Decision Trees & Neural Nets

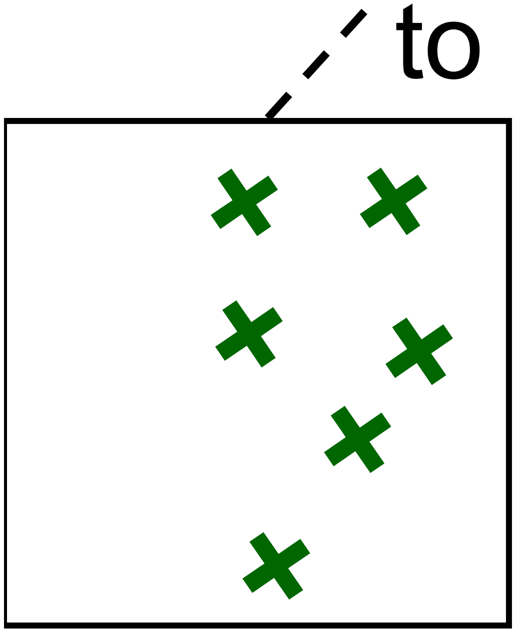
Part II

Illah Nourbakhsh version

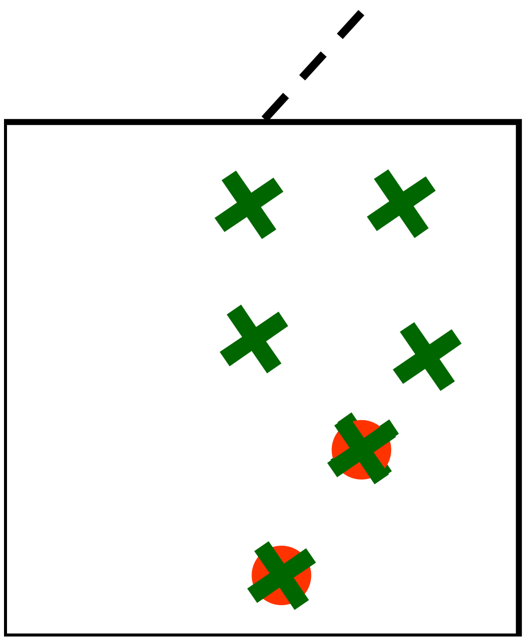
Basic Questions

- 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?

Pure and Impure Leaves and When to Stop Splitting - *forced*



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



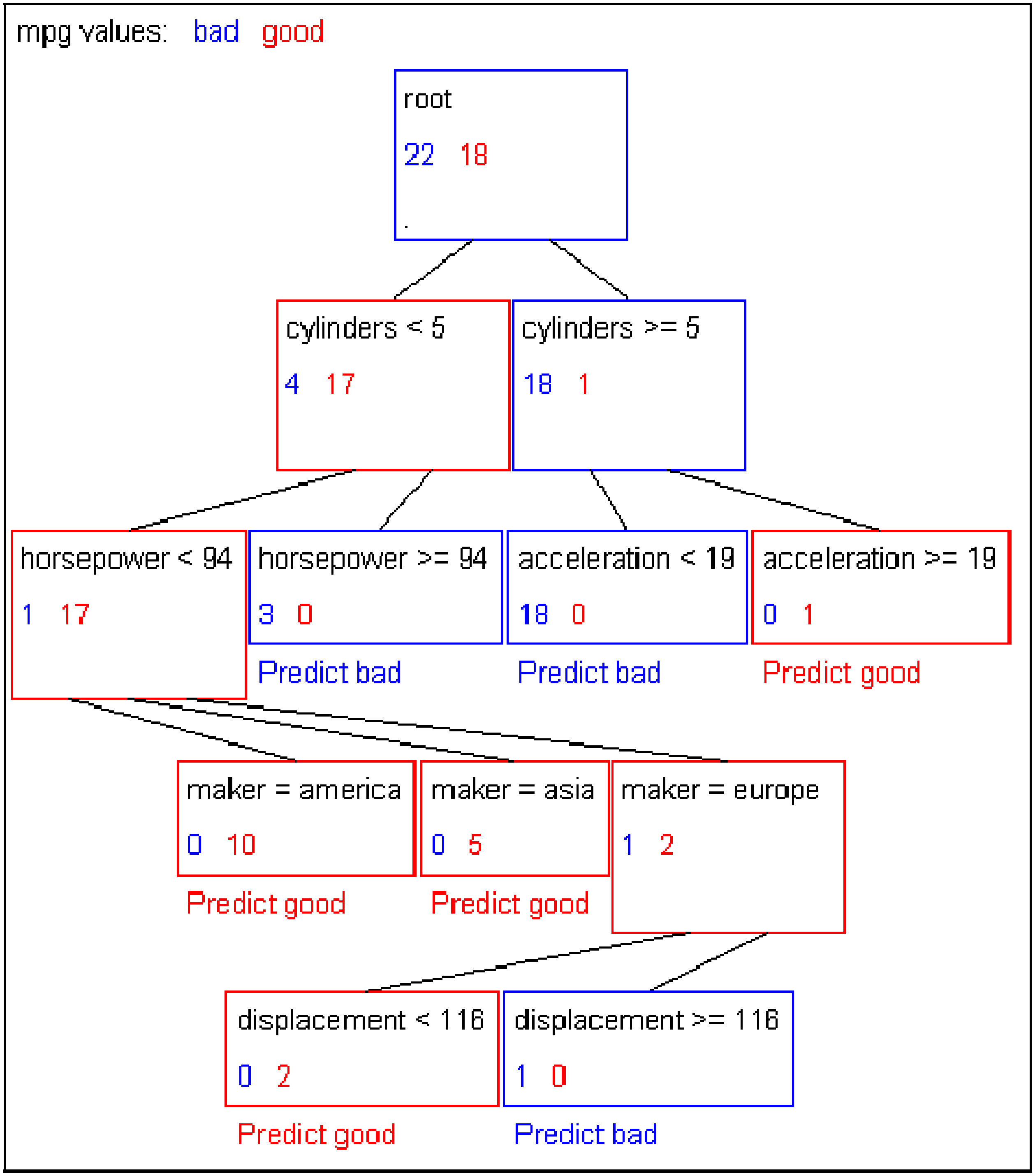
Several data points have exactly the same attributes even though they are from ***different*** classes → We cannot split any further → We still declare the node to be a leaf, but it will output the class that is the ***majority*** of the classes in the node (in this example, '**green** crosses').

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 Trees So Far

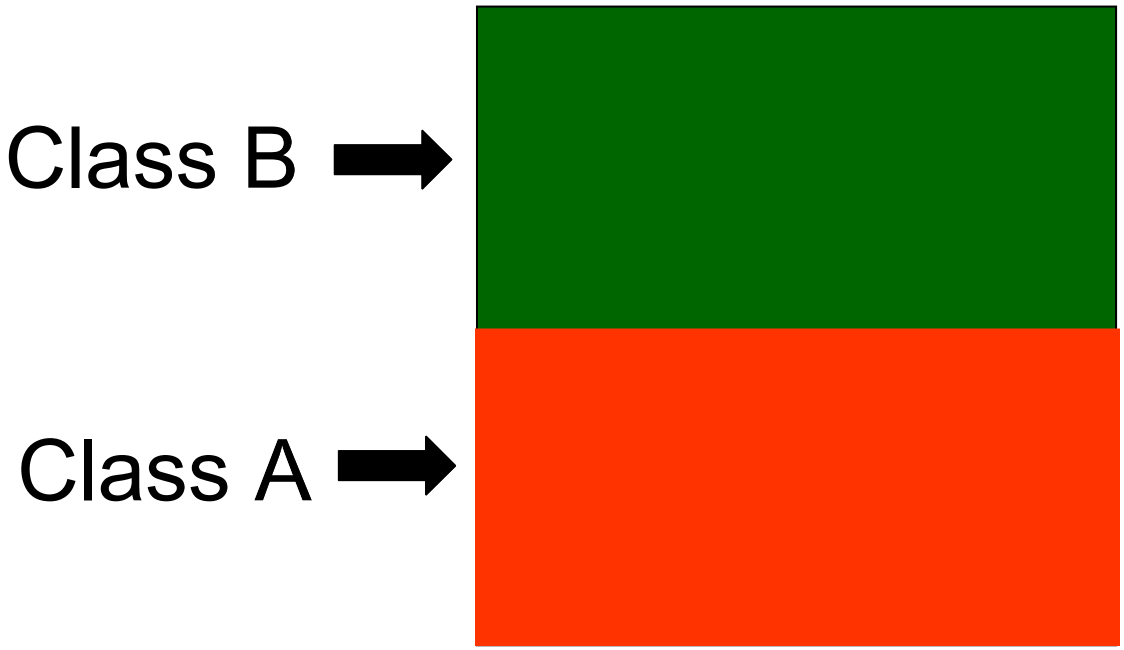
- Given *R* observations from training data, each with *M* attributes *X* and a class attribute *Y*, construct a sequence of tests (decision tree) to predict the class attribute *Y* from the attributes *X*
- Basic strategy for defining the tests ("when to split") → maximize the information gain on the 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***



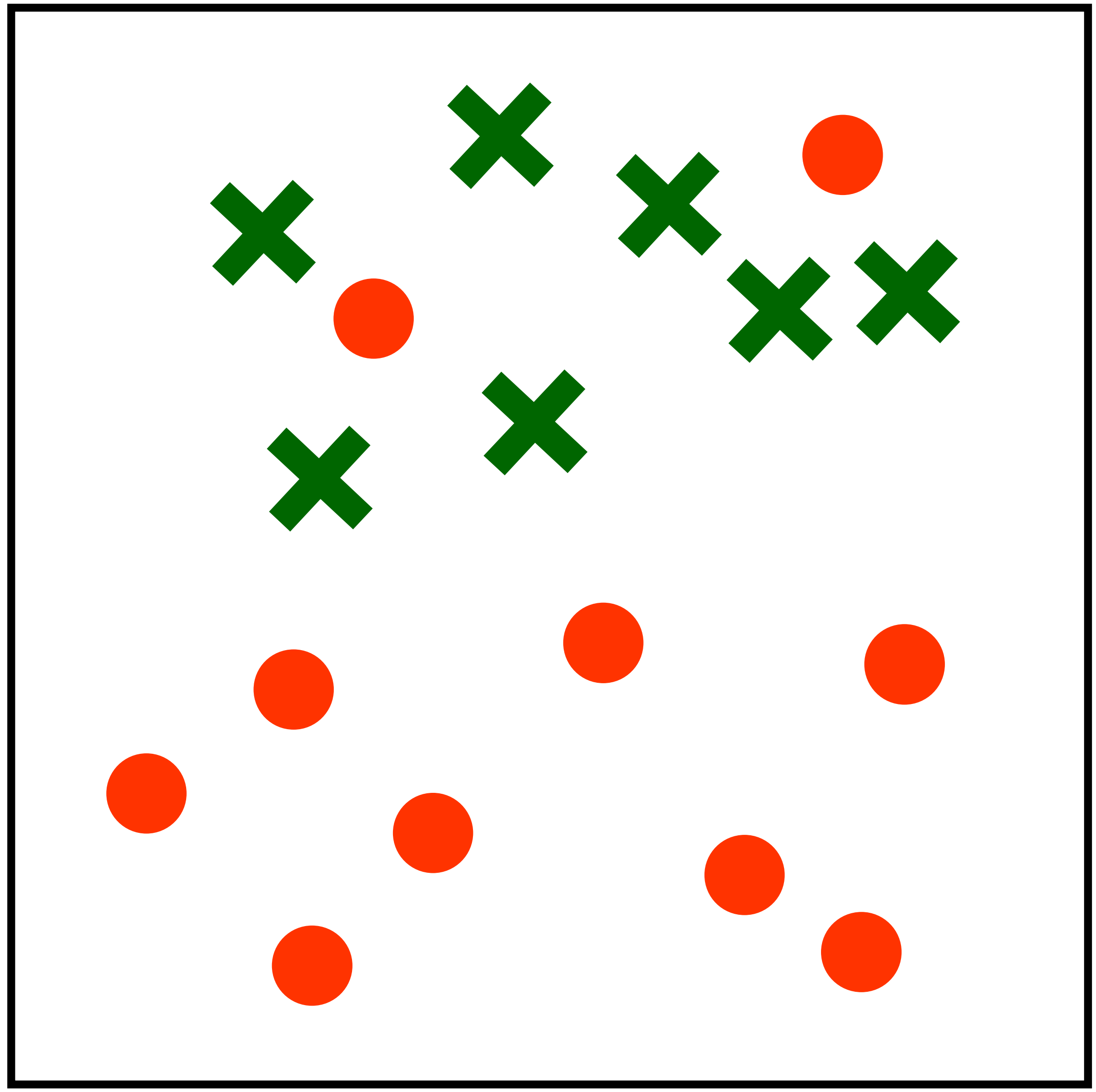
Side example with both discrete and continuous attributes: Predicting MPG ('GOOD' or 'BAD') from attributes:

- Cylinders
- Horsepower
- Acceleration
- Maker (discrete)
- Displacement

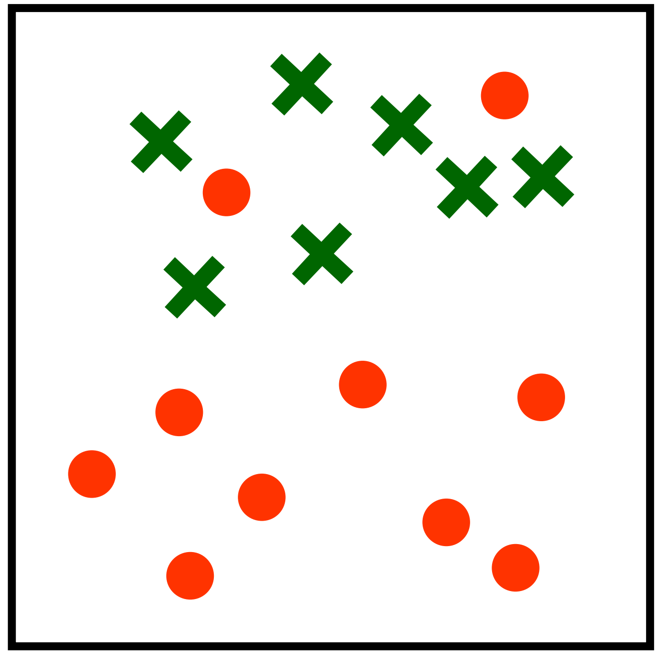
The Overfitting Problem: Example



- 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

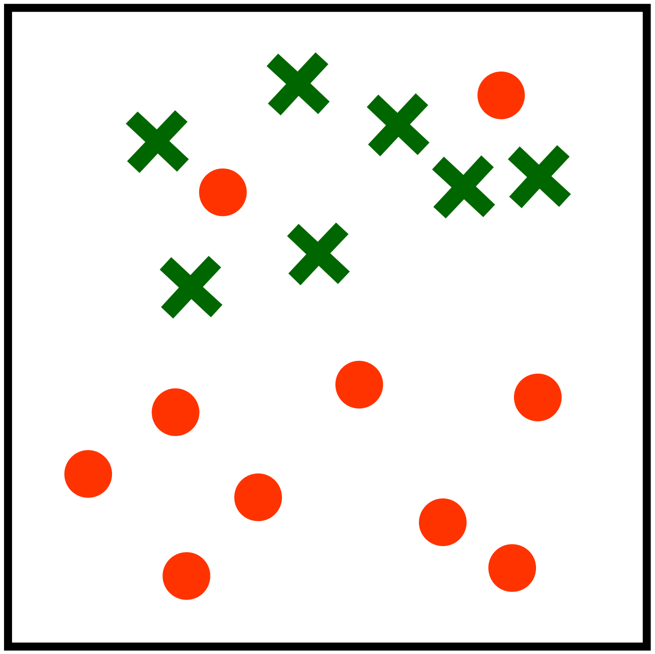


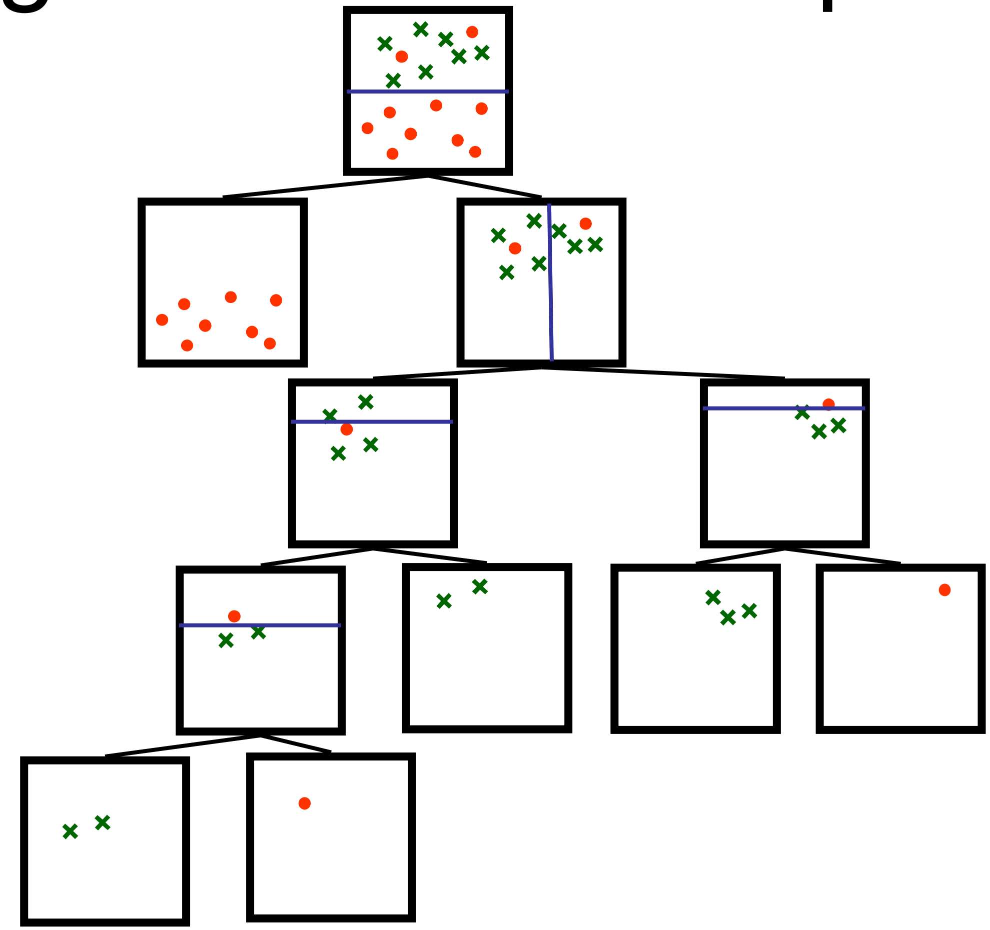
The Overfitting Problem: Example



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

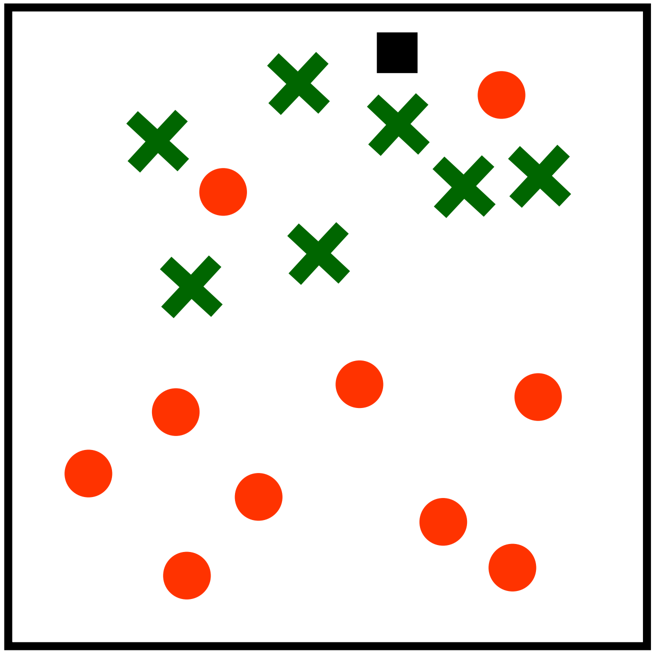
The Overfitting Problem: Example

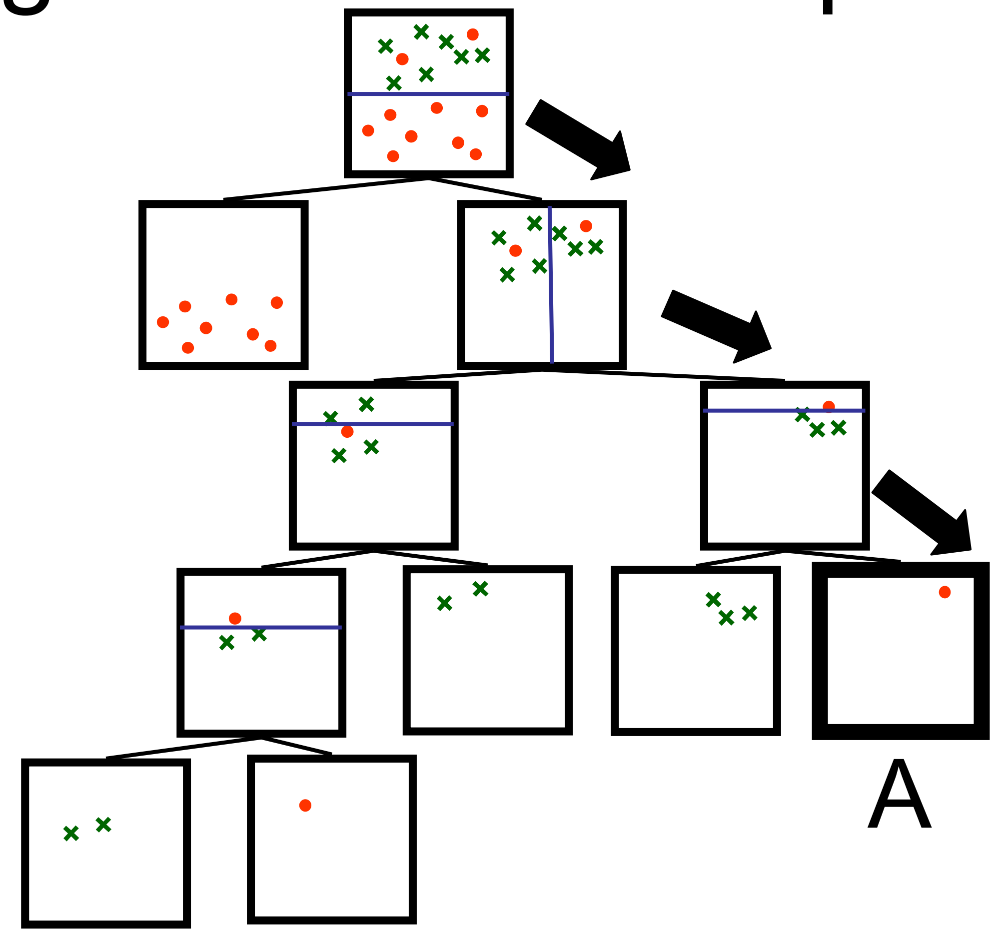




- Because of the noise, the resulting decision tree is far more complicated than it should be
- This is because the learning algorithm tries to classify *all of the training set perfectly* → This is a fundamental problem in learning: *overfitting*

