Machine Learning

Rudolf Mayer

(mayer@ifs.tuwien.ac.at)

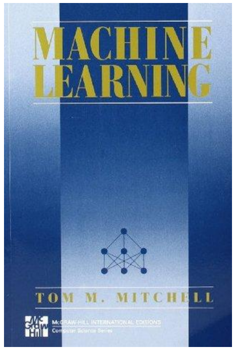
October 18th, 2017

Outline

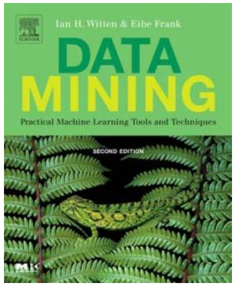
- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

Lecture materials

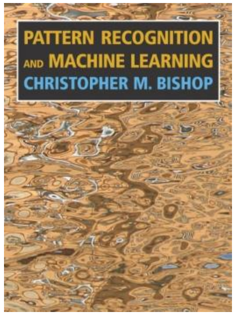
- Books
  - Tom Mitchell, “Machine Learning”



  - Ian Witten, “Data Mining: Practical Machine Learning Tools and Techniques” (WEKA Authors)



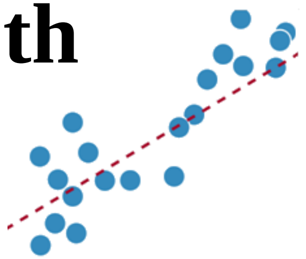
  - Christopher Bishop, “Pattern Recognition and Machine Learning”



Outline

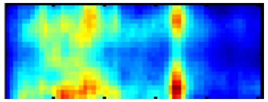
- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

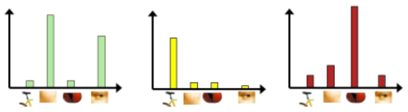
Recap – Lecture October 11th



- ML Definitions & setting
  - supervised, unsupervised, regression, classification reinforcement, ..

  - Feature extraction





- Data preparation
  - Scaling/Normalisation, 1-n coding, ..., missing values/imputation, .. ➔ *more in this lecture*
- Outlook on evaluation
  - Accuracy, Training/test set, ... *– more next lecture*

Short Recap

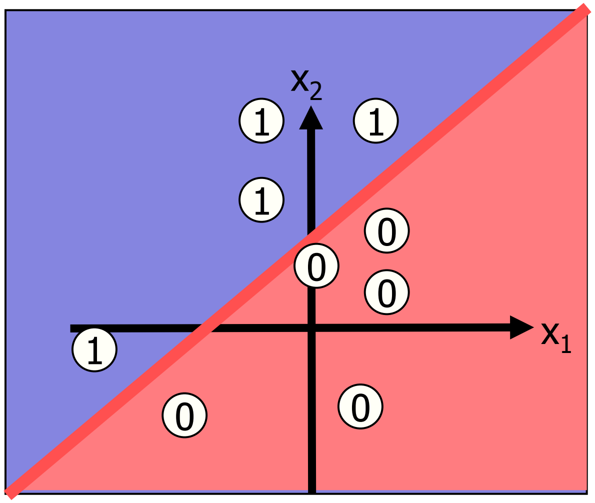
- Perceptron
  - Linear separation
    - by linear combination of inputs

$$a = \sum_{i=1}^{n} w_i x_i$$

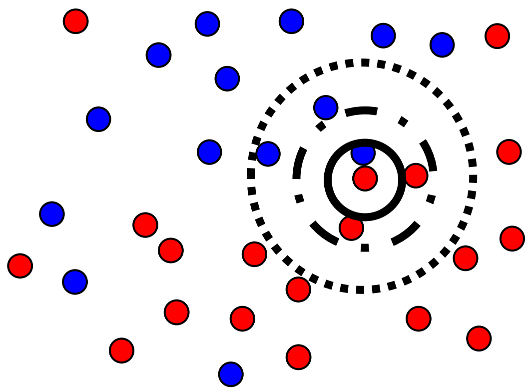
    - Passed through activation function

$$y = f(x) = \begin{cases} 1 \text{ if } a \geq \theta \\ 0 \text{ if } a < \theta \end{cases}$$

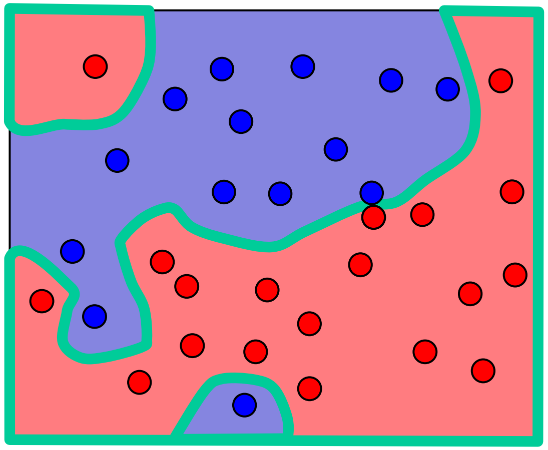
  - Iteratively learning weights & *bias*
    - As long as input samples are not classified correctly
  - Linear separation only



Short Recap

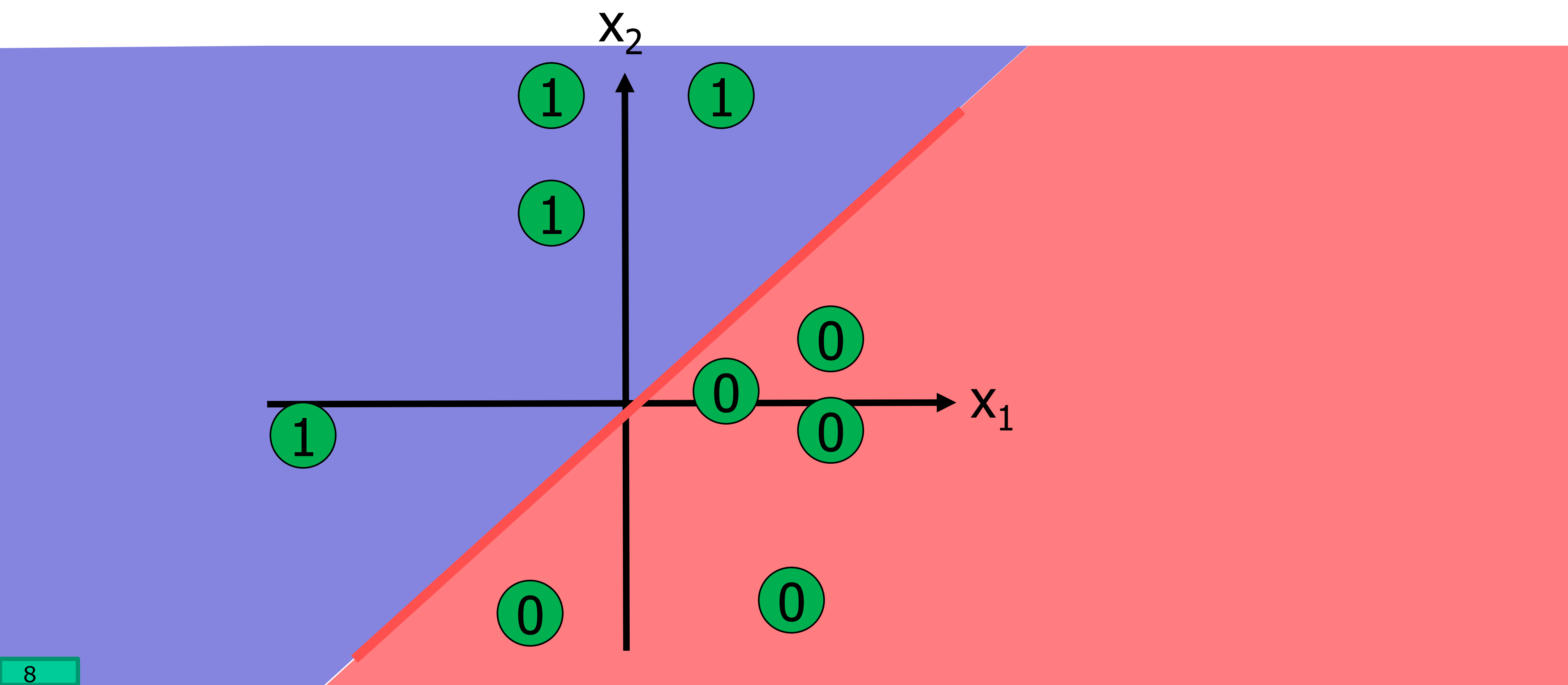


- k-NN Classification
  - Searching for k-closest neighbours
    - k needs to be defined by the analyst
  - Classification follows majority of neighbours
  - Lazy learner / Instance based learning
    - Does not build a model beforehand
    - All the computation at classification step
  - Not limited to linear separation!
  - *Optimisations for finding neighbours*

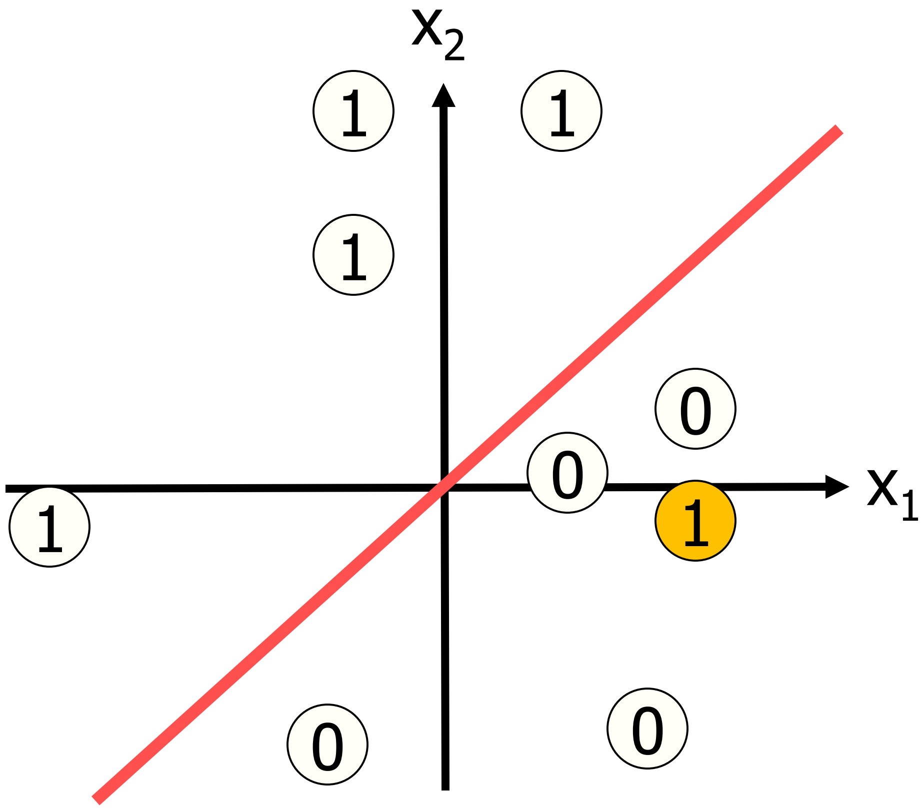


Perceptron: properties

- Separates linearly
  - Converges to a stable state when data is *linear separable*
  - Otherwise: not (necessarily) deterministic

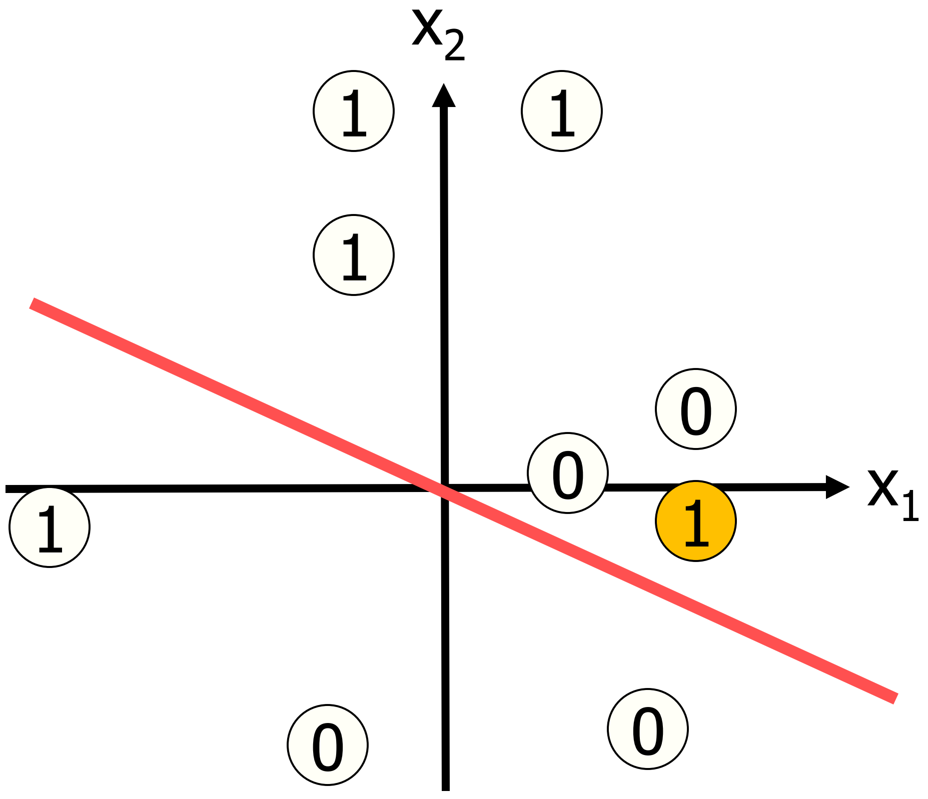


Perceptron: Non-linear separable



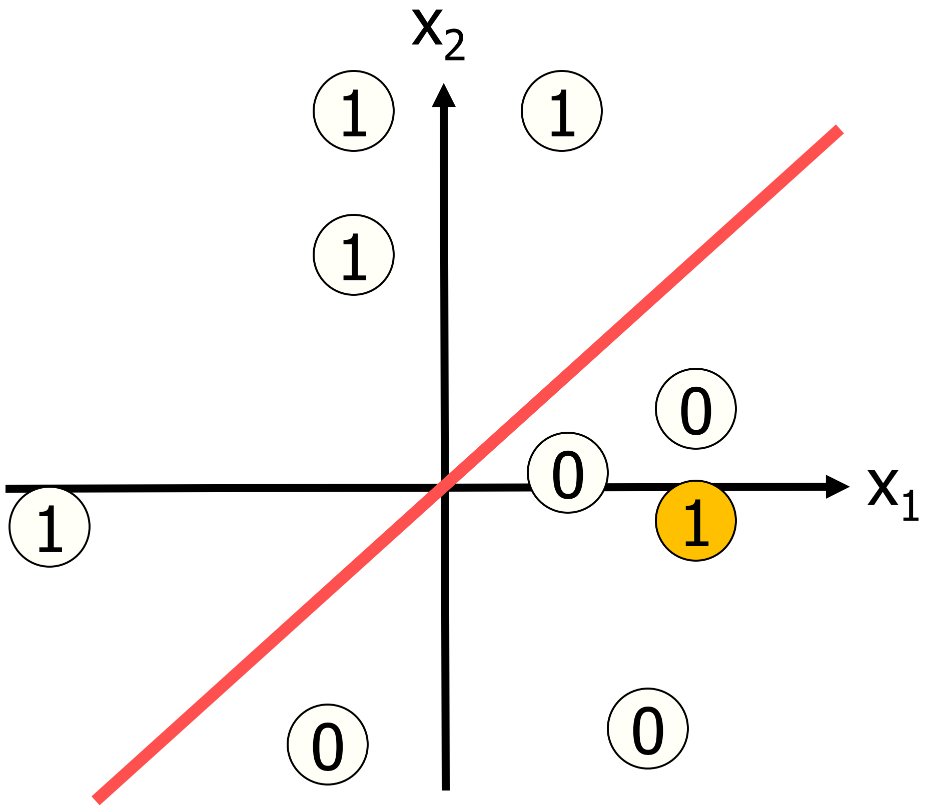
- *Perceptron will not converge to a stable state, but oscilate*

Perceptron: Non-linear separable



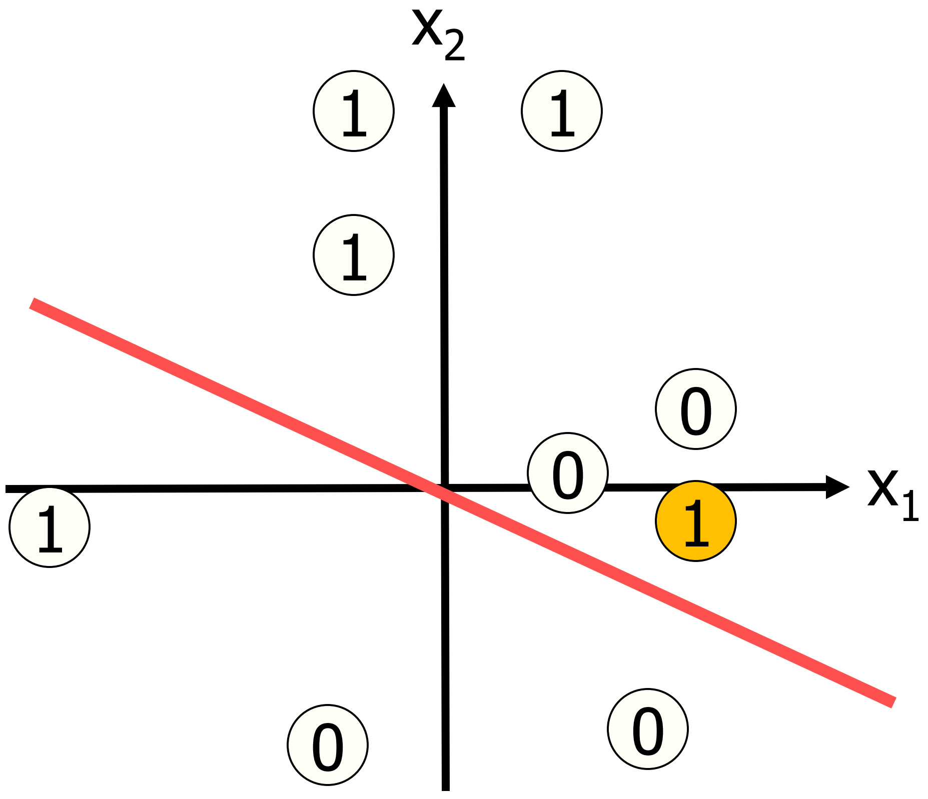
- *Perceptron will not converge to a stable state, but oscilate*

Perceptron: Non-linear separable



- *Perceptron will not converge to a stable state, but oscilate*
- *Solution?*

Perceptron: Non-linear separable



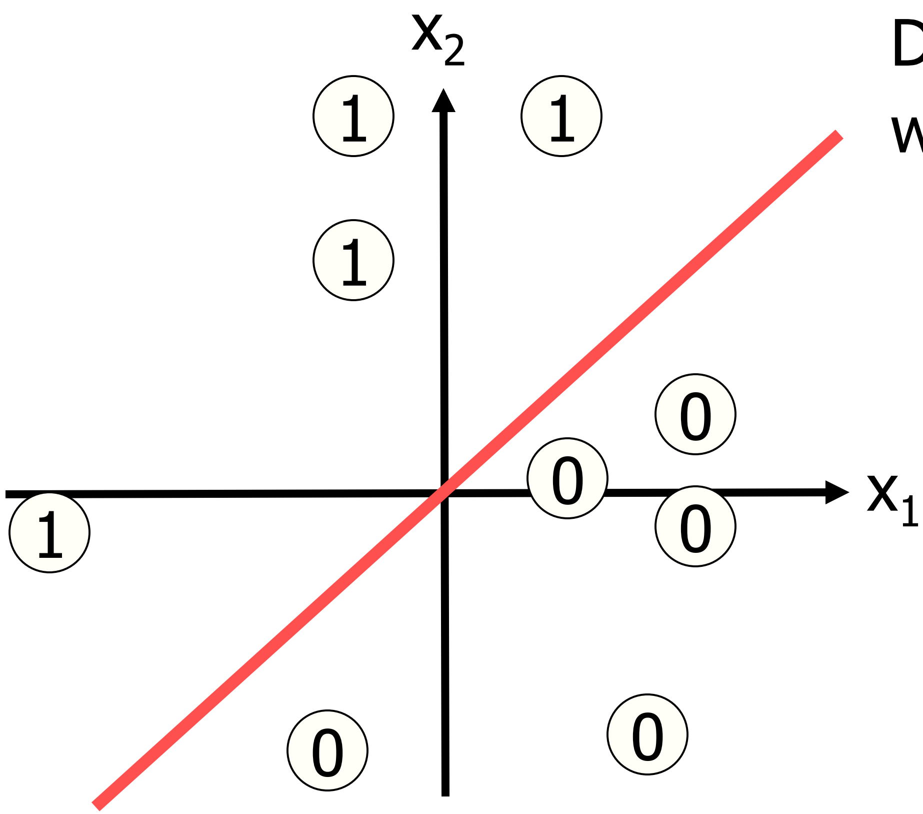
- *Perceptron will not converge to a stable state, but oscilate*

➔ *Need stopping criterion to end training*

Perceptron: Non-linear separable

- Perceptron stopping criteria
  - Stop after a maximum number of n training iterations
    - *How to set n?*
  - Stop if there is no more improvement since k iterations
    - Improvement measured e.g. as Accuracy (correctly classified samples)
  - ....
- Pocket algorithm:
  - Keeps the best solution found so far
    - (e.g. highest accuracy)
  - Returns that solution (instead of last state)
- Other optimisations (e.g. Kernels, see SVM)

Perceptron: bias

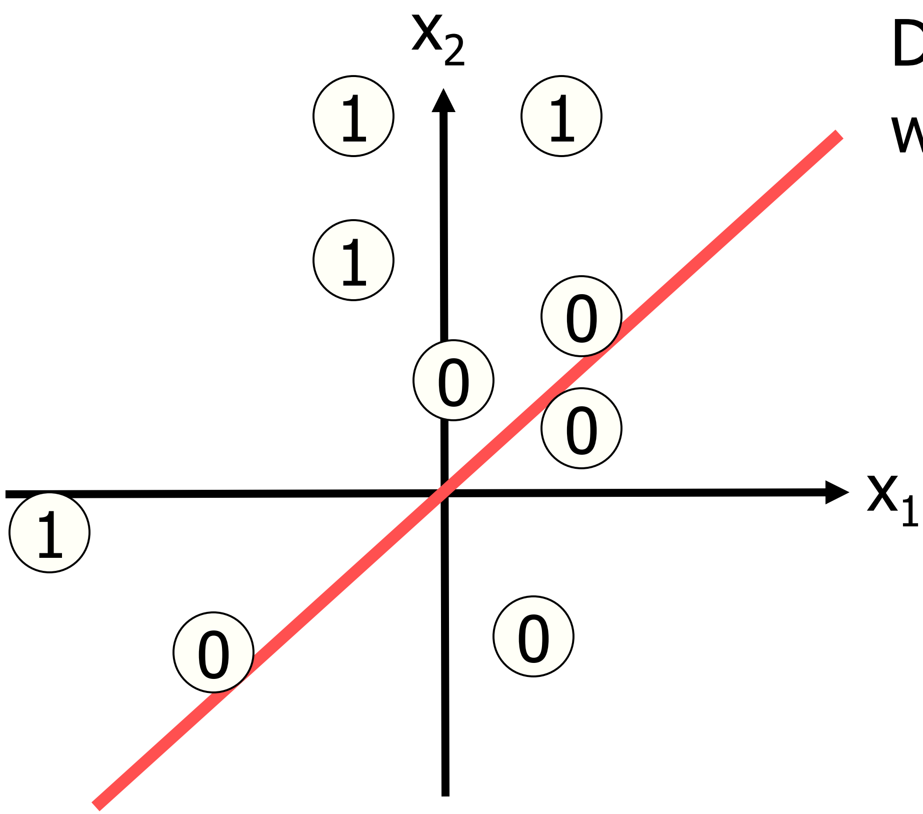


Decision line

$w_1 x_1 + w_2 x_2 = \theta$

(cf. $y = ax$)

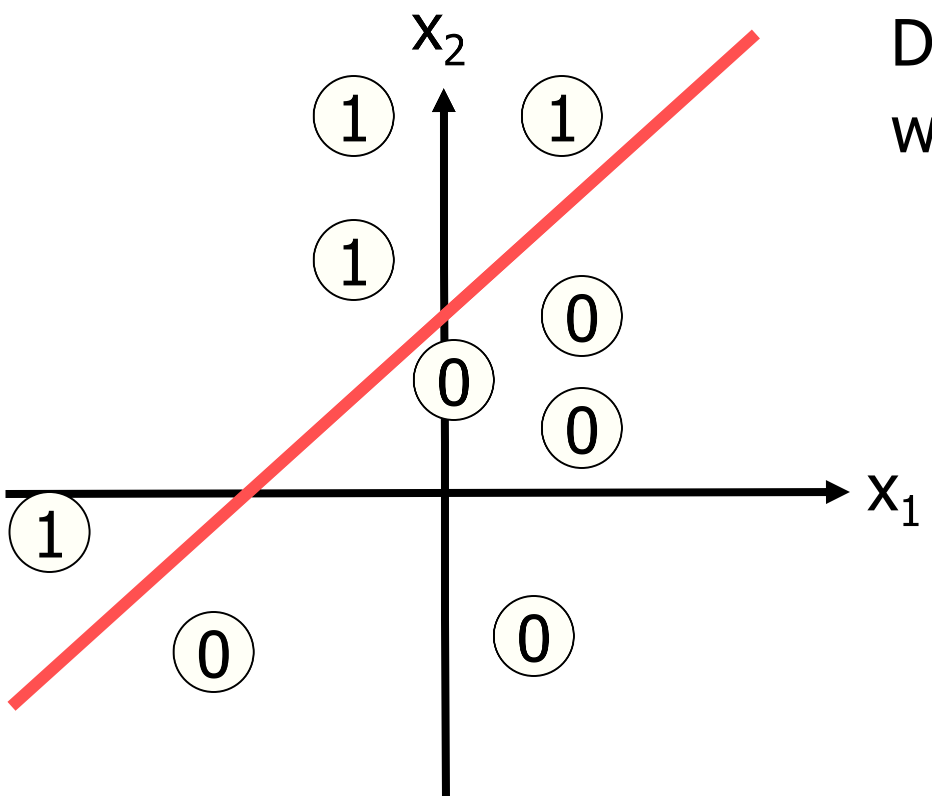
Perceptron: bias



Decision line

$w_1 x_1 + w_2 x_2 = \theta$

Perceptron: bias

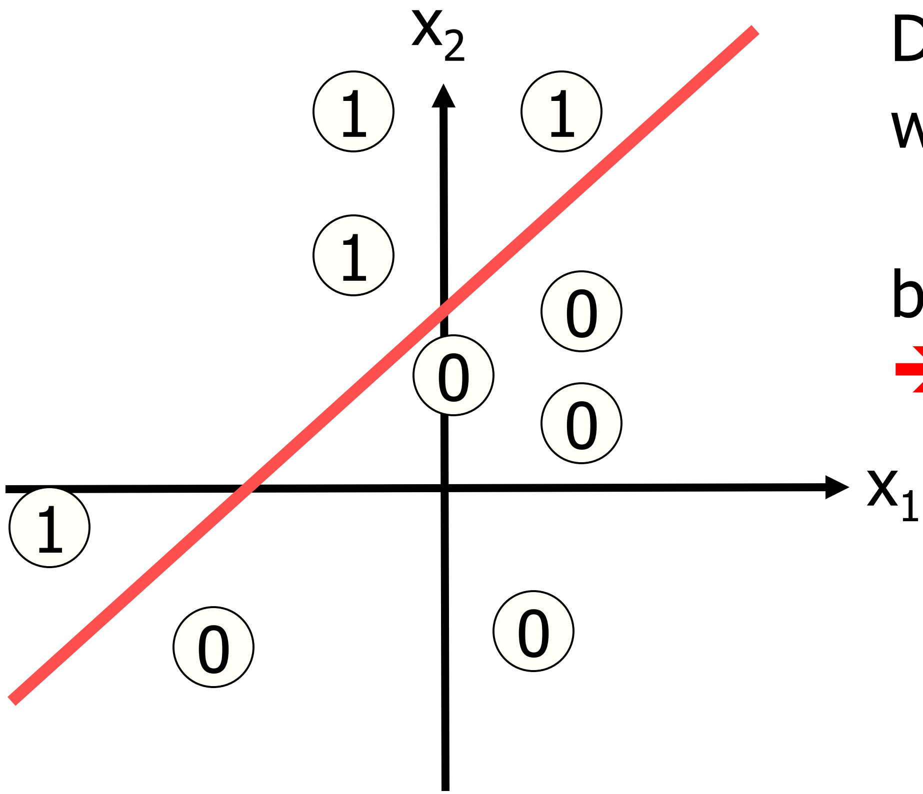


Decision line

$w_1 x_1 + w_2 x_2 + b = \theta$

(cf. y = ax + b)

Perceptron: bias



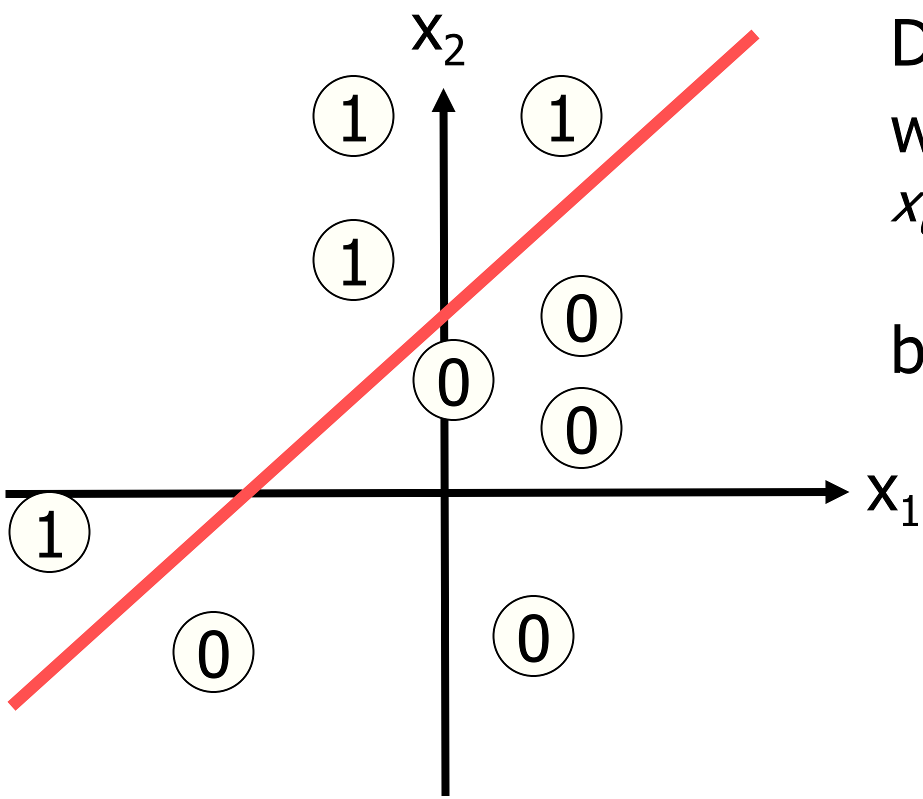
Decision line

$w_1\, x_1 + w_2\, x_2 + b = \theta$

bias b: constant

➔ how to determine?

