Decision Trees

Manuela Veloso

Co-instructor: Pradeep Ravikumar

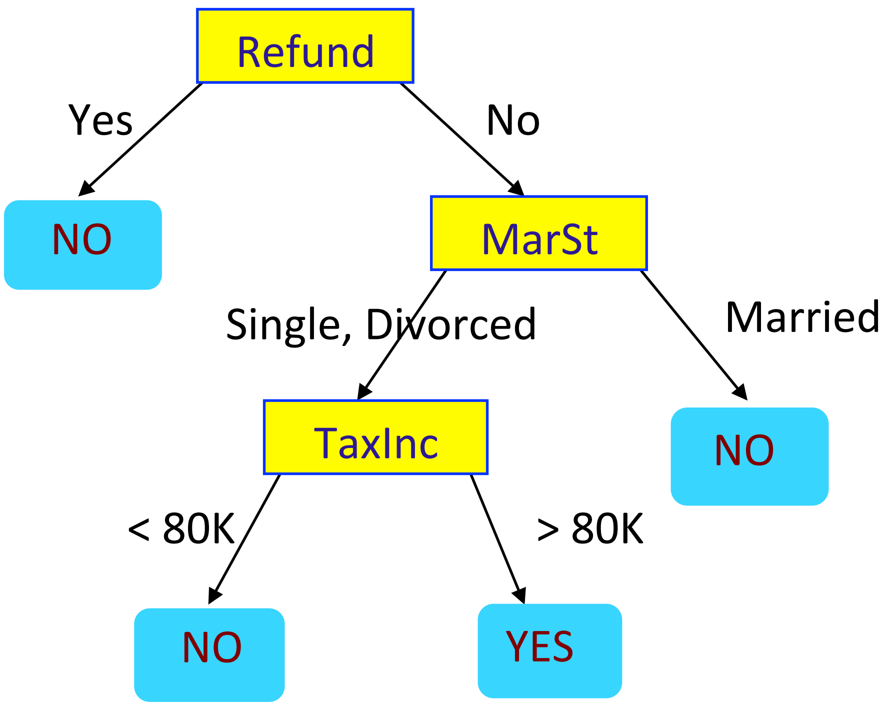
Machine Learning 10-701

(Readings: Mitchell ch.3, and thanks to Andrew Moore)

Decision Trees

- Start with discrete features, then discuss continuous
- What is a decision tree and what does it represent

Decision Tree for Tax Fraud Detection



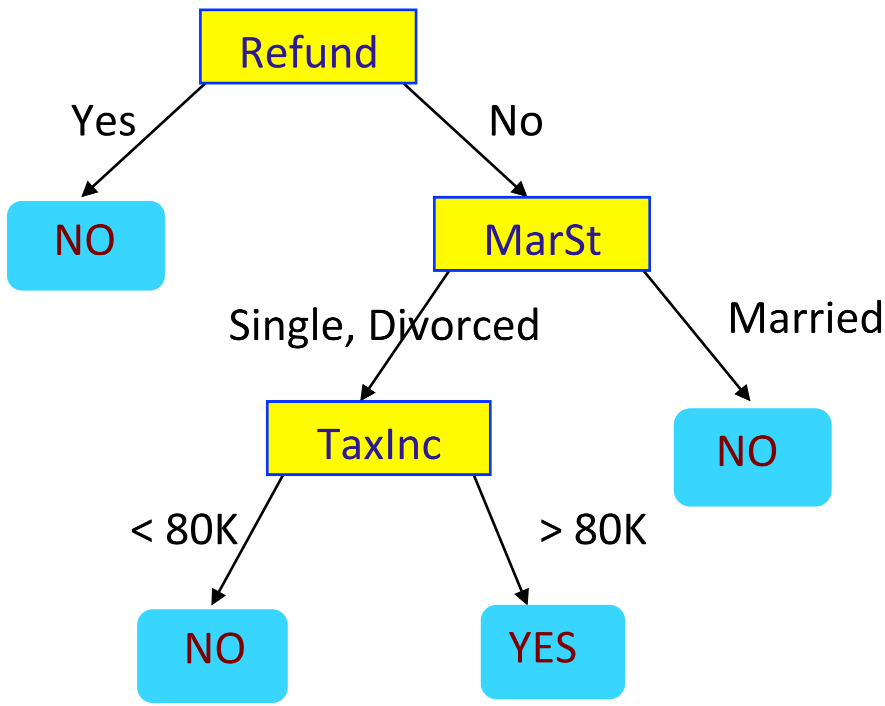
X_1	X_2	X_3	Y
Refund	**Marital Status**	**Taxable Income**	**Cheat**

- Each internal node: test one feature X_i
- Each branch from a node: selects some value for X_i
- Each leaf node: prediction for Y

Prediction

- Given a decision tree, how do we assign label to a test point

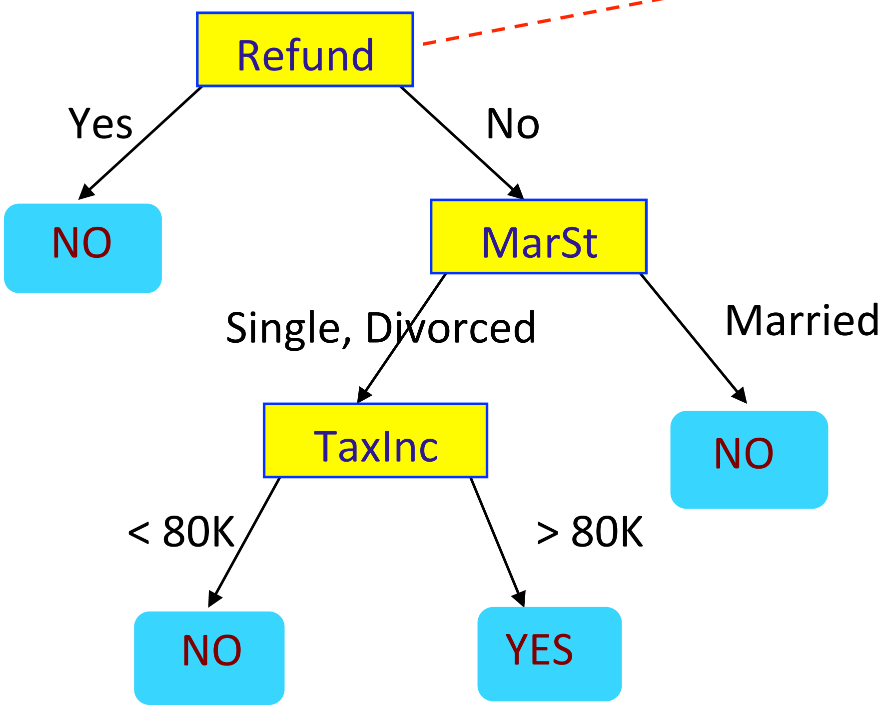
Decision Tree for Tax Fraud Detection



Query Data

X_1	X_2	X_3	Y
Refund	**Marital Status**	**Taxable Income**	**Cheat**
No	Married	80K	**?**

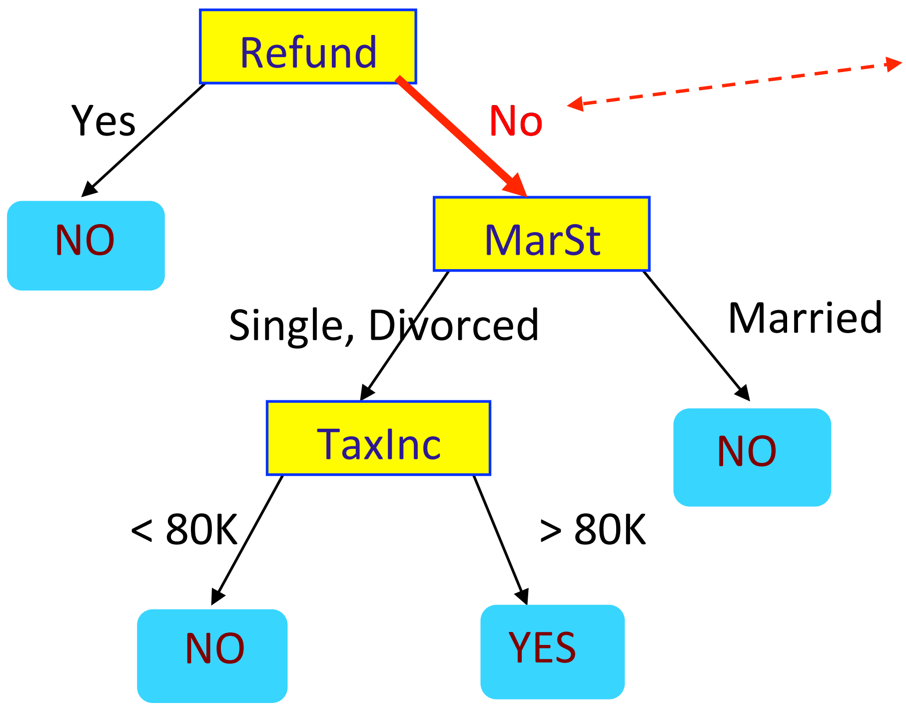
Decision Tree for Tax Fraud Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

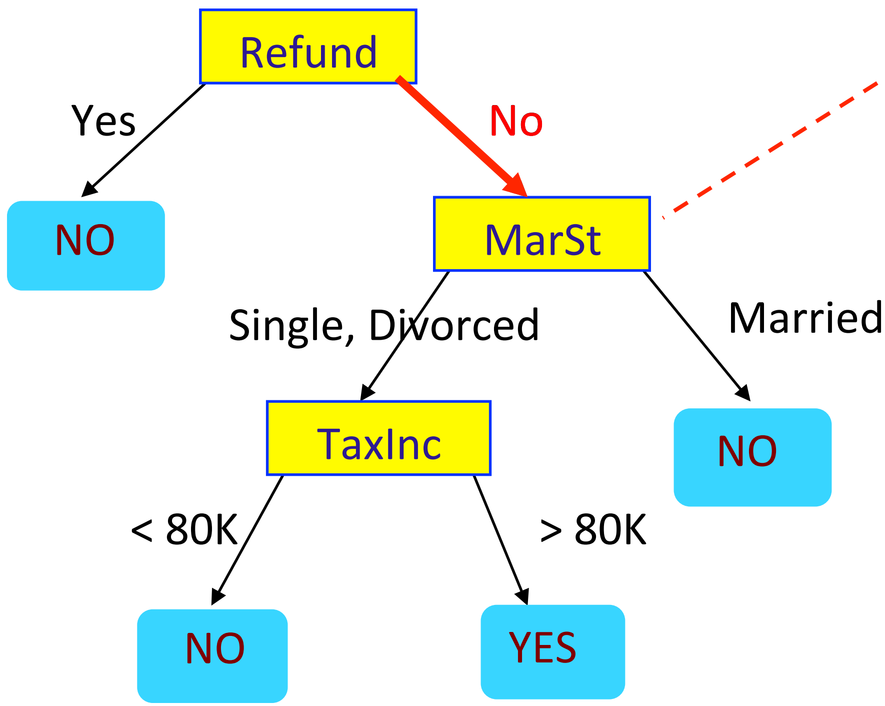
Decision Tree for Tax Fraud Detection



Query Data

X_1	X_2	X_3	Y
Refund	**Marital Status**	**Taxable Income**	**Cheat**
No	Married	80K	**?**

Decision Tree for Tax Fraud Detection



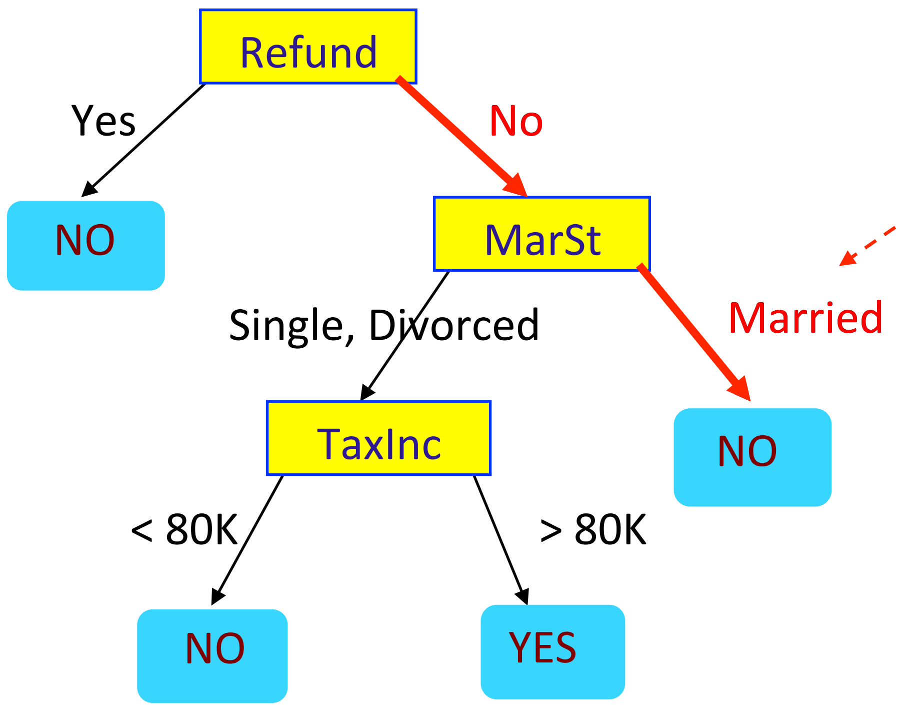
Query Data

X_1	X_2	X_3	Y
Refund	**Marital Status**	**Taxable Income**	**Cheat**
No	Married	80K	?

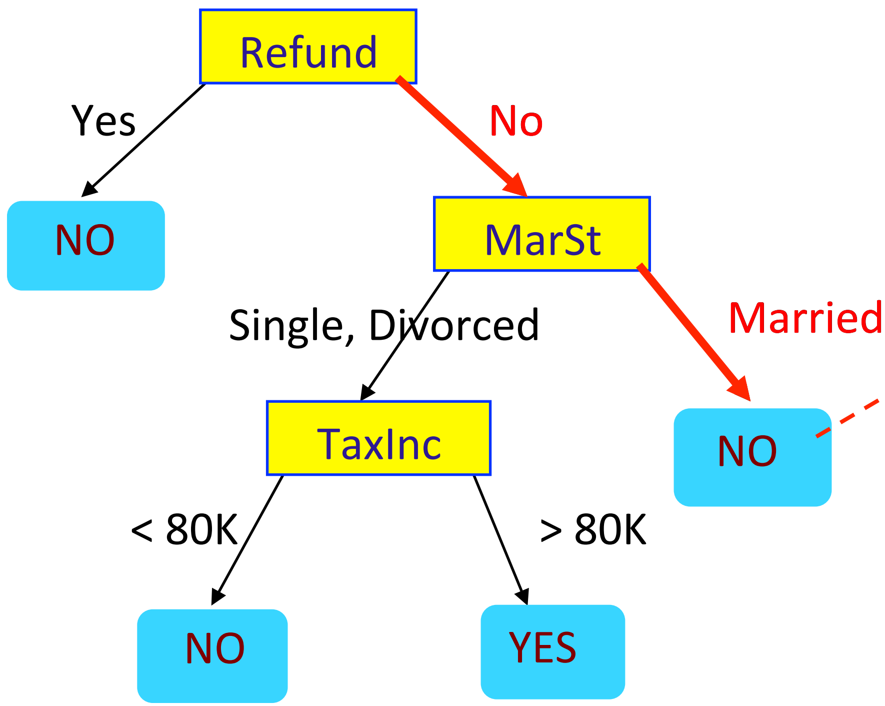
Decision Tree for Tax Fraud Detection

Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?



Decision Tree for Tax Fraud Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

Assign Cheat to "No"

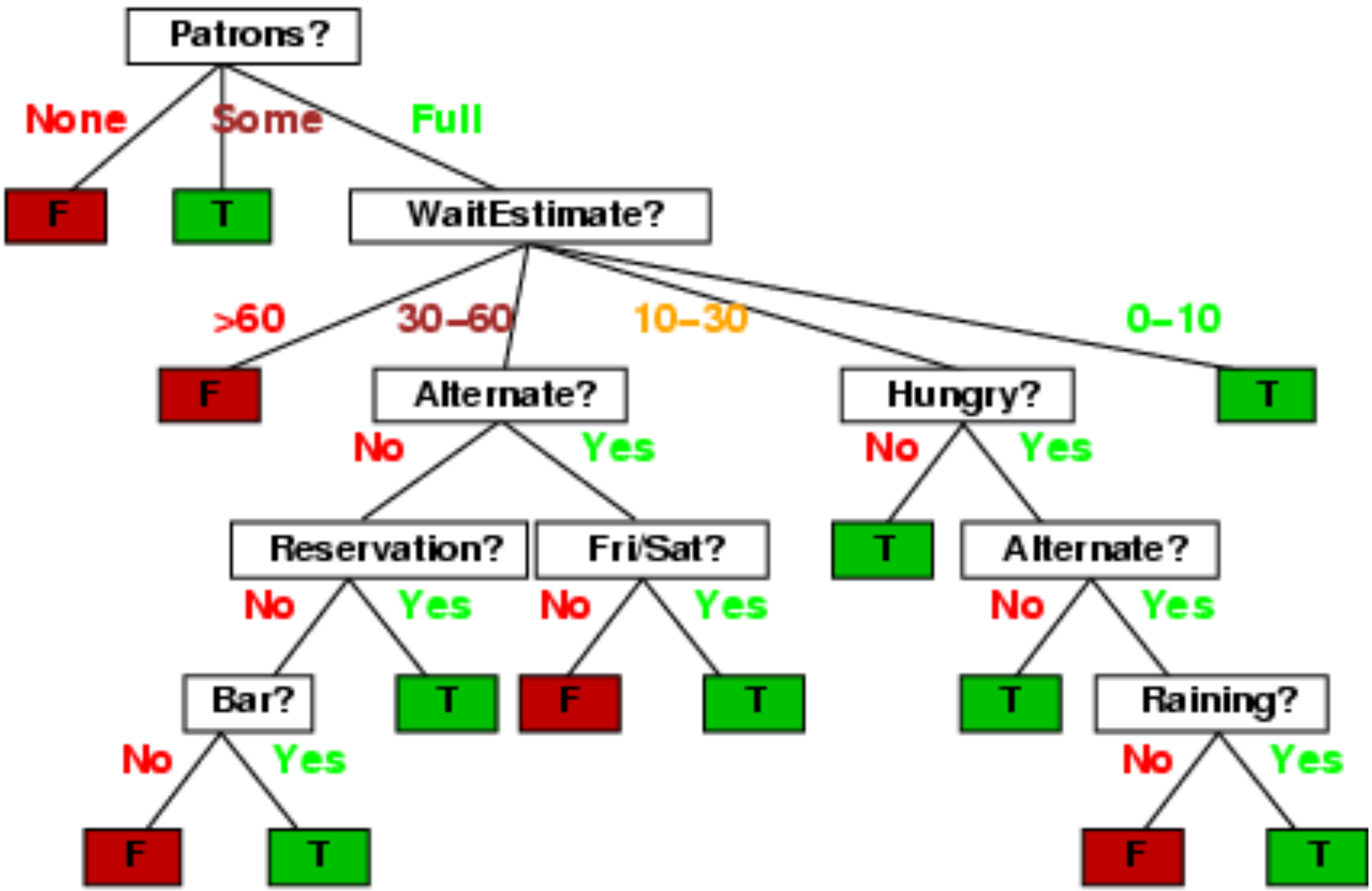
Example Data

- Examples described by features/attributes (Boolean, discrete, continuous)
- Function is class; example wait/not wait for table at restaurant

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	$Wait$
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Data Representation - Decision Trees

- The "true" tree for deciding whether to wait:



So far…

- What does a decision tree represent
- Given a decision tree, how do we assign label to a test point

Now …

- How to learn a decision tree from training data

Questions

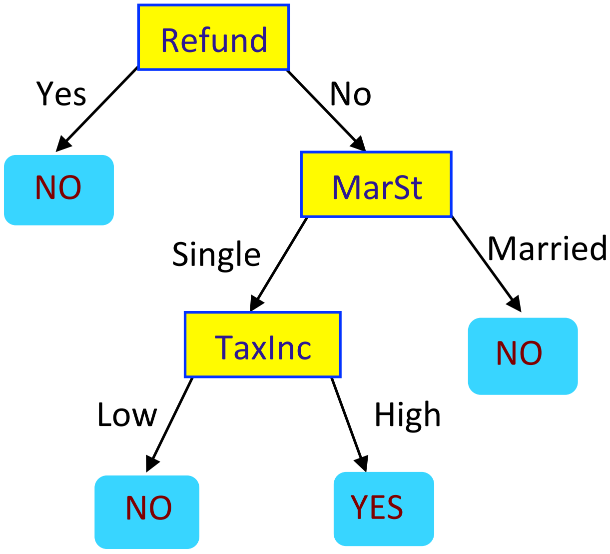
- How **to choose the feature (attribute) to split** on at each level of the tree?
- When **to stop splitting**? When should a node be declared a leaf?
- How should the **class label be assigned**?
- If the tree is too large, how can it be **pruned**?

How to learn a decision tree

- Top-down induction [ID3]

Main loop:

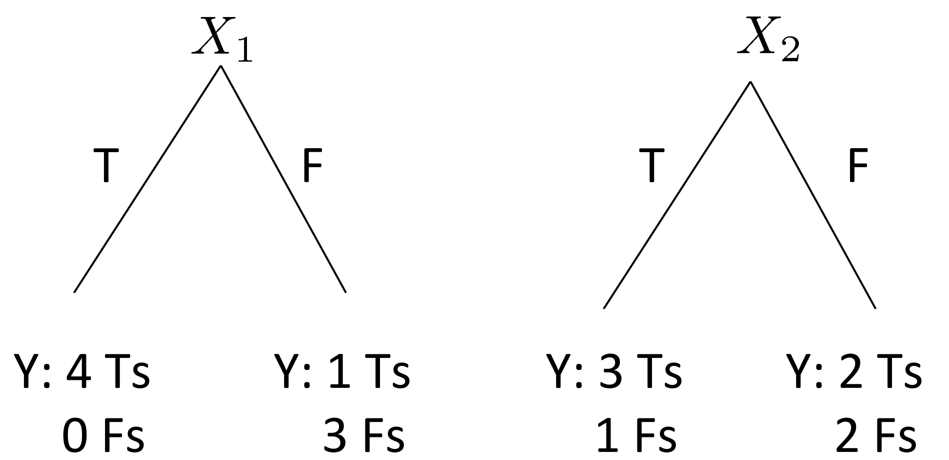
1. $X \leftarrow$ the $\boxed{\text{“best”}}$ decision feature for next *node*
2. Assign X as decision feature for *node*
3. For each value of X, create new descendant of *node* (Discrete features)
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes (steps 1-5) after removing current feature



6. When all features exhausted, assign majority label to the leaf node

Which feature is best?