The Overfitting Problem: Example

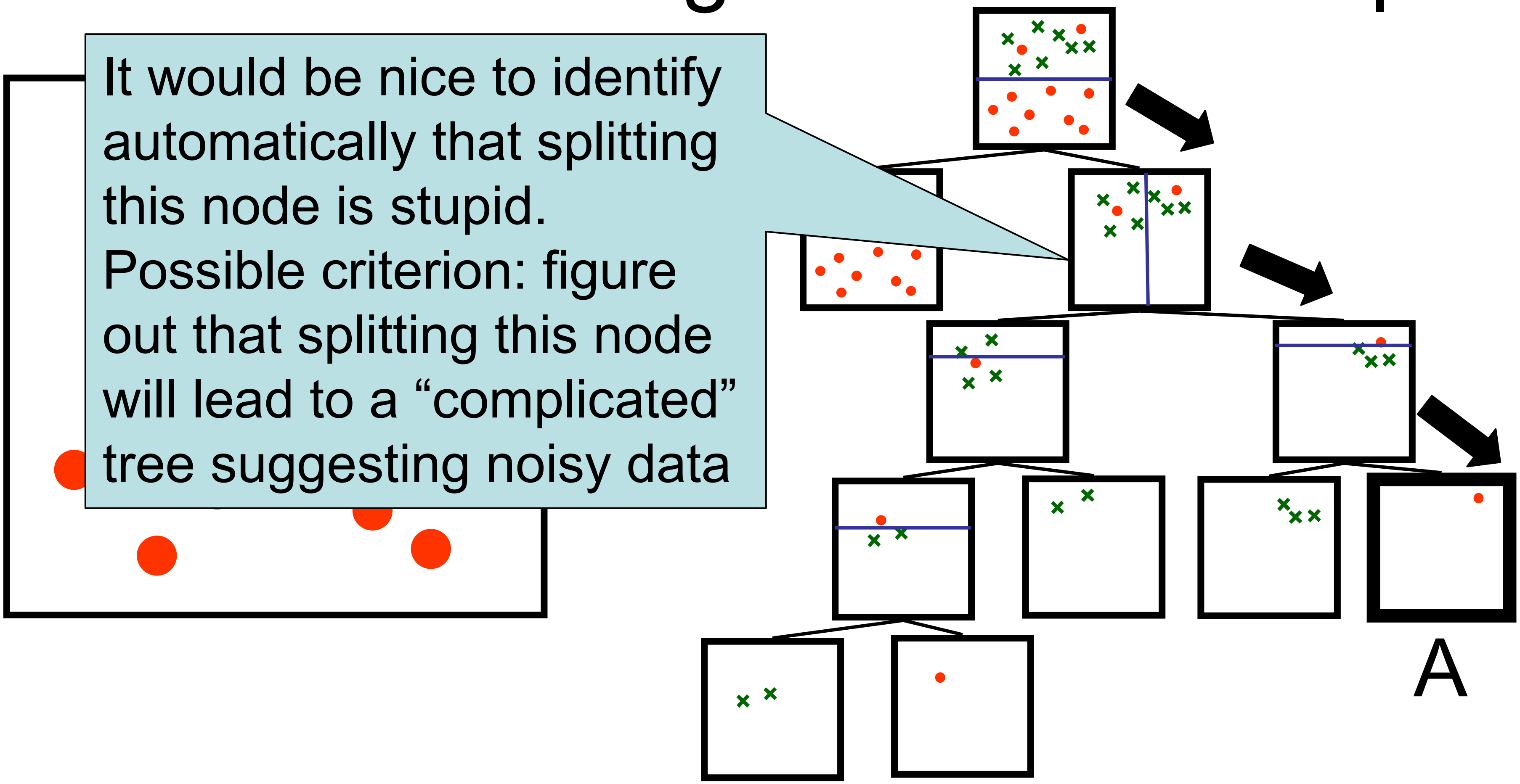




- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

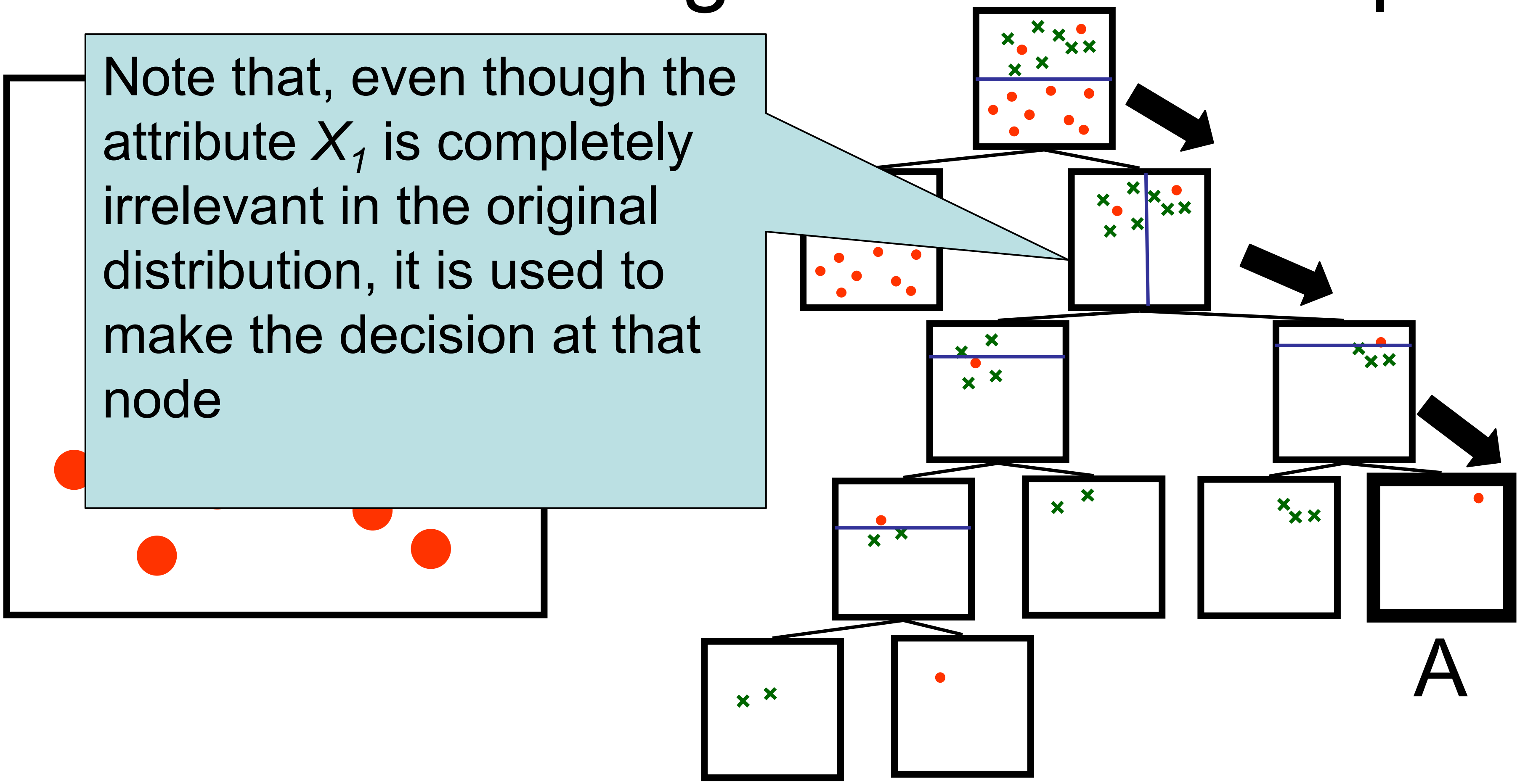
The Overfitting Problem: Example

It would be nice to identify automatically that splitting this node is stupid.
Possible criterion: figure out that splitting this node will lead to a “complicated” tree suggesting noisy data



- 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’

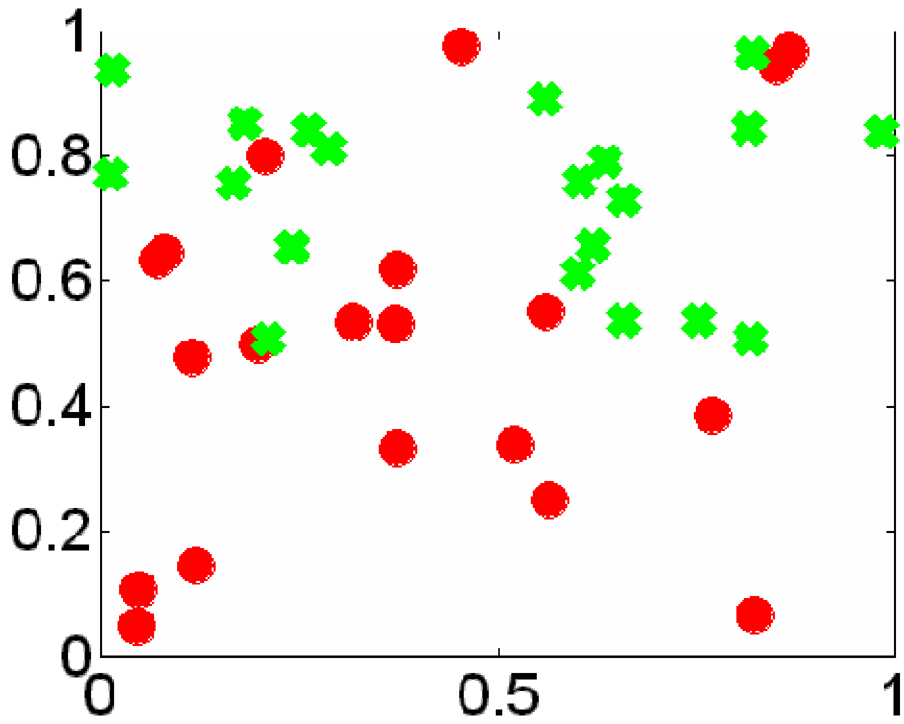
The Overfitting Problem: Example



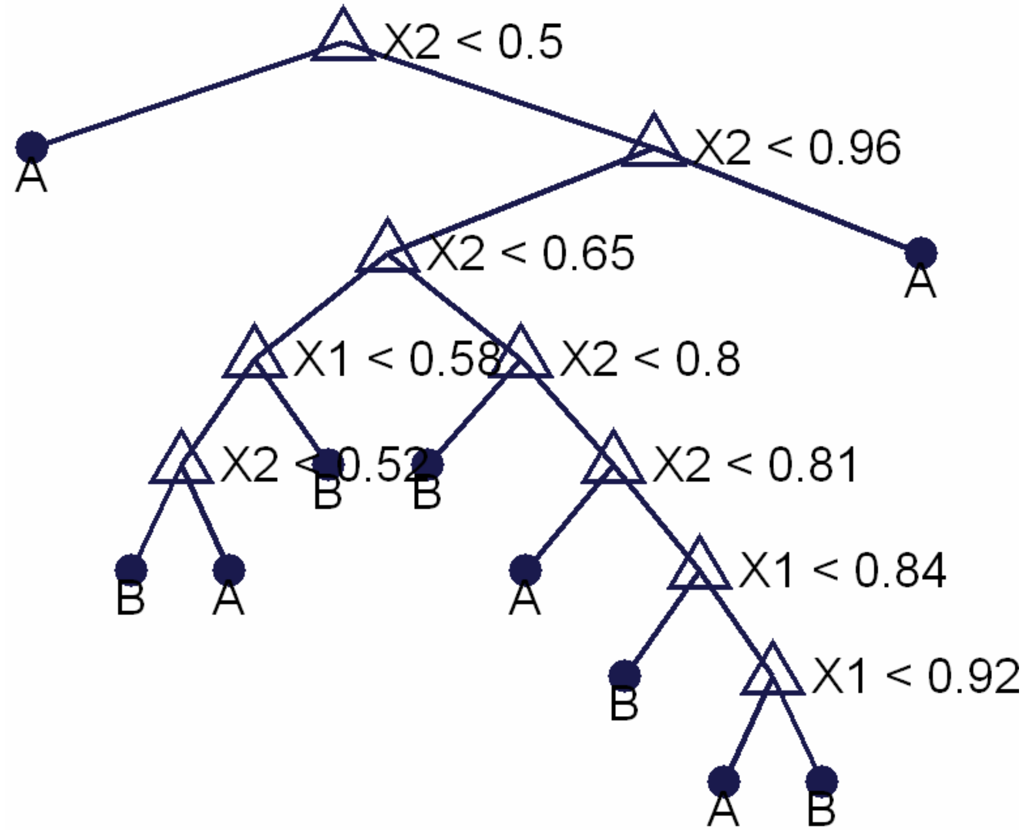
- The effect of overfitting is that the tree is guaranteed to classify the training data perfectly, but it may do a terrible job at classifying new test data.
- Example: (0.6,0.9) is classified as ‘A’

Possible Overfitting Solutions

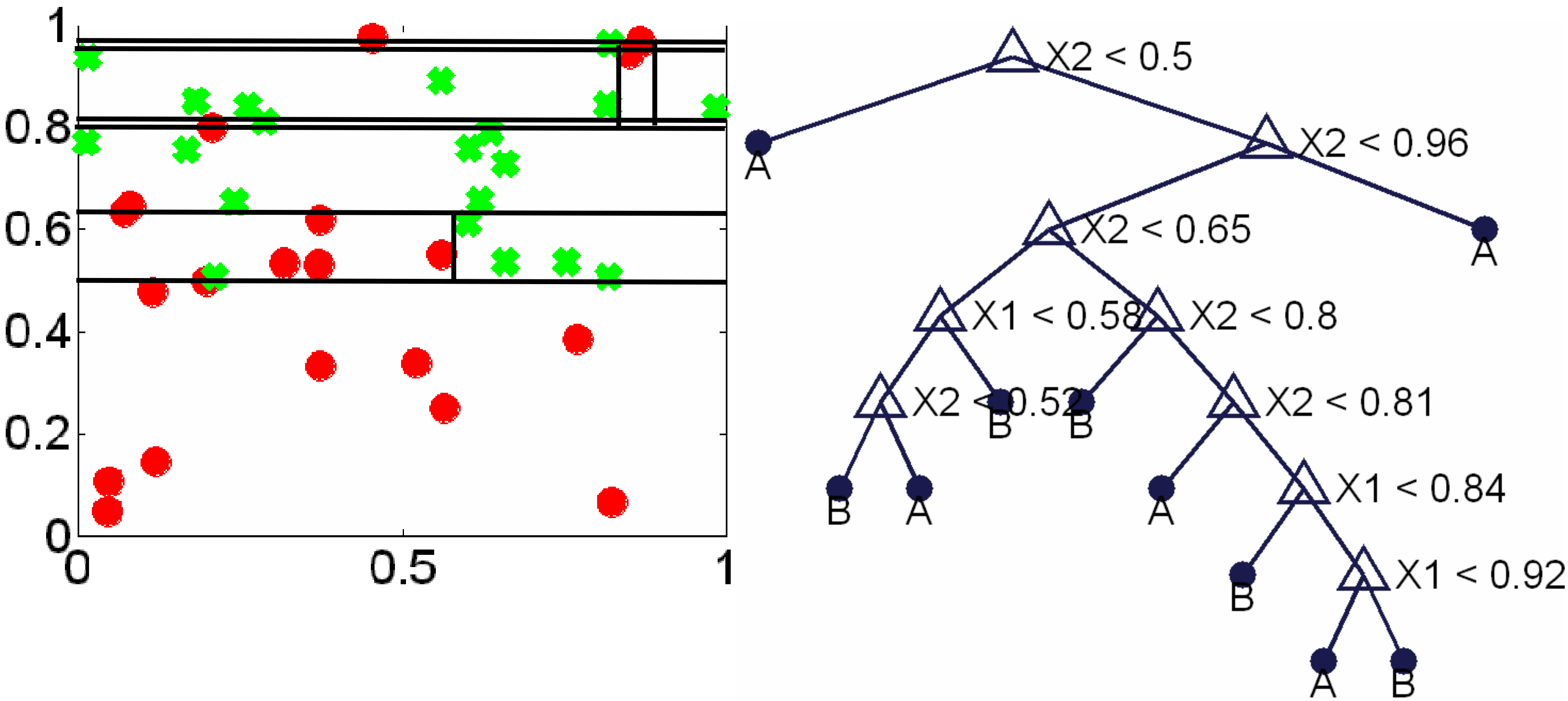
- Grow tree based on training data (*unpruned* tree)
- Prune the tree by removing useless nodes based on:
  - Additional test data (not used for training)
  - Statistical significance tests



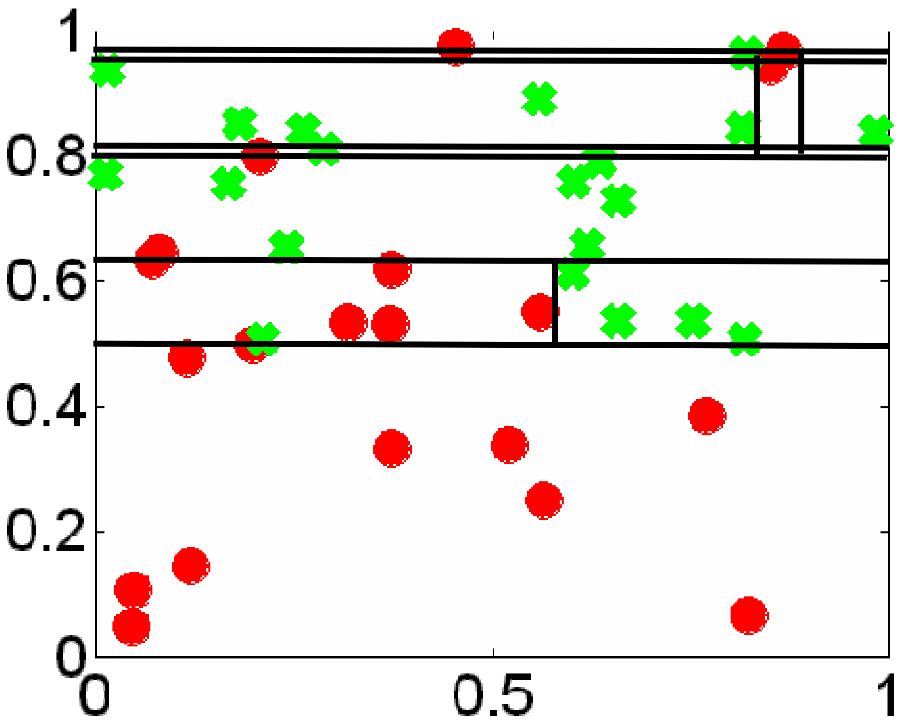
Training Data



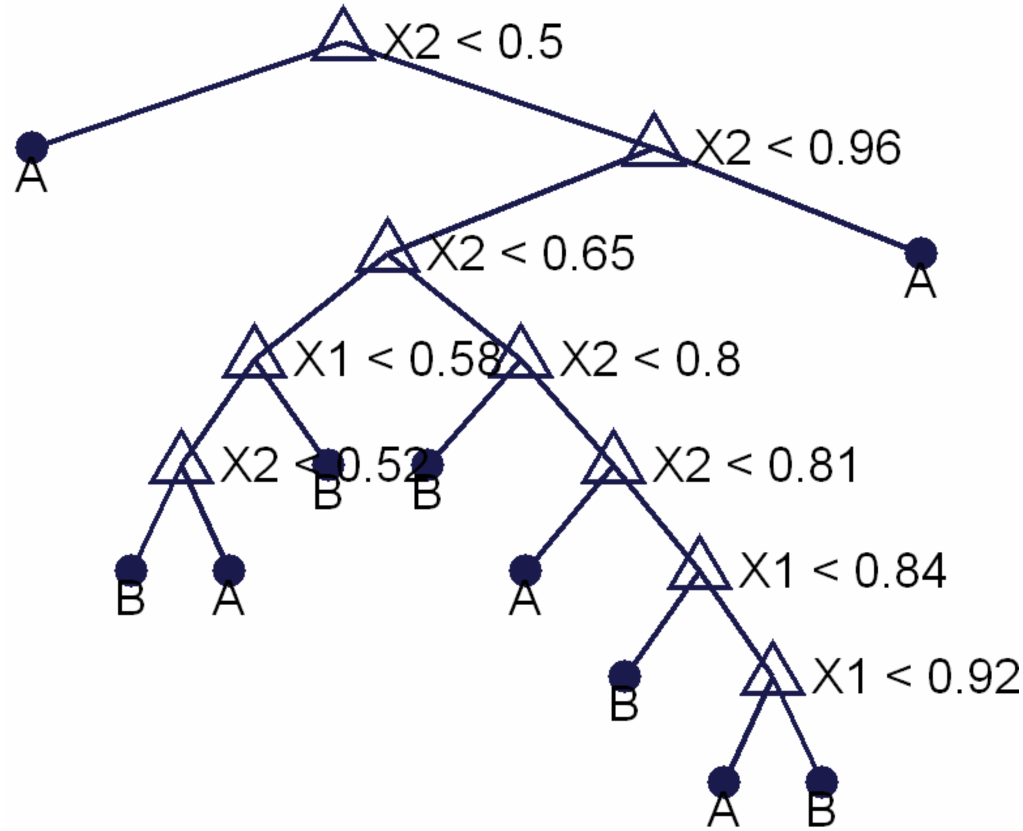
Unpruned decision tree
from training data



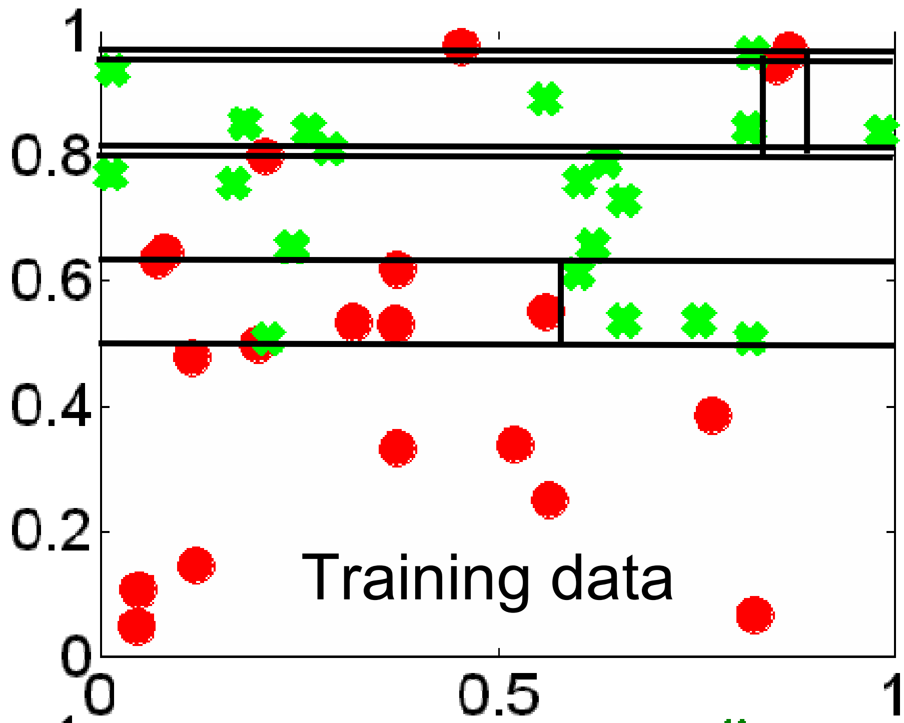
Unpruned decision tree
from training data

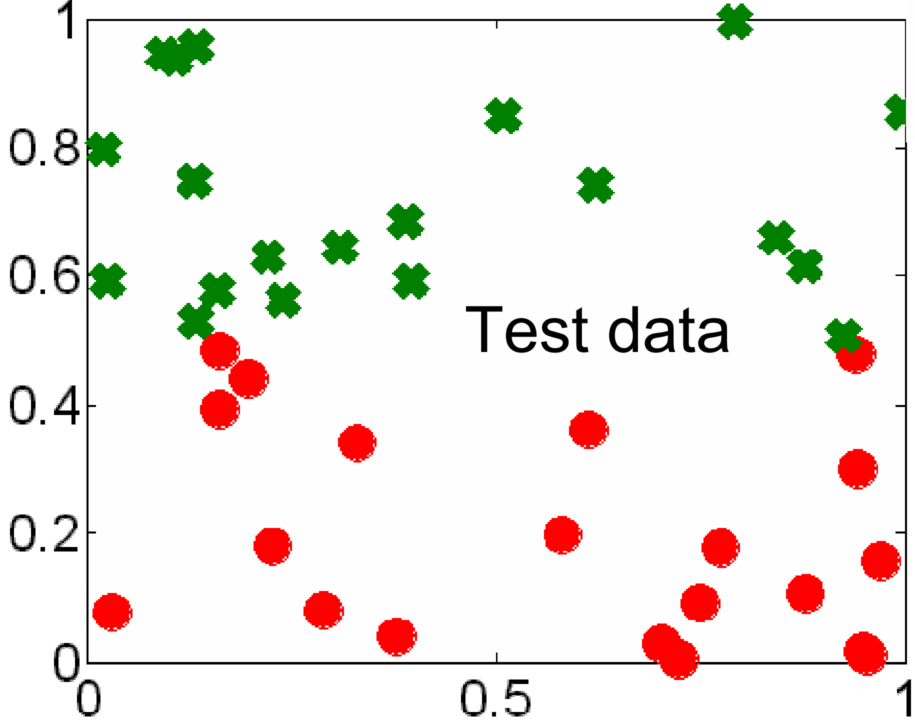


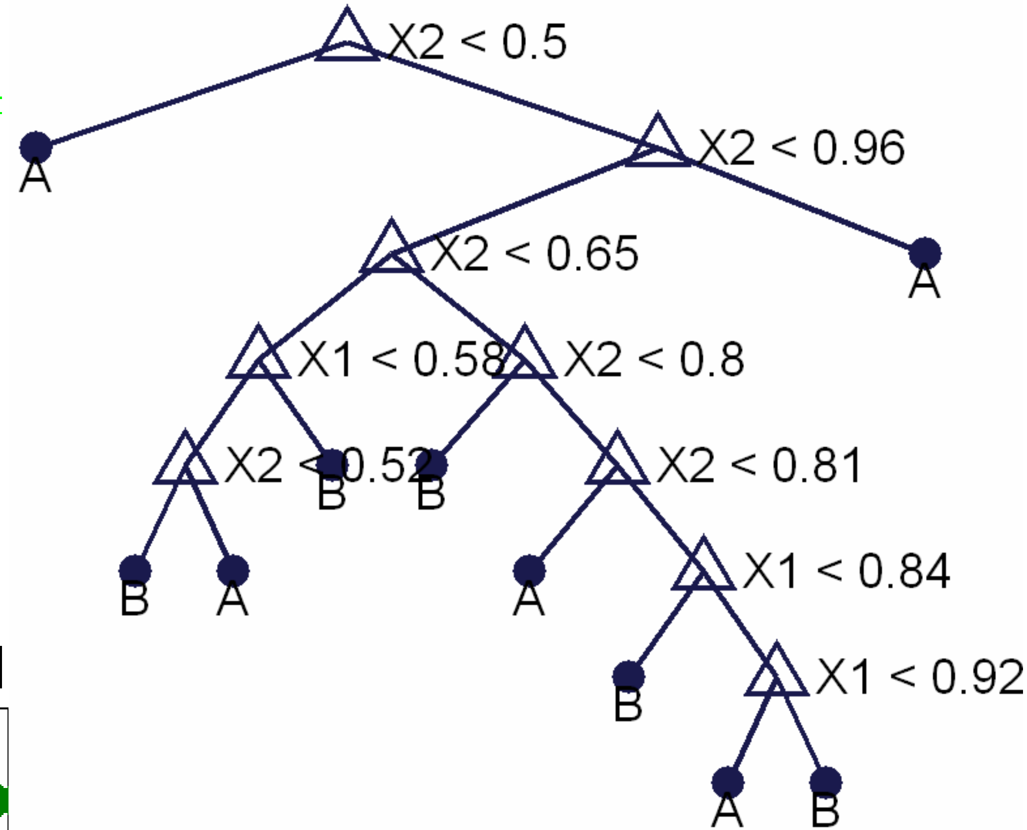
Training data
with the partitions induced by the decision tree
(Notice the tiny regions at the top necessary to correctly classify the ‘A’ outliers!)



