Lecture 07. Decision Tree

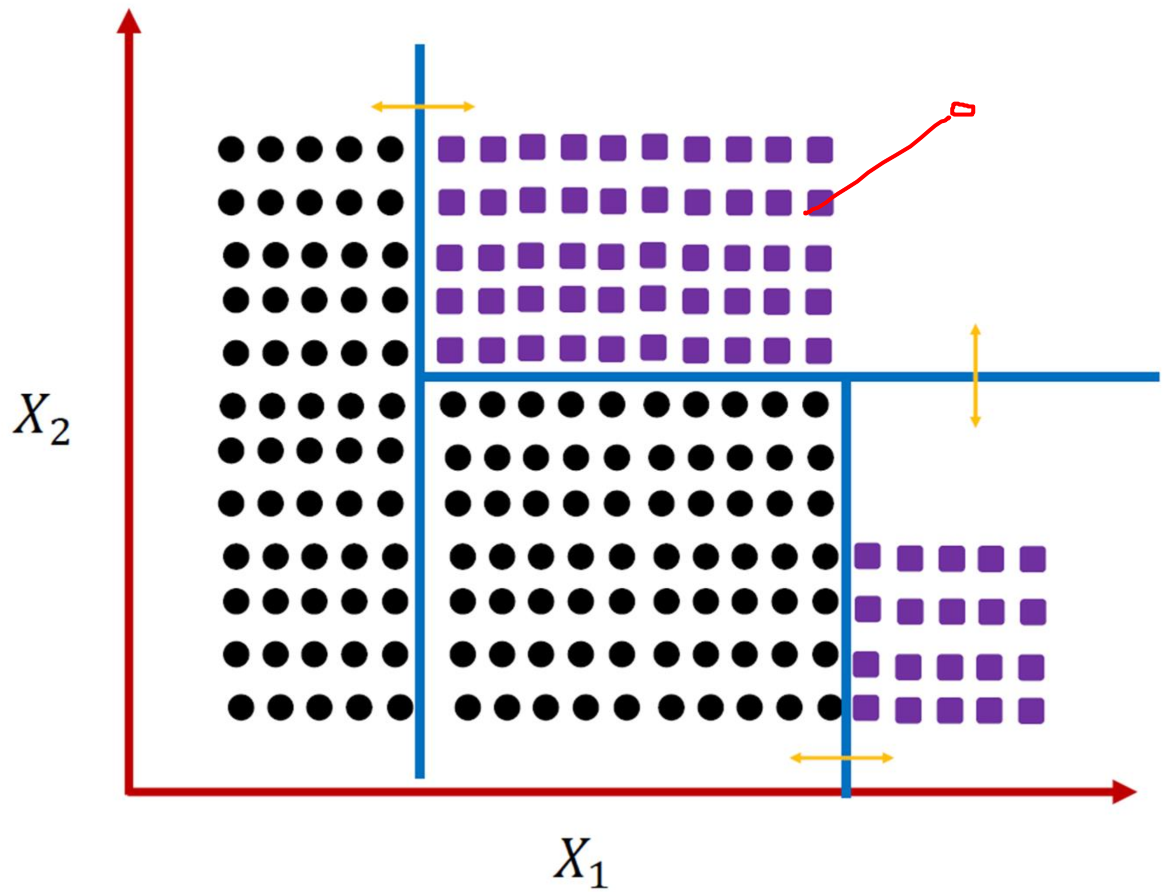
Xin Chen

These slides are based on slides from Mahdi Roozbahani

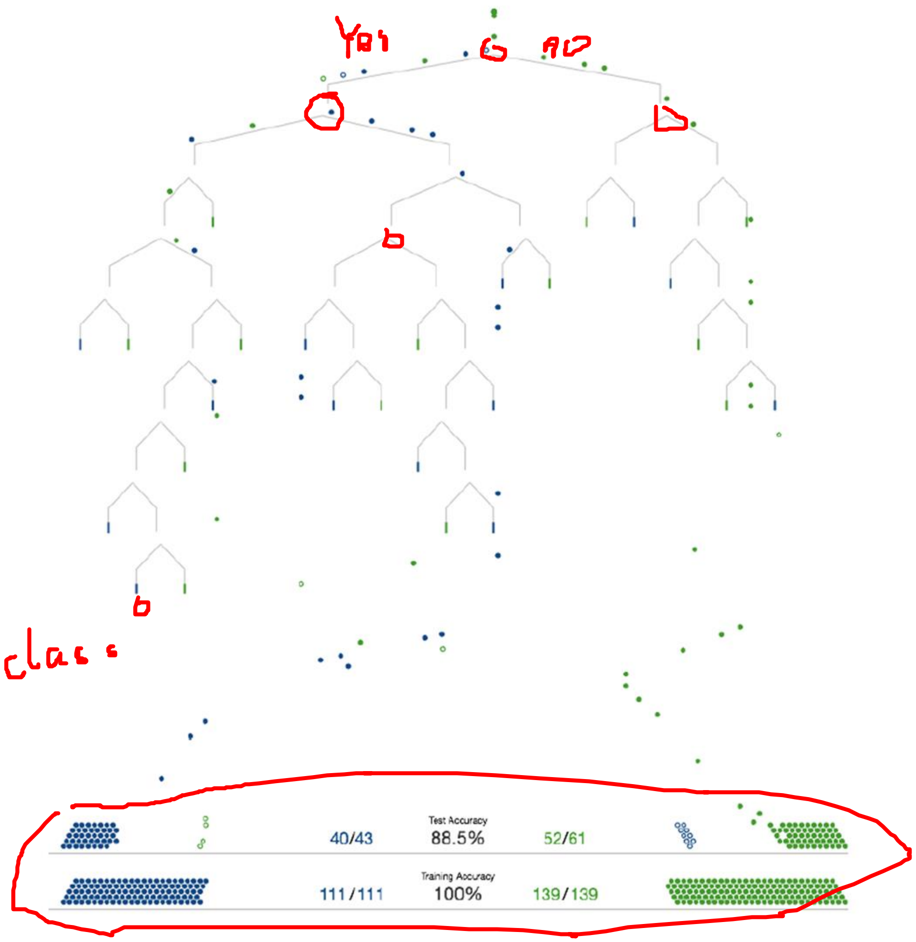
Logistics

- Random team assignment is done, in the “Pages” on Canvas.
- Project proposal is extended to Jun 17th
- Project proposal length: the limit is two pages, 1 page is acceptable. Font size is 12.
- Individual contribution: by default, each team member on the report share the same grade. I will ask for more information from all team members if someone think he deserves more or less.
- Project topic: anything is fine as long as it is related ML algorithms.

Decision Tree Example



Visual introduction to decision tree



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: **S**unny, **O**vercast, **R**ainy

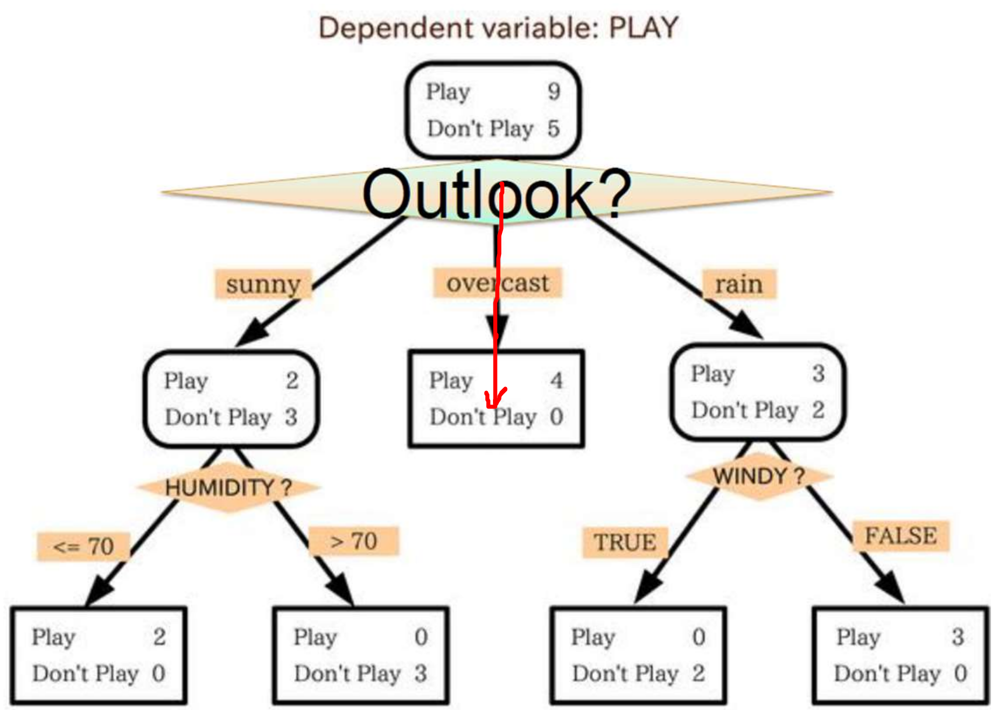
Temperature: **H**ot, **M**edium, **C**ool

Humidity: **H**igh, **N**ormal, **L**ow

Wind: **S**trong, **W**eak

Will I play tennis today?