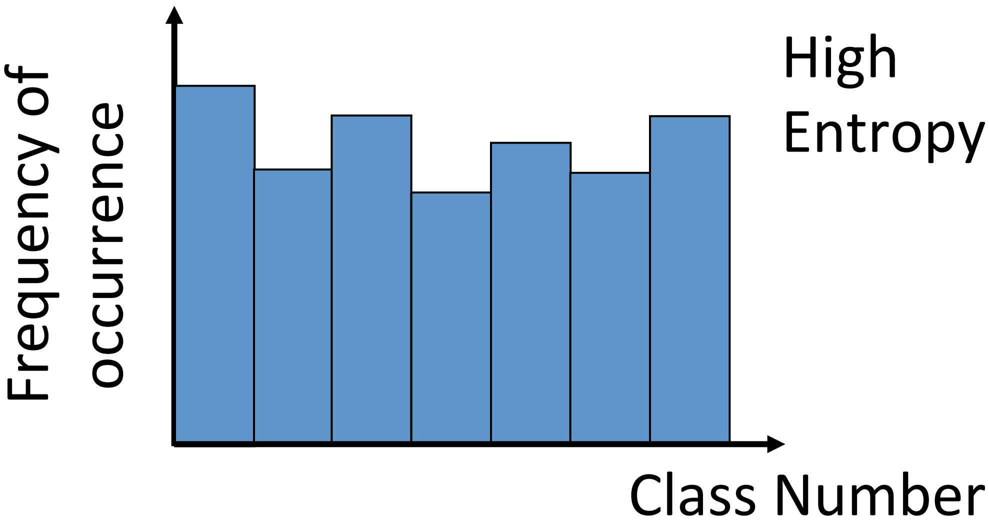
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F

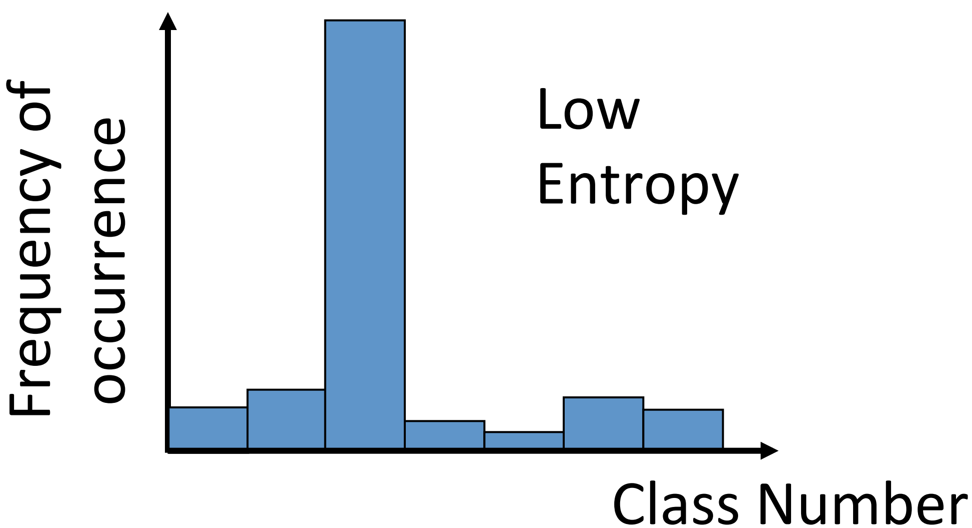


Absolutely sure | Kind of sure | Kind of sure | Absolutely unsure

Good split if we are more certain about classification after split – Uniform class distribution is bad…

Entropy *(Shannon and Weaver 1949)*





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

Entropy – Example

$$\text{Entropy(S)} = \sum_{i=1}^{c} -p_i \log_2 p_i$$

- $|S| = 12$, $c = 2(+,-)$, $|S_+| = 4$, $|S_-| = 8$

$$\begin{aligned} E(S) &= -\frac{4}{12}\log_2\frac{4}{12} - \frac{8}{12}\log_2\frac{8}{12} \\ &= 0.918 \end{aligned}$$

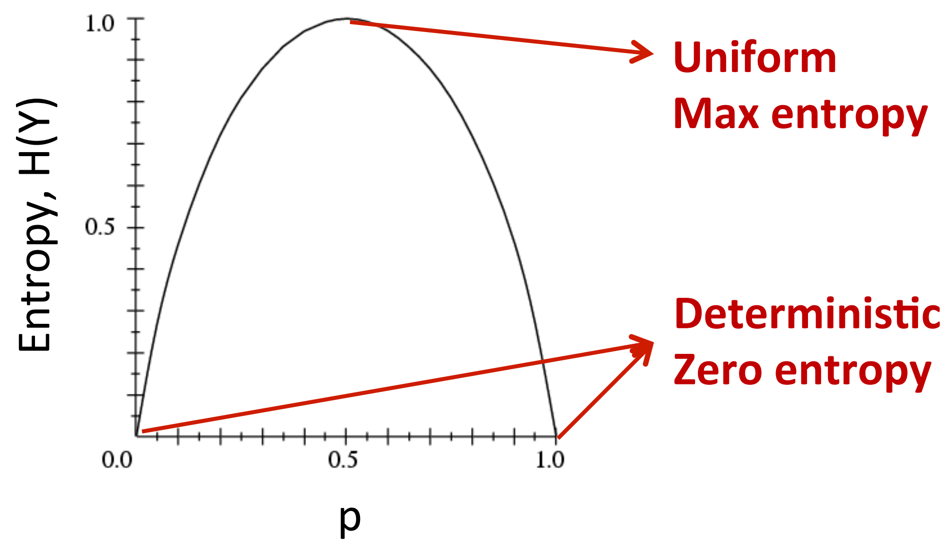
Entropy

- Entropy of a random variable Y

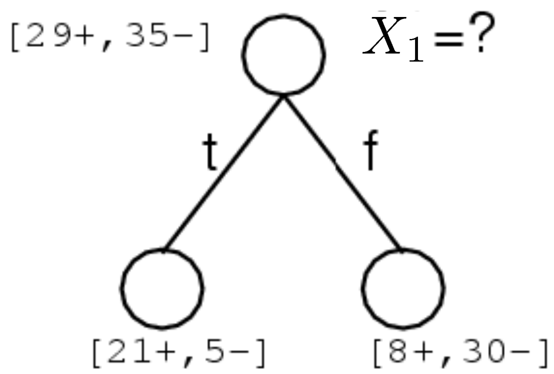
$$H(Y) = -\sum_y P(Y=y)\log_2 P(Y=y)$$

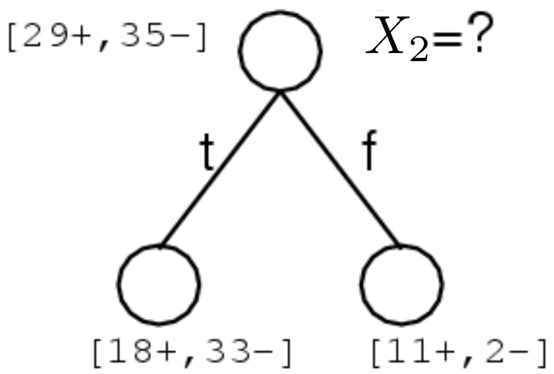
More uncertainty, more entropy!

Y ~ Bernoulli(p)



Which feature is best?





H(Y) – entropy of Y **$H(Y|X_i)$ – conditional entropy of Y**

Pick the attribute/feature which yields maximum **information gain**:

$$\arg\max_i I(Y, X_i) = \arg\max_i [H(Y) - H(Y|X_i)]$$

Information Gain

- Advantage of attribute = decrease in uncertainty
  - Entropy of Y before split

$$H(Y) = -\sum_y P(Y=y)\log_2 P(Y=y)$$

  - Entropy of Y after splitting based on X_i
    - Weight by probability of following each branch

$$\begin{aligned} H(Y \mid X_i) &= \sum_x P(X_i = x) H(Y \mid X_i = x) \\ &= -\sum_x P(X_i = x) \sum_y P(Y = y \mid X_i = x) \log_2 P(Y = y \mid X_i = x) \end{aligned}$$

- Information gain is difference

$$I(Y, X_i) = H(Y) - H(Y \mid X_i)$$

Max Information gain = min conditional entropy

Which feature is best to split?

Pick the attribute/feature which yields maximum information gain:

$$\arg\max_i I(Y, X_i) = \arg\max_i [H(Y) - H(Y|X_i)]$$

$$= \arg\min_i H(Y|X_i)$$

Entropy of Y $\quad H(Y) = -\sum_y P(Y=y) \log_2 P(Y=y)$

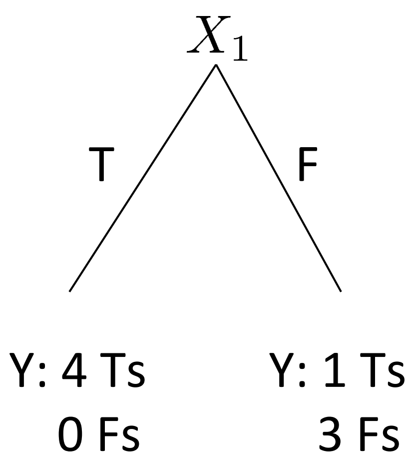
Conditional entropy of Y $\quad H(Y \mid X_i) = \sum_x P(X_i = x) H(Y \mid X_i = x)$

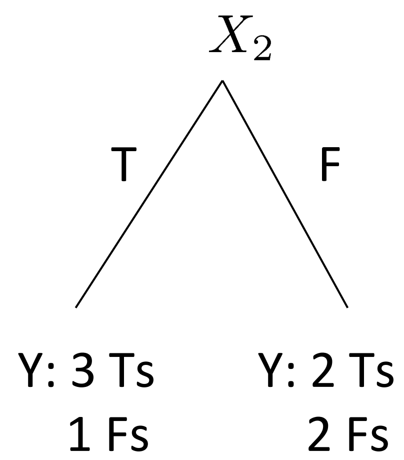
Feature which yields maximum reduction in entropy (uncertainty) provides maximum information about Y

Information Gain

$$H(Y \mid X_i) \;=\; -\sum_x P(X_i = x) \sum_y P(Y = y \mid X_i = x) \log_2 P(Y = y \mid X_i = x)$$

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





$$\widehat{H}(Y|X_1) = \cancel{-\frac{1}{2}[1\log_2 1 + 0\log_2 0]} - \frac{1}{2}[\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}]$$

$$\widehat{H}(Y|X_2) = -\frac{1}{2}[\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}] \underbrace{- \frac{1}{2}[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}]}_{> 0}$$

$$\widehat{H}(Y|X_1) < \widehat{H}(Y|X_2)$$

How to learn a decision tree

- Top-down induction [ID3]

Main loop:

1. $X \leftarrow$ the $\boxed{\text{“best”}}$ decision feature for next *node*
2. Assign X as decision feature for *node*
3. For each value of X, create new descendant of *node* (Discrete features)
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes (steps 1-5) after removing current feature

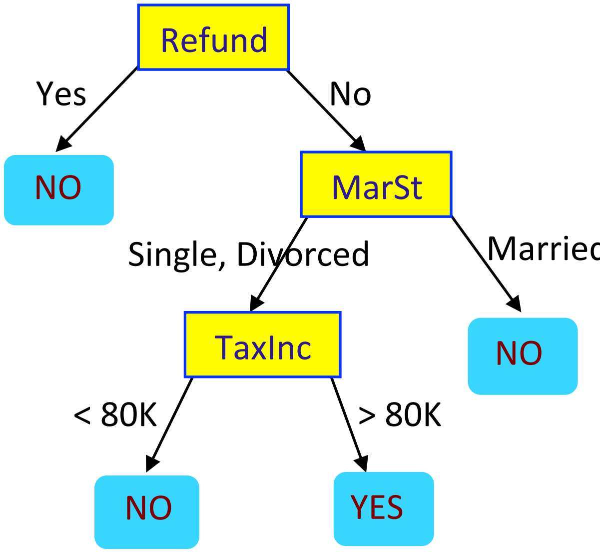
6. When all features exhausted, assign majority label to the leaf node

How to learn a decision tree

- Top-down induction [ID3, C4.5, C5, …]

Main loop: C4.5

1. $X \leftarrow$ the "best" decision feature for next *node*
2. Assign X as decision feature for *node*
3. For "best" split of X, create new descendants of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes
6. Prune back tree to reduce overfitting
7. Assign majority label to the leaf node



Handling continuous features (C4.5)

Convert continuous features into discrete by setting a threshold.

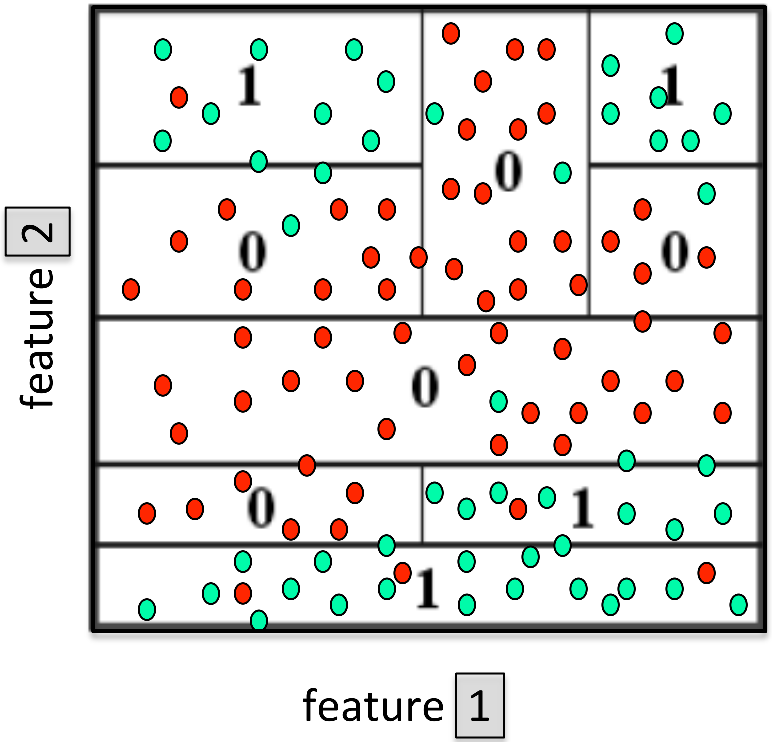
What threshold to pick?

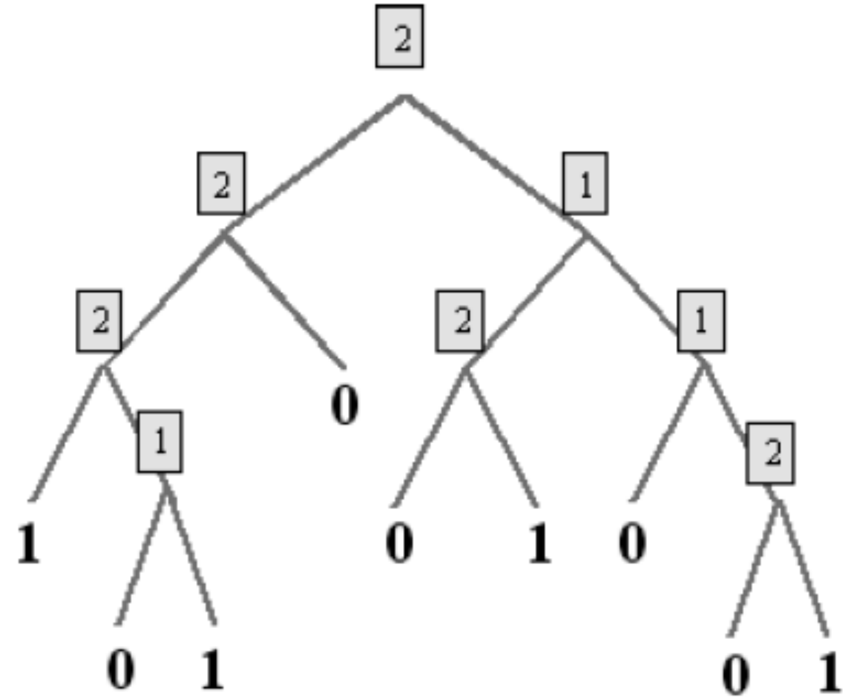
Search for best one as per information gain. Infinitely many??

Don't need to search over more than ~ n (number of training data),e.g. say X_1 takes values $x_1^{(1)}, x_1^{(2)}, \dots, x_1^{(n)}$ in the training set. Then possible thresholds are

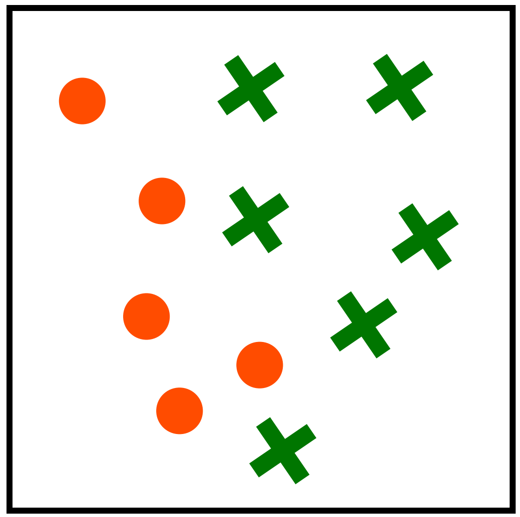
$$[x_1^{(1)} + x_1^{(2)}]/2,\ [x_1^{(2)} + x_1^{(3)}]/2,\ \dots,\ [x_1^{(n-1)} + x_1^{(n)}]/2$$

Dyadic decision trees
(split on mid-points of features)



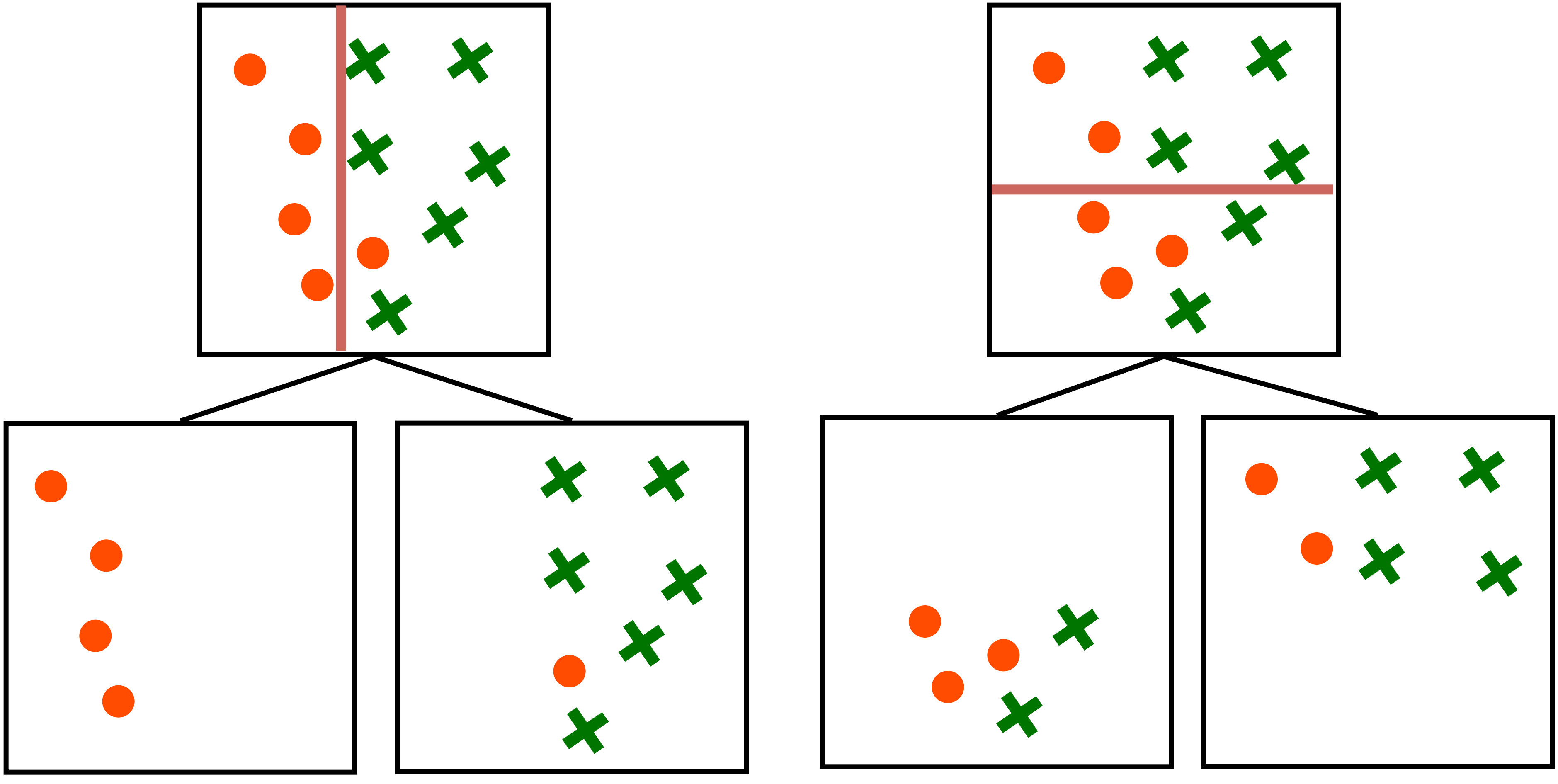


Continuous Case- How to Split?



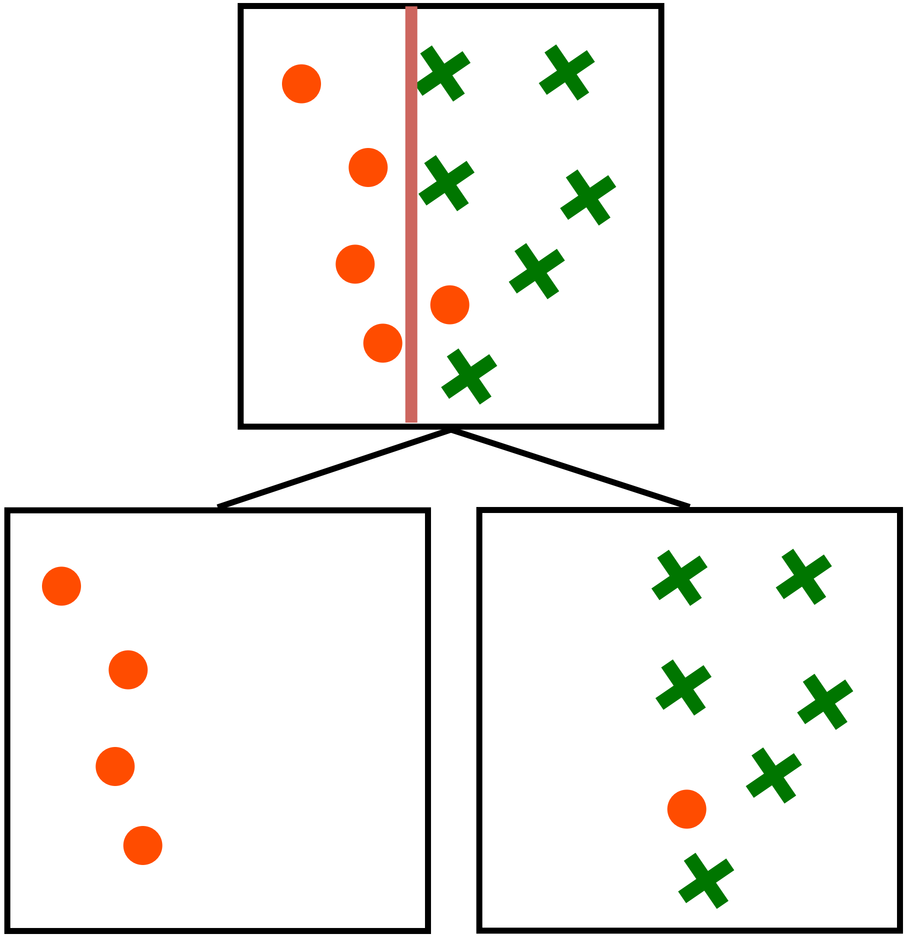
- Two classes (red circles/green crosses)
- Two attributes: X_1 and X_2
- 11 points in training data
- Idea → Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples

