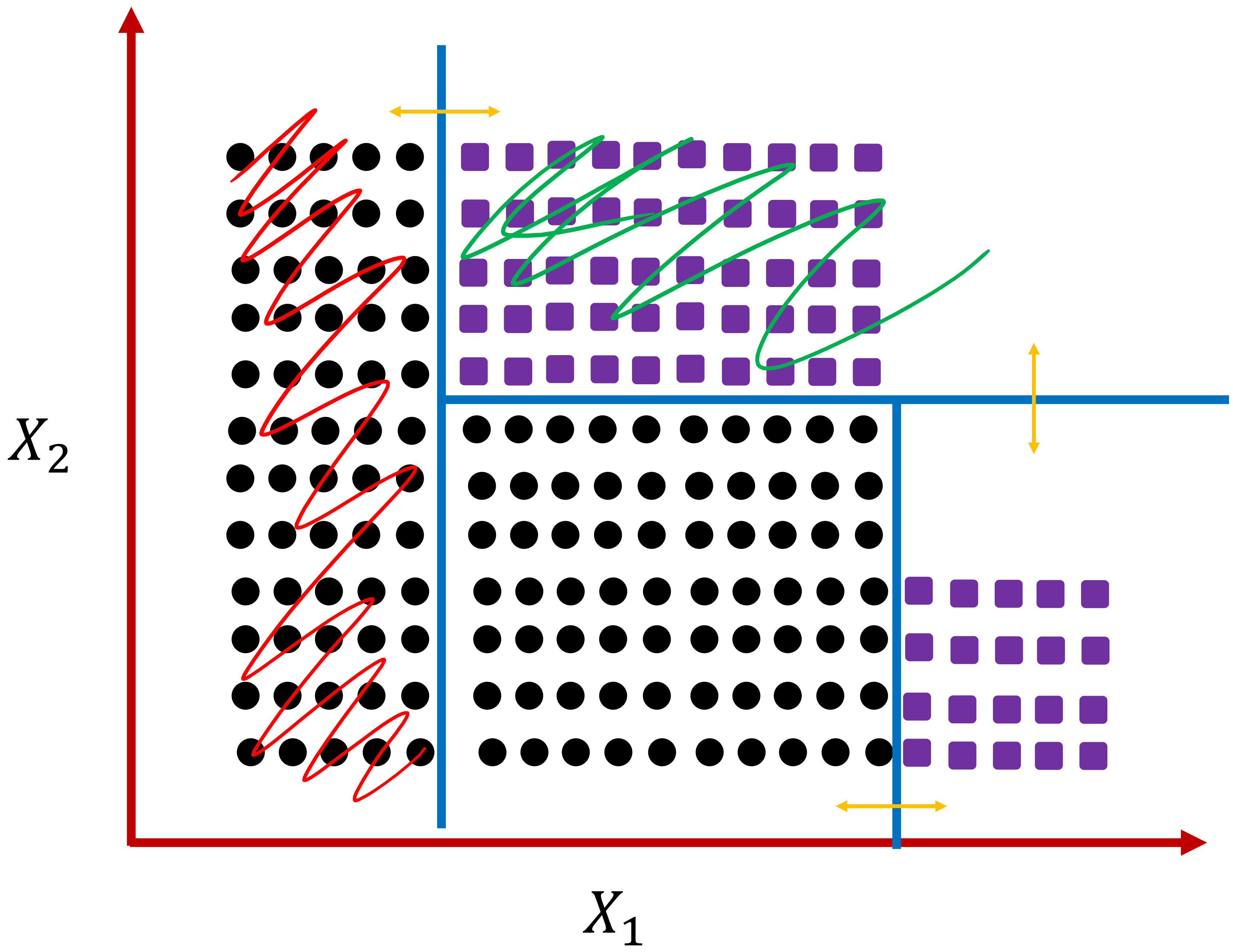
Decision Tree

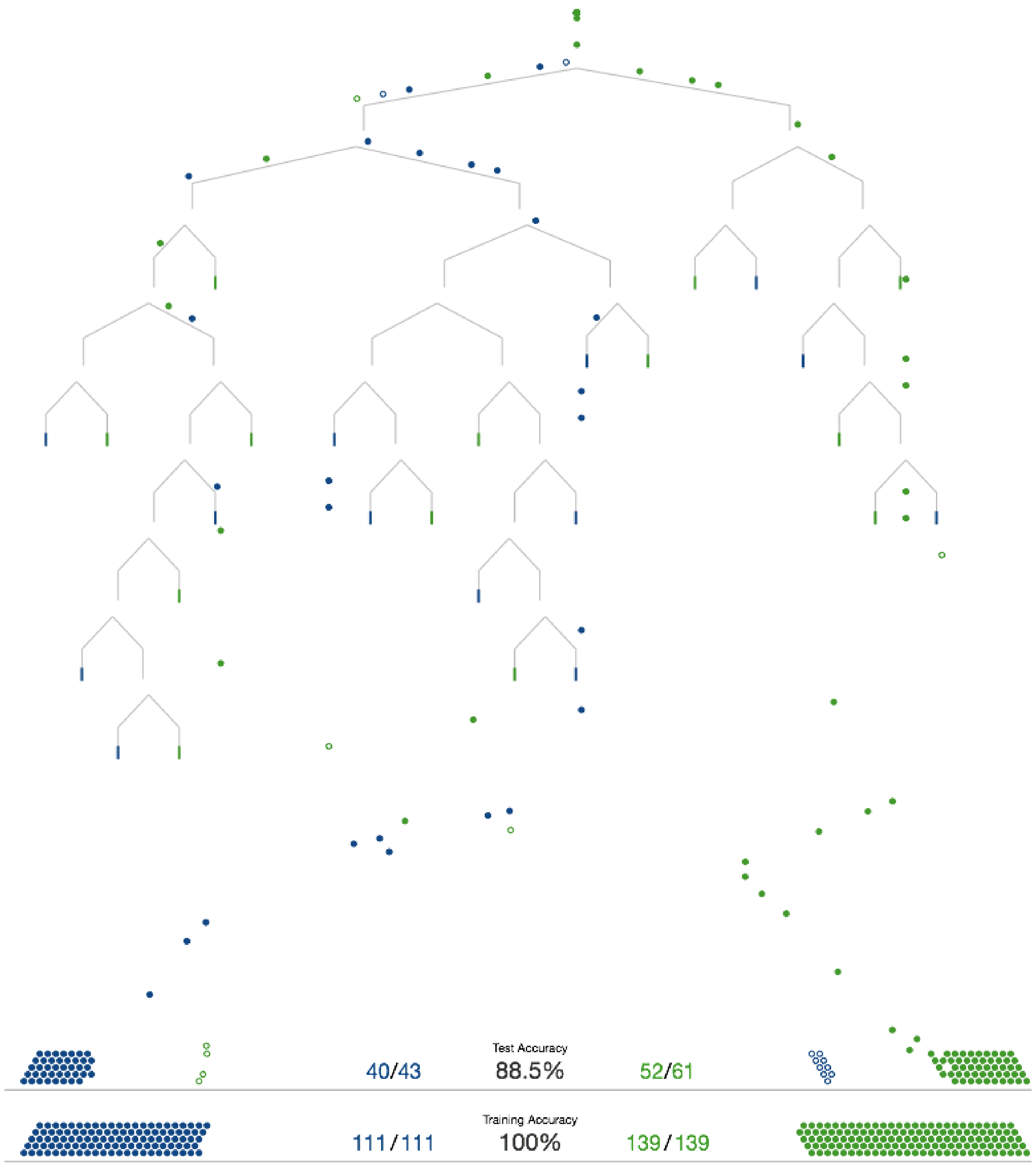
Mahdi Roozbahani
Georgia Tech

These slides are adopted from Polo, Vivek Srikumar, and Chao Zhang.


Adding new feature to the existing code 😂😂


X_2
X_1

Visual Introduction to Decision Tree



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

Decision Tree: Example (2)

	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

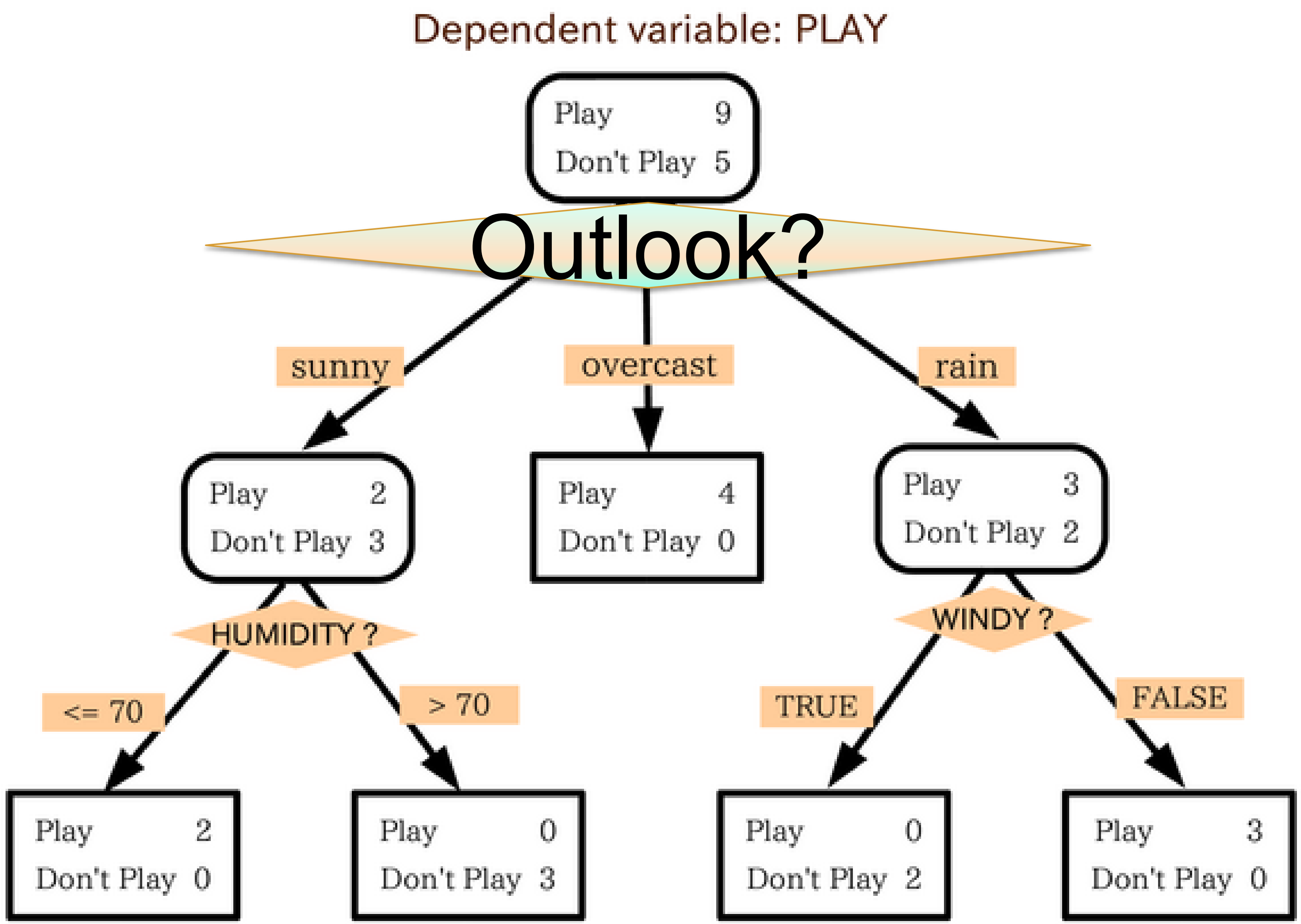
Temperature: Hot, Medium, Cool

Humidity: High, Normal, Low

Wind: Strong, Weak

Will I play tennis today?

Decision trees (DT)



The classifier:

$f_T(x)$: majority class in the leaf in the tree T containing x

Model parameters: The tree structure and size

$X = \begin{bmatrix} \quad \end{bmatrix}_{n \times d}$ $\quad Y = \begin{bmatrix} \text{cat} \\ \text{dog} \end{bmatrix}$ → label

Decision trees

$\boxed{\text{Attribute} = \text{feature} = \text{dimension} = \text{Variable}}$

Pieces:

1. Find the best attribute to split on

2. Find the best split on the chosen attribute

3. Decide on when to stop splitting

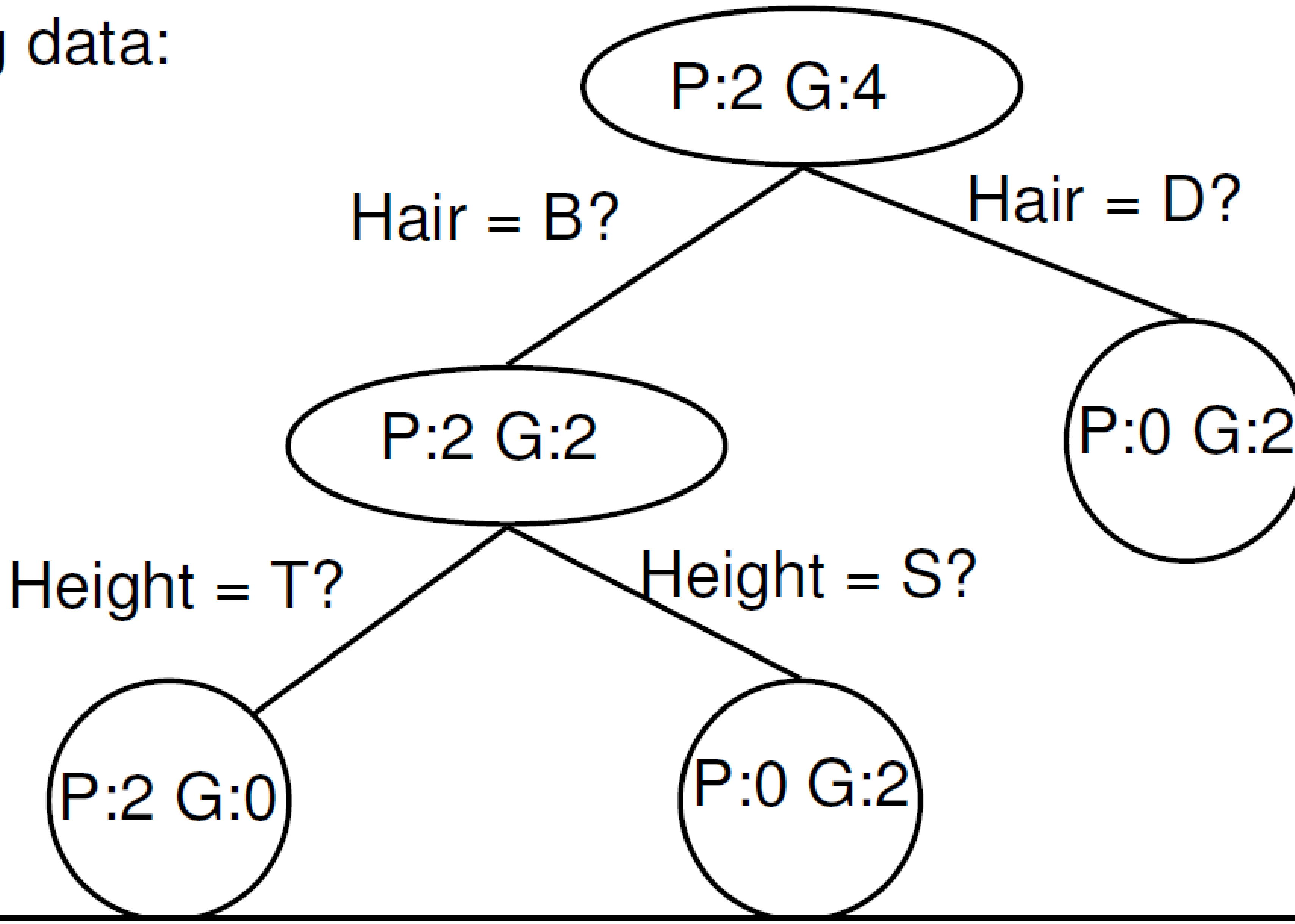
Categorical or Discrete attributes

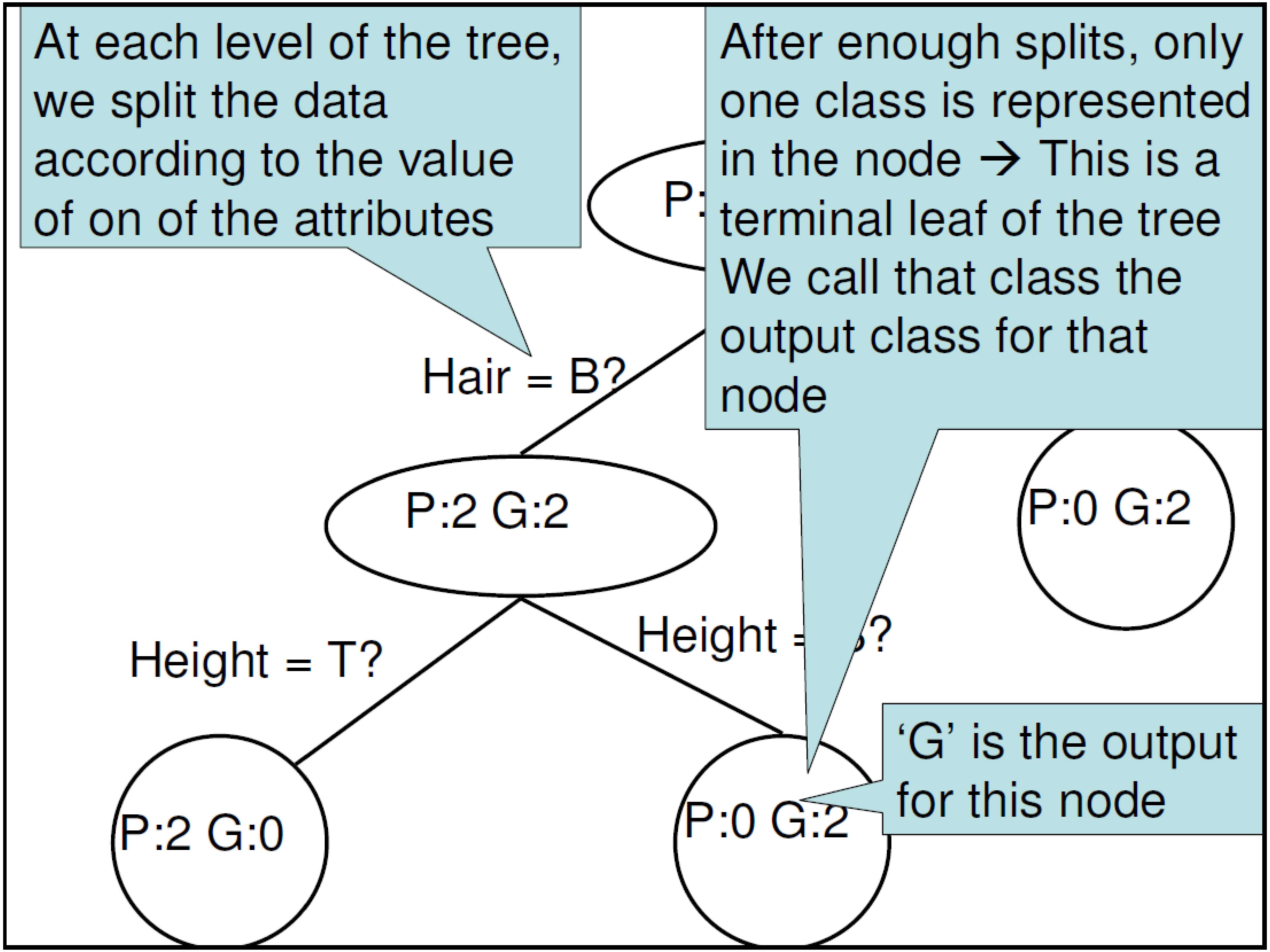
- Three variables:
  - Hair = {blond, dark}
  - Height = {tall,short}
  - Country = {Gromland, Polvia}

Label

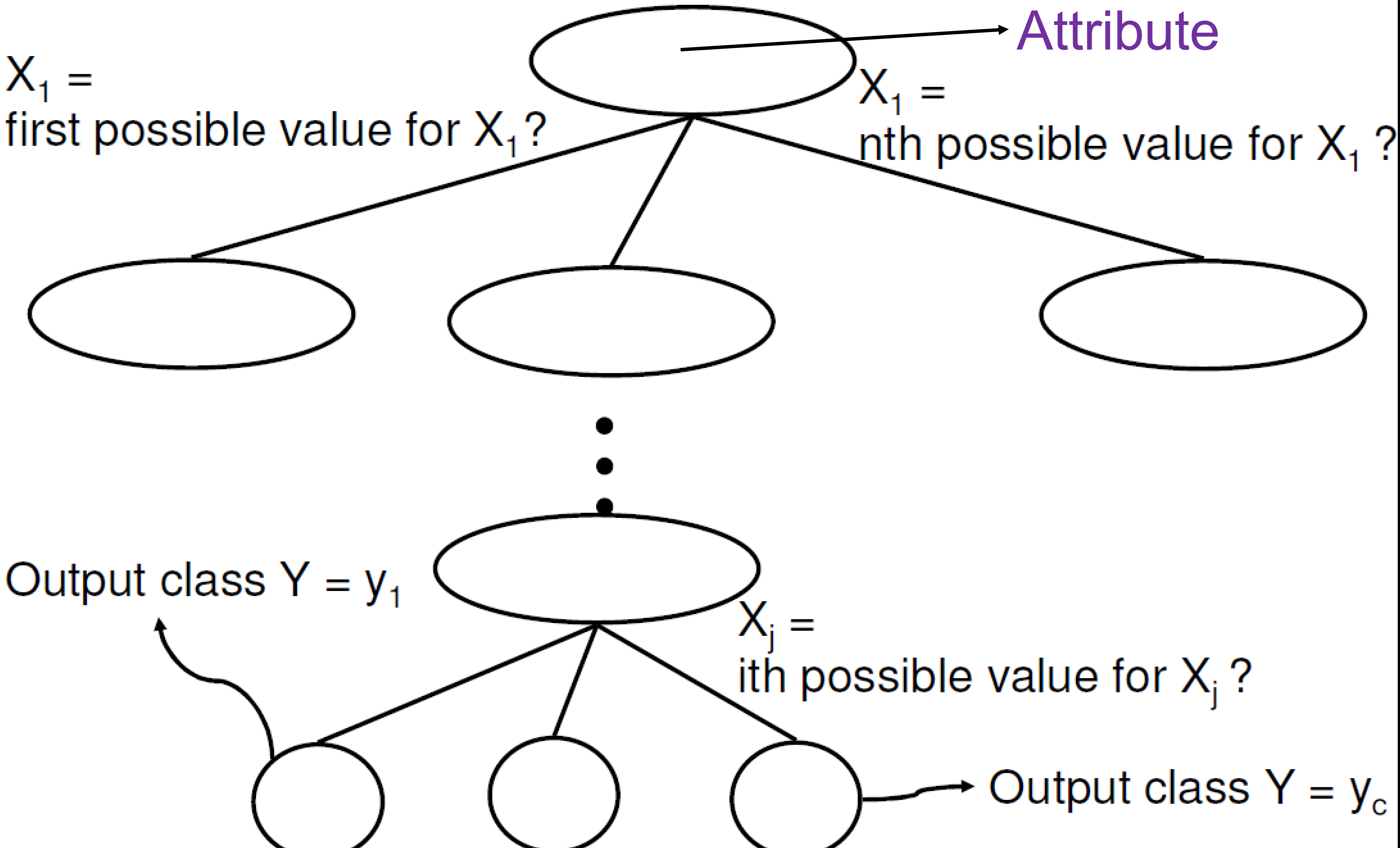
Training data:

(B,T,P)
(B,T,P)
(B,S,G)
(D,S,G)
(D,T,G)
(B,S,G)

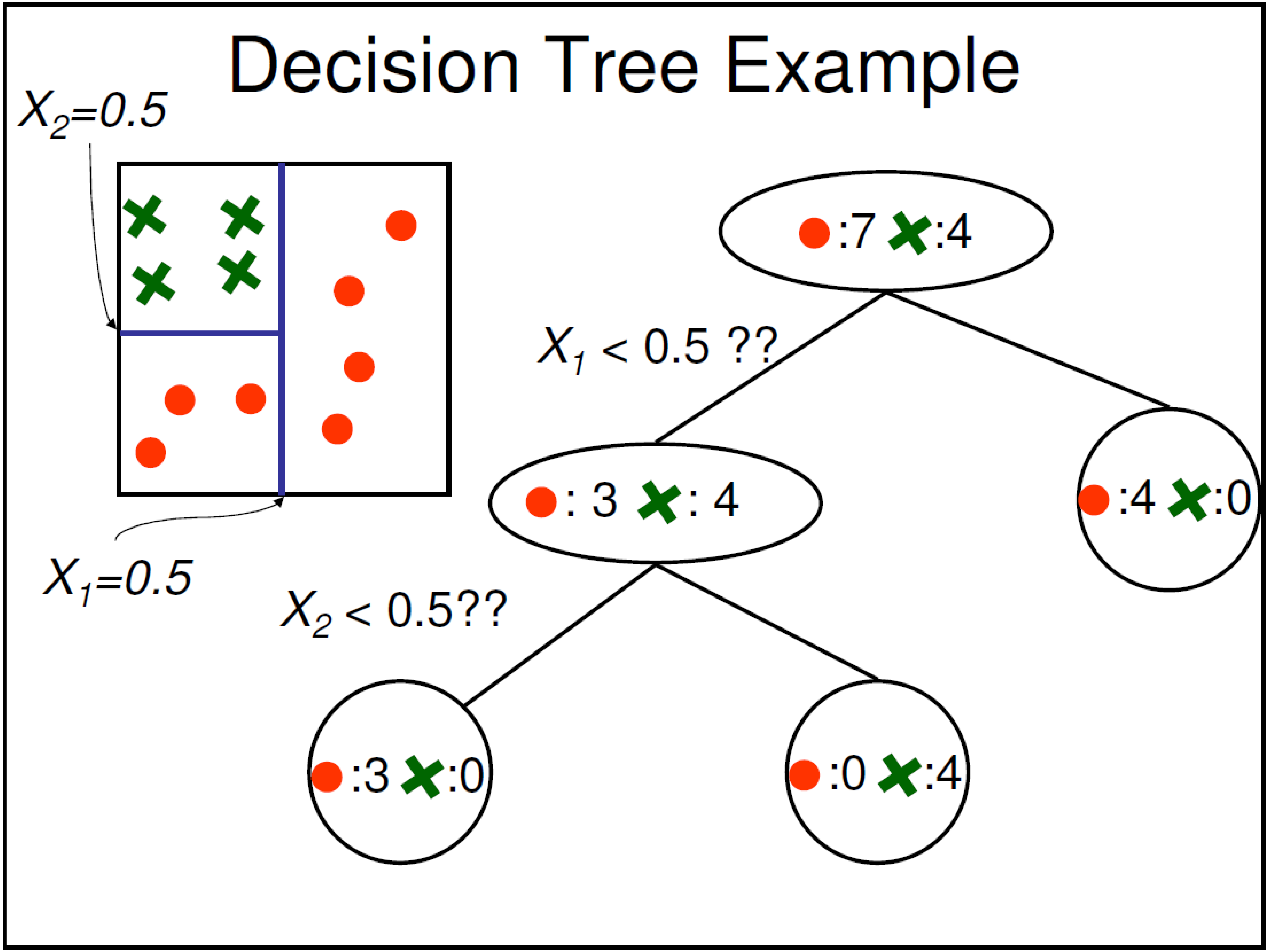



At each level of the tree, we split the data according to the value of on of the attributes
After enough splits, only one class is represented in the node → This is a terminal leaf of the tree We call that class the output class for that node
Hair = B?
P:2 G:2
P:0 G:2
Height = T?
P:2 G:0
P:0 G:2
'G' is the output for this node

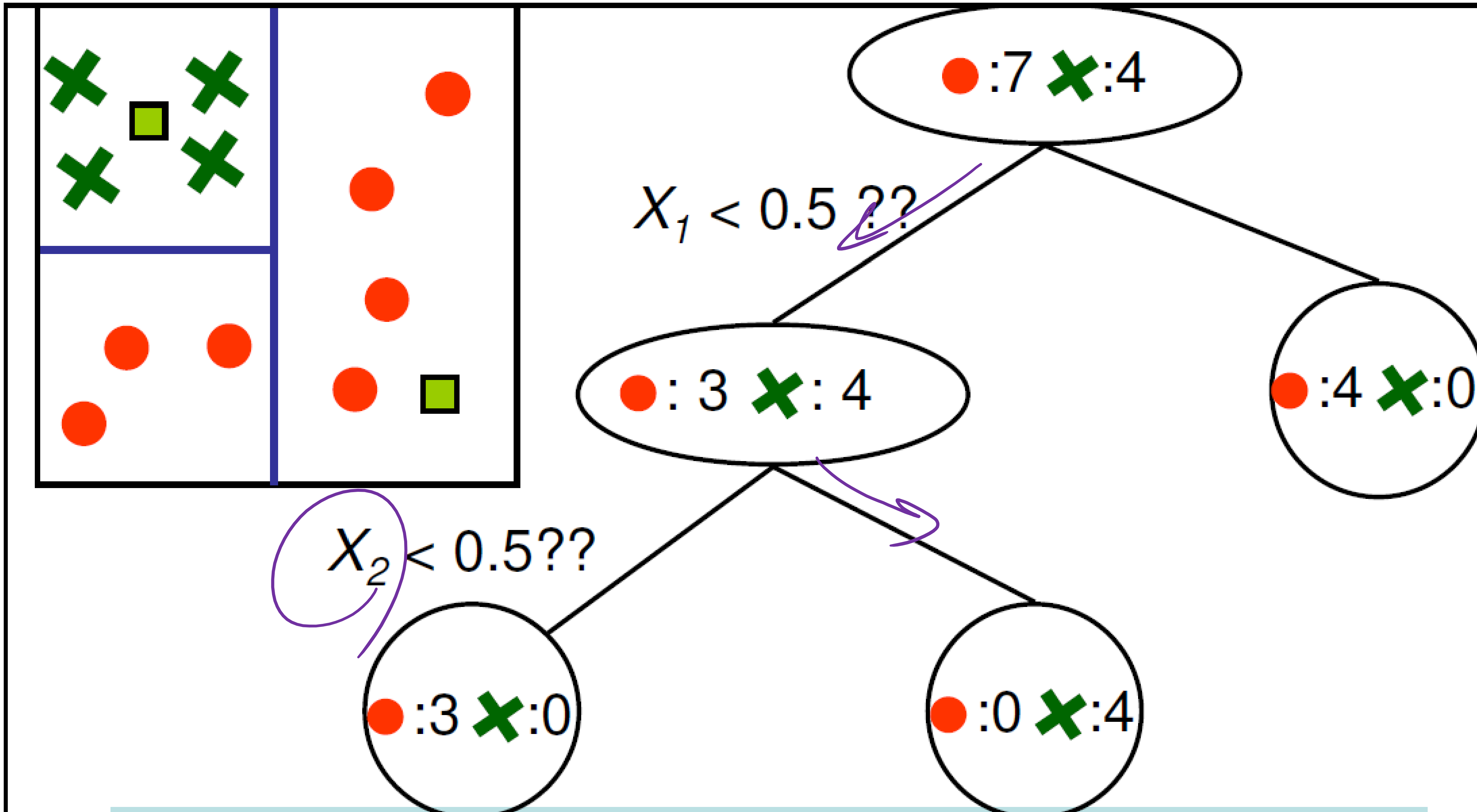
General Decision Tree (Discrete Attributes)