Decision Tree Example

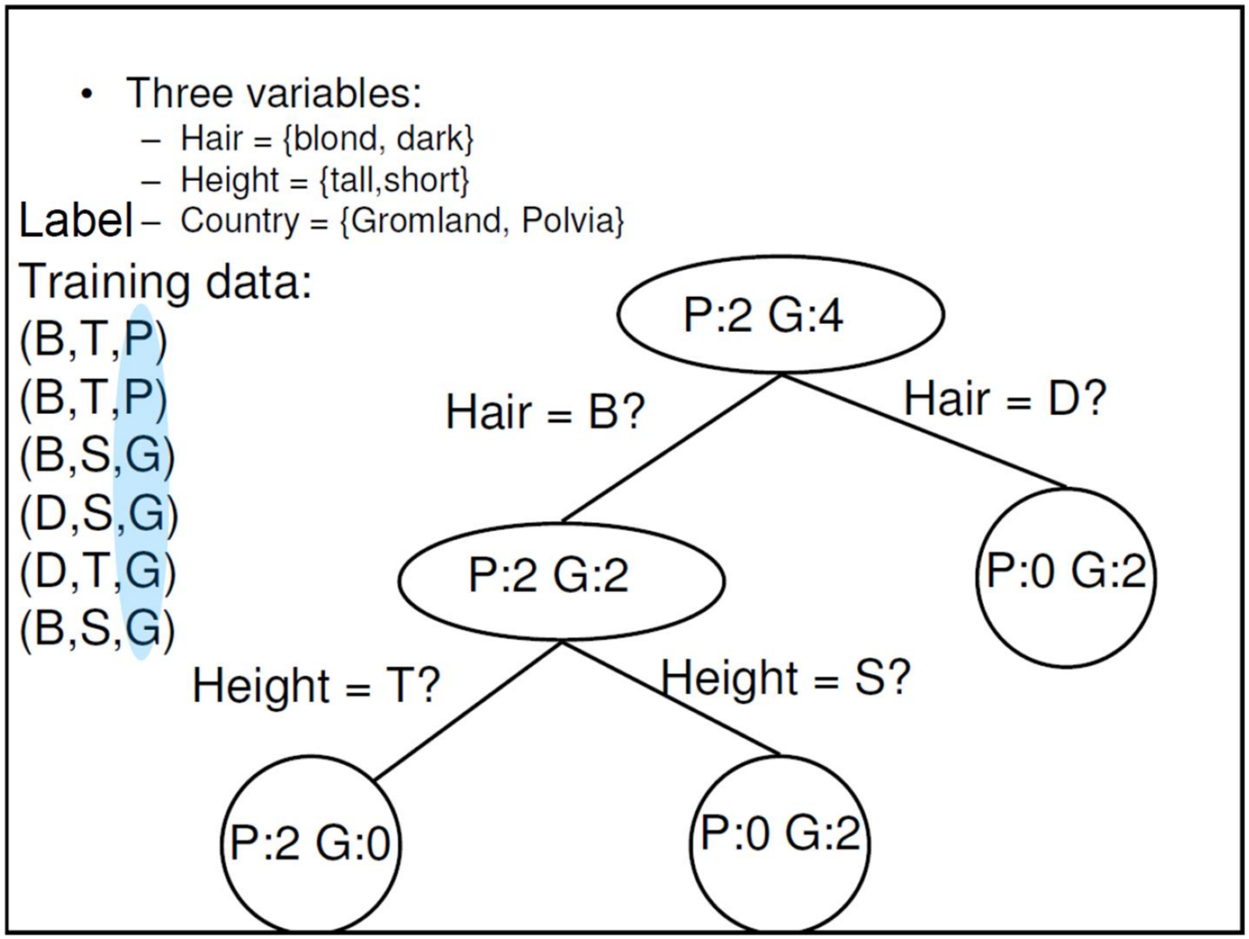


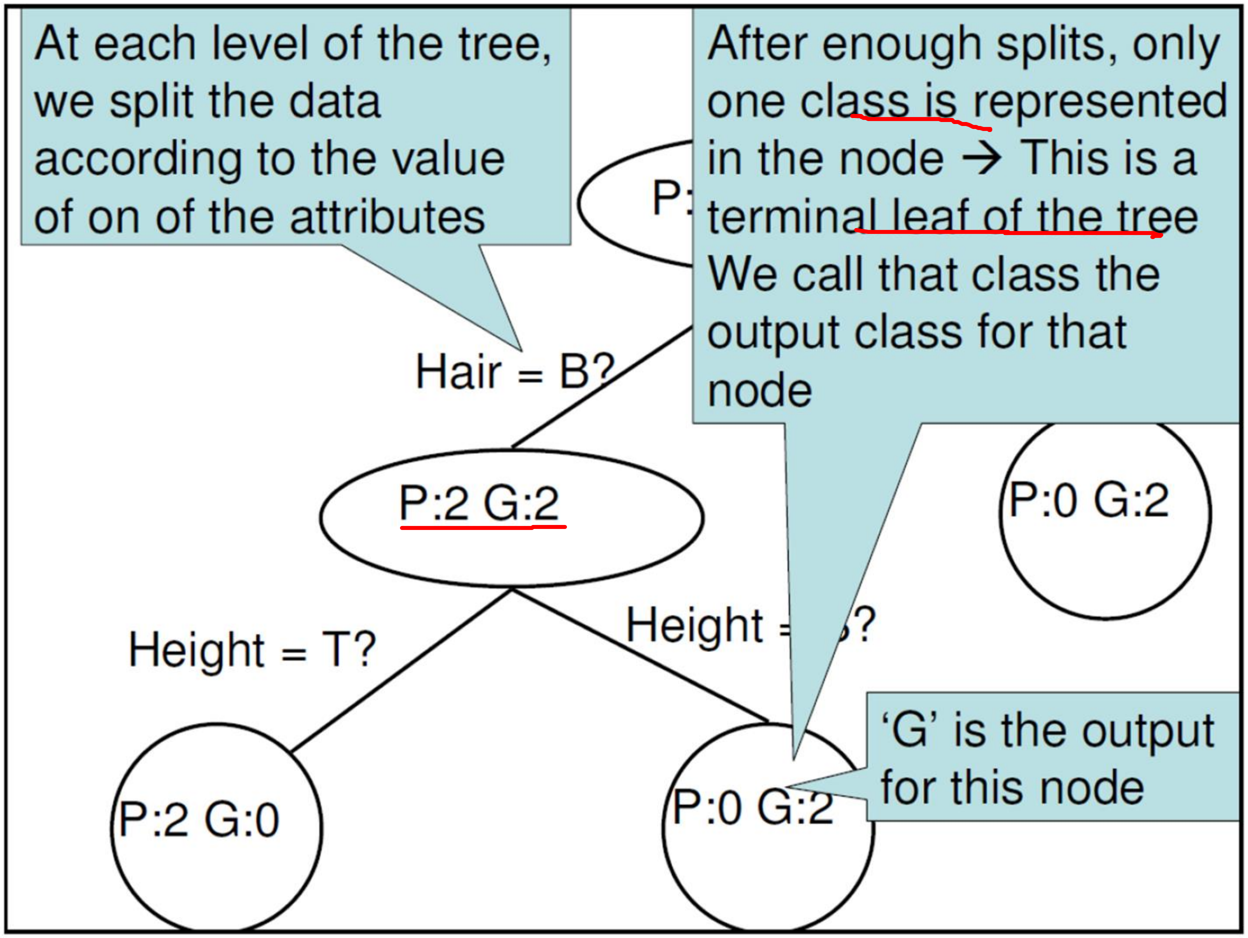
The classifier: F(x): majority class in the leaf in the tree T containing x
Model parameters: the tree structure and size

Decision trees

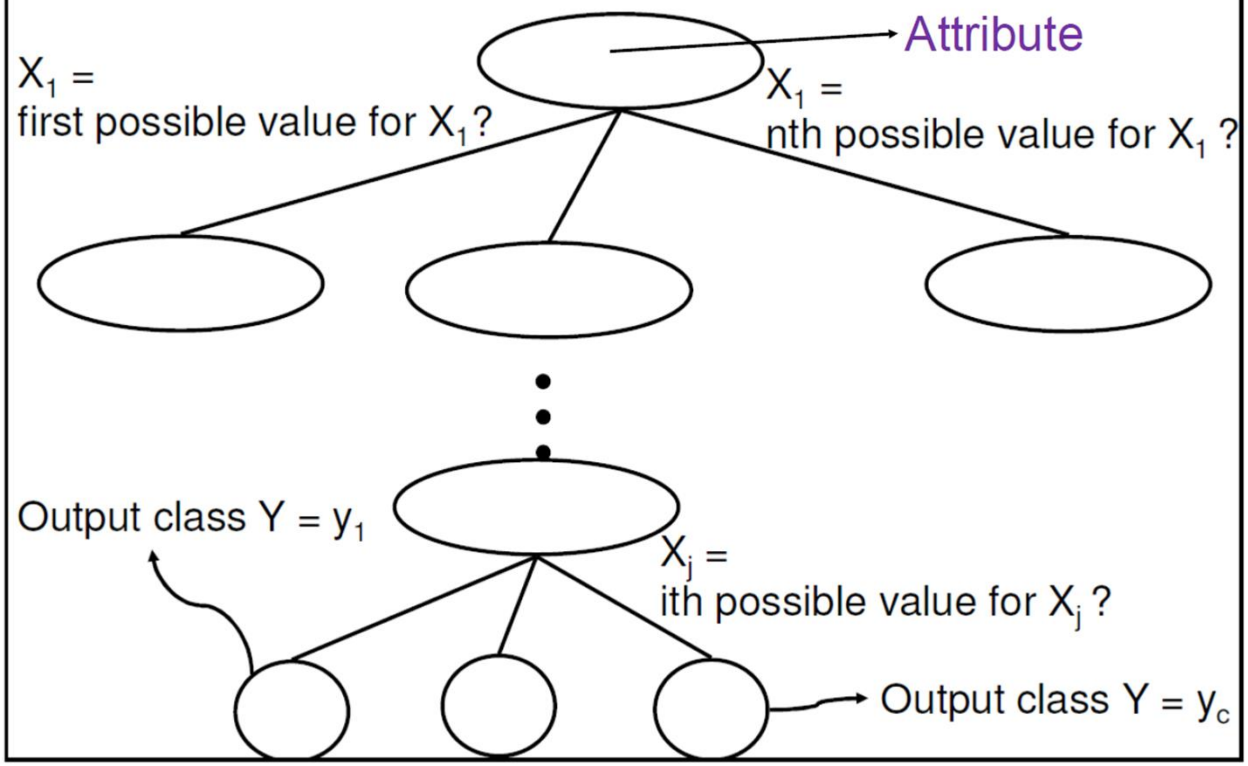
- Pieces:
  - Find the best attribute to split on
  - Find the best split on the chosen attribute
  - Decide on when to stop splitting