Good and Bad Splits

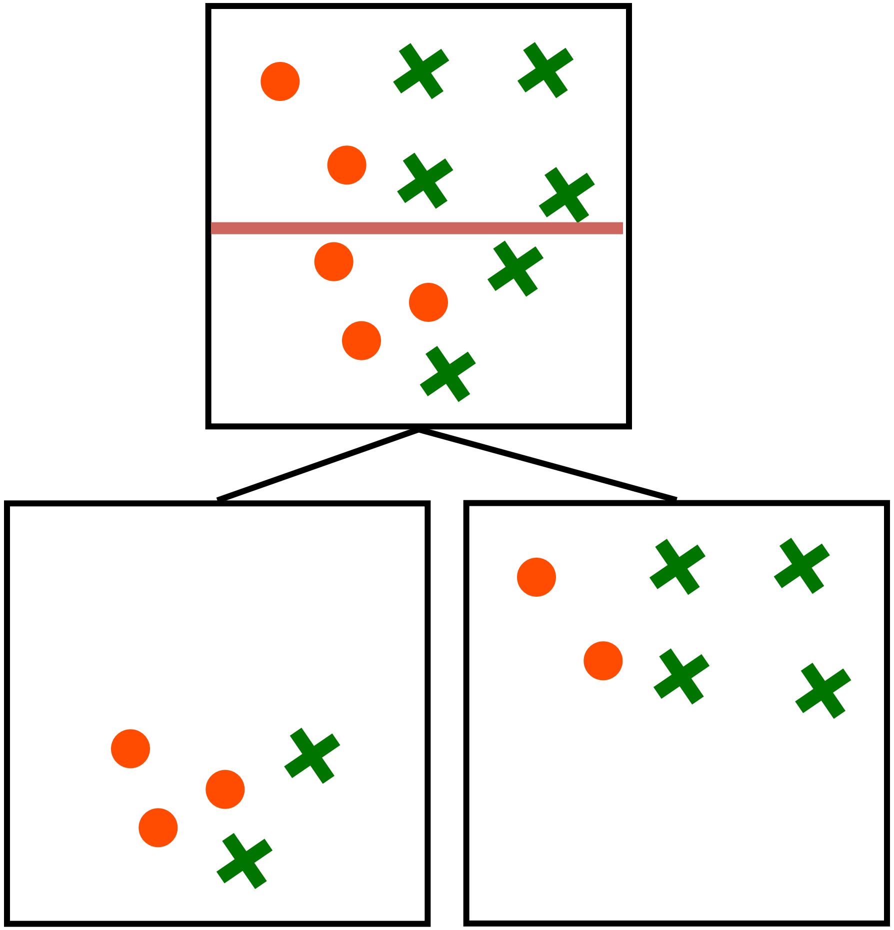


Good

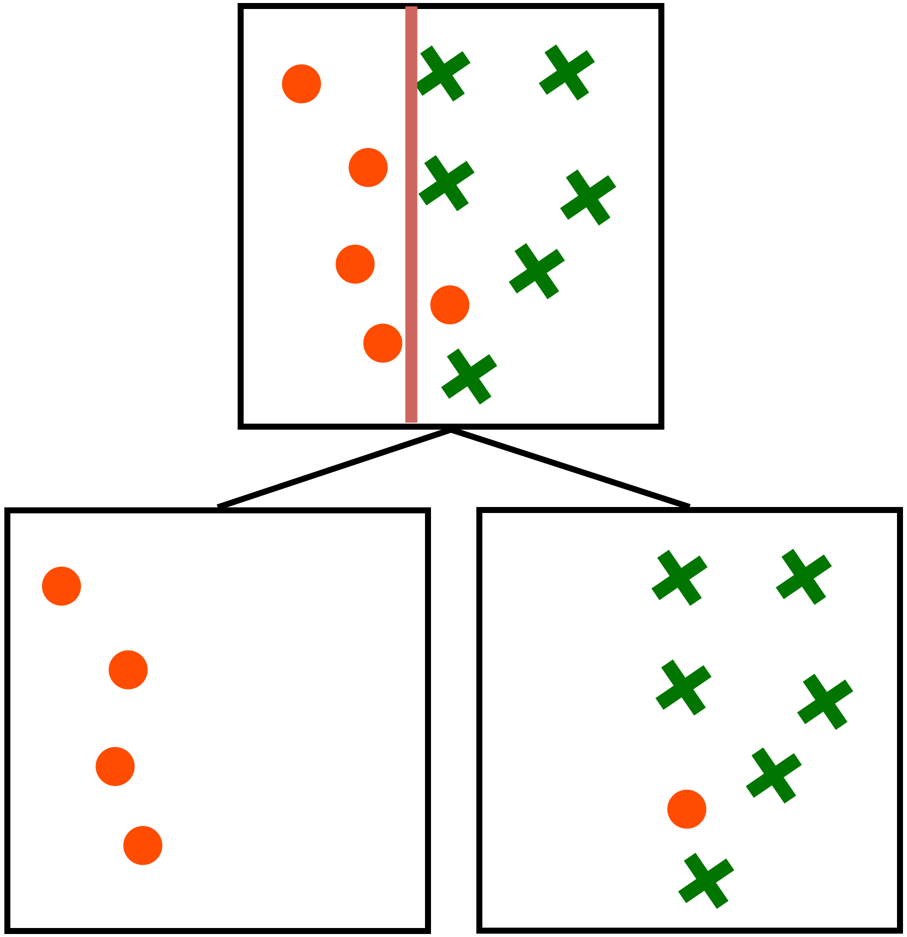
Bad



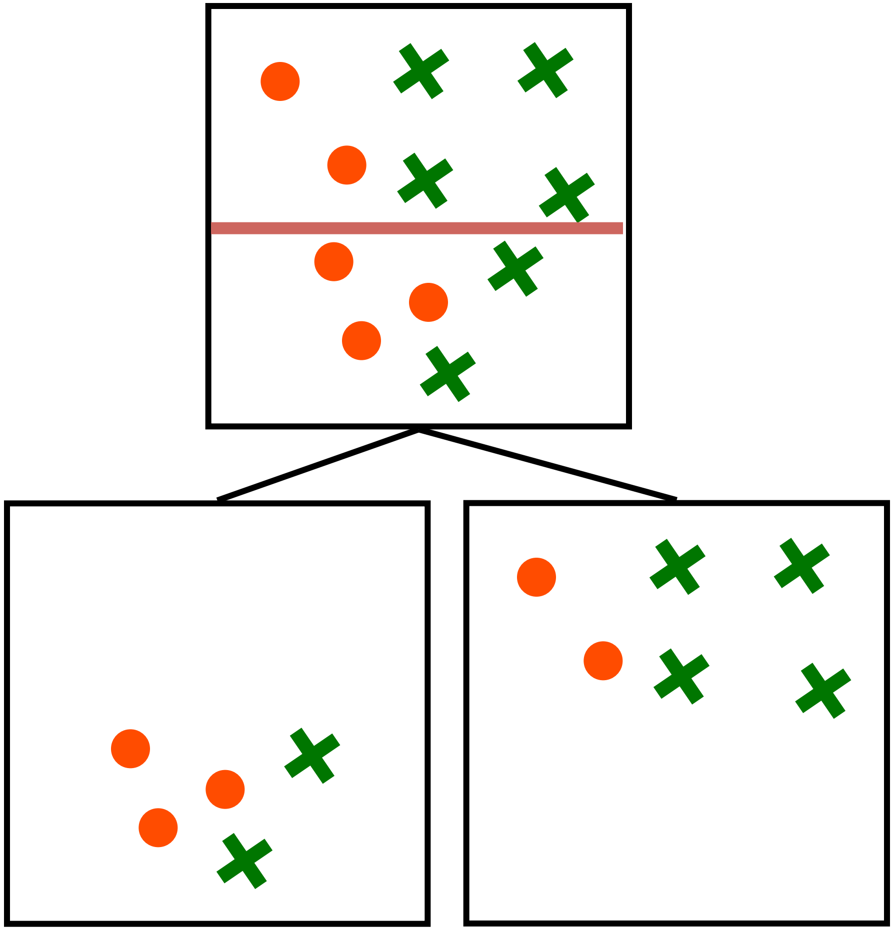
Good



Bad

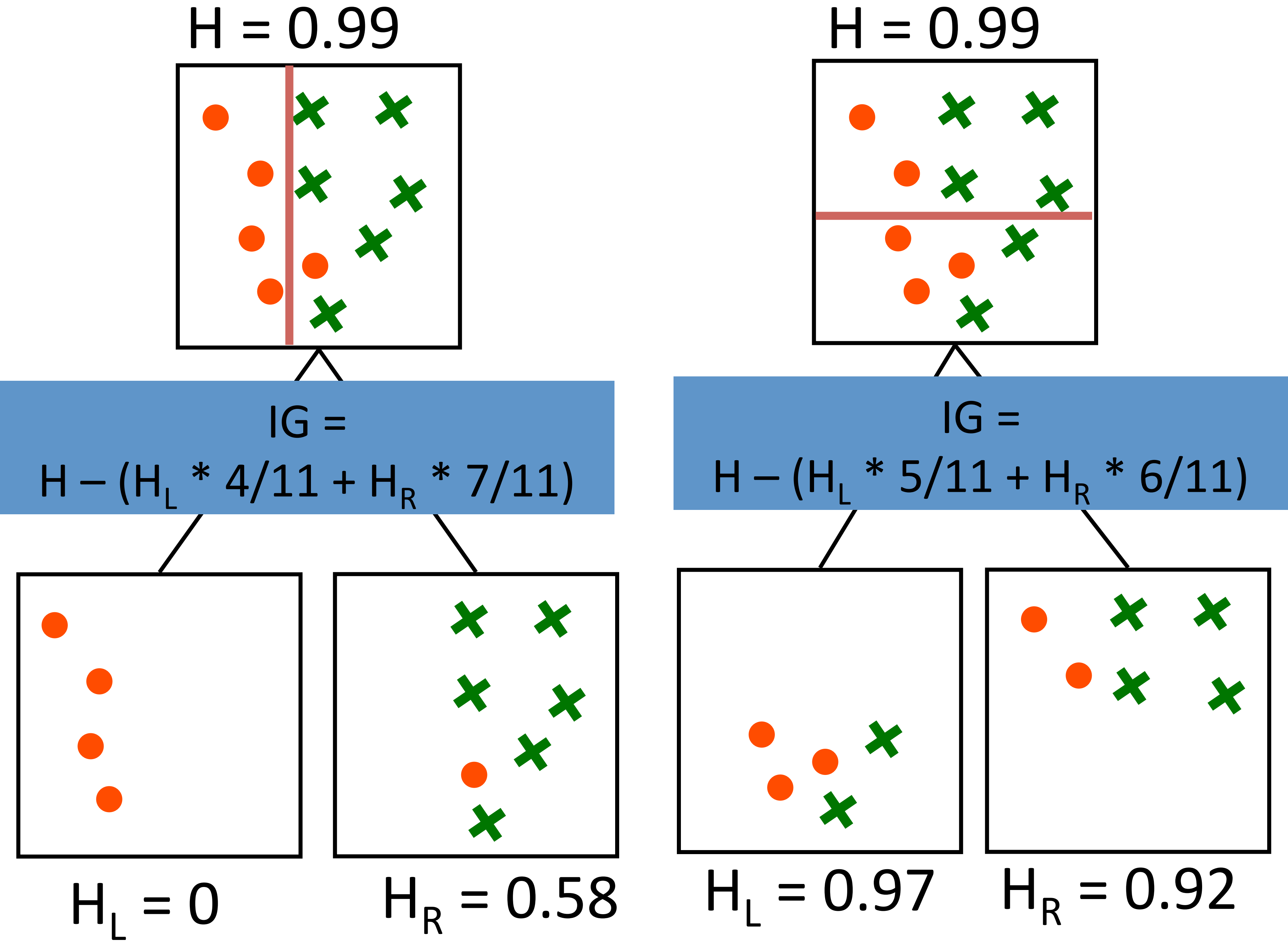


Good

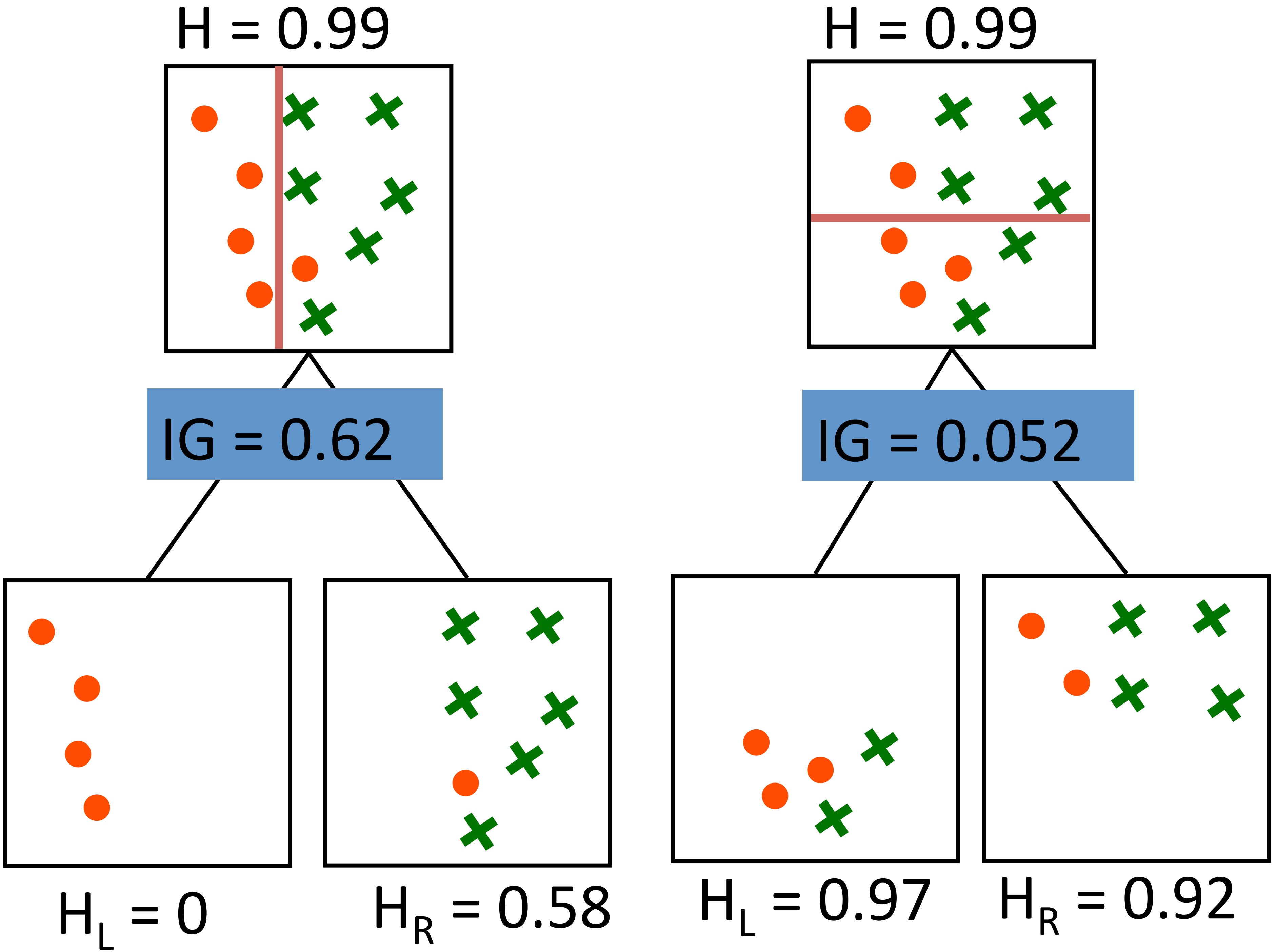


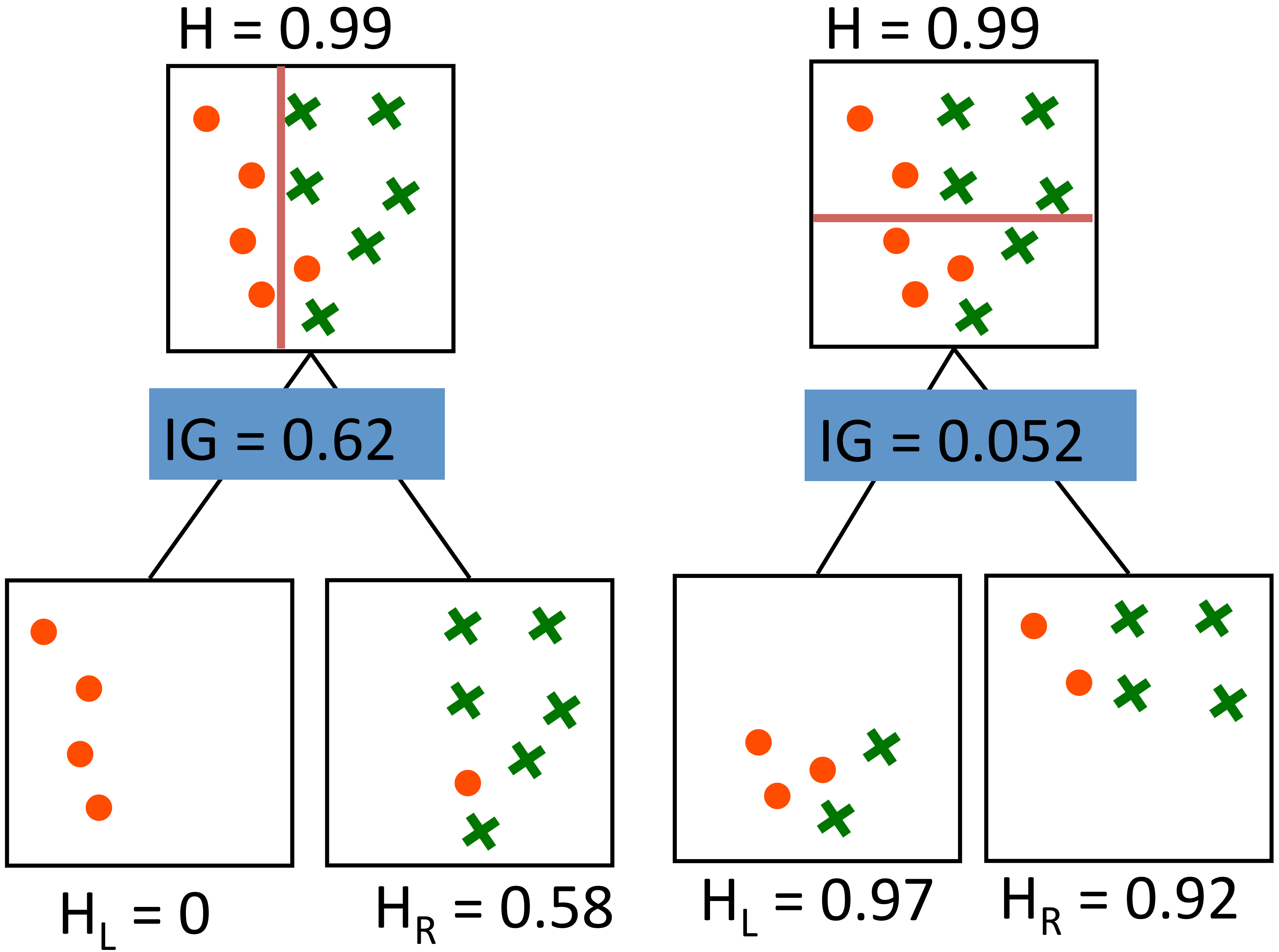
Bad

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

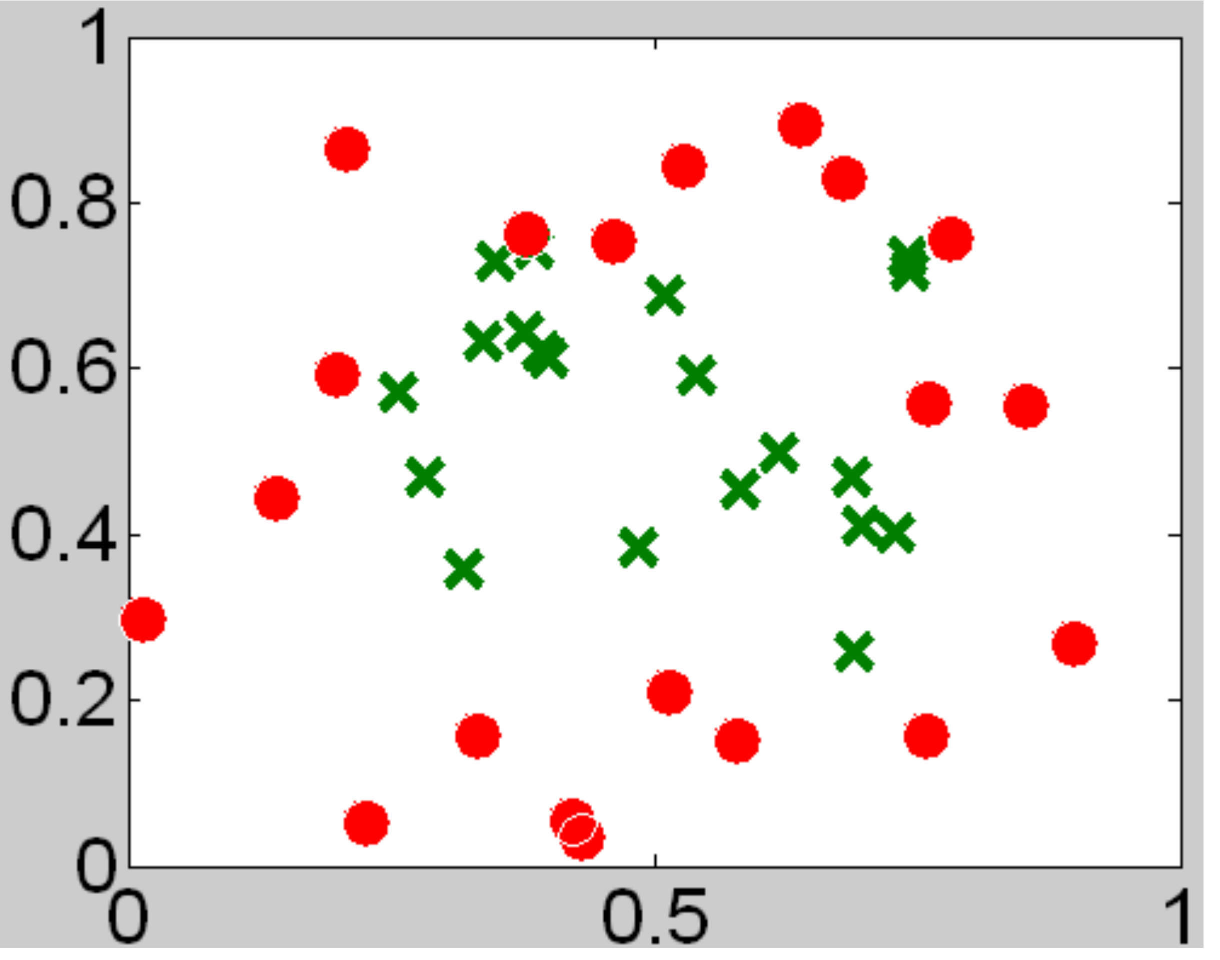


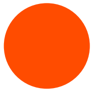
H = 0.99
H = 0.99
IG =
H – (H_L * 4/11 + H_R * 7/11)
IG =
H – (H_L * 5/11 + H_R * 6/11)
H_L = 0
H_R = 0.58
H_L = 0.97
H_R = 0.92