Perceptron: bias



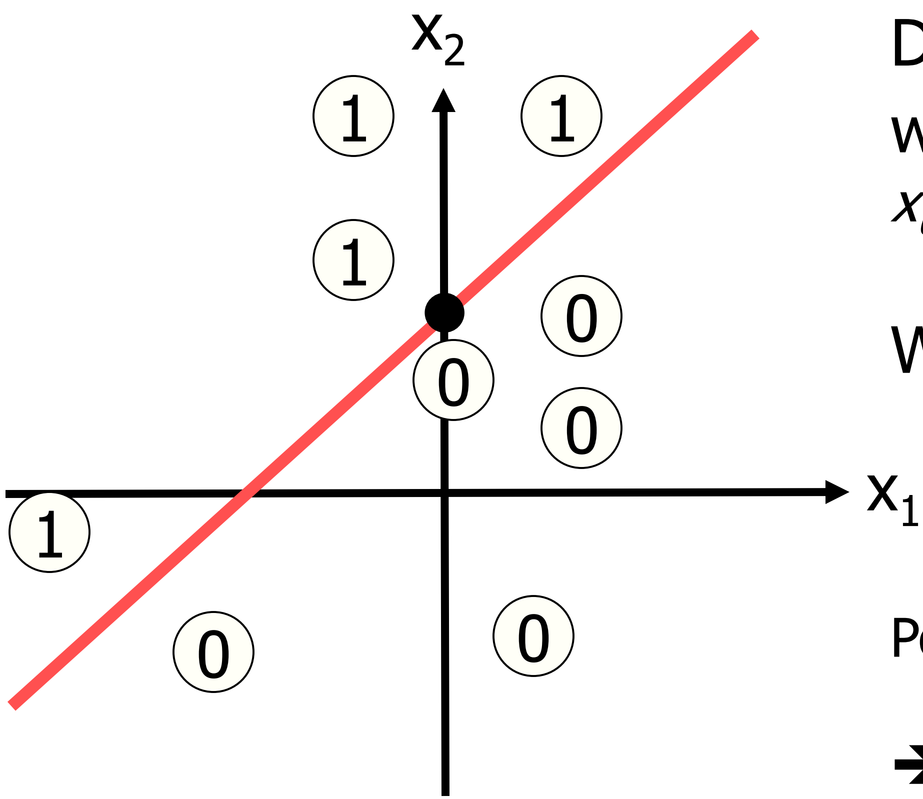
Decision line

$w_0 x_0 + w_1 x_1 + w_2 x_2 = \theta$

x_0 = 1 (constant)

bias w_0: learned

Perceptron: bias



Decision line

$w_0 x_0 + w_1 x_1 + w_2 x_2 = \theta$

x_0 = 1 (constant)

W_1=-1, W_2=1, b = -2

Point on decision line

(0,2)

➔-2*1 + -(1)*0 + 1*2 =

= -2 + 0 + 2 = 0 = θ

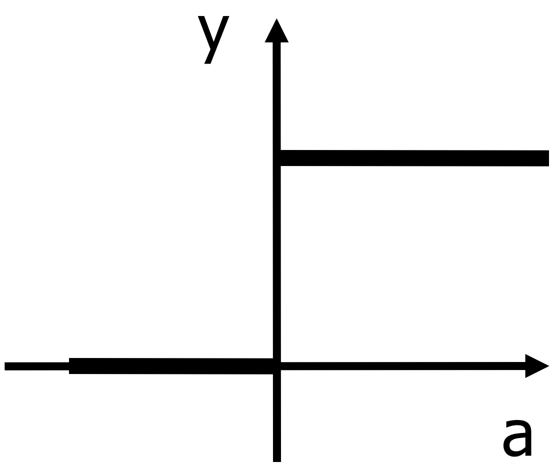
Extensions to Perceptrons

- Perceptron is a simple, one-neuron neural network
  - Basis for architectures that combine many basic neurons
    - Input layer & output layer
    - Hidden layers
    - Early motivation: model of human brain
      - Neurons connected via synaptic links

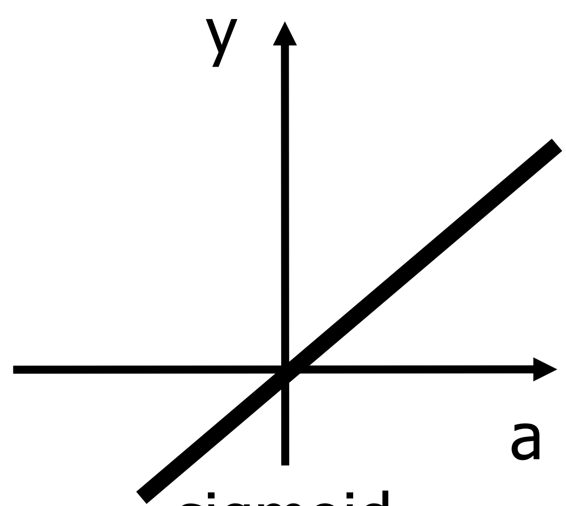
- Extensions for non-linear separable classes
  - Data space projections (Kernels), similar to SVM
    - More on that later ....

Perceptron: Activation Functions

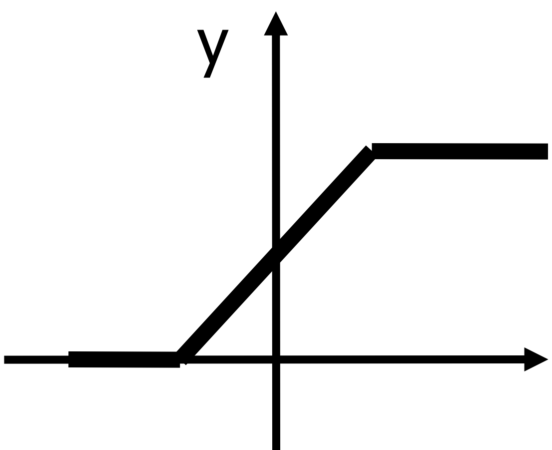
threshold



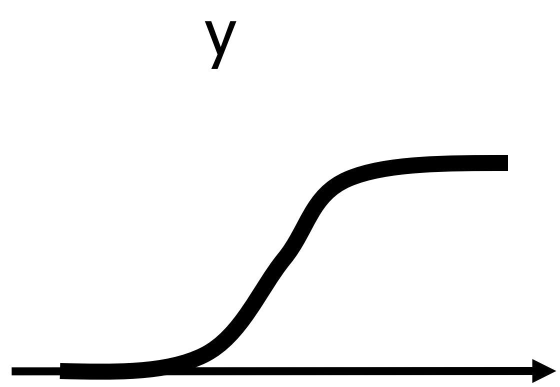
linear



piece-wise linear



sigmoid



Outline

- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

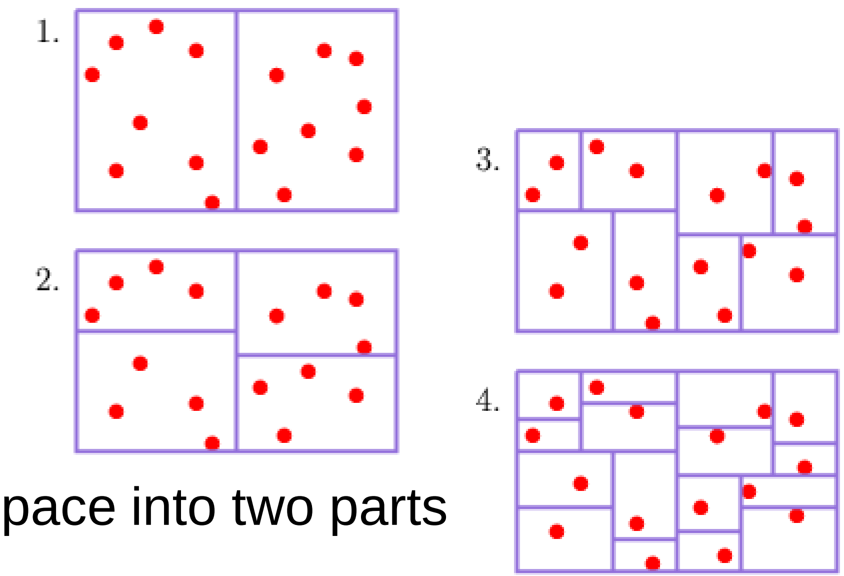
k-NN: majority voting

- Simple case: all k neighbours are equally important
  - Majority decides on the class
- Weighted approaches: more influence to closer neighbours
  - Rank weighted: weight determined by rank
    - E.g. each neighbour has a vote 1/r, where r is the rank (1, 2, .., k)
  - Distance weighted: weight determined by distance
    - E.g. each neighbour has a vote 1/d, where d is the (Euclidean) distance from the sample to the neighbour
  - *Differences in these two approaches?*

k-NN: other search strategies

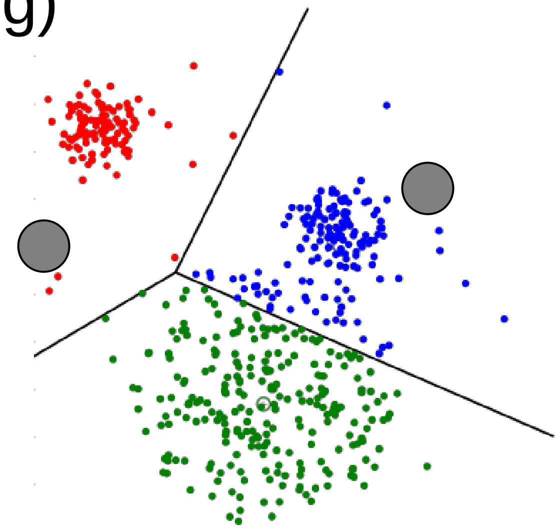
- Becomes computationally expensive with many items to classify
  - Linear (brute-force) search: O (Nd) *N = # samples, d = dimension*

- Search space-partitioning
  - E.g. kd-Tree
    - binary tree, every node is k-dimensional point
    - non-leaf node: splitting hyperplane, divides space into two parts
    - Traverse tree to find search space
    - Only search for neighbours in that area



k-NN: other search strategies

- Search Space Partitioning
  - E.g. Vector Quantisation (Clustering)
    - Obtain *prototypes* (e.g. k-Means)



    - For a new sample: find closest prototype
    - Then search for neighbours in the same cluster

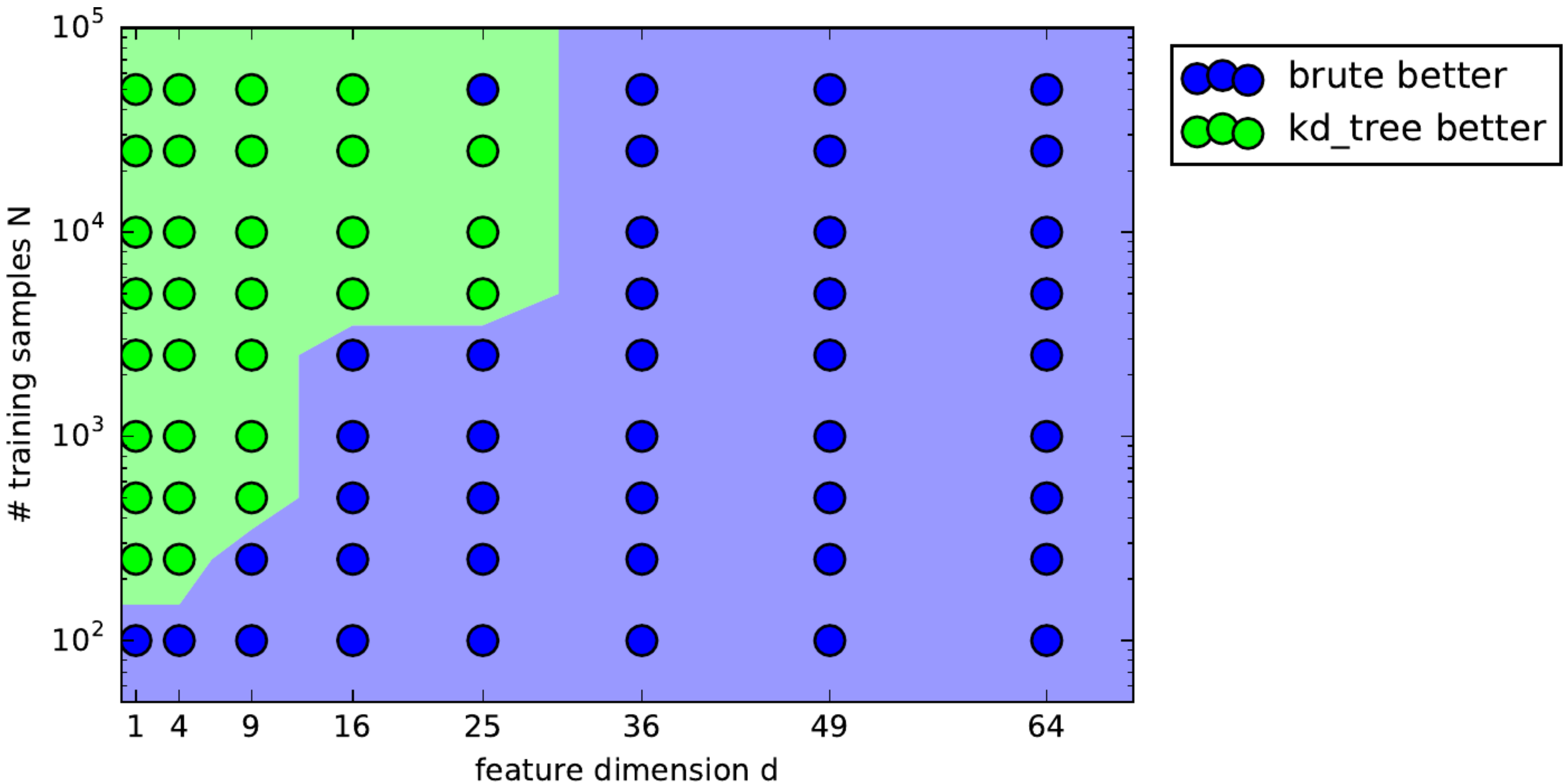
k-NN: other search strategies

- Advantage: can significantly reduce time
  - Practical experiments: up to 10 times
  - Important: need to consider both the time for the search (classification, **AND** for creating the tree/vector quantisation (this becomes your training time/model)

- *Disadvantages ?*
  - Does ***not*** pay off for small datasets
    - kd-Tree shown to perform worse with high d
  - May yield different predictions *(why?)*

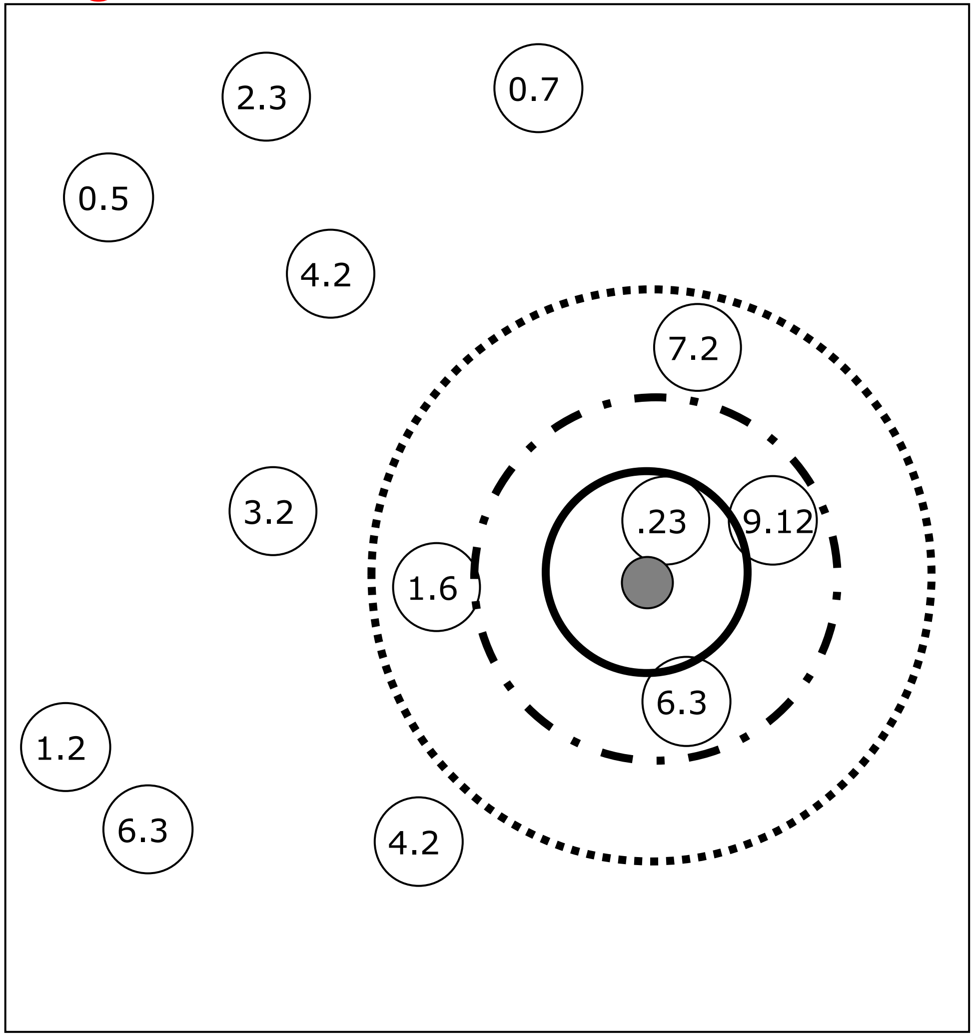
k-NN: other search strategies

- Kd-Tree evaluation (**combined** training & classification)



k-nn: regression

- *Can k-NN be used for regression?*
  - How ?
  - Output is average of continuous values assigned to k-nearest neighbours
  - Potentially weighted averages
    - (rank, distance, ...)

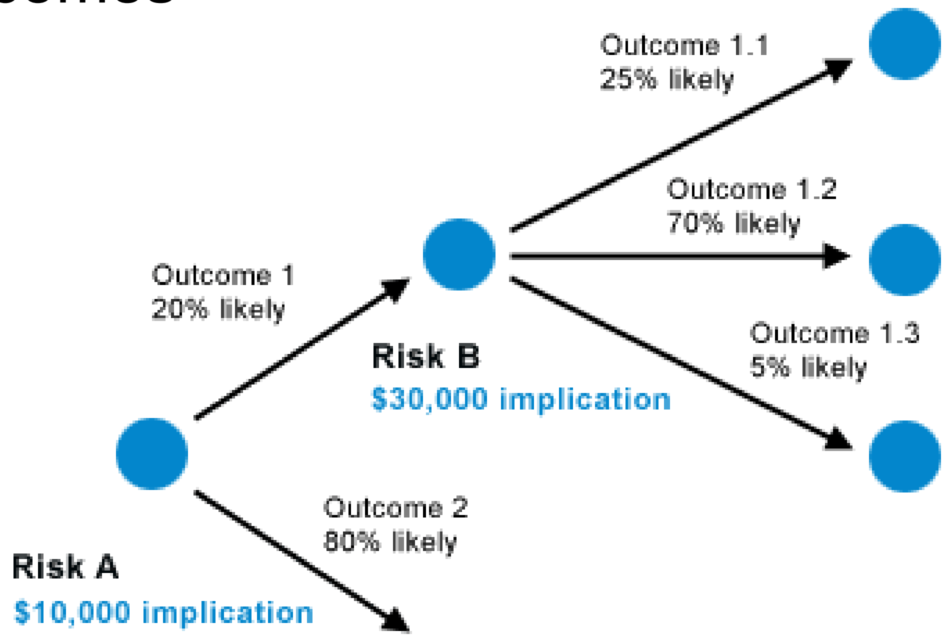


Outline

- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

Decision Trees

- In general: decision support tool
- Tree-like graph, flowchart
- Models decisions and their outcome
  - Potentially including probabilities, costs, …
  - Leaf nodes: events / outcomes
  - Other nodes: decisions



Decision Trees in ML

- Rather old (1960s, 1970s)
- Simple model
- Easy to understand, used in many domains (management, …)
- Classification & regression
- Categorical ***and*** numerical data
- Simplified version: 1R

Decision Trees in ML

- *Can be manually modelled by experts*
- ML: ***learn*** tree from training data
- Use tree to *predict* classes for unlabeled data

- Leaf nodes: define outcome (classes/value)
- Inner nodes: decisions, based on specific attributes and their values
  - Decisions: binary or more (n-ary)
    - I.e. two or more branches from inner nodes
    - Categorical data: number of different attribute values = max number of branches

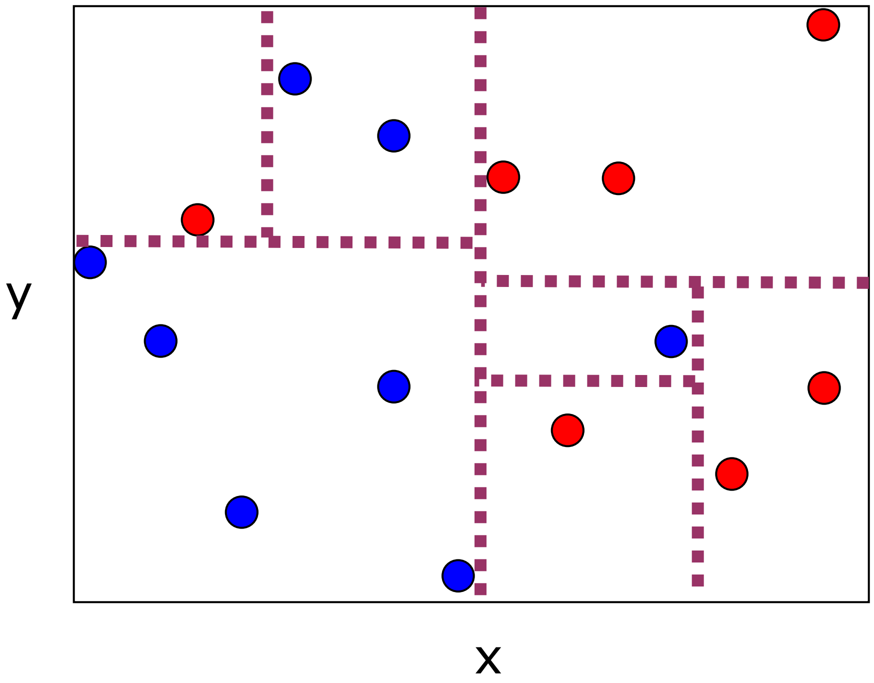
Decision Tree Learning

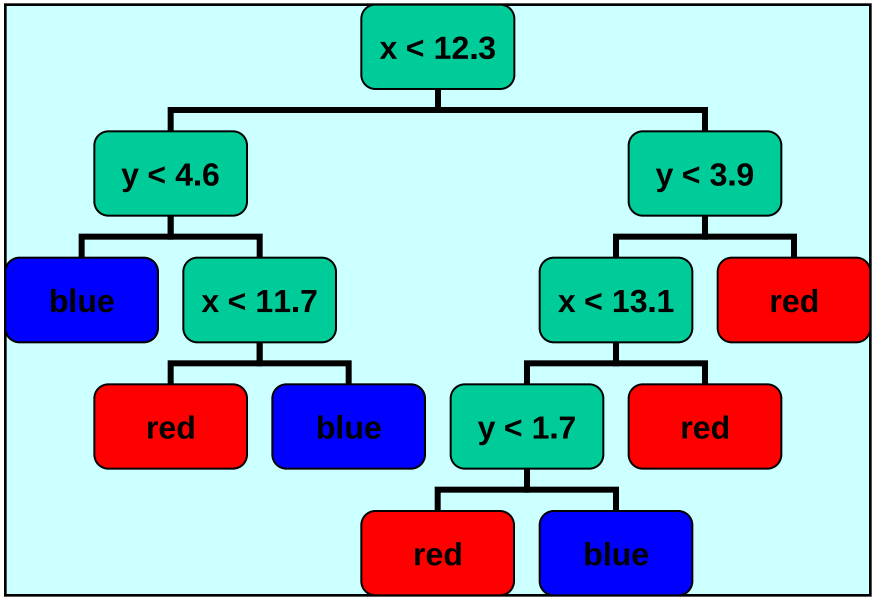
- Training: recursively split feature space into (two or more) sub-spaces at each step
  - Different strategies for evaluating splits

- Classification: traverse through the tree from the root node, until reaching a leaf node
  - Compute prediction based on leaf node instances
    - Majority voting of the leaf node instances

Decision Trees: Example

- 2-dimensional data (x, y), numerical values, two classes





Decision Trees: Algorithm

- Test data in each leaf node
- If not all from the same class *(or other stopping criterion)*
  - For ***each*** attribute
    - Identify possible splits of samples into (two or more) subspaces
  - Compute ***best*** split (over all attributes!)
    - Based on a split goodness measure/criterion
- Until data in all leaf nodes is *pure* (same class)
  - Or cannot be distinguished (*When can this happen?)*
  - *Or other stopping criterion fullfilled (e.g. maximum depth)*

Decision Trees: Algorithm

- For ***each*** attribute
  - Identify possible splits of samples into (two or more) subspaces
    - categorical variables? (e.g. size with values “small” / “medium” / “large”)
      - *By each variable value, i.e. split into 3 sub-branches*
      - *Or one value vs. other values: small vs. rest, medium vs rest, large vs. rest (split into 2 sub-branches)*
      - *Difference?*
    - numerical variables? (e.g. size in centimeters)
      - *sort values & split between each pair of values*
      - *➔ How many candidate splits?*

Decision Trees: Algorithm

- Popular measures to compute best split
  - Error rate
  - Information gain
  - Gini impurity (Gini index)
  - *Variance reduction*
  - *...*

Decision Trees: Algorithm

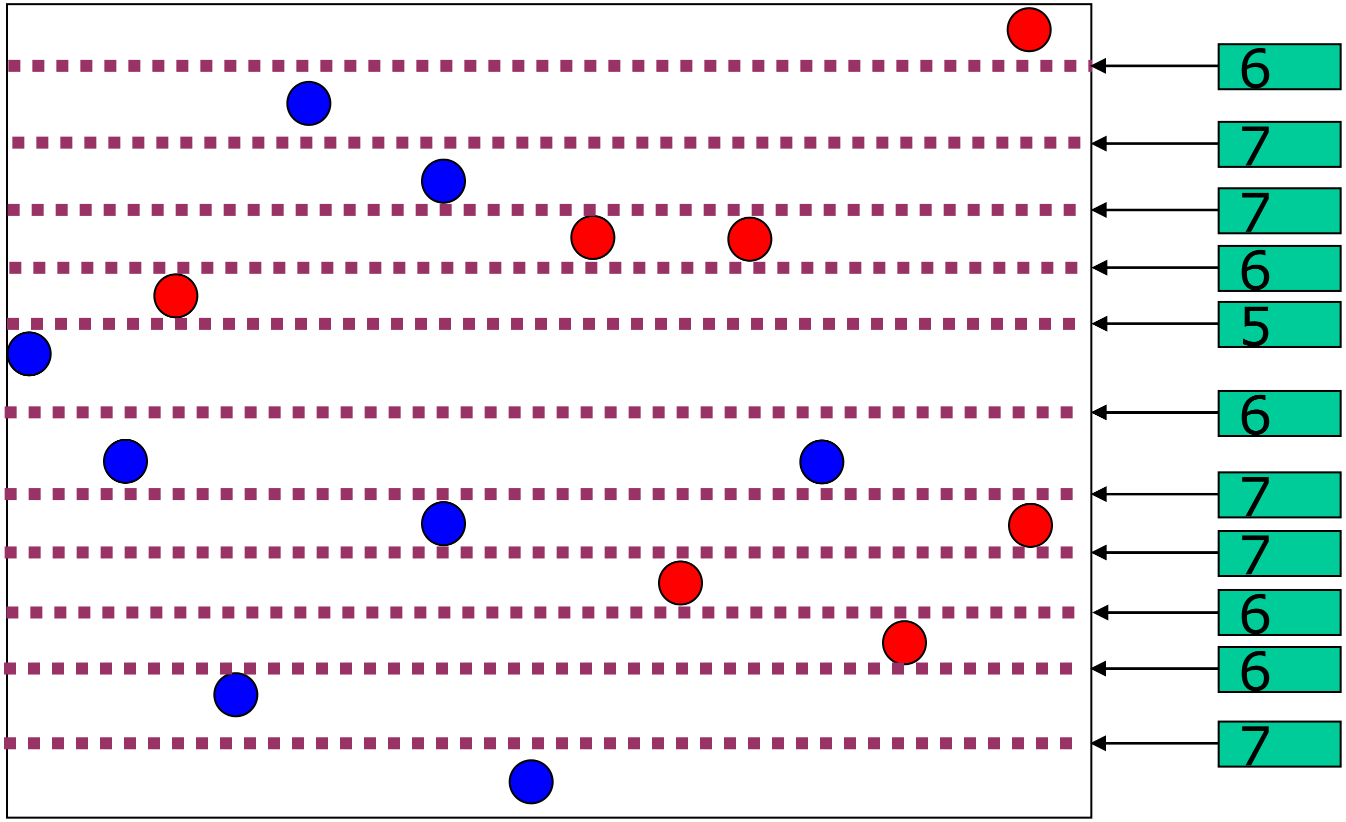
- Popular measures to compute best split
  - Error rate
  - Information gain
  - *Gini impurity (Gini index)*