Unpruned decision tree from training data





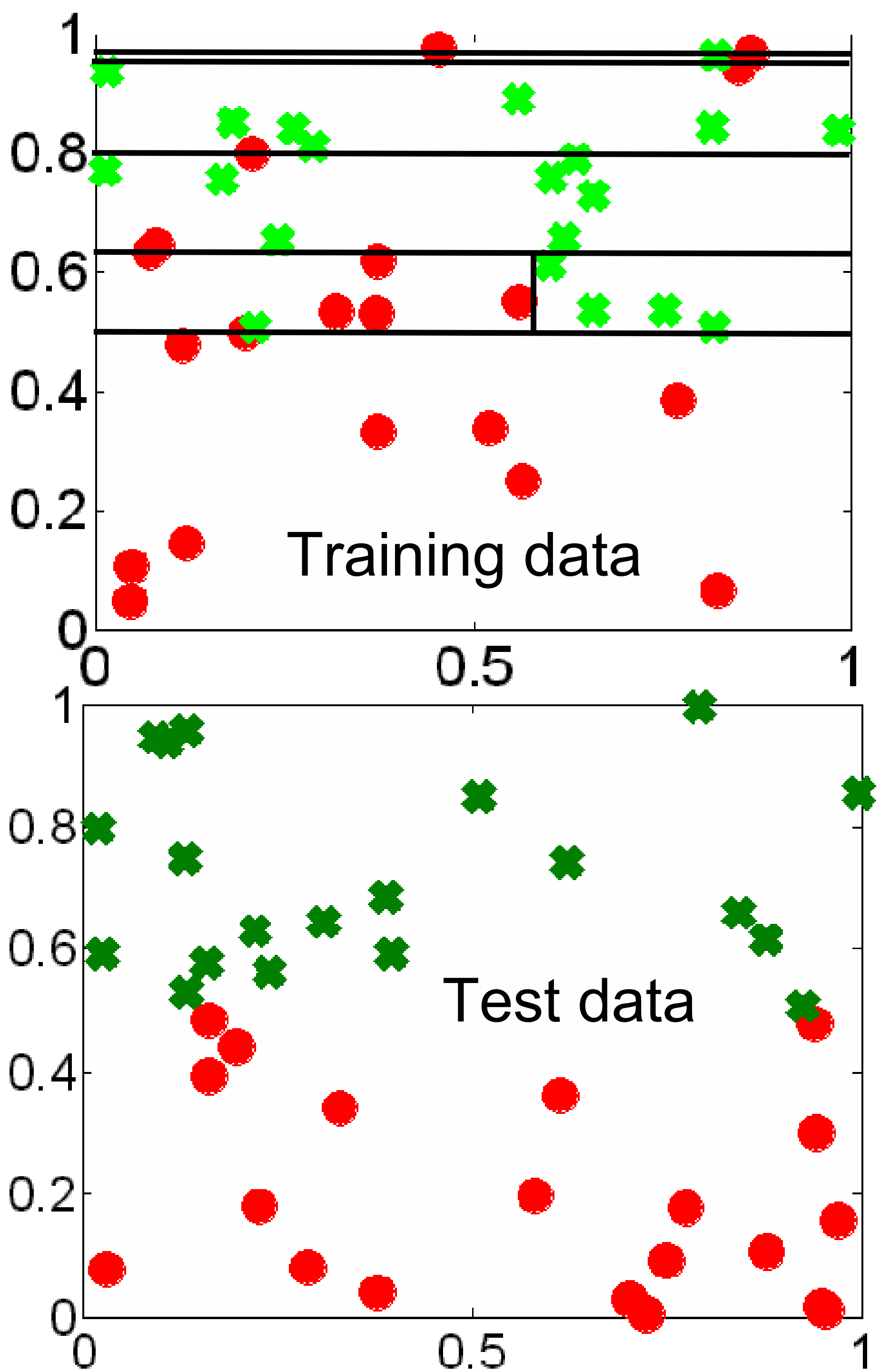


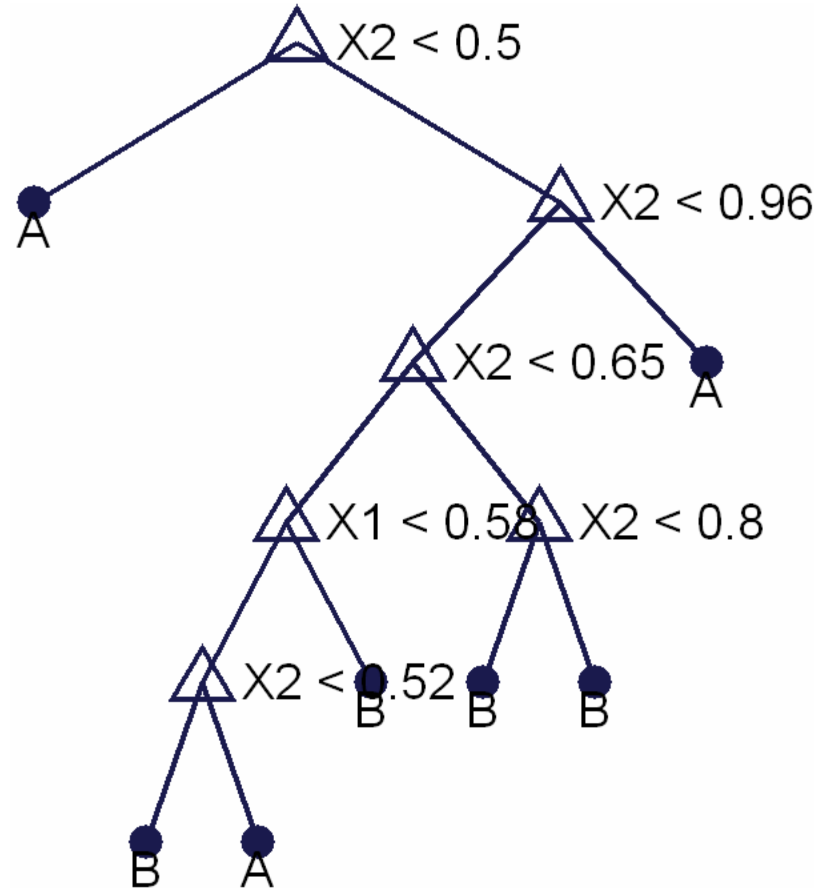
Unpruned decision tree from training data

Performance (% correctly classified)

Training: 100%

Test: 77.5%



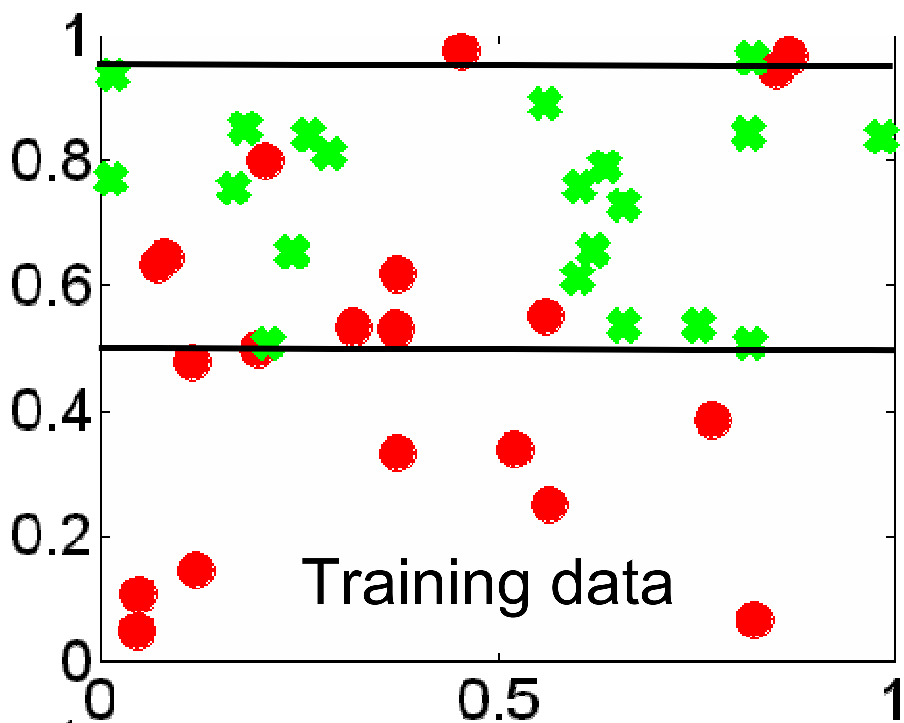


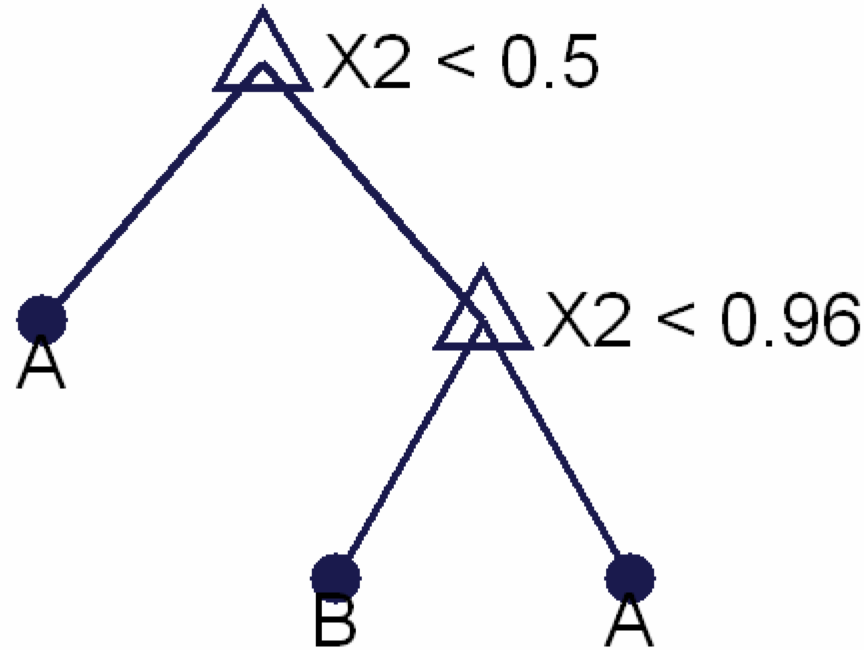
Pruned decision tree from training data

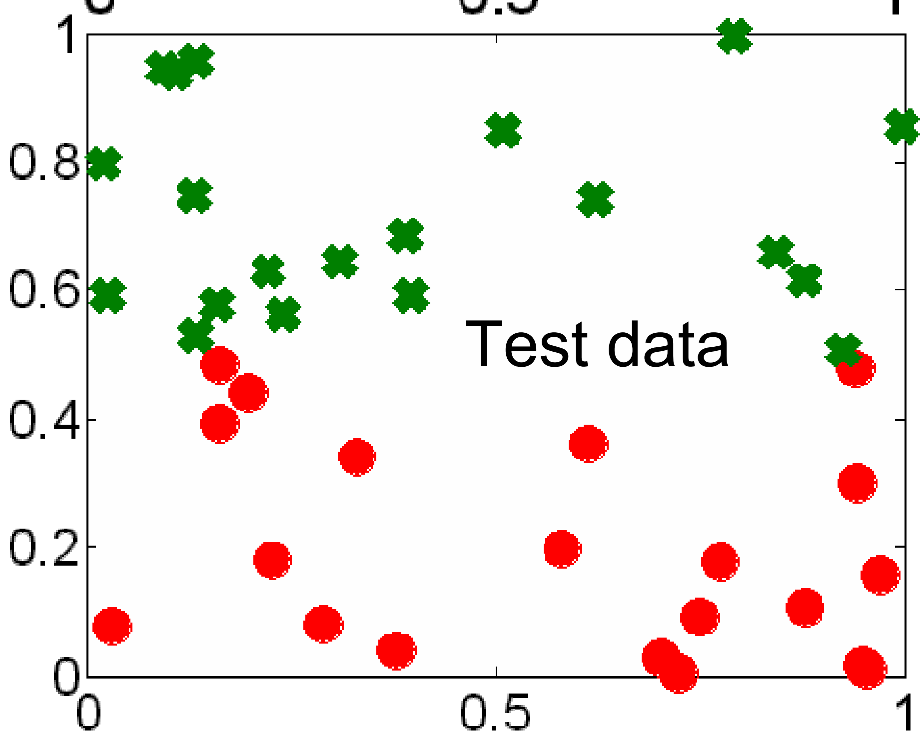
Performance (% correctly classified)

Training: 95%

Test: 80%





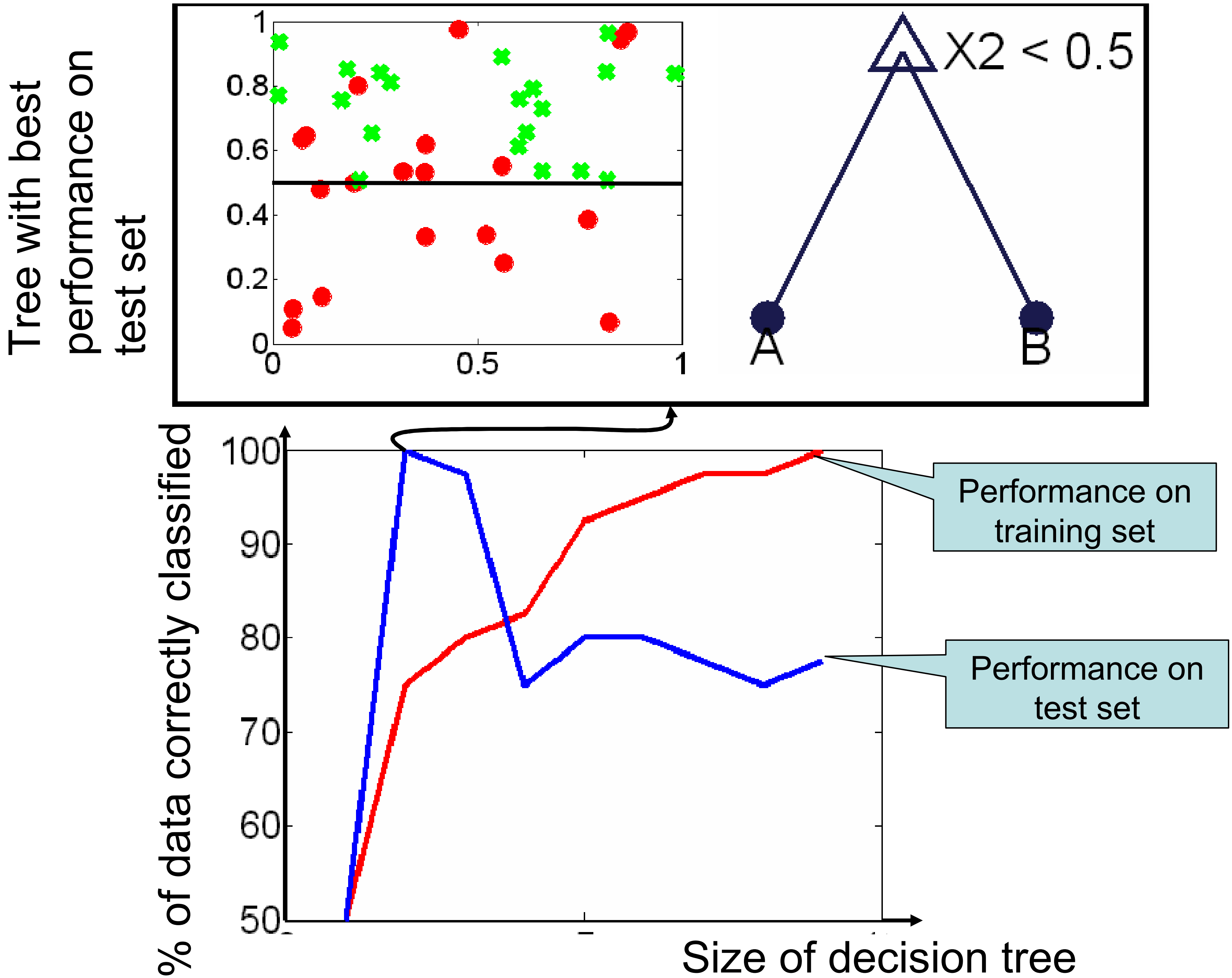


Pruned decision tree from training data

Performance (% correctly classified)

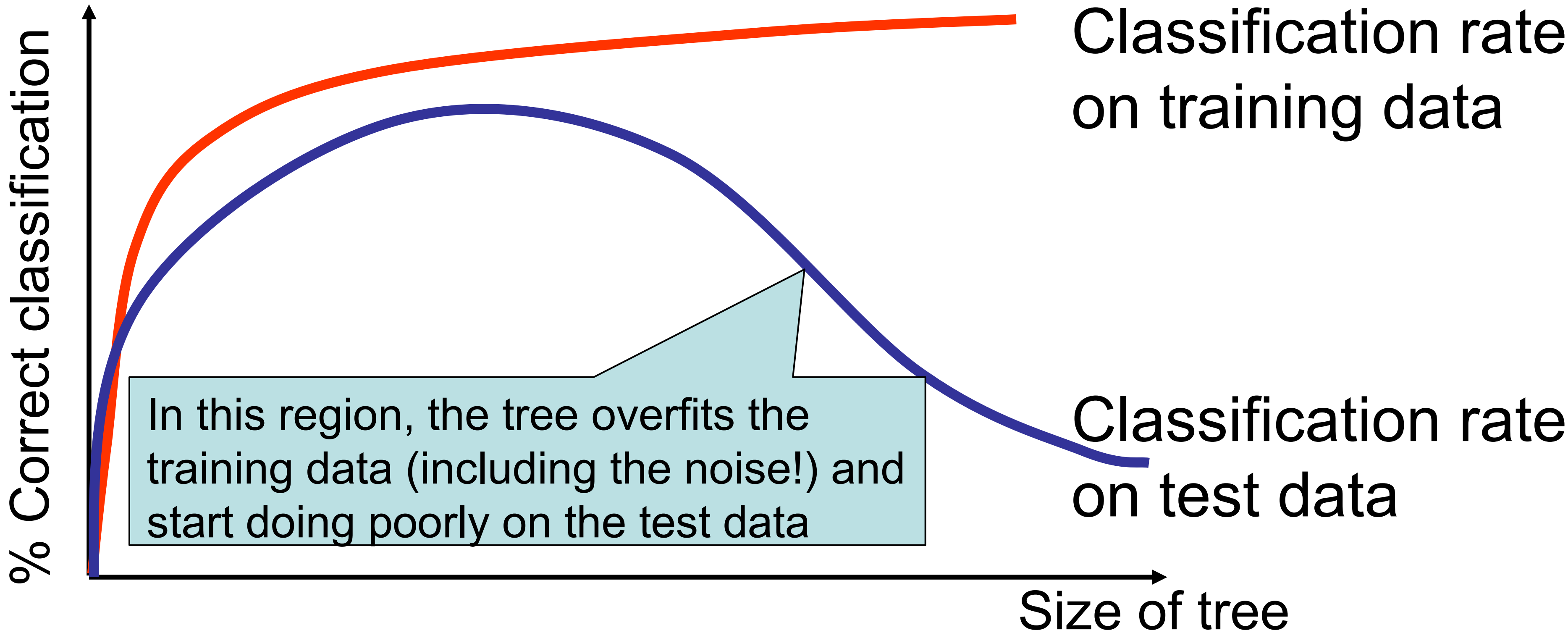
Training: 80%

Test: 97.5%

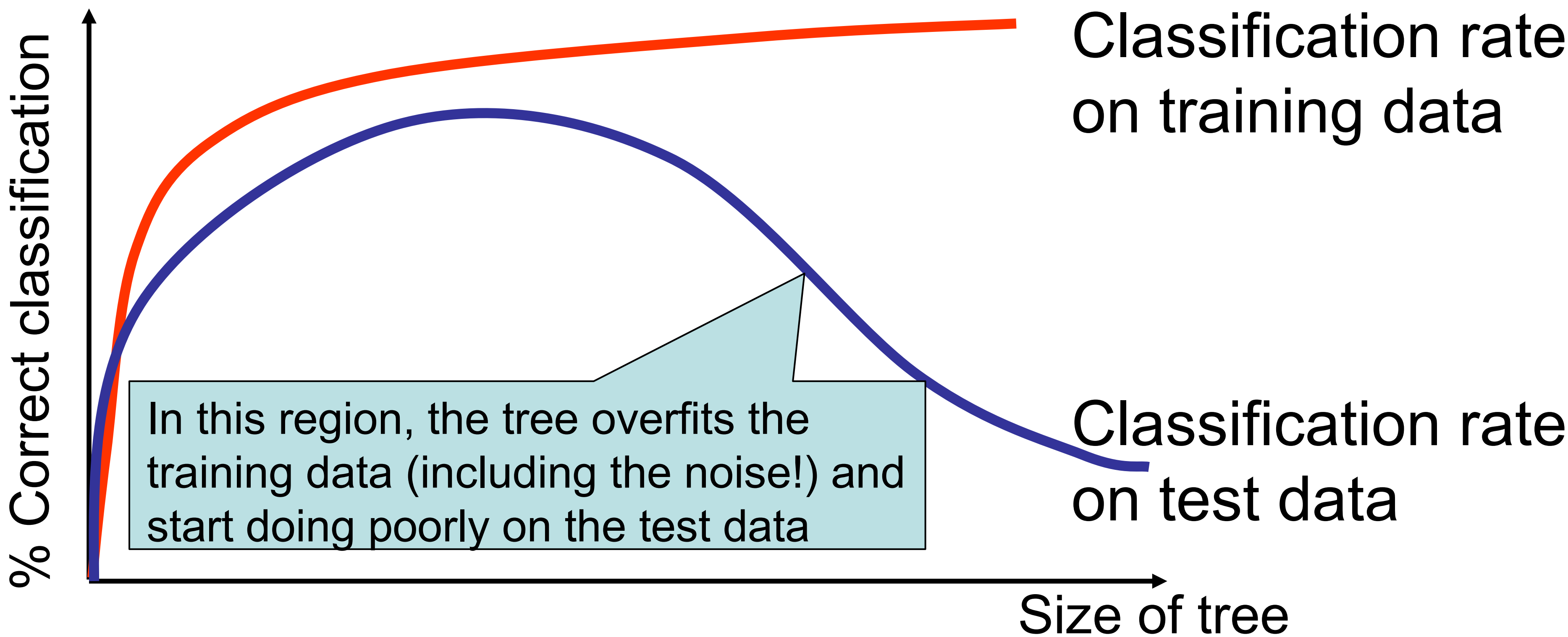


Tree with best performance on test set
X2 < 0.5
A
B
% of data correctly classified
Size of decision tree
Performance on training set
Performance on test set

Using Test Data



Using Test Data



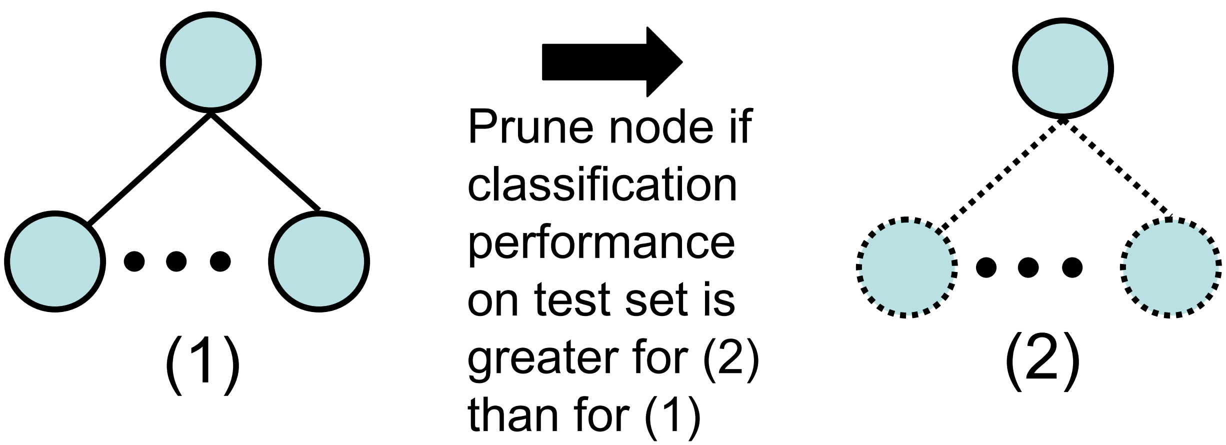
- General principle: As the complexity of the classifier increases (depth of the decision tree), the performance on the training data increases and the performance on the test data decreases when the classifier overfits the training data.