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


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

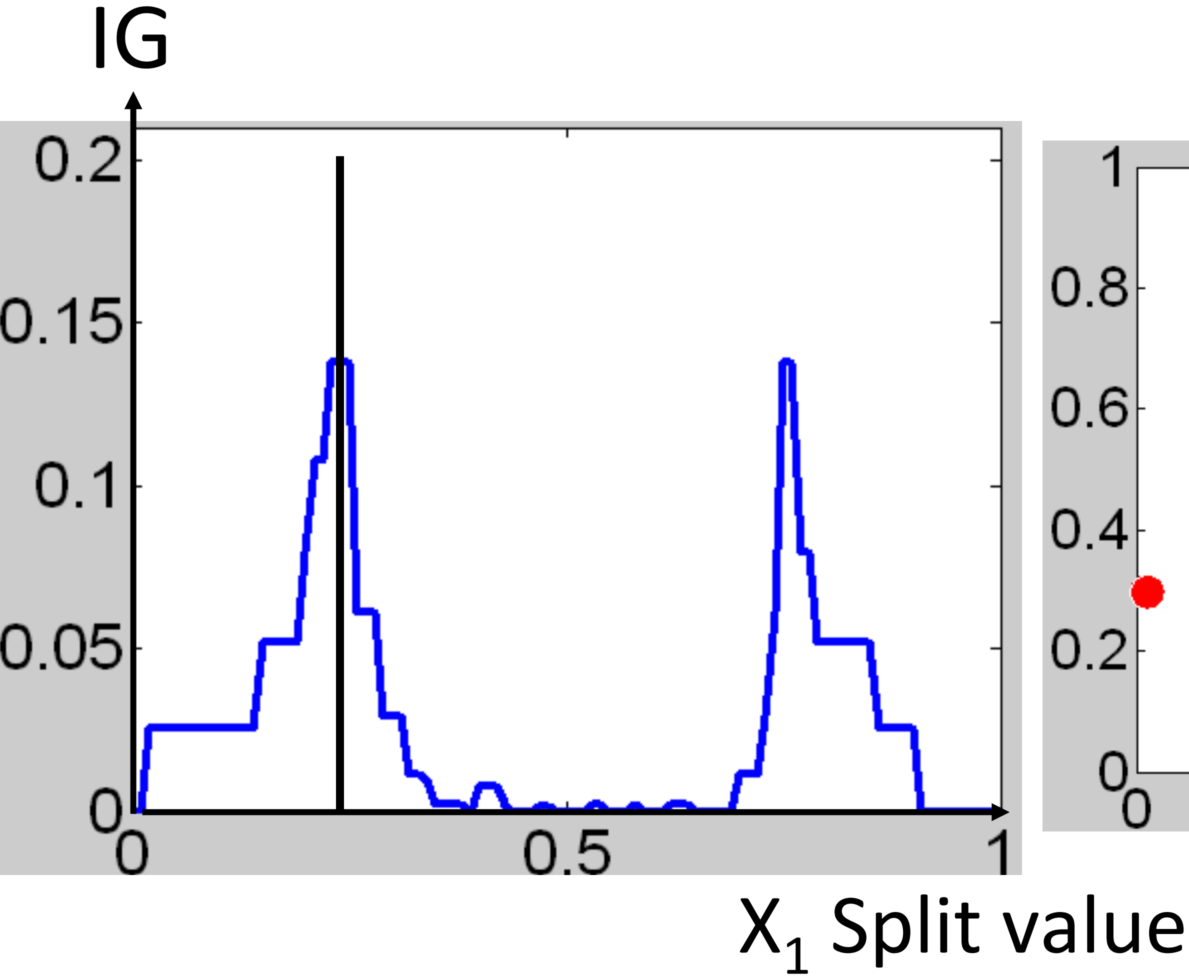
More Complete Example

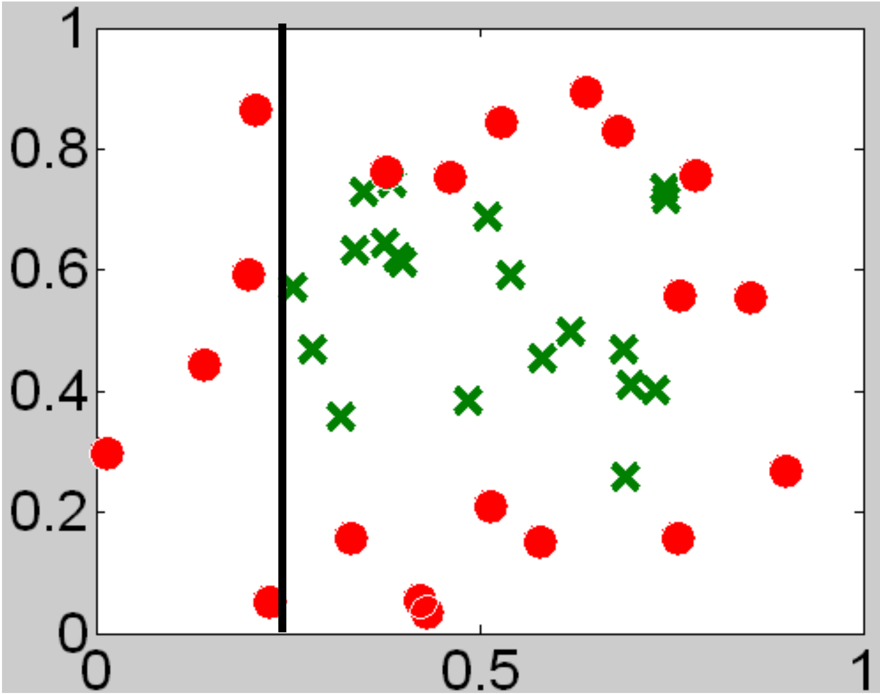


 = 20 training examples from class A

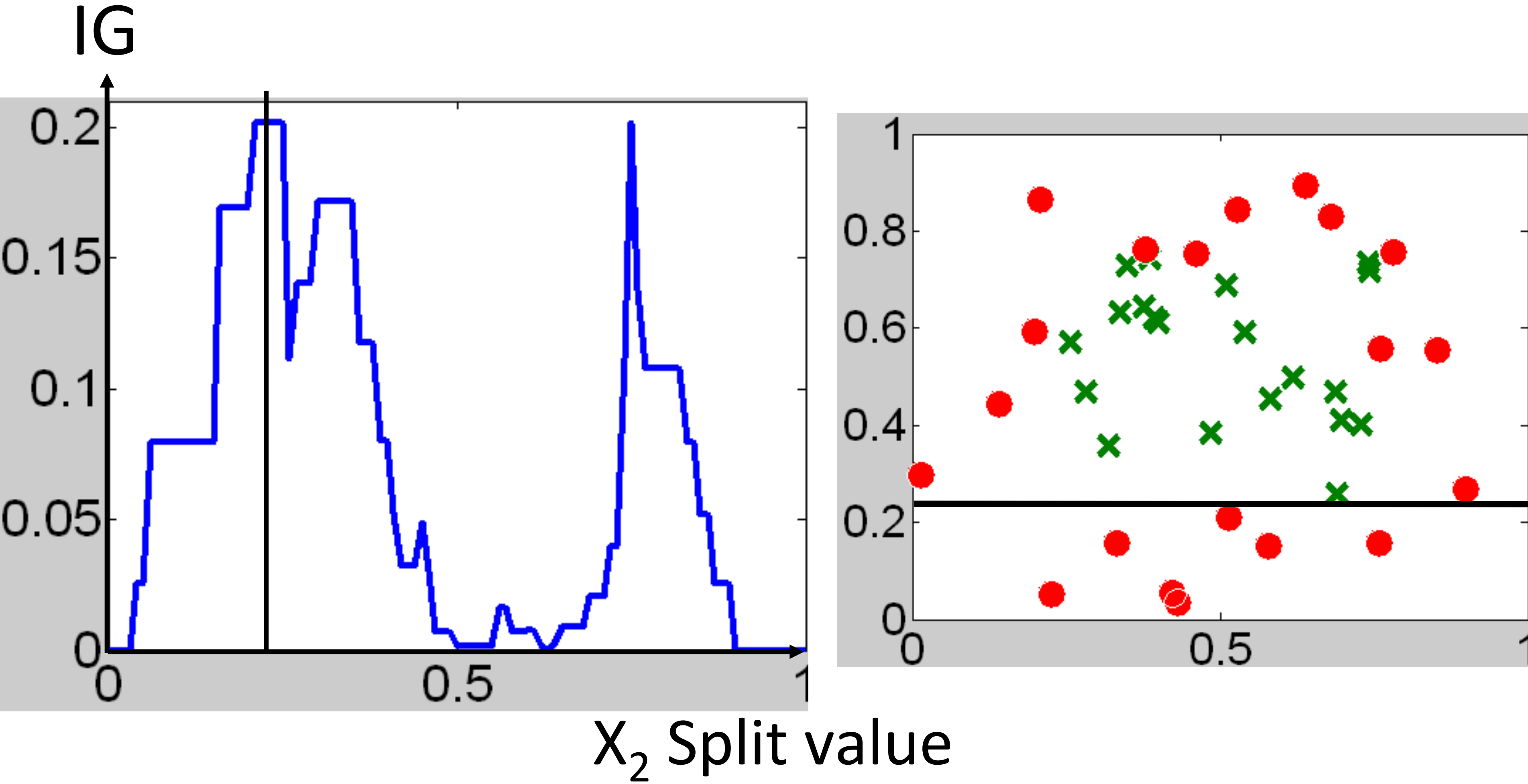
 = 20 training examples from class B

Attributes = X_1 and X_2 coordinates





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

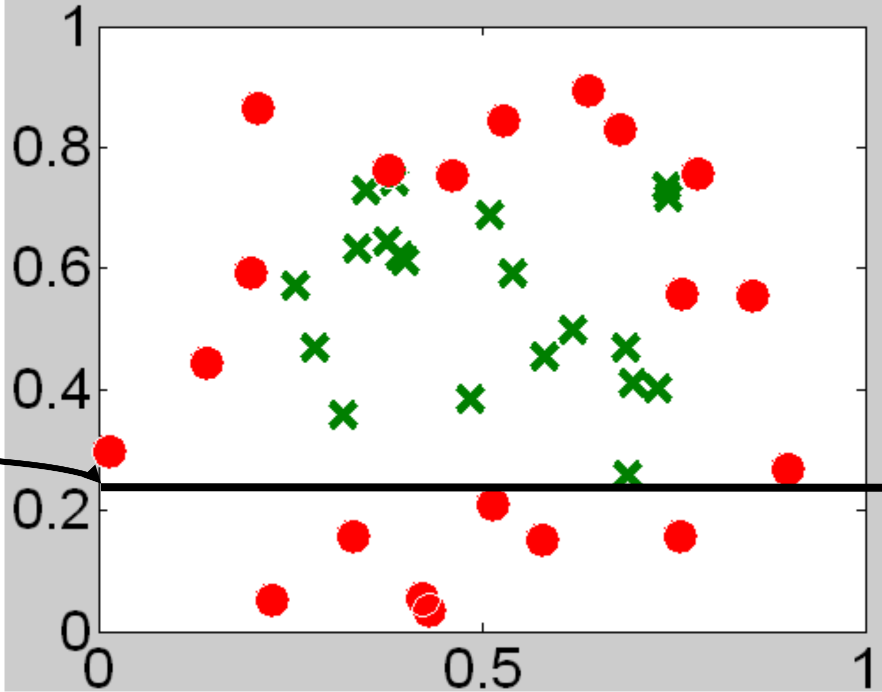


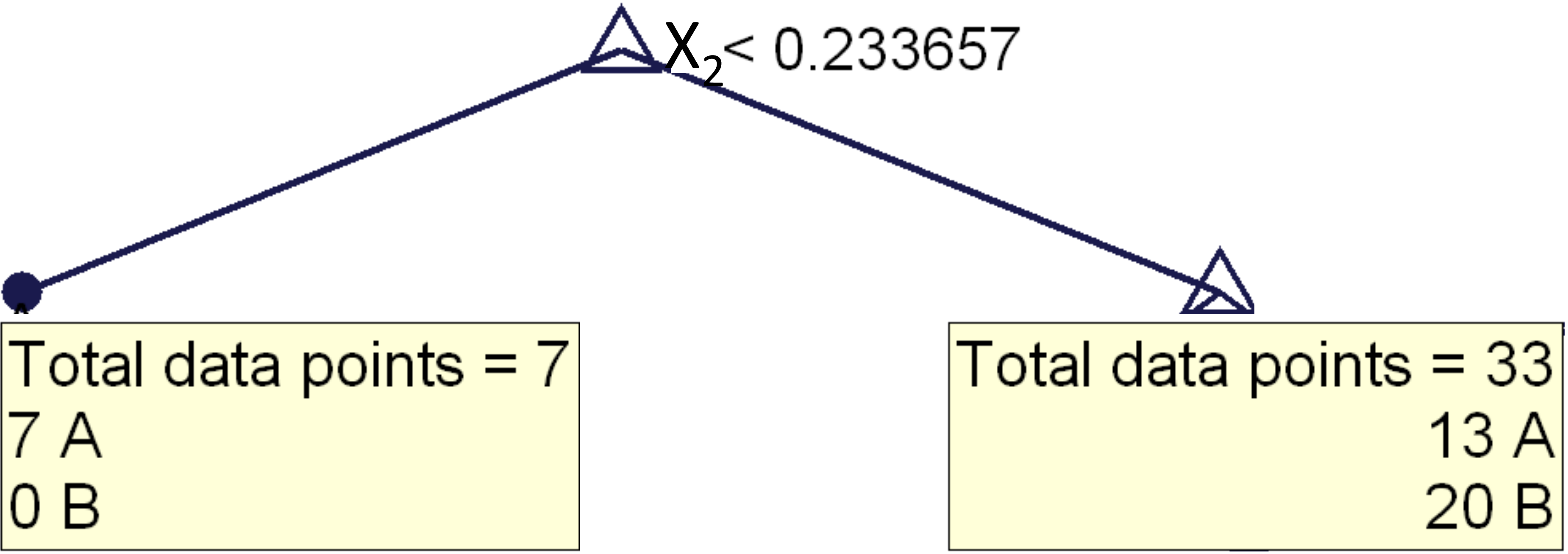
Best split value (max Information Gain) for X_2 attribute: 0.234 with IG = 0.202

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

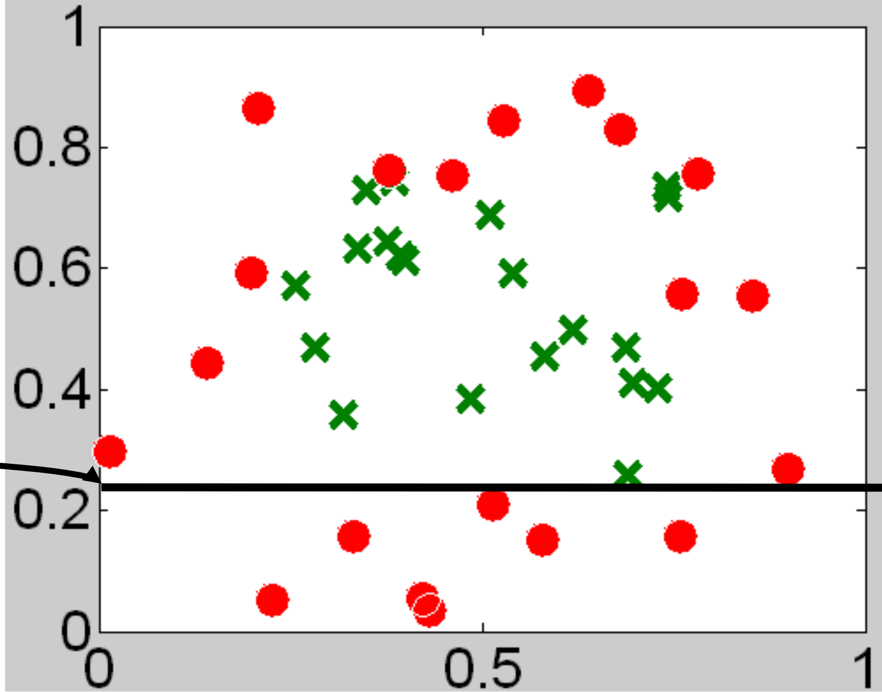


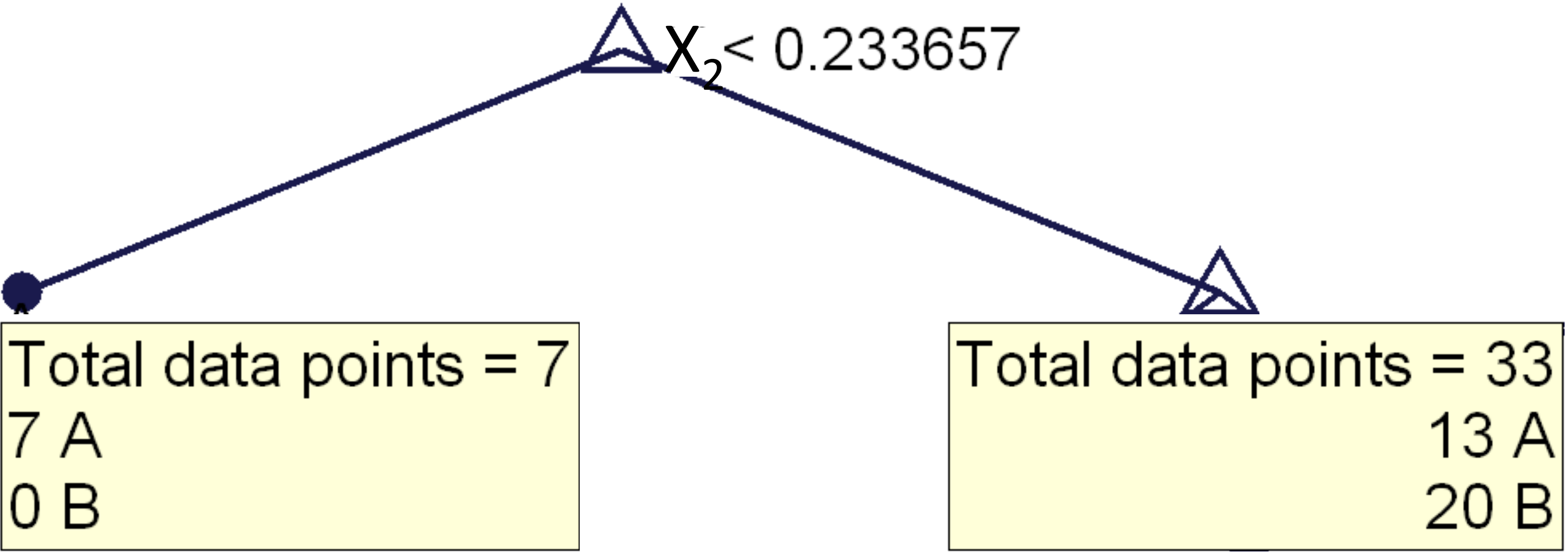


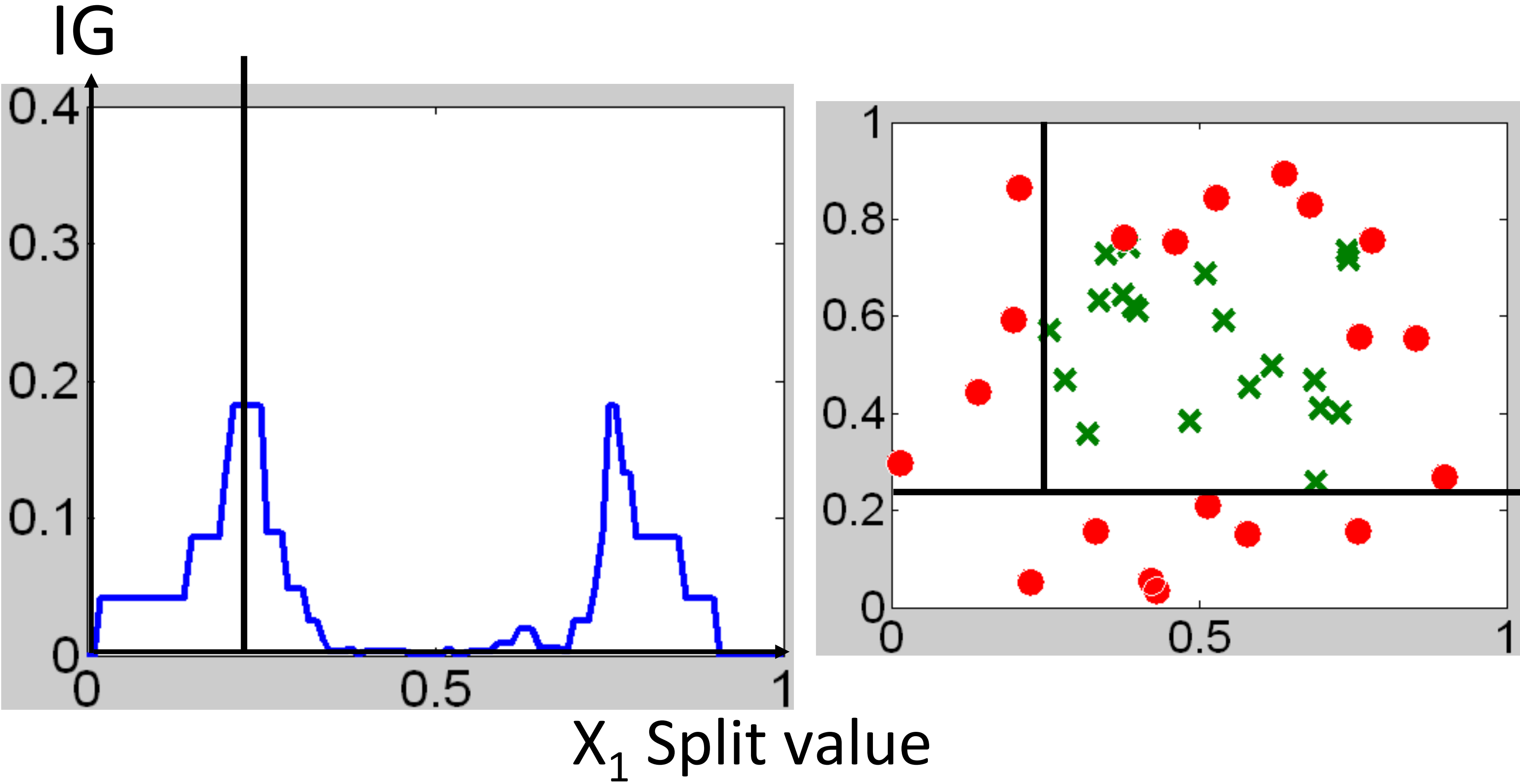
Best X split: 0.24, IG = 0.138

Best Y split: 0.234, IG = 0.202

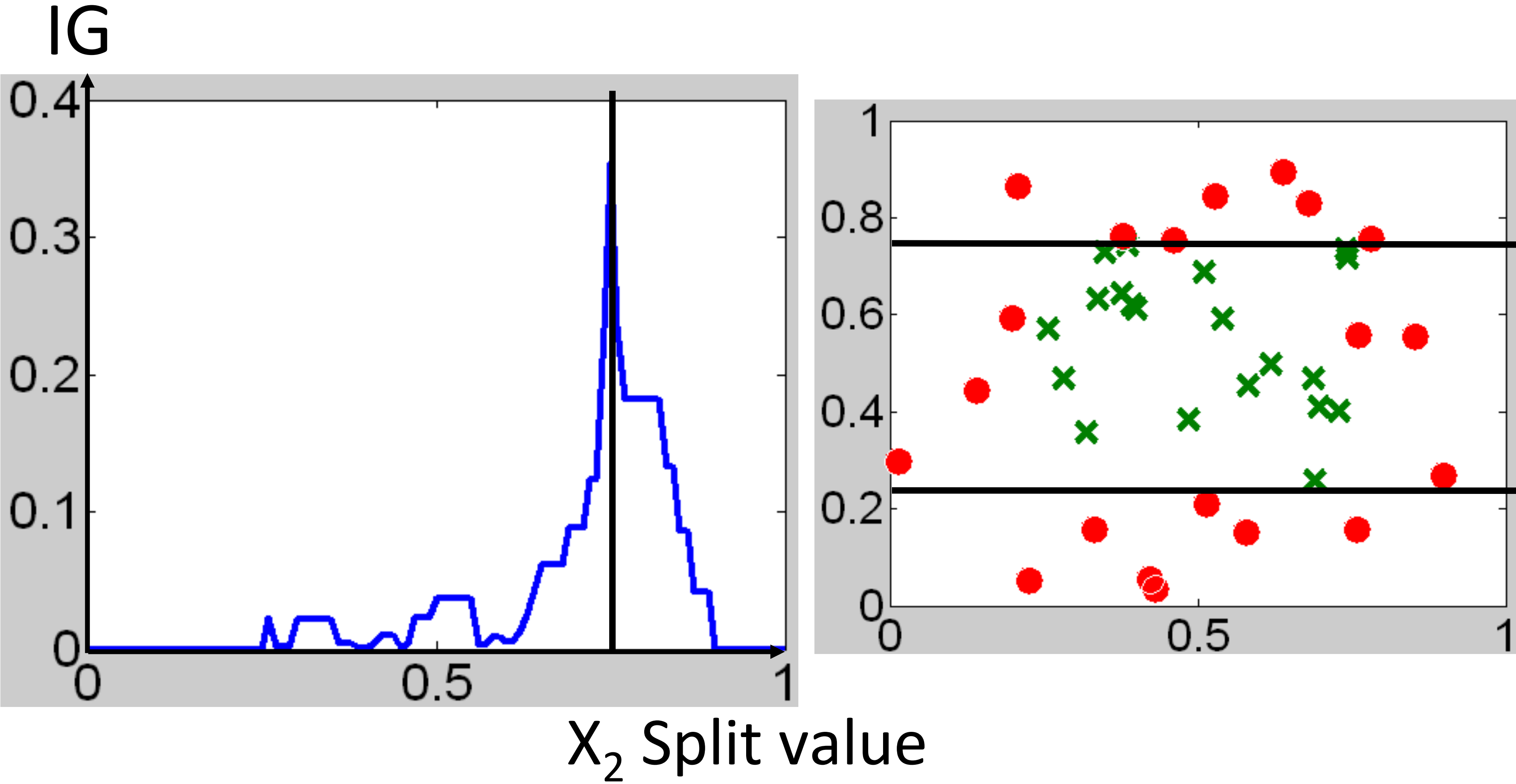
Split on Y with 0.234







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

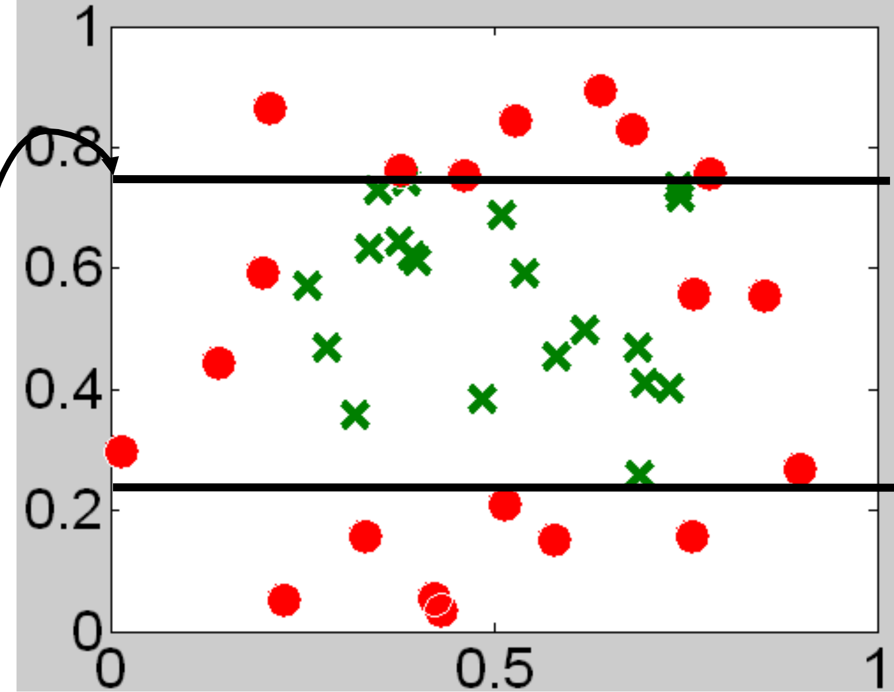


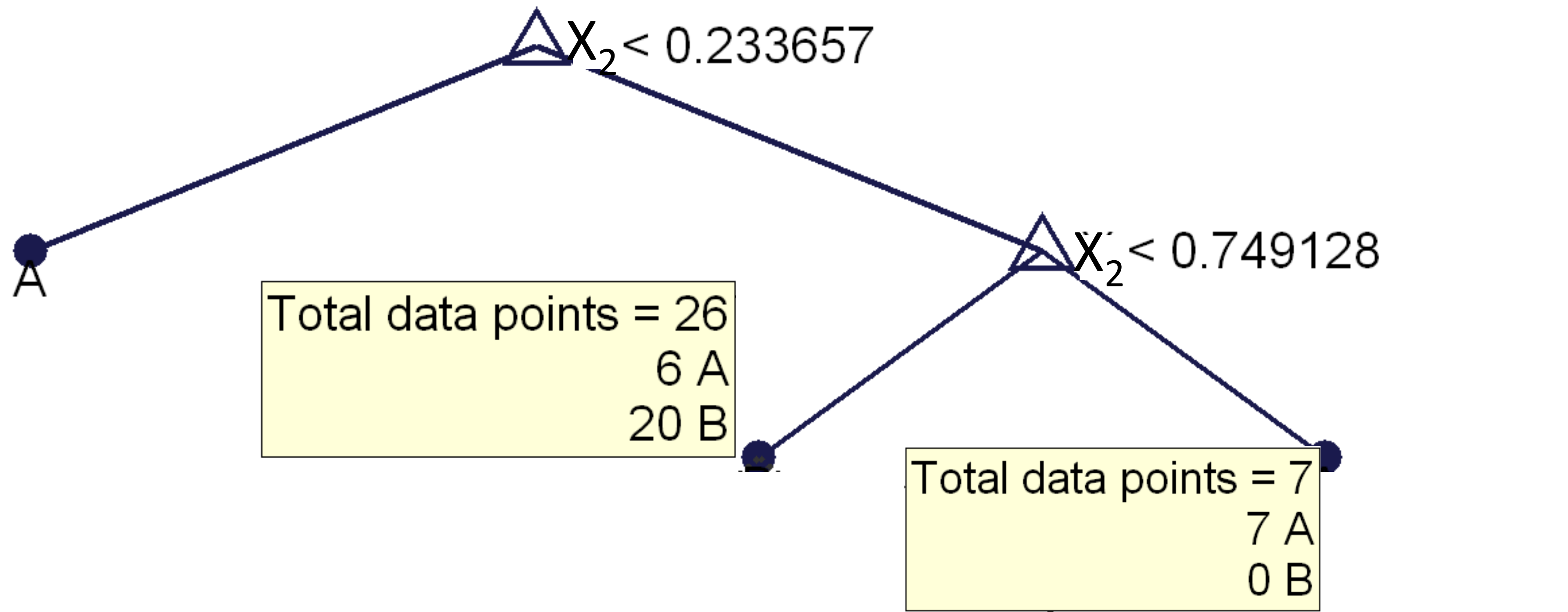
Best split value (max Information Gain) for X_2 attribute: 0.75 with IG ~ 0.353