Decision Trees: Algorithm

- Error rate
  - Related to accuracy (“complement“)
  - Absolute error rate: simply count the number of classification errors when performing a split
    - *How to express in TP/TN/FP/FN ?*
  - The lower the error, the better
  - Absolute error rate vs. relative error rate
    - Relative error in relation to total number of samples (0..1)
    - Absolute error: 0..n
    - Decision trees: often absolute error rate used
      - *Semantic difference?*

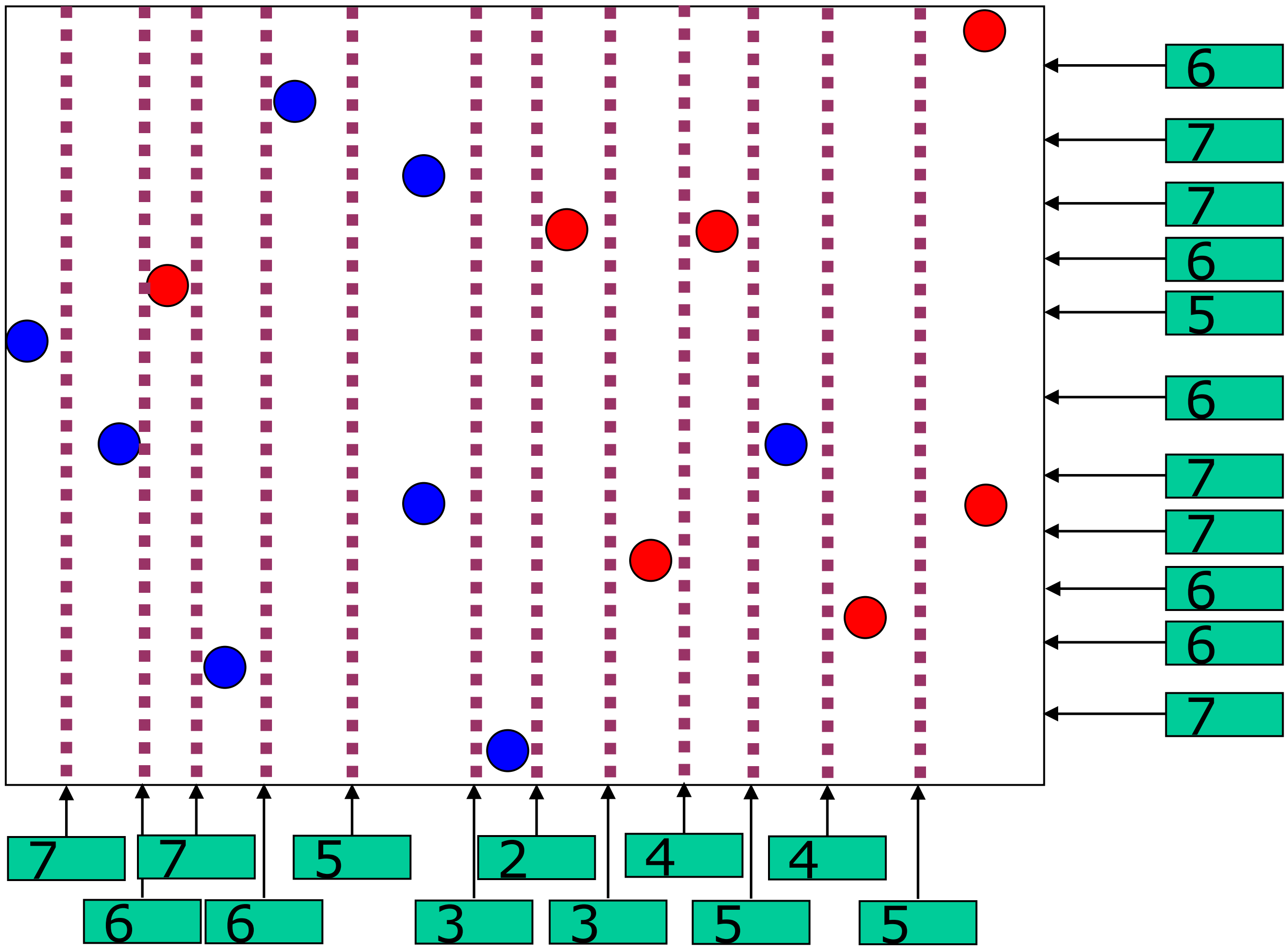
Decision Trees: error rate

- 2-dimensional data (x, y), numerical values, two classes



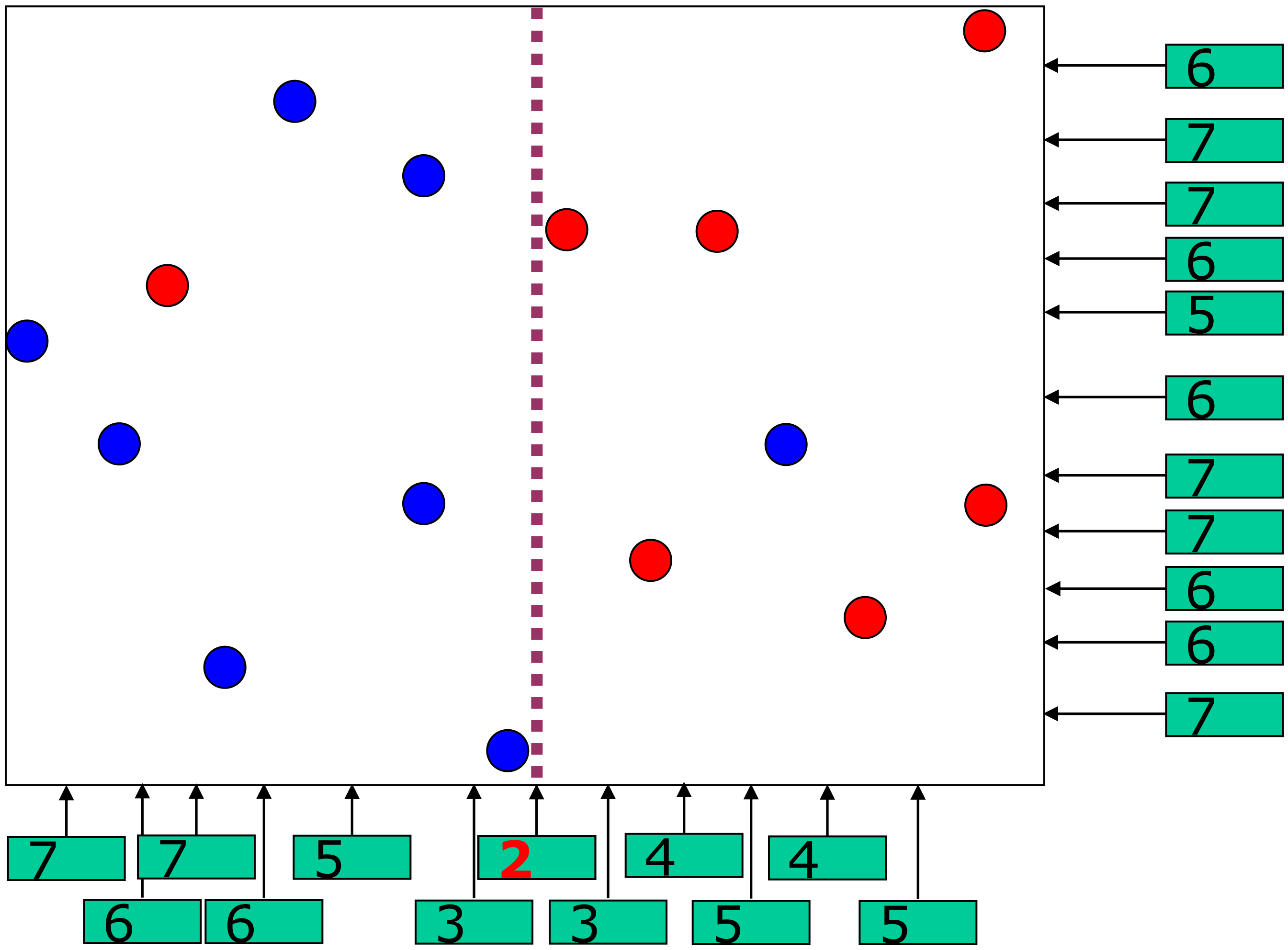
Decision Trees: error rate

- 2-dimensional data (x, y), numerical values, two classes

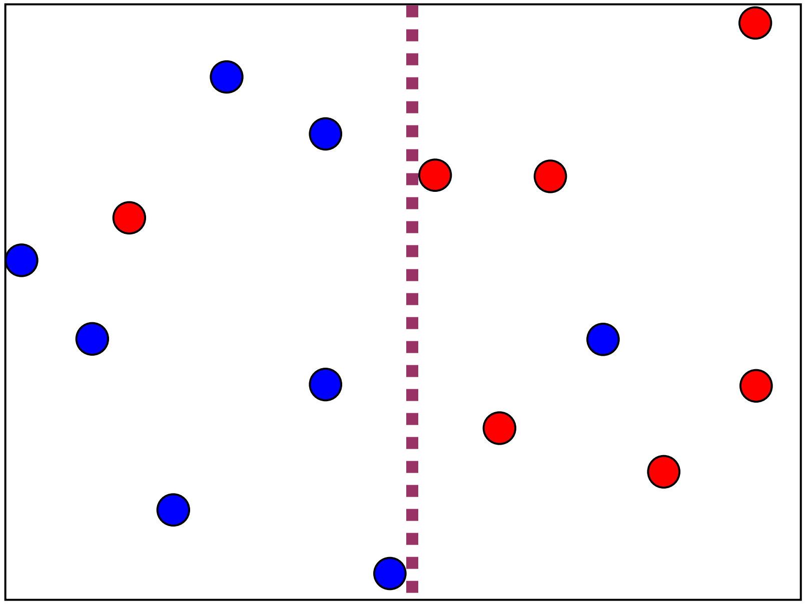


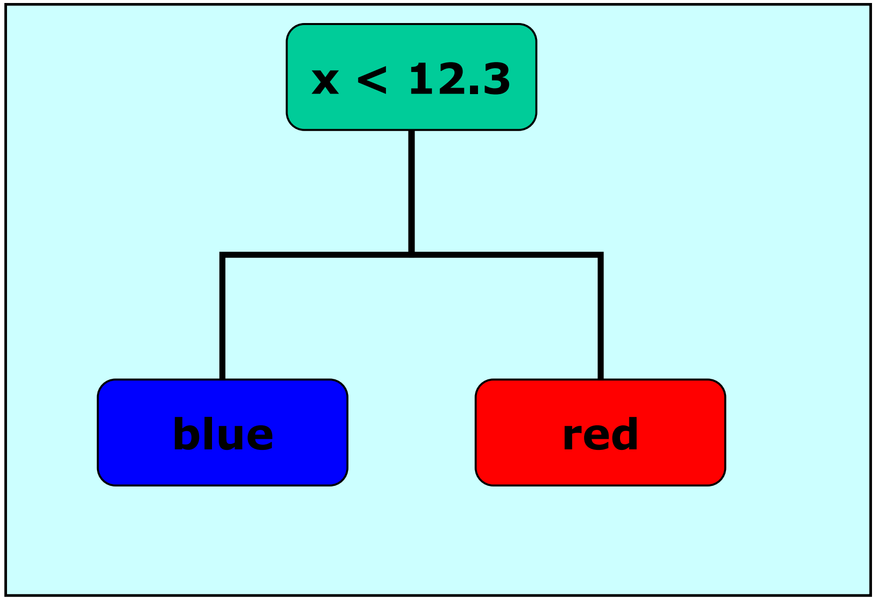
Decision Trees: error rate

- 2-dimensional data (x, y), numerical values, two classes



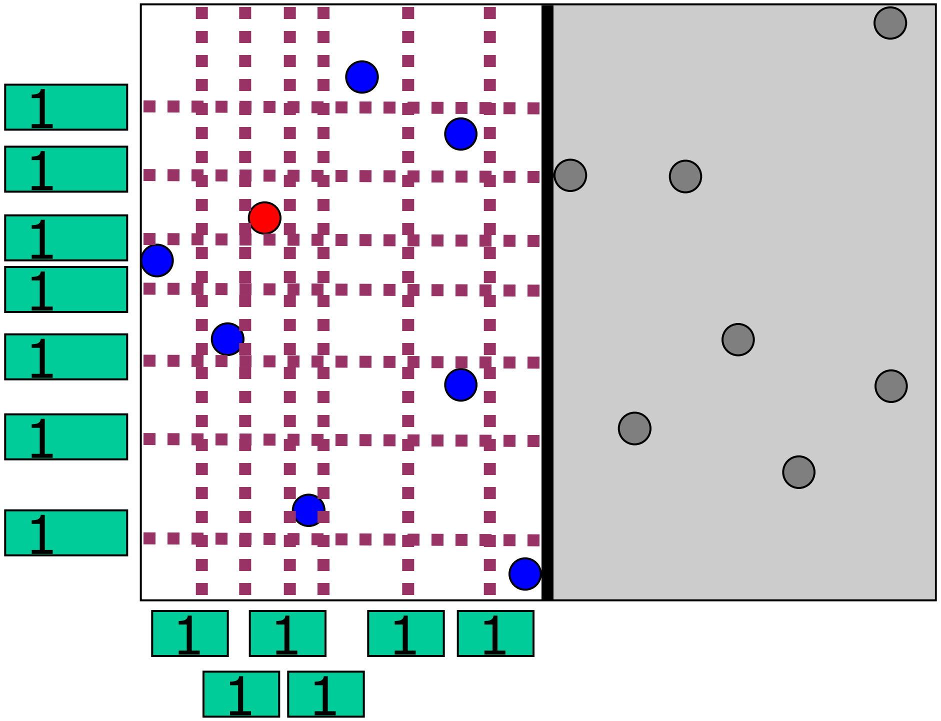
Decision Trees: error rate

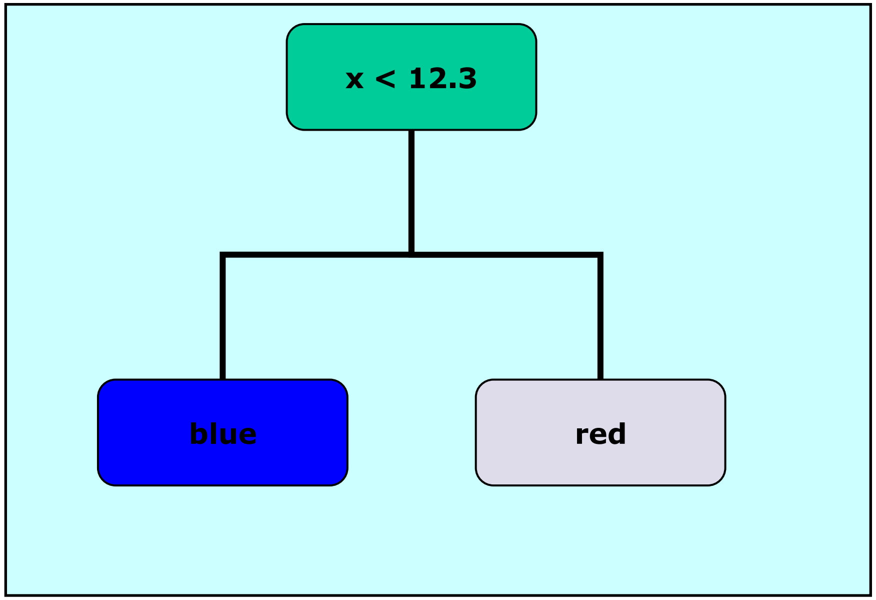




- Split for top-level (root) node found ✓
- *If we stop here – how is this classifier called?*
- *Next step(s)?*

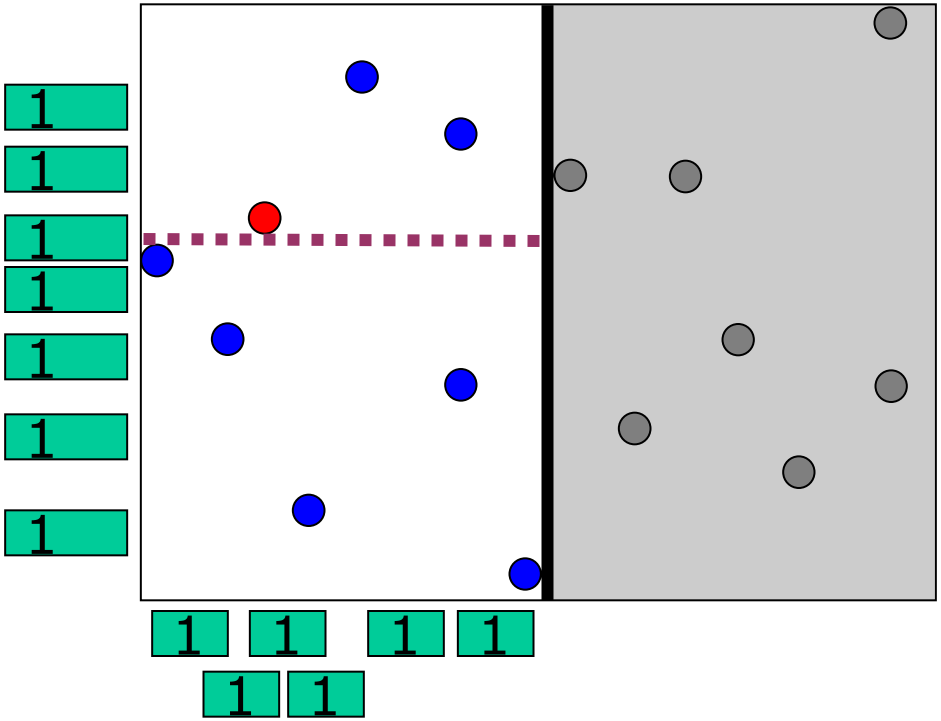
Decision Trees: error rate

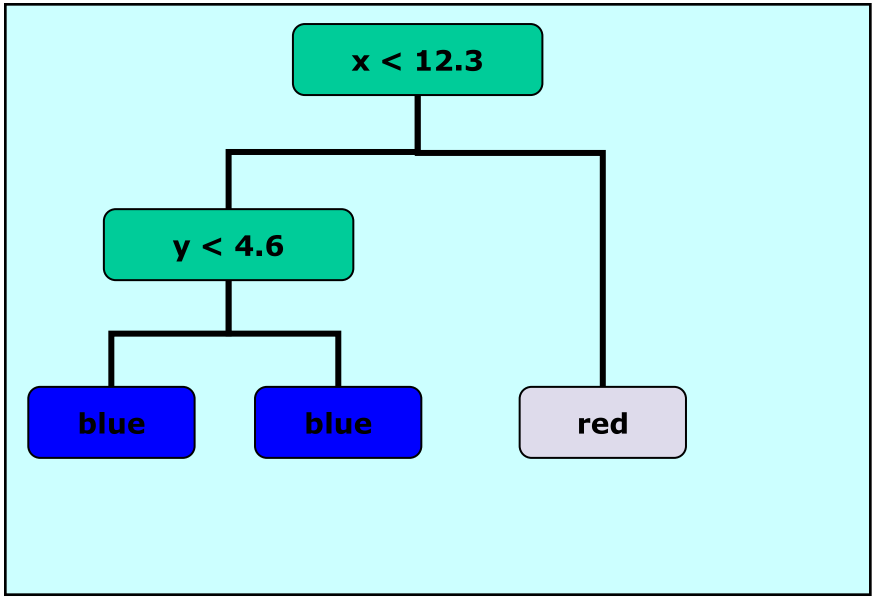




Which split?

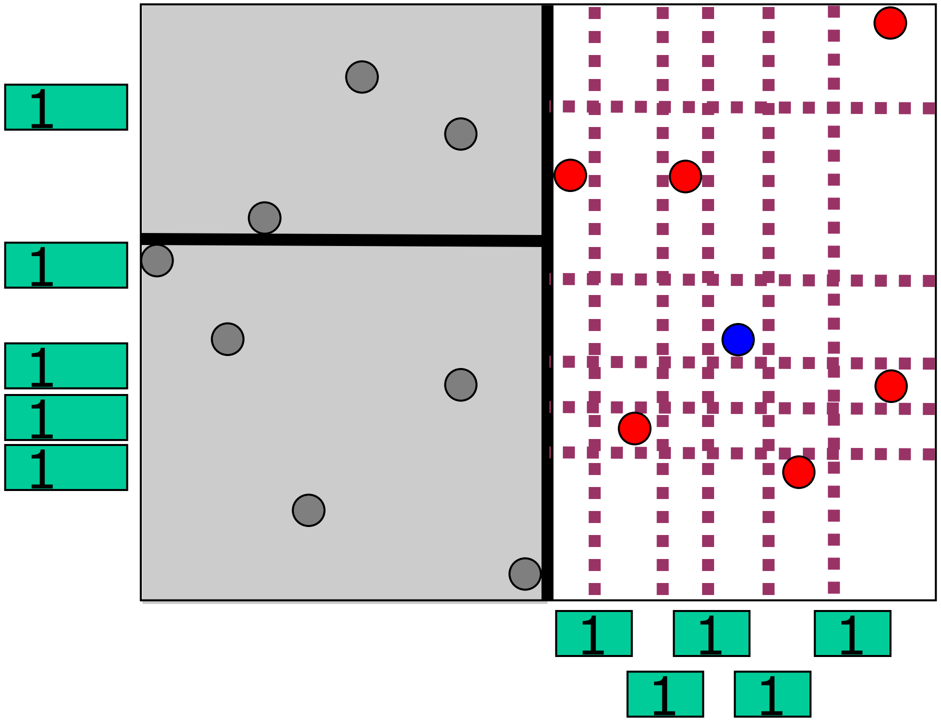
Decision Trees: error rate

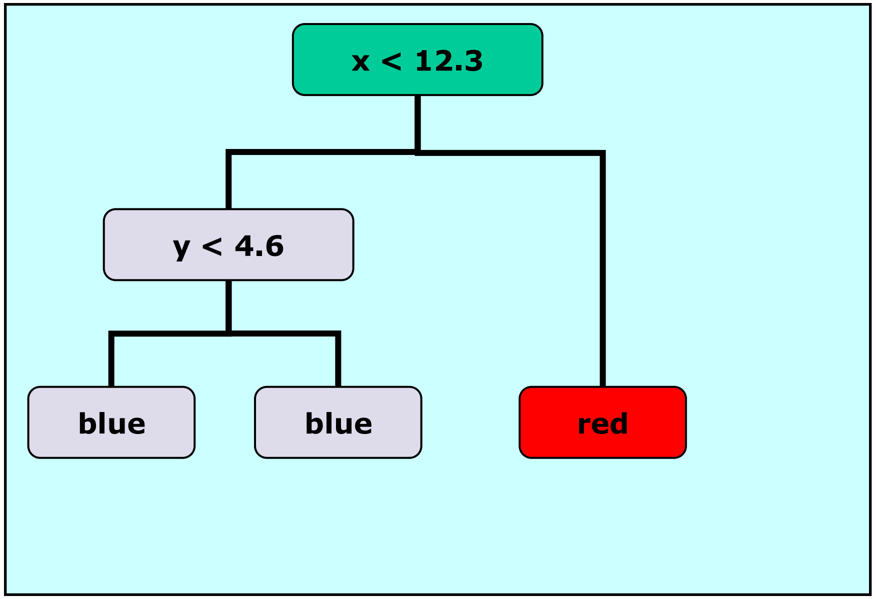




Decision for split in case of equality – e.g. random

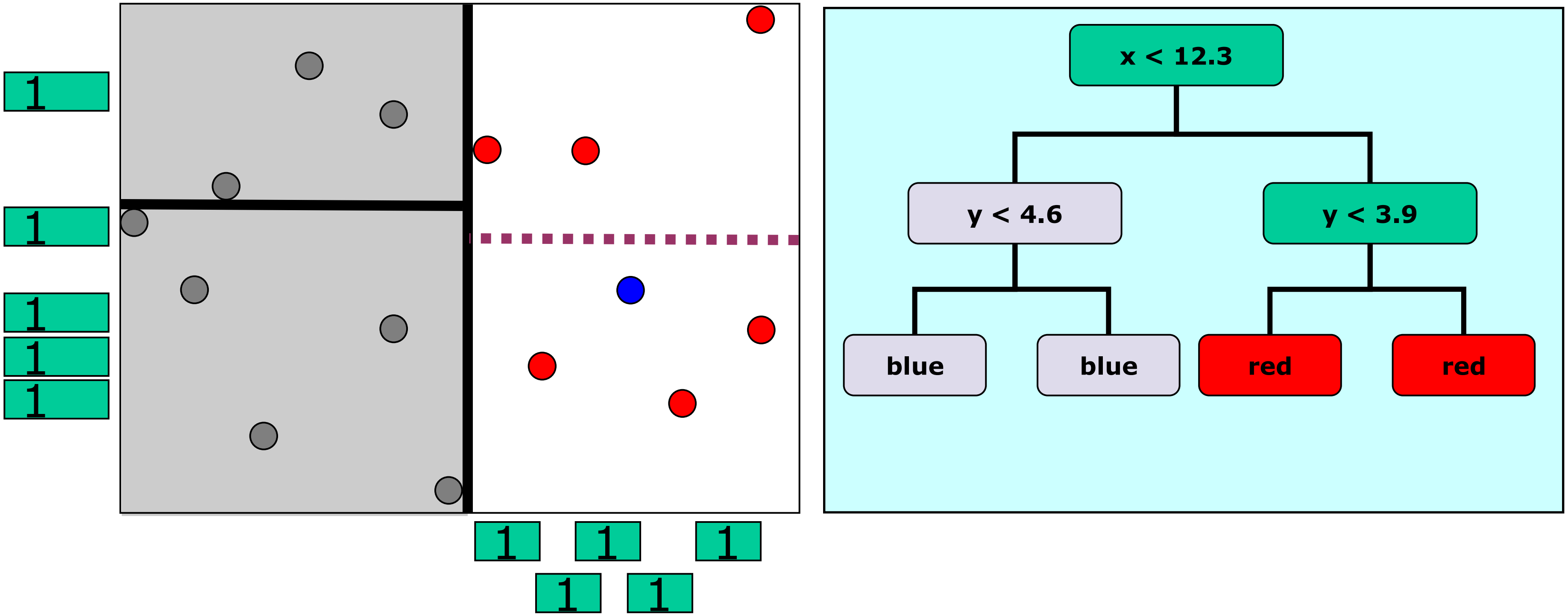
Decision Trees: error rate





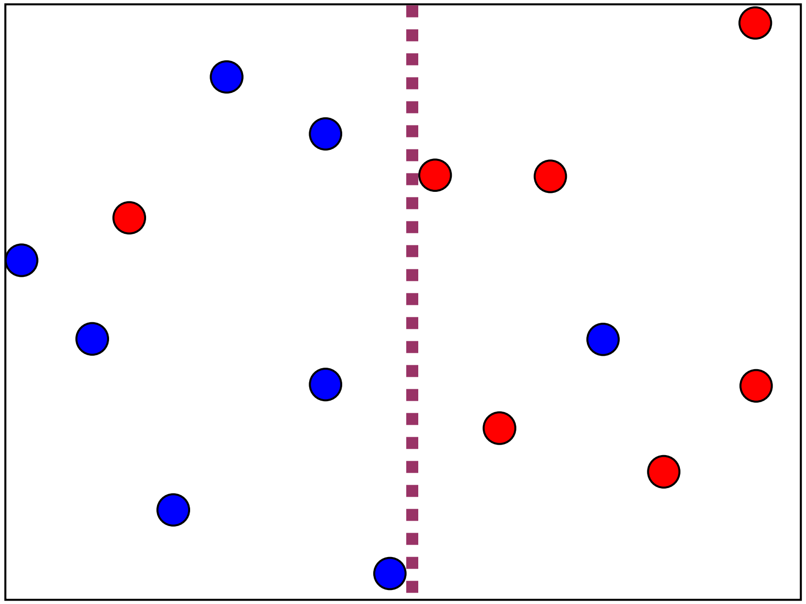
Which split?