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?

Decision Tree Pruning

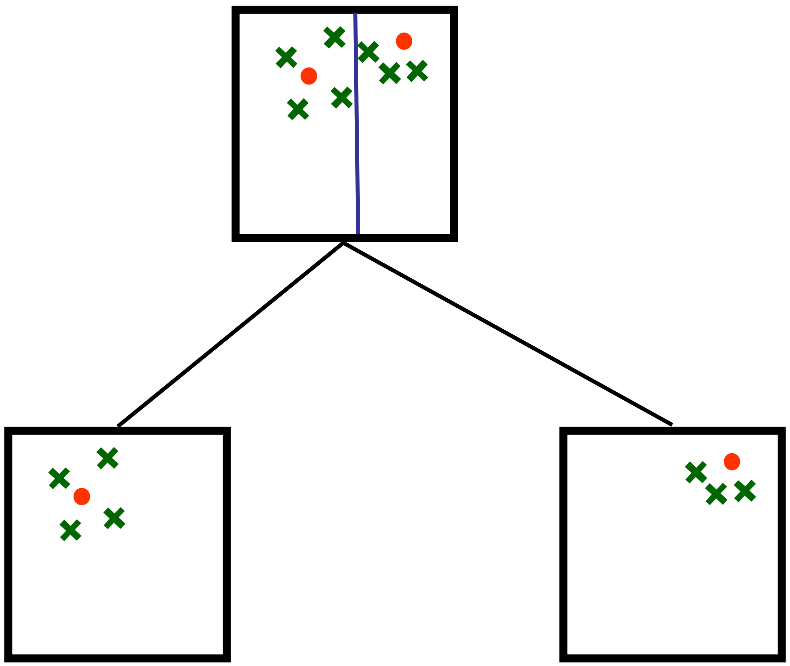
- Construct the entire tree as before
- Starting at the leaves, recursively eliminate splits:
  - Evaluate performance of the tree on test data (also called validation data, or hold out data set)
  - Prune the tree if the classification performance increases by removing the split



Possible Overfitting Solutions

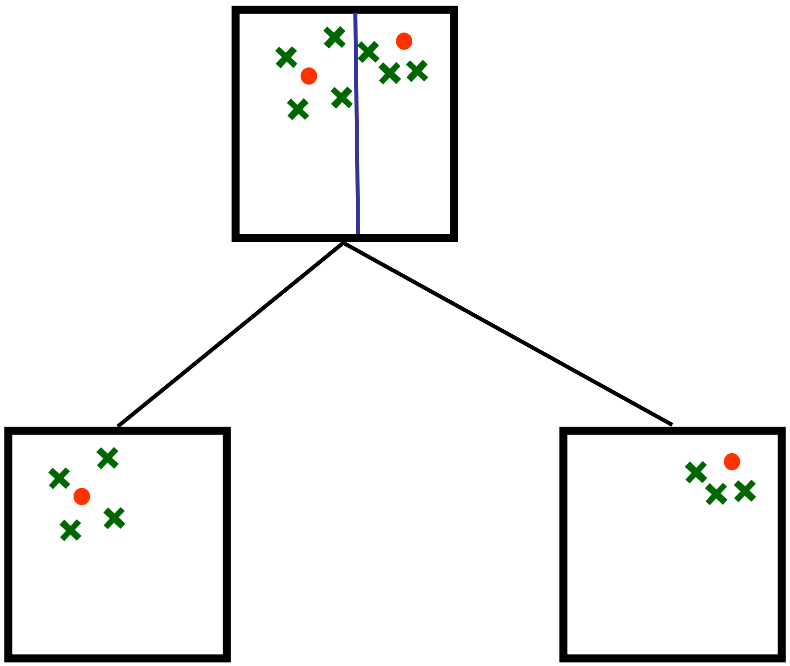
- Grow tree based on training data (*unpruned* tree)
- Prune the tree by removing useless nodes based on:
  - Additional test data (not used for training)
  - Statistical significance tests

A Criterion to Detect Useless Splits



- The problem is that we split whenever the IG increases, but we never check if the change in entropy is *statistically significant*

A Criterion to Detect Useless Splits

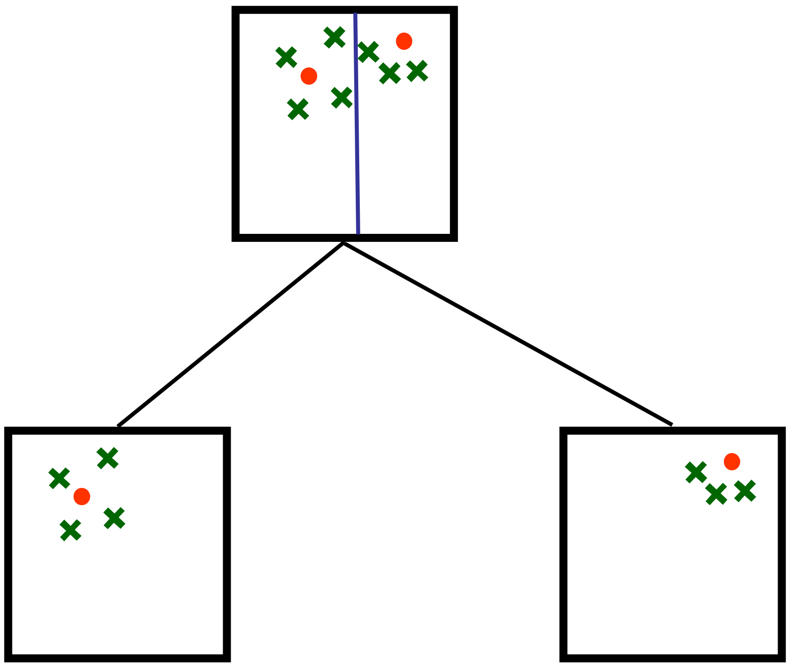


- The number of class A in the root node is $N_A = 2$
- The number of class B in the root node is $N_B = 7$

- The number of class A in the left node is $N_{AL} = 1$
- The number of class B in the left node is $N_{BL} = 4$

- The problem is that we split whenever the IG increases, but we never check if the change in entropy is *statistically significant*
- *Reasoning:*
- The proportion of the data going to the left node is $p_L = (N_{AL} + N_{BL})/(N_A+N_B) = 5/9$

A Criterion to Detect Useless Splits

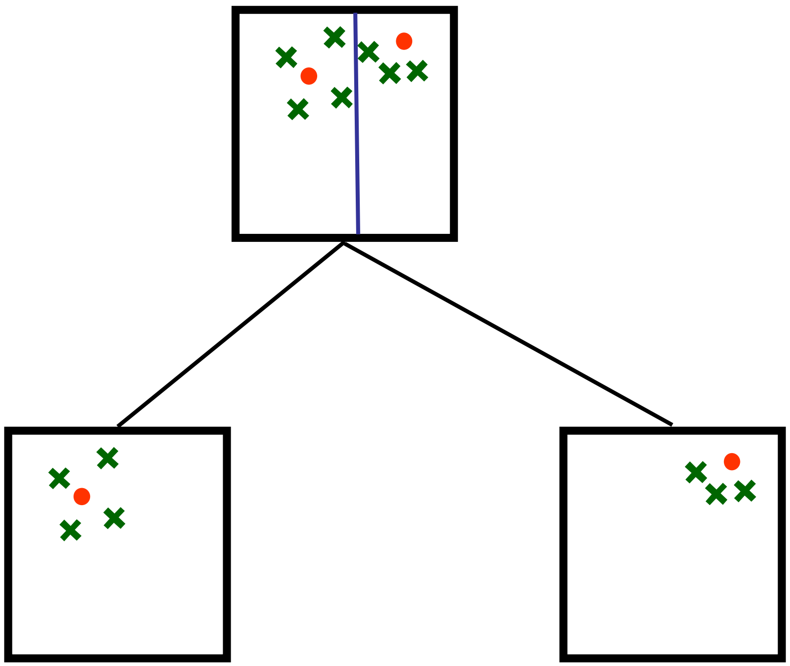


- The number of class A in the root node is $N_A = 2$
- The number of class B in the root node is $N_B = 7$

- The number of class A in the left node is $N_{AL} = 1$
- The number of class B in the left node is $N_{BL} = 4$

- The problem is that we split whenever the IG increases, but we never check if the change in entropy is *statistically significant*
- *Reasoning:*
- The proportion of the data going to the left node is $p_L = (N_{AL} + N_{BL})/(N_A+N_B) = 5/9$
- Suppose now that the data is *completely randomly* distributed (i.e., it does not make sense to split):
- The expected number of class A in the left node would be $N'_{AL} = N_A \times p_L = 10/9$
- The expected number of class B in the left node would be $N'_{BL} = N_B \times p_L = 35/9$

A Criterion to Detect Useless Splits



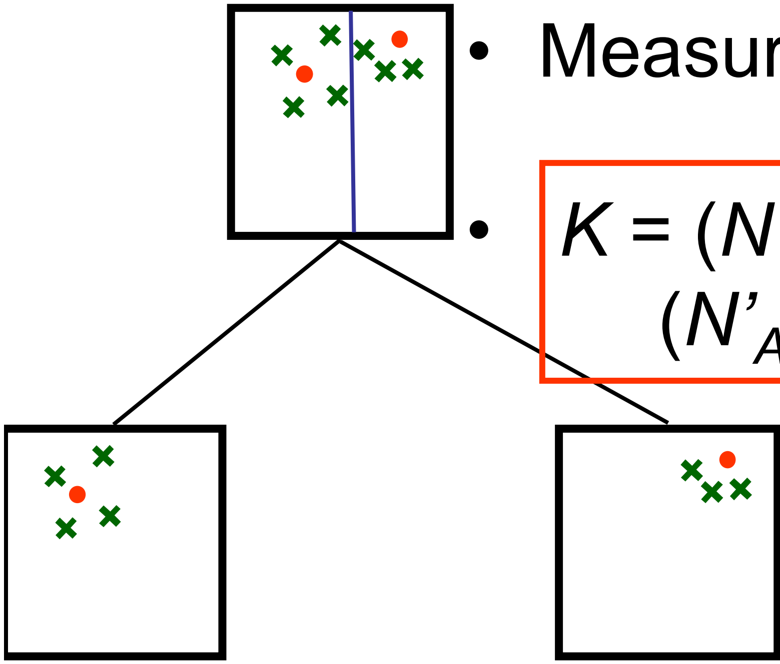
- The number of class A in the root node is $N_A = 2$
- The number of class B in the root node is $N_B = 7$

- The number of class A in the left node is $N_{AL} = 1$
- The number of class B in the left node is $N_{BL} = 4$

- The problem is that we split whenever the IG increases, but we never check if the change in entropy is *statistically significant*
- *Reasoning:*
- The proportion of the data going to the left node is $p_L = (N_{AL} + N_{BL})/(N_A+N_B) = 5/9$
- Suppose now that the data is *completely randomly* distributed (i.e., it does not make sense to split):
- The expected number of class A in the left node would be $N'_{AL} = N_A \times p_L = 10/9$
- The expected number of class B in the left node would be $N'_{BL} = N_B \times p_L = 35/9$

- *Question:*
- Are N_A and N_B sufficiently different from N'_A and N'_B. *If not, it means that the split is not statistically significant and we should not split the root* → The resulting children are not significantly different from what we would get by splitting a random distribution at the root node.

A Criterion to Detect Useless Splits



- Measure of statistically significance:
- $K = (N'_{AL} - N_{AL})^2/N'_{AL} + (N'_{BL} - N_{BL})^2/N'_{BL} + (N'_{AR} - N_{AR})^2/N'_{AR} + (N'_{BR} - N_{BR})^2/N'_{BR}$

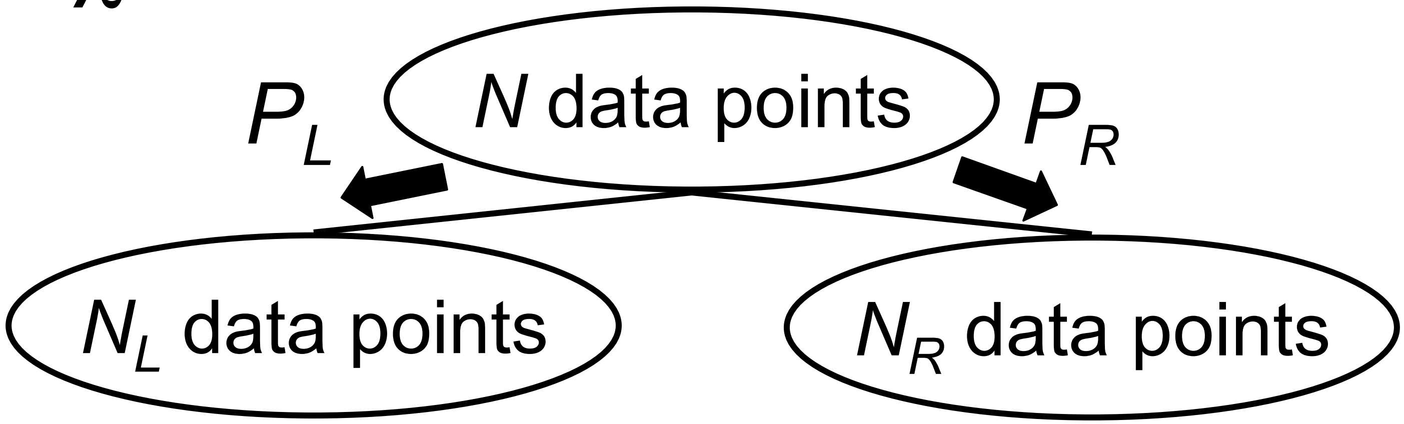
K measures how much the split deviates from what we would get if the data where random

K small → The increase in IG of the split is not significant

In this example (primes are expected): $K =$

$$(10/9 - 1)^2/(10/9) + (35/9 - 4)^2/(35/9) + \ldots = 0.0321$$

χ^2 Criterion: General Case



$$K = \sum_{\substack{\text{all classes } i \\ \text{children } j}} \frac{(N_{ij} - N'_{ij})^2}{N'_{ij}}$$

- N_{ij} = Number of points from class i in child j
- N'_{ij} = Number of points from class i in child j *assuming a random selection*
- $N'_{ij} = N_i \times P_j$

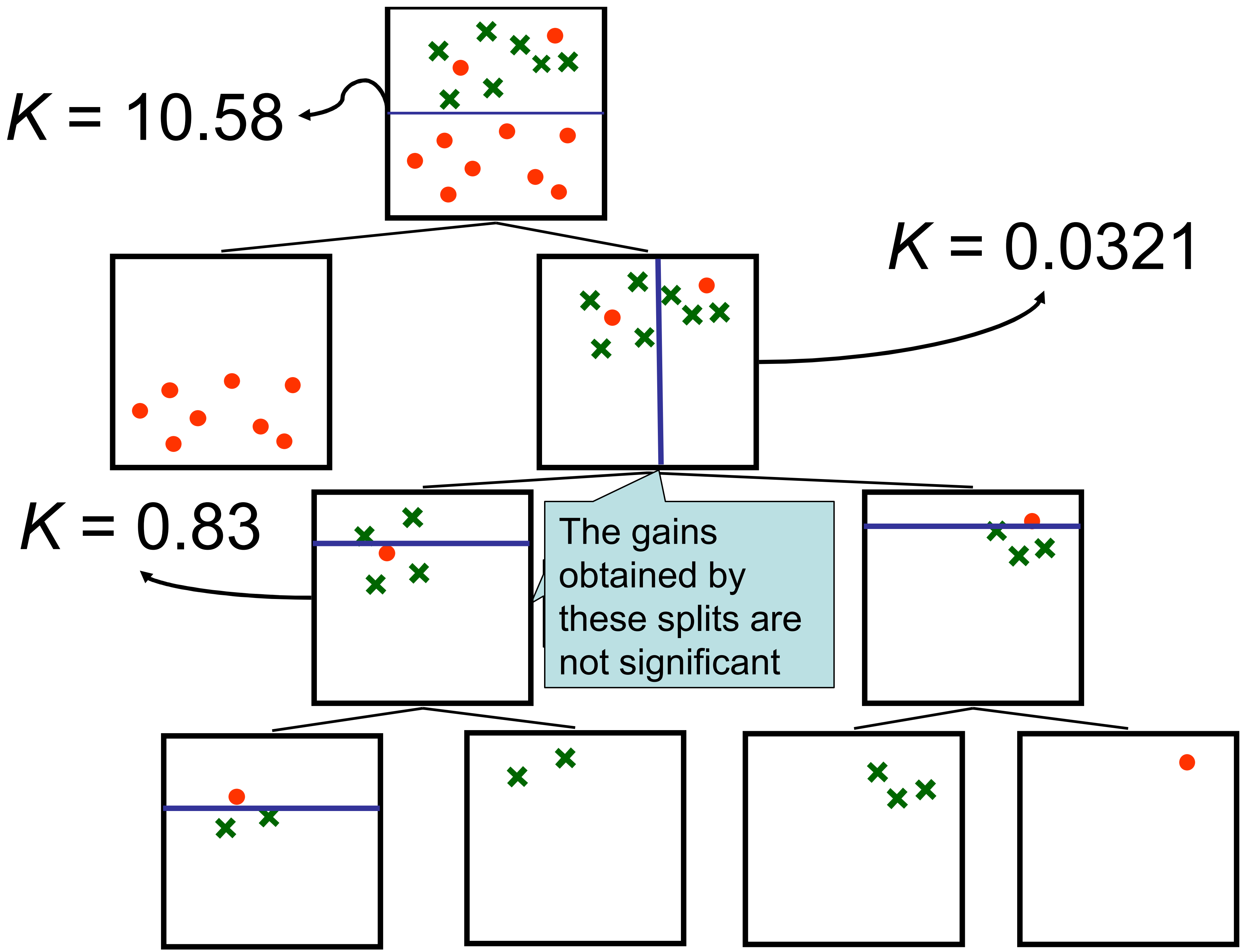
Small (Chi-square) values indicate low statistical significance → Remove the splits that are lower than a threshold $K < t$.
Lower t → bigger trees (more overfitting).
Larger t → smaller trees (less overfitting, but worse classification error).

Difference between the distribution of class i from the proposed split and the distribution from randomly drawing data points in the same proportions as the proposed split

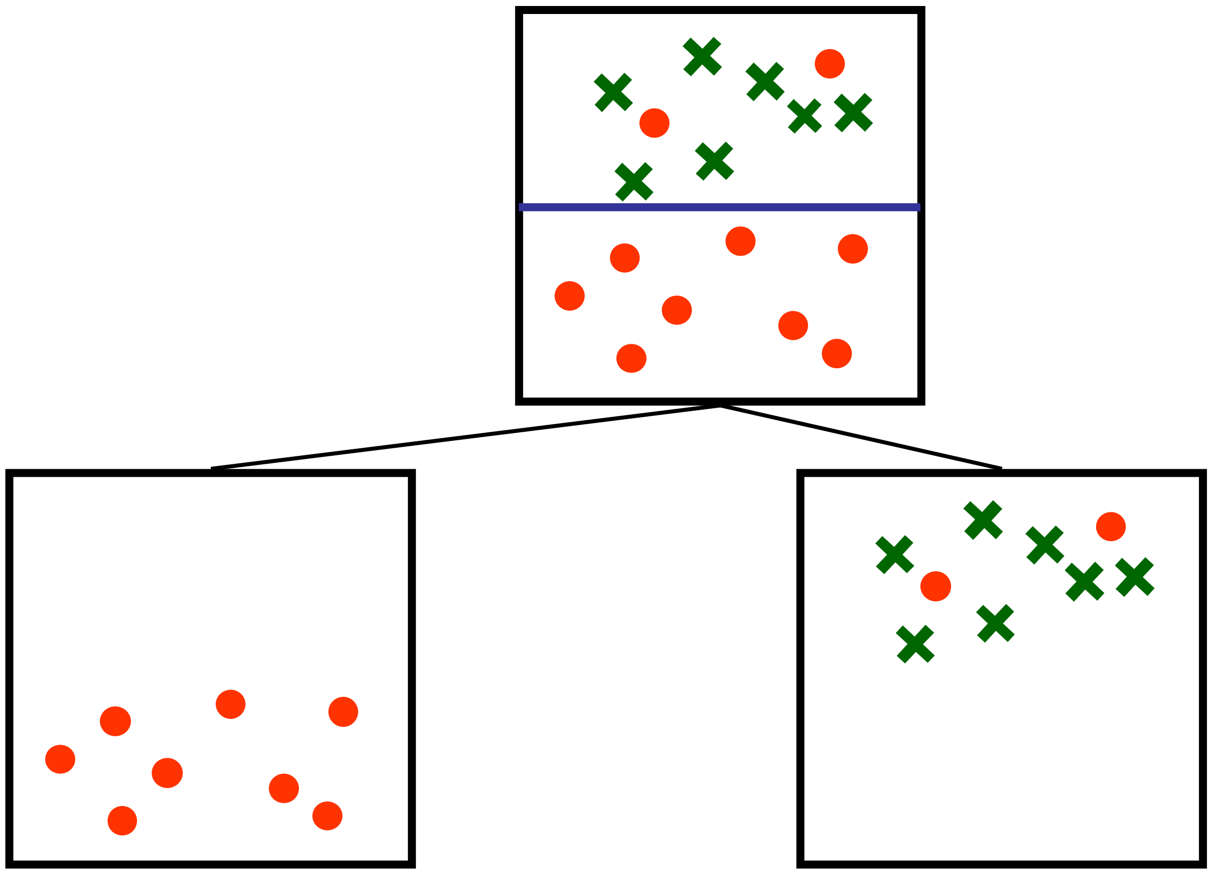
$$K = \sum_{\substack{\text{all classes } i \\ \text{children } j}} \frac{(N_{ij} - N'_{ij})^2}{N'_{ij}}$$

Decision Tree Pruning

- Construct the entire tree as before
- Starting at the leaves, recursively eliminate splits:
  - At a leaf $\mathcal{N}$:
    - Compute the *K* value for $\mathcal{N}$ and its parent $\mathcal{P}$.
    - If the *K* values is lower than the threshold *t*:
      - Eliminate all of the children of $\mathcal{P}$
      - $\mathcal{P}$ becomes a leaf
  - Repeat until no more splits can be eliminated



$K = 10.58$
$K = 0.0321$
$K = 0.83$
The gains obtained by these splits are not significant



- By thresholding *K* we end up with the decision tree that we would expect (i.e., one that does not overfit the data)
- Note: The approach is presented with continuous attributes in this example but it works just as well with discrete attributes.