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

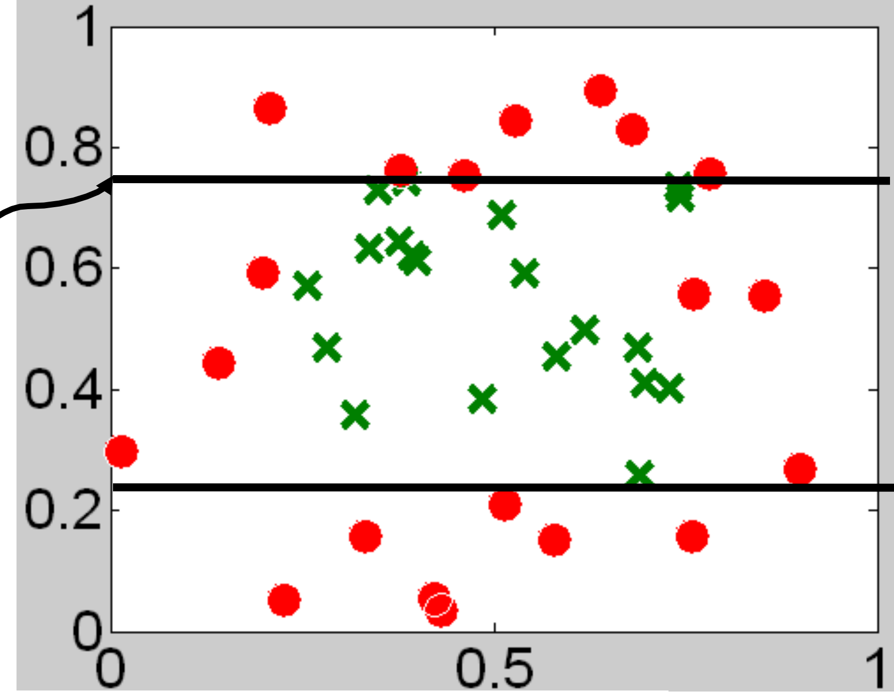


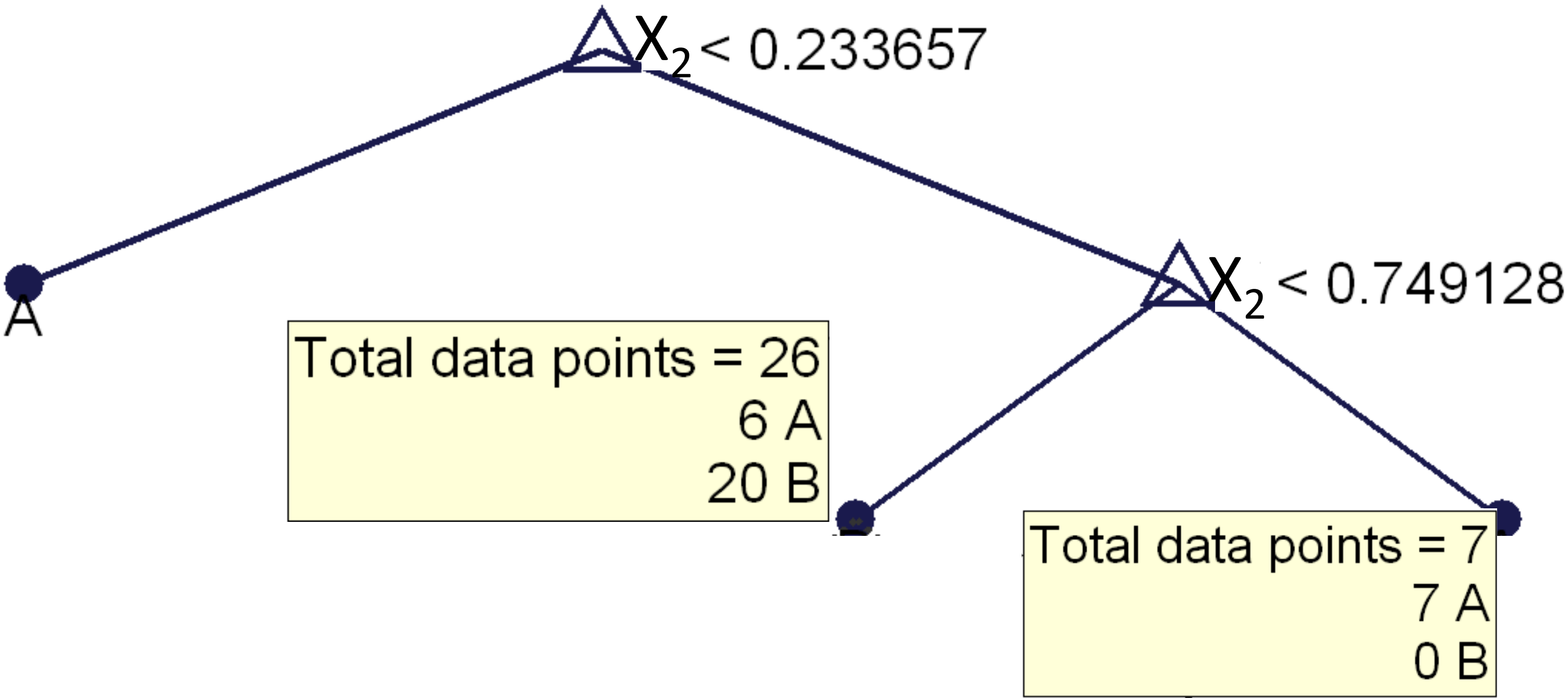


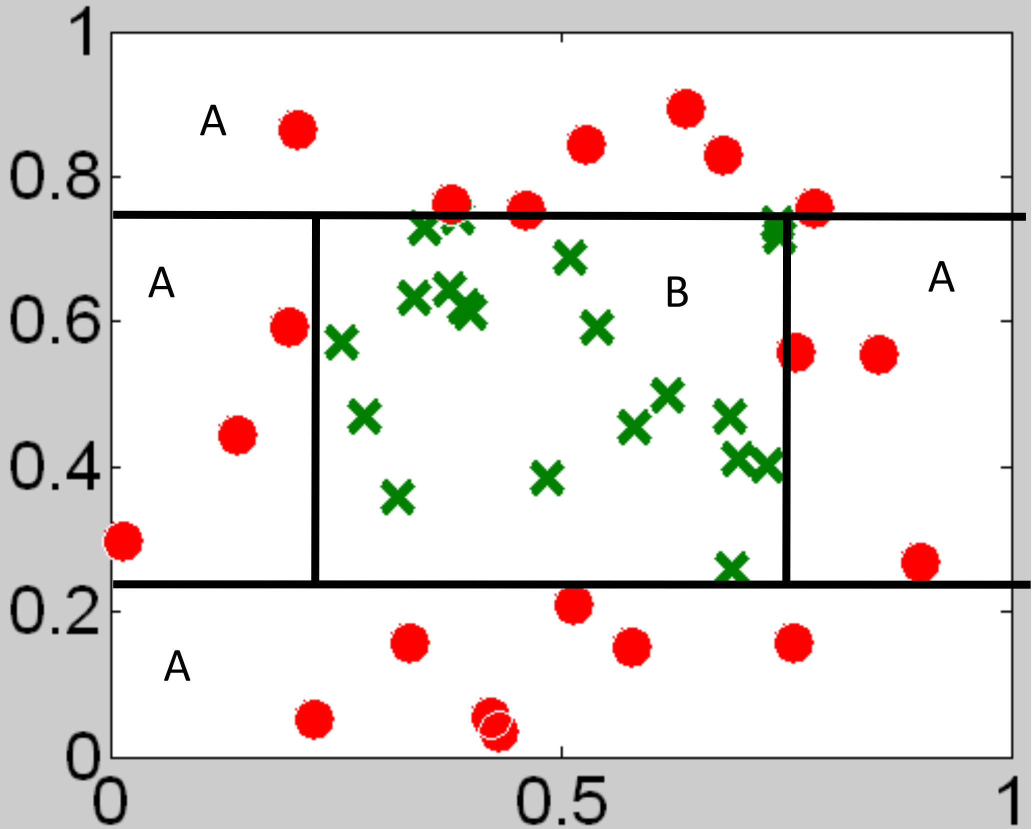
Best X split: 0.22, IG = 0.182

Best Y split: 0.75, IG = 0.353

Split on X with 0.5

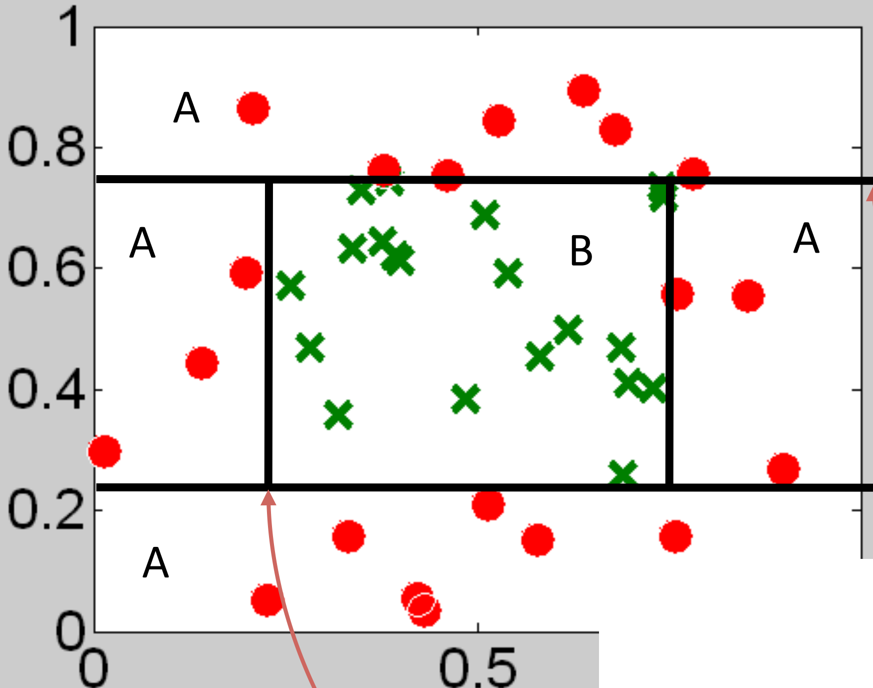


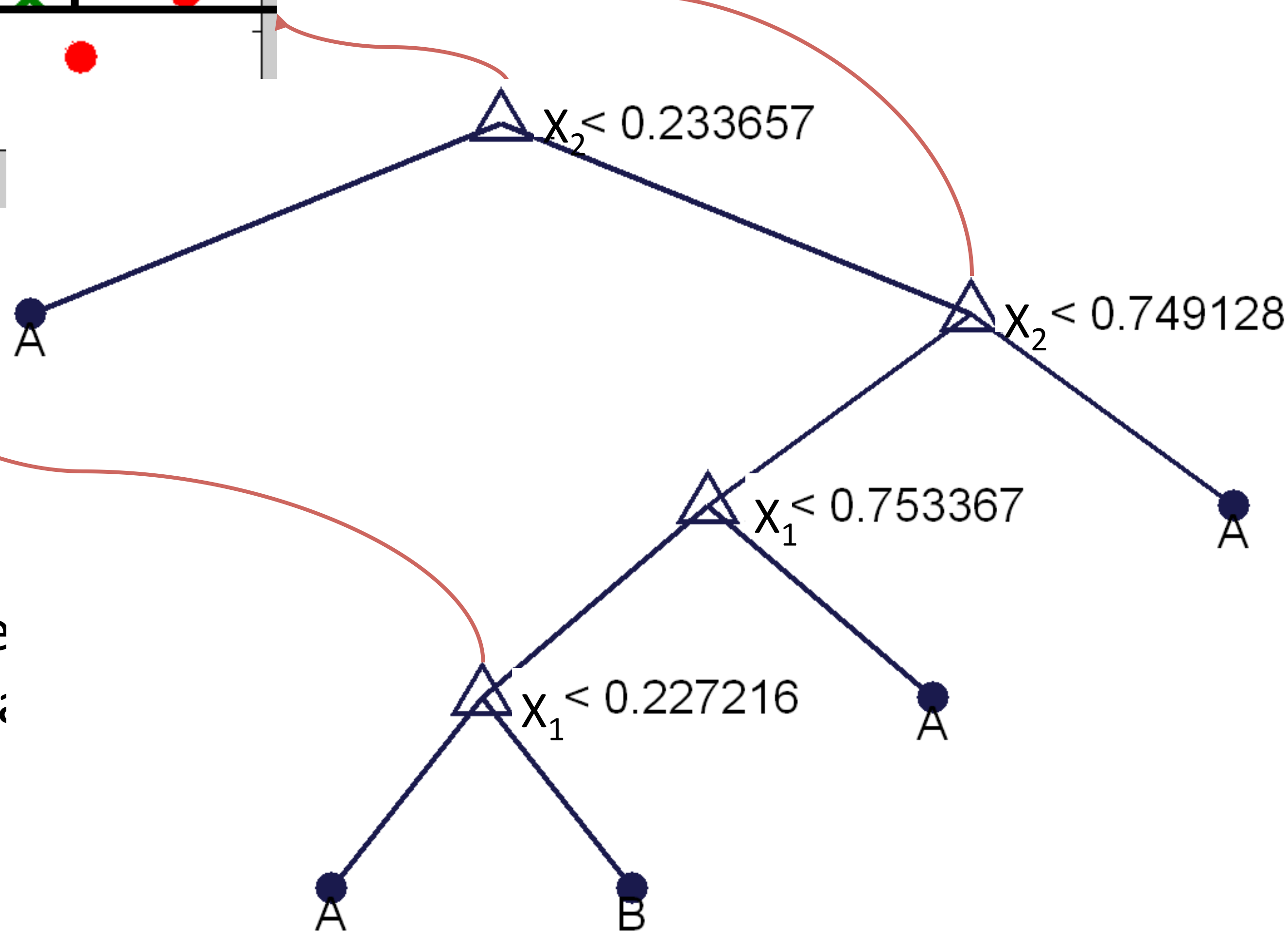




1
0.8
0.6
0.4
0.2
0
0
0.5
1
A
A
B
A
A

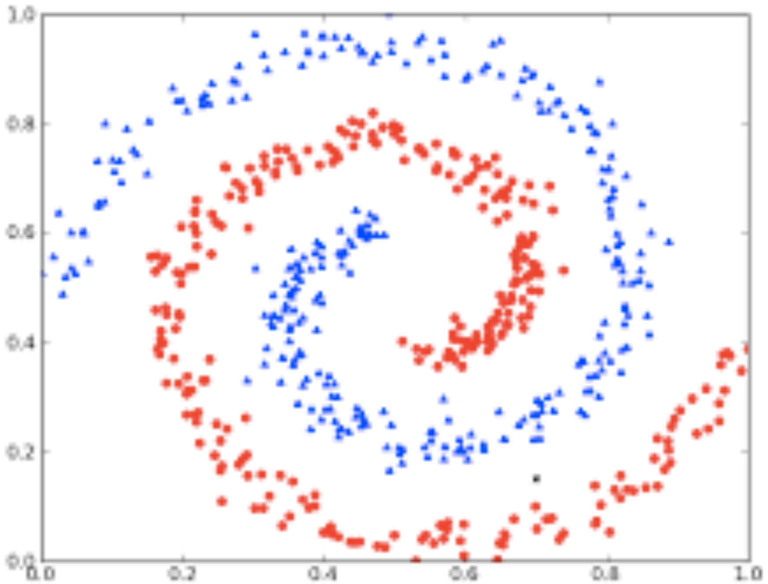
Final decision tree

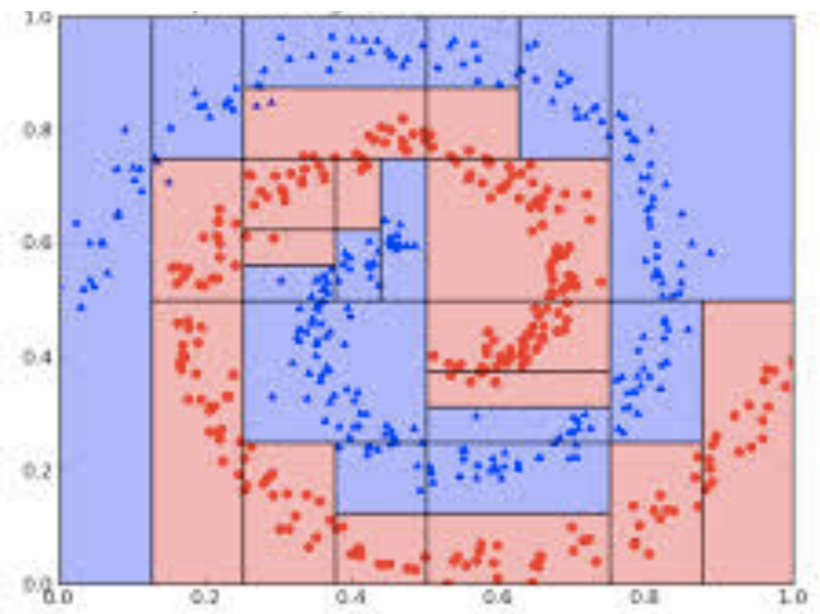




Each of the leaf node is pure → contains data from only one class

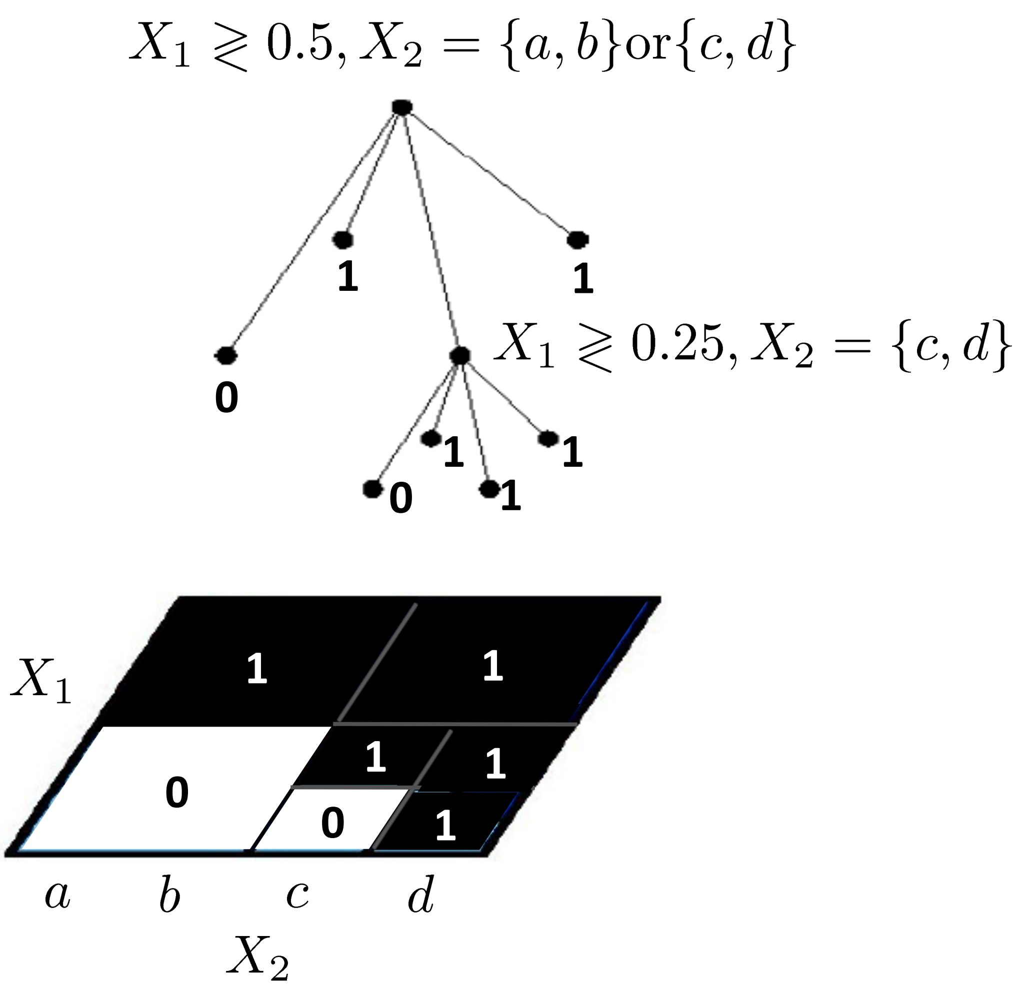
Example of 2-feature decision tree classifier





cs.uchicago.edu

Decision Tree more generally…

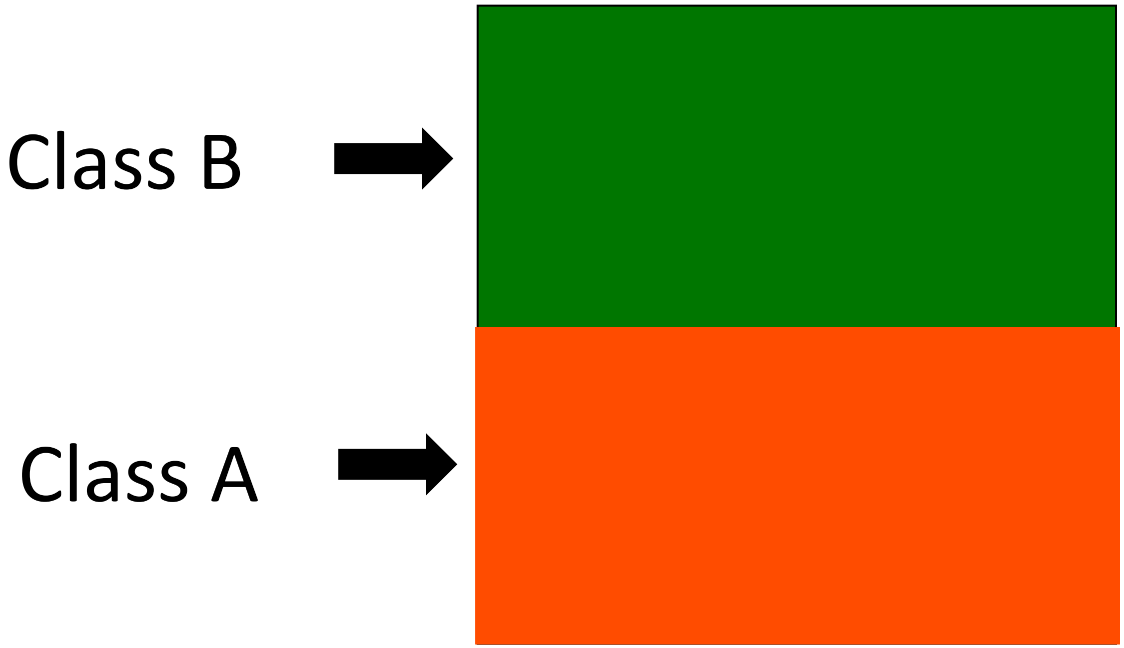


- Features can be discrete, continuous or categorical
- Each internal node: test some set of features $\{X_i\}$
- Each branch from a node: selects a set of value for $\{X_i\}$
- Each leaf node: prediction for Y

When to Stop?

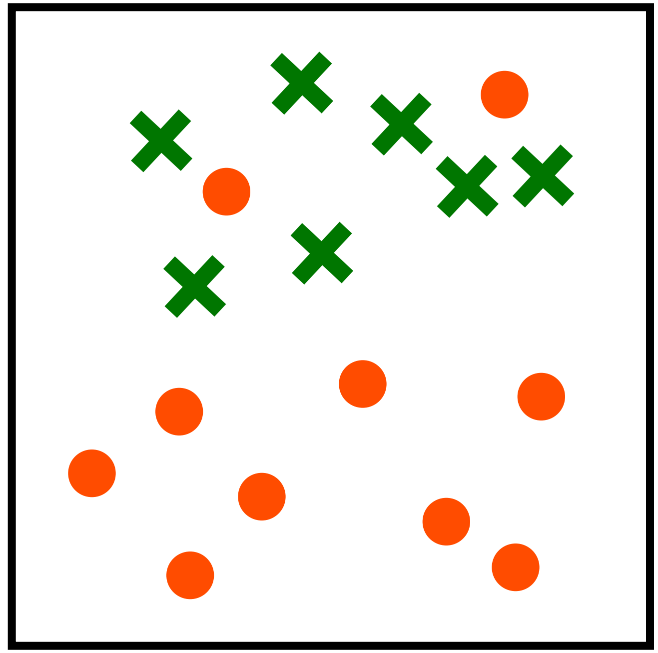
- Many strategies for picking simpler trees:
  - Pre-pruning
    - Fixed depth (e.g. ID3)
    - Fixed number of leaves
  - Use testing

The Overfitting Problem: Example

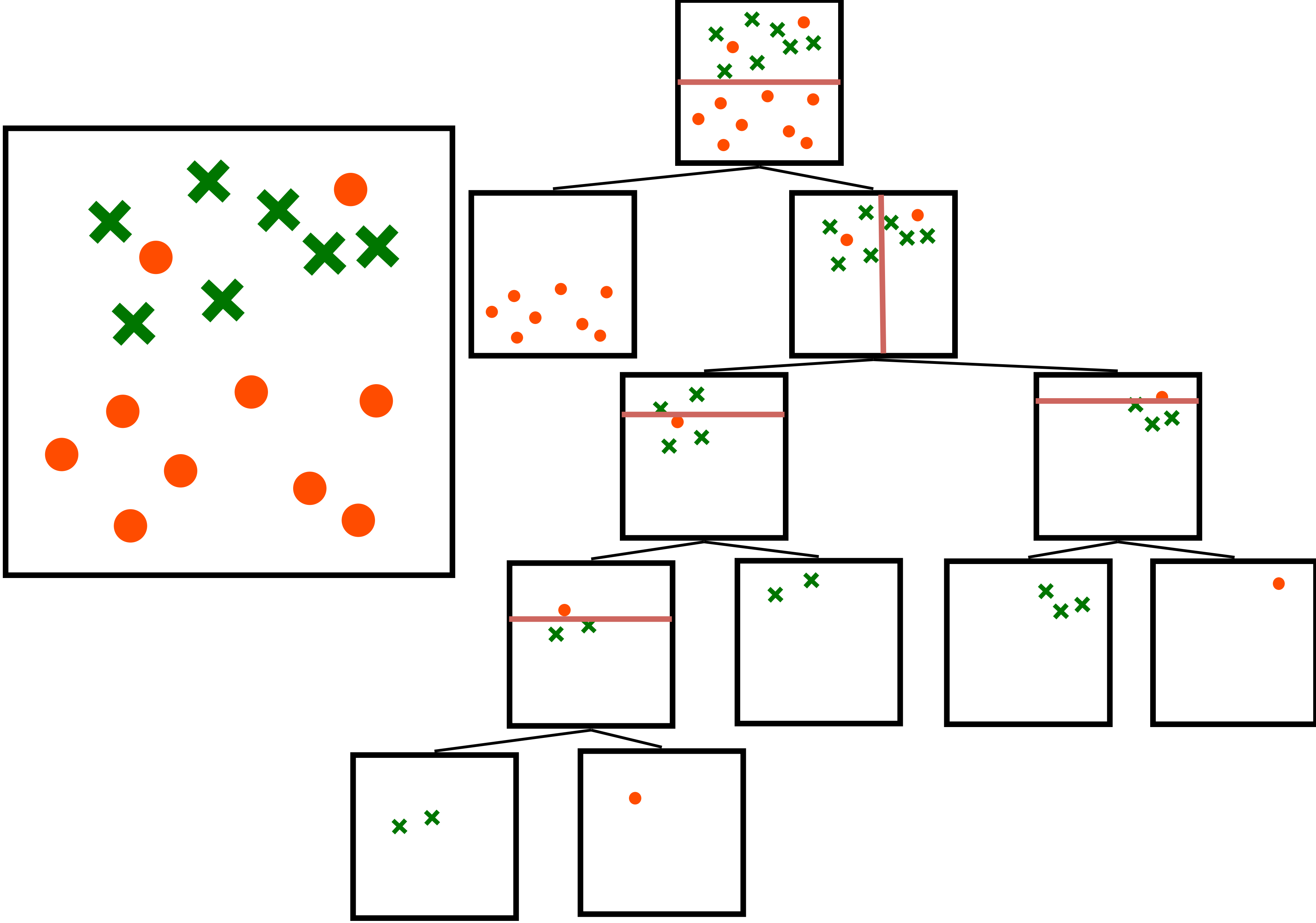


- Suppose that, in an ideal world, class B is everything such that X_2 >= 0.5 and class A is everything with X_2 < 0.5
- Note that attribute X_1 is irrelevant
- Seems like generating a decision tree would be trivial