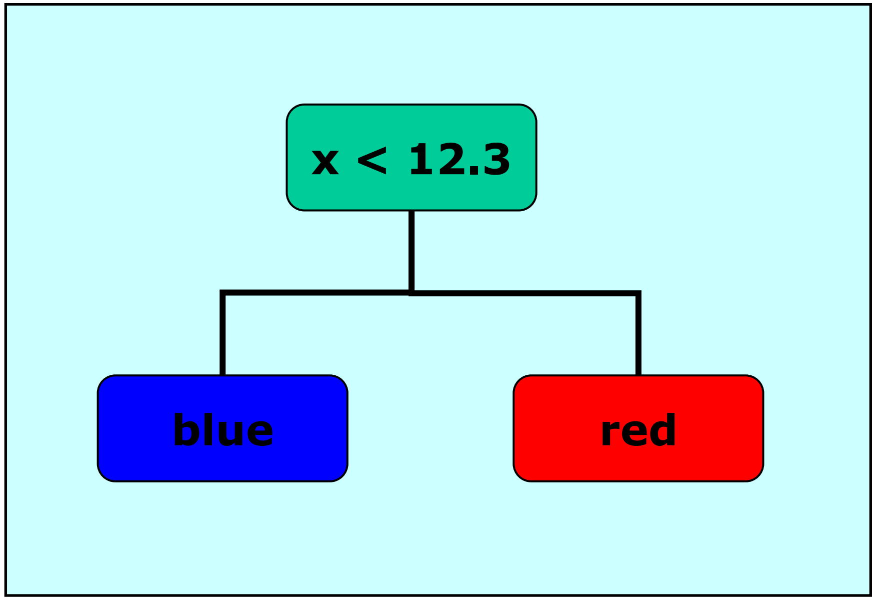
Decision Trees: error rate



Decision Trees: error rate

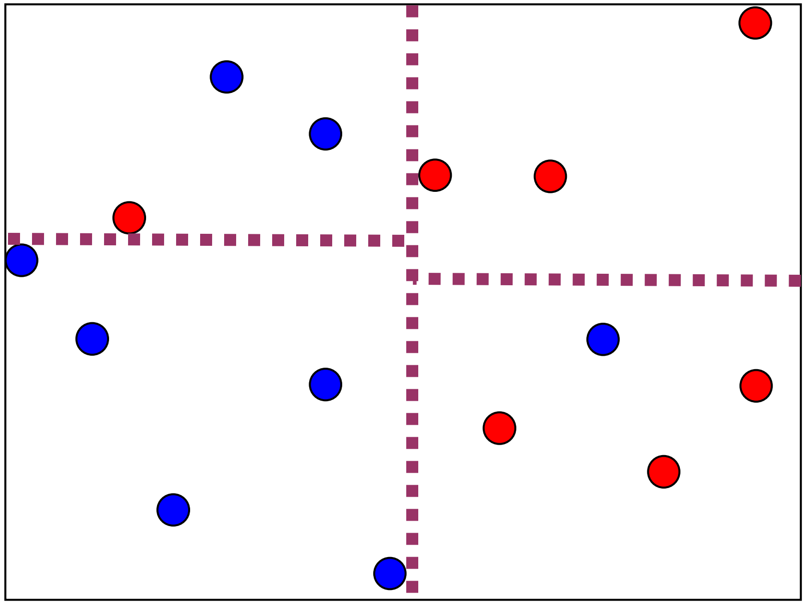
- Tree training, level 1

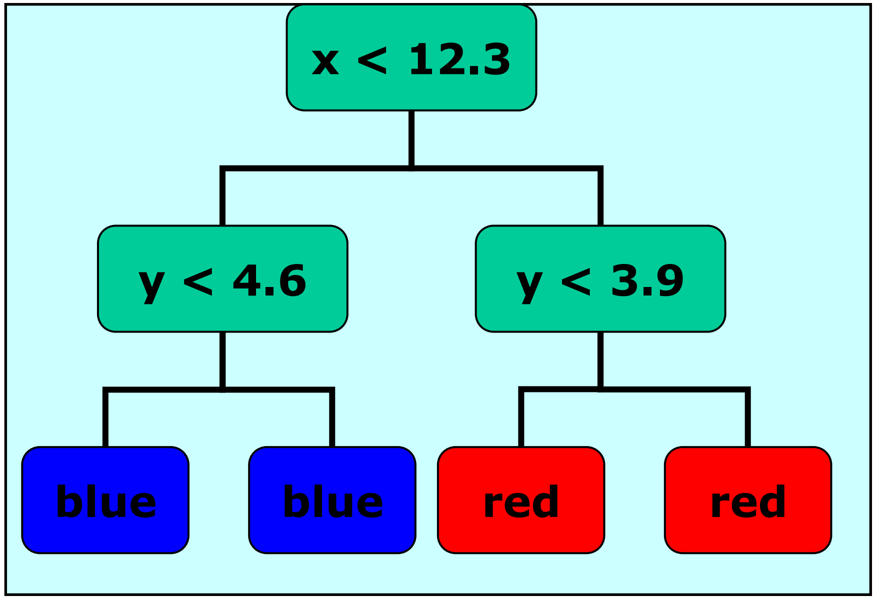




Decision Trees: error rate

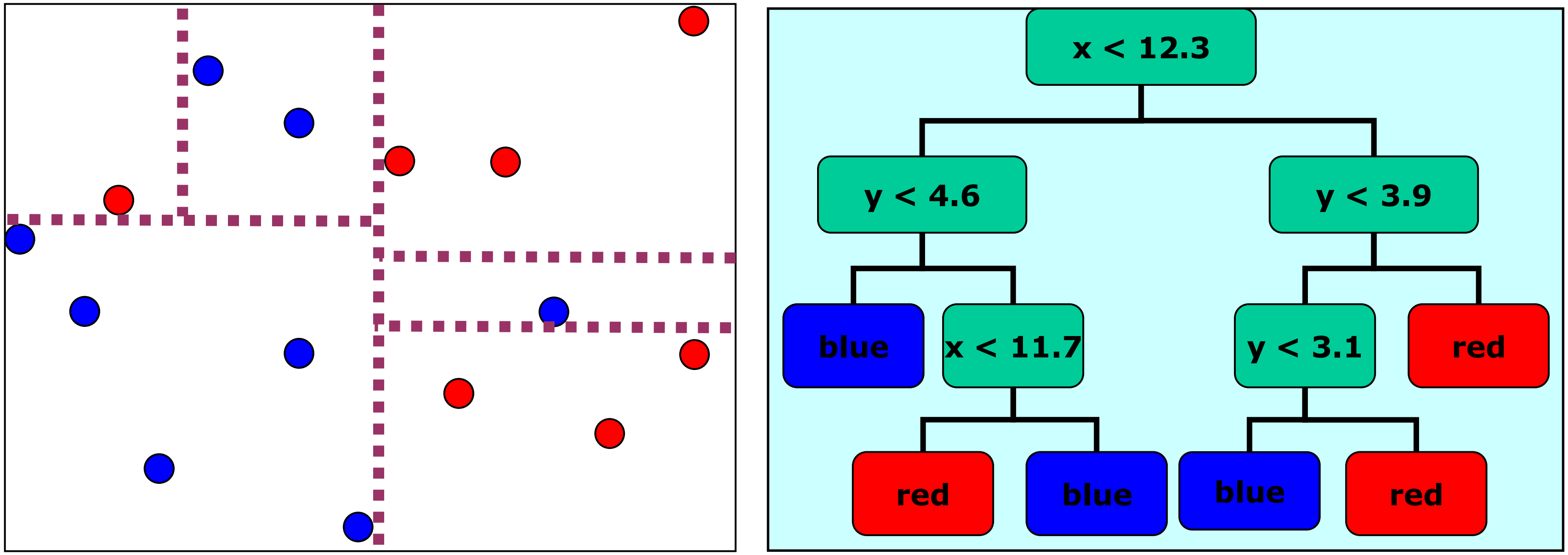
- Tree training, level 2



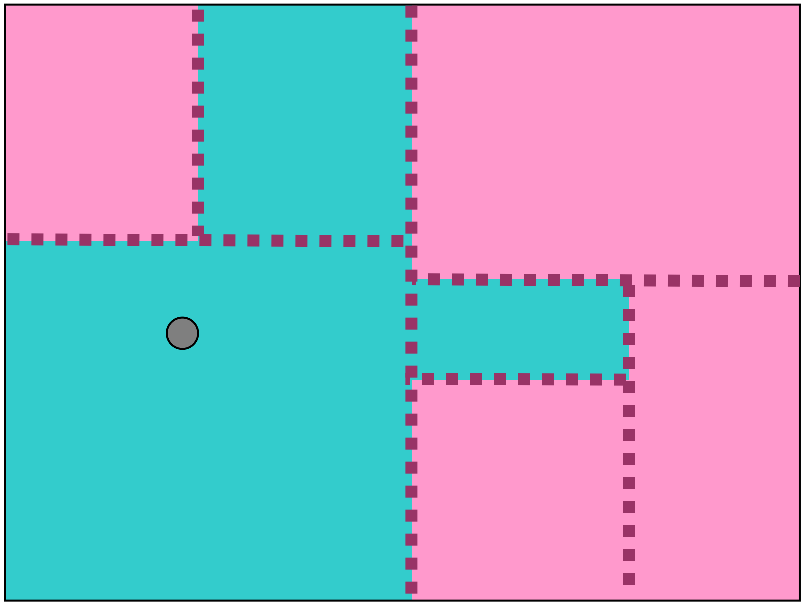


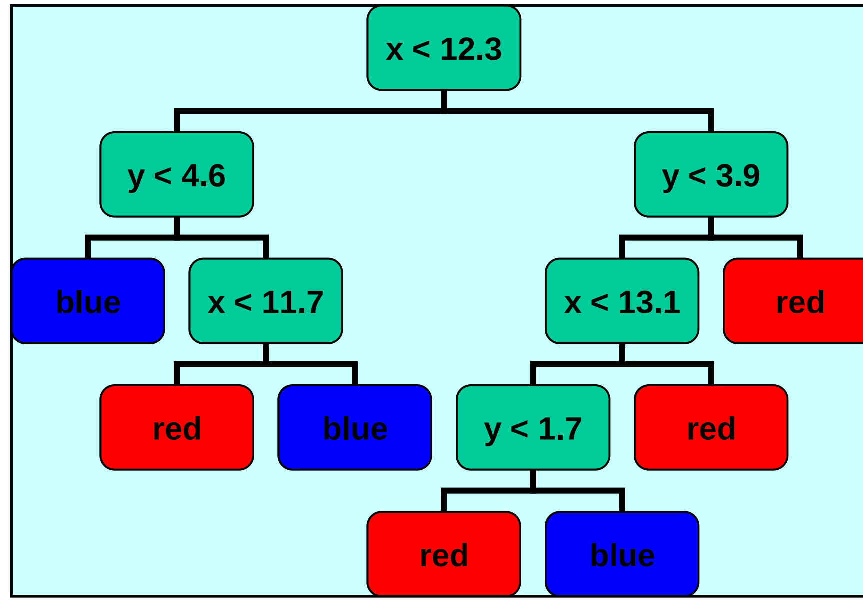
Decision Trees: error rate

- Tree training, level 3: completely built tree



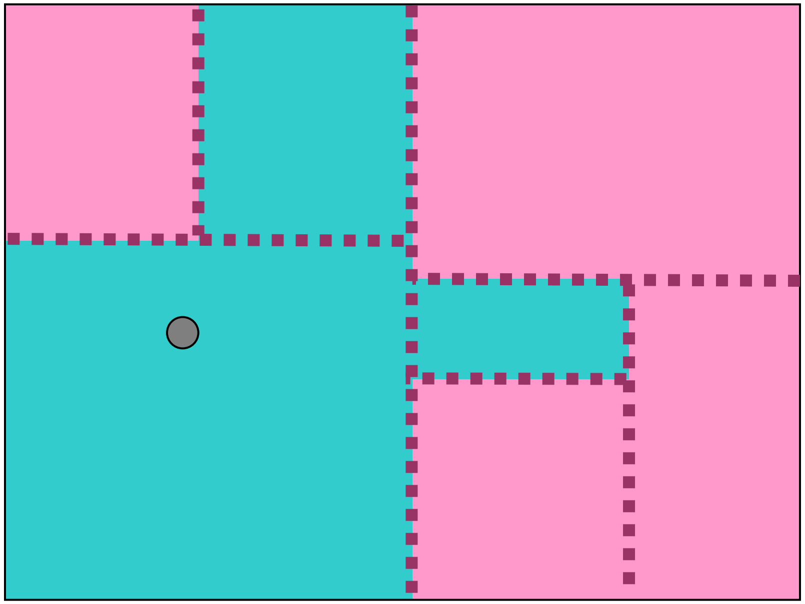
Decision Trees: Classification

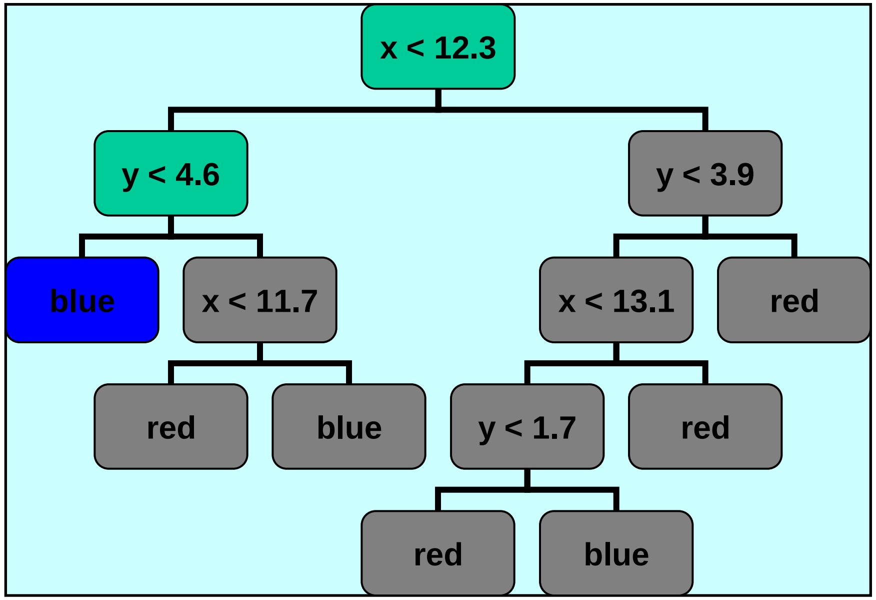




How to classify unknown items?

Decision Trees: Classification

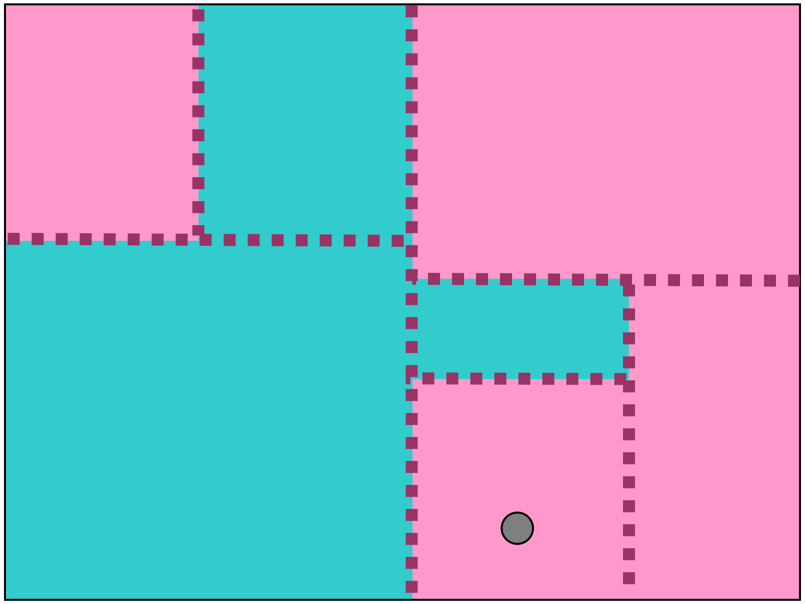


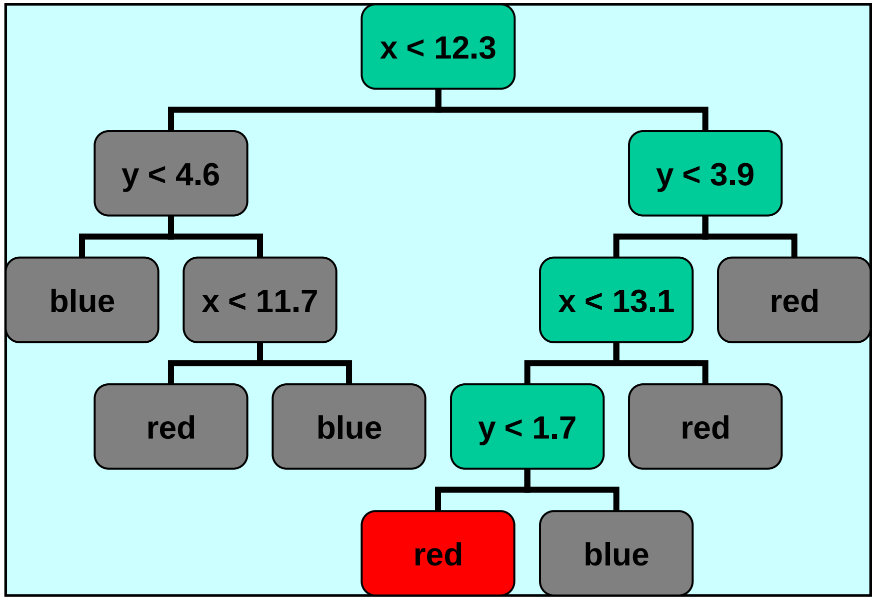


➔ Decend the tree until leaf-node

➔ Use majority of class in that leaf node

Decision Trees: Classification





➔ Decend the tree until leaf-node

➔ Use majority of class in that leaf node

Decision Trees: Algorithm

- Popular measures to compute best split
  - Error rate
  - Information gain
  - *Gini impurity (Gini index)*

Information Theory & Entropy

- Introduced by Claude Shannon (1948)
  - Original for compression & reliable communication
  - Applications in statistical inference, NLP, cryptography,…
- Entropy: # of bits needed for communication
  - Absolute limit for best lossless compression
- Measure of uncertainty
  - High probability – low entropy
- Concerned with measuring actual information vs. redundancy

What is „Information Entropy"?

- Entropy – measure of uncertainty

- ML: measure for the "impurity" of a set
  - High Entropy ➔ bad for prediction
  - High Entropy ➔ needs to be reduced

$$H(X) = \mathrm{E}(\mathrm{I}(X)) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

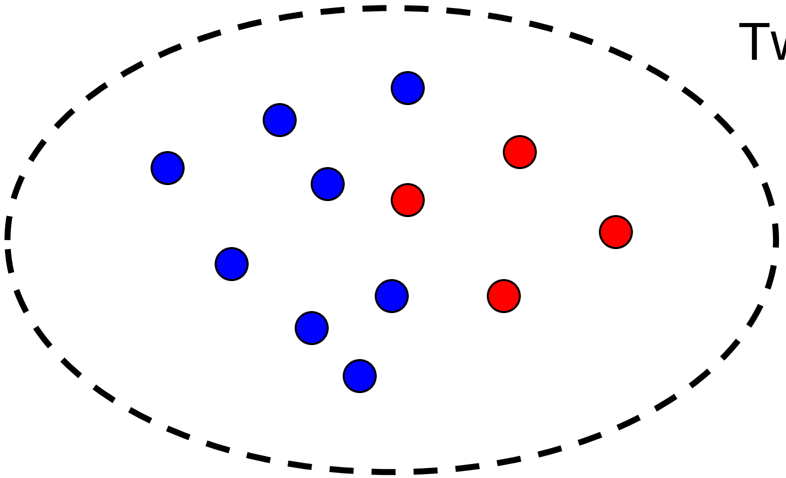
H … Entropy

E … Expected value

I(X) …information content of X

p(...) … probability function

Calculating $H(X)$: example



Two dimensional data, two classes

$$p(x_{\text{red}}) = \frac{4}{12} = 0.33$$

$$p(x_{\text{blue}}) = \frac{8}{12} = 0.67$$

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

$$H(X) = -p(x_{red})\log_2 p(x_{red}) - p(x_{blue})\log_2 p(x_{blue})$$

Remember:

$$\log_2(x) = \log(x)/\log(2)$$

$$H(X) = \frac{1}{3}\times\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\times\log_2\left(\frac{2}{3}\right)$$

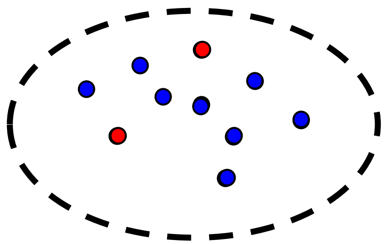
$$= -\frac{1}{3}\times -1.58 \quad - \quad \frac{2}{3}\times -0.58$$

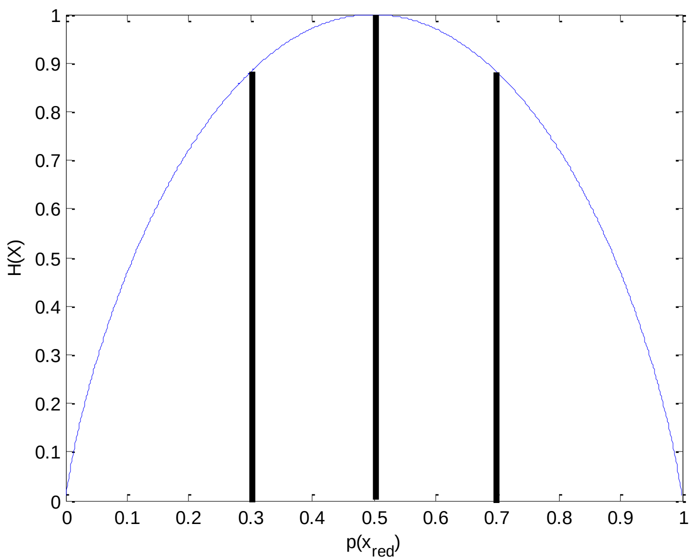
$$= 0.53 + 0.39$$

$$= 0.92$$

FACULTY OF !INFORMATICS

$H(X)$: Example values



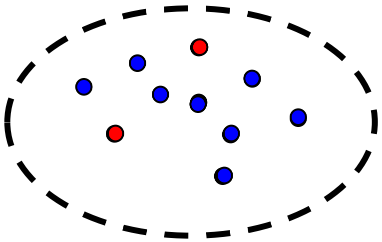


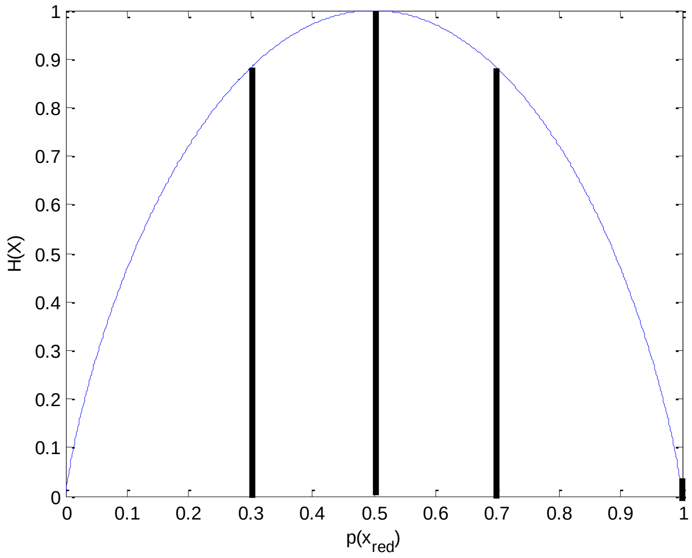
	$p(x_{red})$	$p(x_{blue})$	$H(X)$
I	0.5	0.5	?
II	0.3	0.7	?
III	0.7	0.3	?
IV	0	1	?

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Any guesses?

$H(X)$: Example values



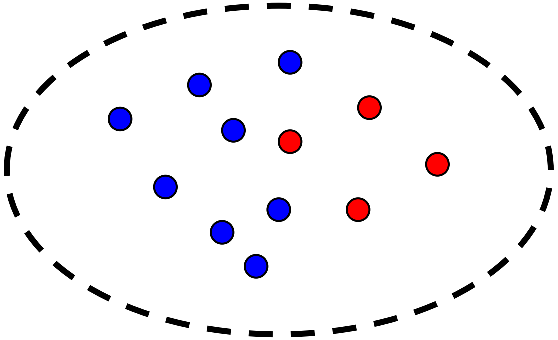


	$p(x_{red})$	$p(x_{blue})$	$H(X)$
I	0.5	0.5	?
II	0.3	0.7	?
III	0.7	0.3	?
IV	0	1	?

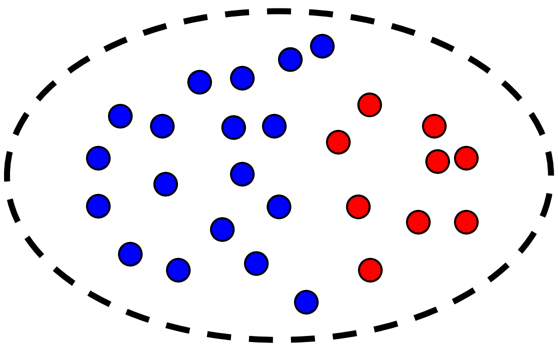
$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

$H(X)$: Relative vs. absolute frequencies

What are the entropies of I and II?



vs.



$$p(x_{\text{red, I}}) = \frac{4}{12} = 0.33 \,;\; p(x_{\text{blue, I}}) = \frac{8}{12} = 0.67$$

$$p(x_{\text{red, II}}) = \frac{9}{27} = 0.33 \,;\; p(x_{\text{blue, II}}) = \frac{18}{27} = 0.67$$

Dataset	red	blue
I	8	4
II	18	9

=> $H(X_I) = H(X_{II})$

Only **relative** frequencies matter!

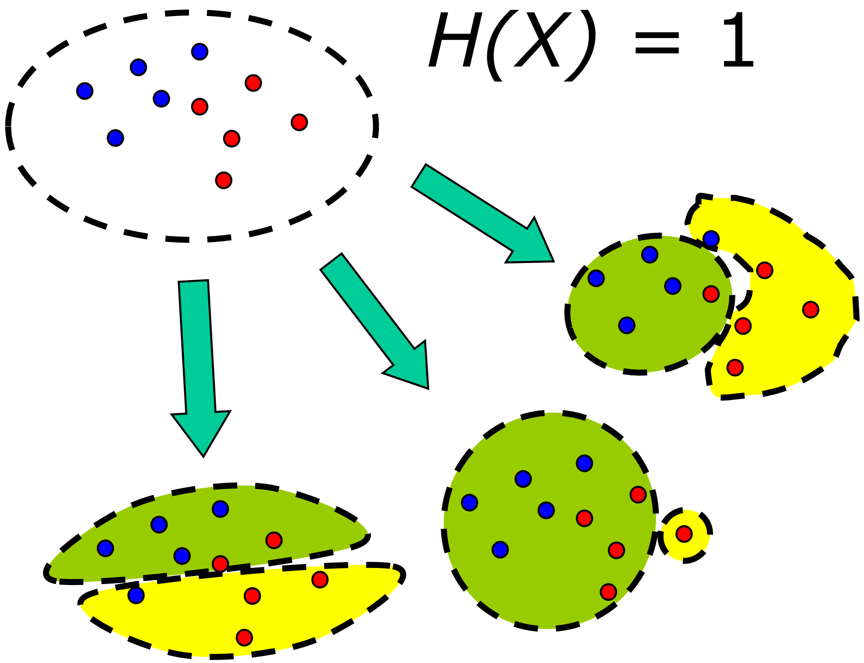
Information Gain

Given a set and a choice between possible subsets, which one is preferable?