χ^2 Pruning

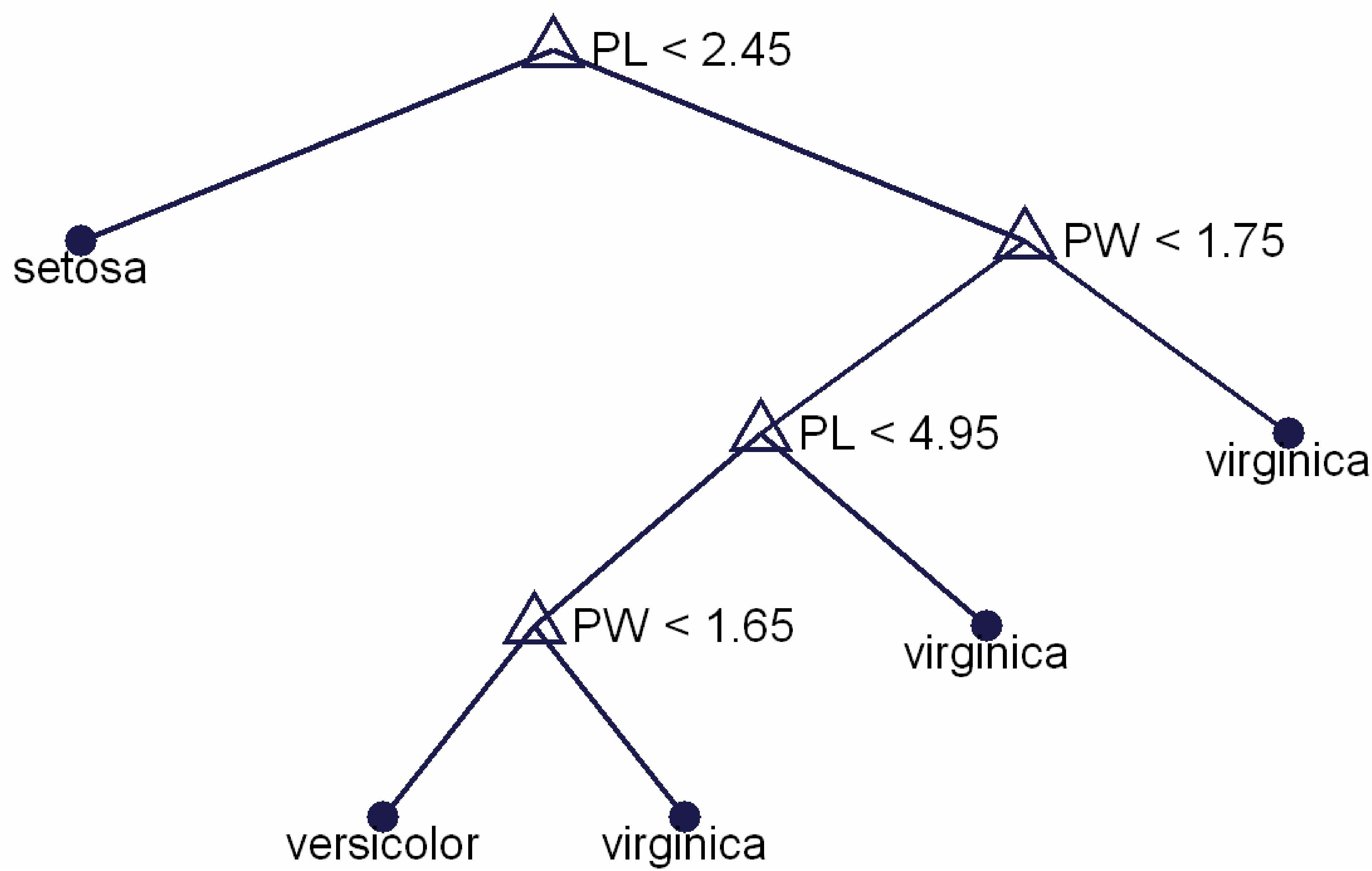
- The test on *K* is a version of a standard statistical test, the χ^2 ('chi-square') test.
- The value of *t* is retrieved from statistical tables. For example, $K > t$ means that, with confidence 95%, the information gain due to the split is significant.
- If $K < t$, with high confidence, the information gain will be 0 over very large training samples
  - Reduces *overfitting*
  - Eliminates *irrelevant attributes*

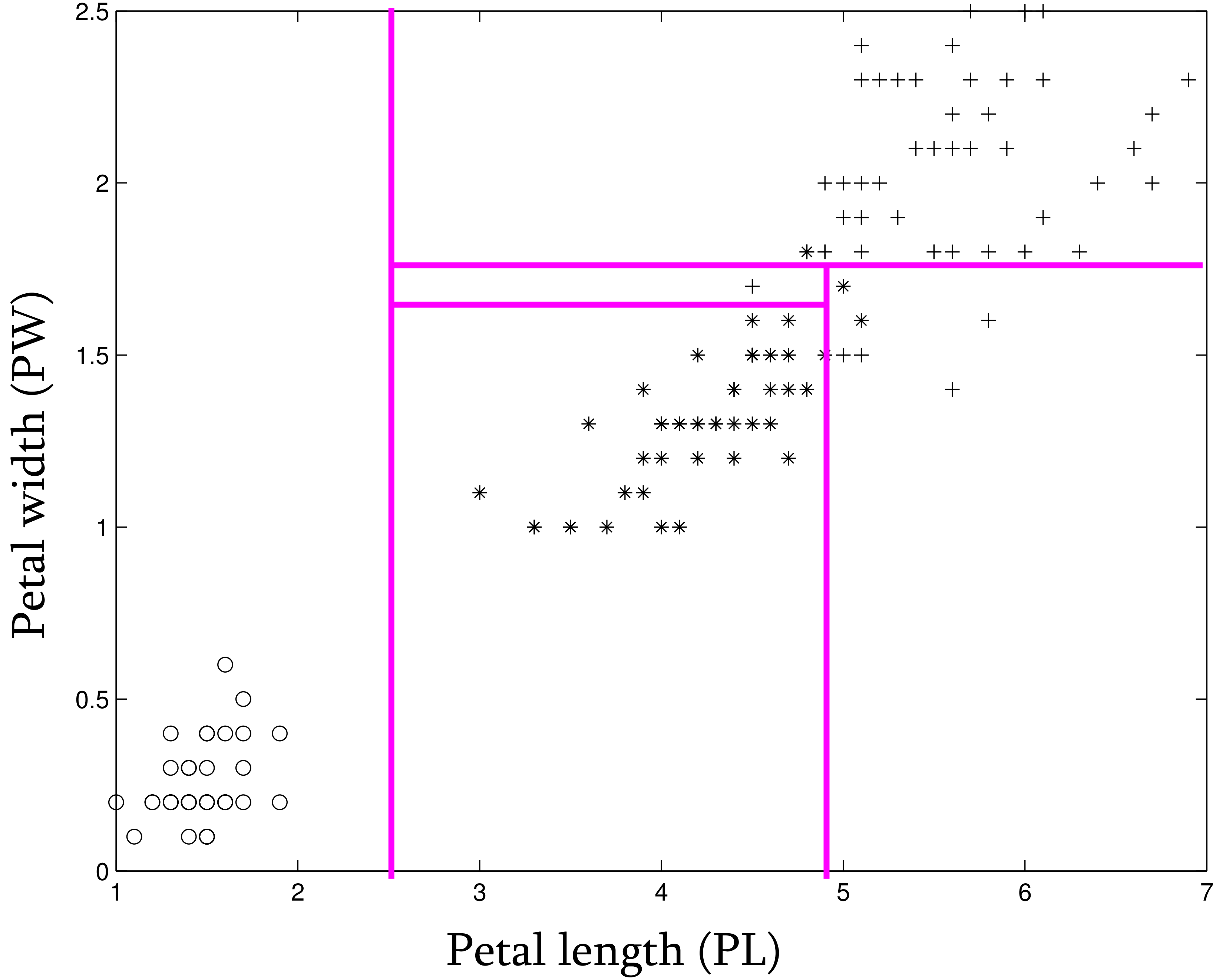
Example

Class	Sepal Length (SL)	Sepal Width (SW)	Petal Length (PL)	Petal Width (PW)
Setosa	5.1	3.5	1.4	0.2
Setosa	4.9	3	1.4	0.2
Setosa	5.4	3.9	1.7	0.4
Versicolor	5.2	2.7	3.9	1.4
Versicolor	5	2	3.5	1
Versicolor	6	2.2	4	1
Virginica	6.4	2.8	5.6	2.1
Virginica	7.2	3	5.8	1.6

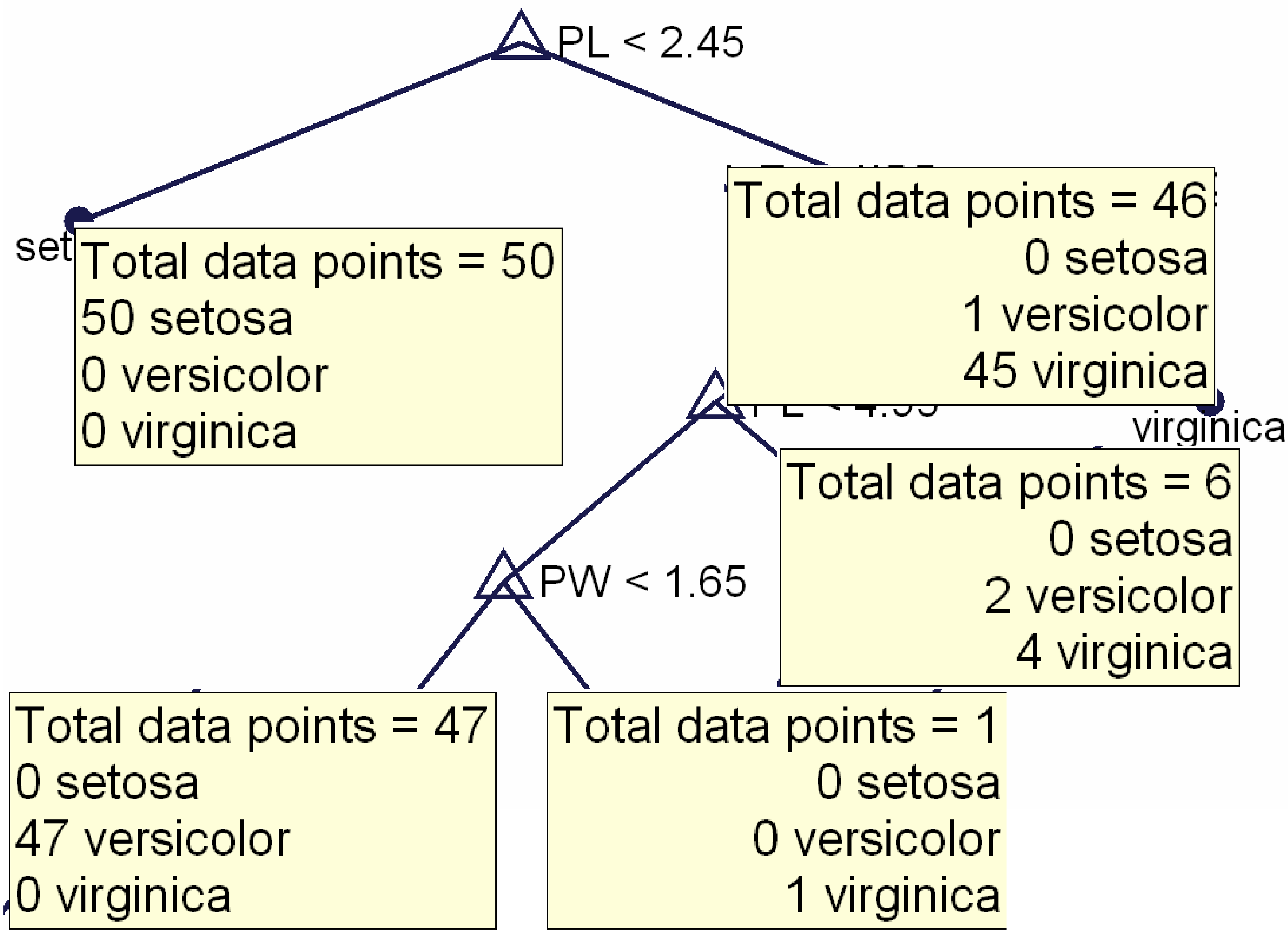
• • • • • 50 examples from each class • • • • •

Full Decision Tree

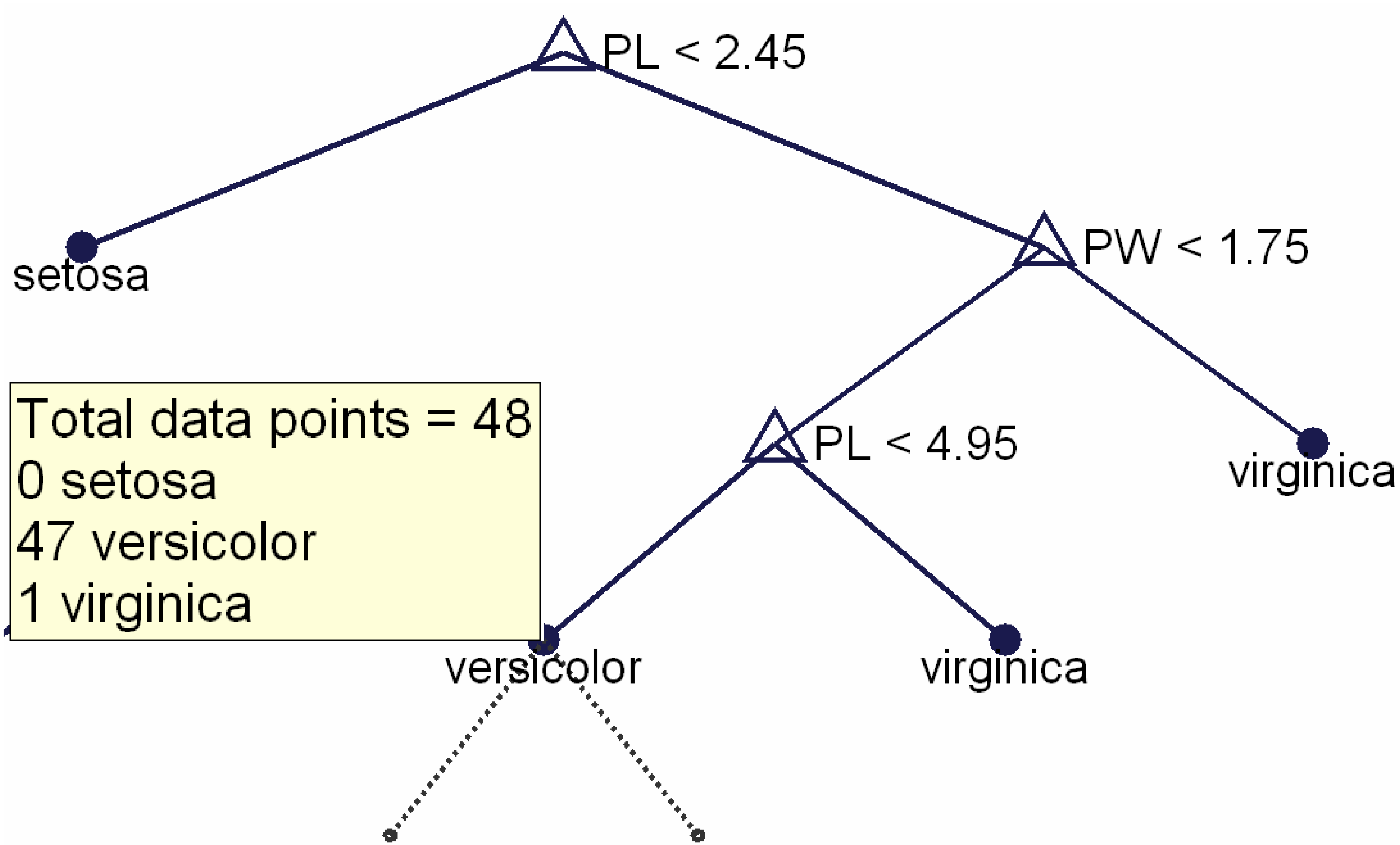




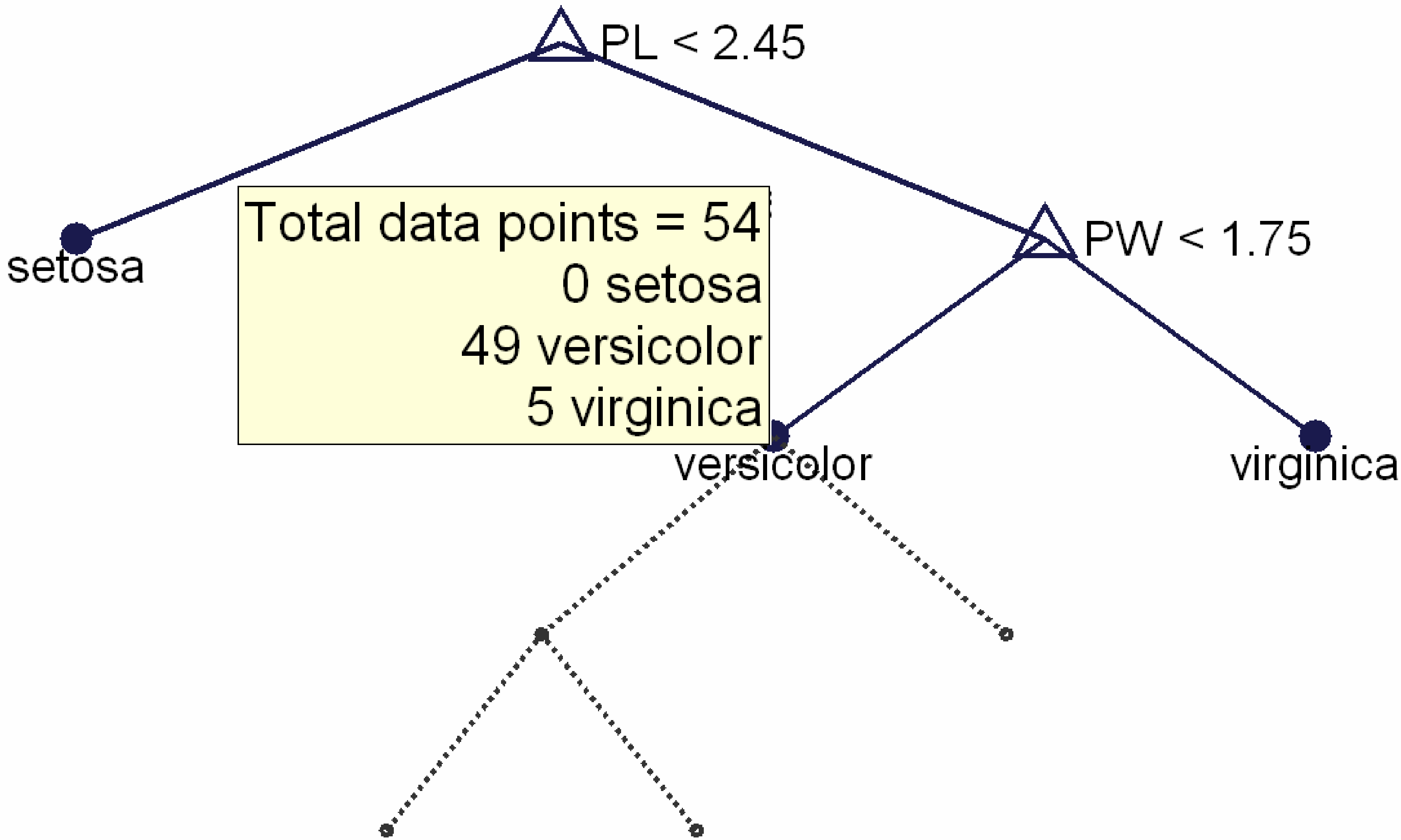
Petal width (PW)
Petal length (PL)

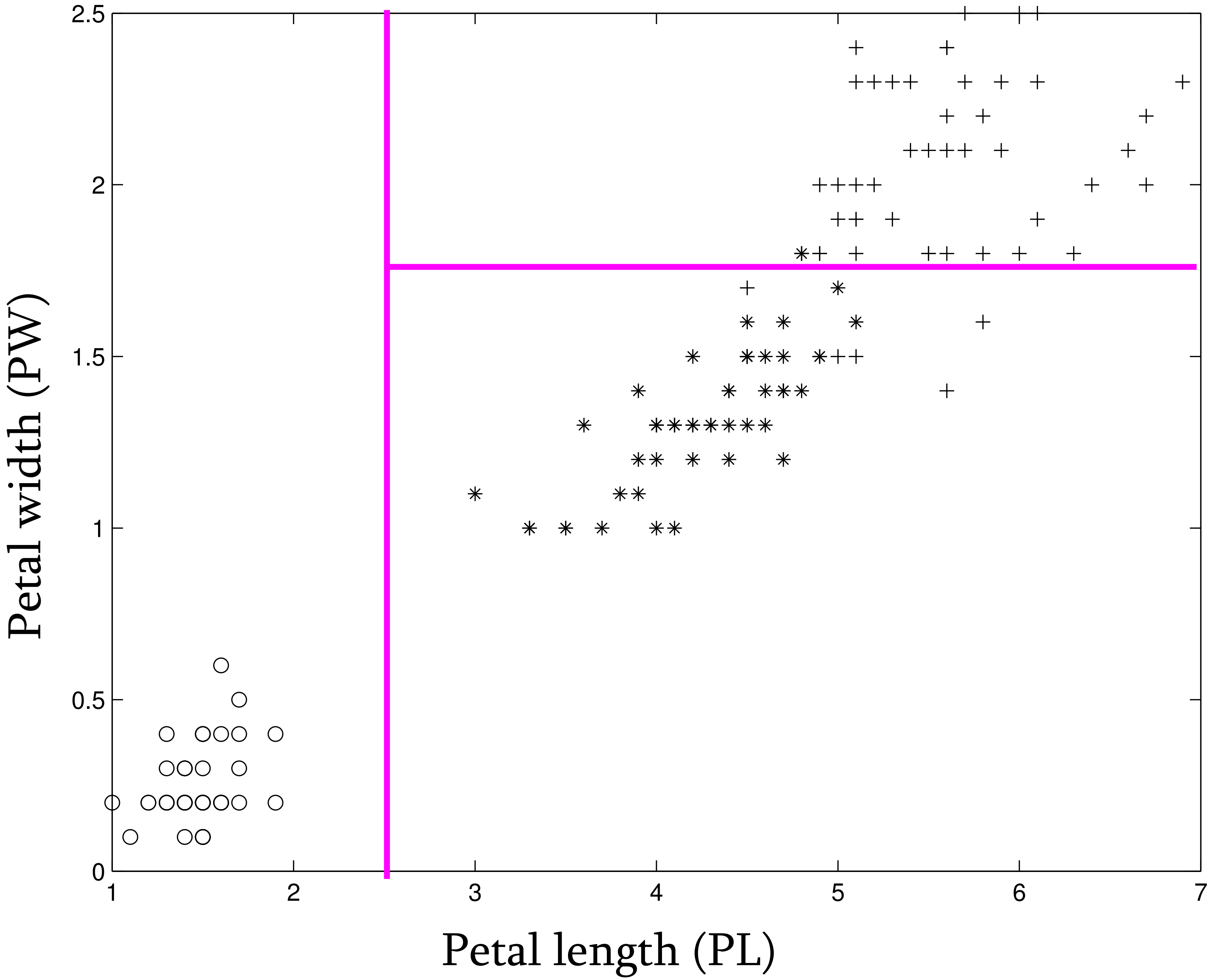

PL < 2.45
set
Total data points = 50
50 setosa
0 versicolor
0 virginica
Total data points = 46
0 setosa
1 versicolor
45 virginica
virginica
Total data points = 6
0 setosa
2 versicolor
4 virginica
PW < 1.65
Total data points = 47
0 setosa
47 versicolor
0 virginica
Total data points = 1
0 setosa
0 versicolor
1 virginica

