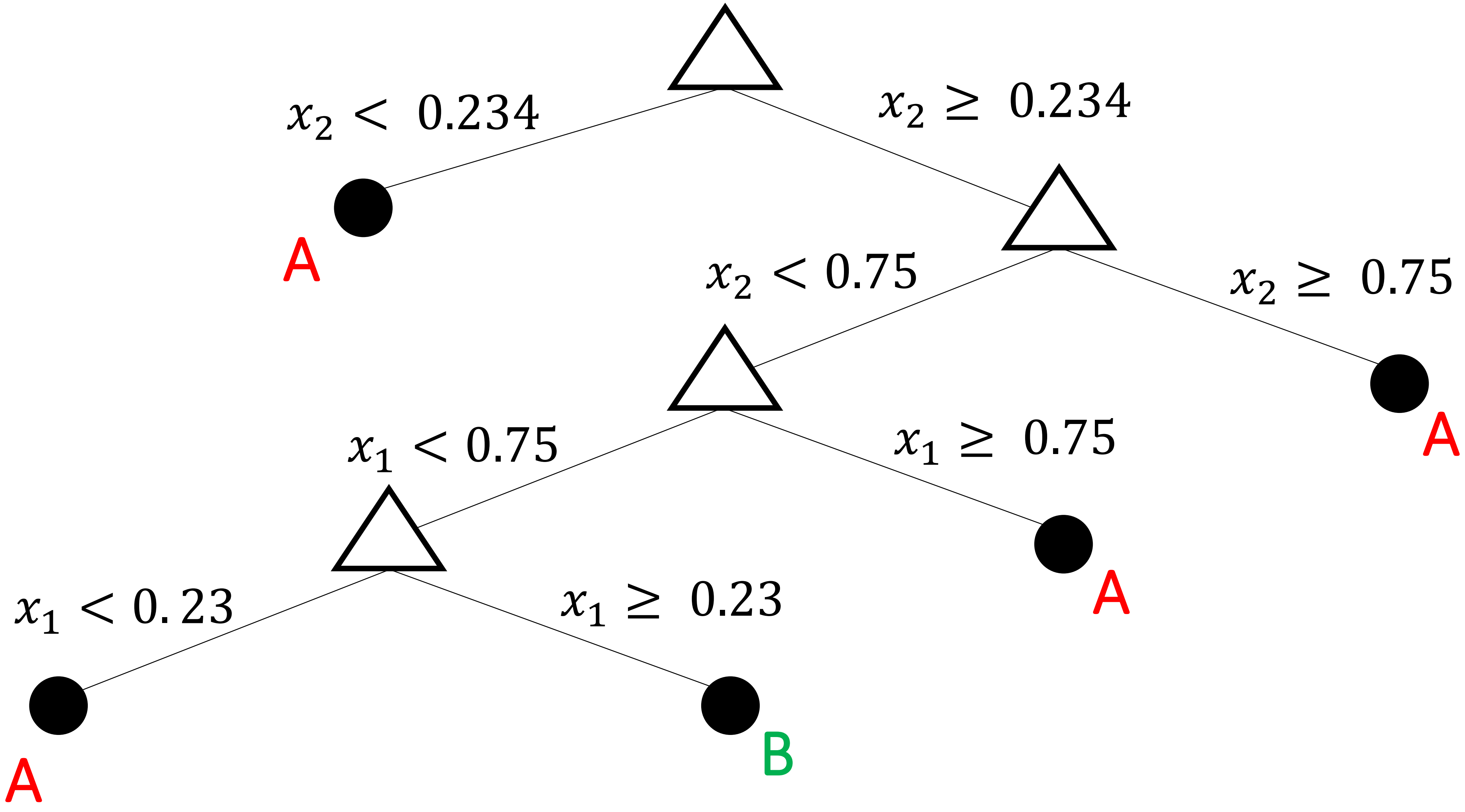




Each of the leaf node is pure → contains data from only one class

Testing new data point



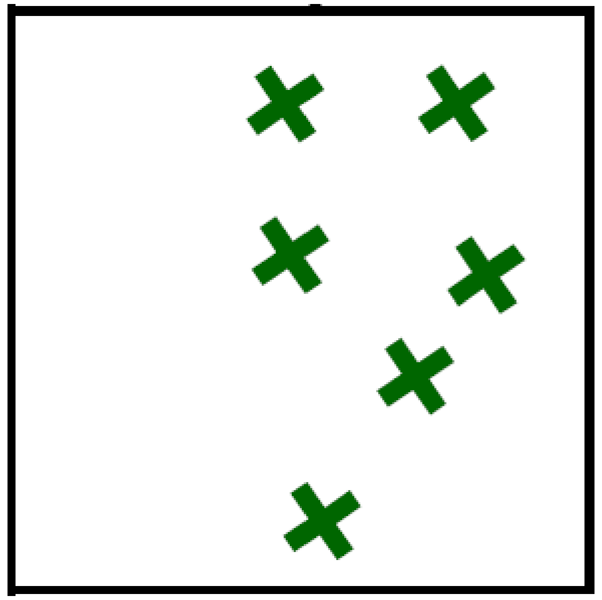


Given an input x_1, x_2 → follow the tree down to a leaf → return corresponding output class for this leaf