Information Gain: Sets that minimize Entropy by largest amount

$$IG(X_A, X_B) = H(X) - p(x_A)H(X_A) - p(x_B)H(X_B)$$

$H(X) = 1$



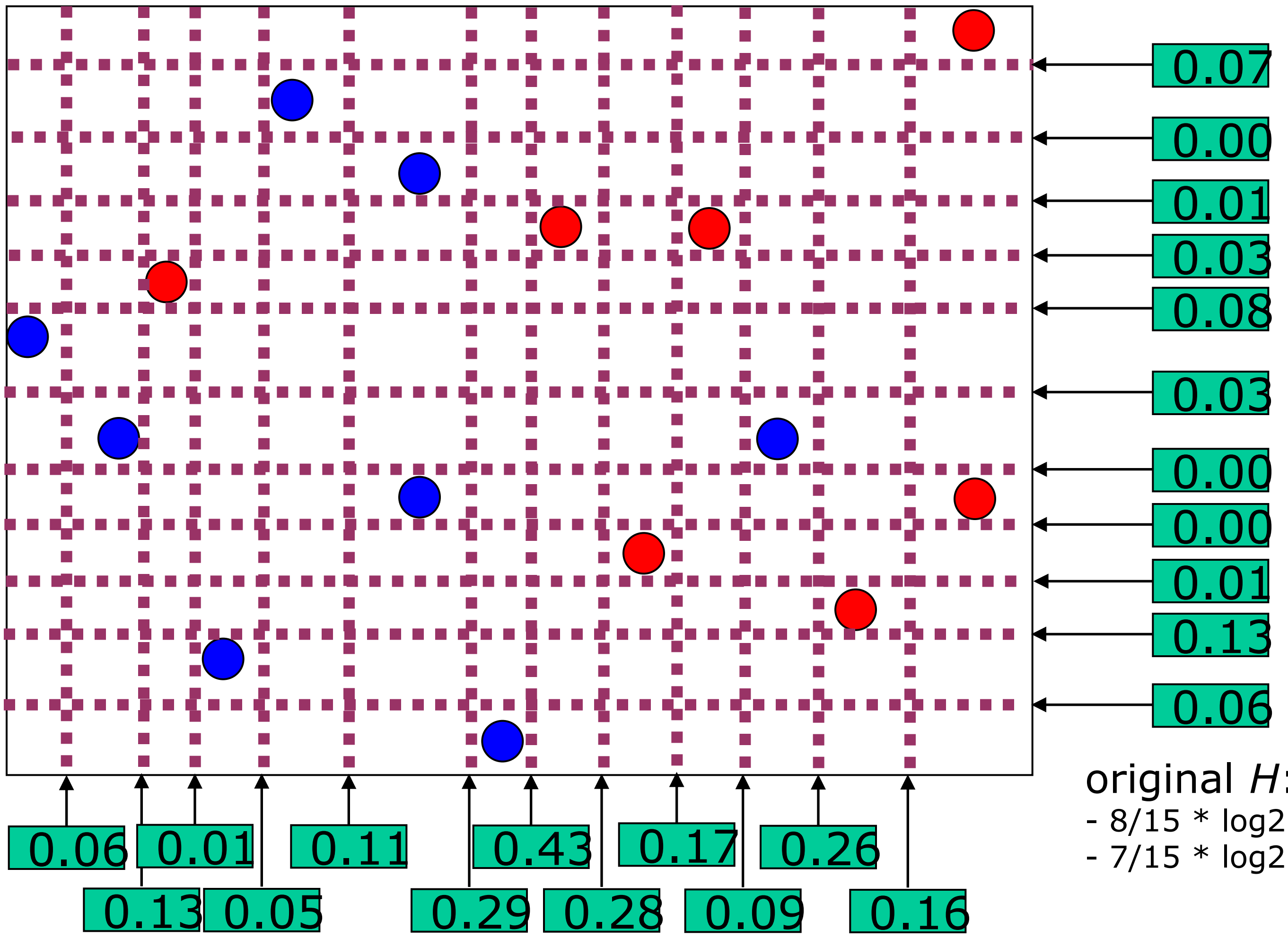
	A (green)	B (yellow)
Points	5 / 9	5 / 4 / 1
p(X.)	0.5 / 0.9	0.5 / 0.4 / 0.1
$p(x_{red})$	0.2 / 0.43 / 0.44	0.75 / 1 / 0.8
$p(x_{blue})$	0.56 / 0.67 / 0.8	0.25
H(X.)	0.92 / 0.99 / 0.72	0.81 / 0.72
IG	0.189 / 0.209 / 0.124 [illegible]	

Information Gain (Properties)

- Information Gain is
  - the amount by which the original Entropy can be reduced by splitting into subsets
  - *Min/max bounds of Information gain?*
  - at most as large as the Entropy of the undivided set
  - at least zero (if Entropy is not reduced)

- $0 \leq IG \leq H(X)$

Decision Trees: information gain



original H: 0.99679

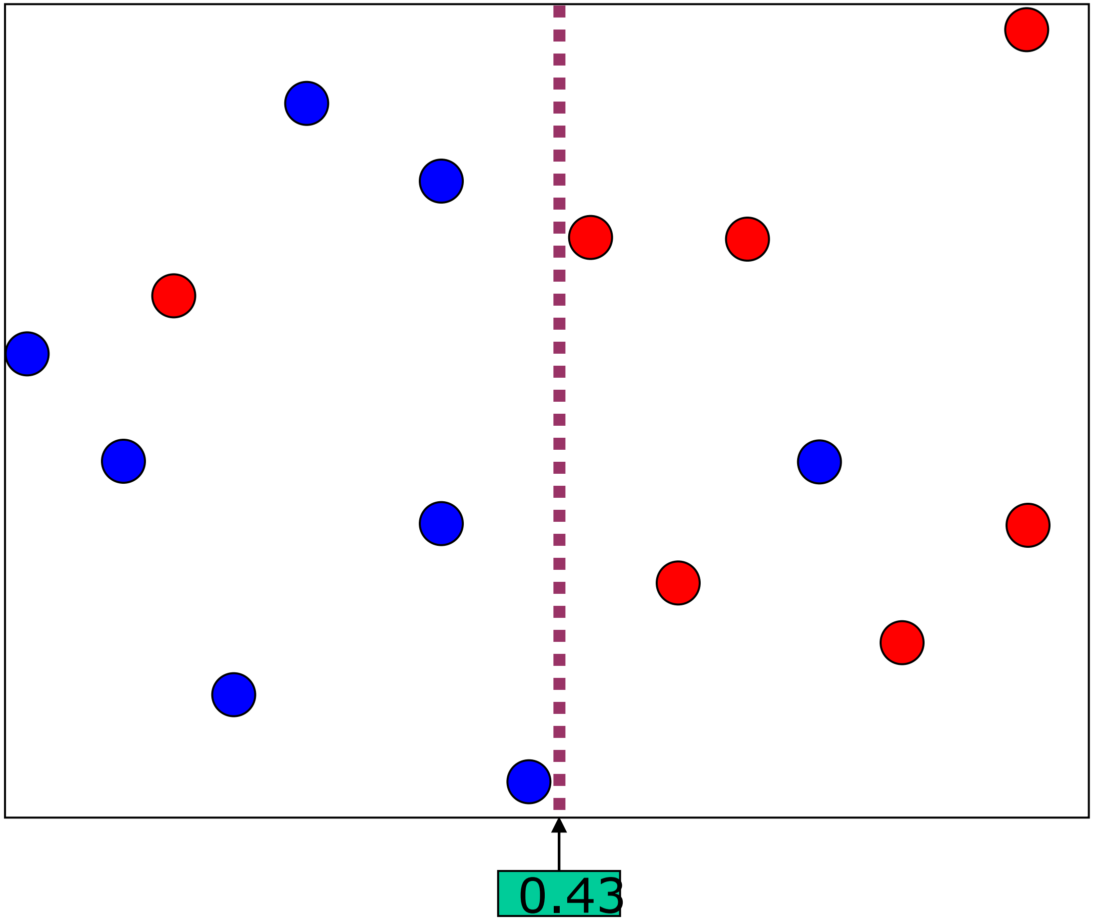
- 8/15 * log2(8/15)

- 7/15 * log2(7/15)

Decision Trees: information gain

H(left) =
$-0.125\log_2 0.125 - 0.875\log_2 0.875 =$
$0.375 + 0.169 = 0.54356$

H(right) =
$-0.143\log_2 0.143 - 0.857\log_2 0.857 =$
$0.401+0.191 = 0.59167$



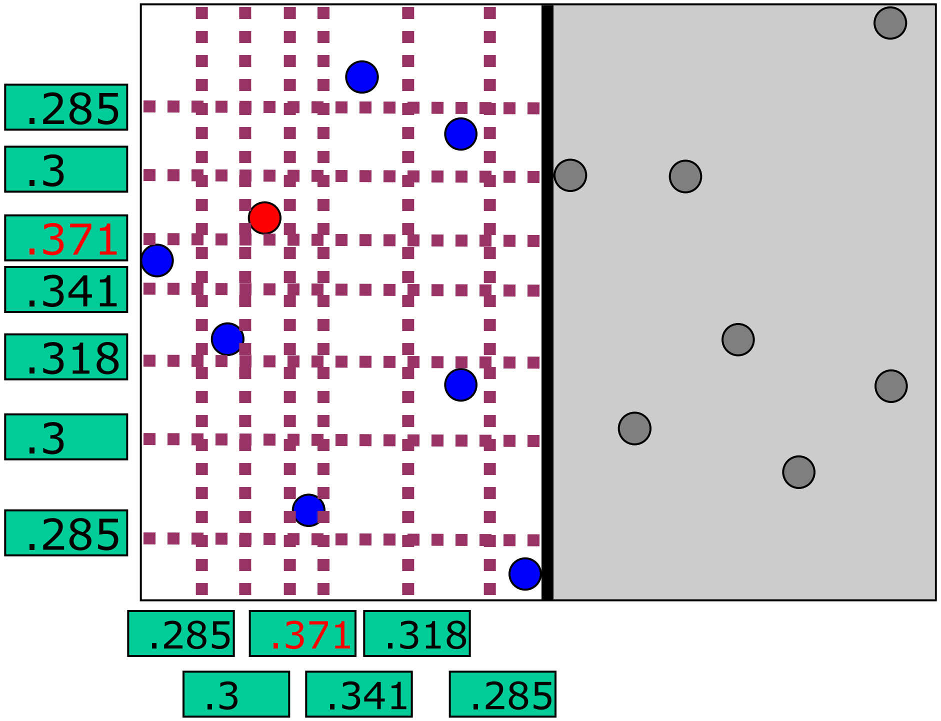
H(split) =
0.54356*8/15 +
0.59167*7/15 =
0.566011333

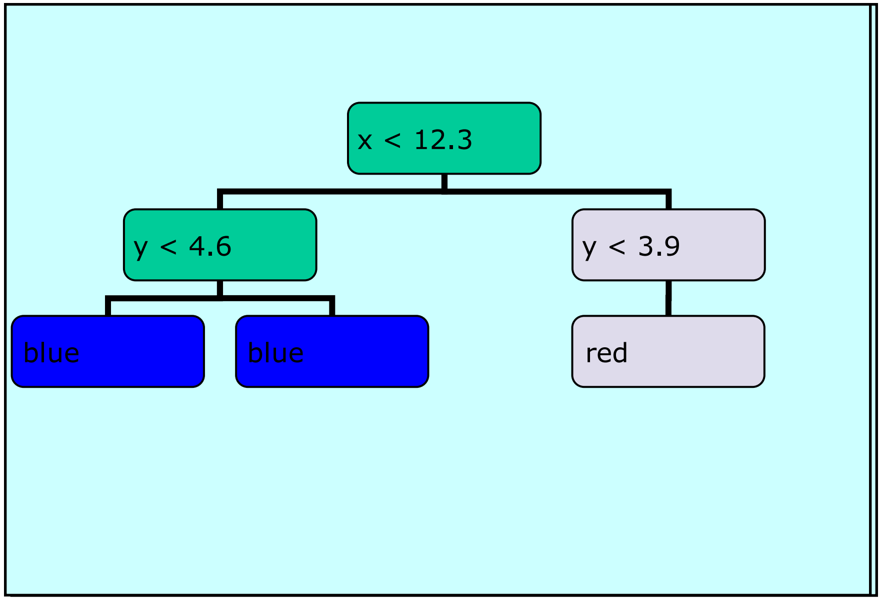
original *Entropy*:
H(x) = 0.99679

- 8/15 * log2(8/15)
- 7/15 * log2(7/15)

IG = 0.43078

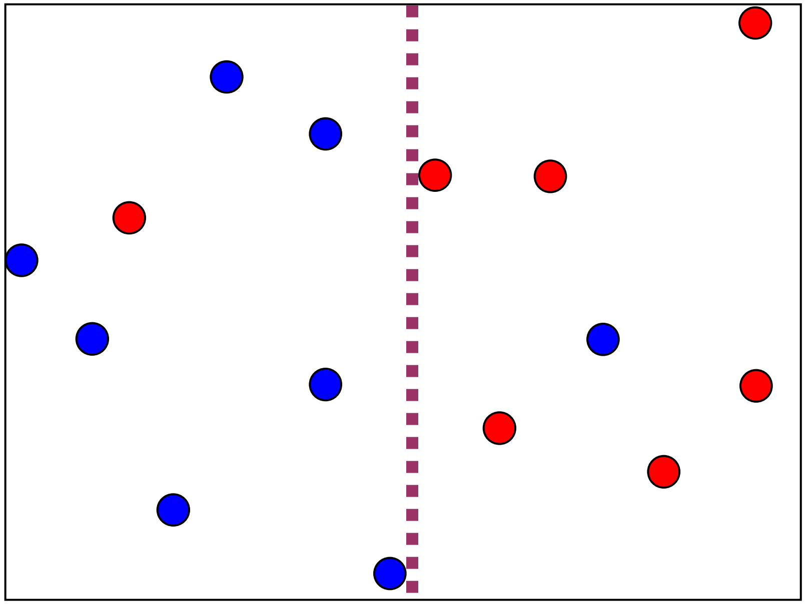
Decision Trees: information gain

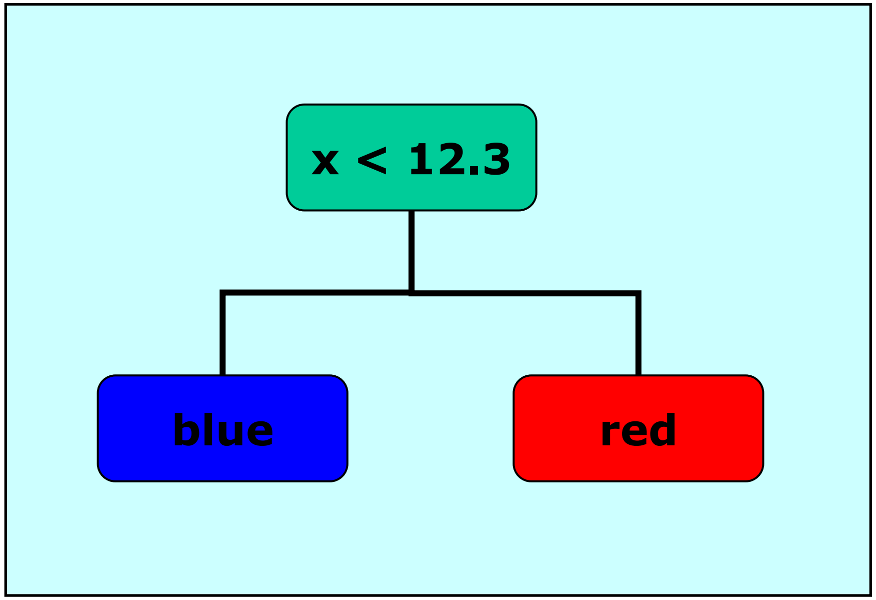




Decision Trees: information gain

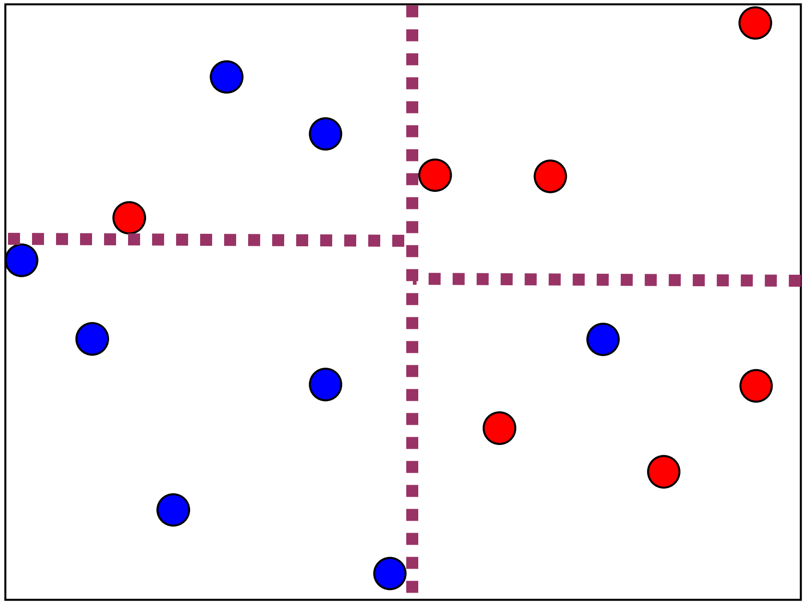
- Tree training, level 1

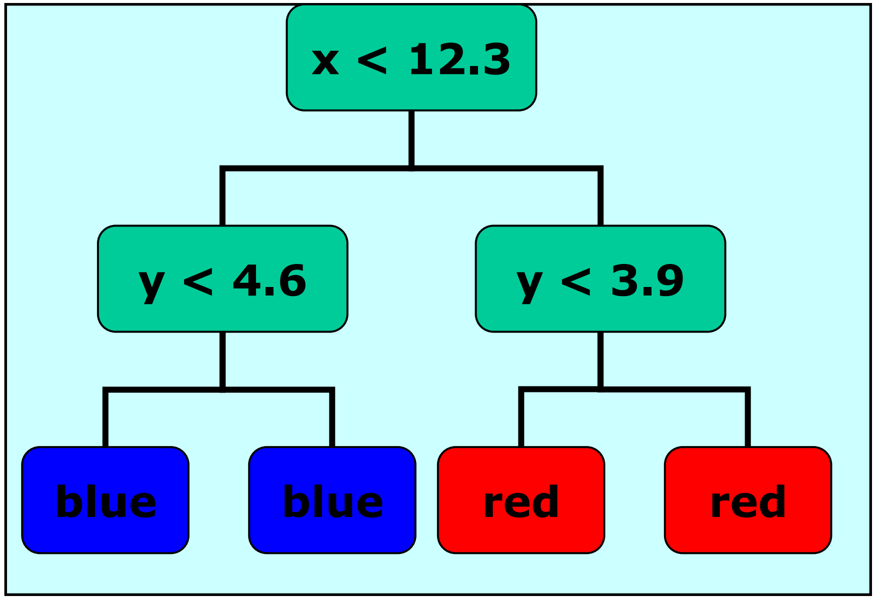




Decision Trees: information gain

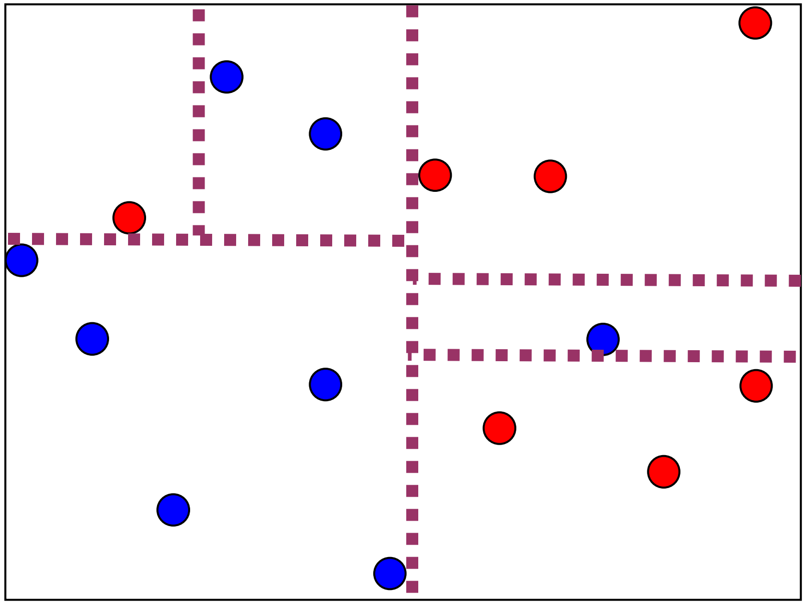
- Tree training, level 2

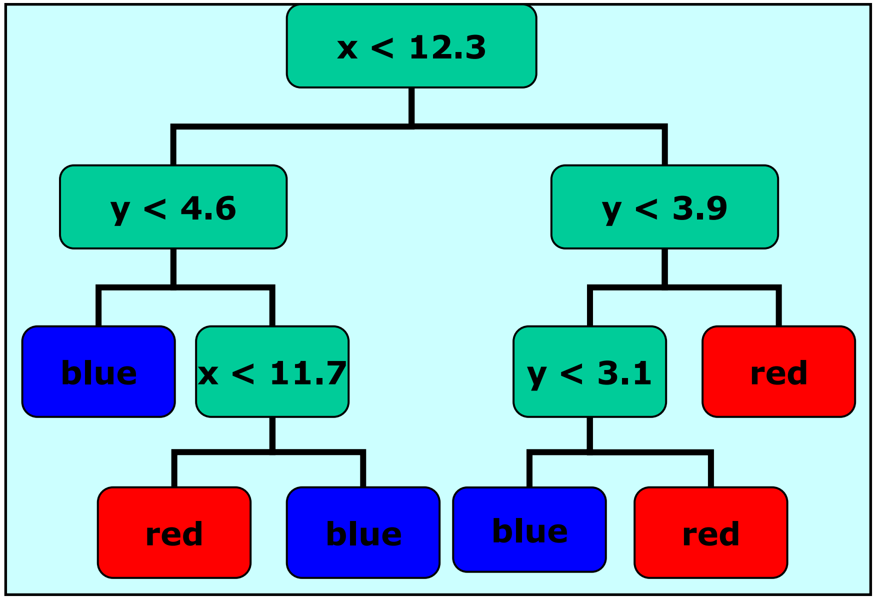




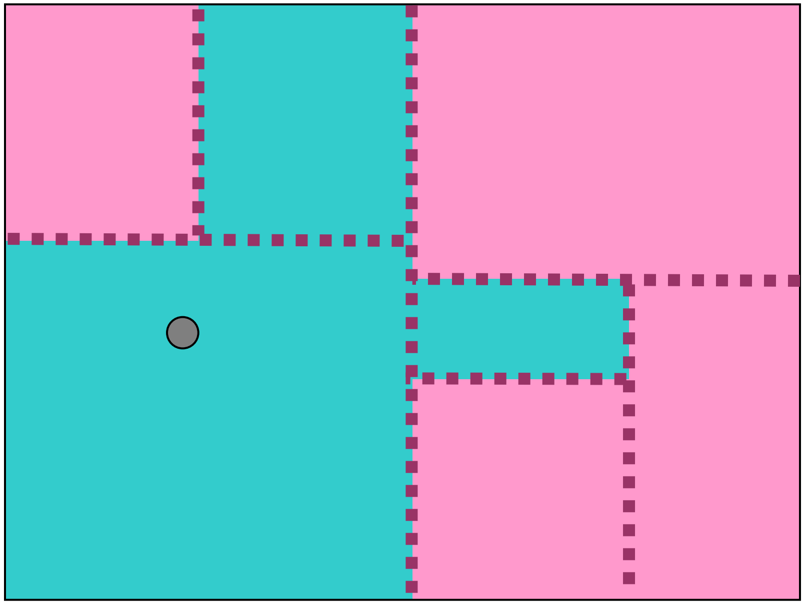
Decision Trees: information gain

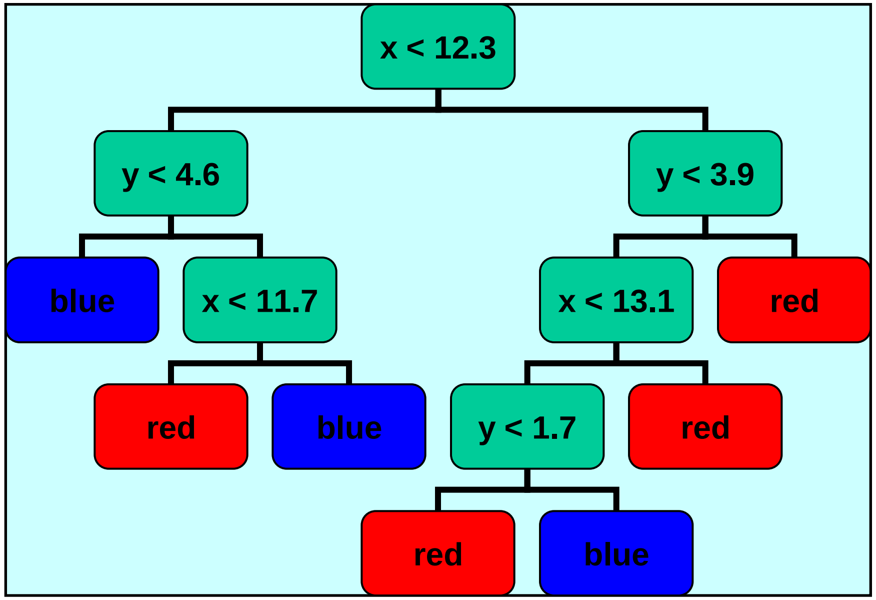
- Tree training, level 3: completely built tree





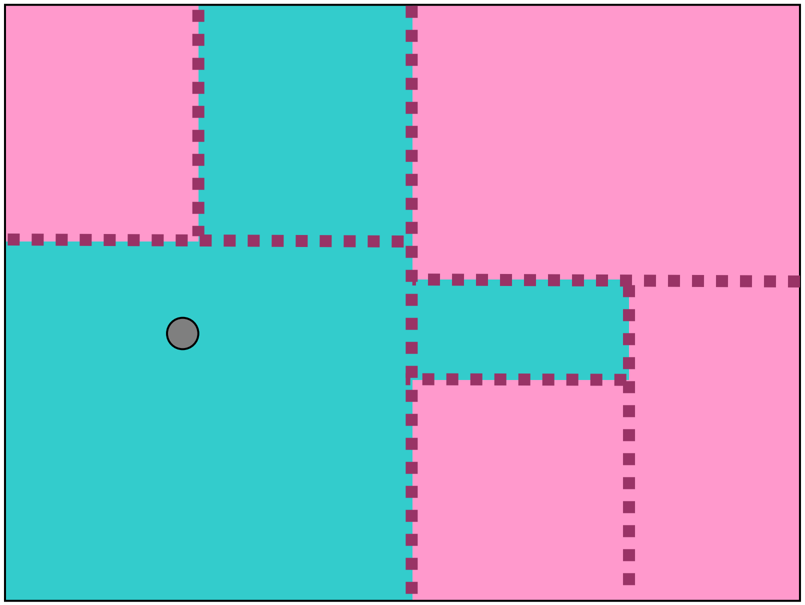
Decision Trees: Classification

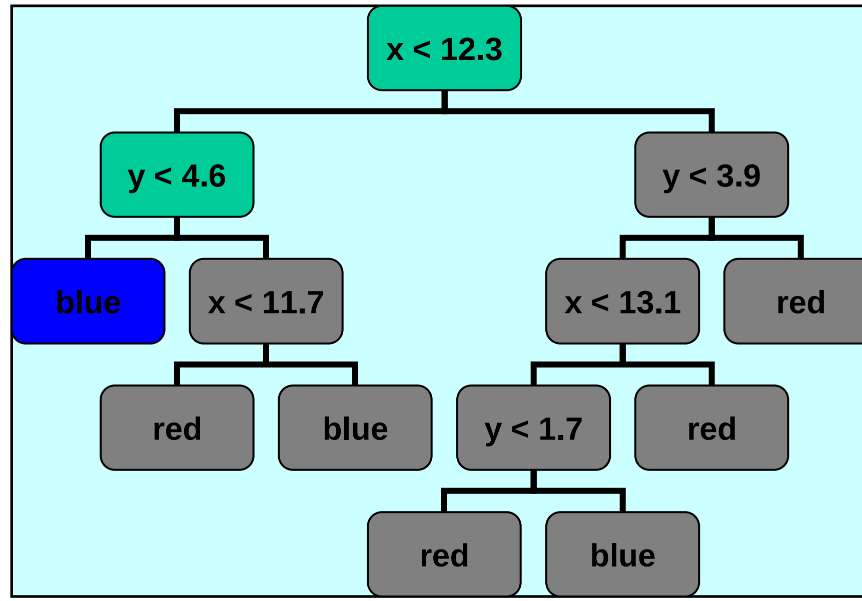




How to classify unknown items?

Decision Trees: Classification

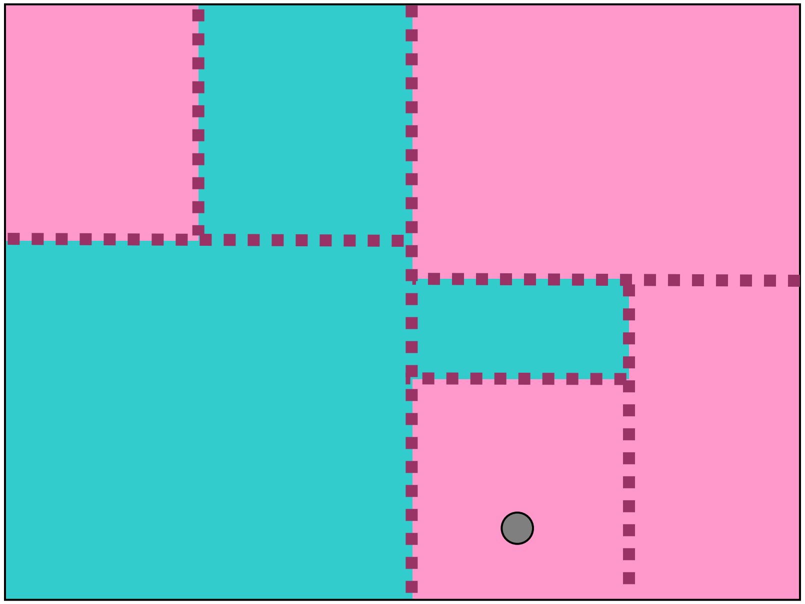


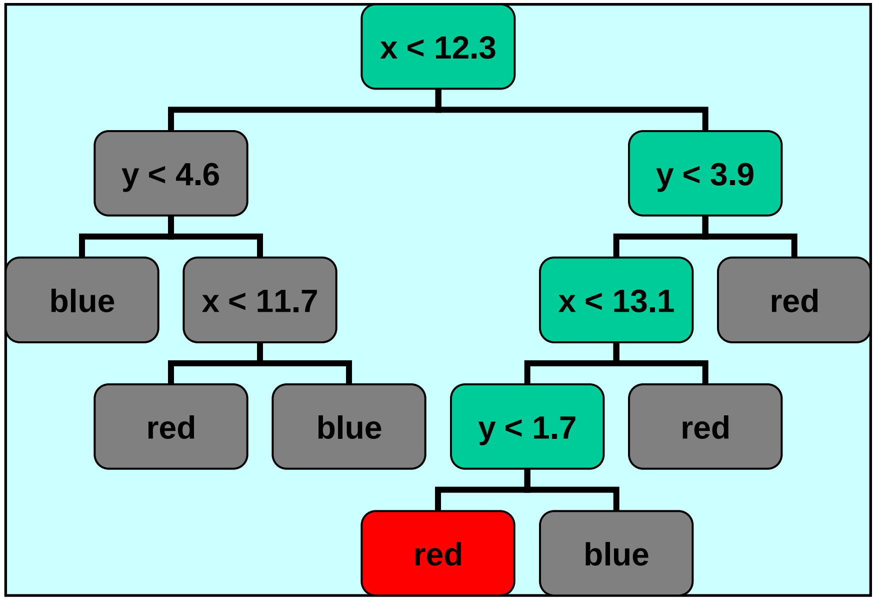


➔ Decend the tree until leaf-node

➔ Use majority of class in that leaf node

Decision Trees: Classification

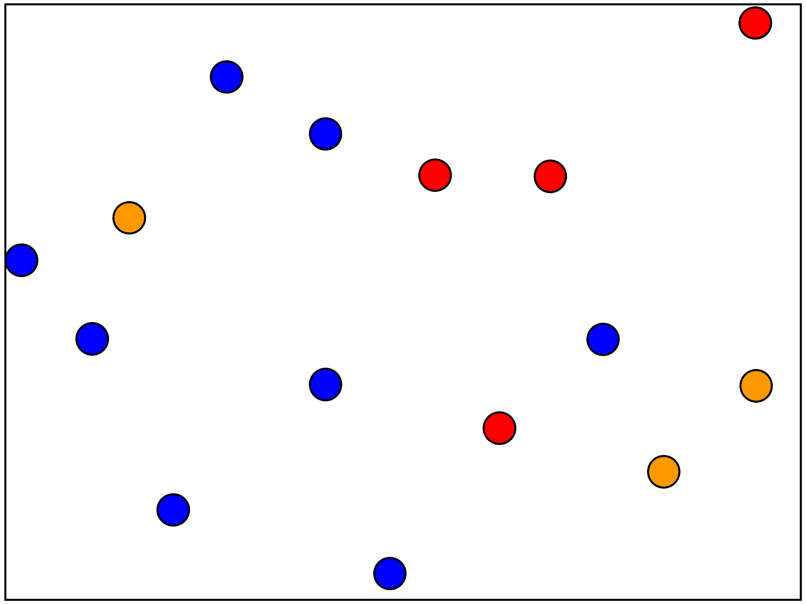




➔ Decend the tree until leaf-node

➔ Use majority of class in that leaf node

Decision Trees: More than 2 classes



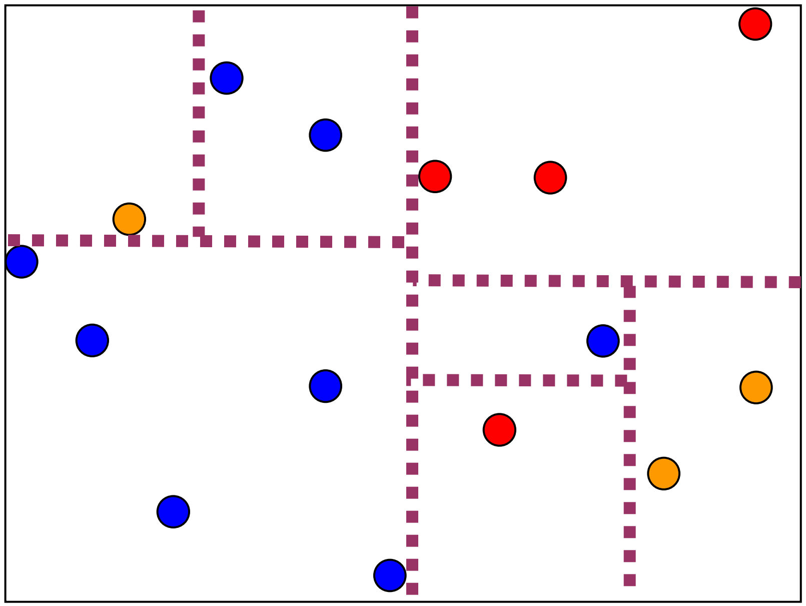
- *Conceptual changes?*

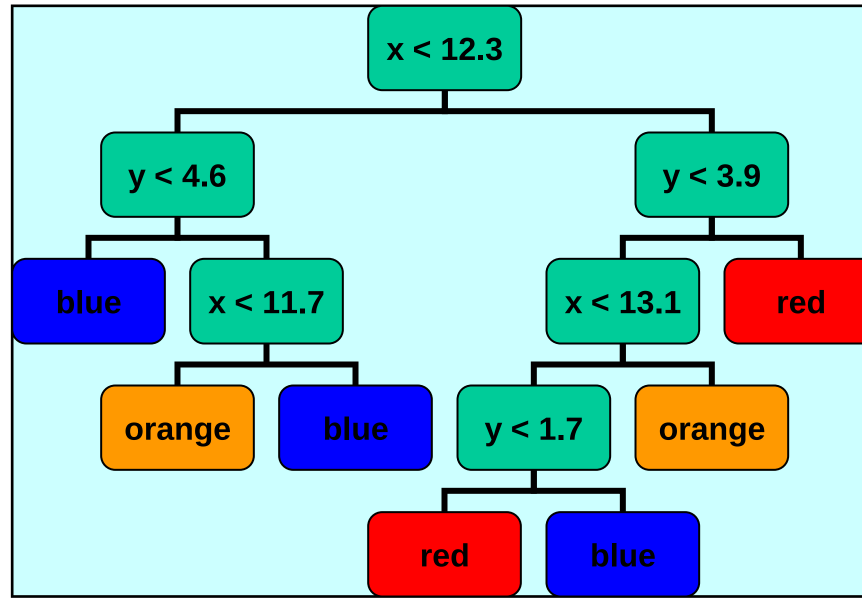
$$IG(X_1,\ldots,X_m) = H(X) - \sum_{j=1}^{m} p(x_j)H(X_j)$$