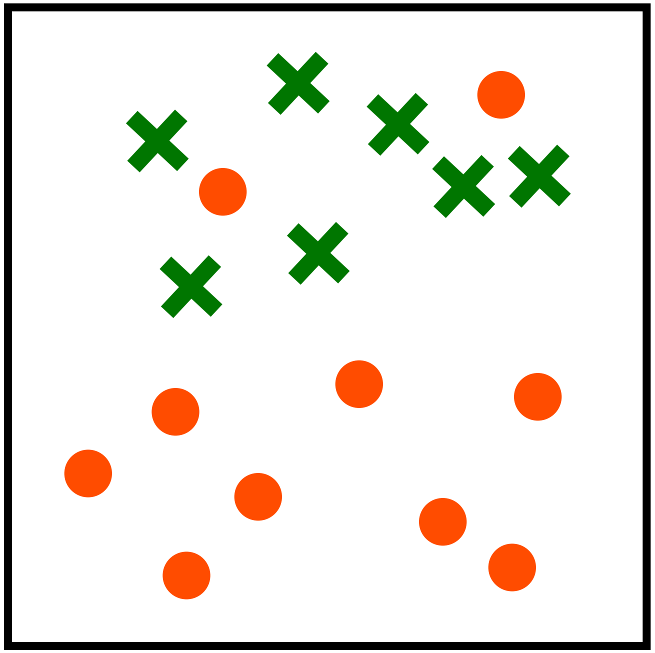
The Overfitting Problem: Example

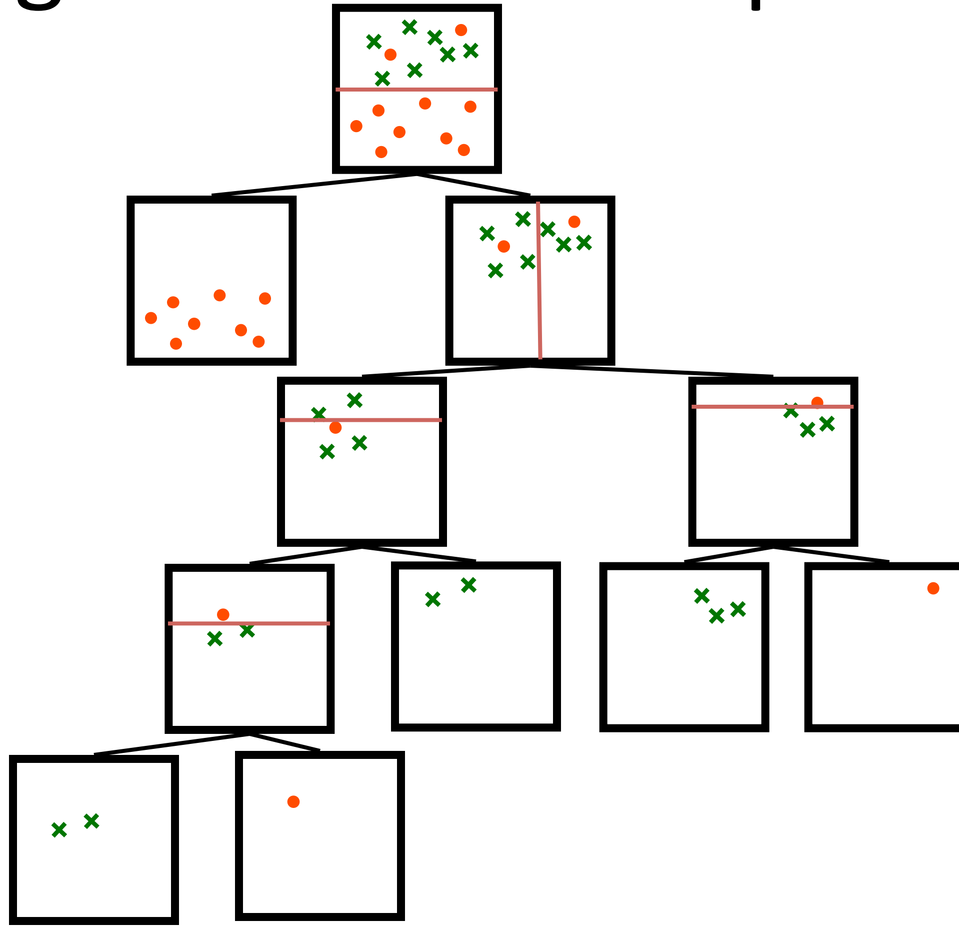


- However, we collect training examples from the perfect world through some imperfect observation device
- As a result, the training data is corrupted by *noise*.



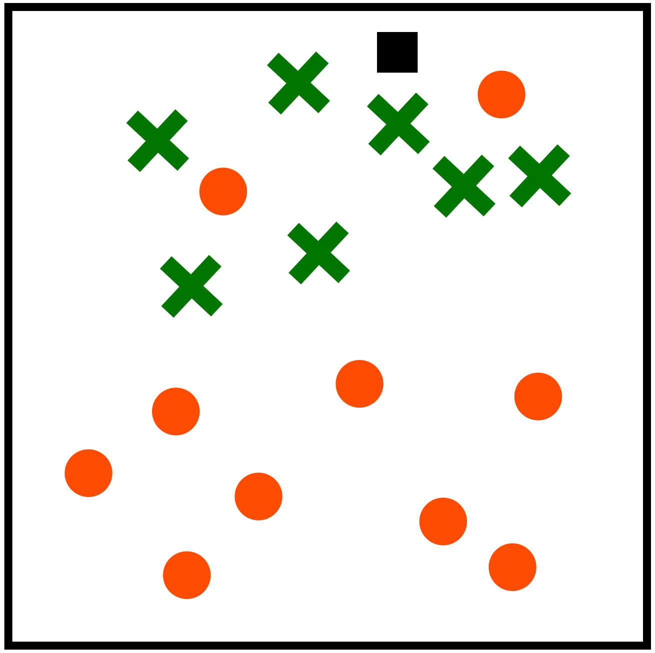
The Overfitting Problem: Example

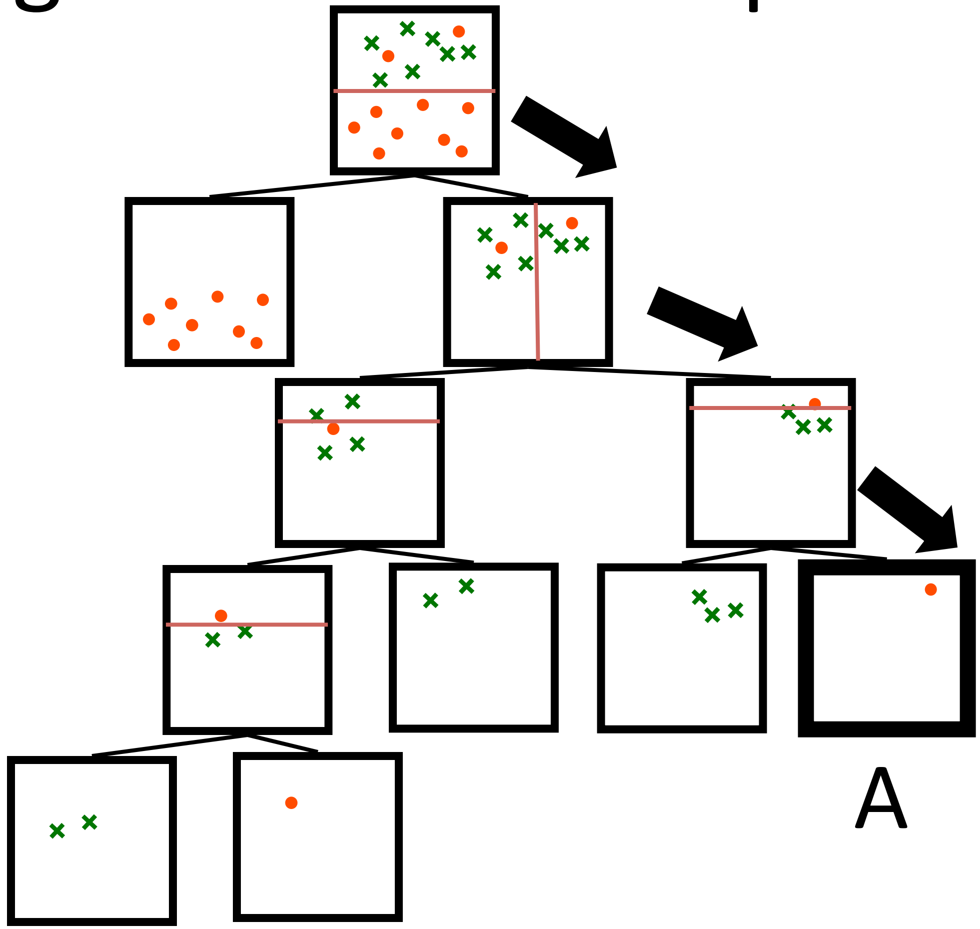




- The resulting decision tree is far more “complicated” than it should be, if the learning algorithm tries to classify *all of the training set perfectly* → *overfitting*

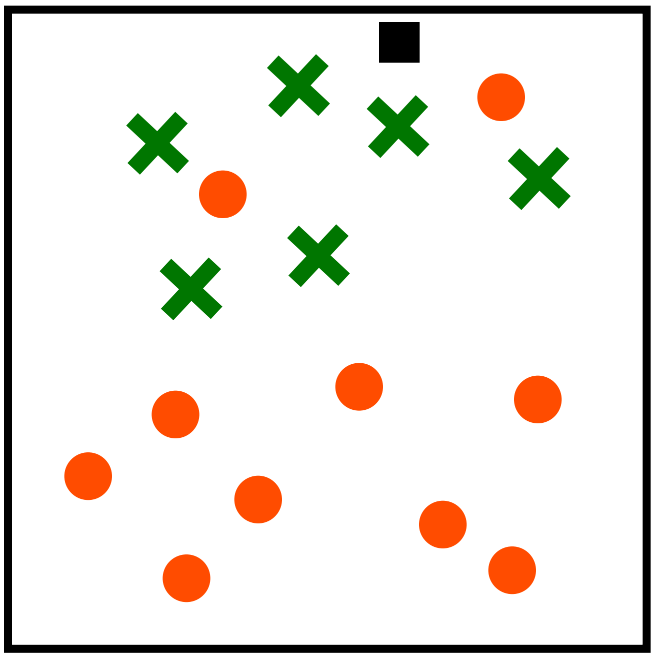
The Overfitting Problem: Example

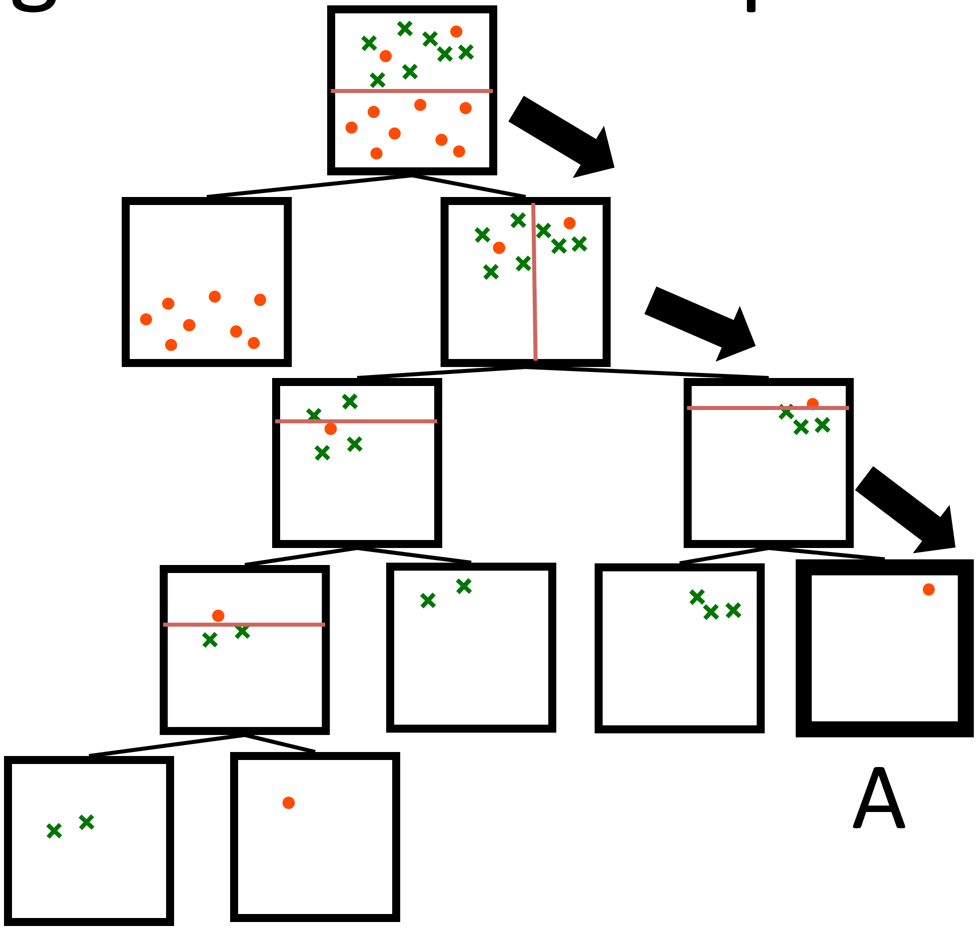




- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

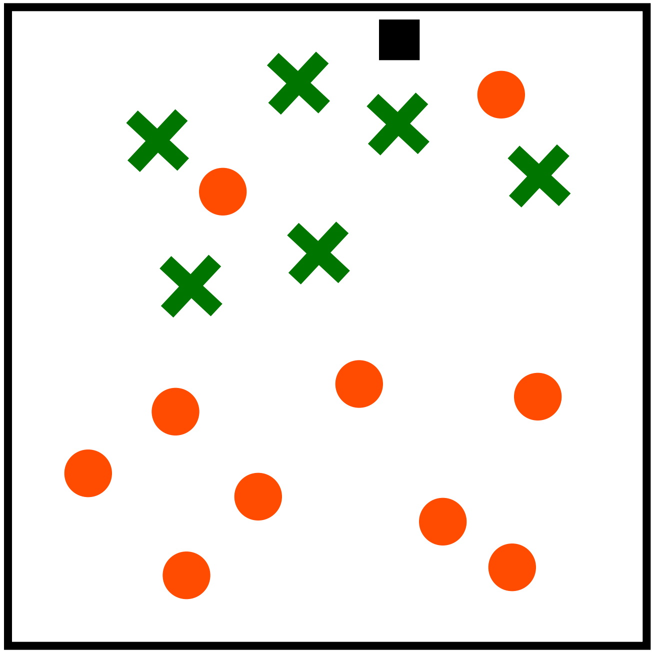
The Overfitting Problem: Example

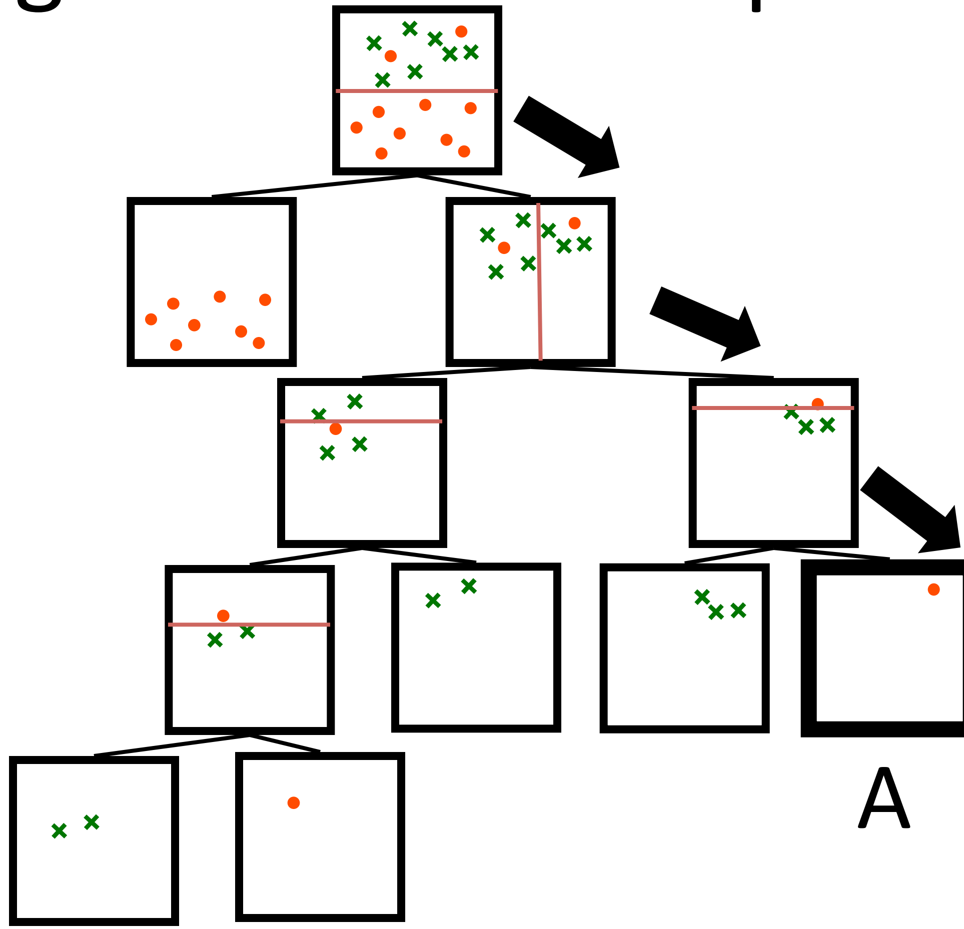




- Effect of overfitting: the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as 'A'

The Overfitting Problem: Example



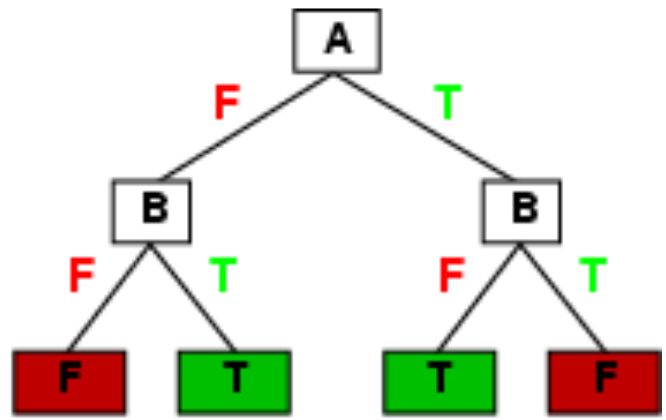


- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

Expressiveness of Decision Trees

- Decision trees in general (without pruning) can express any function of the input features.
- E.g., for Boolean functions, truth table row → path to leaf:

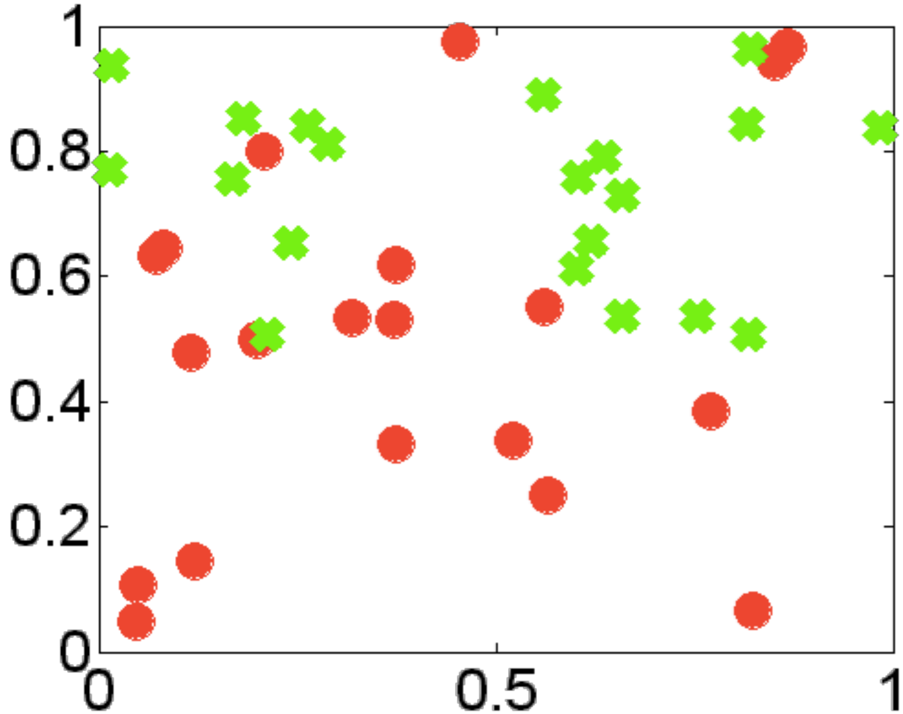
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



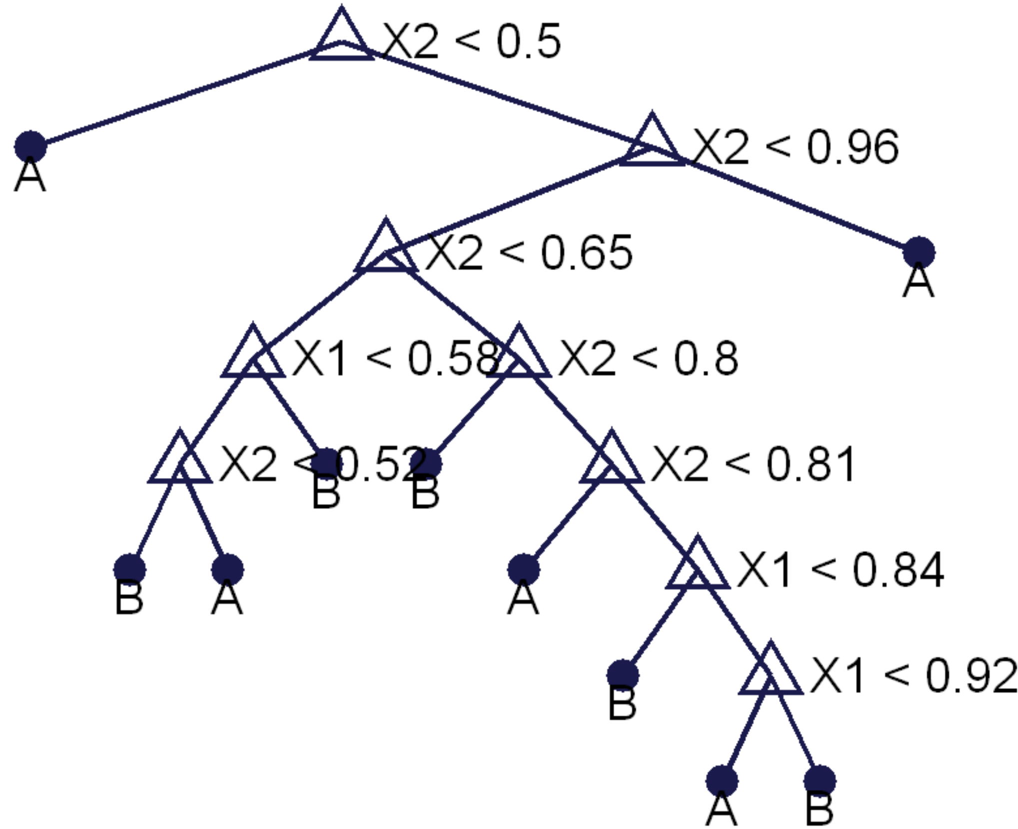
- There is a decision tree which perfectly classifies a training set with one path to leaf for each example - overfitting
- But it won't generalize well to new examples - prefer to find more compact decision trees