Categorical or discrete attributes

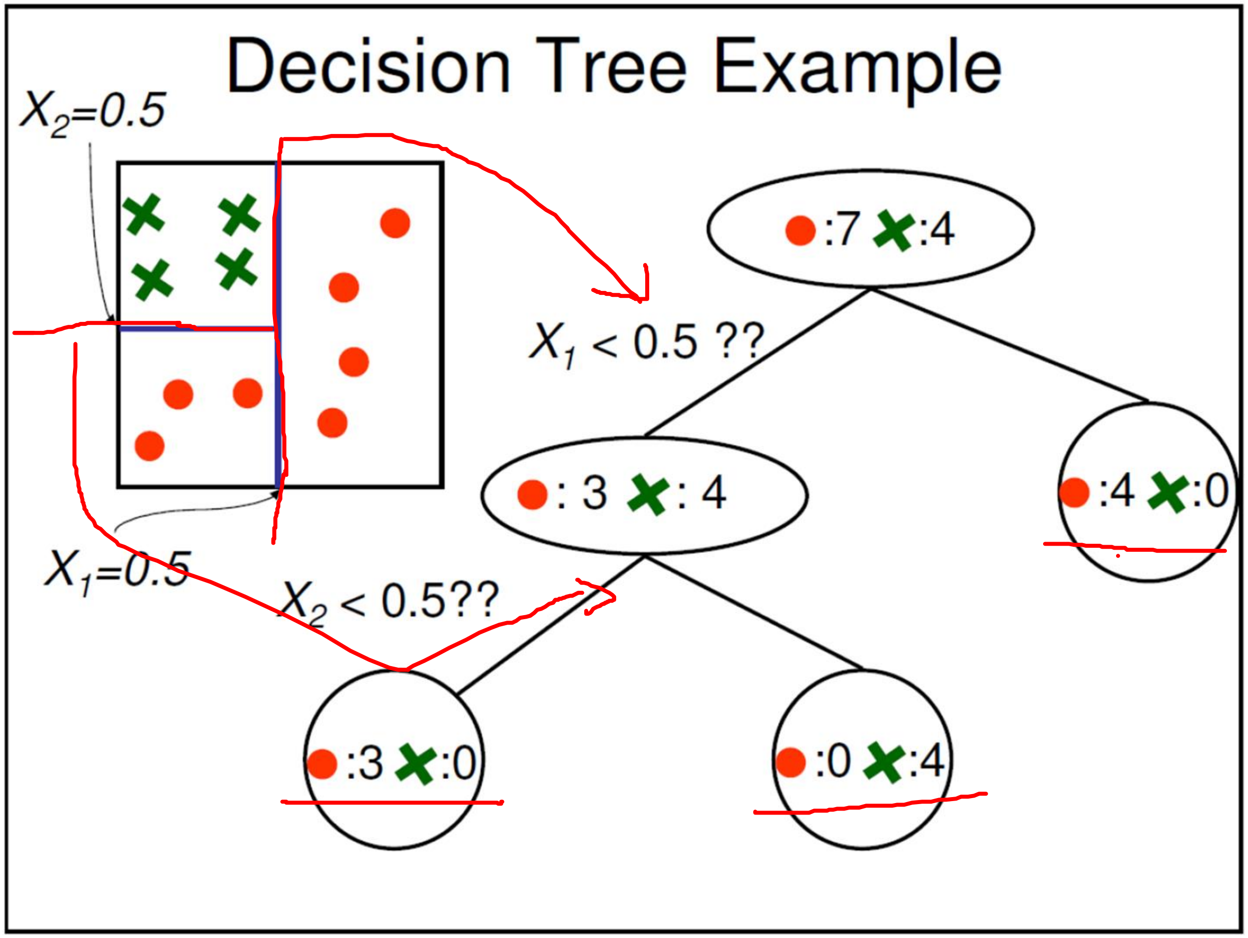


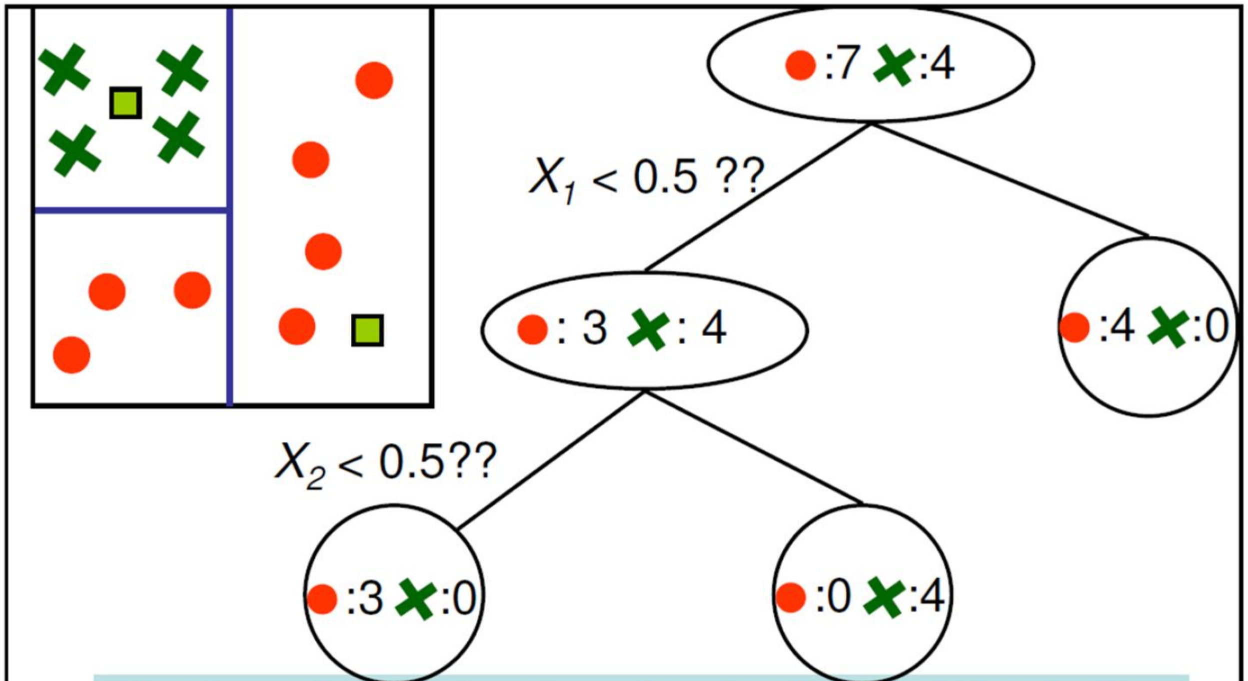

At each level of the tree, we split the data according to the value of on of the attributes
After enough splits, only one class is represented in the node → This is a terminal leaf of the tree We call that class the output class for that node
Hair = B?
P:2 G:2
P:0 G:2
Height = T?
Height =
P:2 G:0
P:0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

General Decision Tree (Discrete Attributes)



General decision tree (Continuous Attributes)



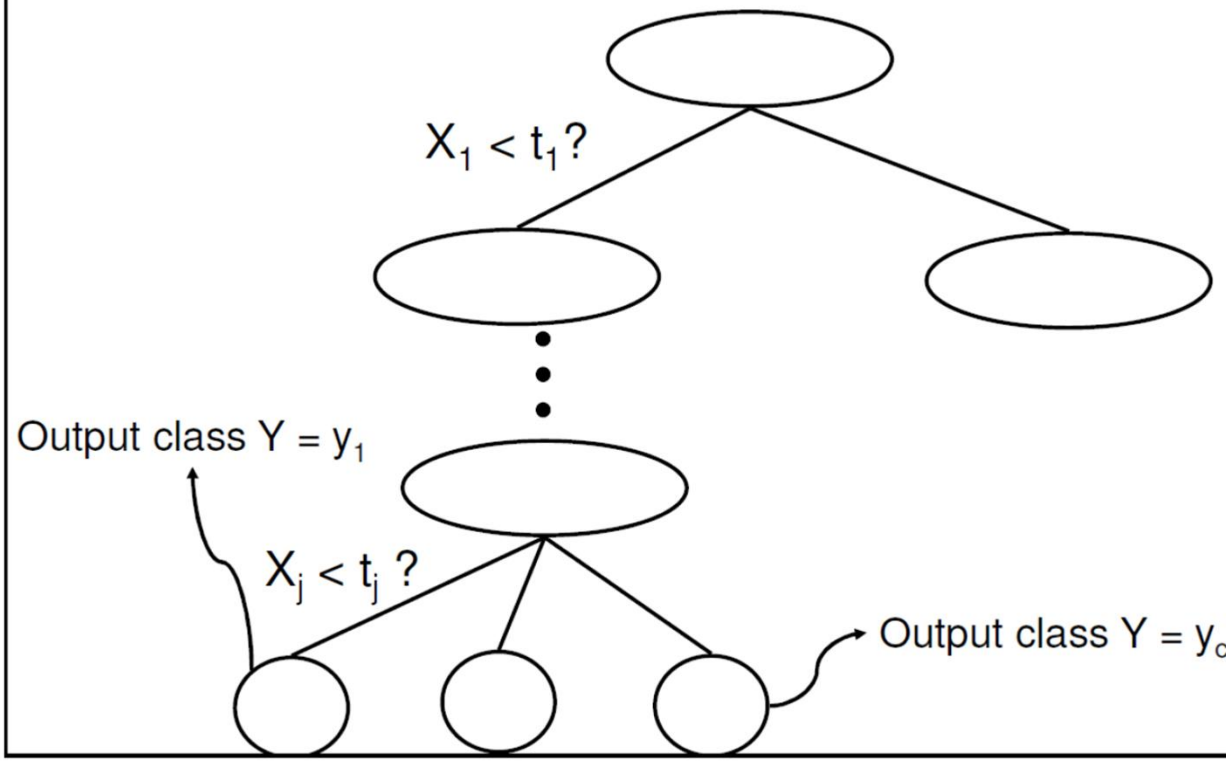


The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:

(0.2,0.8) is classified as ✖

(0.8,0.2) is classified as ●

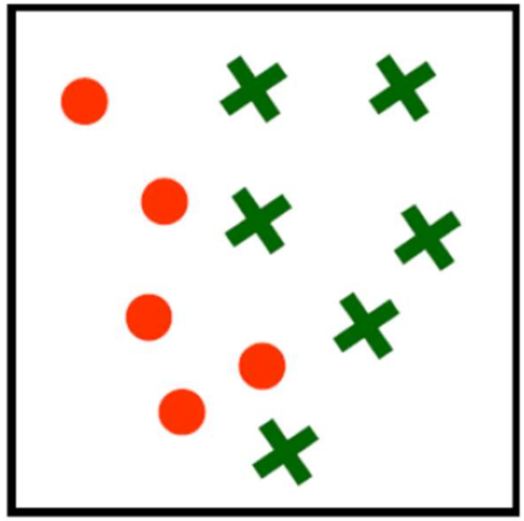
General Decision Tree (Continuous Attributes)