Continuous attributes



Test data

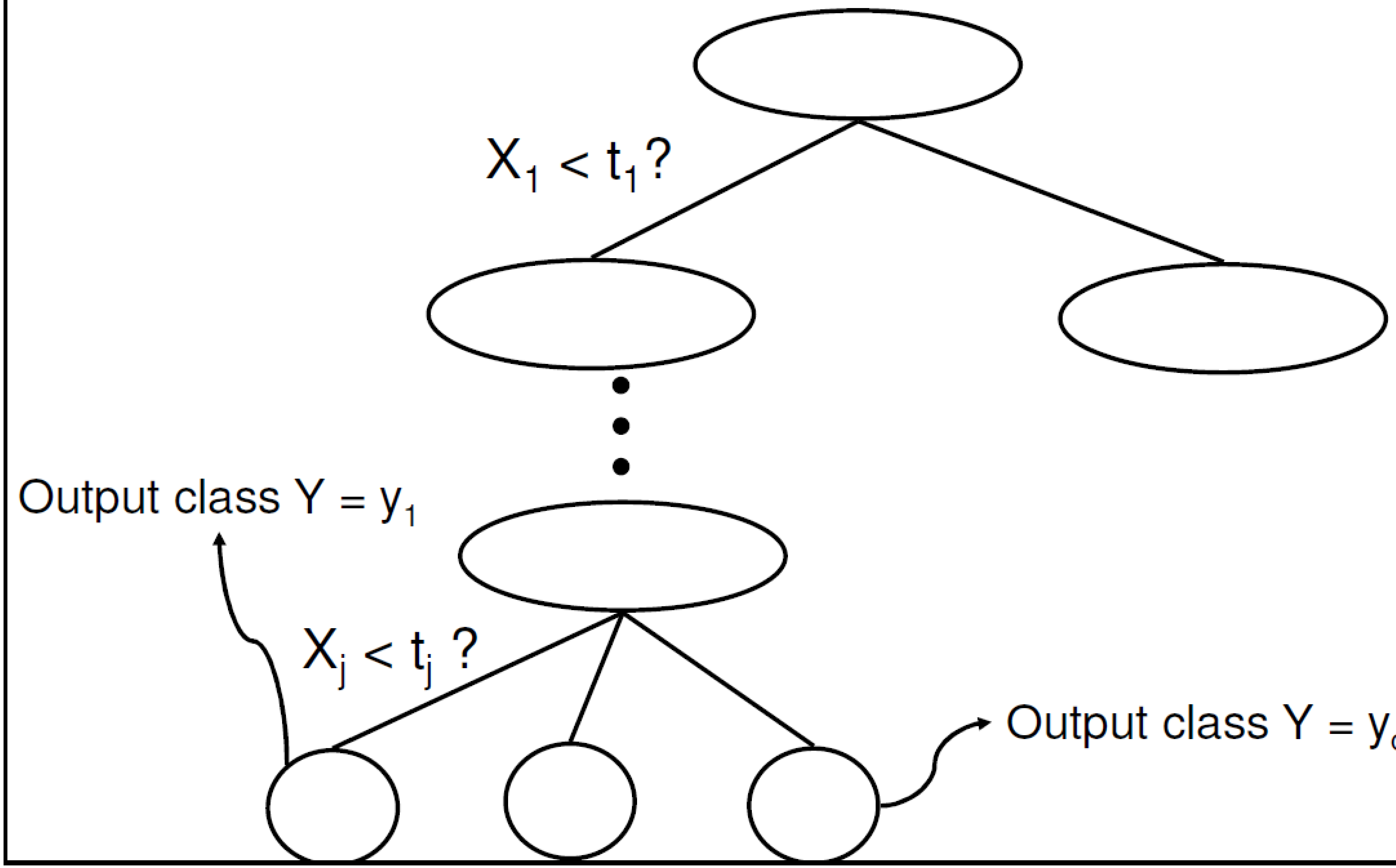


The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:

$x^{\{1\}}$ (0.2,0.8) is classified as

$x^{\{2\}}$ (0.8,0.2) is classified as

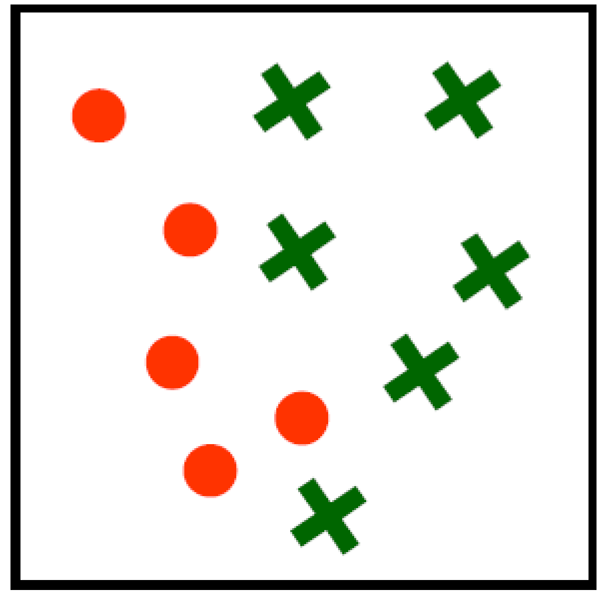
General Decision Tree (Continuous Attributes)



Basic Questions

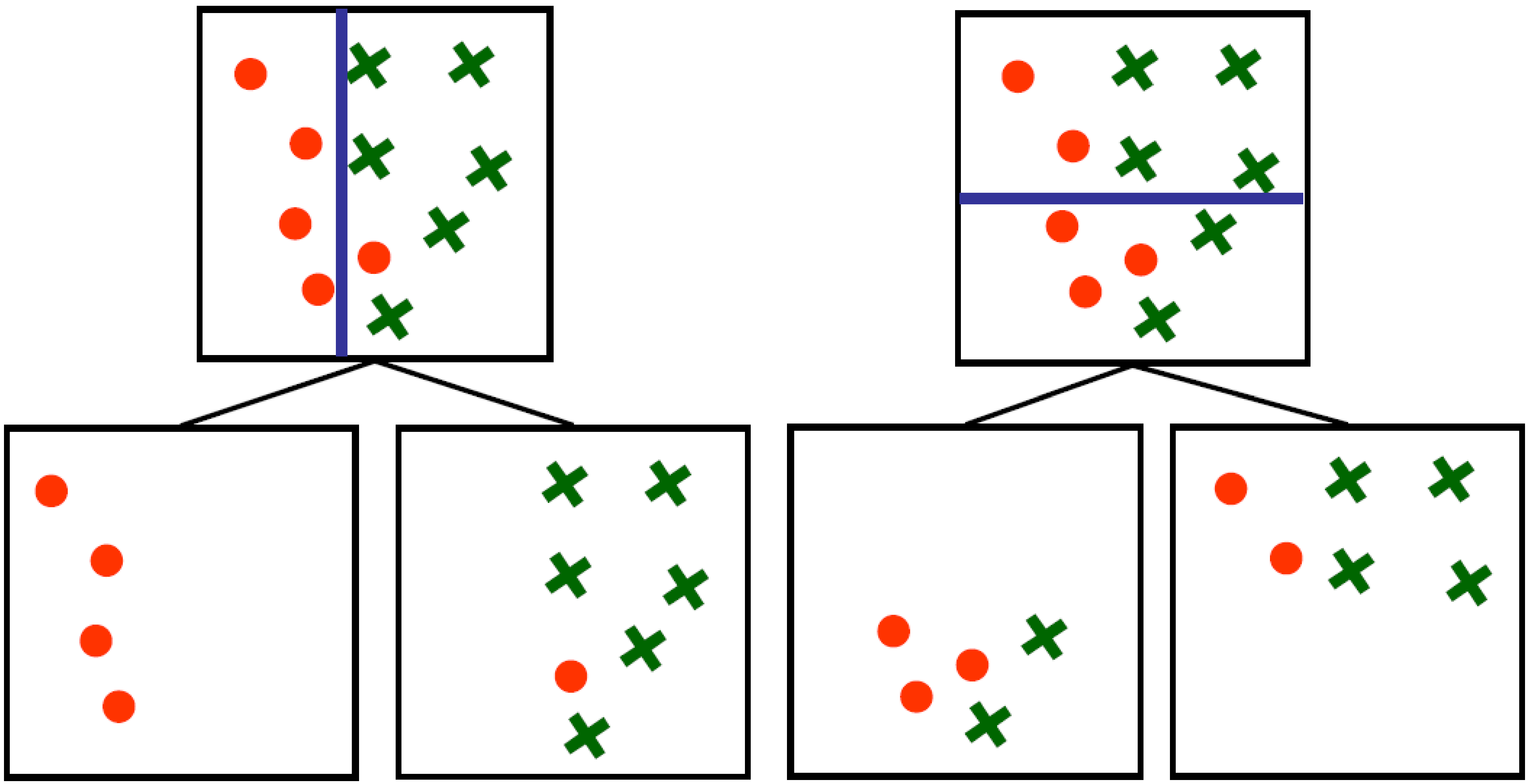
- How to choose the attribute/value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

How to choose the attribute/value to split on at each level of the tree?



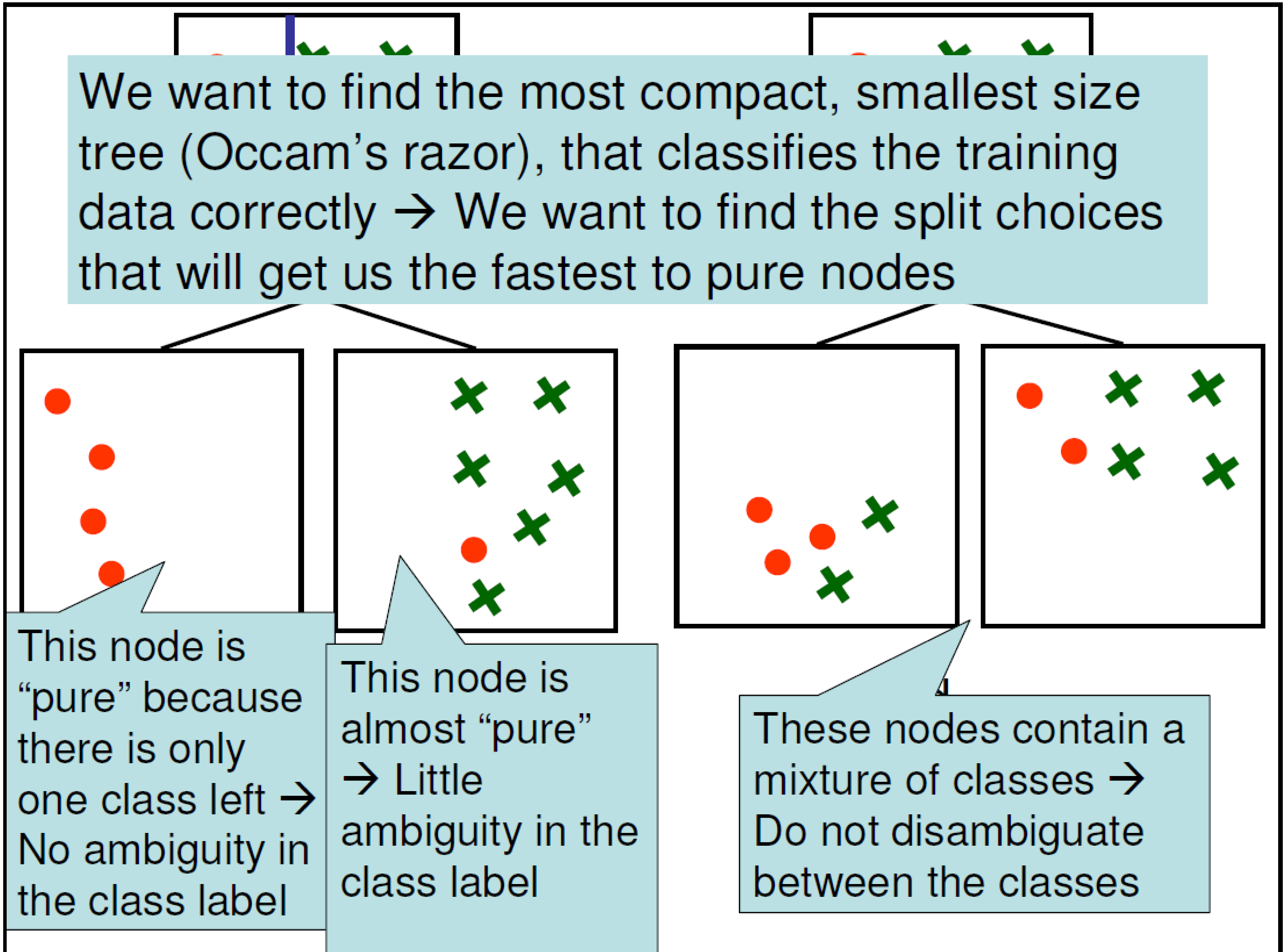
- Two classes (red circles/green crosses)
- Two attributes: X_1 and X_2
- 11 points in training data
- Idea → Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples

How to choose the attribute/value to split on at each level of the tree?



Good

Bad


We want to find the most compact, smallest size tree (Occam's razor), that classifies the training data correctly → We want to find the split choices that will get us the fastest to pure nodes
This node is "pure" because there is only one class left → No ambiguity in the class label
This node is almost "pure" → Little ambiguity in the class label
These nodes contain a mixture of classes → Do not disambiguate between the classes

Information Content

$I(x) = \log_2 \frac{1}{p(x)}$

$H(x) = \sum p(x) I(x) = \sum p(x) \log_2 \frac{1}{p(x)}$

Coin flip

C_{1H}	0
C_{1T}	6

P(C_{1H}) = 0/6 = 0

P(C_{1T}) = 6/6 = 1

C_{2H}	1
C_{2T}	5

P(C_{2H}) = 1/6

P(C_{2T}) = 5/6

C_{3H}	2
C_{3T}	4

P(C_{3H}) = 2/6

P(C_{3T}) = 4/6

Which coin will give us the purest information?