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

$$H(X) = -p(x_{red})\log_2 p(x_{red}) - p(x_{blue})\log_2 p(x_{blue}) - p(x_{yellow})\log_2 p(x_{yellow})$$

Decision Trees: More than 2 classes

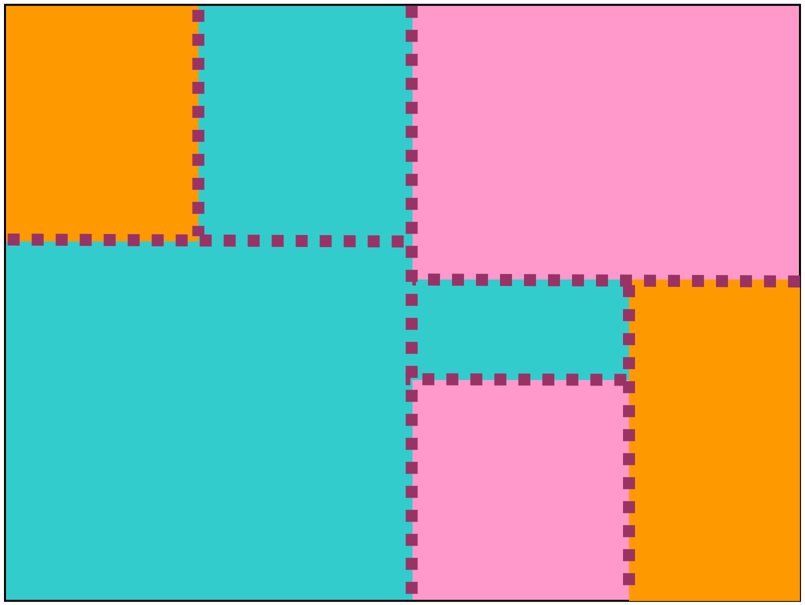


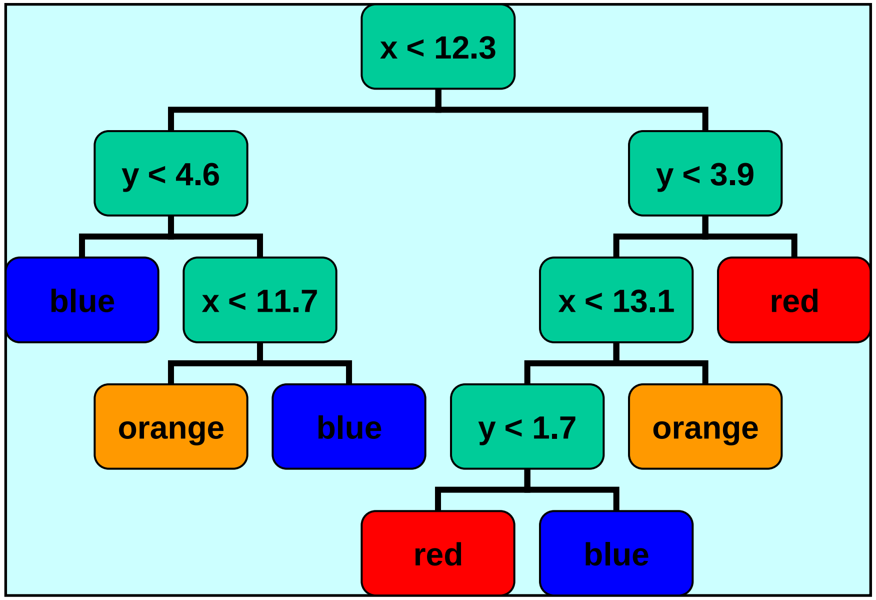


$$IG(X_1,\ldots,X_m) = H(X) - \sum_{j=1}^{m} p(x_j)\, H(X_j)$$

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Decision Trees: More than 2 classes

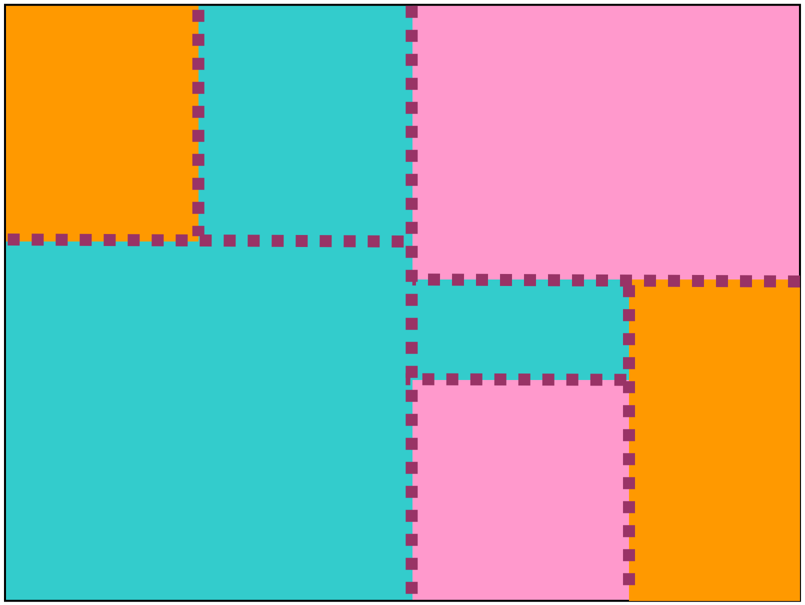


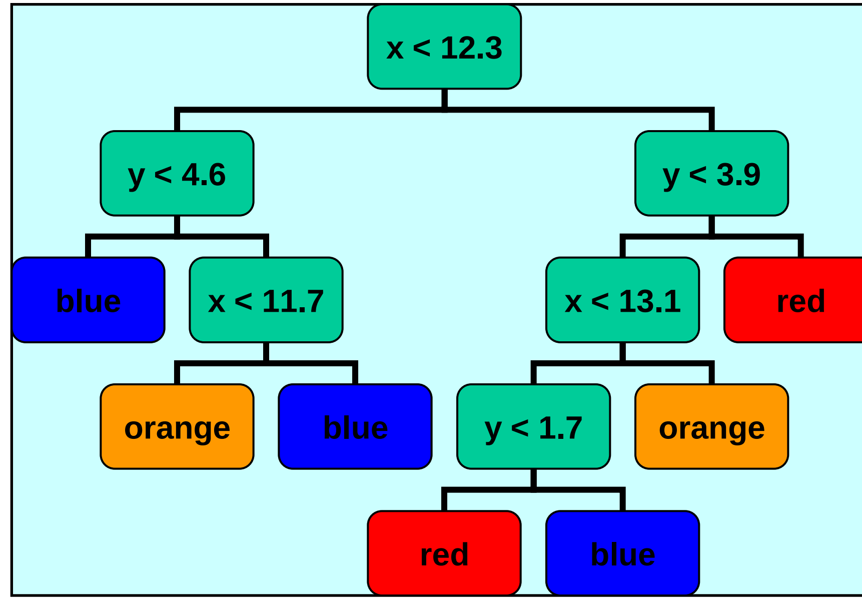


$$IG(X_1,\ldots,X_m) = H(X) - \sum_{j=1}^{m} p(x_j) H(X_j)$$

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Decision Trees: More than 2 classes



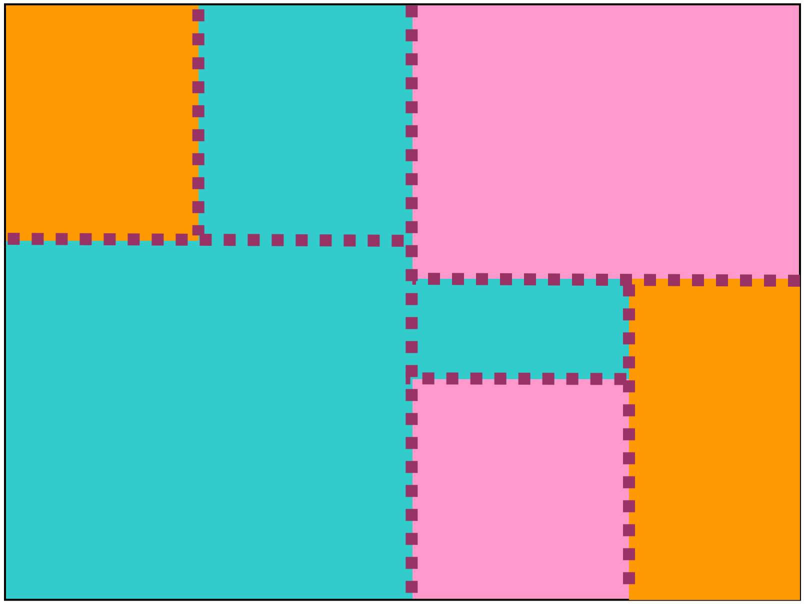


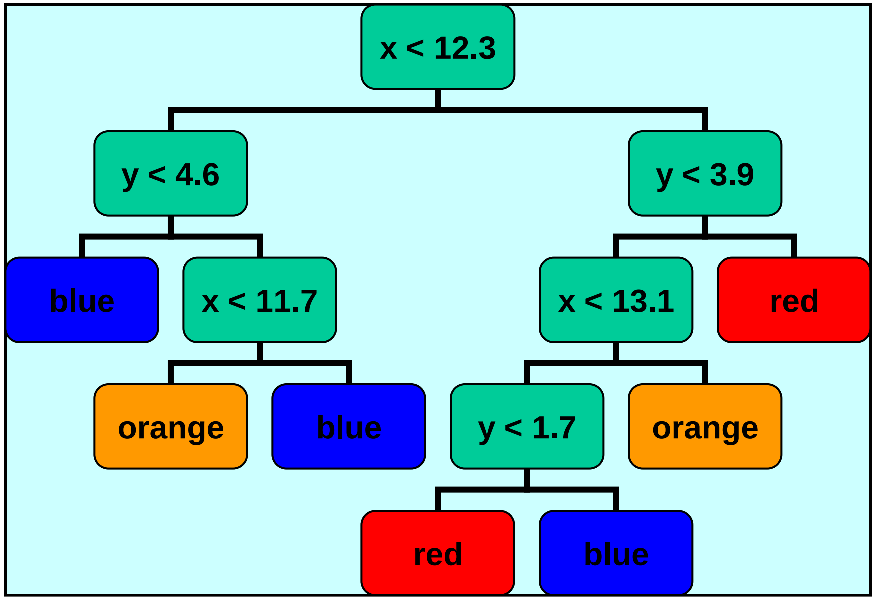
$$IG(X_1,\ldots,X_m) = H(X) - \sum_{j=1}^{m} p(x_j) H(X_j)$$

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

Maximum value of Entropy?

Decision Trees: More than 2 classes





$$IG(X_1, \ldots, X_m) = H(X) - \sum_{j=1}^{m} p(x_j) H(X_j)$$

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

$$H(X) = -p(x_{red}) \log_2 p(x_{red}) - p(x_{blue}) \log_2 p(x_{blue}) - p(x_{yellow}) \log_2 p(x_{yellow})$$

Maximum Entropy? $H(X) = \frac{1}{3} \times \log_2(\frac{1}{3}) - \frac{1}{3} \times \log_2(\frac{1}{3}) - \frac{1}{3} \times \log_2(\frac{1}{3}) = 1.5849$

Decision Trees: Algorithm

- Popular measures to compute best split
  - Error rate
  - Information gain
  - *Gini impurity (Gini index)*

Gini Index / ratio / impurity

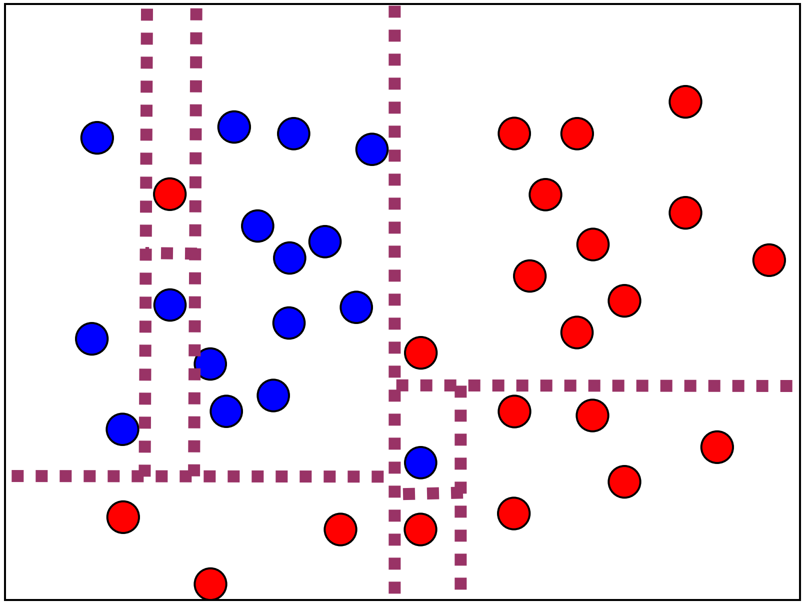
- Inequality among values of a distribution
  - Developed initial for income levels

$$\sum_{i=1}^{|C|} f_i(1 - f_i) = \sum_{i=1}^{|C|} (f_i - f_i^2) = \sum_{i=1}^{|C|} f_i - \sum_{i=1}^{|C|} f_i^2 = 1 - \sum_{i=1}^{|C|} f_i^2$$

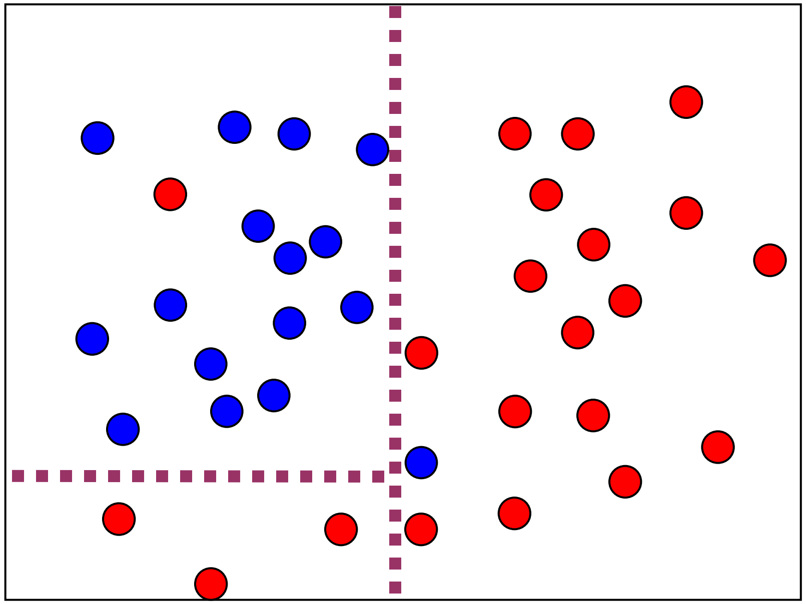
- Value range?
  - Between 0 (uniform distribution) and 1 (total inequality)
  - Values > 1 possible – if negative values are allowed
    - E.g. negative income

Decision Trees: Overfitting

- Fully grown trees are usually too complicated
  - *Why is that an issue?*
    - Generalisation
    - Understanding
  - Especially useful when there is noisy/”useless” data



Fully grown



Simplified

Decision Trees: Stopping Criteria

- *How to achieve simplified trees?*
  - Avoid fully growing trees! *How?*
    - ➔ Alternative stopping criteria
  - Stop splitting a node when
    - *Data in each node from only one class*
    - Absolute number of samples is low (< threshold)
    - Entropy is already relatively low (< threshold)
    - Information Gain is low (< threshold)
    - If the deviation in the data is statistically significant (chi-square pruning)
    - Depth of tree has reached a max value (> threshold)
  - Threshold values depend on data set
    - *“hyper parameters”*

Decision Trees: Pruning

- *How to achieve simplified trees?*
  - “Cut back” complicated trees!

- "Pruning" means removing nodes from a tree after training has finished
  - Stopping criteria are sometimes referred to as "pre-pruning“

- Nodes/branches with no/little power to classify are removed
  - Reduces complexity of tree

Decision Trees: Pruning

- Simple bottom-up approach: reduced error pruning

  1. Starting from leaves, remove a node from the tree
     1. Replace it with the majority class
  2. Evaluate the performance without the pruned node
  3. Repeat steps 1 & 2
     1. Until no improvement is obtained from pruning
     2. As long as performance is still acceptable

  – Take the best performing tree as the final decision tree

- Other approaches, e.g. cost complexity pruning (bottom up), Pessimistic Error Pruning (top-down), …

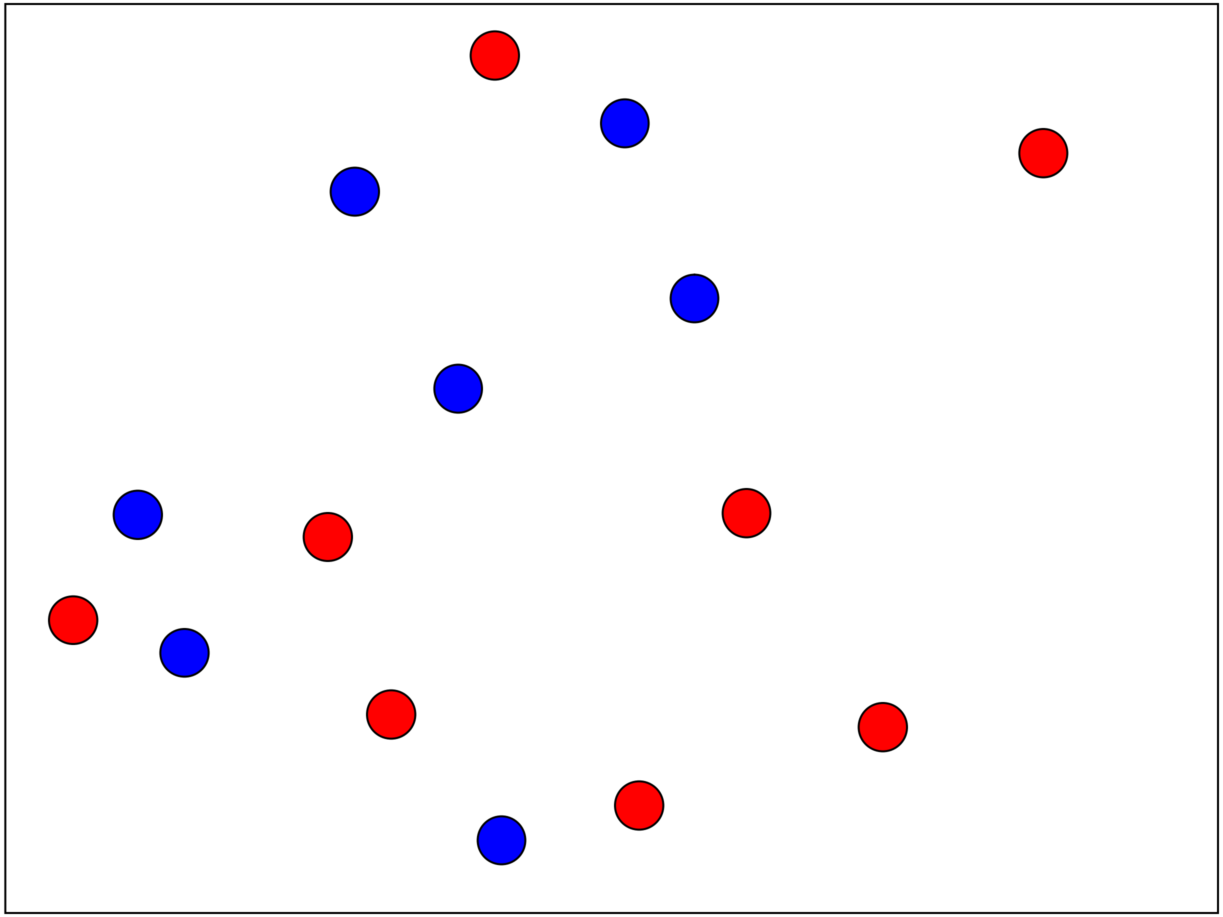
Decision Trees: Overfitting

- *What's the difference between pruning & pre-pruning?*

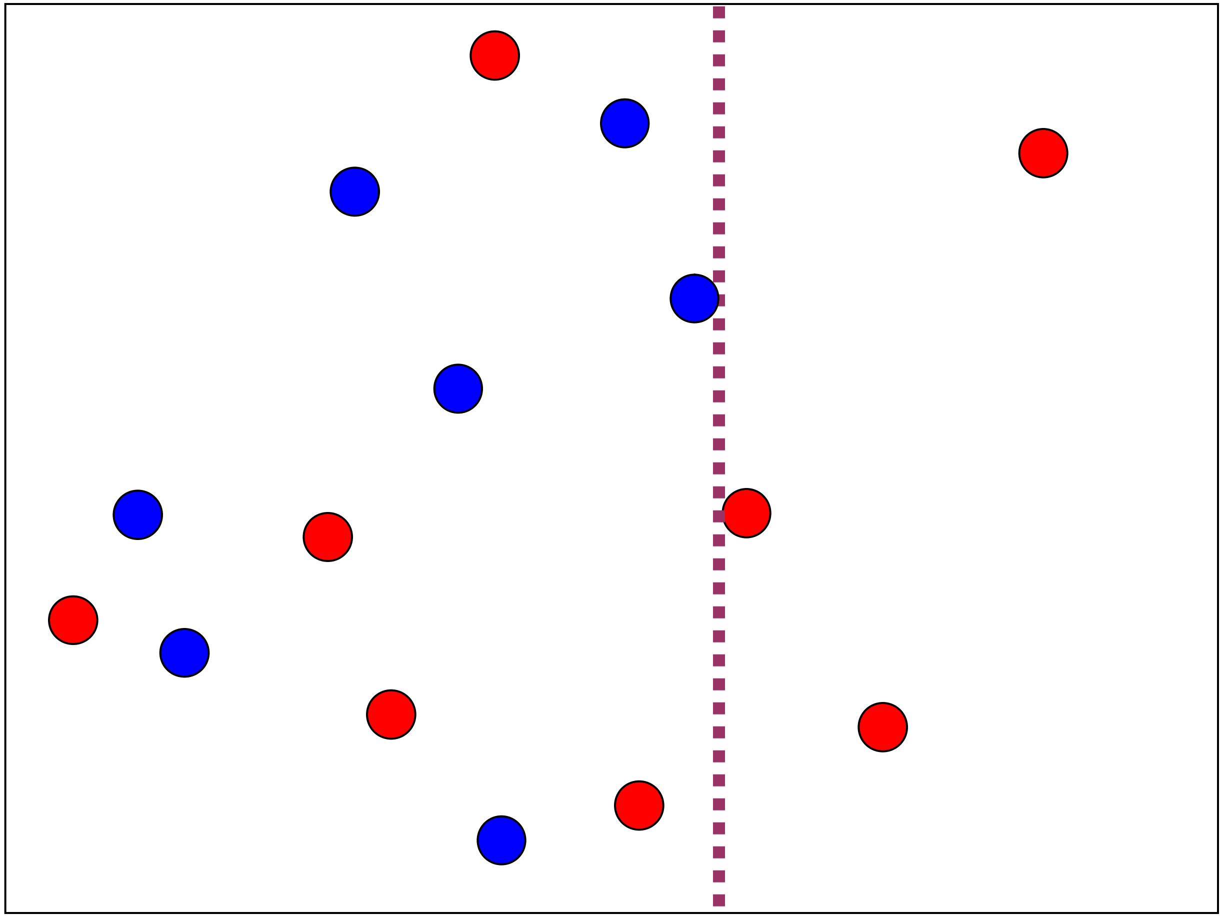
  – Effectiveness of final tree
    - Might be better for pruned trees
    - Or simply still more fitted to training data?