Basic questions

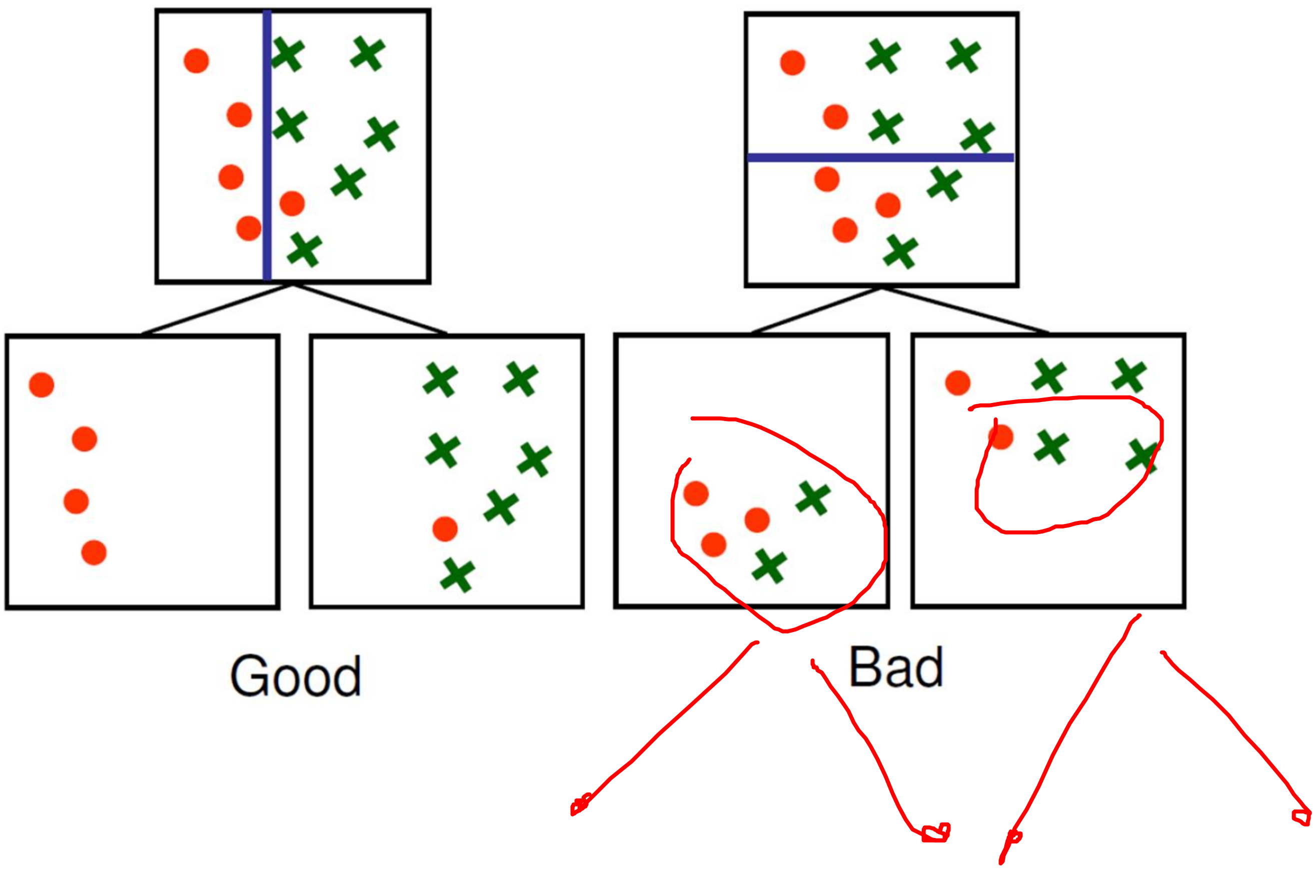
- How to choose the attribute to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

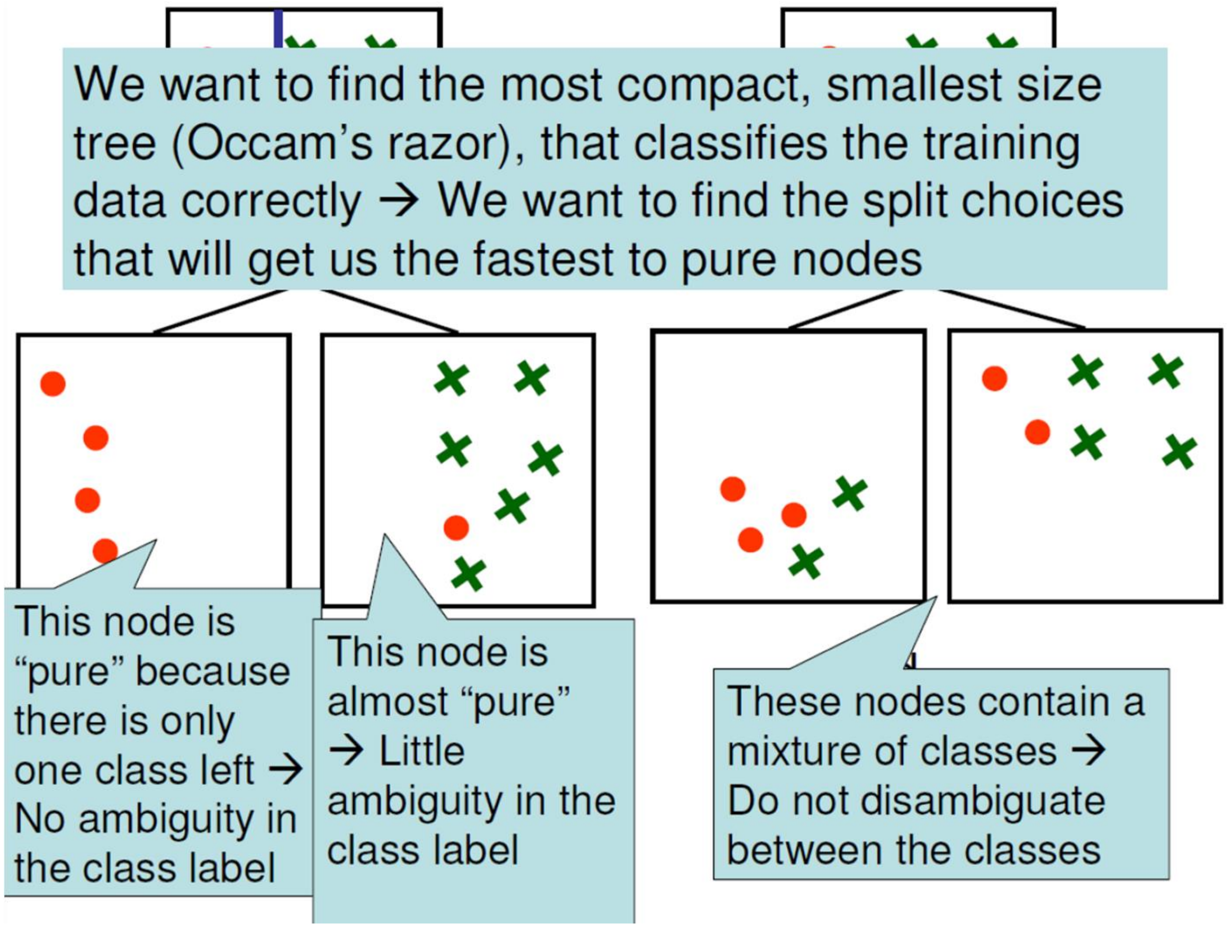
How to choose the attribute to split?



- Two classes (red circles/green crosses)
- Two attributes: x_1, x_2
- 11 points in training data
- Idea:
  - Construct a decision tree such that the leaf nodes predict correctly the class for all training samples

How to choose the attribute to split?




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

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) $\log_2$ (1/6) = 0.65

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

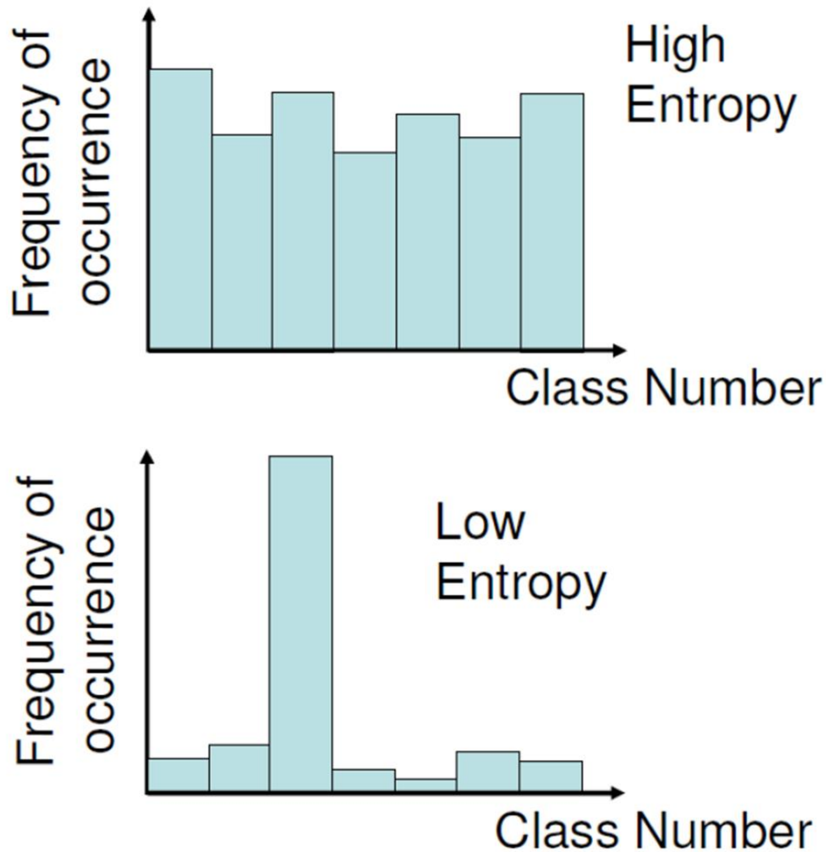
Entropy

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

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

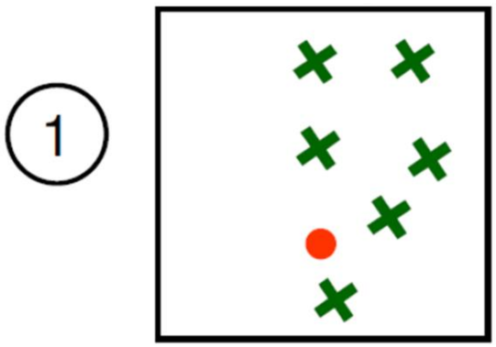
- P_i= probability of occurrence of vlaue I
  - High entropy -> all classes are nearly equally likely
  - Low entropy -> a few classes are likely; most of the classes are rarely observed