Entropy ~ Uncertainty

Lower uncertainty, higher information gain

$$\boxed{H(X) = -\sum_{i=1}^{N} P(x=i) \log_2 P(x=i)}$$

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

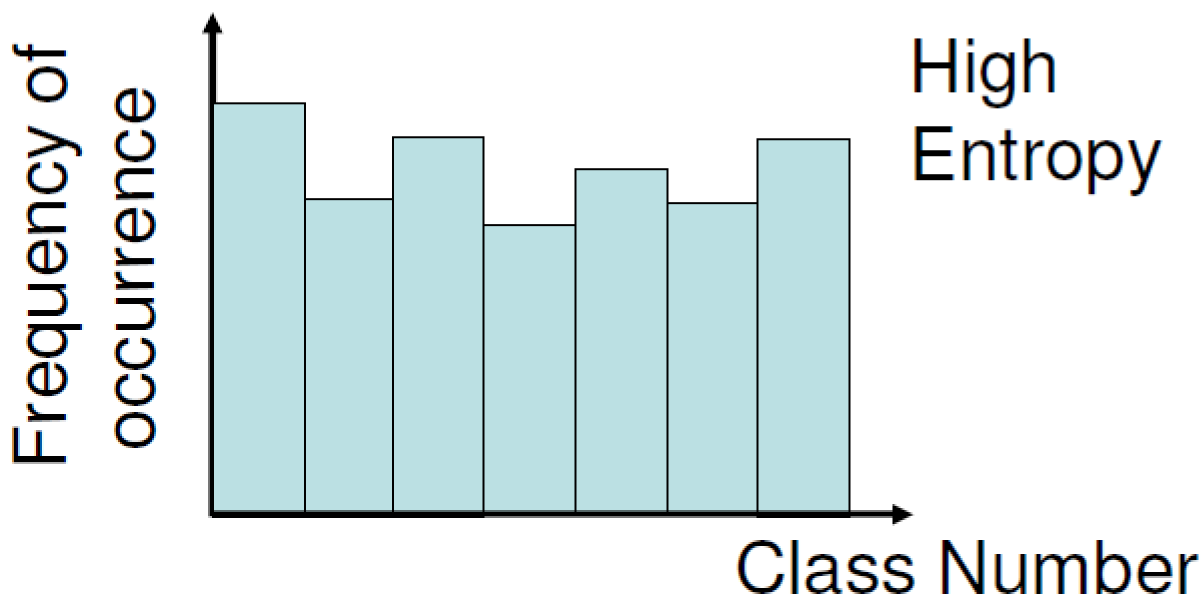
Entropy

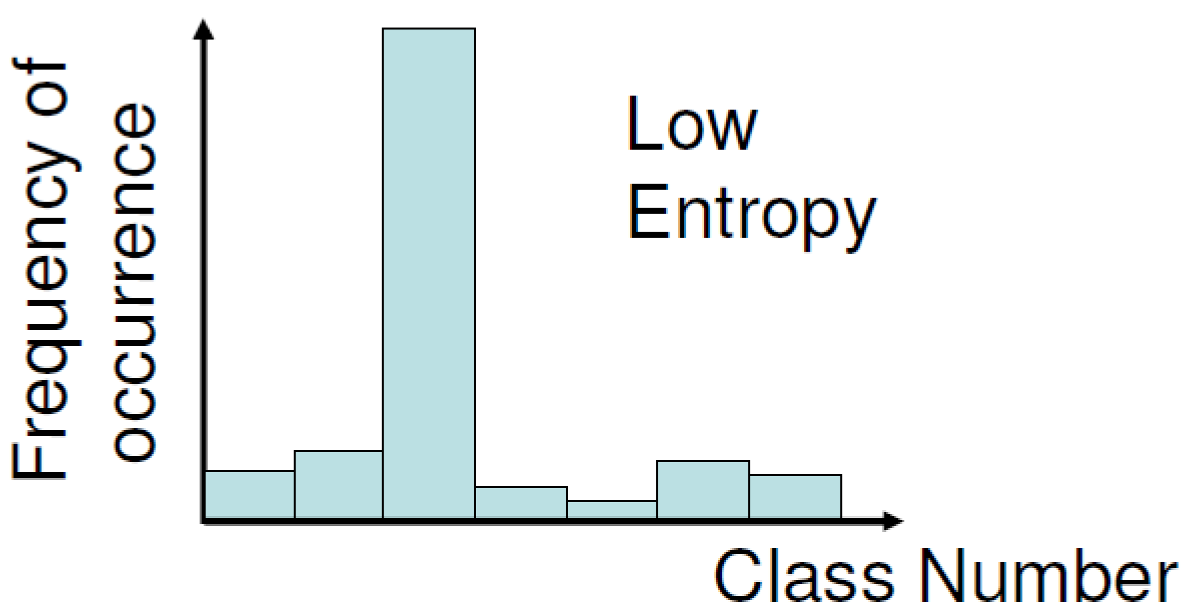
- In general, the average number of bits necessary to encode *n* values is the entropy:

$$\boxed{H = -\sum_{i=1}^{n} P_i \log_2 P_i}$$

- P_i = probability of occurrence of value i
  - High entropy → All the classes are (nearly) equally likely
  - Low entropy → A few classes are likely; most of the classes are rarely observed

Entropy

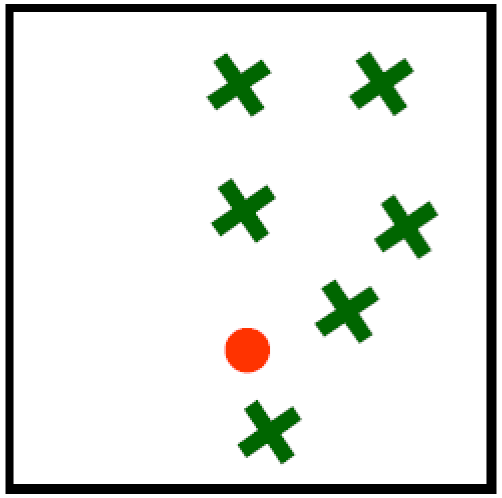




The entropy captures the degree of “purity” of the distribution

Example Entropy Calculation

1



$N_A = 1$

$N_B = 6$

$-\frac{1}{7}\log\frac{1}{7} - \frac{6}{7}\log\frac{6}{7} = H$

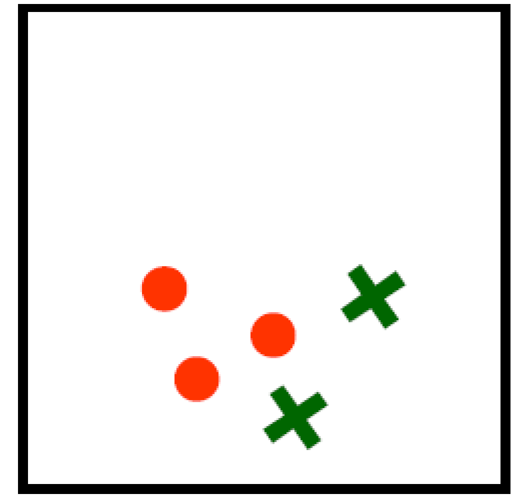
$p_A = N_A/(N_A+N_B) = 1/7$

$p_B = N_B/(N_A+N_B) = 6/7$

$H_1 = -p_A\log_2 p_A - p_B\log_2 p_B$

$= 0.59$

2



$N_A = 3$

$N_B = 2$

$p_A = N_A/(N_A+N_B) = 3/5$

$p_B = N_B/(N_A+N_B) = 2/5$

$H_2 = -p_A\log_2 p_A - p_B\log_2 p_B$

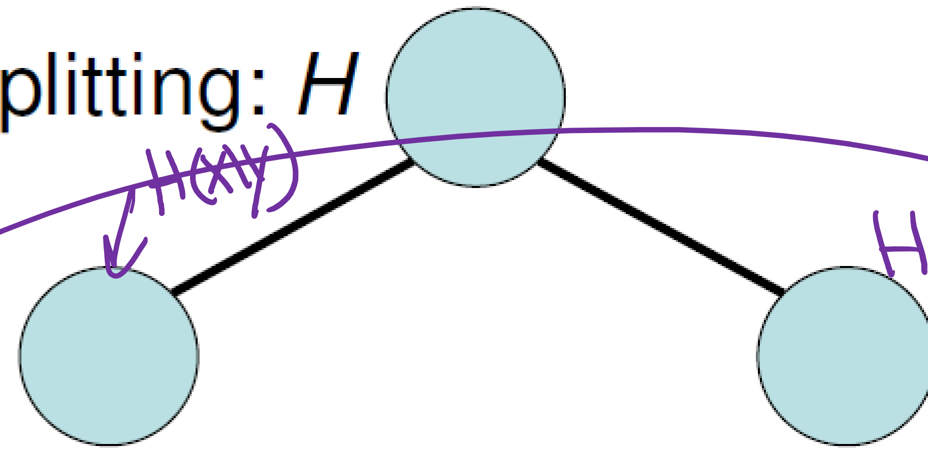
$= 0.97$

$H_1 < H_2 \Rightarrow$ (2) less pure than (1)

$= \sum p(x)\, H(X_R | X_d)$

Conditional Entropy

Entropy before splitting: *H*



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 entropy after splitting is:

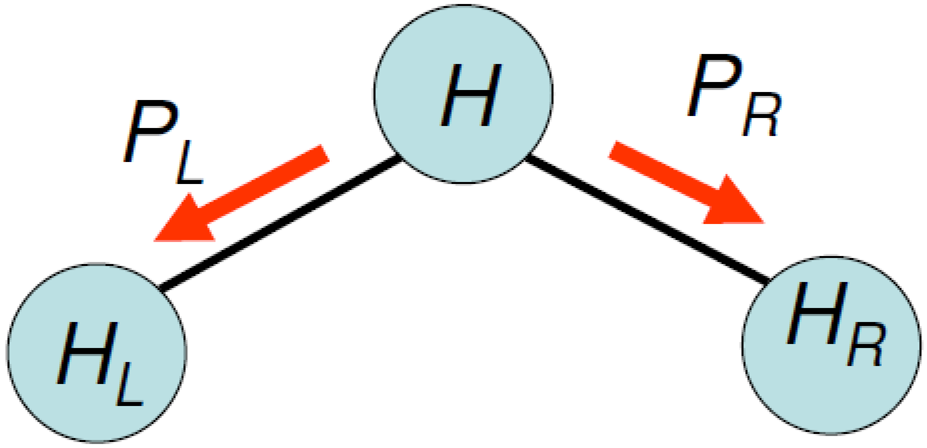
$$\boxed{H_L \times P_L + H_R \times P_R}$$

Entropy of left node

"Conditional Entropy"

Probability that a random input is directed to the 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

Information Gain (IG) = Amount by which the ambiguity is decreased by splitting the node

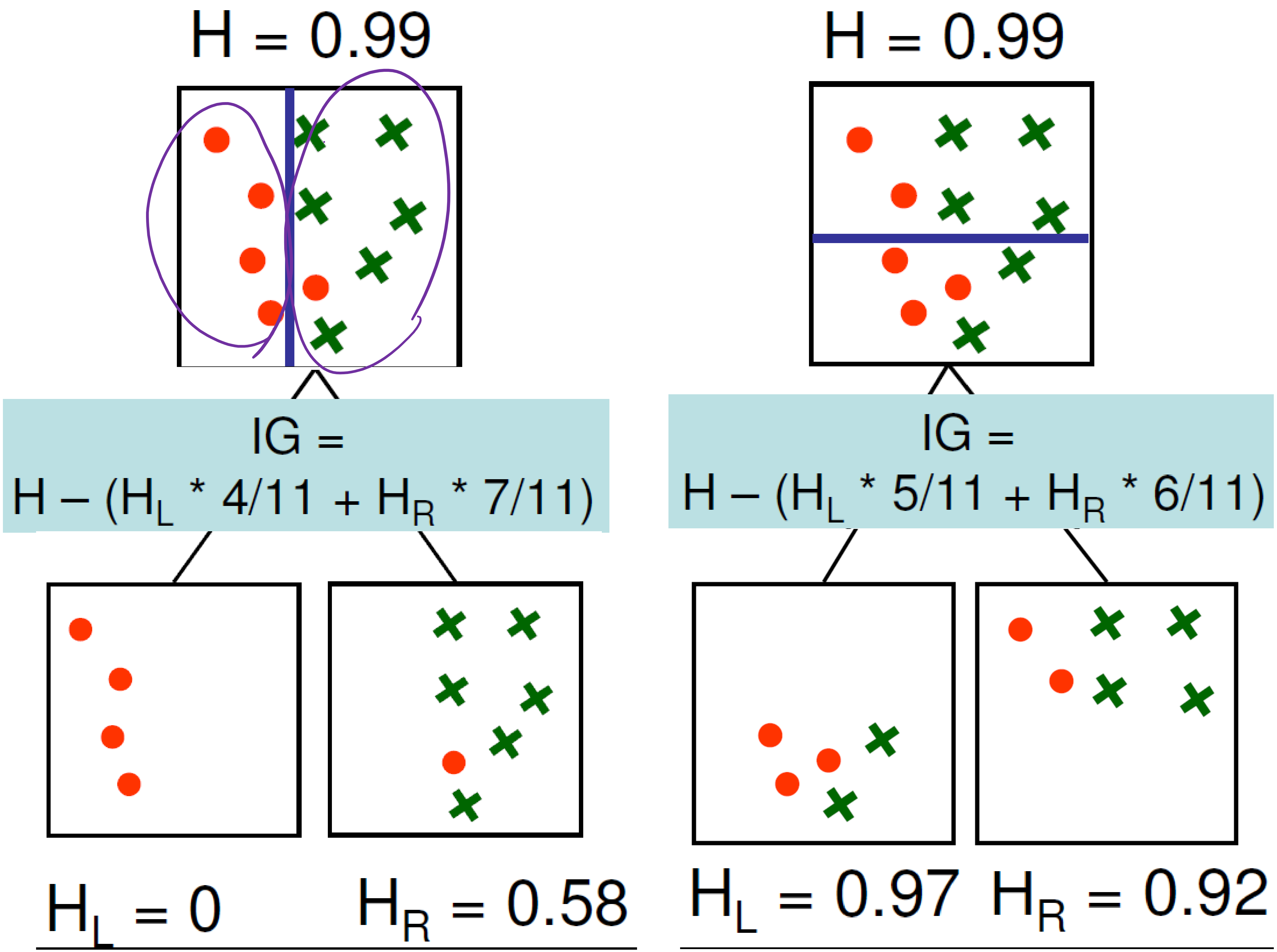
Maximize:

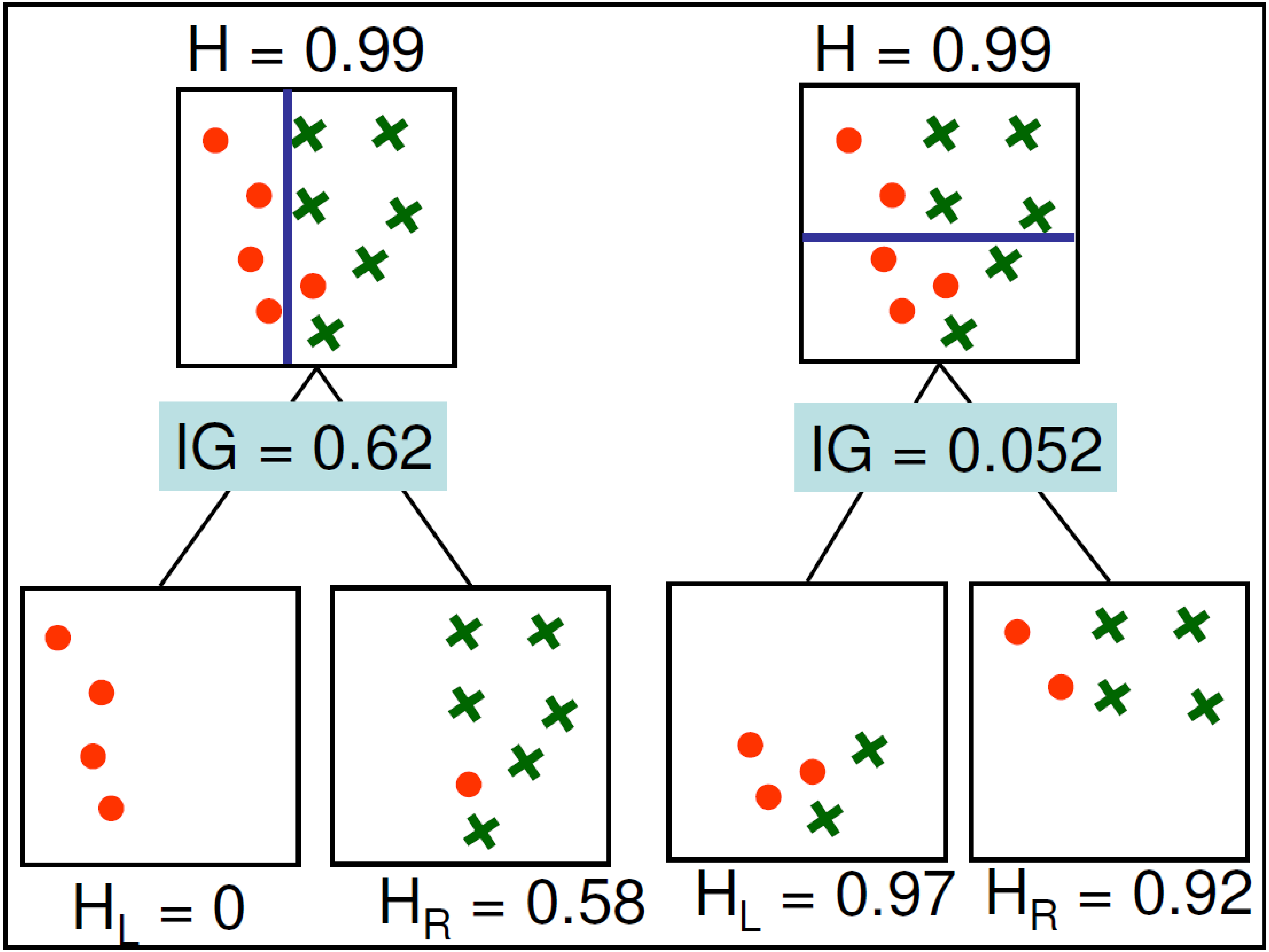
$$\boxed{IG = H - (H_L \times P_L + H_R \times P_R)}$$

Notations

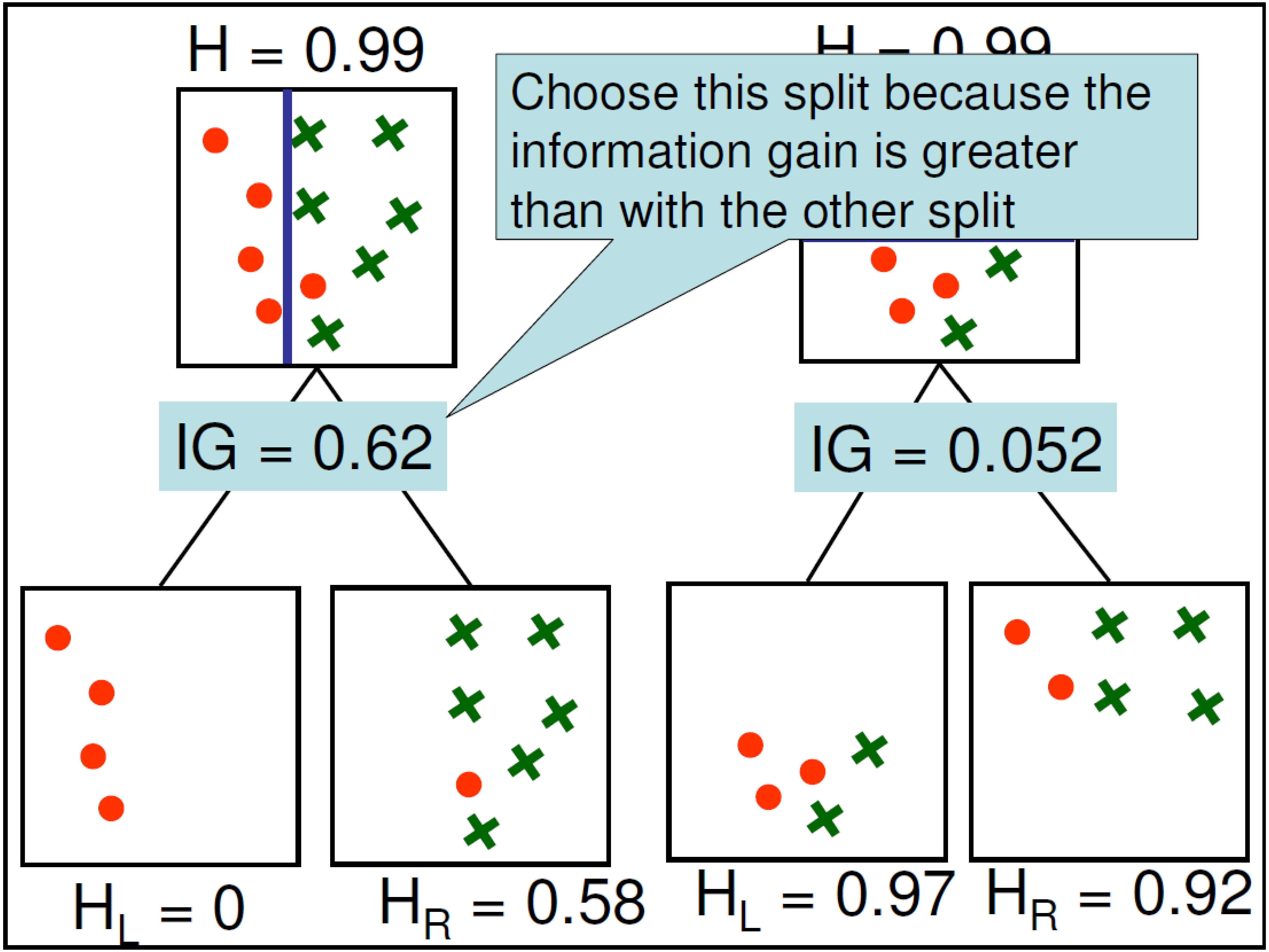
- Entropy: $H(Y)$ = Entropy of the distribution of classes at a node
- Conditional Entropy:
  - *Discrete*: $H(Y|X_j)$ = Entropy after splitting with respect to variable j
  - *Continuous*: $H(Y|X_j,t)$ = Entropy after splitting with respect to variable j with threshold t
- Information gain:
  - *Discrete*: $IG(Y|X_j) = H(Y) - H(Y|X_j)$ = Entropy after splitting with respect to variable j
  - *Continuous*: $IG(Y|X_j,t) = H(Y) - H(Y|X_j,t)$ = Entropy after splitting with respect to variable j with threshold t

H_{parent} $\quad P_{red} = \frac{5}{11}$ $\quad P_{green} = \frac{6}{11}$ $\quad H_p = -\frac{5}{11}\log_2\frac{5}{11} - \frac{6}{11}\log_2\frac{6}{11}$



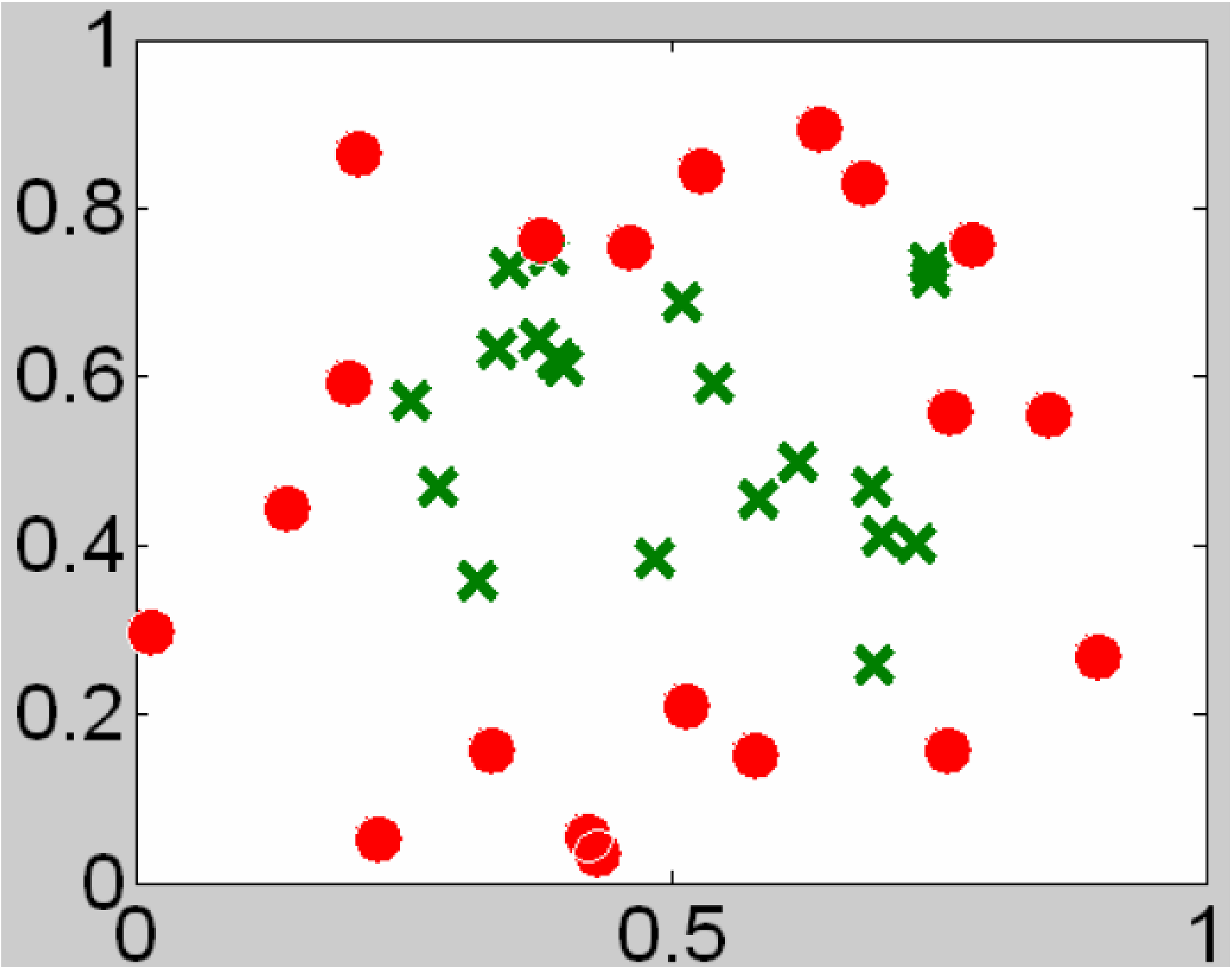


H = 0.99
H = 0.99
IG = 0.62
IG = 0.052
H_L = 0
H_R = 0.58
H_L = 0.97
H_R = 0.92



H = 0.99
H = 0.99
Choose this split because the information gain is greater than with the other split
IG = 0.62
IG = 0.052
H_L = 0
H_R = 0.58
H_L = 0.97
H_R = 0.92