Possible Overfitting Solutions

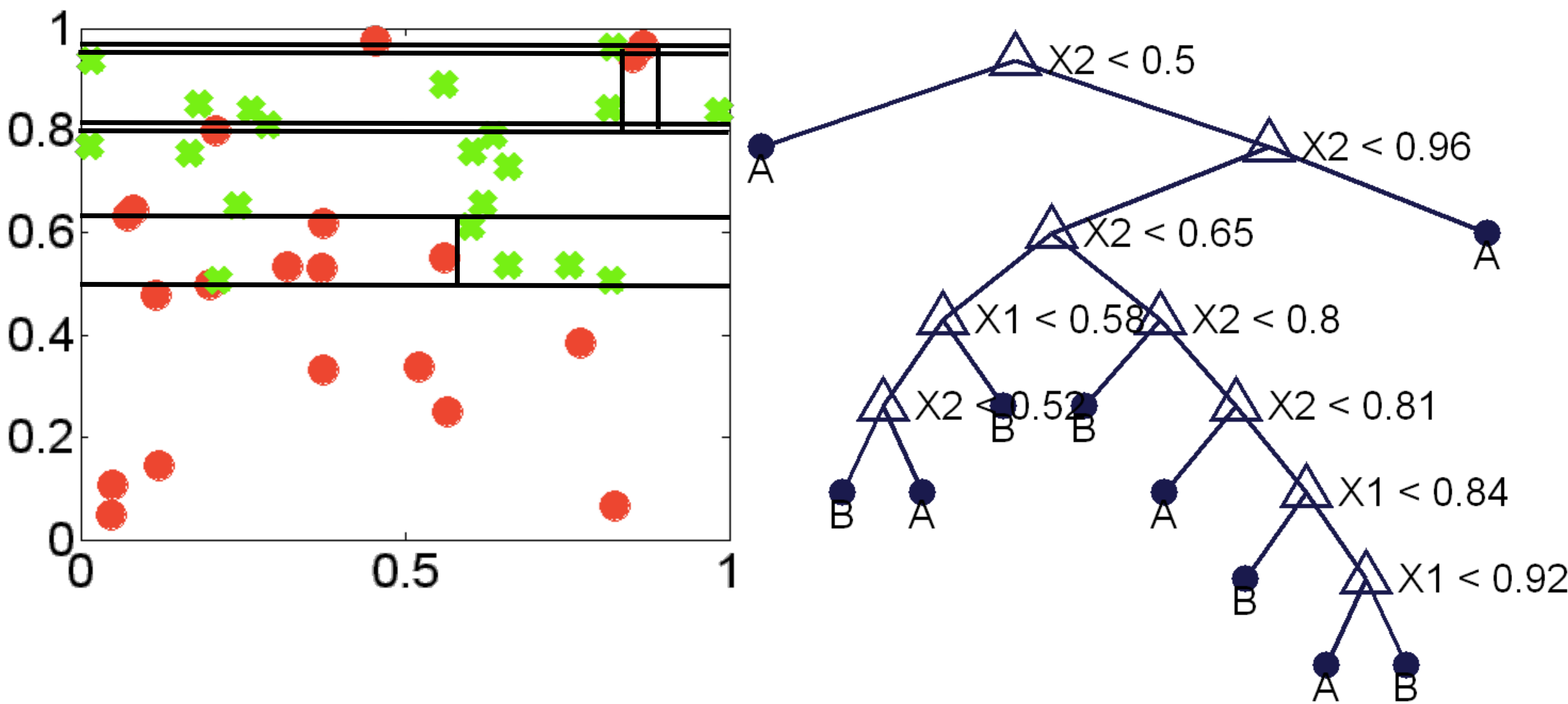
- Grow tree based on training data (*unpruned* tree)
- Prune the tree by removing useless nodes based on:
  - Additional test data (not used for training)
  - Statistical significance tests



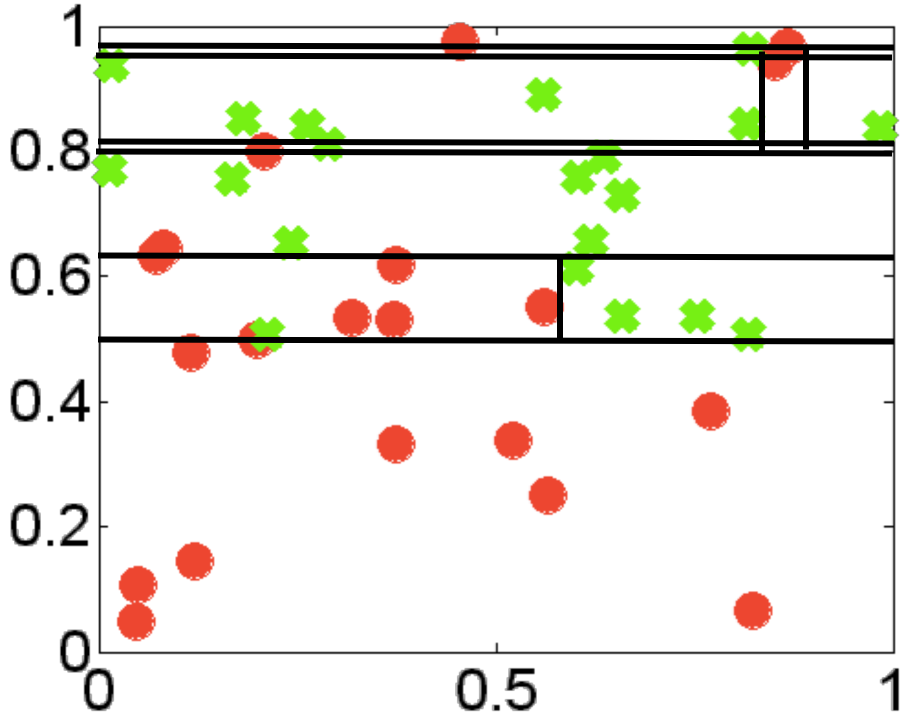
Training Data



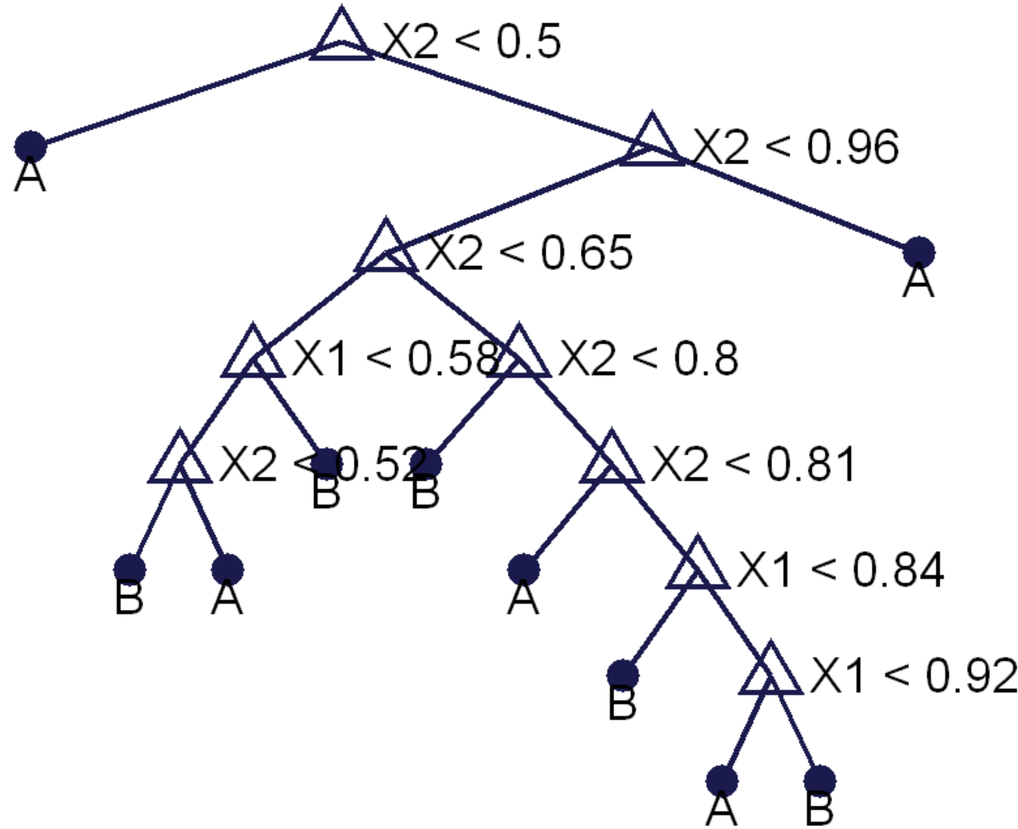
Unpruned decision tree from training data



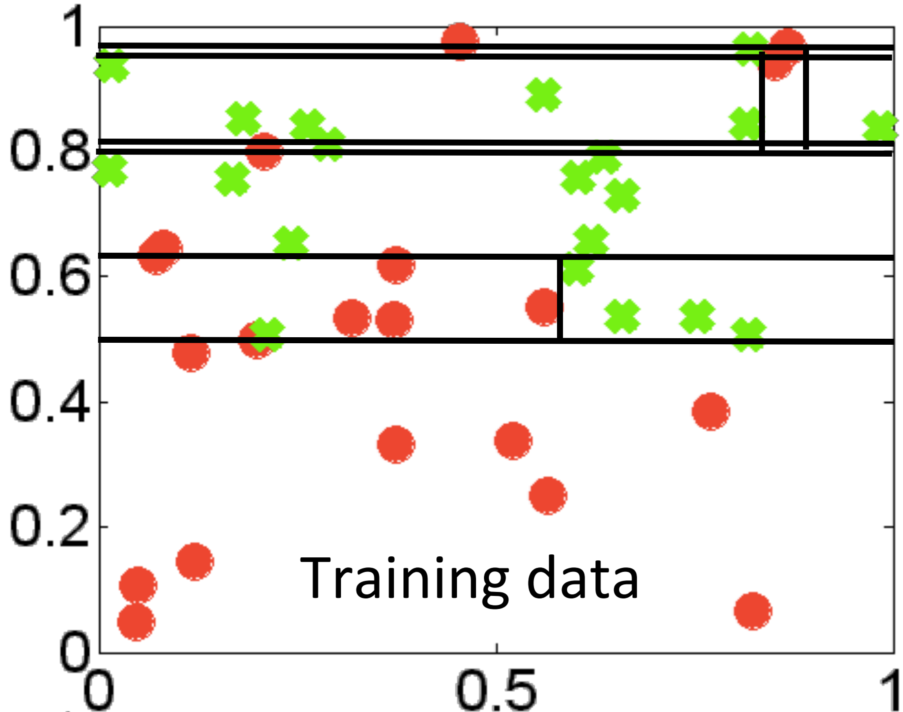
Unpruned decision tree from training data

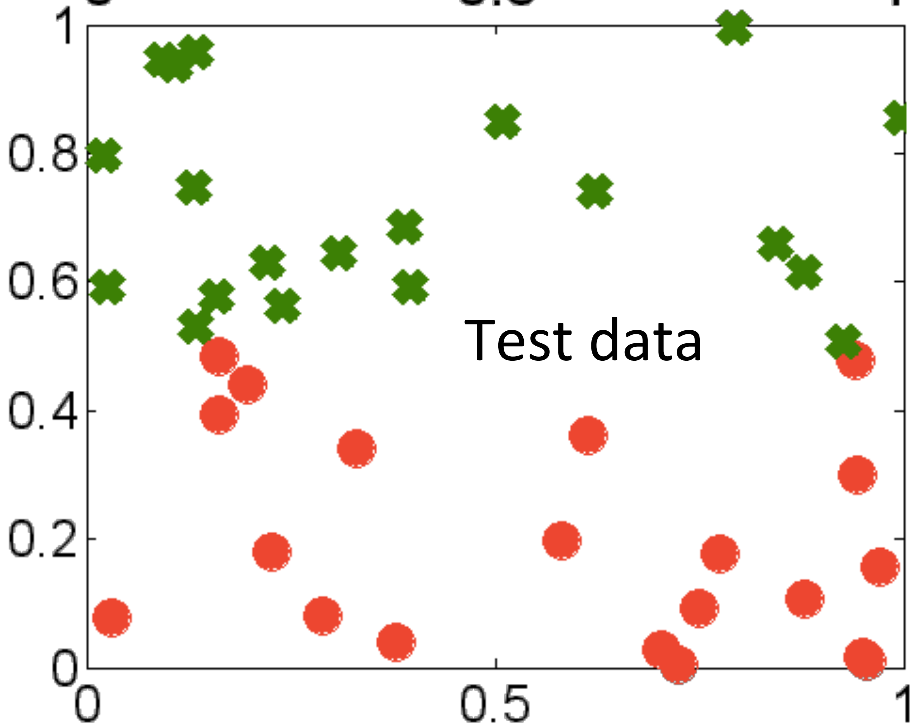


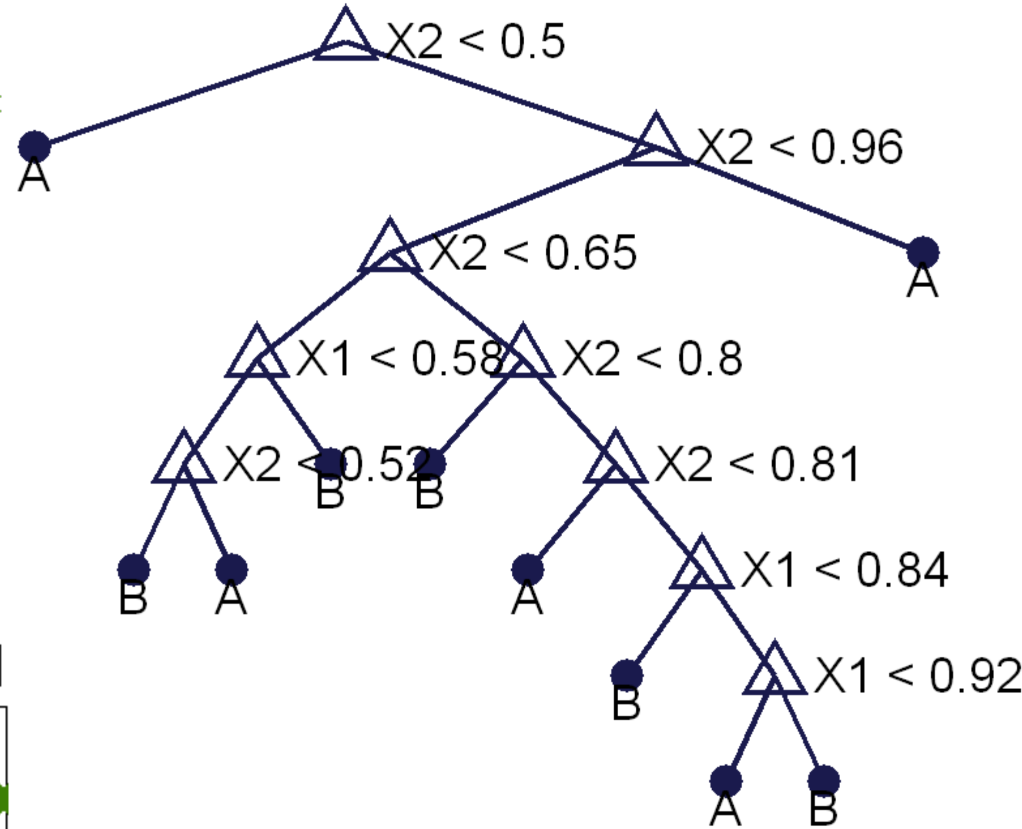
Training data
with the partitions induced
by the decision tree
(Notice the tiny regions at
the top necessary to
correctly classify the 'A'
outliers!)



Unpruned decision tree
from training data





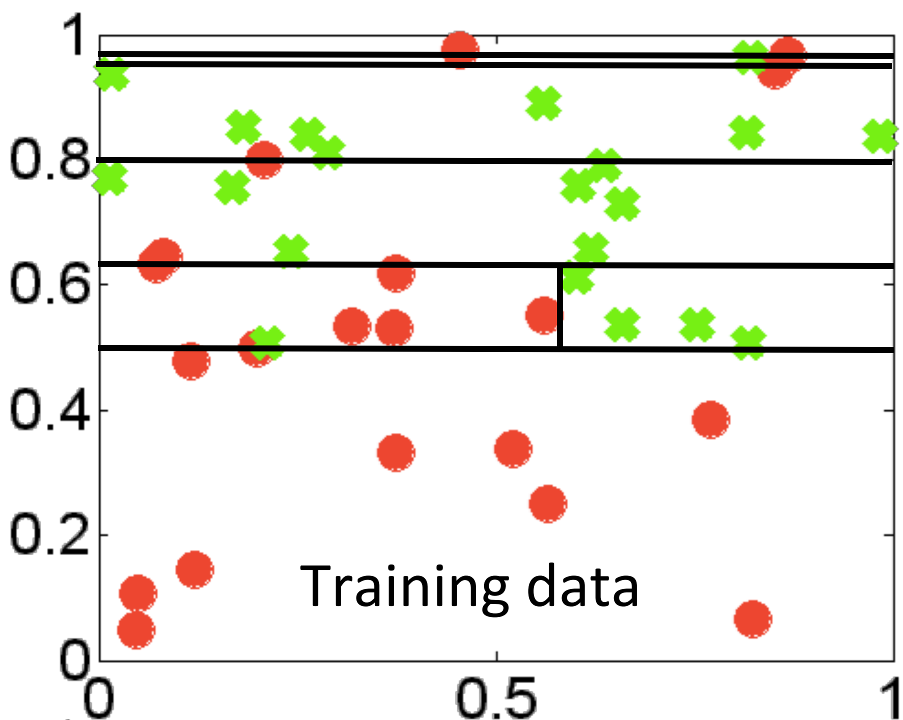


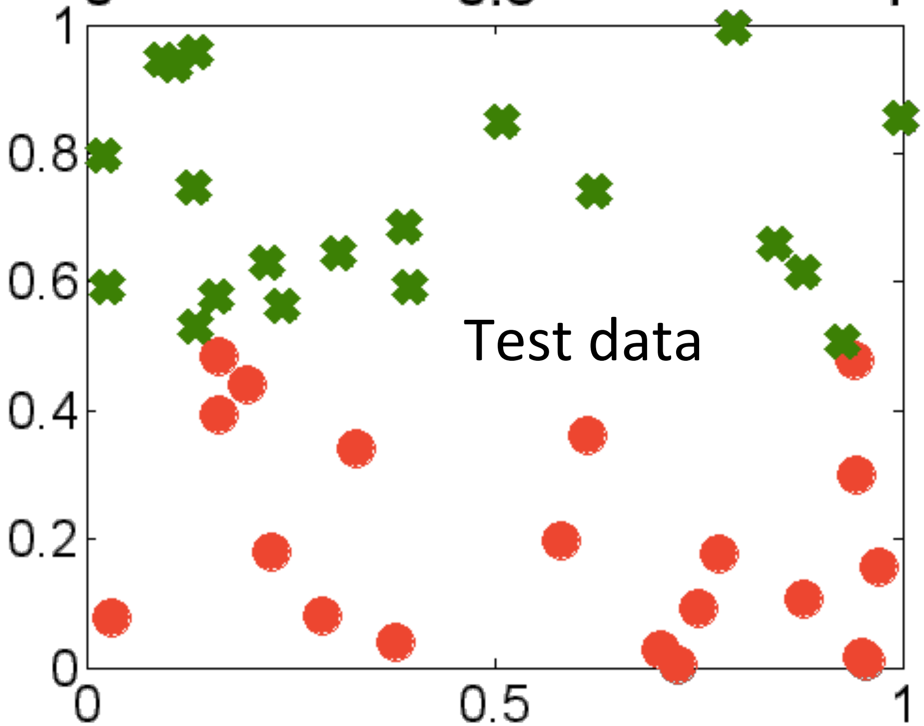
Unpruned decision tree from training data

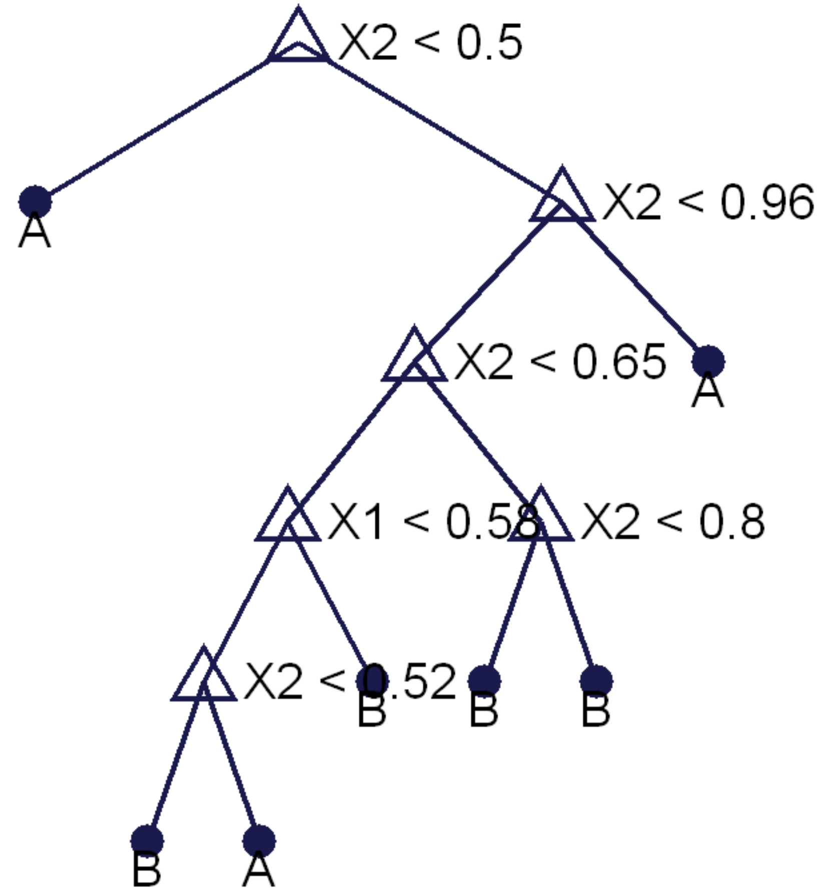
Performance (% correctly classified)

Training: 100%

Test: 77.5%





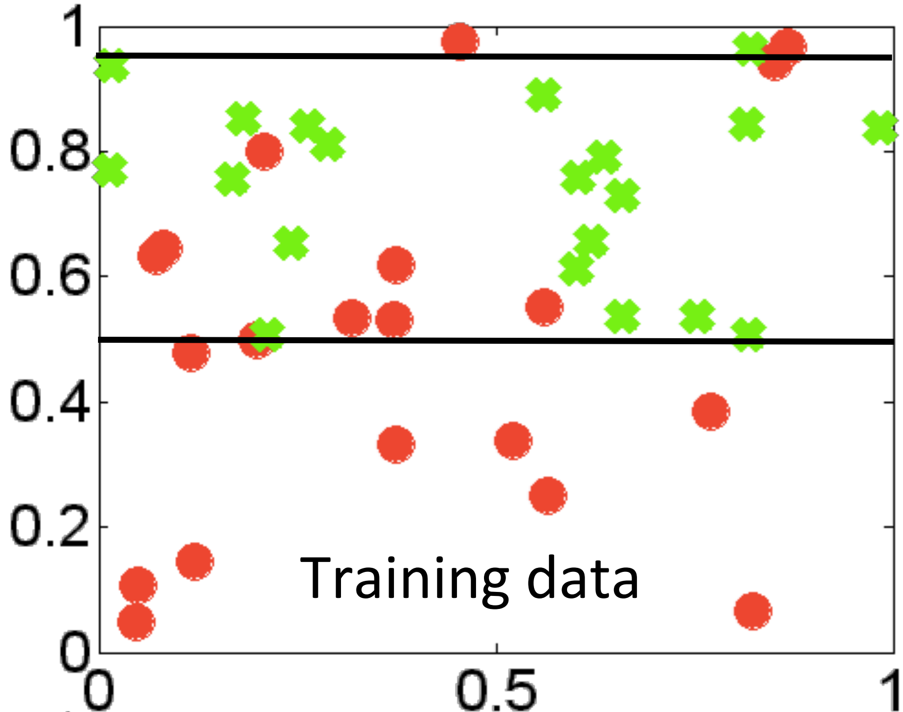


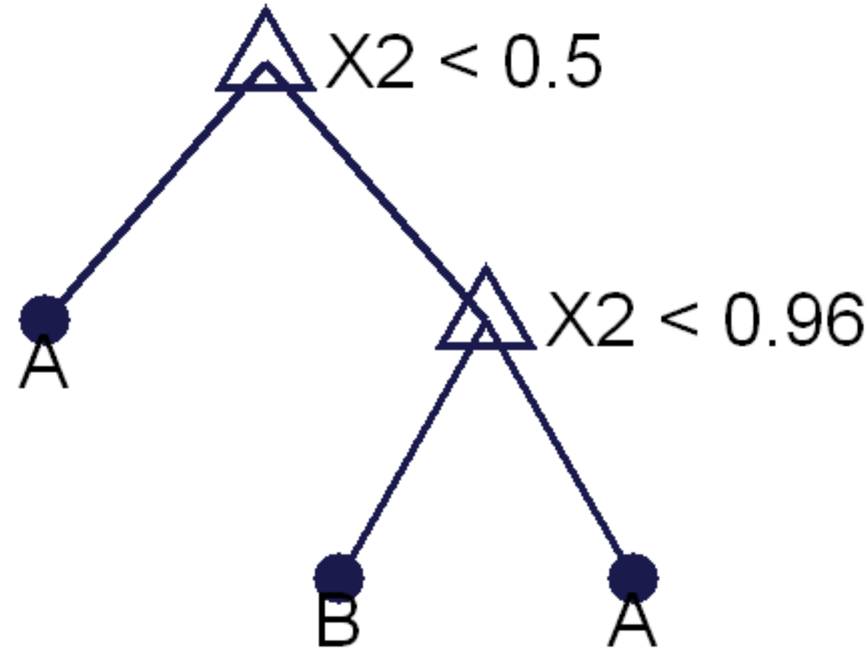
Pruned decision tree from training data

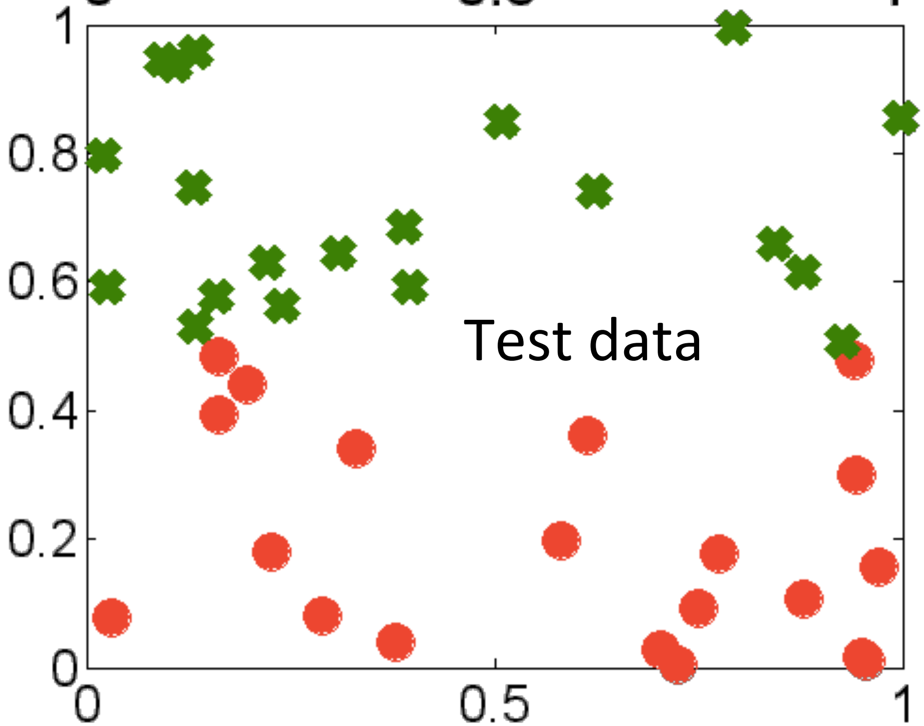
Performance (% correctly classified)