  – Effort for pruning algorithm (runtime)

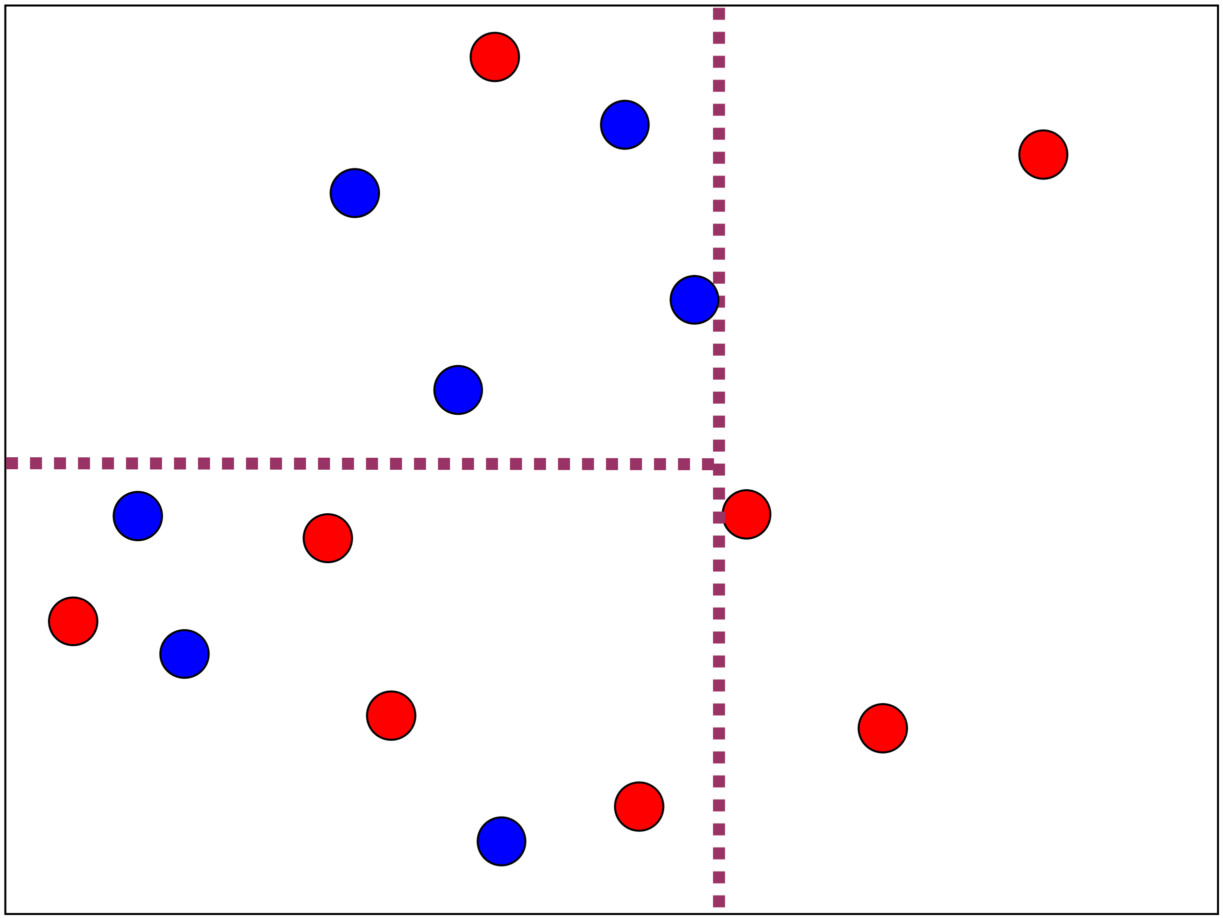
Decision Trees: Stability



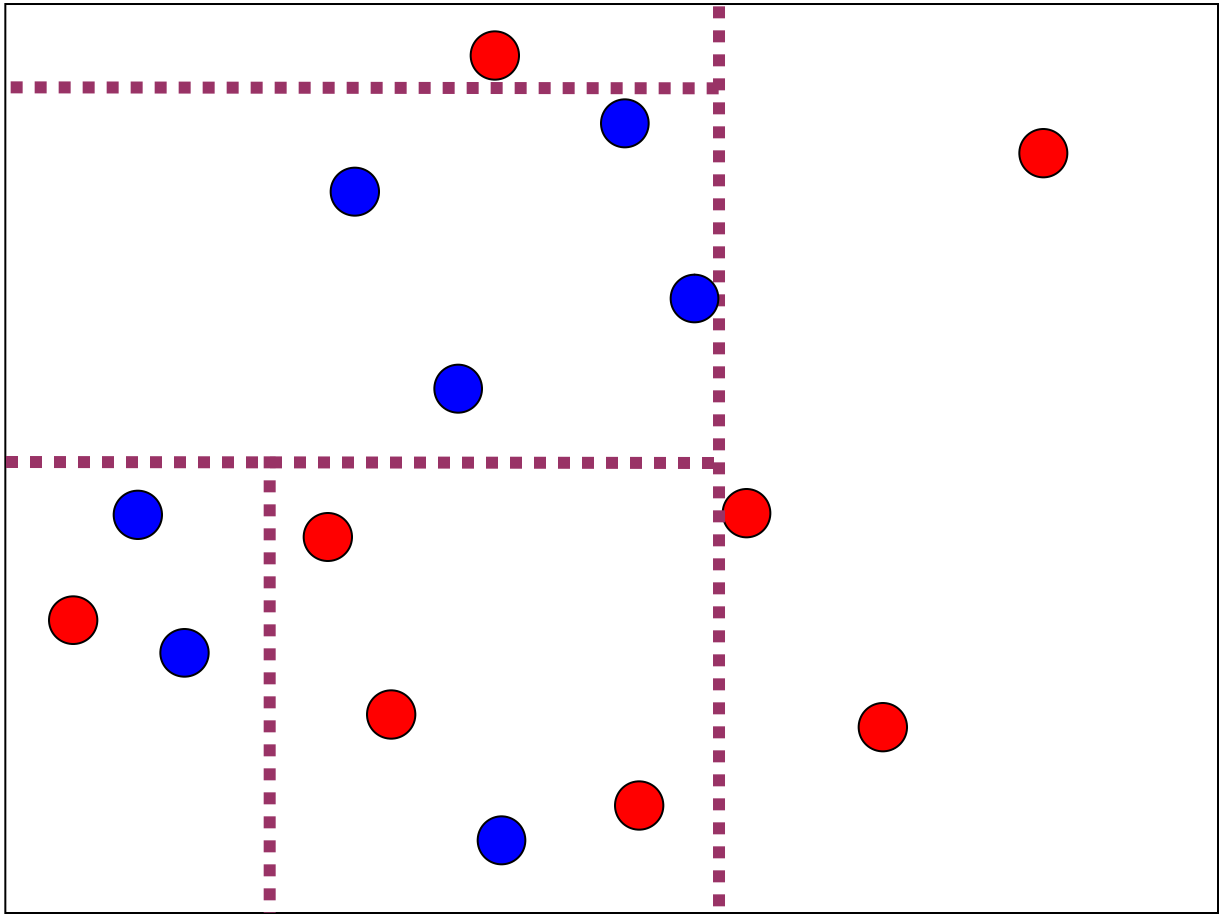
Decision Trees: Stability



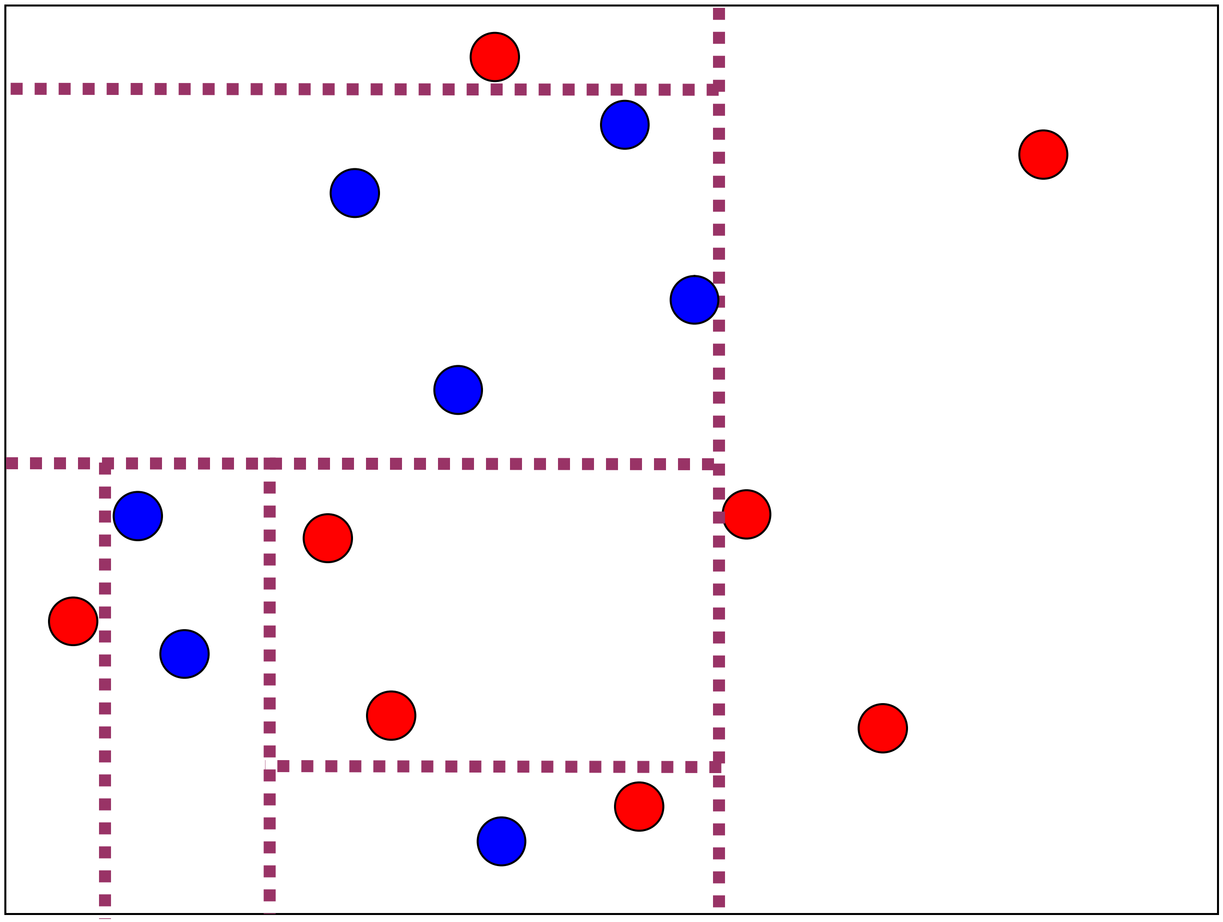
Decision Trees: Stability



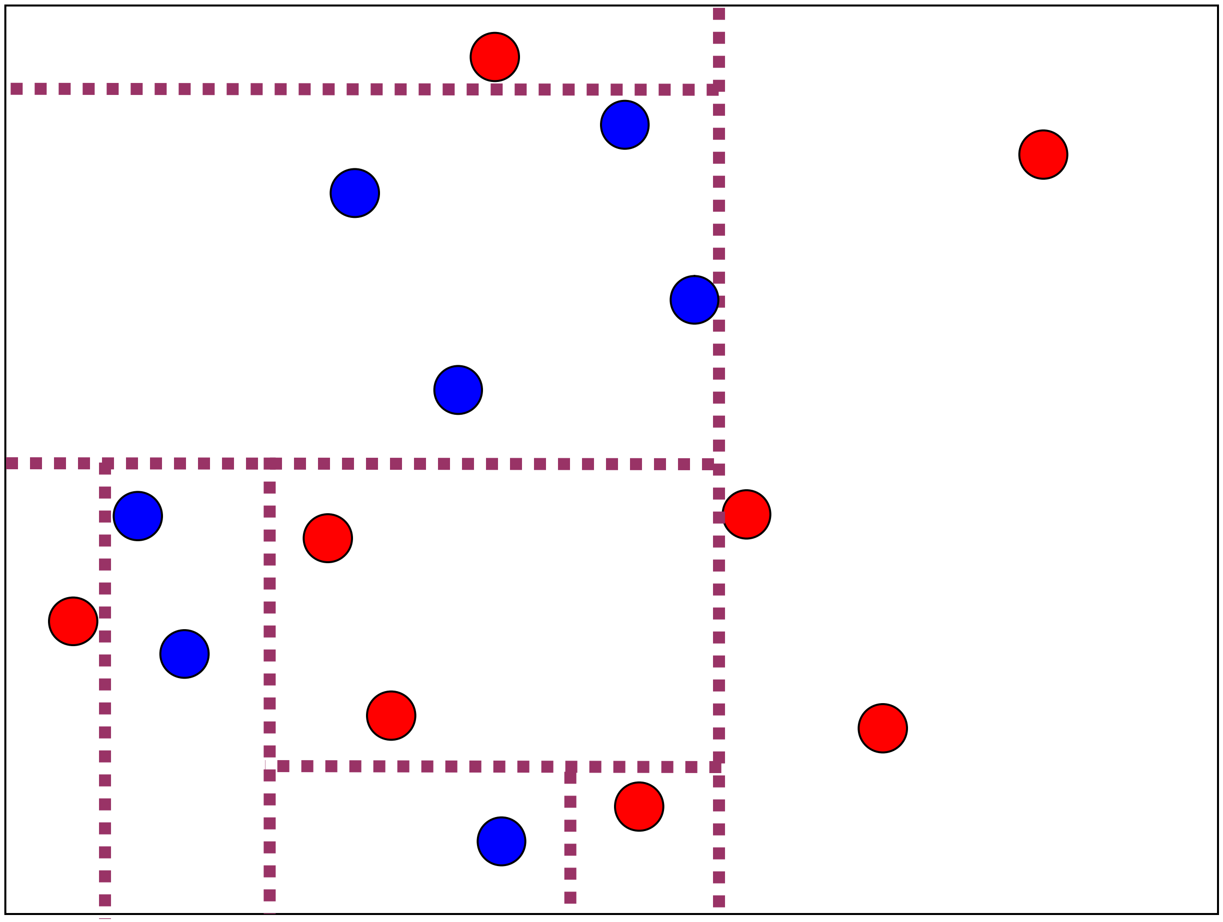
Decision Trees: Stability



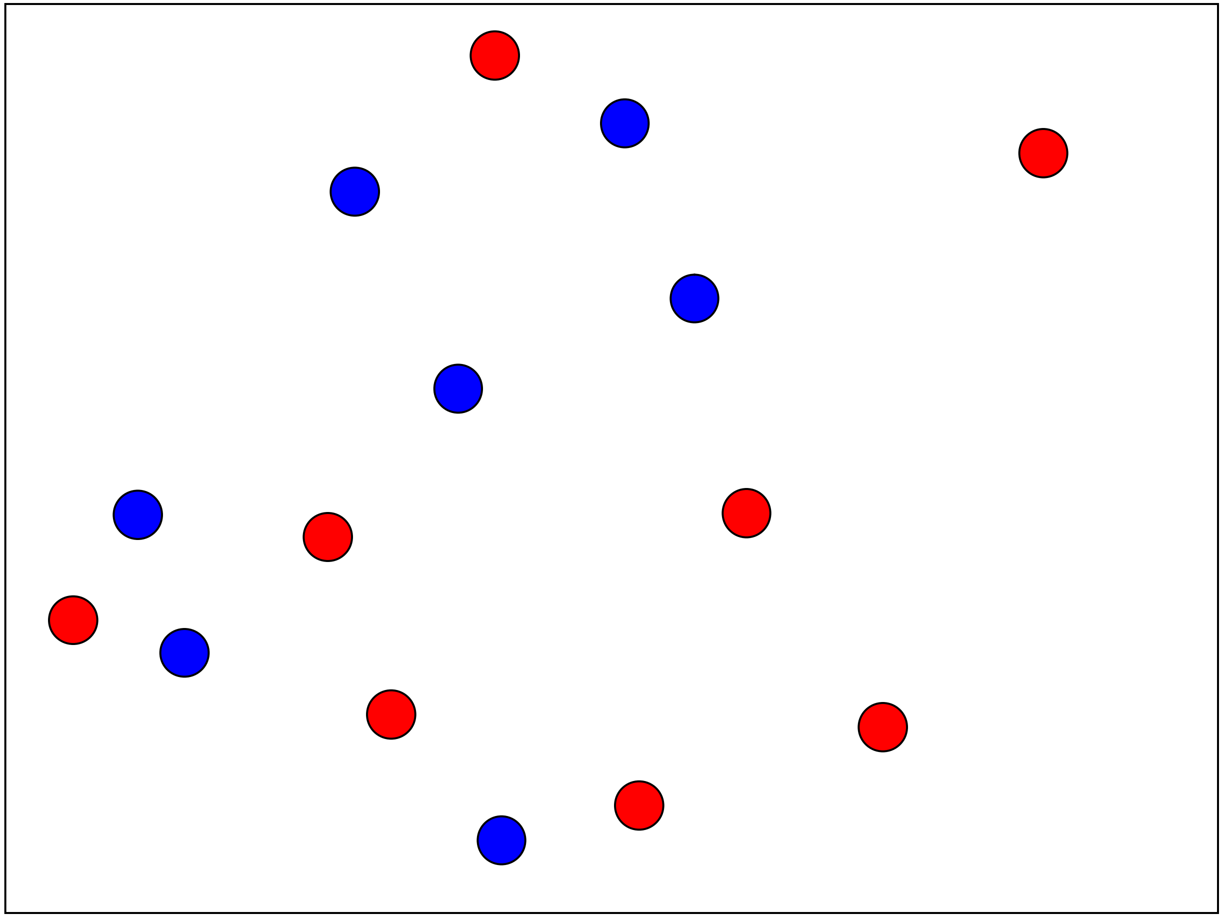
Decision Trees: Stability



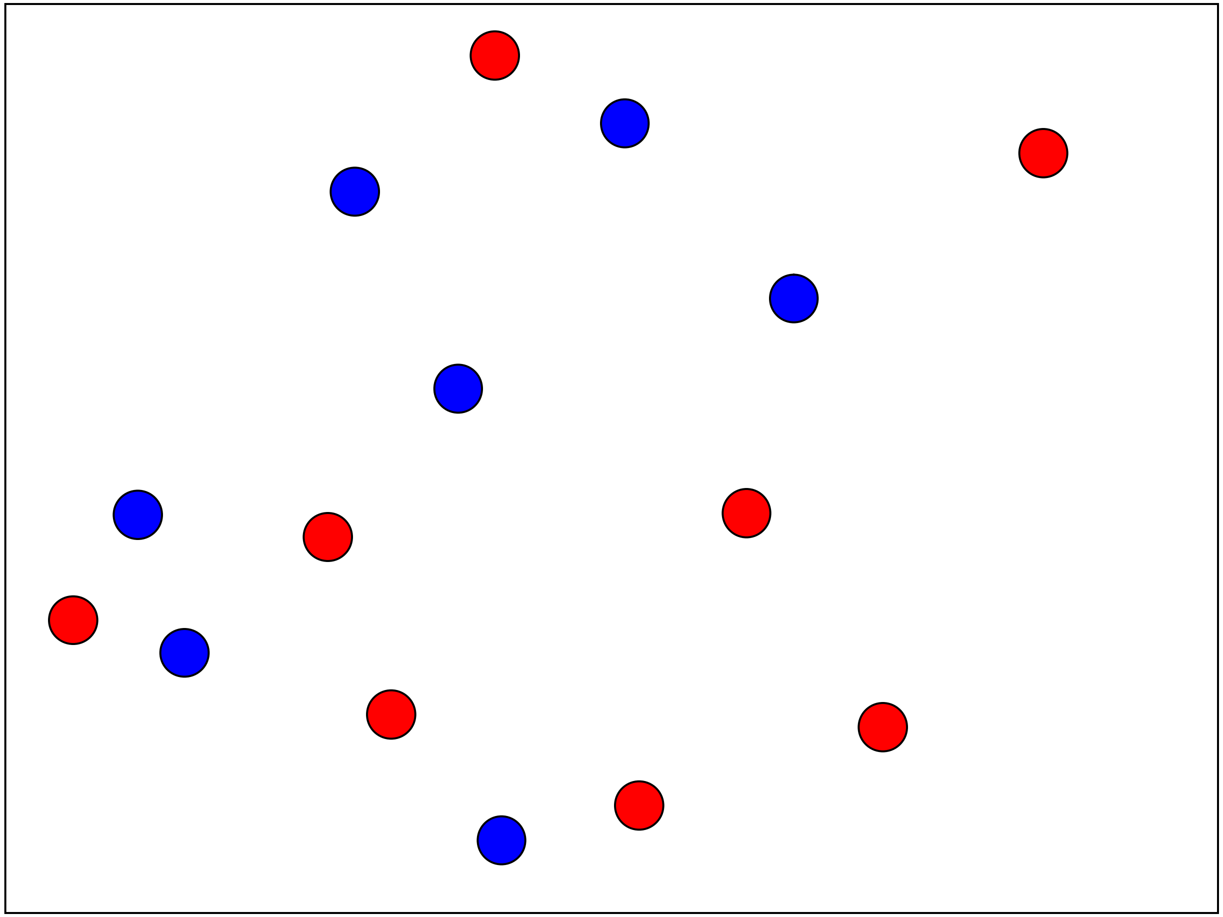
Decision Trees: Stability



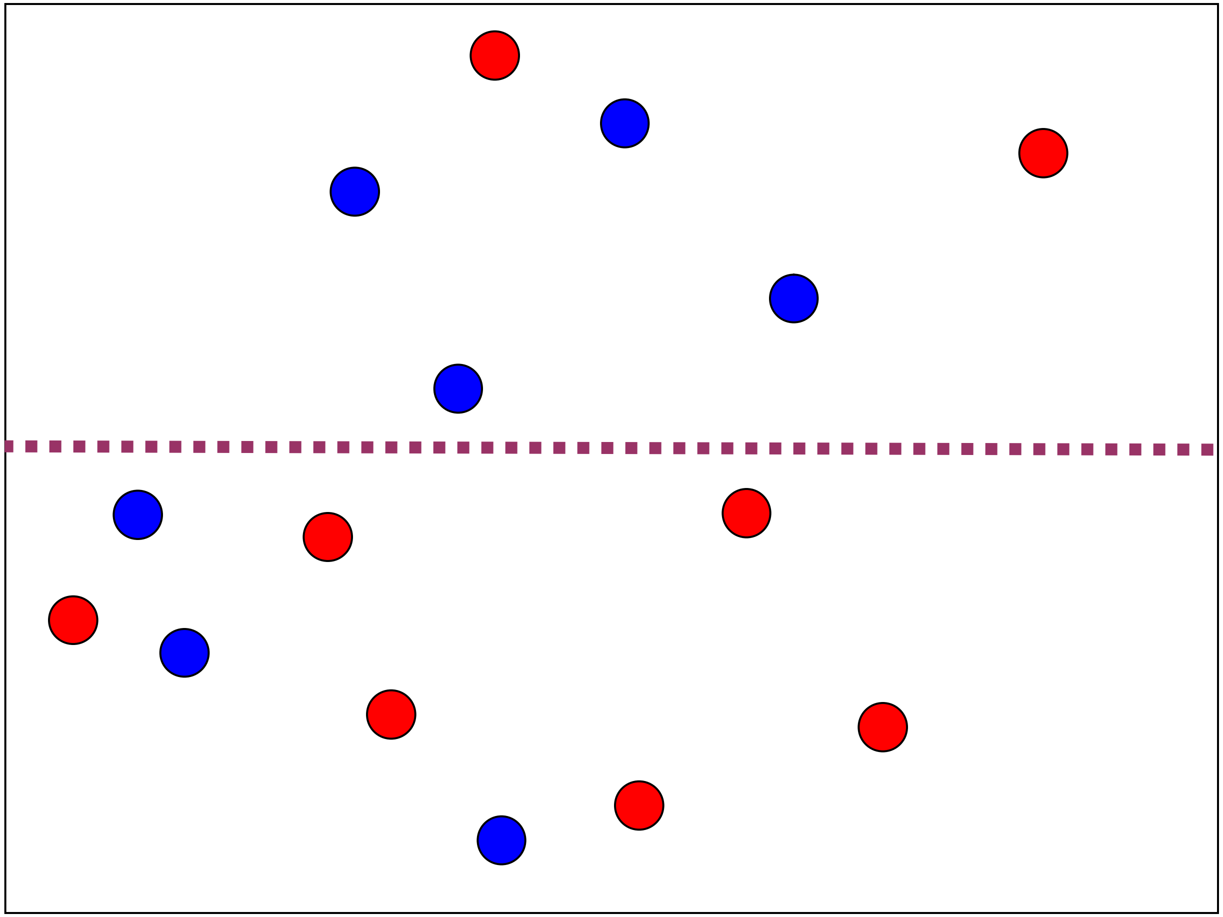
Decision Trees: Stability



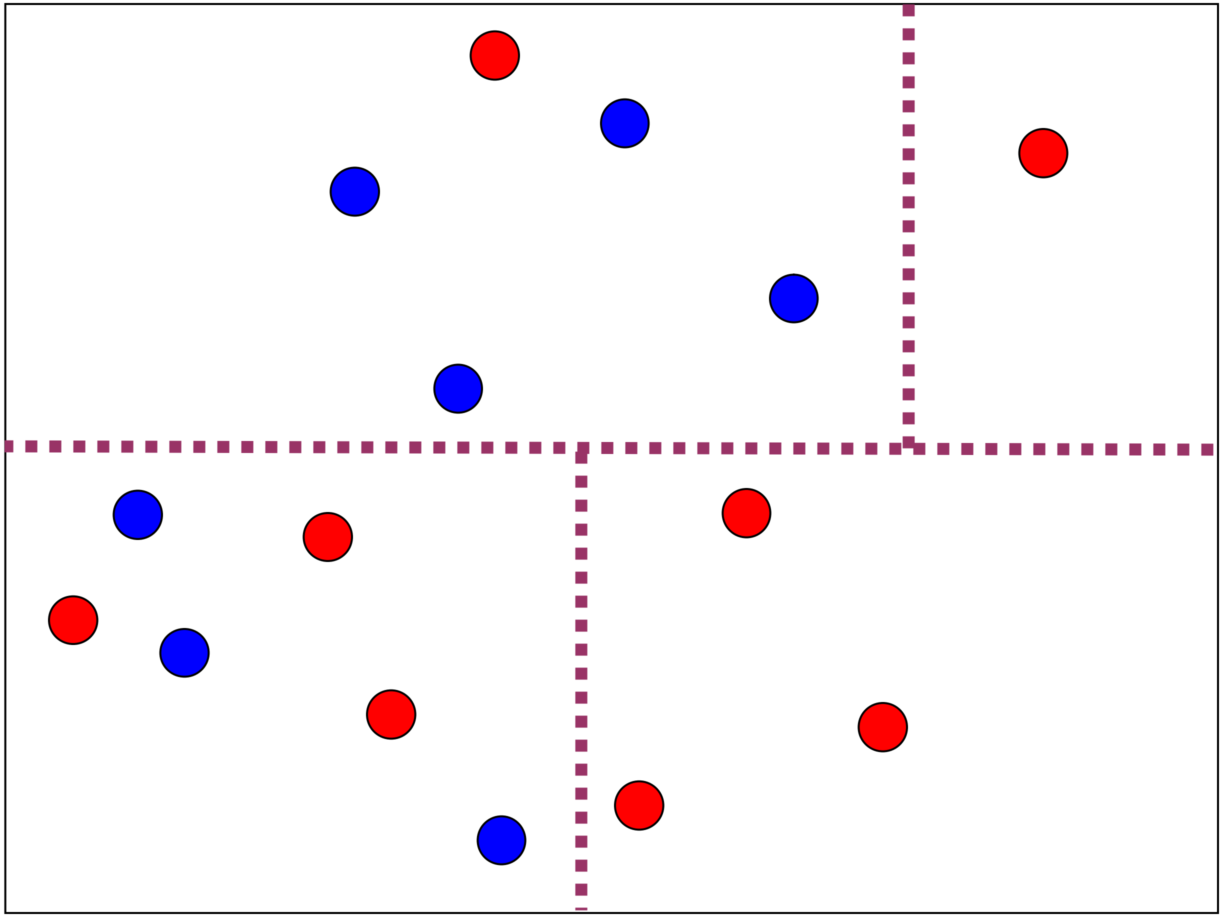
Decision Trees: Stability



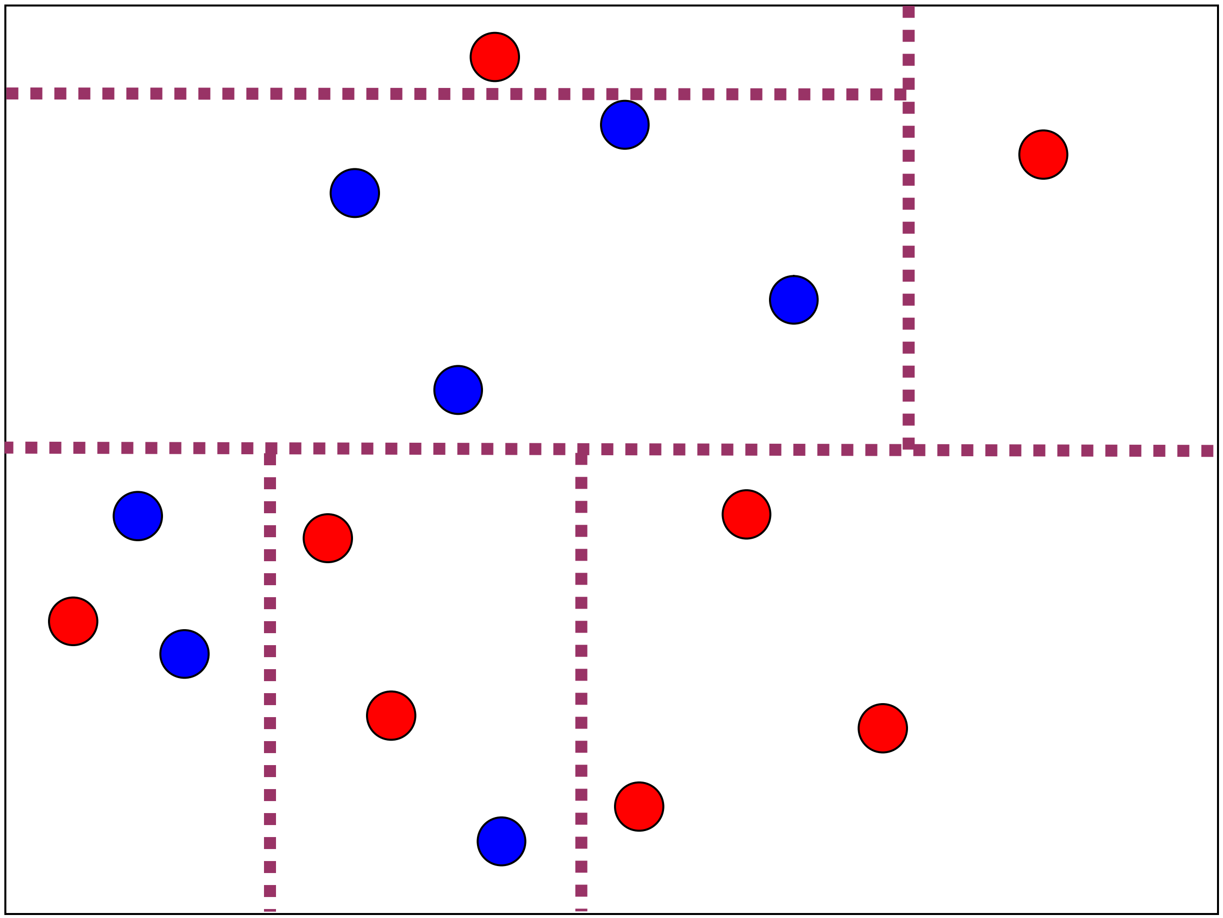
Decision Trees: Stability



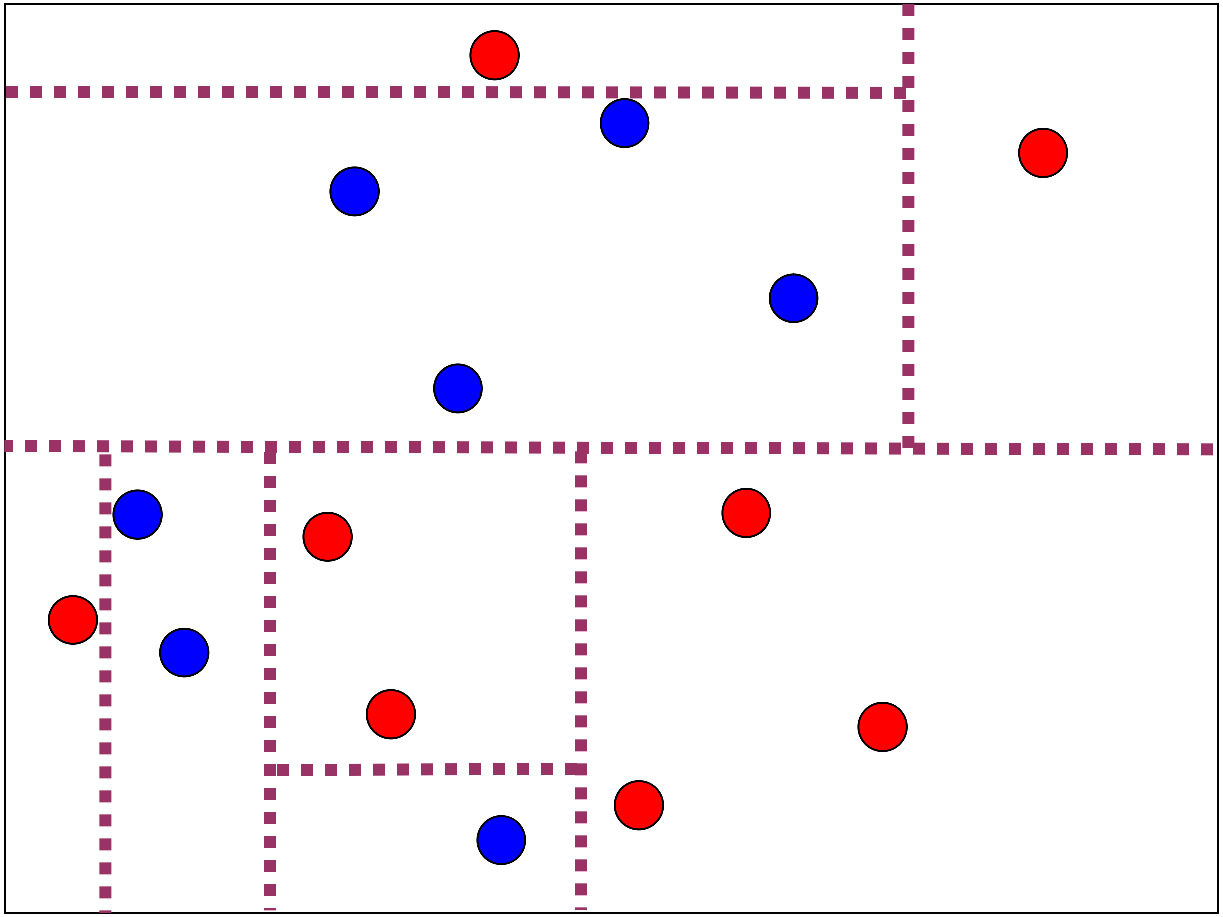
Decision Trees: Stability



Decision Trees: Stability

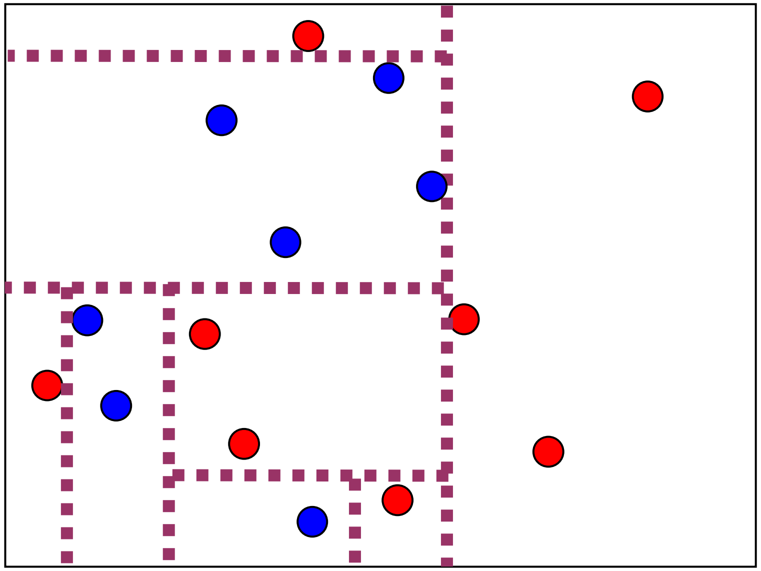


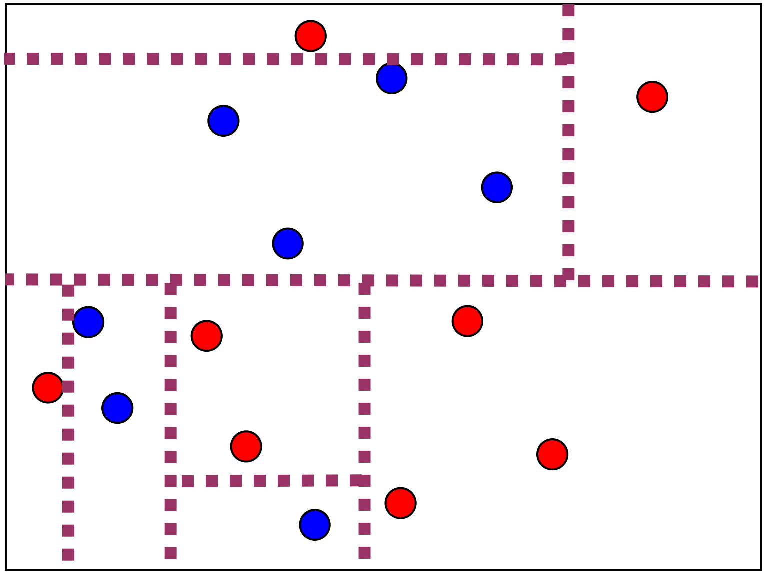
Decision Trees: Stability



Decision Trees: Stability

Small changes in data → potentially very different tree!