More Complete Example



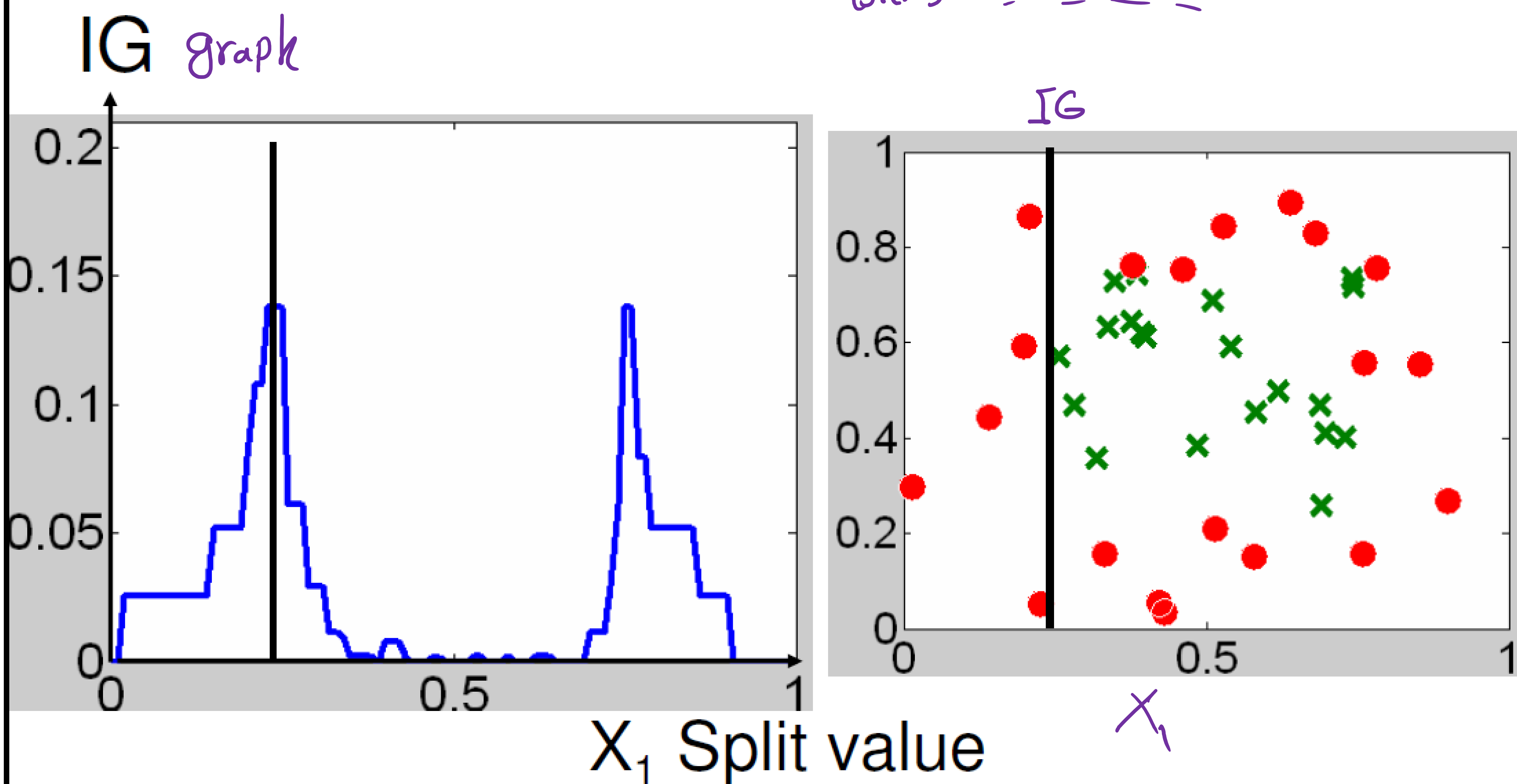
= 20 training examples from class A

= 20 training examples from class B

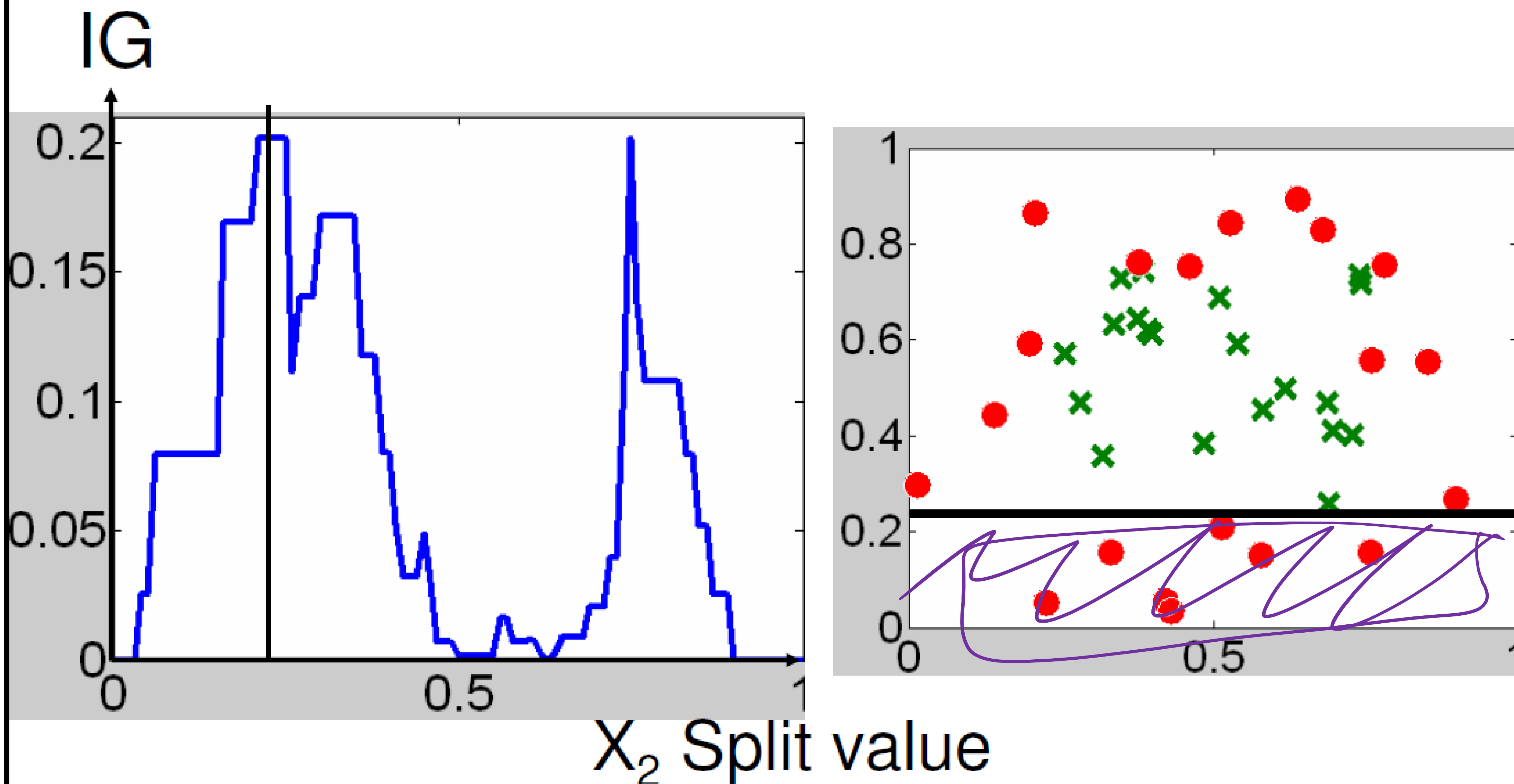
Attributes = X_1 and X_2 coordinates

0 | 0.1 | 0.2 ... 0.9 1

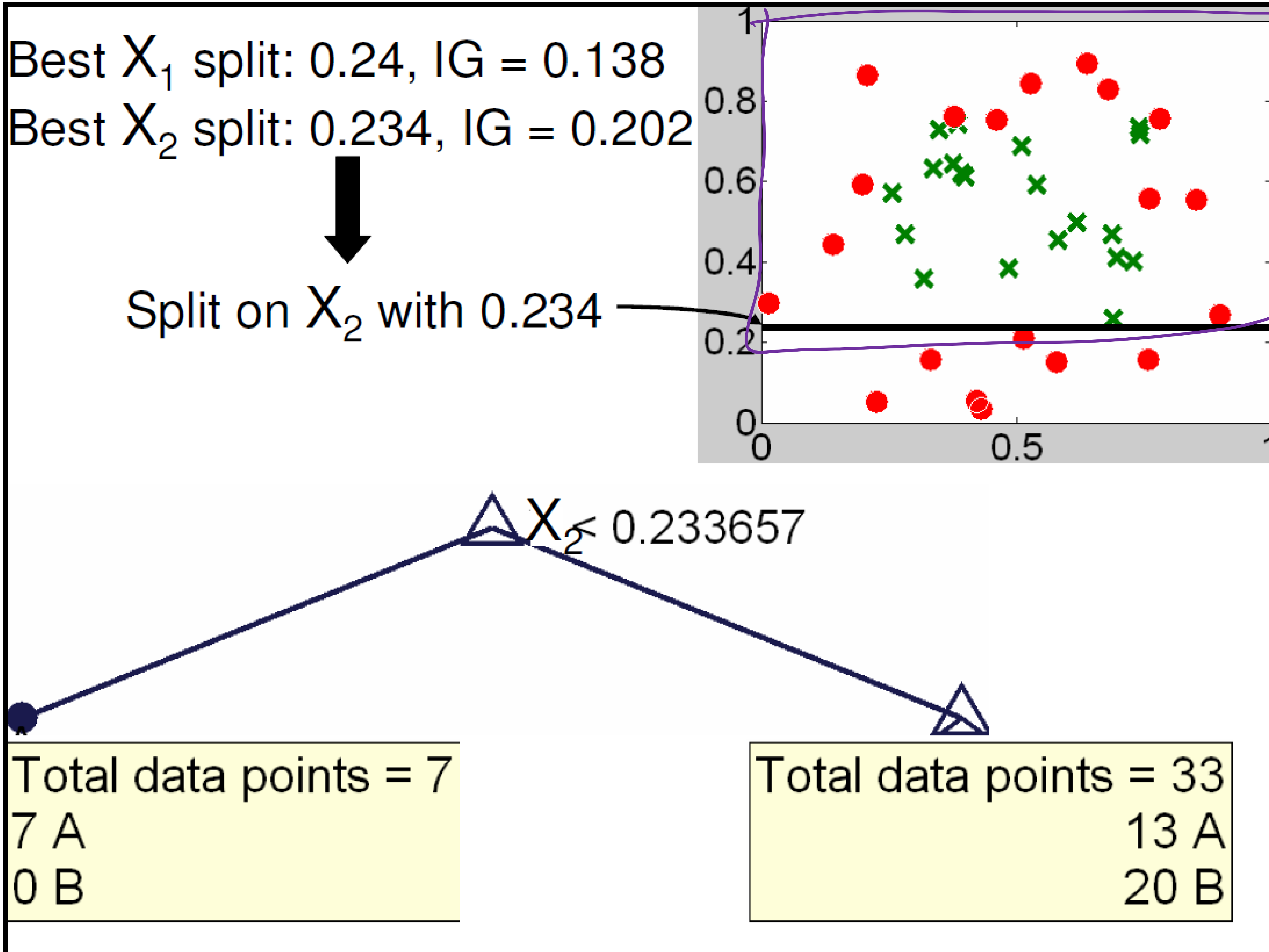
0.05 ----



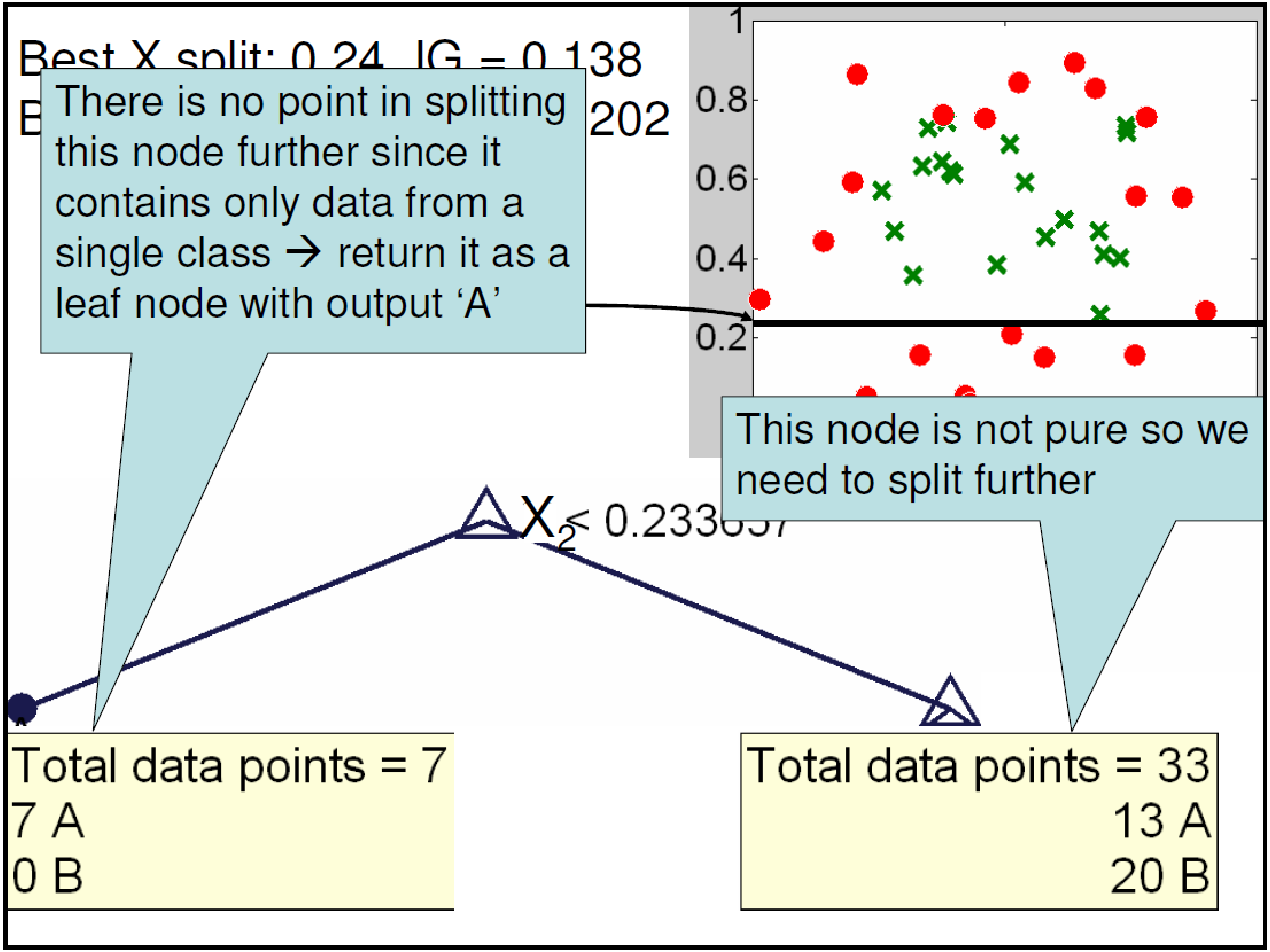
Best split value (max Information Gain) for X_1 attribute: 0.24 with IG = 0.138