Entropy



The entropy captures the degree of "purity" of the distribution

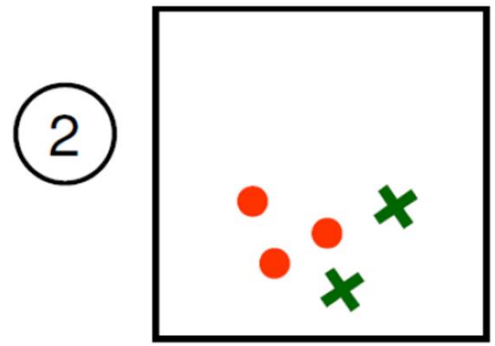
Example entropy calculation



$N_A = 1$
$N_B = 6$

$p_A = N_A/(N_A+N_B) = 1/7$
$p_B = N_B/(N_A+N_B) = 6/7$

$H_1 = -p_A \log_2 p_A - p_B \log_2 p_B$
$= 0.59$



$N_A = 3$
$N_B = 2$

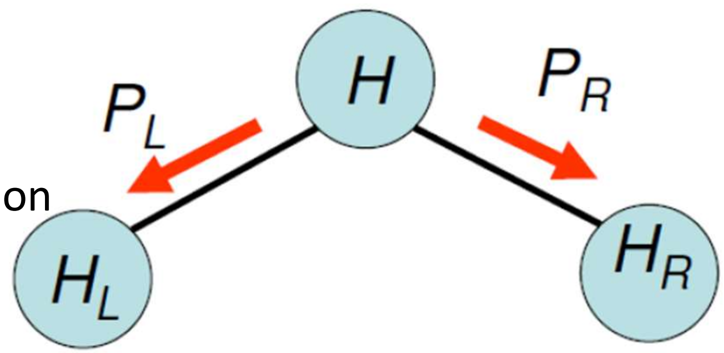
$p_A = N_A/(N_A+N_B) = 3/5$
$p_B = N_B/(N_A+N_B) = 2/5$

$H_2 = -p_A \log_2 p_A - p_B \log_2 p_B$
$= 0.97$

$H_1 < H_2$ => (2) less pure than (1)

Conditional entropy

Entropy before splitting: H



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

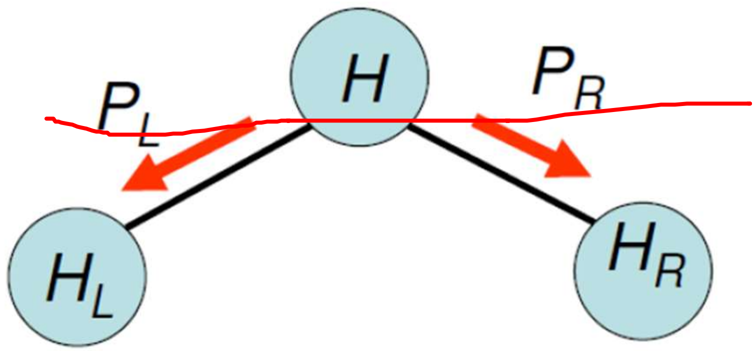
After splitting, a fraction P_R of data goes to the right node, which has entropy H_R.

The average entropy after splitting is:

$$H_L \times P_L + H_R \times P_R$$

Probability that a random input is directed to the left node

Information Gain



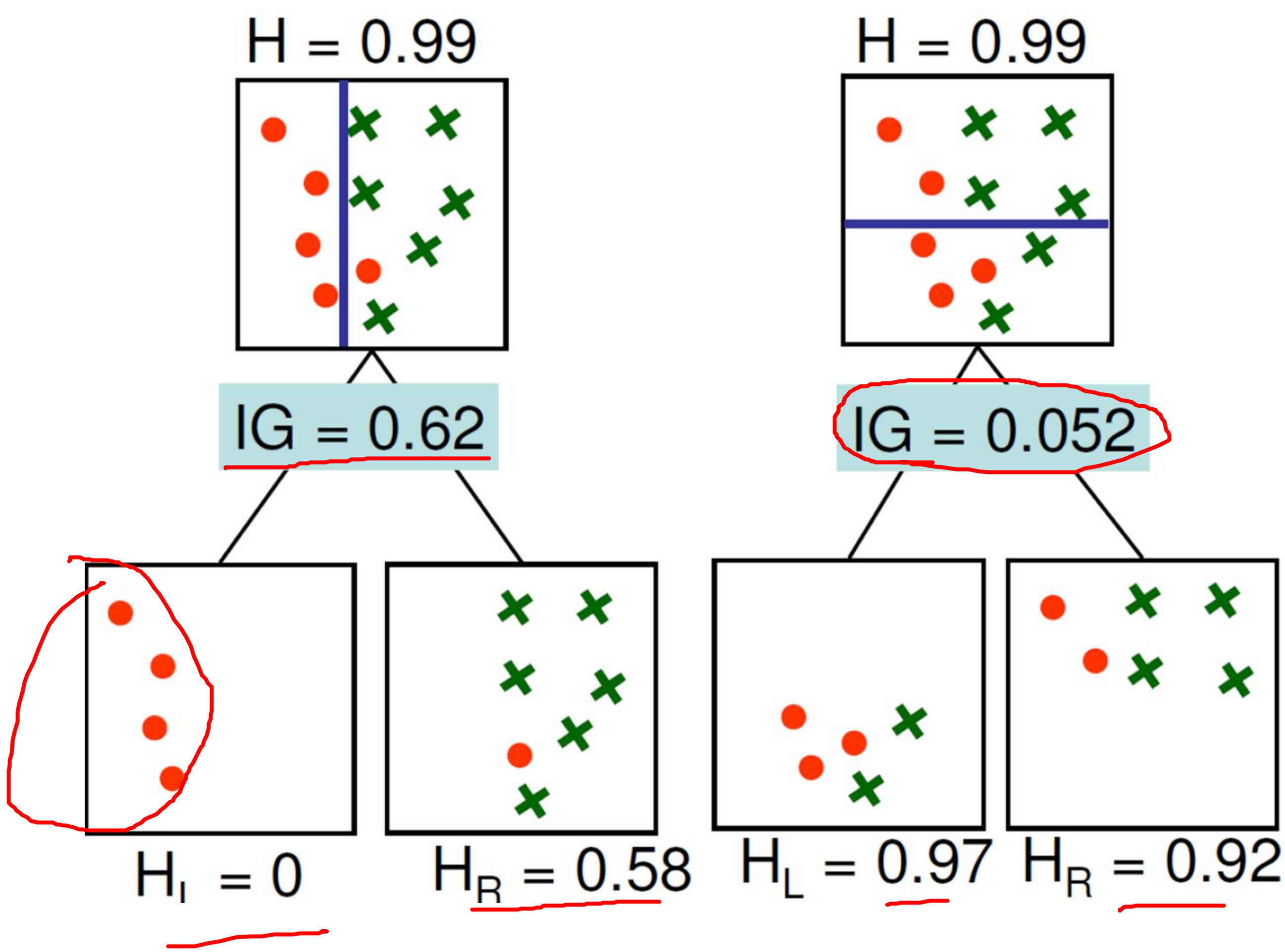
- We want nodes as pure as possible
  - We want to reduce the entropy as much as possible
  - We want to maximize the difference between the entropy of the parent node and the expected entropy of the children.

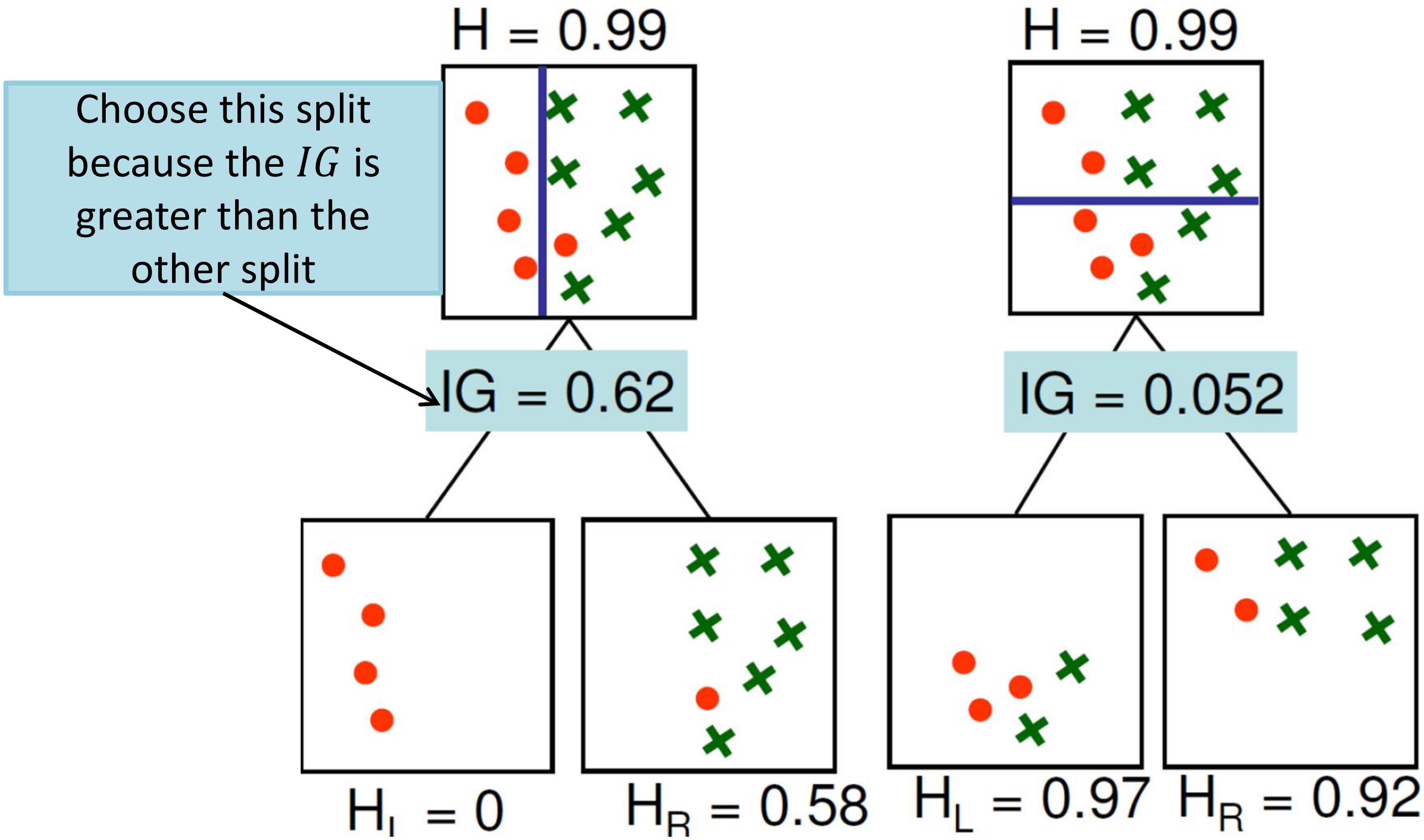
$$IG = H - (H_L \times P_L + H_R \times P_R)$$