Pruning One Level

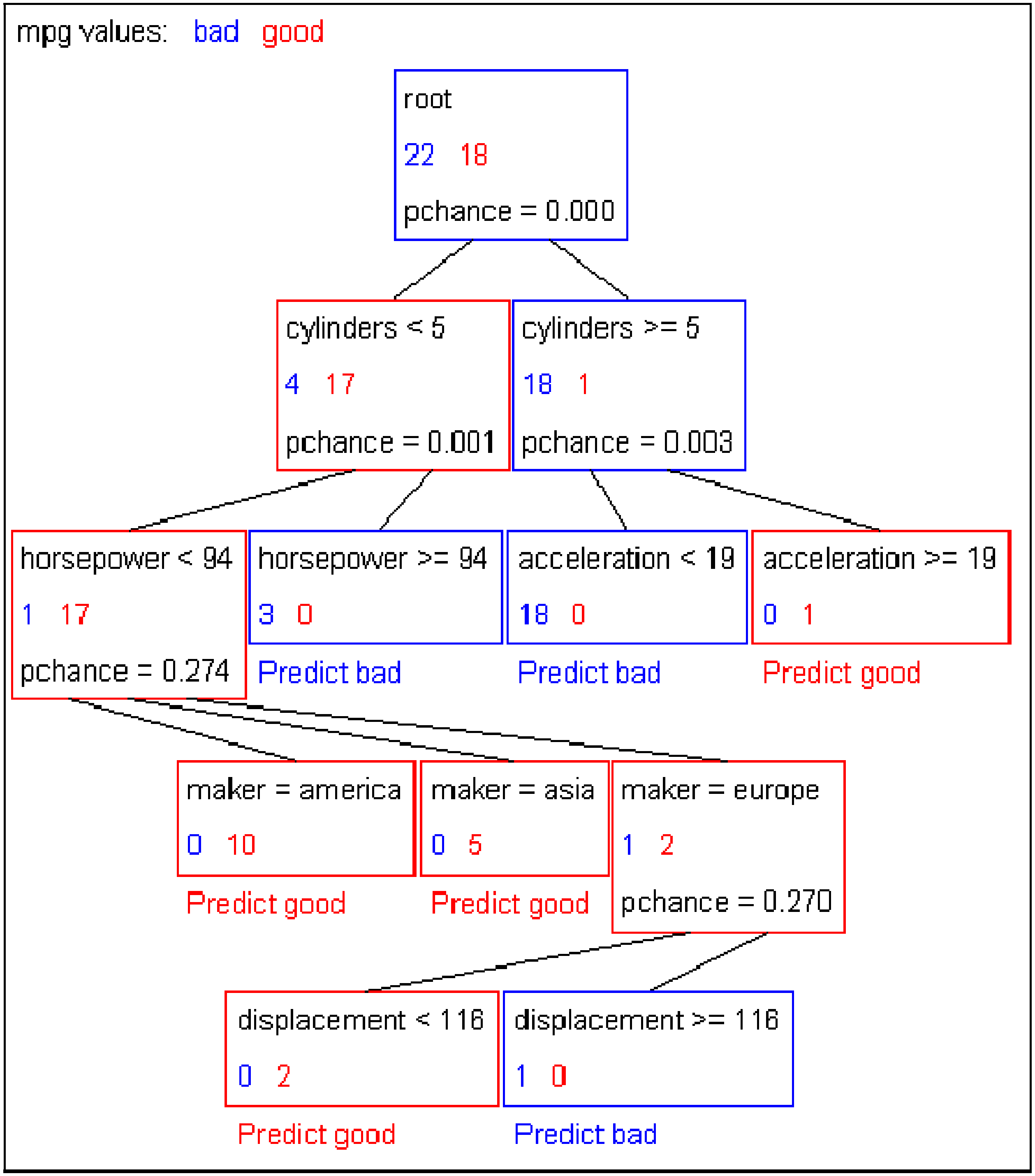


Pruning Two Levels



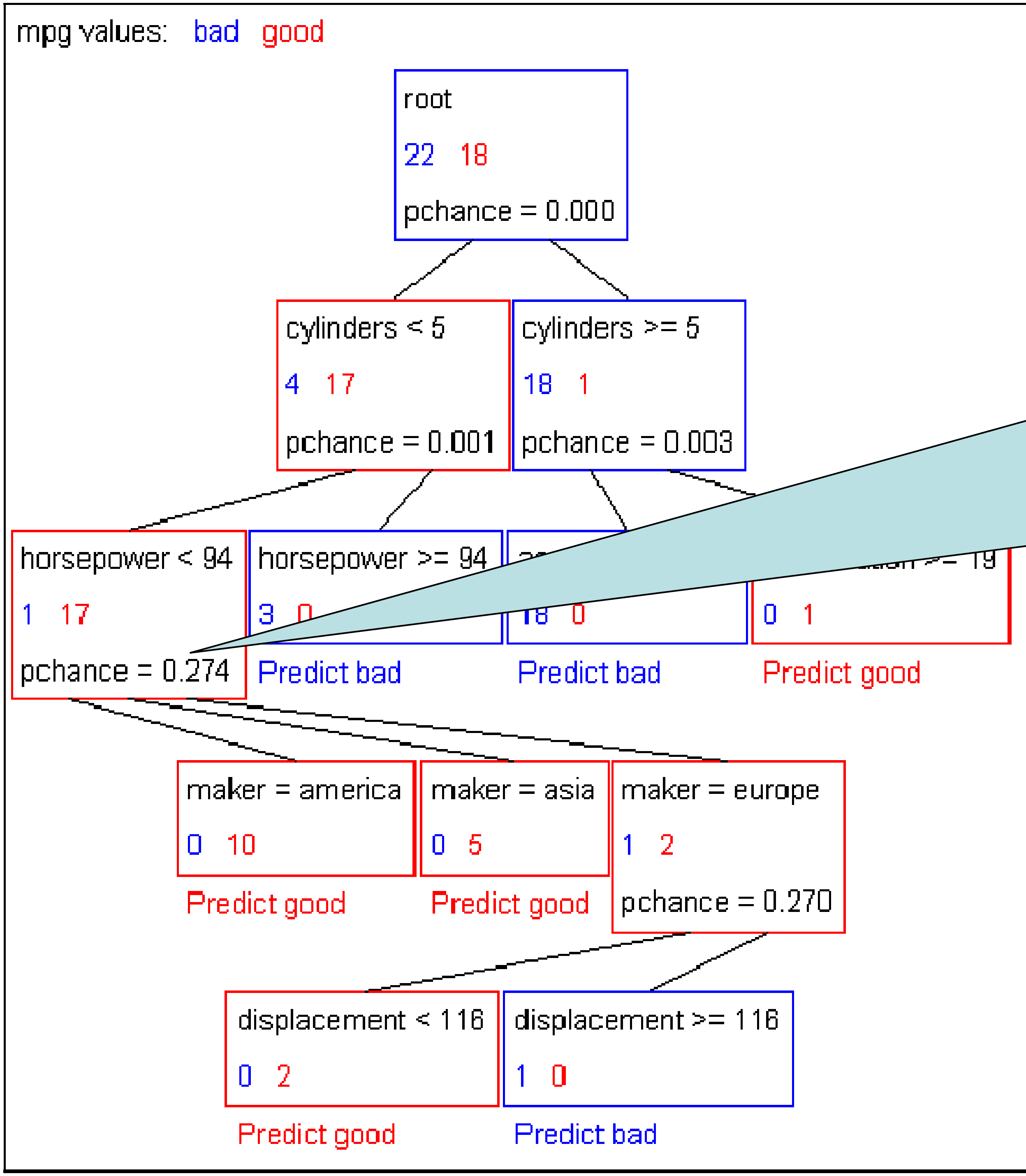

Petal width (PW)
Petal length (PL)
0
0.5
1
1.5
2
2.5
1
2
3
4
5
6
7

Unpruned



mpg values: bad good
root
22 18
pchance = 0.000
cylinders < 5
4 17
pchance = 0.001
cylinders >= 5
18 1
pchance = 0.003
horsepower < 94
1 17
pchance = 0.274
horsepower >= 94
3 0
Predict bad
acceleration < 19
18 0
Predict bad
acceleration >= 19
0 1
Predict good
maker = america
0 10
Predict good
maker = asia
0 5
Predict good
maker = europe
1 2
pchance = 0.270
displacement < 116
0 2
Predict good
displacement >= 116
1 0
Predict bad

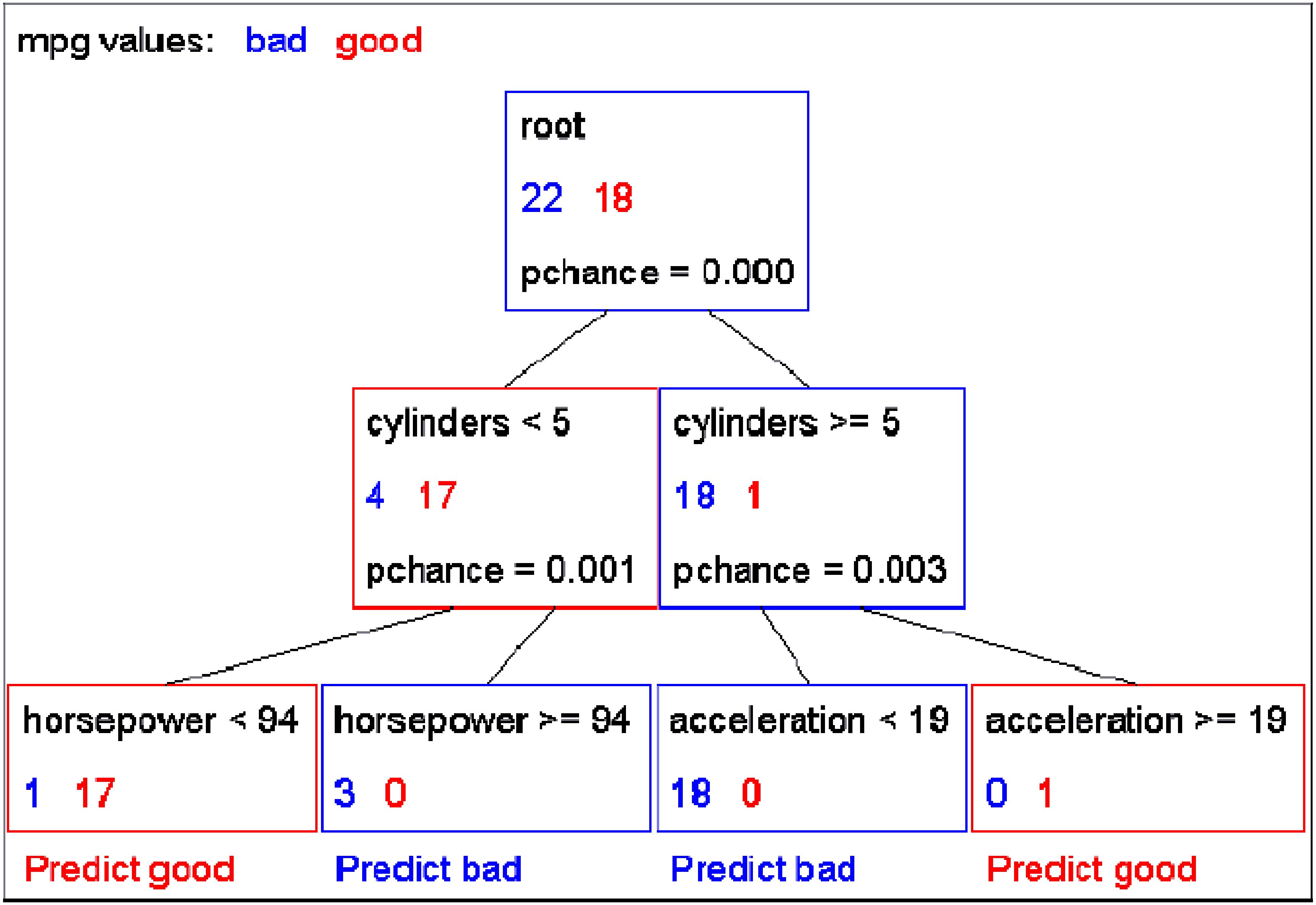
Unpruned



27% probability that this is a “chance” node according to χ^2 test.

Node should be pruned.

Pruned



mpg values: bad good
root
22 18
pchance = 0.000
cylinders < 5
4 17
pchance = 0.001
cylinders >= 5
18 1
pchance = 0.003
horsepower < 94
1 17
horsepower >= 94
3 0
acceleration < 19
18 0
acceleration >= 19
0 1
Predict good
Predict bad
Predict bad
Predict good

Decision Trees

- Information Gain (IG) criterion for choosing splitting criteria at each level of the tree.
- Versions with continuous attributes and with discrete (categorical) attributes
- Basic tree learning algorithm leads to overfitting of the training data
- Pruning with:
  - Additional test data (not used for training)
  - Statistical significance tests
- Example of inductive learning

But what if…

We don’t have labels / classes ?

But what if…

We don't have labels / classes ?

Everything so far has been "supervised" learning. The algorithms get to see class labels.

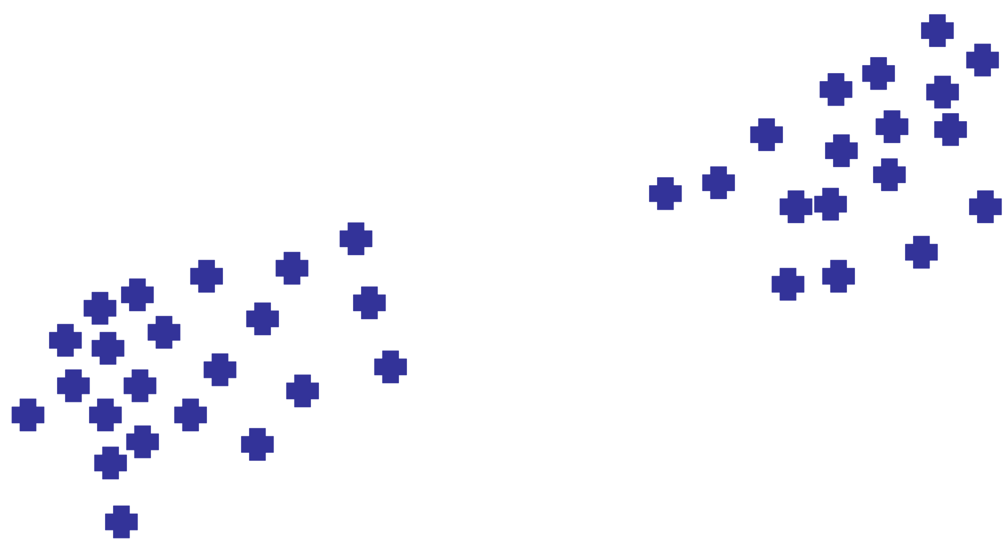
But what if…

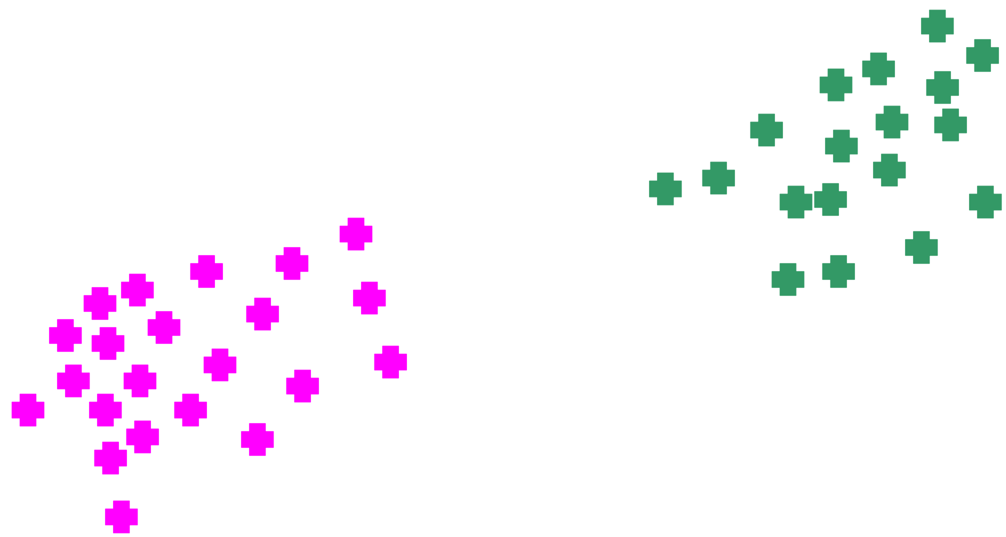
We don’t have labels / classes ?

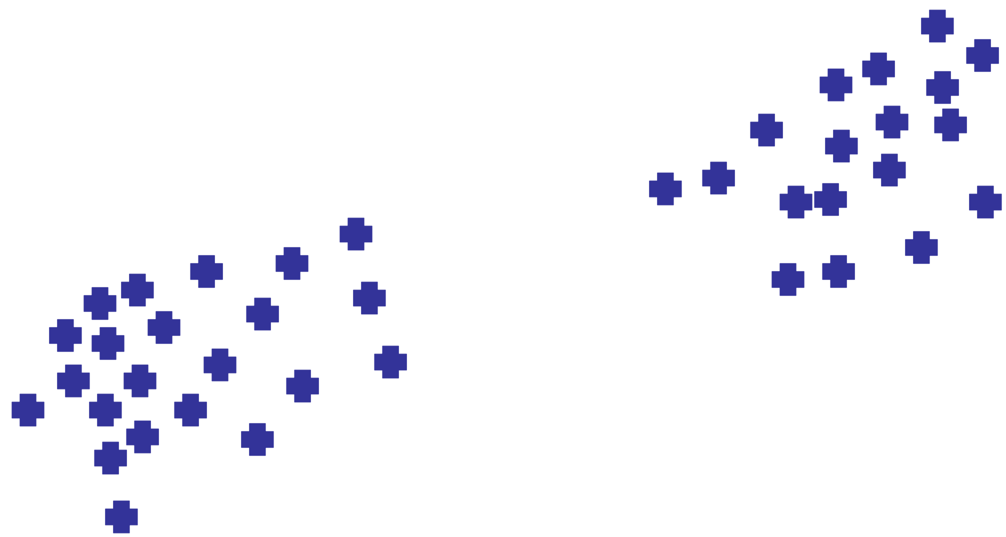
Everything so far has been “supervised” learning. The algorithms get to see class labels.

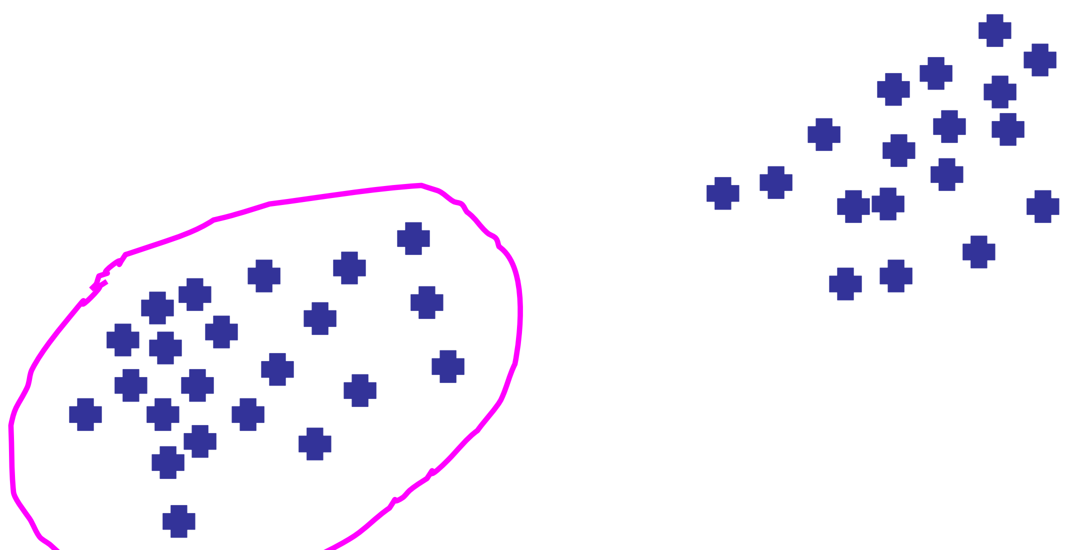
We can do unsupervised learning too!

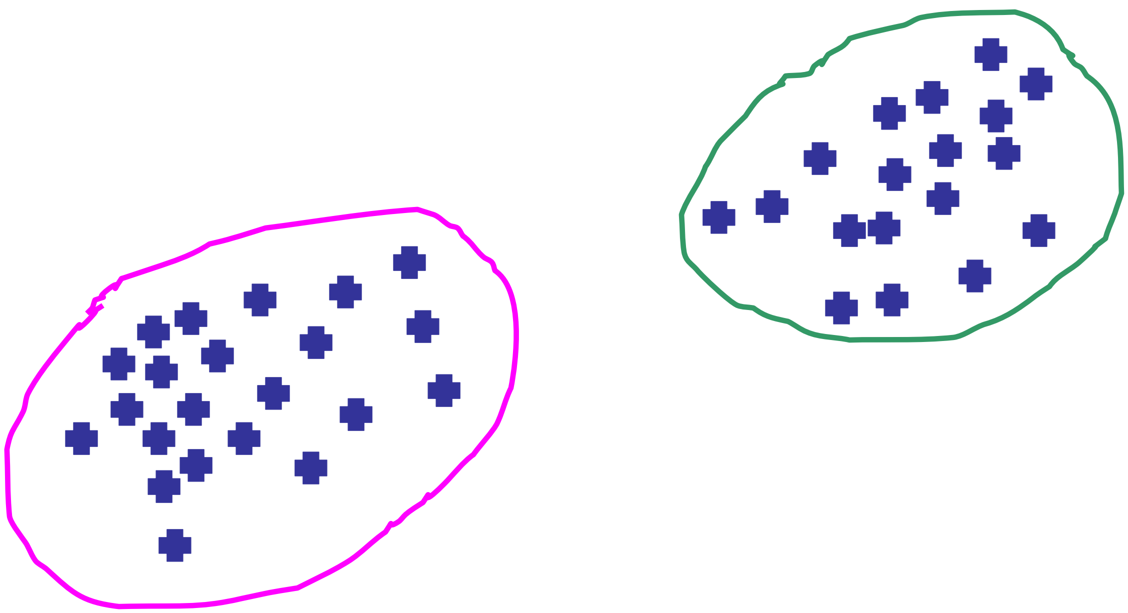
Called clustering.