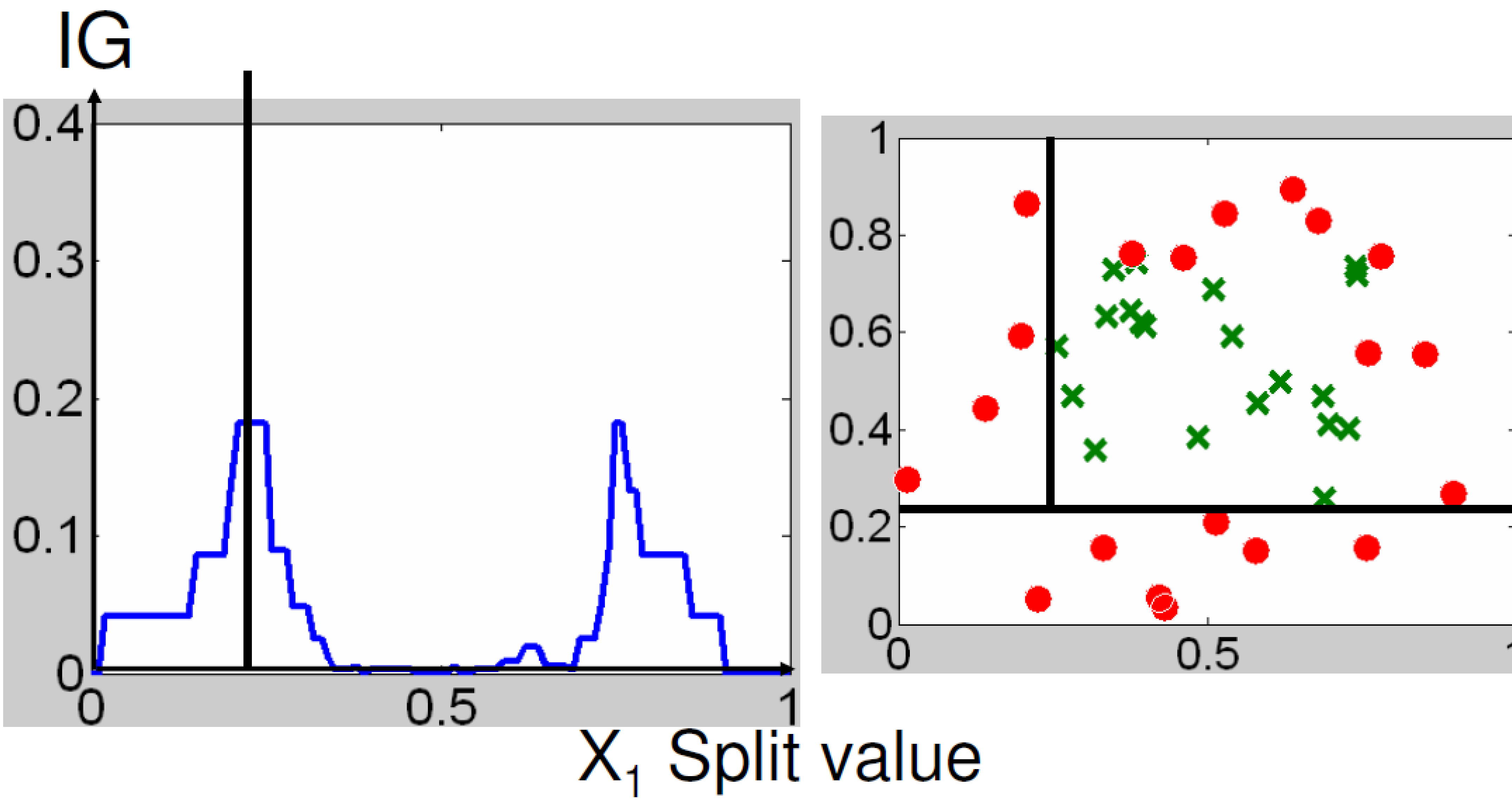


Best split value (max Information Gain) for X_2 attribute: 0.234 with IG = 0.202

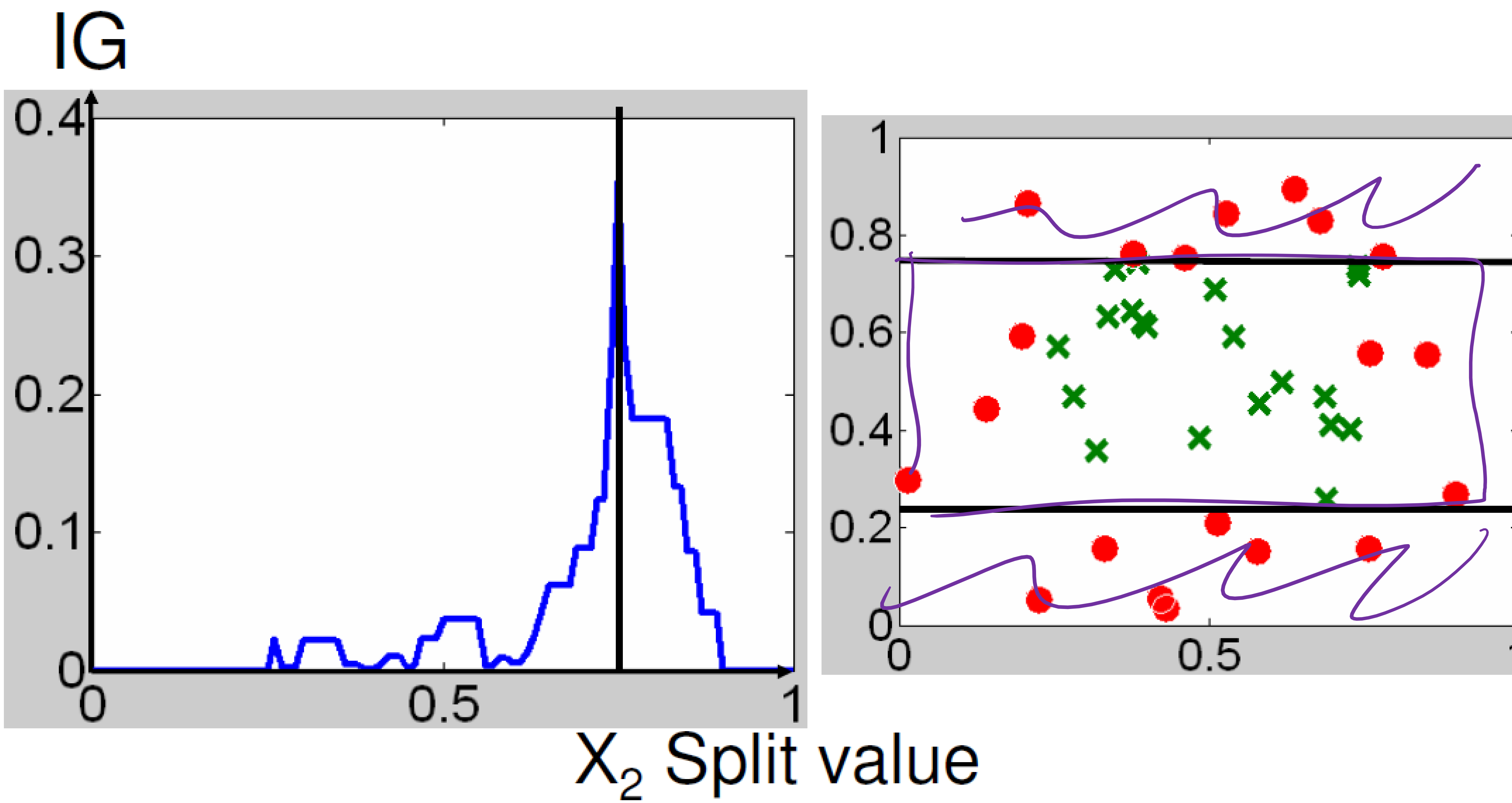


Left direction for **smaller** value, **right** direction for **bigger** value

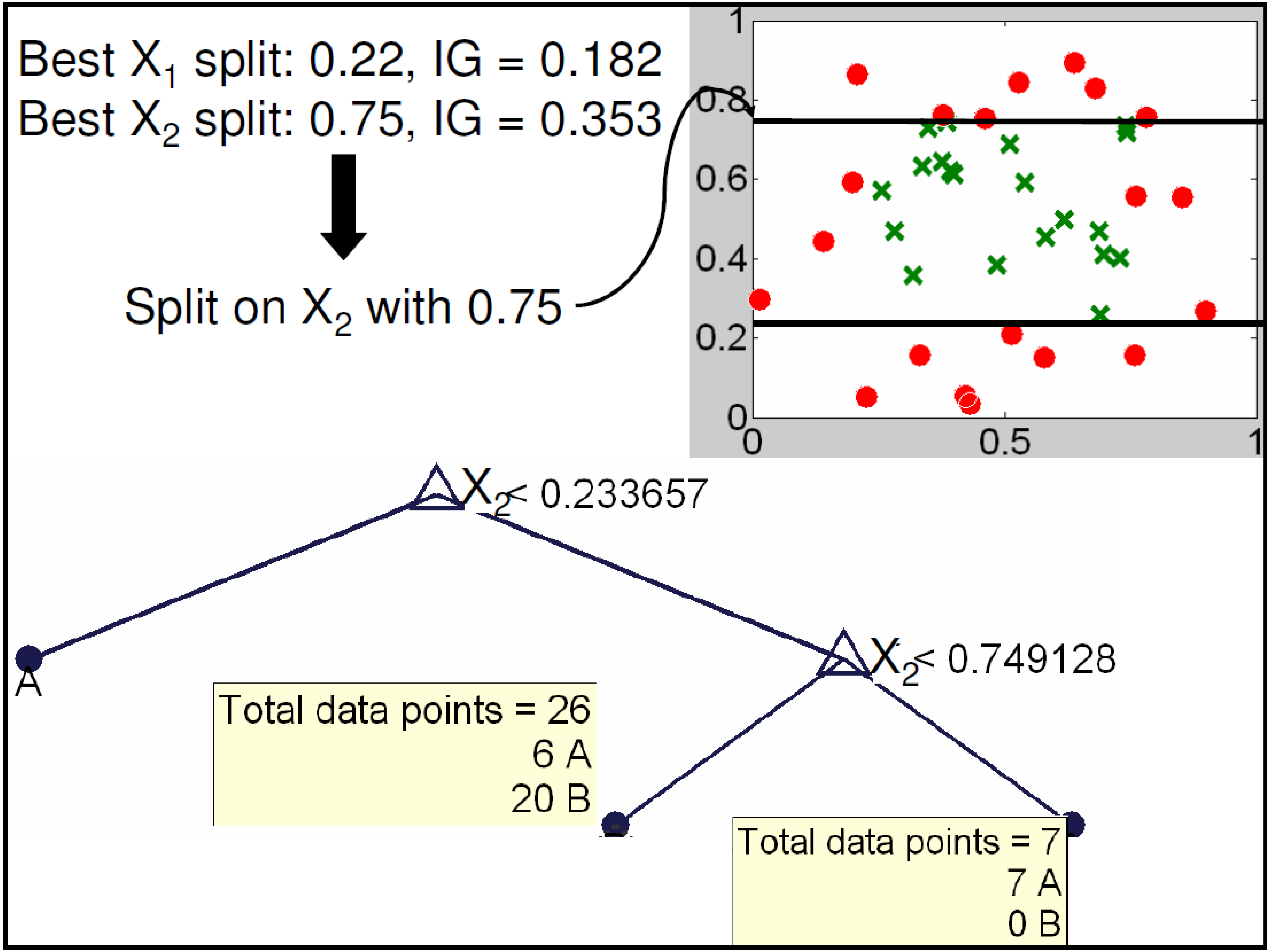

Best X split: 0.24, IG = 0.138
202
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output 'A'
This node is not pure so we need to split further
1
0.8
0.6
0.4
0.2
X2 < 0.233
Total data points = 7
7 A
0 B
Total data points = 33
13 A
20 B



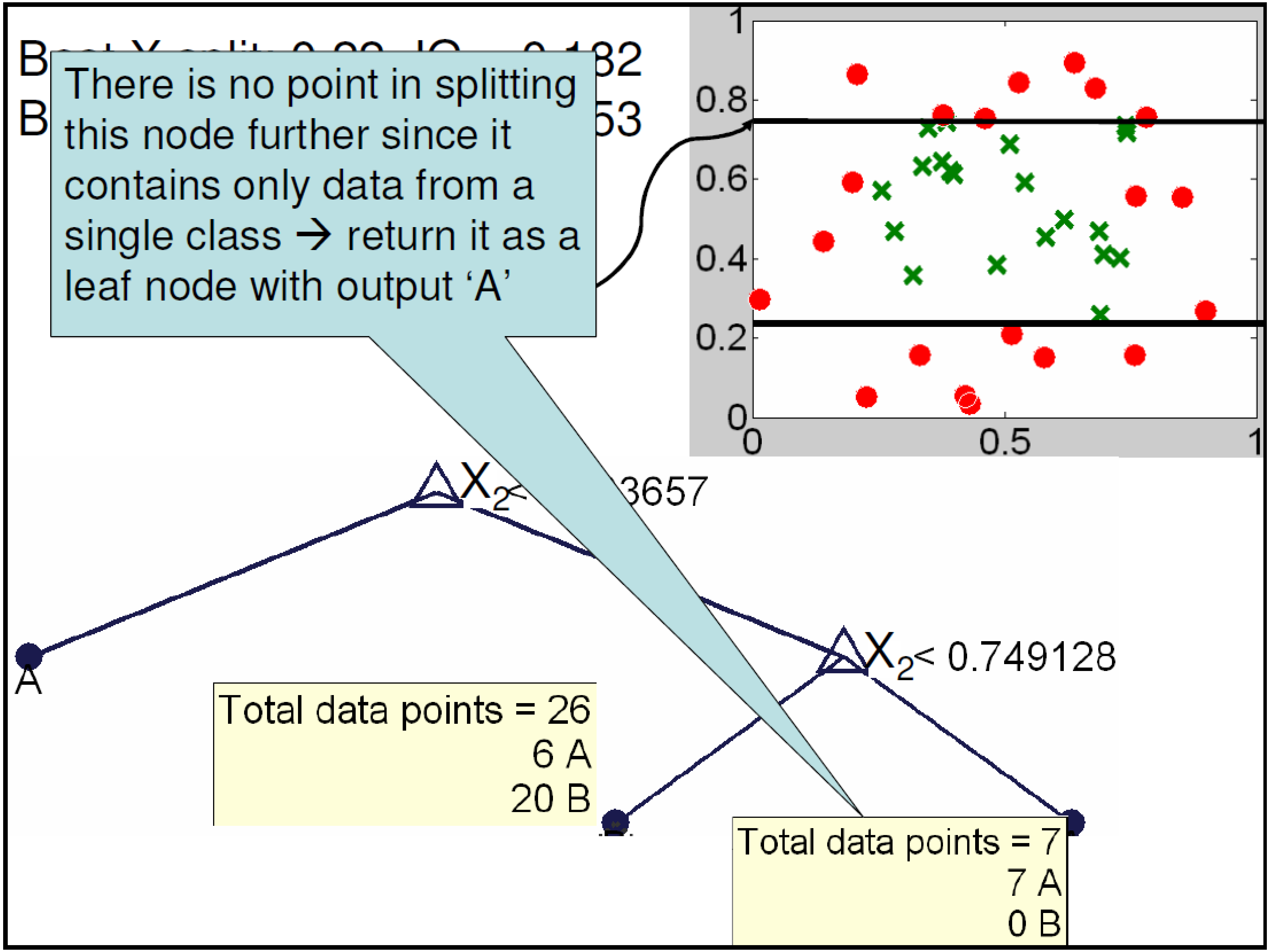
Best split value (max Information Gain) for X_1 attribute: 0.22 with IG ~ 0.182



Best split value (max Information Gain) for X_2 attribute: 0.75 with IG ~ 0.353

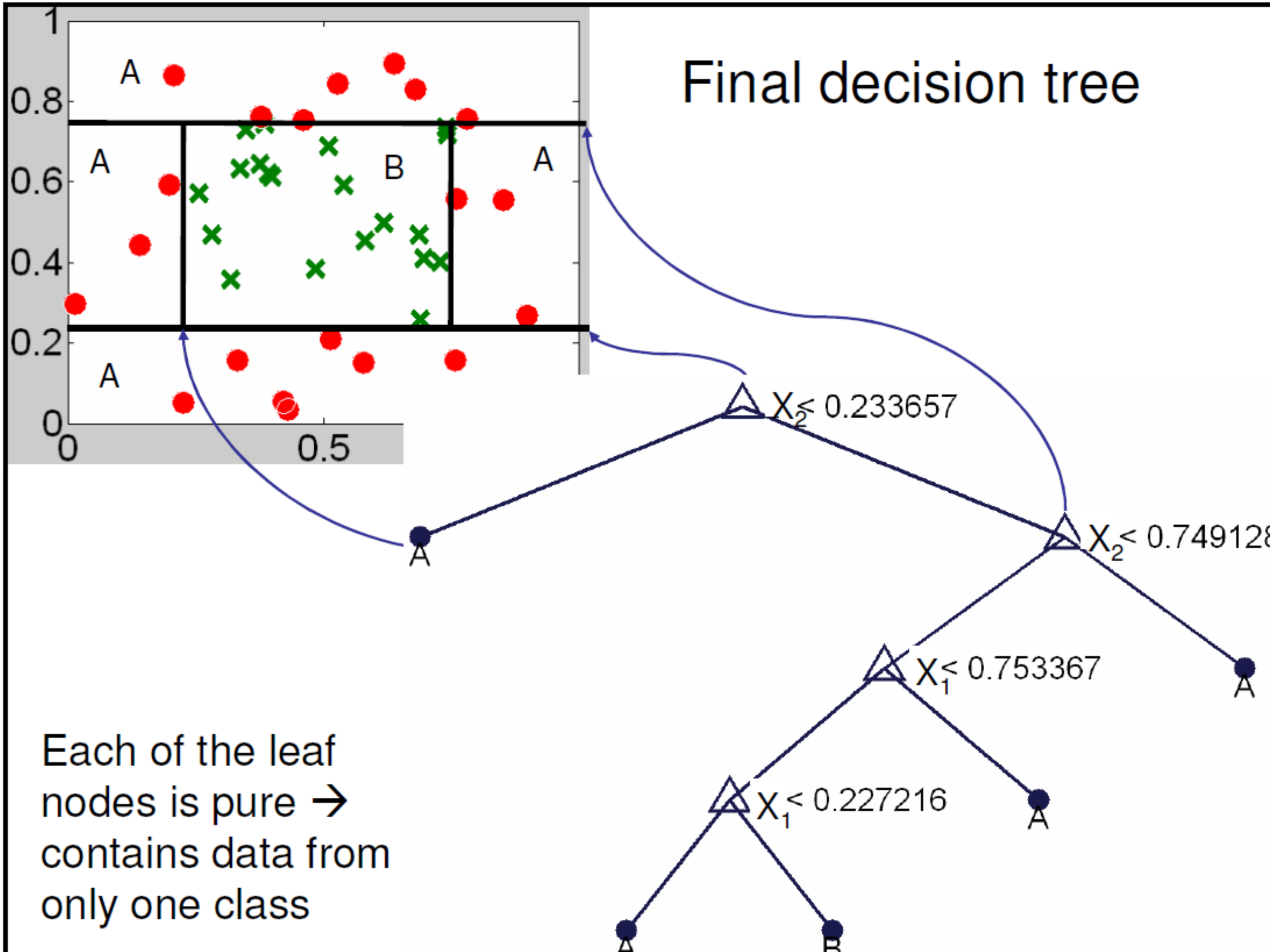


Best X_1 split: 0.22, IG = 0.182
Best X_2 split: 0.75, IG = 0.353
Split on X_2 with 0.75
X_2< 0.233657
X_2< 0.749128
A
Total data points = 26
6 A
20 B
Total data points = 7
7 A
0 B