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

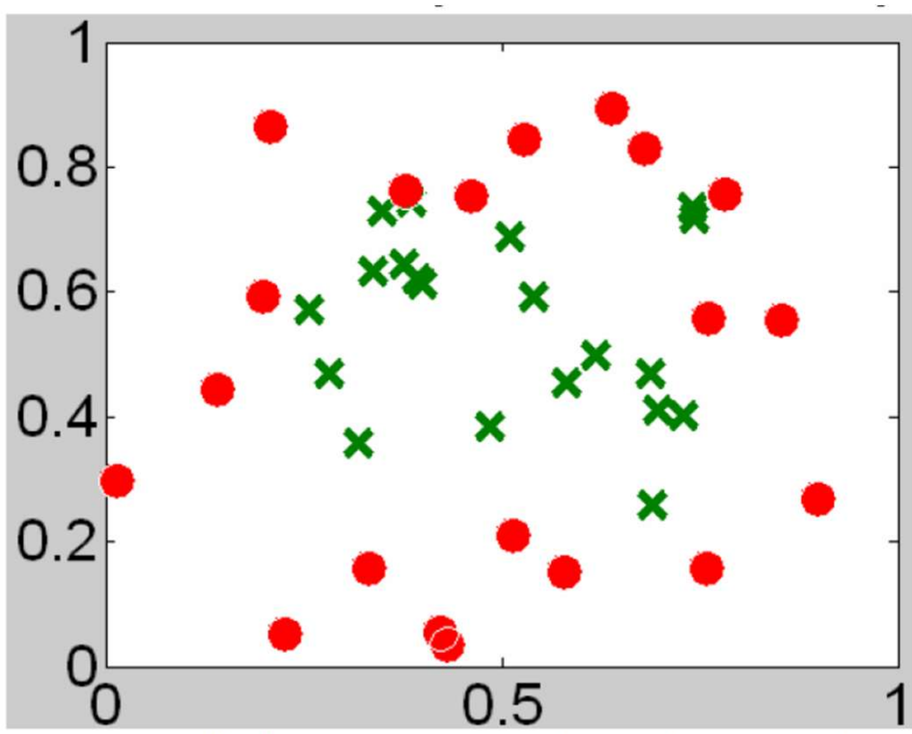
Notations

- 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

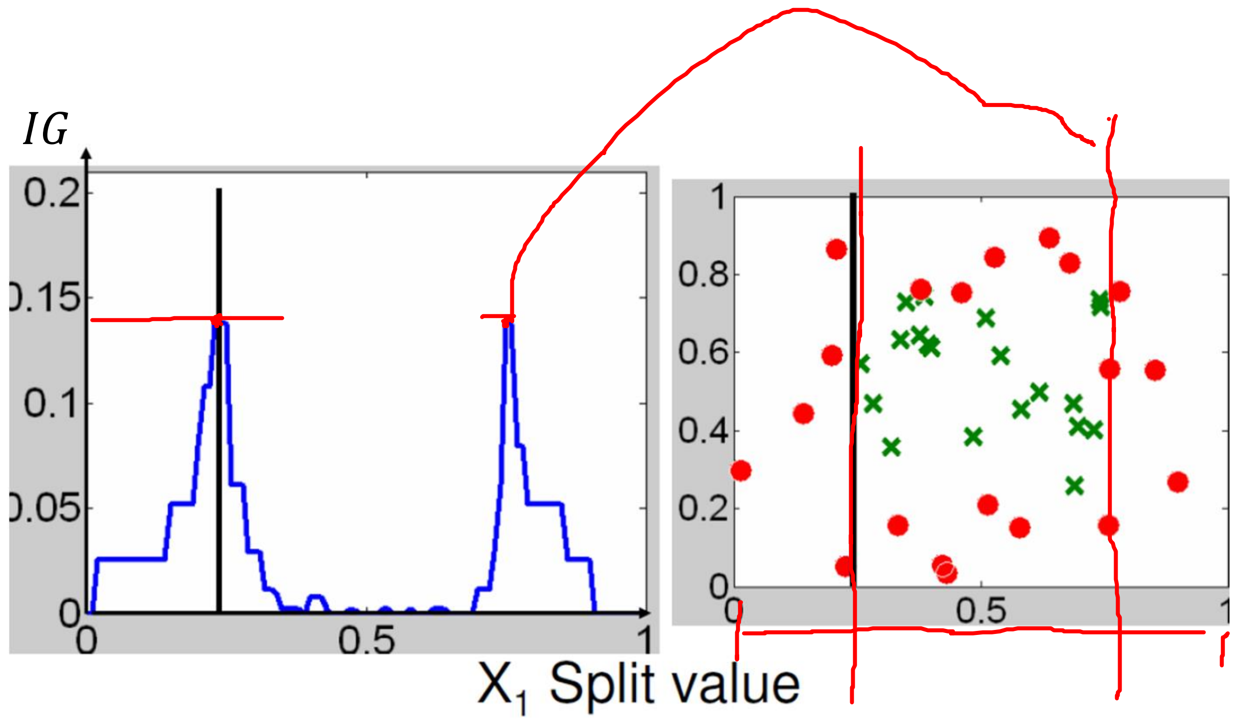

H = 0.99
IG = 0.62
H_l = 0
H_R = 0.58
H = 0.99
IG = 0.052
H_L = 0.97
H_R = 0.92


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

A complete example

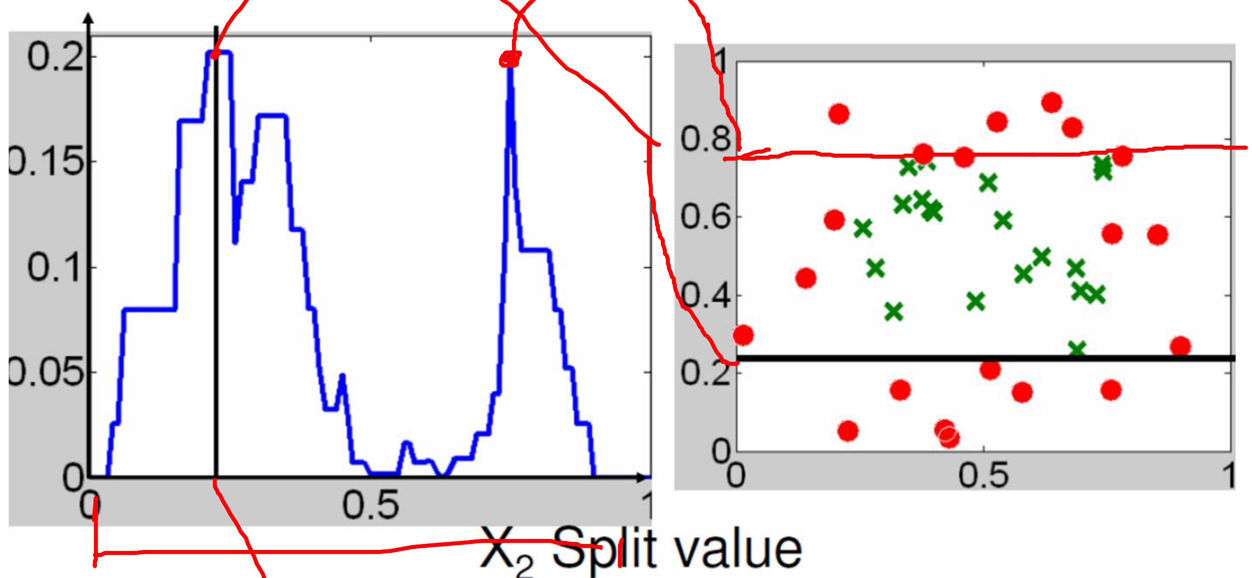


20 training examples from class A
20 training examples from class B
Attributes = x_1 and x_2 coordinates



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

I G

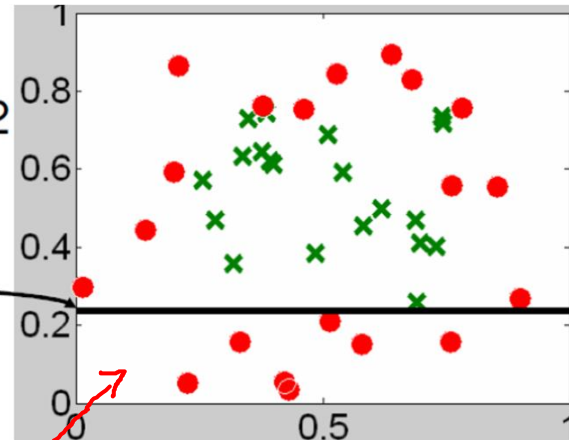


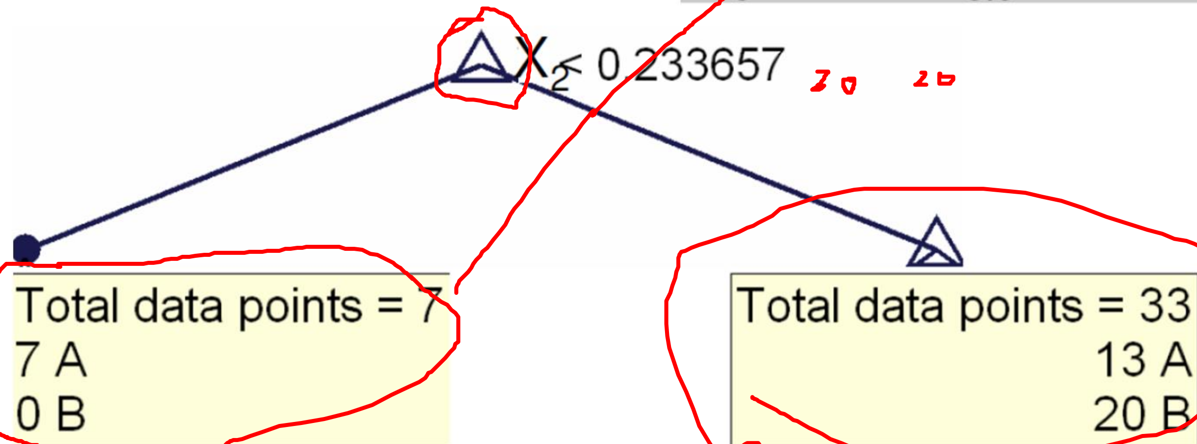
Best split value (max Information Gain) for X_2 attribute: 0.234 with IG = 0.202