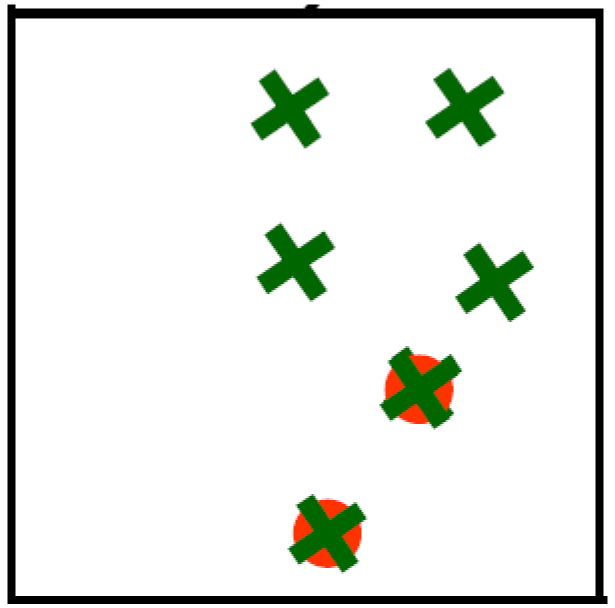
Important questions

- How to choose the attribute and value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

Pure and impure leaves and when to stop splitting



- All the data in the node comes from a single class → we declare the node to be a leaf and stop splitting. This leaf will output the class of the data it contains



- Several data points have exactly the same attributes even though they are from different classes → we cannot split any further → we still declare the node to be a leaf, but it will output the class that is the majority of the classes in the node (in this example, "B")

Outline

- Intuition
- Overview
- Learning a tree
- Algorithm

Decision tree algorithm: continuous features

- $LearnTree(\mathbf{X}, \mathbf{t})$
  - Input:
    - Set $\mathbf{X}$ of N training vectors, each containing the values of $(x_1, \ldots, x_D)$ of D features.
    - A vector $\mathbf{t}$ of N elements where t_n = class of the n-th datapoint
  - If all the datapoints in $\mathbf{X}$ have the same class value $t = k$
    - Return a leaf node that predicts $y = k$
  - If all the datapoints in $\mathbf{X}$ have the same feature value $(x_1, \ldots, x_D)$
    - Return a leaf node that predicts the majority of the class values $y = mode(\mathbf{t})$
  - Try all possible features x_d and thresholds val and choose the one, d^*, for which $IG(\mathbf{t}|x_d, val)$ is maximum
  - $\mathbf{X}_L, \mathbf{t}_L =$ set of datapoints for which $x_{d^*} < val$ and corresponding classes
  - $\mathbf{X}_R, \mathbf{t}_R =$ set of datapoints for which $x_{d^*} \geq val$ and corresponding classes
  - Left child $\leftarrow LearnTree(\mathbf{X}_L, \mathbf{t}_L)$
  - Right child $\leftarrow LearnTree(\mathbf{X}_R, \mathbf{t}_R)$

Decision tree algorithm: discrete features

- $LearnTree(\mathbf{X}, \mathbf{t})$
  - Input:
    - Set $\mathbf{X}$ of N training vectors, each containing the values of $(x_1, \dots, x_D)$ of D features.
    - A vector $\mathbf{t}$ of N elements where t_n = class of the n-th datapoint
  - If all the datapoints in $\mathbf{X}$ have the same class value $t = k$
    - Return a leaf node that predicts $y = k$
  - If all the datapoints in $\mathbf{X}$ have the same feature value $(x_1, \dots, x_D)$
    - Return a leaf node that predicts the majority of the class values $y = mode(\mathbf{t})$
  - Try all possible features x_d and choose the one, d^*, for which $IG(\mathbf{t}|x_d, val)$ is maximum:
  - For every possible value val of x_d:
    - $\mathbf{X}_{val}, \mathbf{t}_{val}$ = set of datapoints for which $x_{d^*} = val$ and corresponding classes
    - Child ← $LearnTree(\mathbf{X}_{val}, \mathbf{t}_{val})$