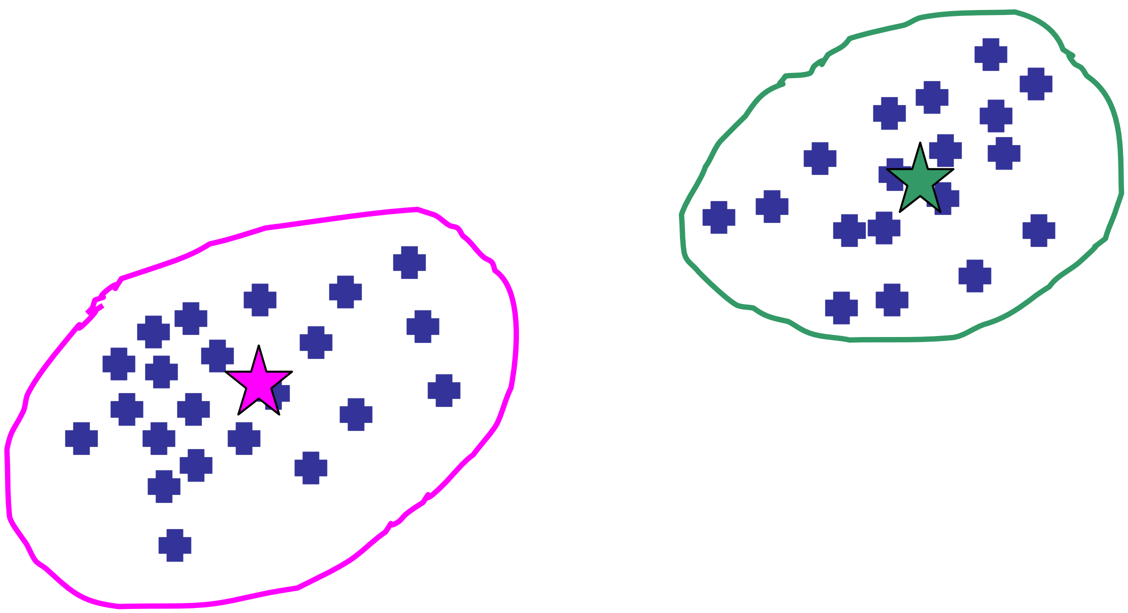


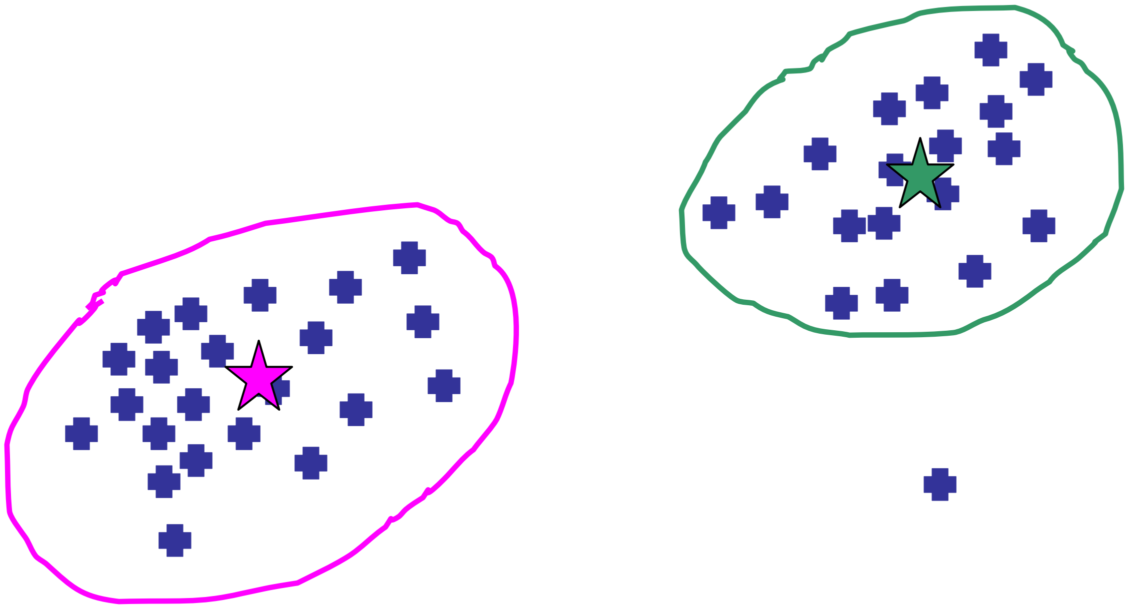


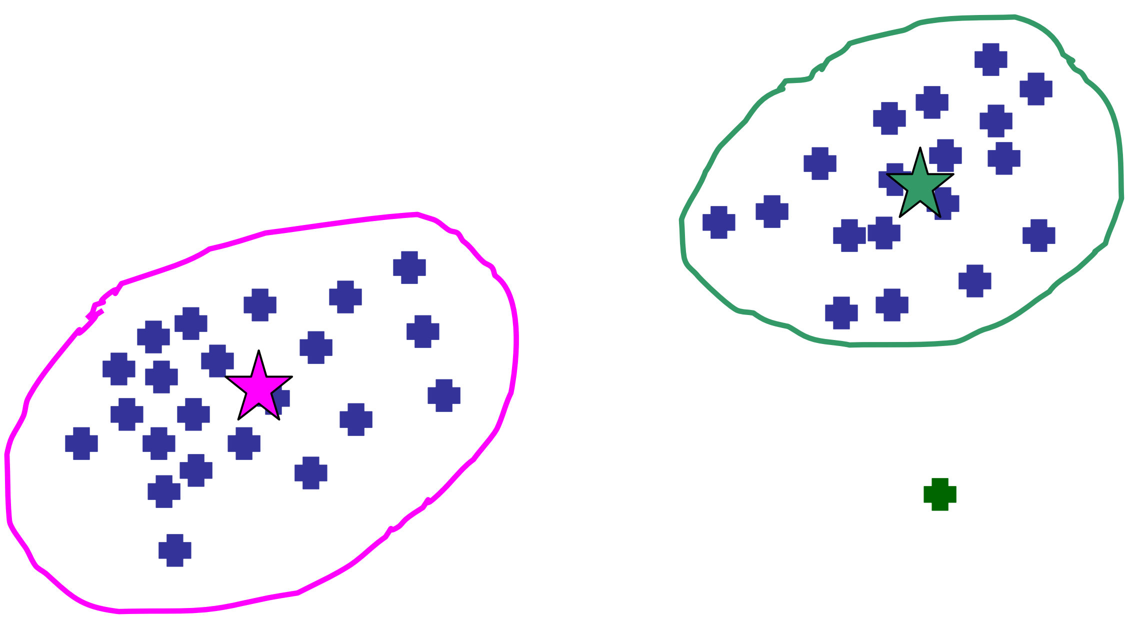


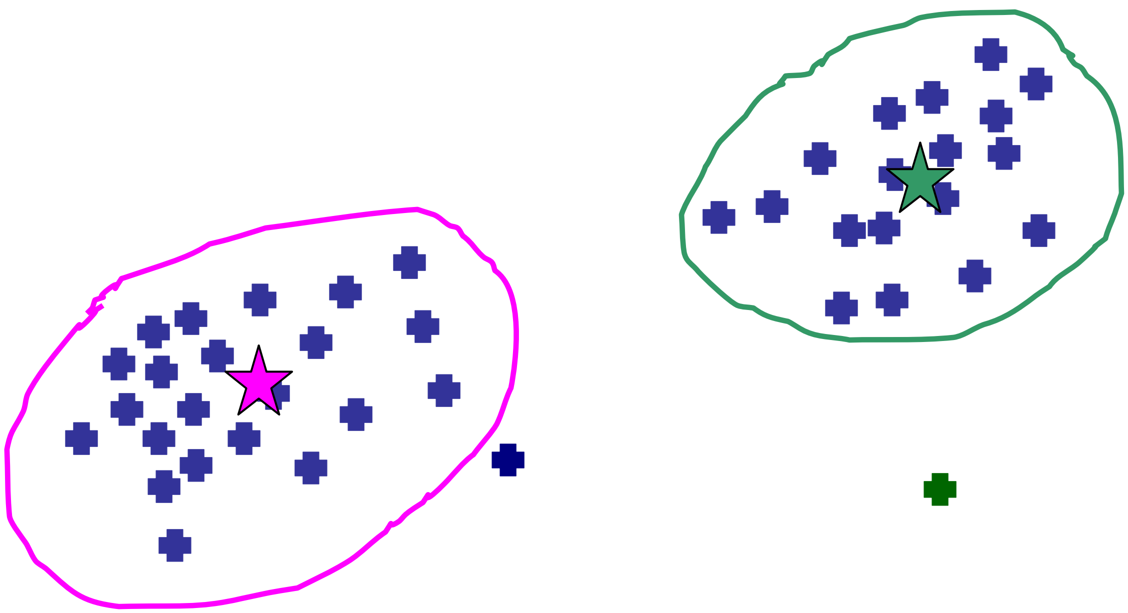


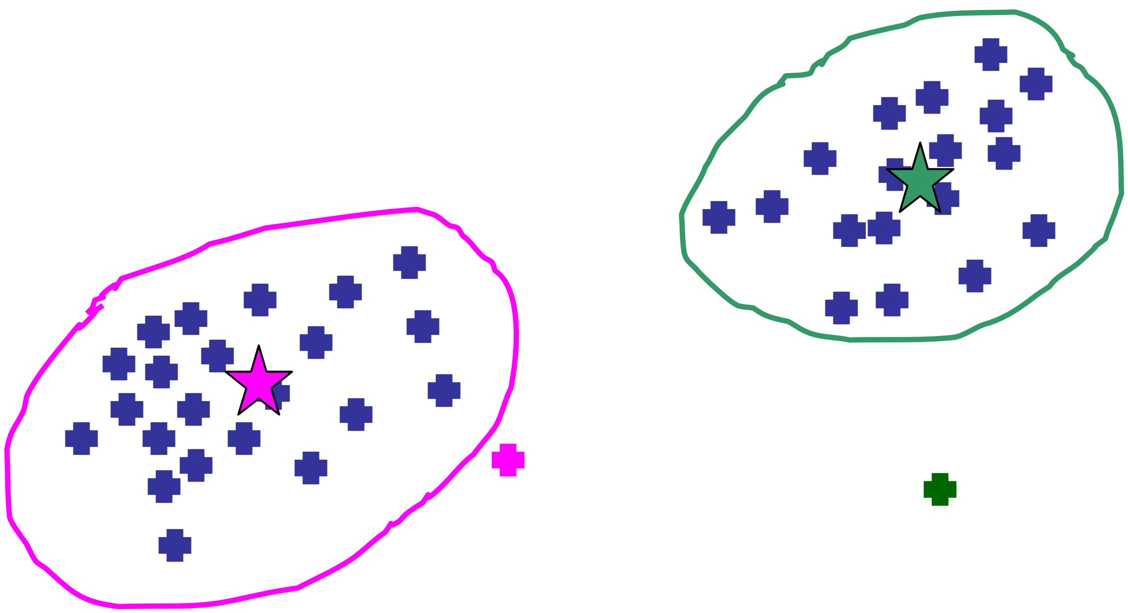


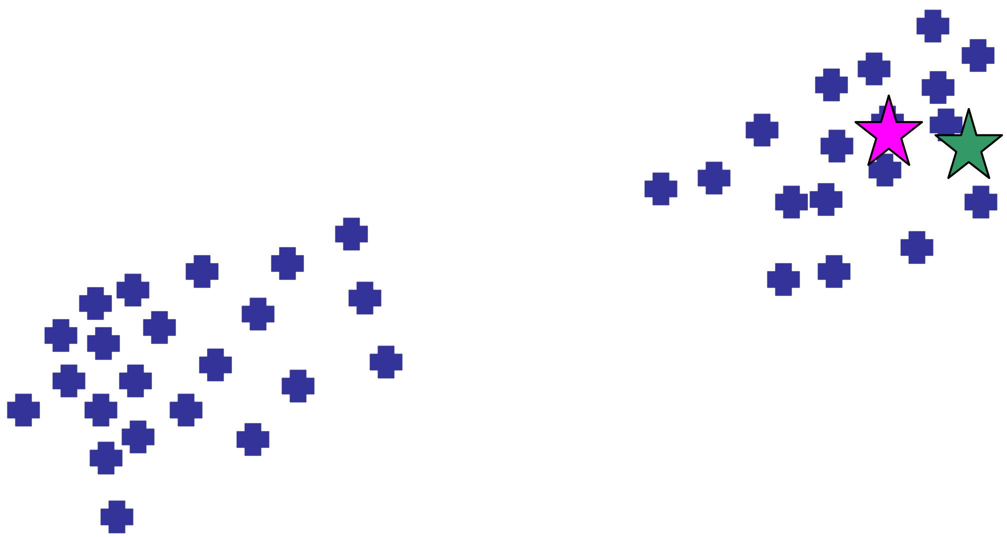


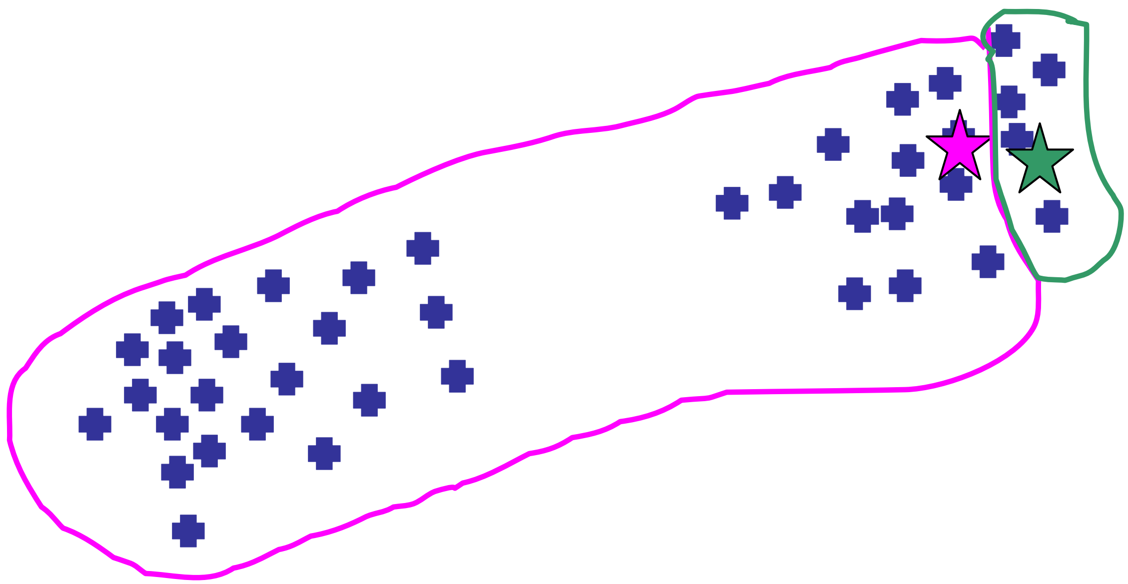


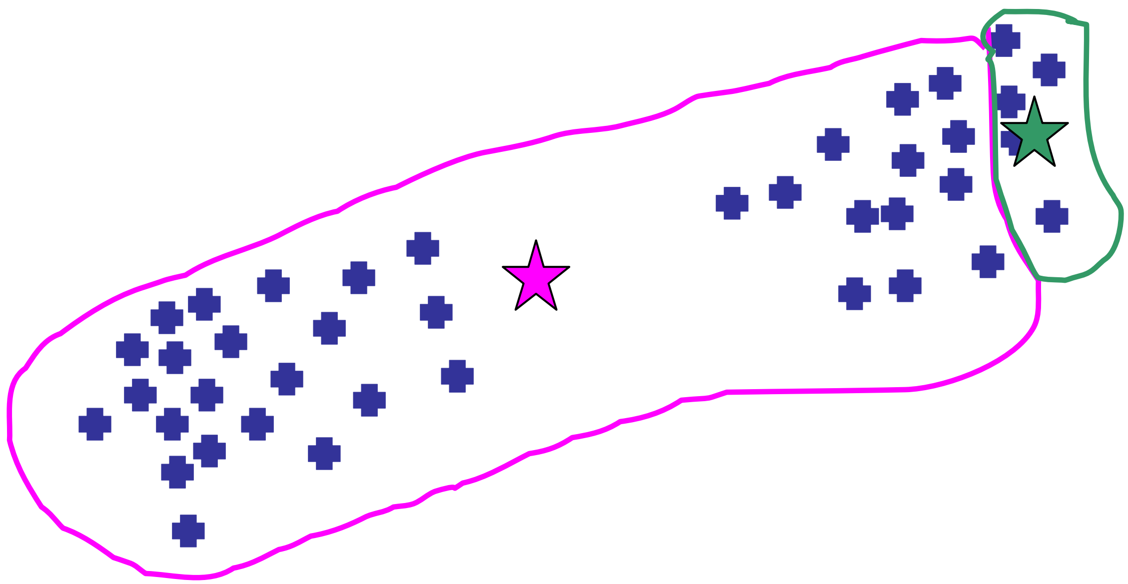


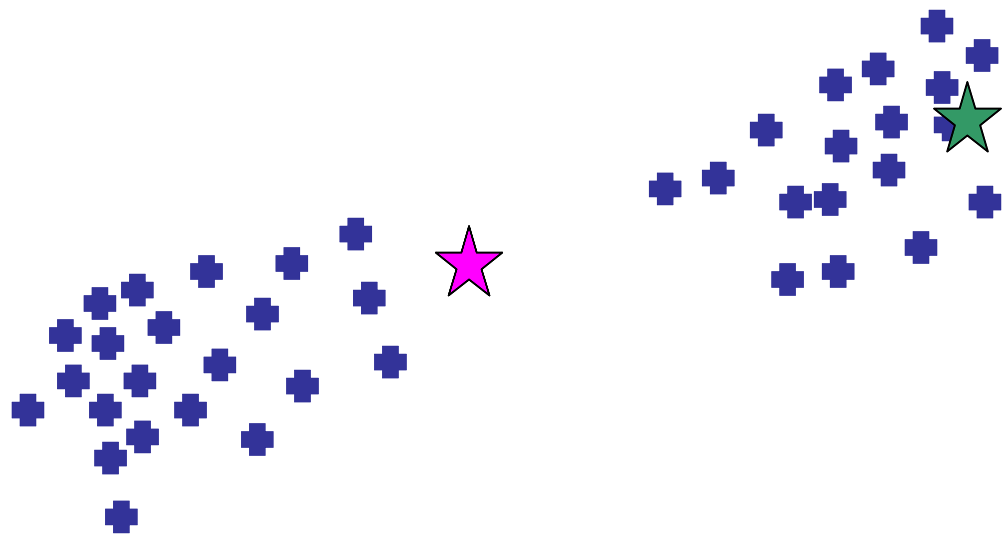


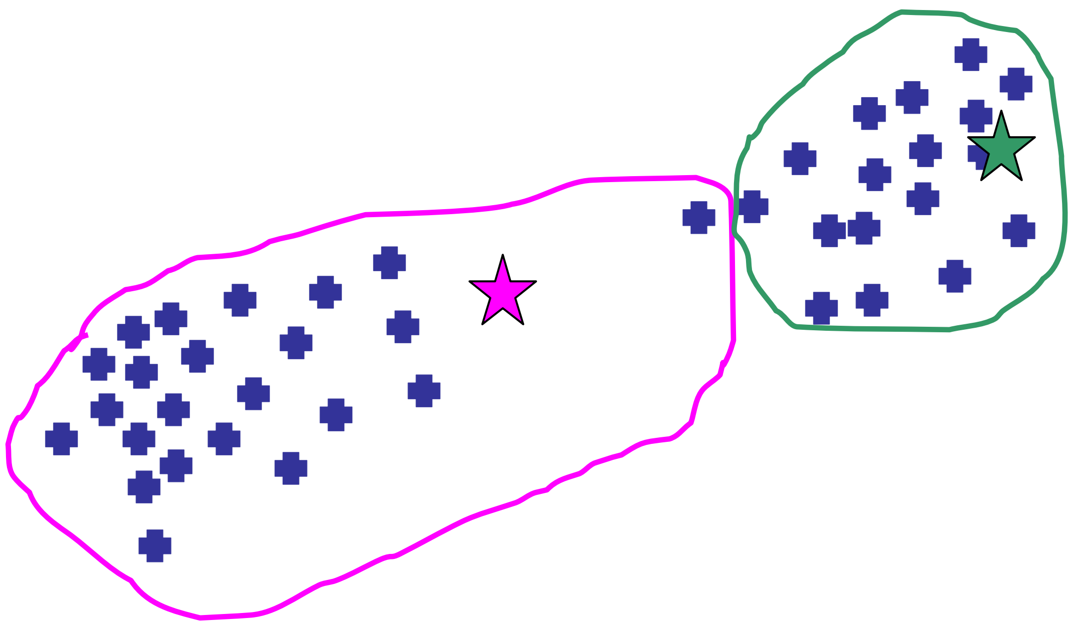


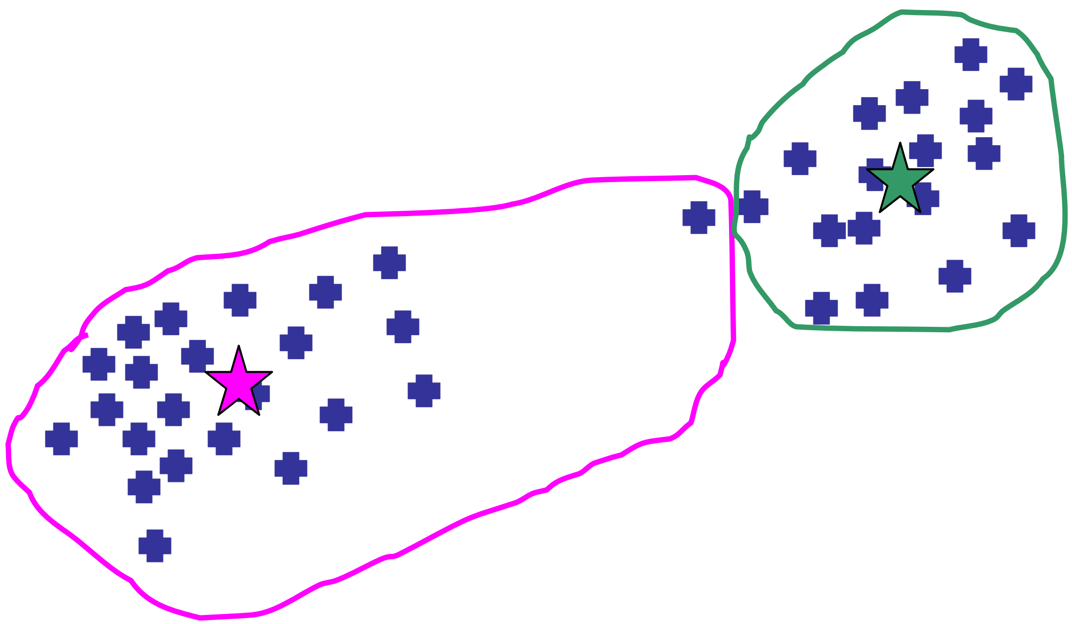


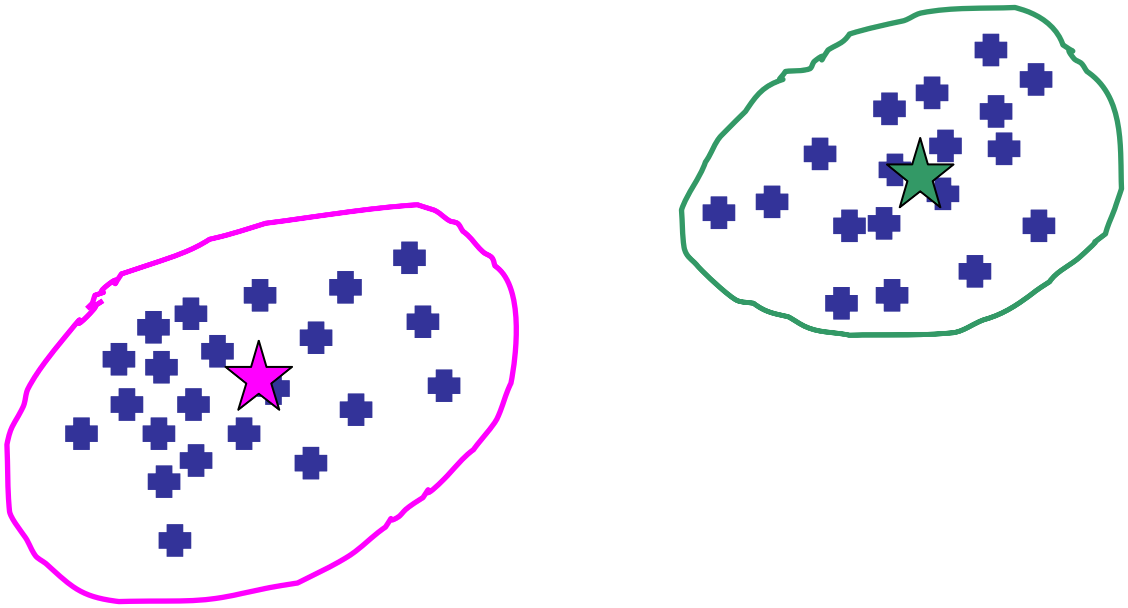










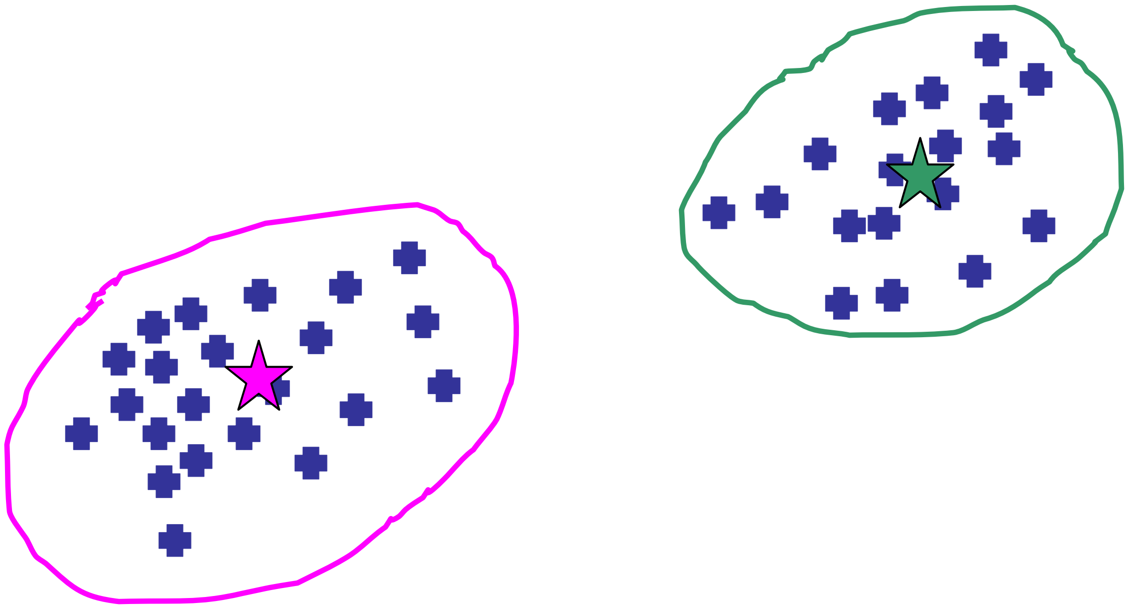


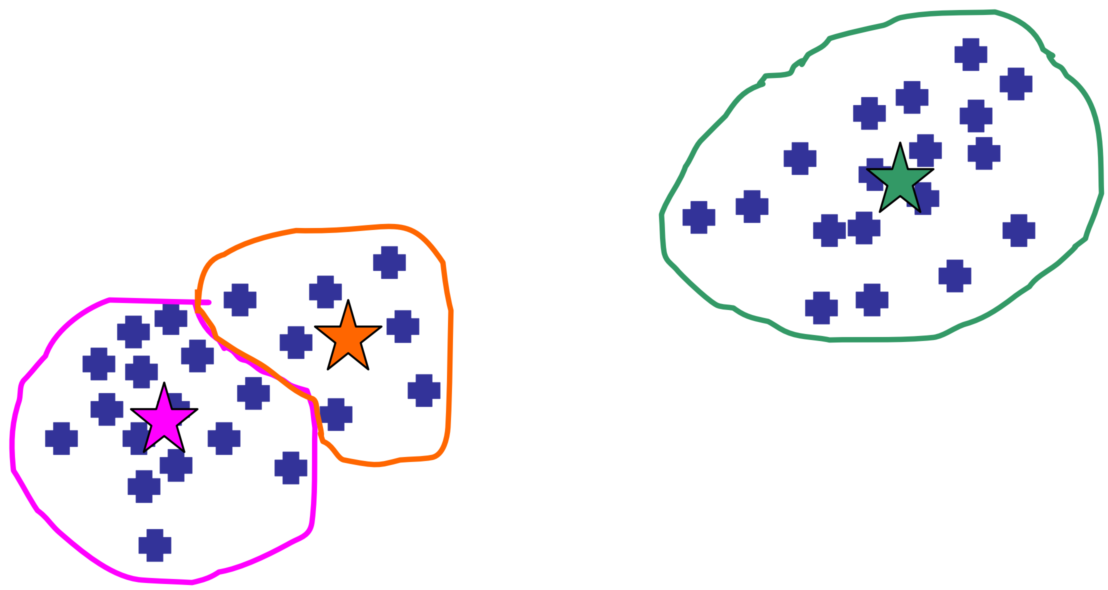
K-Means

- Pick cluster centers.
- Assign data to clusters.
- Find new cluster centers.
- Repeat.

K-Means

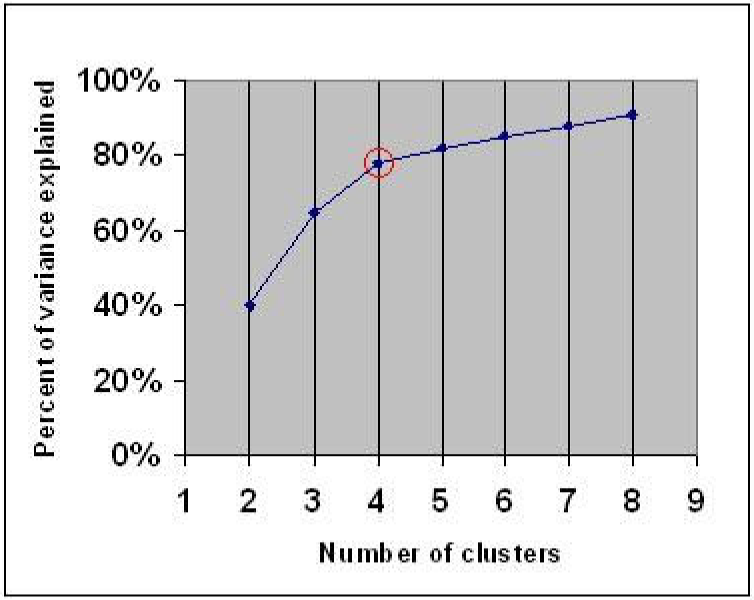
- Problems:
  - Number of clusters?





K-Means

- Problems:
  - Number of clusters?



K-Means

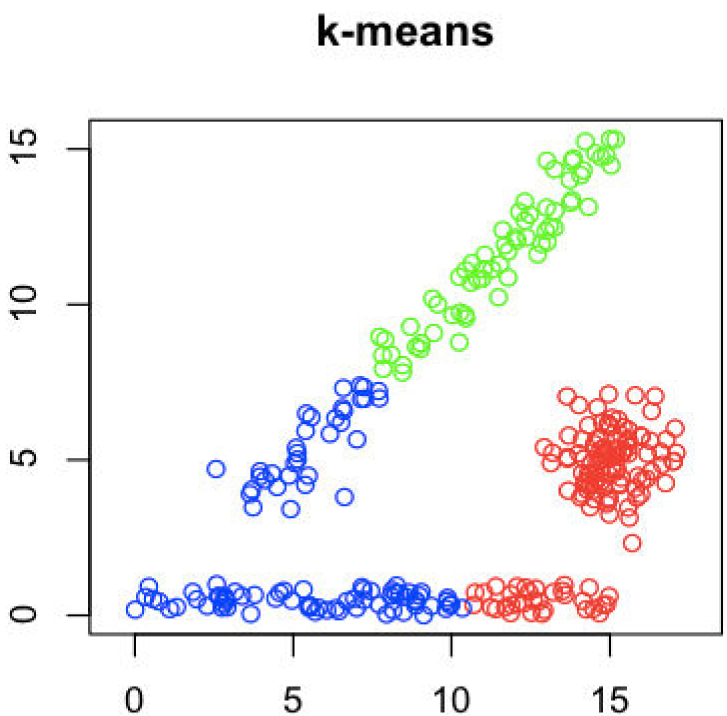
- Problems:
  - Number of clusters?
  - Starting positions?
    - Remember hill-climbing?

Neat Algorithms / Paradigms

- Spectral Clustering

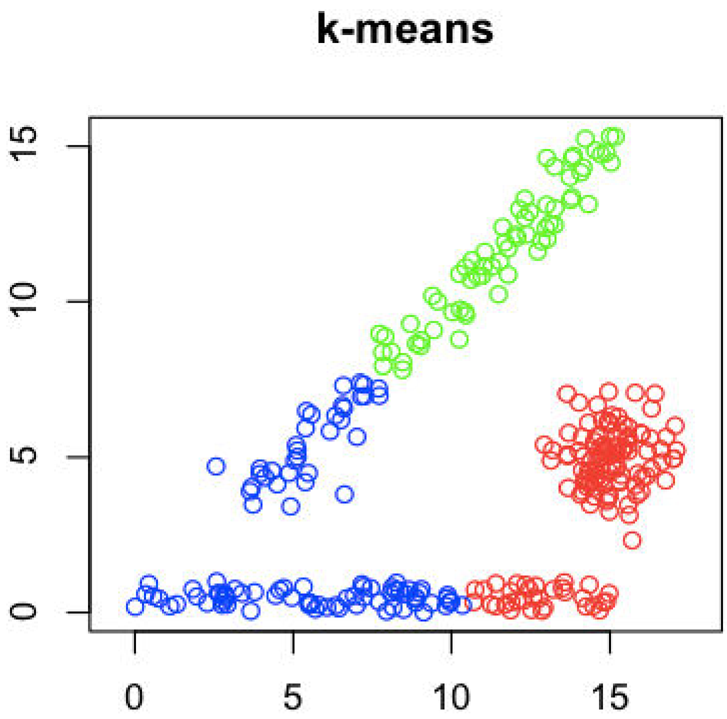
Neat Algorithms / Paradigms

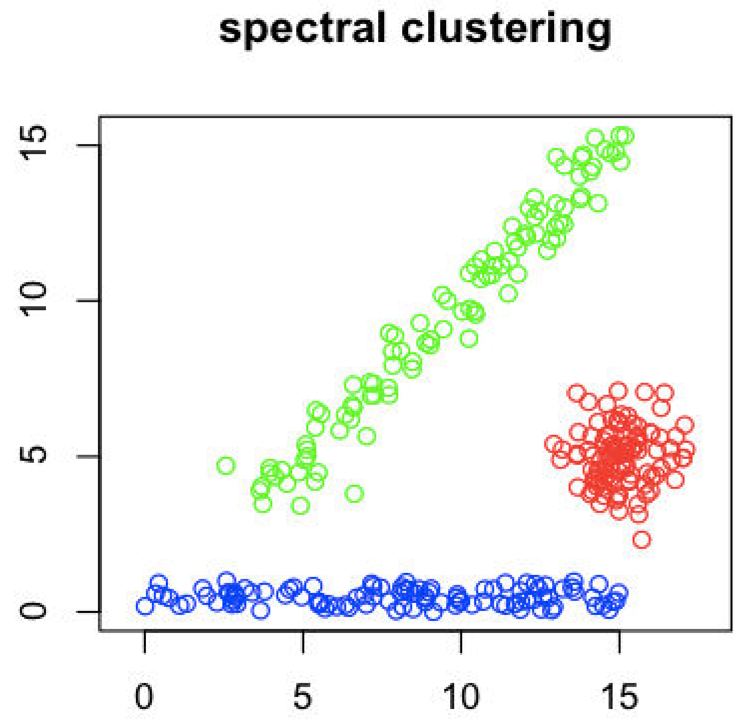
- Spectral Clustering