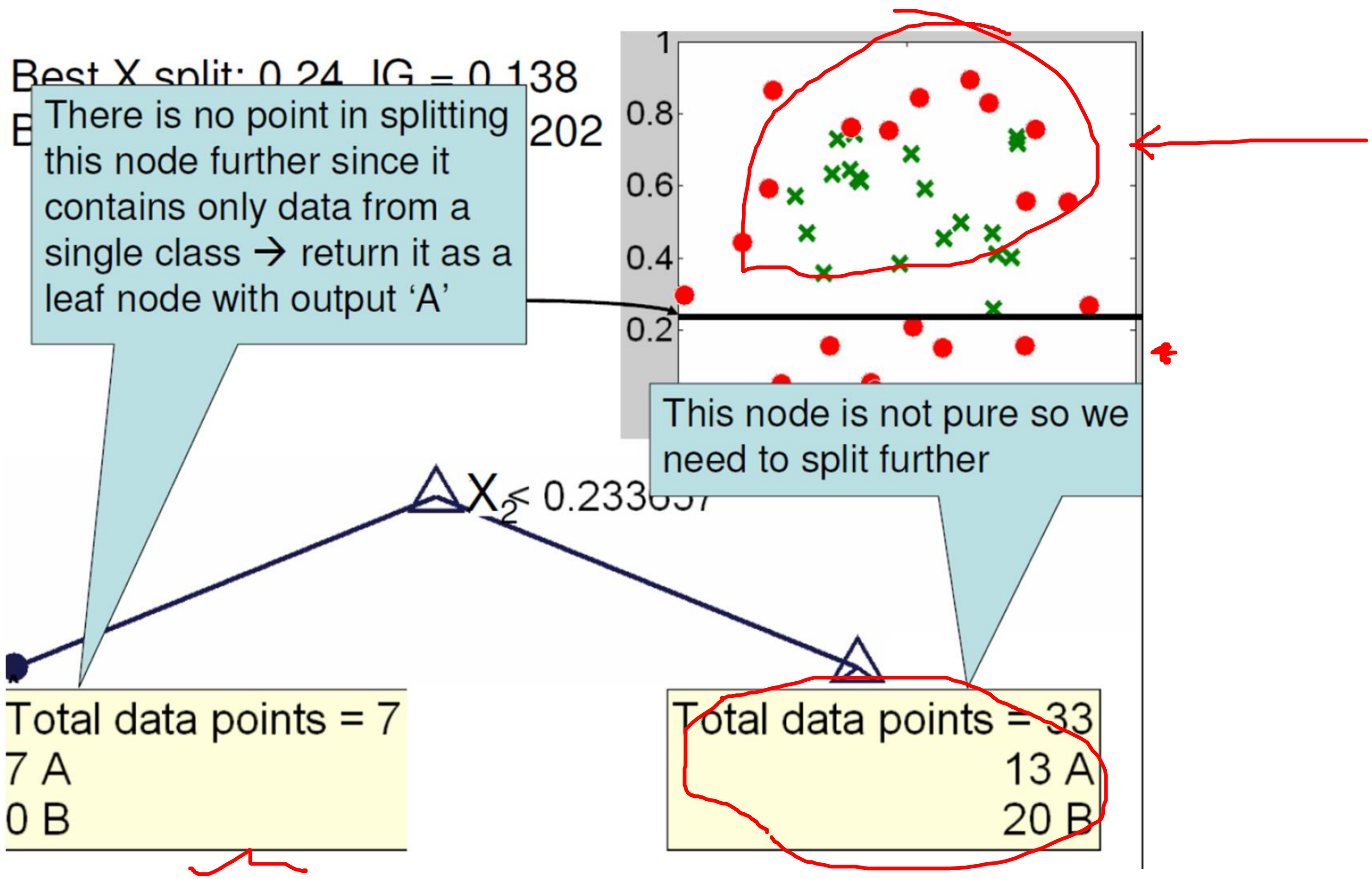
Best X_1 split: 0.24, IG = 0.138

Best X_2 split: 0.234, IG = 0.202

Split on X_2 with 0.234

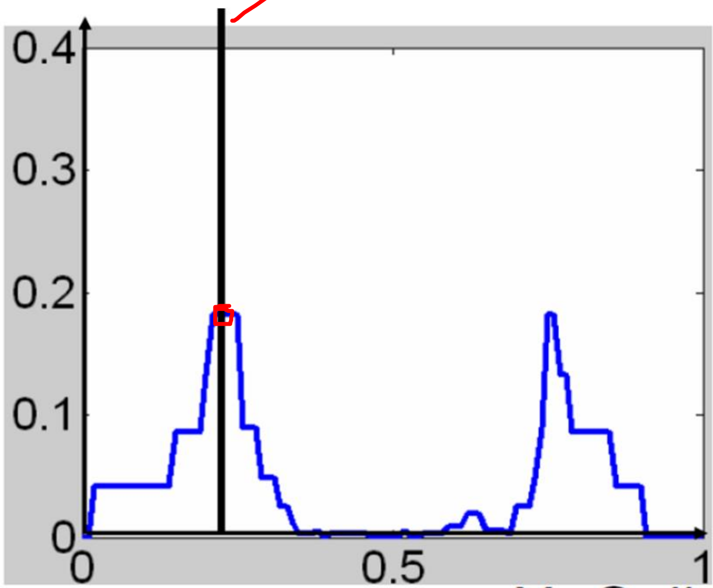


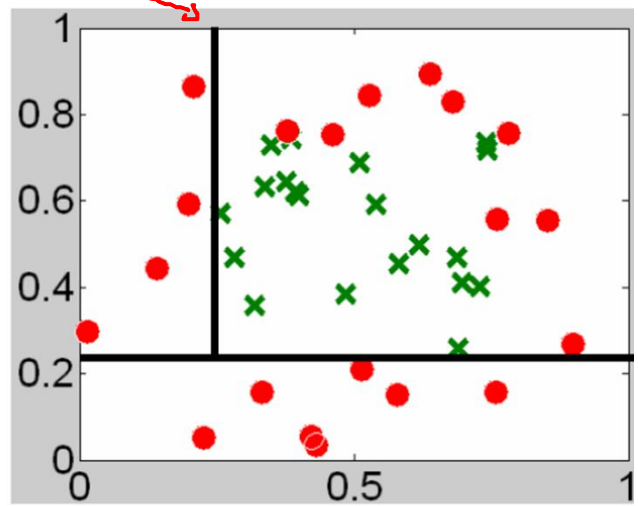




Best X split: 0.24, IG = 0.138
202
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output 'A'
1
0.8
0.6
0.4
0.2
This node is not pure so we need to split further
X_2 < 0.233
Total data points = 7
7 A
0 B
Total data points = 33
13 A
20 B

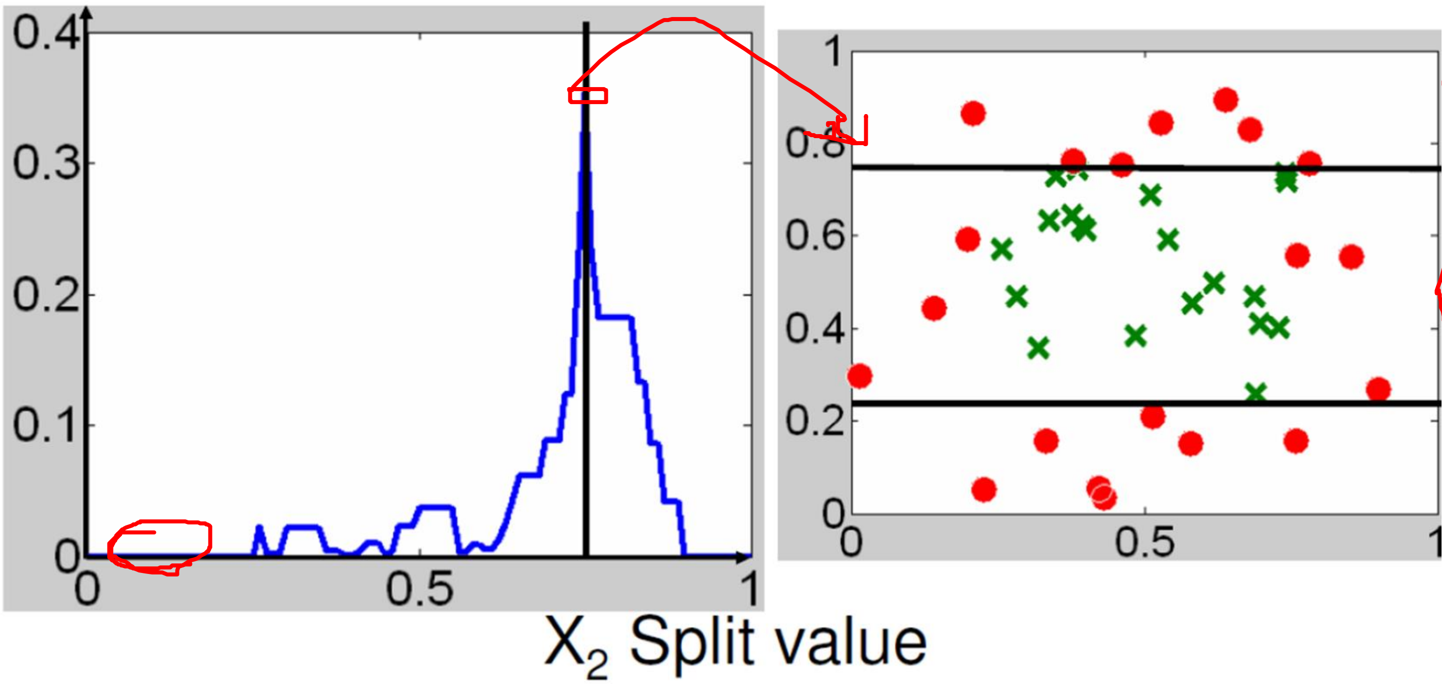
IG





X_1 Split value

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

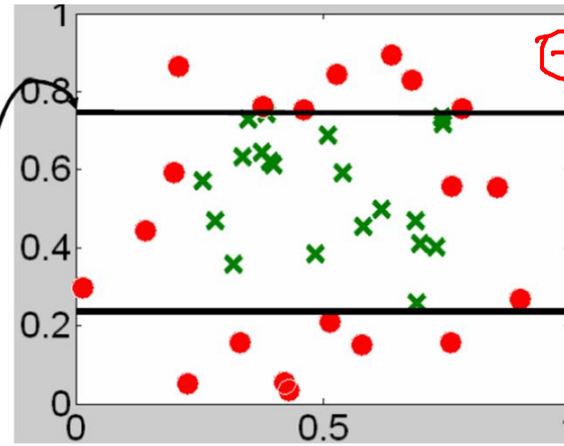


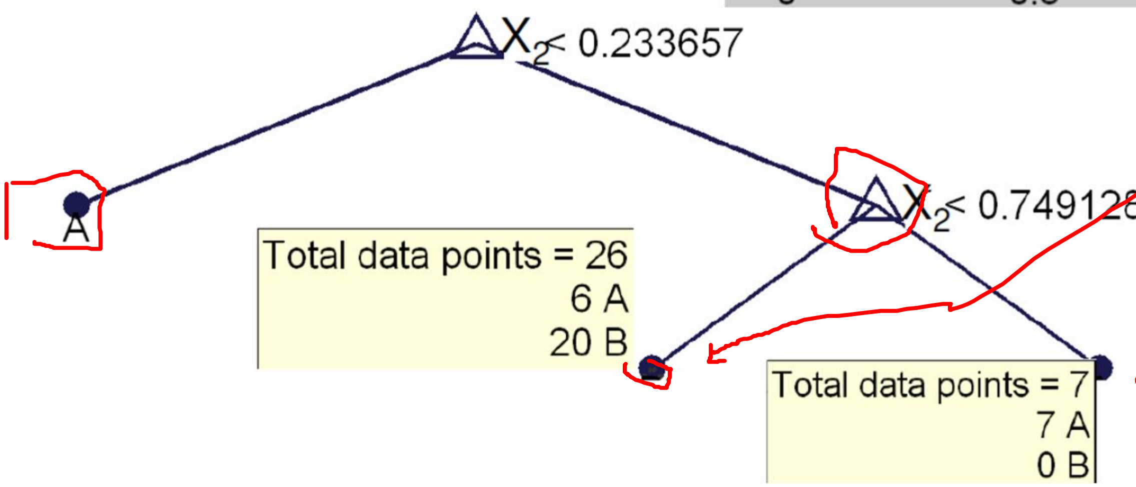
Best split value (max Information Gain) for X_2 attribute: 0.75 with IG ~ 0.353