Training: 95%

Test: 80%





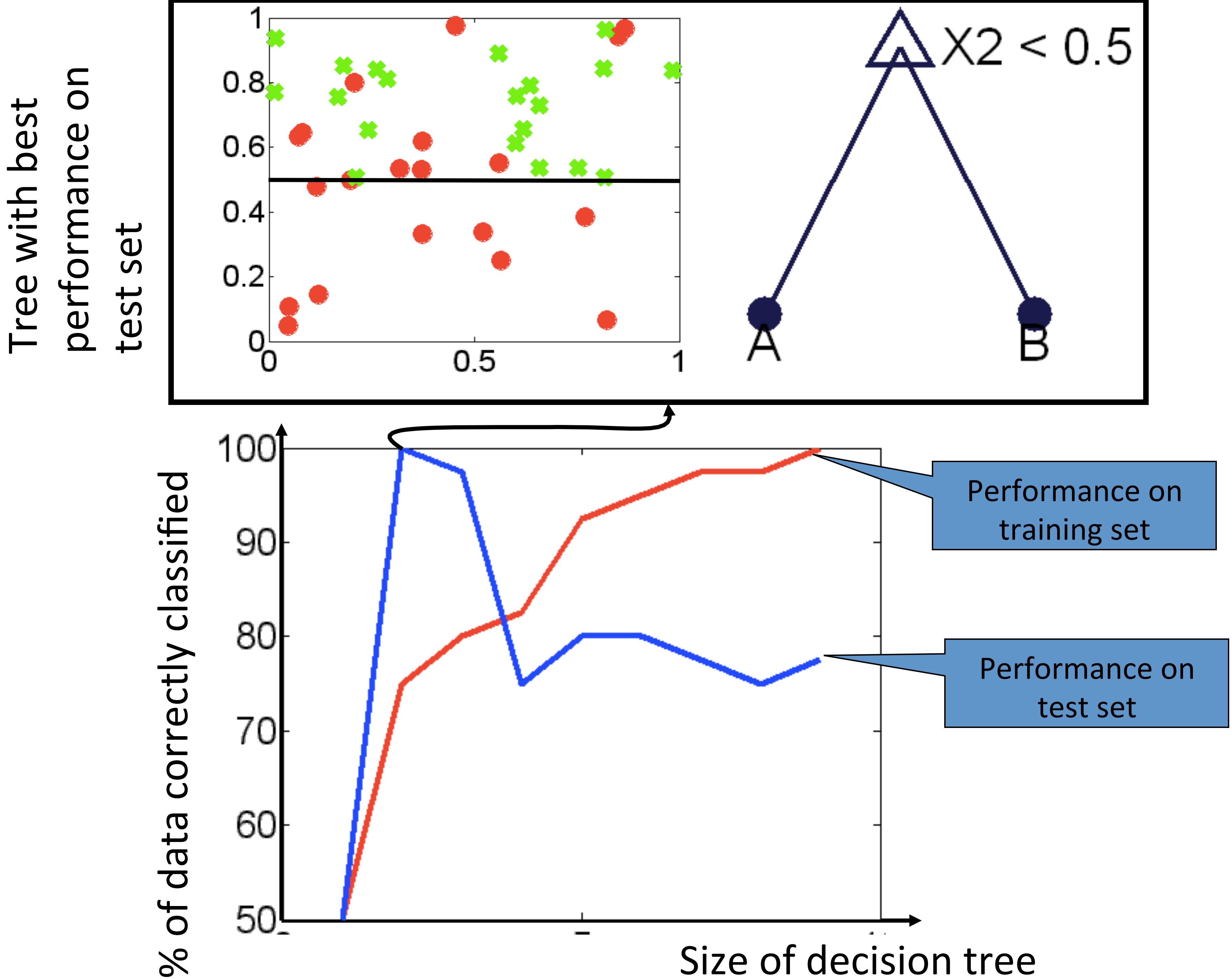


Pruned decision tree from training data

Performance (% correctly classified)

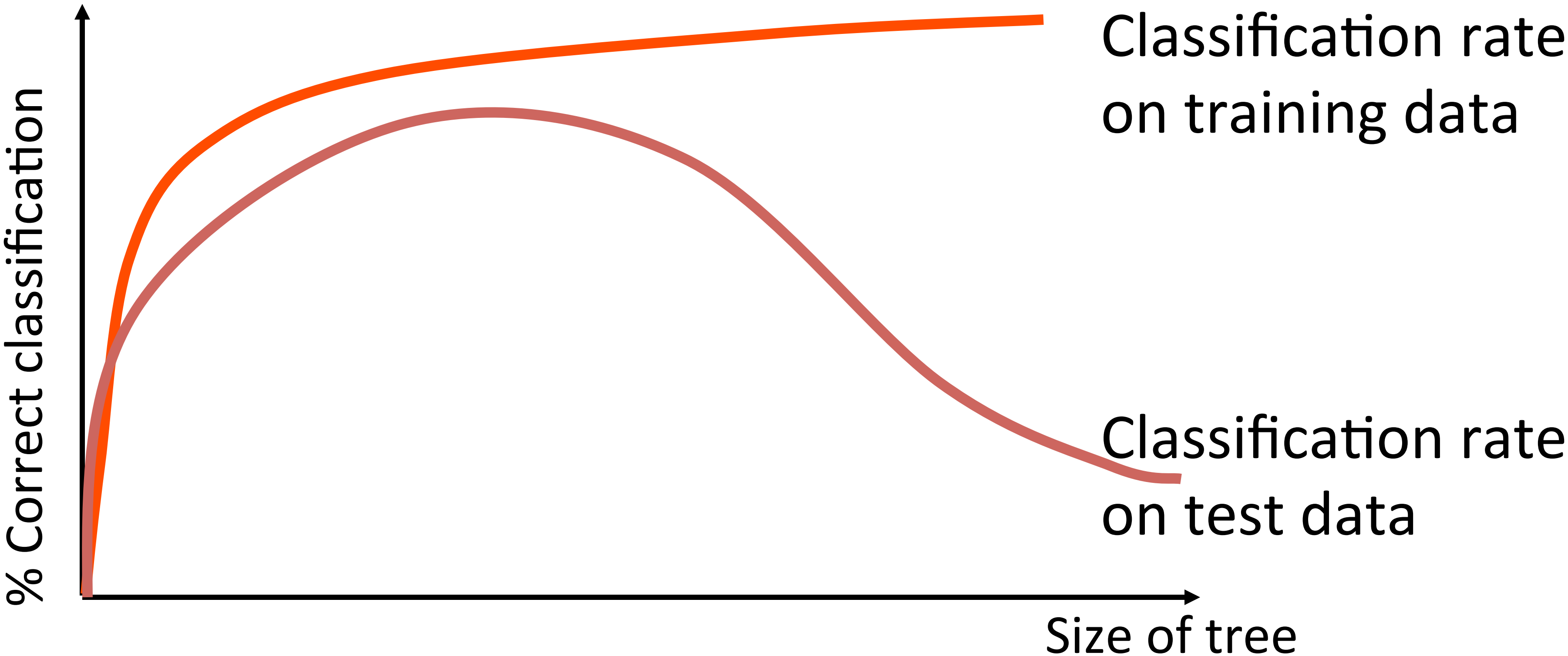
Training: 80%

Test: 97.5%

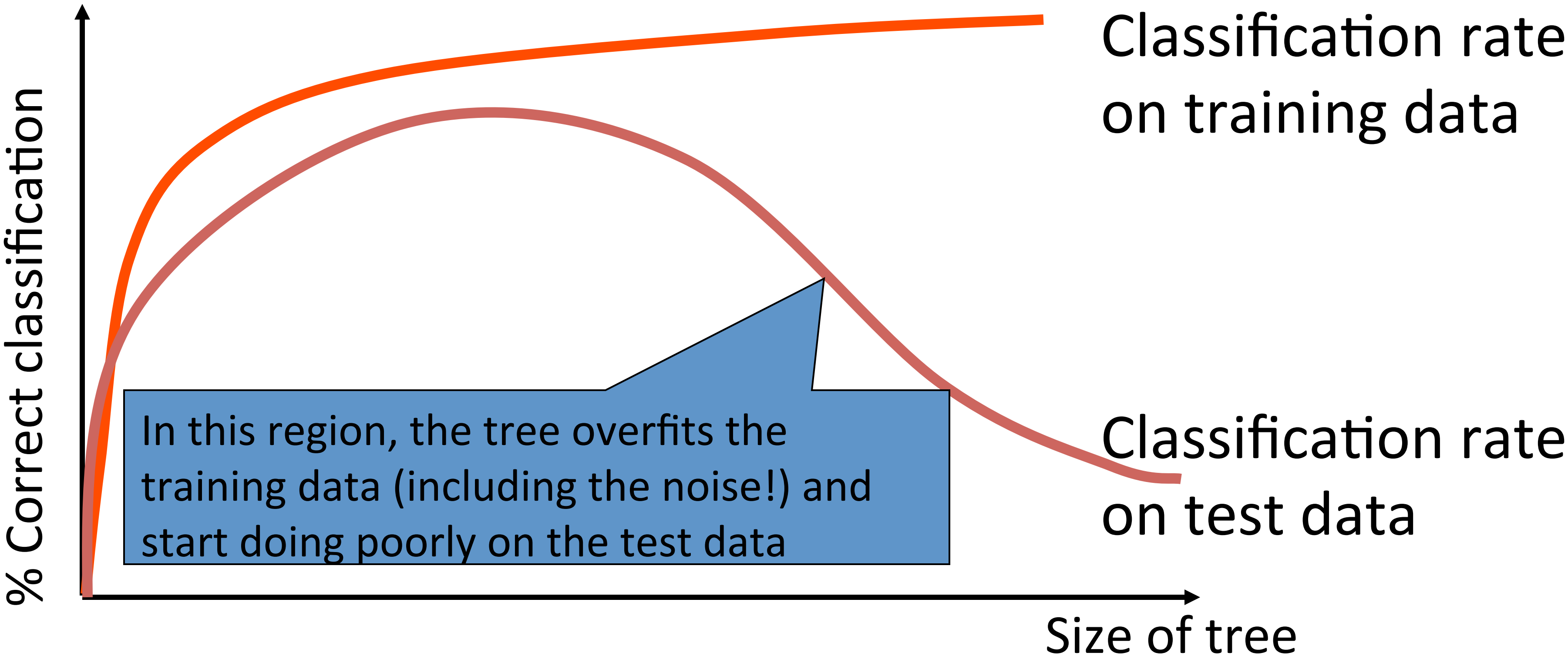


Tree with best performance on test set
X2 < 0.5
A
B
Performance on training set
Performance on test set
% of data correctly classified
Size of decision tree

Using Test Data



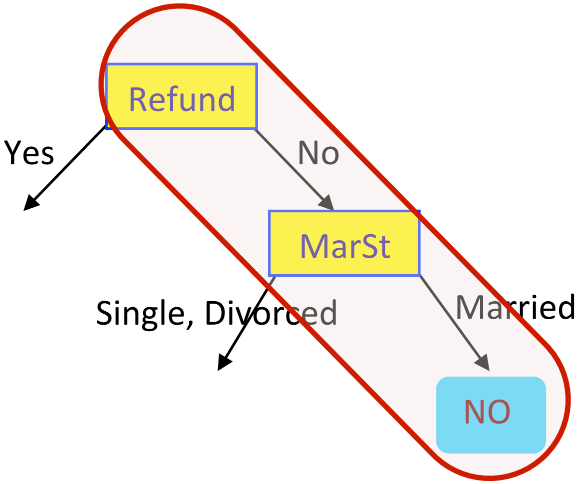
Using Test Data



- General principle: As the complexity of the classifier increases (depth of the decision tree), the performance on the training data increases and the performance on the test data decreases when the classifier overfits the training data.

When to Stop?

- Many strategies for picking simpler trees:
  - Pre-pruning
    - Fixed depth (e.g. ID3)
    - Fixed number of leaves
  - Use test data



  - Post-pruning
    - Chi-square test
      - Convert decision tree to a set of rules
      - Eliminate variable values in rules which are independent of label (using chi-square test for independence)
      - Simplify rule set by eliminating unnecessary rules
  - Information Criteria: MDL(Minimum Description Length)

Information Criteria

- Penalize complex models by introducing cost

$$\widehat{f} = \arg\min_T \left\{ \underbrace{\frac{1}{n}\sum_{i=1}^{n} \text{loss}(\widehat{f}_T(X_i), Y_i)}_{\text{log likelihood}} + \underbrace{\text{pen}(T)}_{\text{cost}} \right\}$$

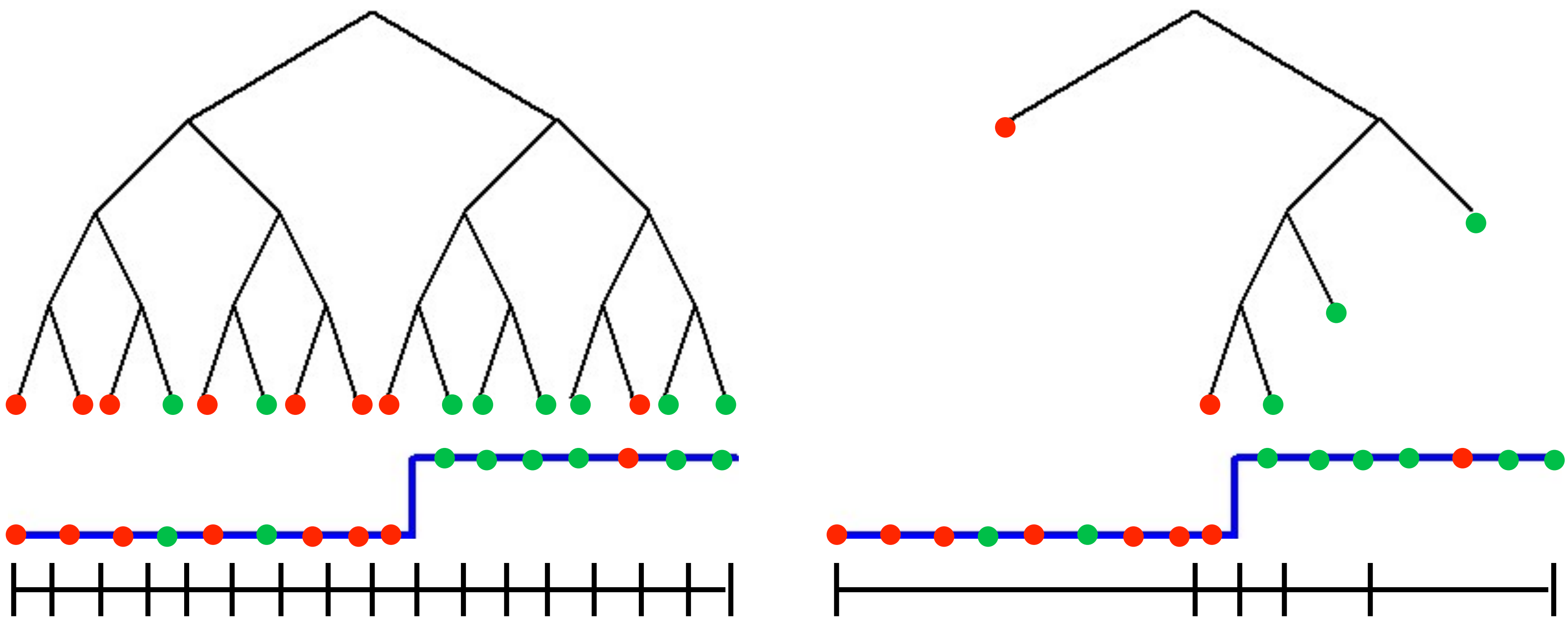
$$\begin{aligned} \text{loss}(\widehat{f}_T(X_i), Y_i) &= (\widehat{f}_T(X_i) - Y_i)^2 && \text{regression} \\ &= \mathbf{1}_{\widehat{f}_T(X_i) \neq Y_i} && \text{classification} \end{aligned}$$

$\text{pen}(T) \propto |T|$ penalize trees with more leaves

CART – optimization can be solved by dynamic programming

Decision Trees - Overfitting

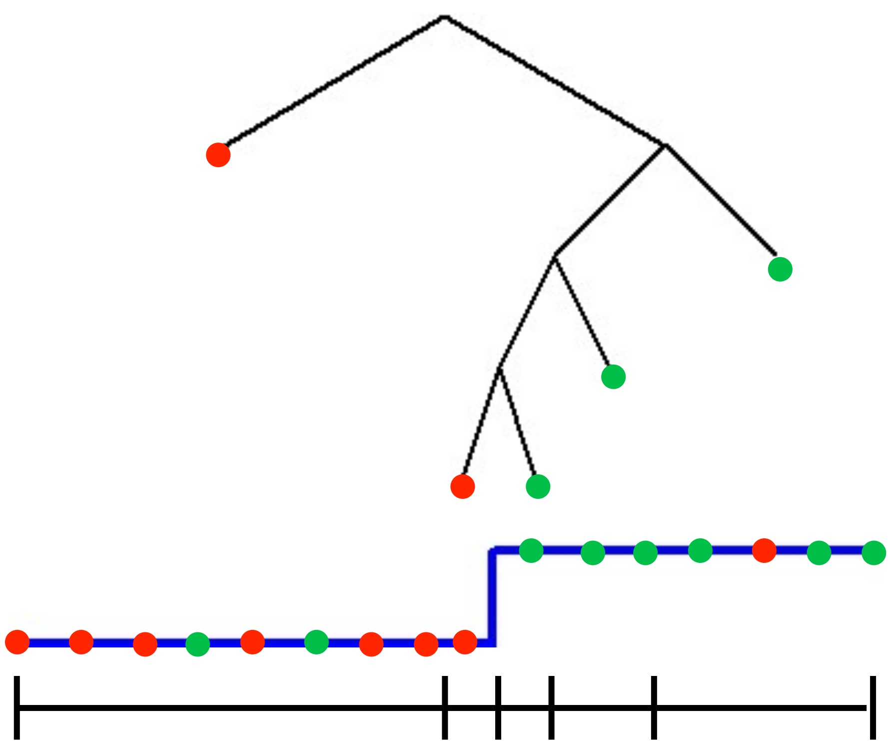
One training example per leaf – overfits, need compact/pruned decision tree



How to assign label to each leaf

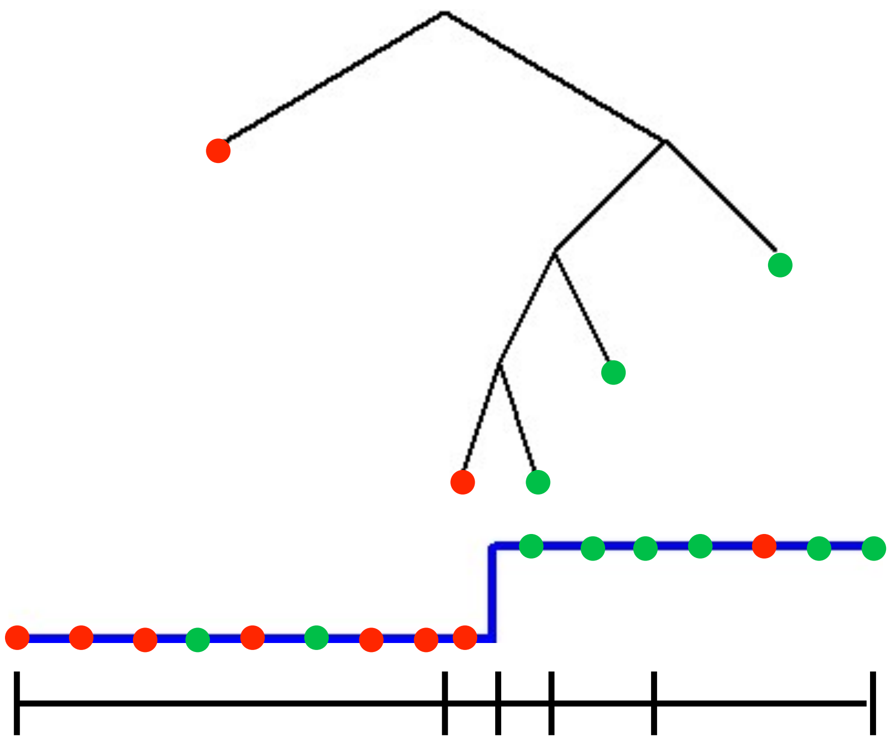
Classification – Majority vote

Regression – ?

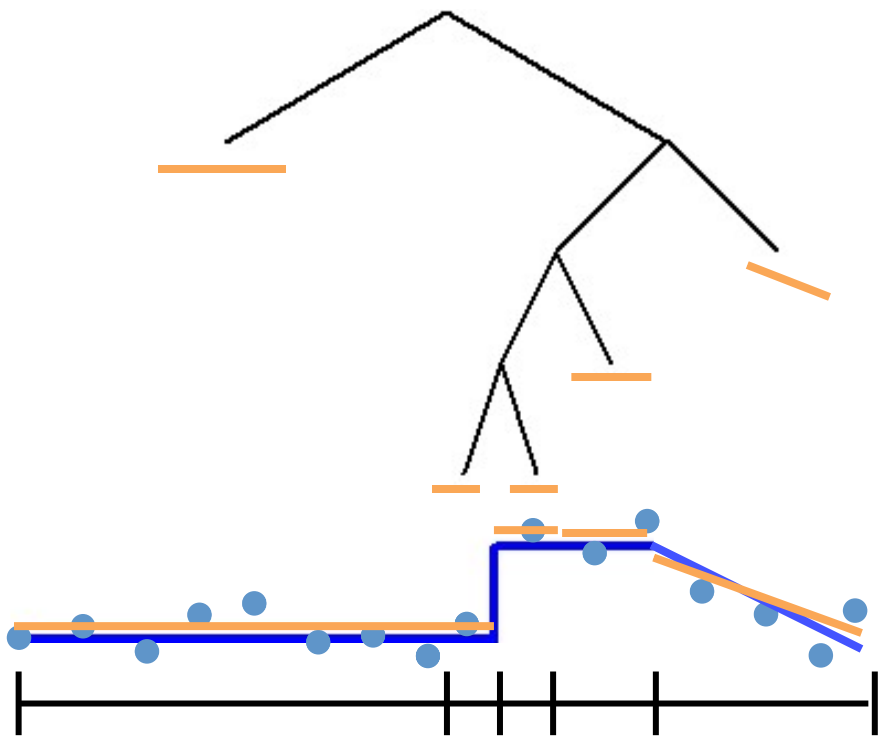


How to assign label to each leaf

Classification – Majority vote



Regression – Constant/ Linear/Poly fit

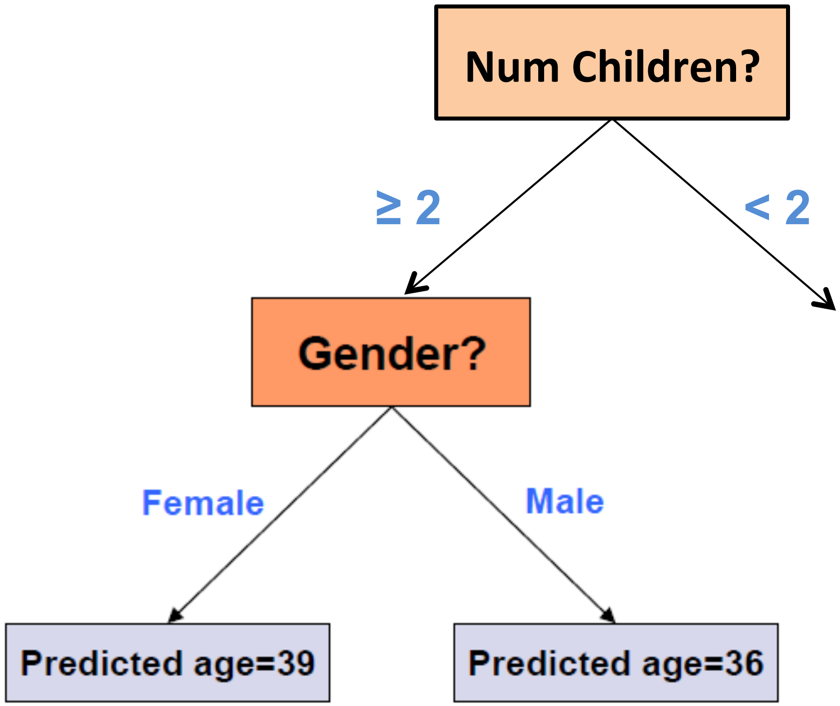


Regression trees

Predicted outcome is not a class, but a real number

$X^{(1)} \quad \ldots\ldots \quad X^{(p)} \quad Y$

Gender	Rich?	Num. Children	# travel per yr.	Age
F	No	2	5	38
M	No	0	2	25
M	Yes	1	0	72
:	:	:	:	:



Average (fit a constant) using training data at the leaves

What you should know

- Decision trees are one of the most popular data mining tools
  - Simplicity of design
  - Interpretability
  - Ease of implementation
  - Good performance in practice (for small dimensions)
- Information gain to select attributes (ID3, C4.5,…)
- Decision trees will overfit
  - Must use tricks to find "simple trees", e.g.,
    - Pre-Pruning: Fixed depth/Fixed number of leaves
    - Post-Pruning: Chi-square test of independence
    - Complexity Penalized/MDL model selection
- Can be used for classification, regression and density estimation too