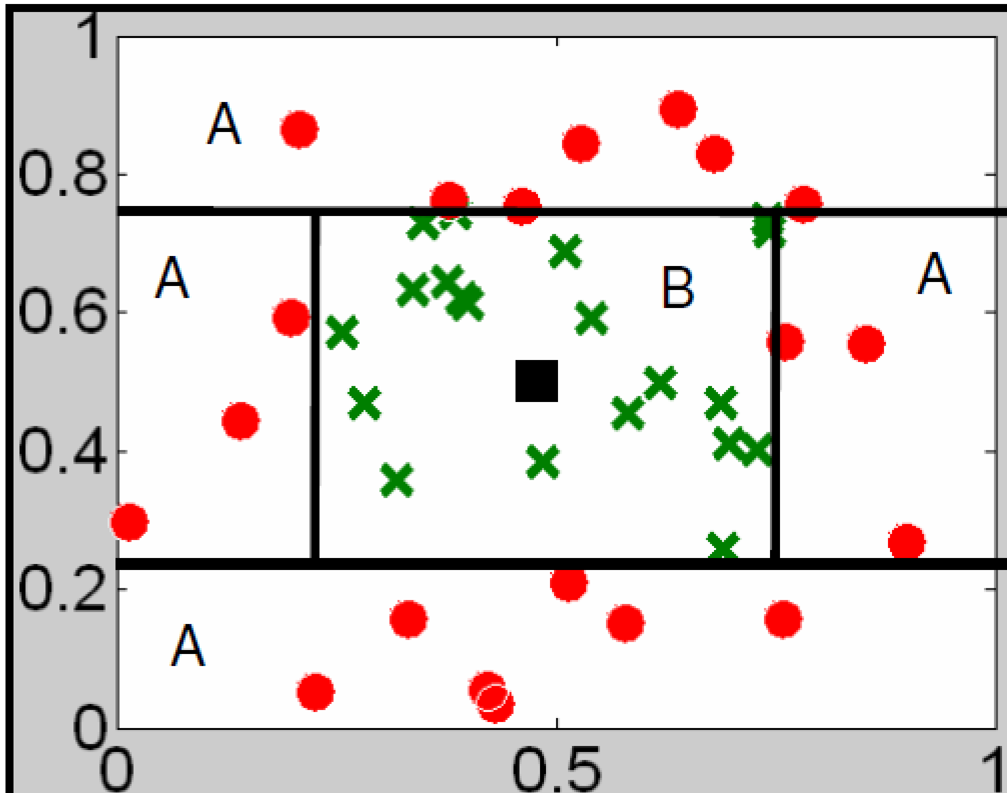


There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output 'A'
X2< 0.749128
A
Total data points = 26
6 A
20 B
Total data points = 7
7 A
0 B

Final decision tree



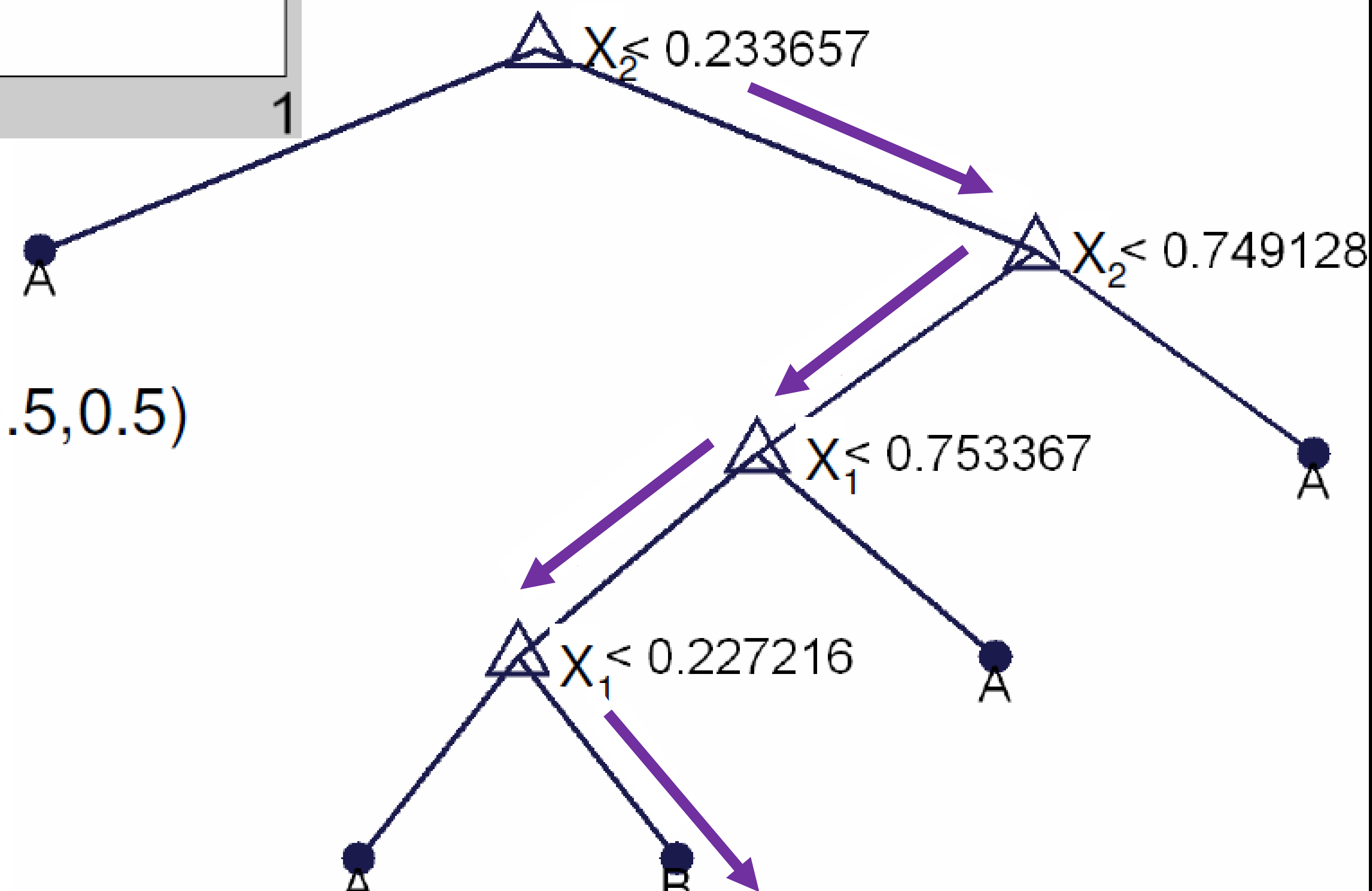
Each of the leaf nodes is pure → contains data from only one class



Final decision tree

Given an input (X,Y) → Follow the tree down to a leaf.

Return corresponding output class for this leaf



Example (X,Y) = (0.5,0.5)

Basic Questions

- How to choose the attribute/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?

When to stop splitting? Common strategies:

1. Pure and impure leave nodes
   - All points belong to the same class; OR
   - All points from one class completely overlap with points from another class (i.e., same attributes)
     - Output majority class as this leaf's label

2. Node contains points fewer than some threshold

3. Node purity is higher than some threshold

4. Further splits provide no improvement in *training loss*
   ($loss(T) <= loss(T_L) + loss(T_R)$)

Graphics from: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/DTs.pdf

Decision Tree Algorithm (Continuous Attributes)

- LearnTree(X, Y)
  - Input:
    - Set X of R training vectors, each containing the values (x_1,..,x_M) of M attributes (X_1,..,X_M)
    - A vector Y of R elements, where y_j = class of the j[th] datapoint
  - If all the datapoints in X have the same class value y
    - Return a leaf node that predicts y as output
  - If all the datapoints in X have the same attribute value (x_1,..,x_M)
    - Return a leaf node that predicts the majority of the class values in Y as output
  - Try all the possible attributes X_j and threshold t and choose the one, j^*, for which IG($Y|X_j,t$) is maximum
  - X_L, Y_L = set of datapoints for which $x_{j^*} < t$ and corresponding classes
  - X_H, Y_H = set of datapoints for which $x_{j^*} >= t$ and corresponding classes
  - Left Child ← LearnTree(X_L, Y_L)
  - Right Child ← LearnTree(X_H, Y_H)

Decision Tree Algorithm (Discrete Attributes)

- LearnTree(X, Y)
  - Input:
    - Set X of R training vectors, each containing the values $(x_1,..,x_M)$ of M attributes $(X_1,..,X_M)$
    - A vector Y of R elements, where y_j = class of the jth datapoint
  - If all the datapoints in X have the same class value y
    - Return a leaf node that predicts y as output
  - If all the datapoints in X have the same attribute value $(x_1,..,x_M)$
    - Return a leaf node that predicts the majority of the class values in Y as output
  - Try all the possible attributes X_j and choose the one, j^*, for which IG($Y|X_j$) is maximum
  - For every possible value v of X_{j^*}:
    - X_v, Y_v= set of datapoints for which $x_{j^*} = v$ and corresponding classes
    - Child$_v$ $\leftarrow$ LearnTree(X_v, Y_v)

Decision Trees So Far

- Given N observations from training data, each with D attributes X and a class attribute Y, construct a sequence of tests (decision tree) to predict the class attribute Y from the attributes X
- Basic strategy for defining the tests ("when to split") → maximize the information gain on the training data set at each node of the tree
- Problems (next):
  - Computational issues → How expensive is it to compute the IG
  - The tree will end up being much too big → *pruning*
  - Evaluating the tree on training data is dangerous → *overfitting*

Basic Questions

- How to choose the attribute/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?

What will happen if a tree is too large?

Overfitting

High variance

Instability in predicting test data

How to avoid overfitting?

- Acquire more training data
- Remove irrelevant attributes (manual process – not always possible)
- Grow full tree, then post-prune
- Ensemble learning

Reduced-Error Pruning

Split data into training and validation sets

Grow tree based on training set

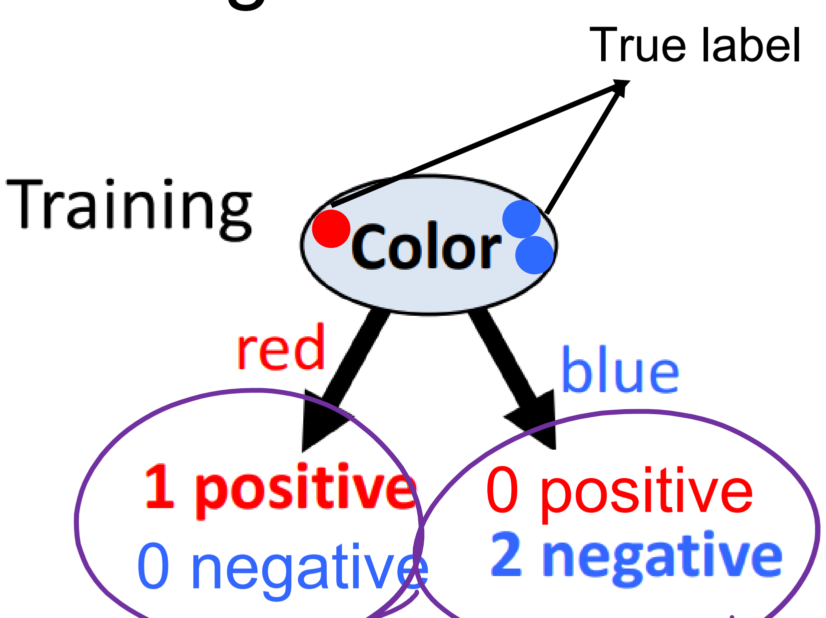
Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)

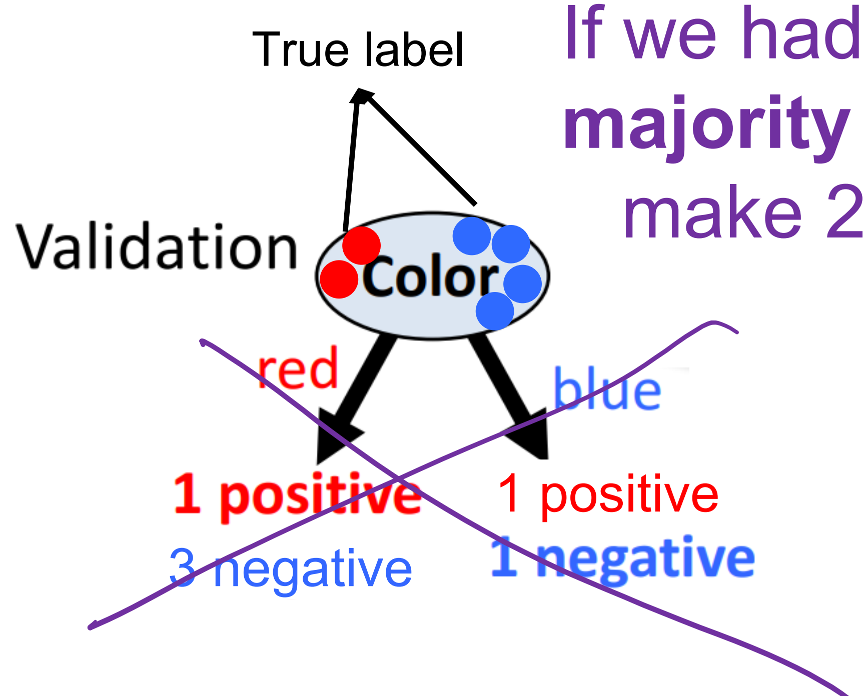
2. Greedily remove the node that most improves validation set accuracy

How to decide to remove it a node using pruning

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node.
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf.



3 training data points
Actual Label: 1 positive and 2 negative
Predicted Label: 1 positive and 2 negative
3 correct and **0** incorrect



6 validation data points
Actual label:2 positive and 4 negative
Predicted Label: 4 positive and 2 negative
2 correct and **4** incorrect

If we had simply predicted the **majority class** (negative), we make 2 errors instead of 4

Pruned!