Best X_1 split: 0.22, IG = 0.182

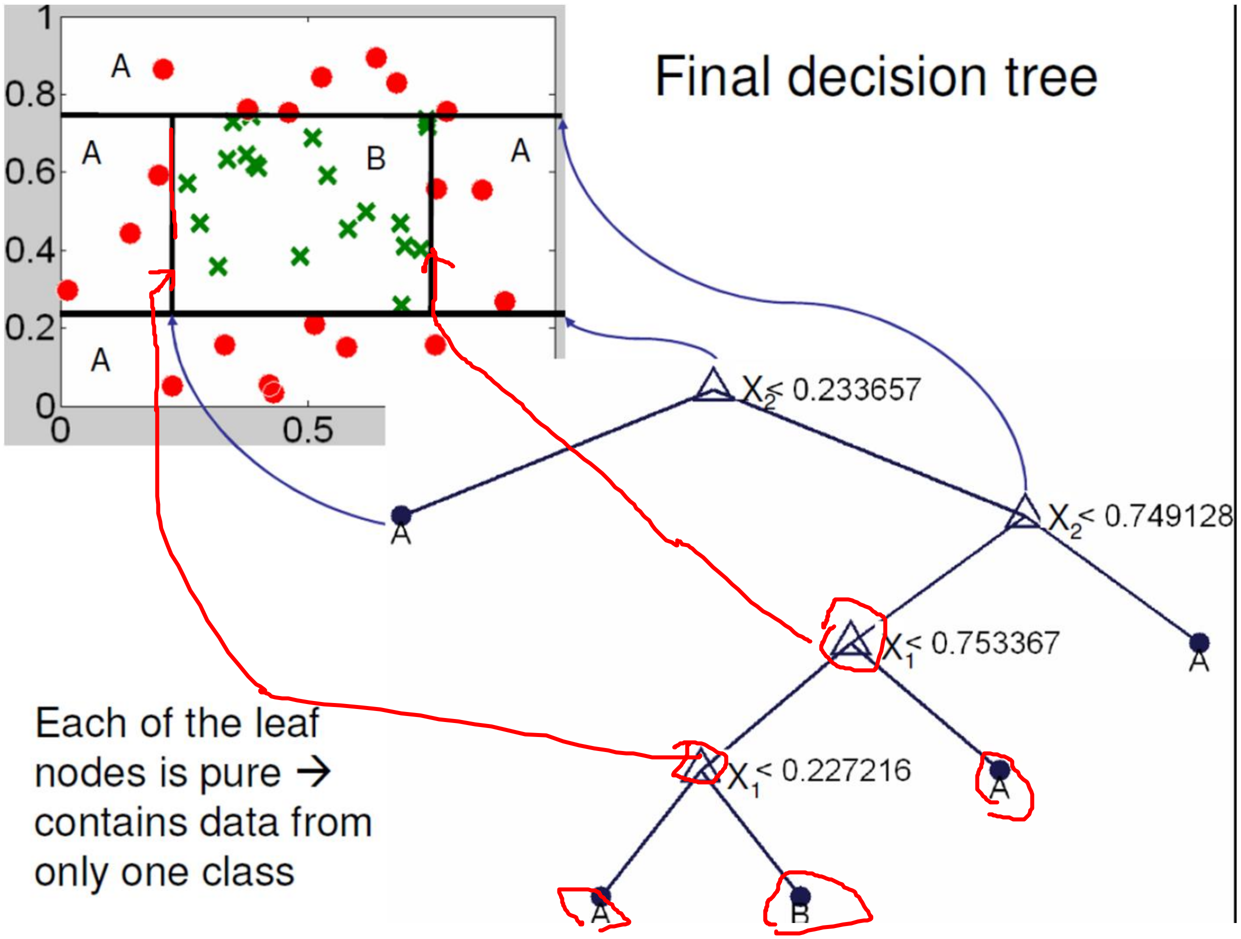
Best X_2 split: 0.75, IG = 0.353

Split on X_2 with 0.75

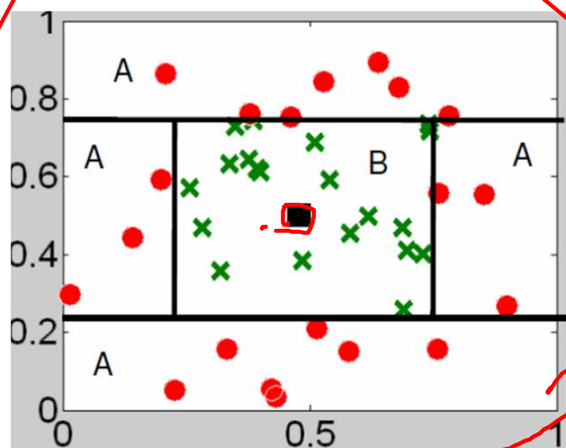




Final decision tree



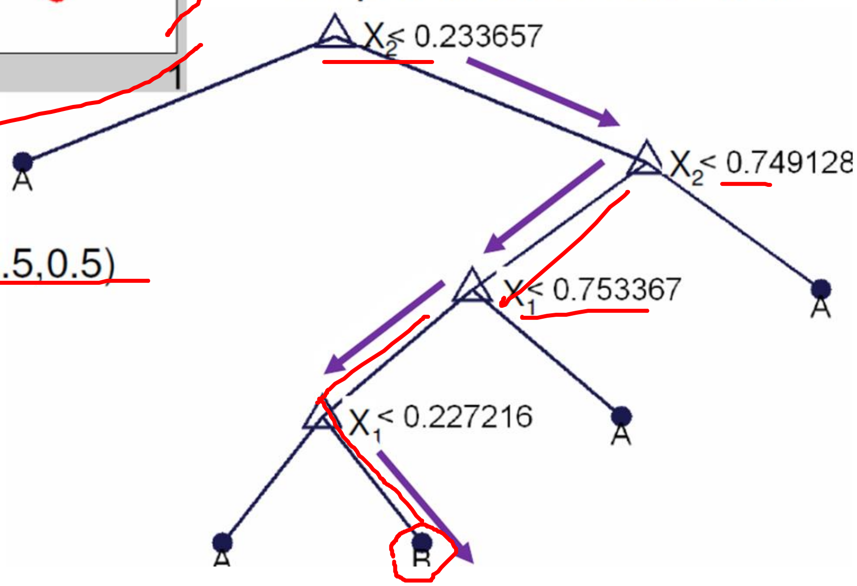
Each of the leaf nodes is pure → contains data from only one class



Final decision tree

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

Return corresponding output class for this leaf

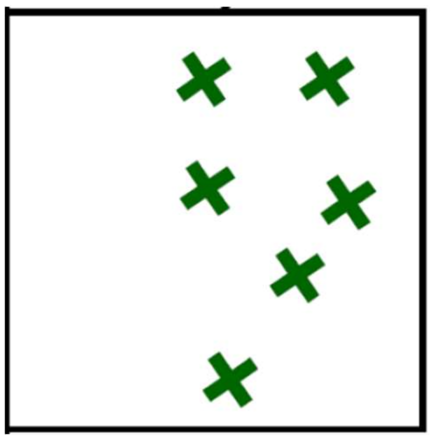


Example (X,Y) = (0.5,0.5)

Basic questions

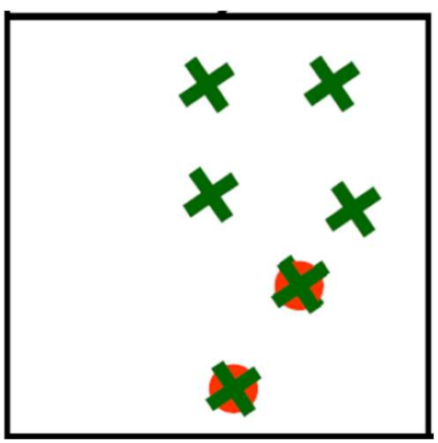
- How to choose the attribute to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?

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 node and stop splitting. The leaf represents the class of the label in the data



Several data points have exactly the same attributes even though they are from different class.

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.

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$_v$ ← LearnTree(X_v, Y_v)

Decision tree 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.

- Problem (next):
  - Computational issues
  - The tree will end up being too large => pruning
  - Evaluating the tree on the training data is dangerous => overfitting

Basic questions

- How to choose the attribute 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?

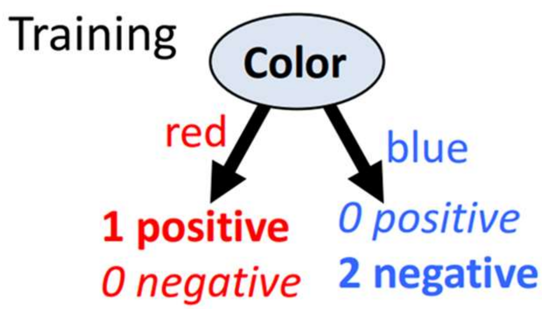
- Acquire more training data
- Remove irrelevant attributes
- Grow full tree and the post-prune
- Ensemble learning

Reduce-error pruning

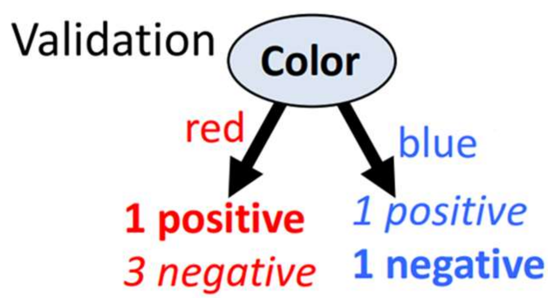
- Split data into training and validation sets
- Grow tree based on training set
- Do until further pruning is harmful
  - Evaluating impact on validation set of pruning each possible node
  - Greedily remove the node that most improves validation set accuracy

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