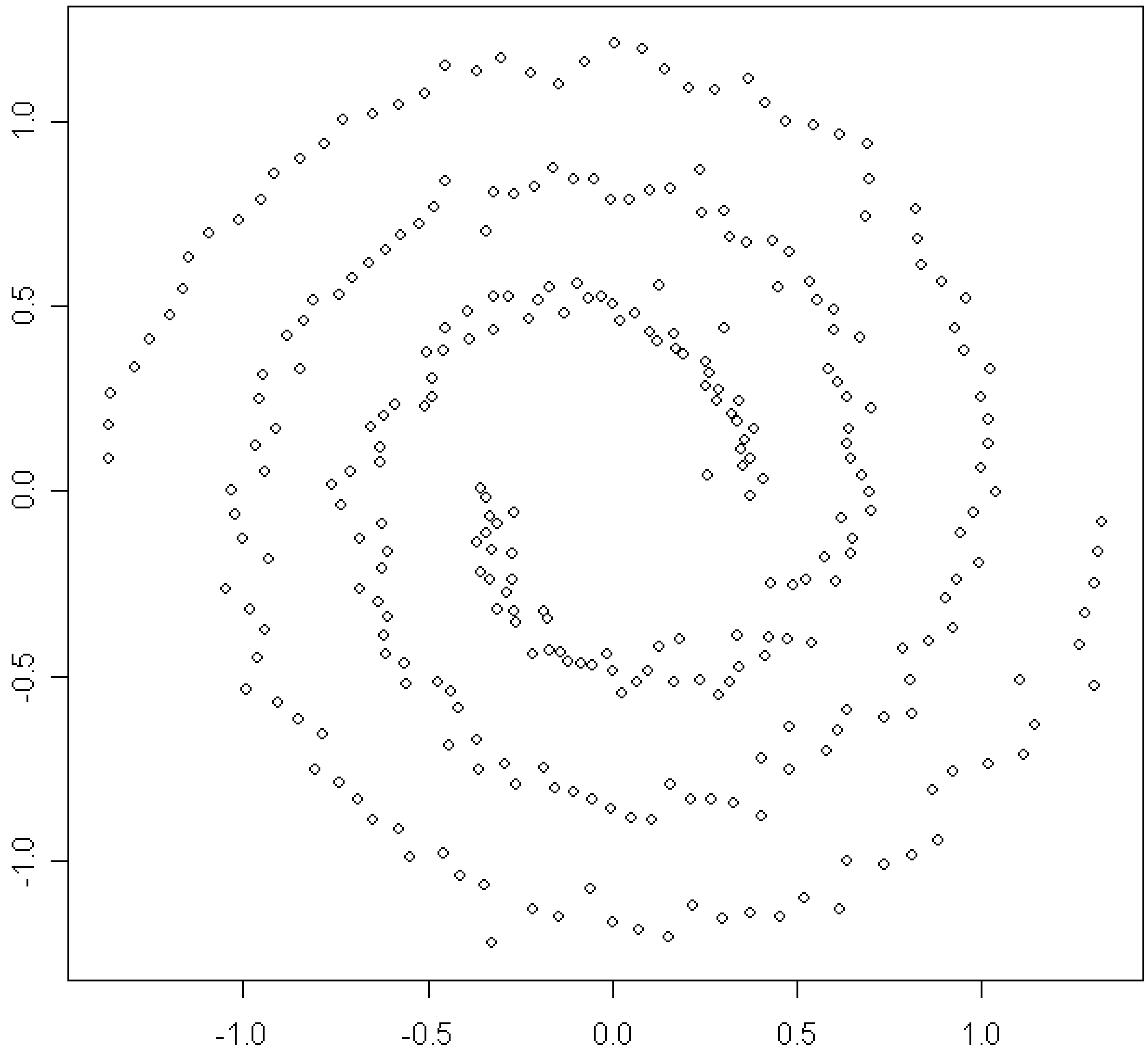


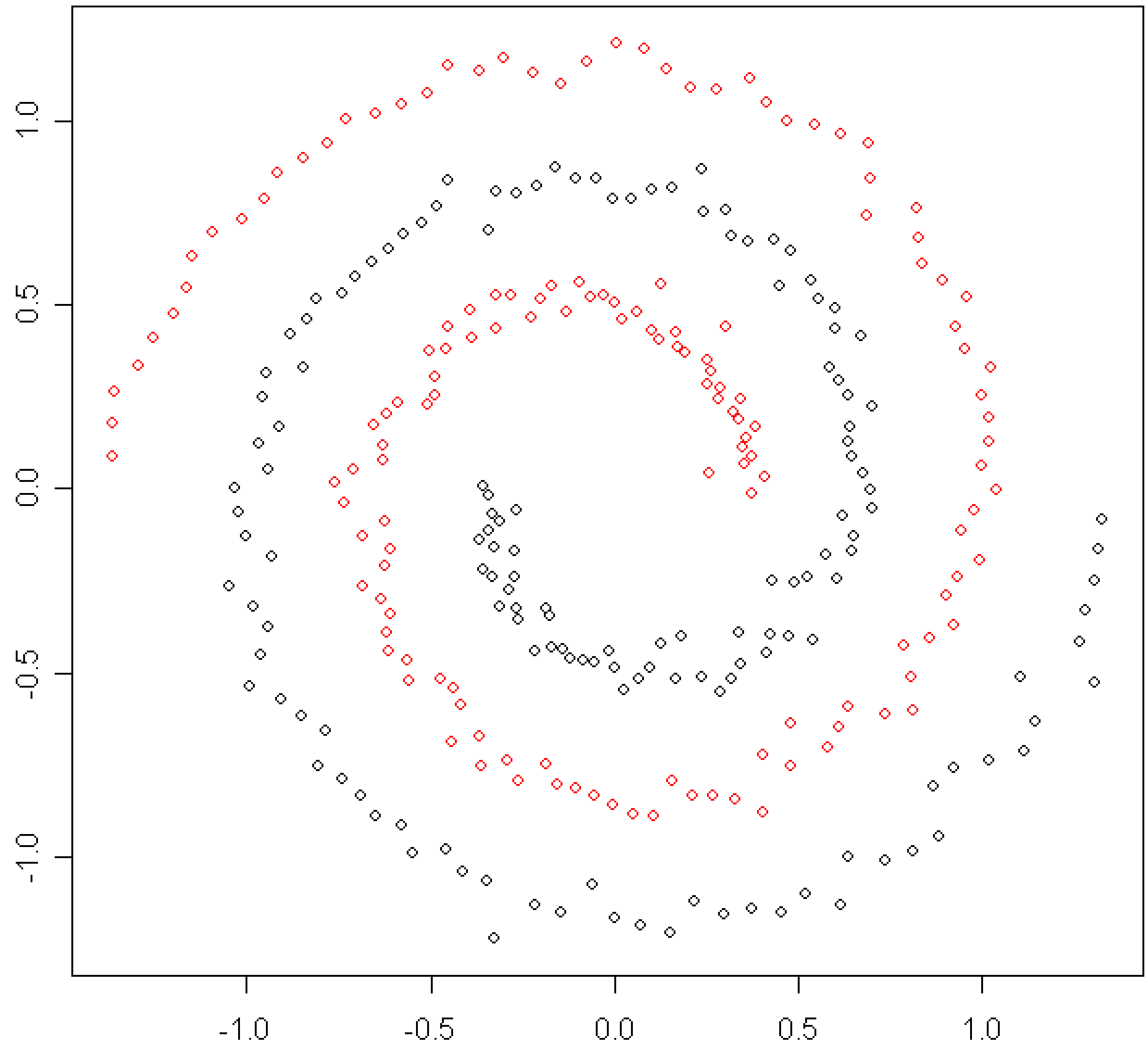
Neat Algorithms / Paradigms

- Spectral Clustering



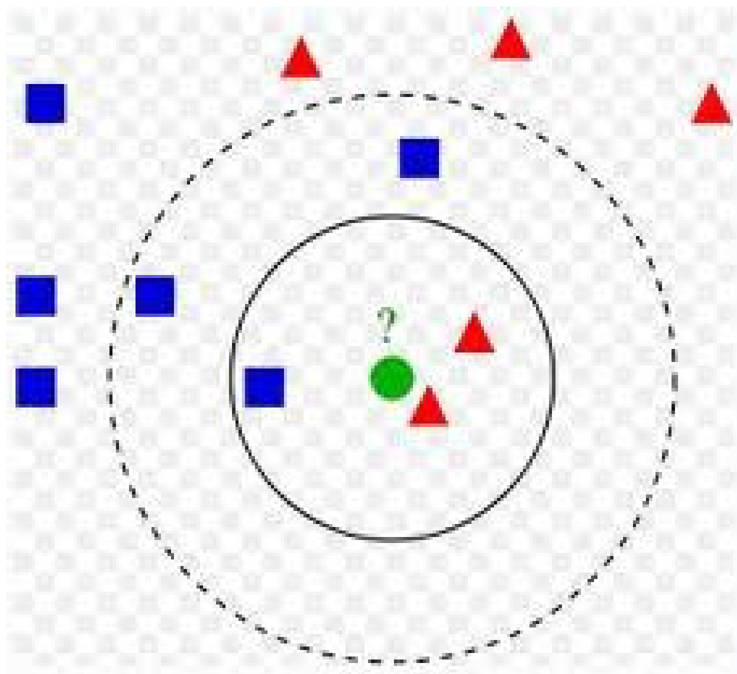



1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0


-1.0
-0.5
0.0
0.5
1.0

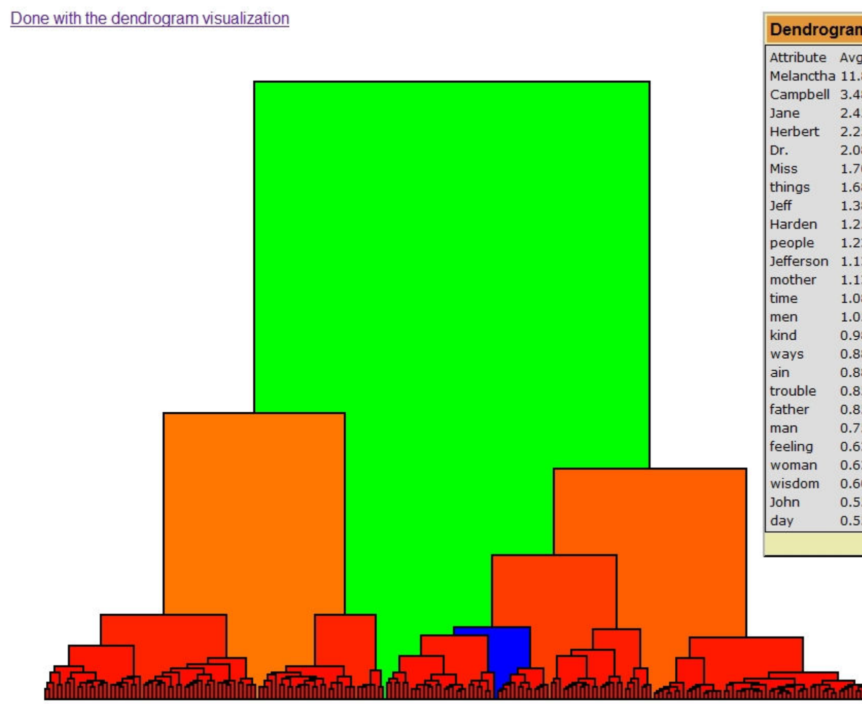
Neat Algorithms / Paradigms

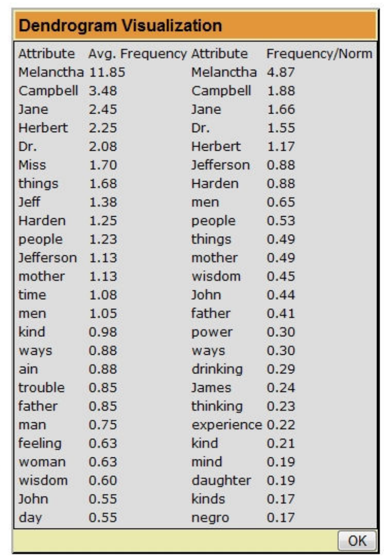
- Spectral Clustering
- k-Nearest Neighbors



Neat Algorithms / Paradigms

- Spectral Clustering
- k-Nearest Neighbors
- Hierarchical Agglomerative Clustering





Dendrogram Visualization

Attribute	Avg. Frequency	Attribute	Frequency/Norm
Melanctha	11.85	Melanctha	4.87
Campbell	3.48	Campbell	1.88
Jane	2.45	Jane	1.66
Herbert	2.25	Dr.	1.55
Dr.	2.08	Herbert	1.17
Miss	1.70	Jefferson	0.88
things	1.68	Harden	0.88
Jeff	1.38	men	0.65
Harden	1.25	people	0.53
people	1.23	things	0.49
Jefferson	1.13	mother	0.49
mother	1.13	wisdom	0.45
time	1.08	John	0.44
men	1.05	father	0.41
kind	0.98	power	0.30
ways	0.88	ways	0.30
ain	0.88	drinking	0.29
trouble	0.85	James	0.24
father	0.85	thinking	0.23
man	0.75	experience	0.22
feeling	0.63	kind	0.21
woman	0.63	mind	0.19
wisdom	0.60	daughter	0.19
John	0.55	kinds	0.17
day	0.55	negro	0.17

OK