Decision Trees – properties

- Rather old model, well known
- Simple algorithm ➔ easy to understand
  - White box, rather than black-box
  - Used in many non-IT domains
  - Can be used to illustrate expert knowledge
- Various split criteria

- Problems with overfitting – (pre)pruning helps
- Problems with stability ➔ can be exploited!

Decision trees – other examples

- Previous example
  - 2D data, along x & y axis in Cartesian space
- Input data can be of any dimensionality
  - E.g. in 3D space of numerical data: planes diving the space along x, y or z axis

- Input data does not have to be numerical

➔ decision trees also work on categorical data

- There can be more than two classes

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
4	low	low	low	high	75-78	Asia	**good**
6	medium	medium	medium	medium	70-74	America	**bad**
4	medium	medium	medium	low	75-78	Europe	**bad**
8	high	high	high	low	70-74	America	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
4	low	medium	low	medium	70-74	Asia	**bad**
4	low	medium	low	low	70-74	Asia	**bad**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	medium	high	high	79-83	America	**good**
8	high	high	high	low	75-78	America	**bad**
4	low	low	low	low	79-83	America	**good**
6	medium	medium	medium	high	75-78	America	**bad**
4	medium	low	low	low	79-83	America	**good**
4	low	low	medium	high	79-83	America	**good**
8	high	high	high	low	70-71	America	**bad**
4	low	medium	low	medium	75-78	Europe	**good**
5	medium	medium	medium	medium	75-78	Europe	**bad**

18/40 Records subsample (similar in UCI Machine learning repository)

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
4	low	low	low	high	75-78	Asia	**good**
6	medium	medium	medium	medium	70-74	America	**bad**
4	medium	medium	medium	low	75-78	Europe	**bad**
8	high	high	high	low	70-74	America	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
4	low	medium	low	medium	70-74	Asia	**bad**
4	low	medium	low	low	70-74	Asia	**bad**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	medium	high	high	79-83	America	**good**
8	high	high	high	low	75-78	America	**bad**
4	low	low	low	low	79-83	America	**good**
6	medium	medium	medium	high	75-78	America	**bad**
4	medium	low	low	low	79-83	America	**good**
4	low	low	medium	high	79-83	America	**good**
8	high	high	high	low	70-71	America	**bad**
4	low	medium	low	medium	75-78	Europe	**good**
5	medium	medium	medium	medium	75-78	Europe	**bad**

Entropy of data set $H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$

12 samples class bad (2/3), 6 samples good (1/3)

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
4	low	low	low	high	75-78	Asia	**good**
6	medium	medium	medium	medium	70-74	America	**bad**
4	medium	medium	medium	low	75-78	Europe	**bad**
8	high	high	high	low	70-74	America	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
4	low	medium	low	medium	70-74	Asia	**bad**
4	low	medium	low	low	70-74	Asia	**bad**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	medium	high	high	79-83	America	**good**
8	high	high	high	low	75-78	America	**bad**
4	low	low	low	low	79-83	America	**good**
6	medium	medium	medium	high	75-78	America	**bad**
4	medium	low	low	low	79-83	America	**good**
4	low	low	medium	high	79-83	America	**good**
8	high	high	high	low	70-71	America	**bad**
4	low	medium	low	medium	75-78	Europe	**good**
5	medium	medium	medium	medium	75-78	Europe	**bad**

Entropy of data set:

$- 1/3 \times \log_2 1/3 - 2/3 \times \log_2 2/3$ $= - 1/3 \times \log(1/3)/\log(2) - 2/3 \times \log(2/3)/\log(2)$

$= - 1/3 \times -1{,}59946 - 2/3 \times -0{,}58496$

$= 0{,}918295834$

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
4	low	low	low	high	75-78	Asia	**good**
4	low	low	low	low	79-83	America	**good**
4	low	medium	low	medium	75-78	Europe	**good**
4	medium	low	low	low	79-83	America	**good**
4	low	low	medium	high	79-83	America	**good**
4	medium	medium	medium	low	75-78	Europe	**bad**
4	low	medium	low	medium	70-74	Asia	**bad**
4	low	medium	low	low	70-74	Asia	**bad**
5	medium	medium	medium	medium	75-78	Europe	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
6	medium	medium	medium	high	75-78	America	**bad**
8	high	medium	high	high	79-83	America	**good**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	high	high	low	70-71	America	**bad**

Split on first attribute – cylinders
- 4 distinct values – split in 4 sets
- Compute IG – compute entropy for each subset

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
4	low	low	low	high	75-78	Asia	**good**
4	low	low	low	low	79-83	America	**good**
4	low	medium	low	medium	75-78	Europe	**good**
4	medium	low	low	low	79-83	America	**good**
4	low	low	medium	high	79-83	America	**good**
4	medium	medium	medium	low	75-78	Europe	**bad**
4	low	medium	low	medium	70-74	Asia	**bad**
4	low	medium	low	low	70-74	Asia	**bad**

5 samples class good (5/8), 3 samples class bad (3/8)

$H(X_{cylinders=4}) = -5/8 \times \log_2(5/8) - 3/8 \times \log_2(3/8)$

$= (-5\ 8\ \log(5\ 8)\ \log(2)) + (-3\ 8\ \log(3\ 8)\ \log(2))$

$= 0{,}954434003$

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
5	medium	medium	medium	medium	75-78	Europe	**bad**

1 sample class bad

$H(X_{cylinders=5}) = 0$

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
6	medium	medium	medium	medium	70-74	America	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
6	medium	medium	medium	high	75-78	America	**bad**

3 sample class bad

$H(X_{cylinders=6}) = 0$

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
8	high	medium	high	high	79-83	America	**good**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	high	high	low	70-71	America	**bad**

1 sample class good (1/6), 5 samples class bad (5/6),

$$H(X_{cylinders=8}) = -1/6 \times \log_2(1/6) - 5/6 \times \log_2(5/6)$$

$$= (-1\ \ 6\ \ \log(1\ \ 6)\ \ \log(2))+(-5\ \ 6\ \ \log(5\ \ 6)\ \ \log(2))$$

$$= 0{,}650022422$$

Miles Per Gallon Data Set

cylinders	displacement	horse power	weight	accelleration	Model year	maker	MpG
4	low	low	low	high	75-78	Asia	**good**
4	low	low	low	low	79-83	America	**good**
4	low	medium	low	medium	75-78	Europe	**good**
4	low	low	medium	high	79-83	America	**good**
4	low	medium	low	medium	70-74	Asia	**bad**
4	low	medium	low	low	70-74	Asia	**bad**
4	medium	low	low	low	79-83	America	**good**
5	medium	medium	medium	medium	75-78	Europe	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
6	medium	medium	medium	medium	70-74	America	**bad**
6	medium	medium	medium	high	75-78	America	**bad**
4	medium	medium	medium	low	75-78	Europe	**bad**
8	high	medium	high	high	79-83	America	**good**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	high	high	low	75-78	America	**bad**
8	high	high	high	low	70-74	America	**bad**
8	high	high	high	low	70-71	America	**bad**

Split on second attribute – displacement

- 3 distinct values – split in 3 sets
- Compute IG – compute entropy for each subset

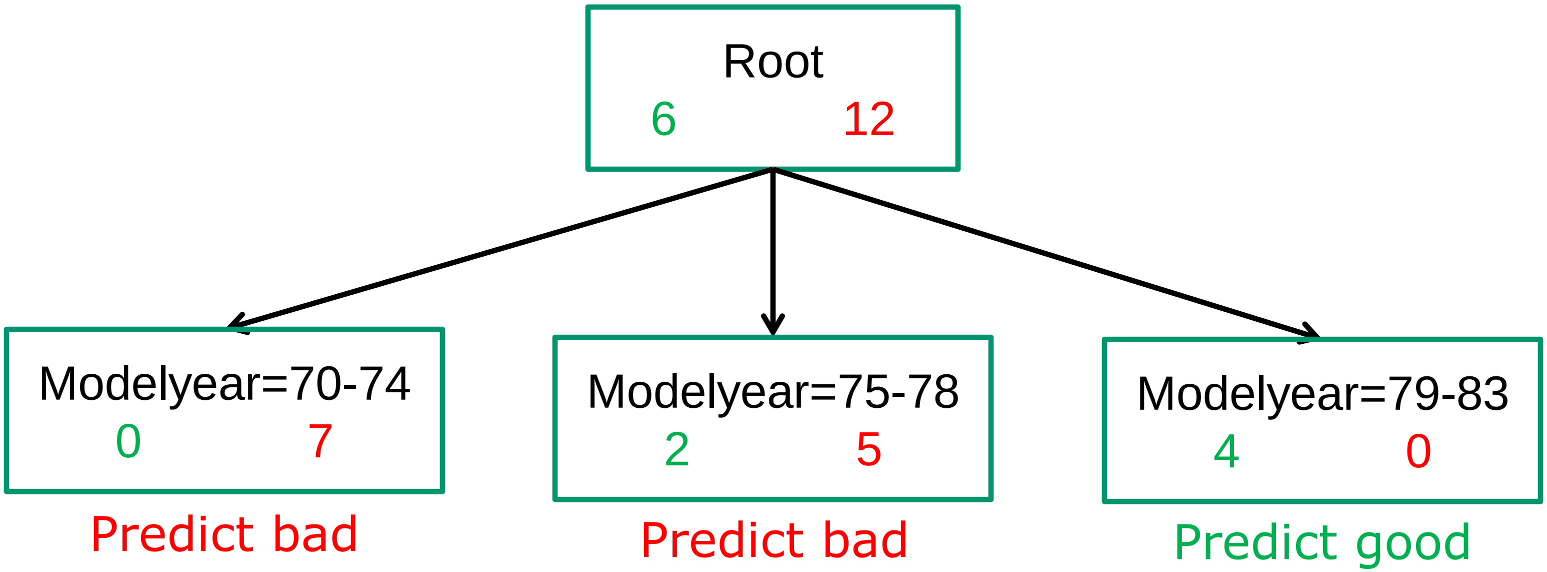
Training the tree

Build a decision tree

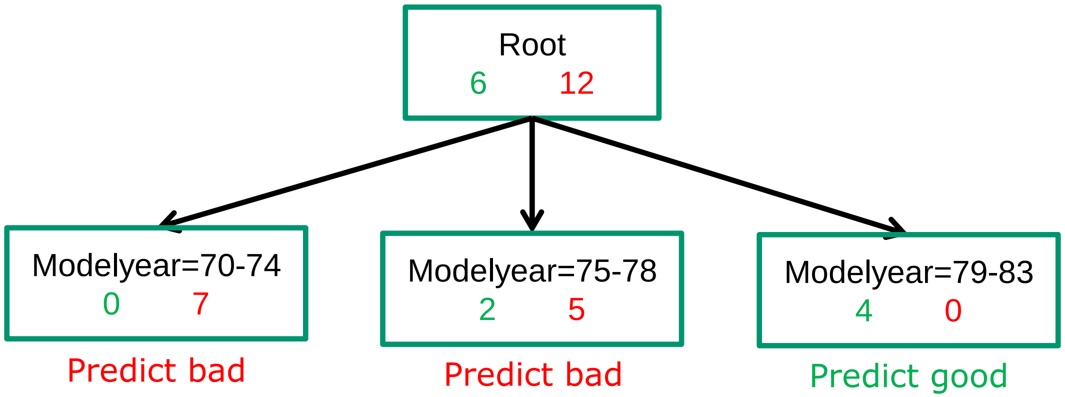
1. Identify splits
2. Compute Igs
3. Select attribute with highest IG
   ➔ Cylinders

Attribute	Attr.Value	good	bad	Distribution	Entropy	Info Gain
Full Dataset		6	12		0,9183	
Attribute	**Attr.Value**	**good**	**bad**	**Distribution**	**Entropy**	**Info Gain**
cylinders	*4*	5	3		0,9544	
	5	0	3		0,	
	6	0	3		0,	
	8	1	3		0,8113	
				Split	***0,6045***	***0,3138***
displacemen	*low*	4	2		0,9183	
	medium	1	5		0,65	
	high	1	5		0,65	
				Split	***0,7394***	***0,1788***
horse power	*low*	4	0		0,	
	medium	2	7		0,7642	
	high	0	5		0,	
				Split	***0,3821***	***0,5362***
weight	*low*	4	2		0,9183	
	medium	1	5		0,65	
	high	1	5		0,65	
				Split	***0,7394***	***0,1788***
accelleratior	*low*	2	7		0,7642	
	medium	1	4		0,7219	
	high	3	1		0,8113	
				Split	***0,7629***	***0,1554***
Model year	*70-74*	0	7		0,	
	75-78	2	5		0,8631	
	79-83	4	0		0,	
				Split	***0,3357***	***0,5826***
maker	*Asia*	1	2		0,9183	
	America	4	8		0,9183	
	Europe	1	2		0,9183	
				Split	0,9183	0,

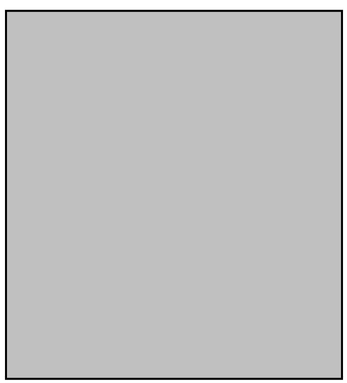
First level of Decision Tree



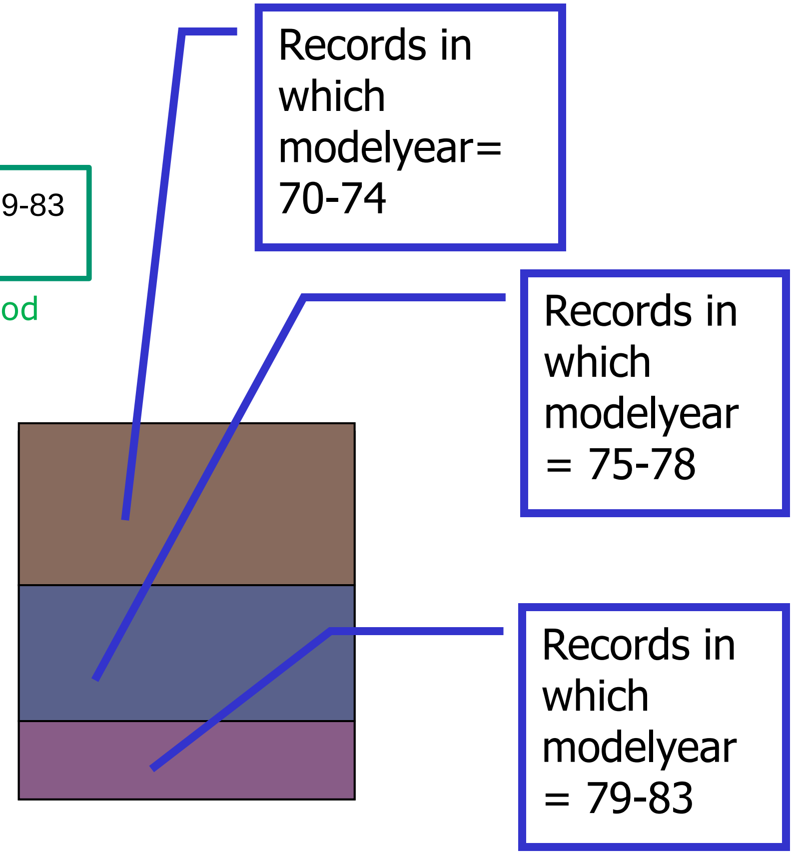
Recursion Step



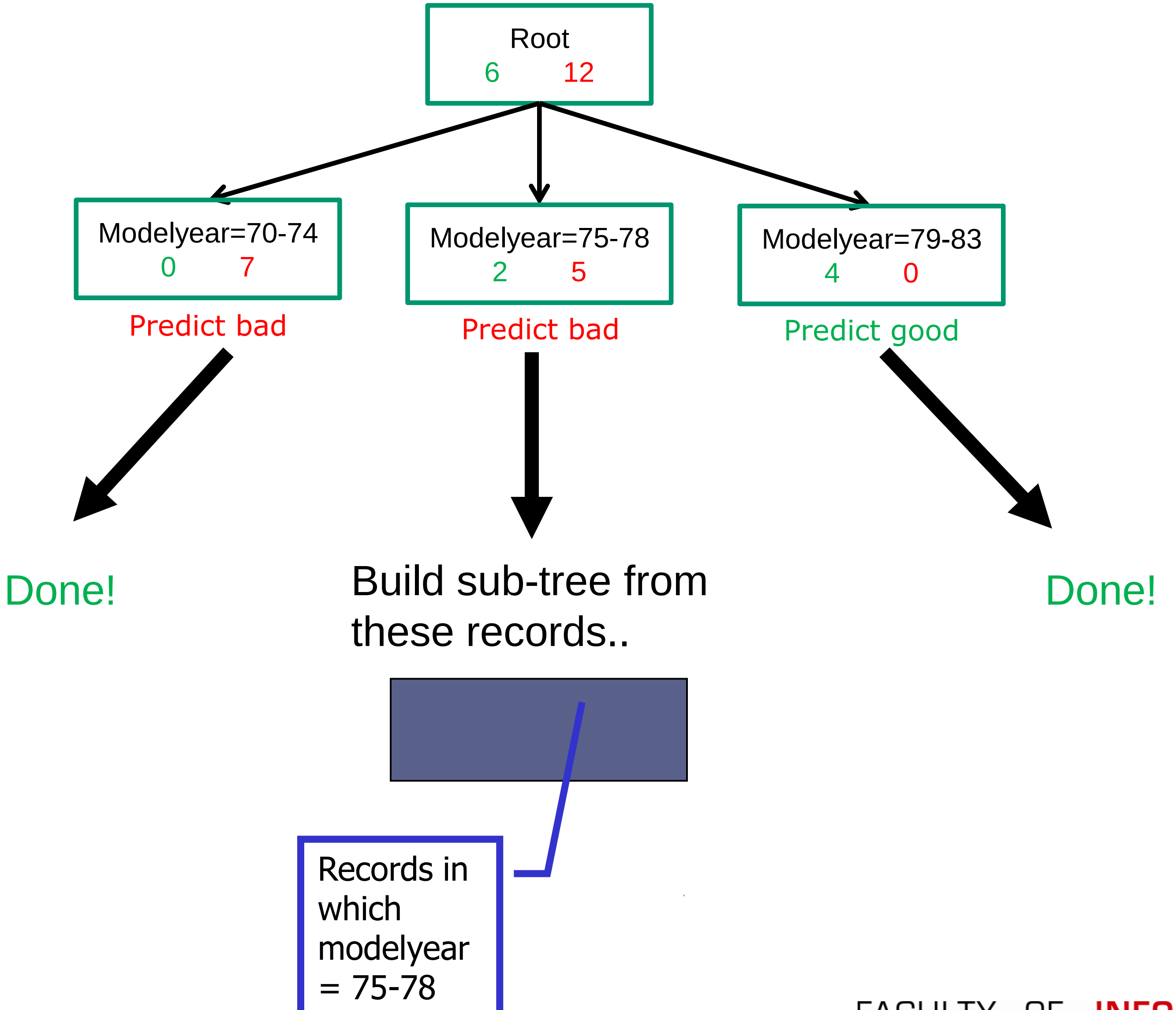
Take the Original Dataset..



And partition it according to the value of the attribute we split on



Recursion Step



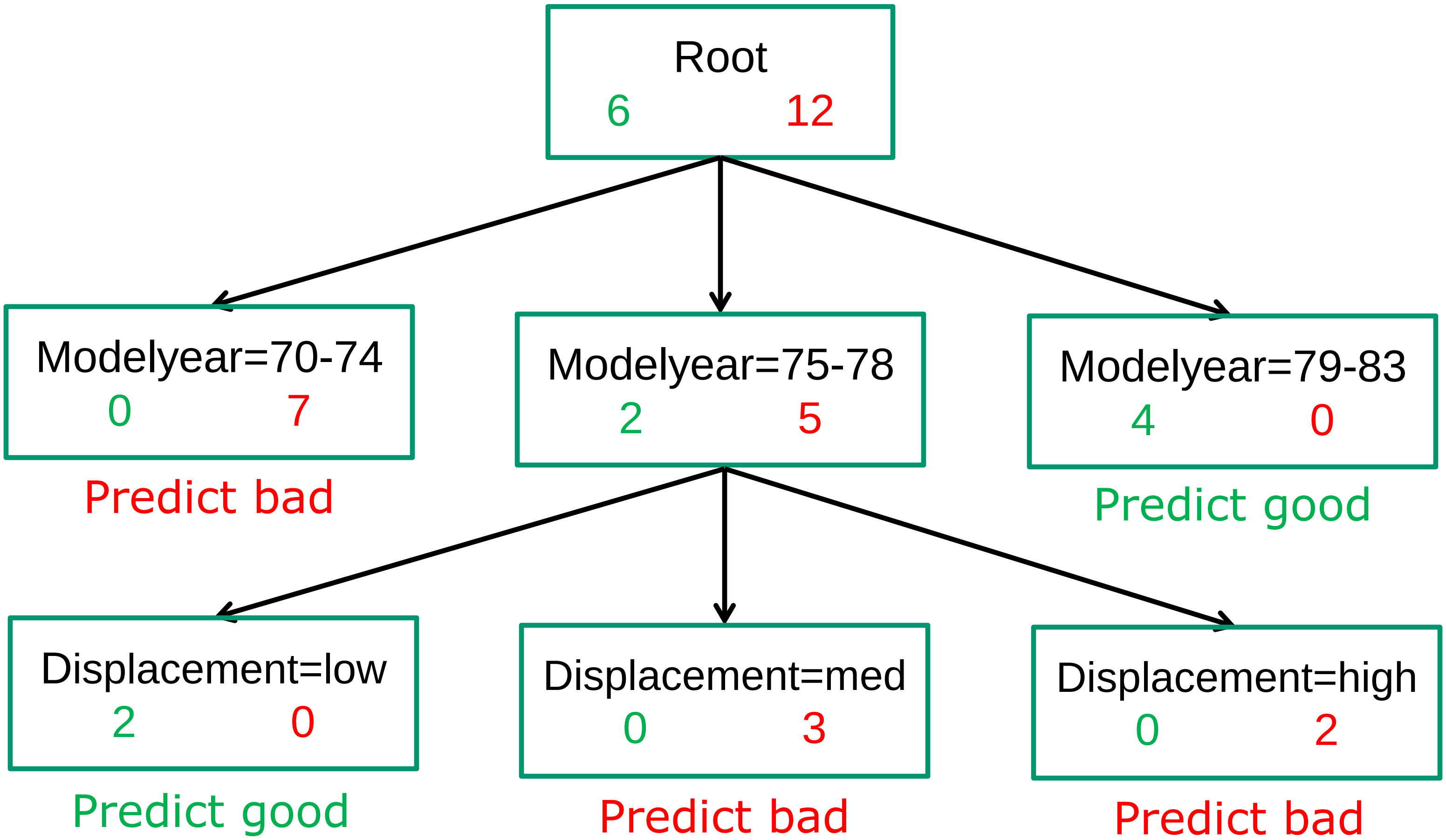
Second level of tree

- Only one node from first level needs expansion

1. Identify splits (on all attributes except *modelyear*)
2. Compute IGs
3. Select attribute with highest IG
   ➔ displacement OR weight

Attribute	Attr.Value	good	bad	Distribution	Entropy	Info Gain
cylinders	*4*	2	1		0,9183	
	5	0	1		0	
	6	0	1		0	
	8	0	2		0	
				Split	***0,3936***	***0,5247***
displacement	*low*	2	0		0	
	medium	0	3		0	
	high	0	2		0	
				Split	***0***	***0,9183***
horse power	*low*	1	0		0	
	medium	1	3		0,8113	
	high	0	2		0	
				Split	***0,4636***	***0,4547***
weight	*low*	2	0		0	
	medium	0	3		0	
	high	0	2		0	
				Split	***0***	***0,9183***
accelleration	*low*	0	3		0	
	medium	1	1		1,	
	high	1	1		1,	
				Split	***0,5714***	***0,3469***
maker	*Asia*	1	0		0	
	America	0	3		0	
	Europe	1	2		0,9183	
				Split	0,3936	0,5247

Second level of tree



Play-golf decision tree

‘Play golf/tennis’ data set

Outlook	Temperature	Humidity	Windy	Play?
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

- *Solve it at home as an exercise !*
- Discussion next lecture

Outline

- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

Evaluation

- When we use a machine learning model, we want to know how good it is (effectiveness)
  - To know how confident we can be in the predictions
  - To know which algorithm to use
  - ....
- ➔ *Model validation*

- Need to measure performance of an algorithm
  - Test on (labelled) data
  - Several different measures

- Orthogonal topic: efficiency, i.e. required runtime
  - *More on that later*

Evaluation (effectiveness)

- Binary classification (classes true/false)
- Table of confusion *(contingency table)*

		Actual value		
		true	false	
Test outcome	true	True positive (TP)	False positive (FP, Type I error)	***Precision*** $\frac{TP}{TP + FP}$
	false	False negative (FN, Type II error)	True negative (TN)	
		Recall Sensitivity $\frac{TP}{TP + FN}$		***Accur-acy*** $\frac{TP + TN}{\#\ samples}$

- *Examples of Type I & II errors?*
  - *Which is worse?*

Evaluation measures

	true	false
true	True positive (TP)	False positive (FP, Type I error)
false	False negative (FN, Type II error)	True negative (TN)

- Accuracy: # correctly predicted samples

$$\frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\#\ samples}$$

  - *Inverse: Error rate*
- Precision $\frac{TP}{TP + FP}$
- Recall $\frac{TP}{TP + FN}$
- F-Measure: trade-off between precision and recall; F1:

$$\frac{2 * (precision * recall)}{(precision + recall)}$$

- *Value ranges?*

Evaluation: data

- Which data to do evaluation on?

- *The samples used for training?*
  - *Why not?*

  - *If we test on training data – we are biased*
    - Perceptron on linear separable data: training data will always be 100% correctly "predicted"
    - Similar for other algorithms, e.g. Decision trees
    - K-NN, Naïve Bayes: not necessarily 100% correct on training data. Still biased!
  - *We want to actually find out:* how well will our model perform on **unseen** data!

Training & Test set split

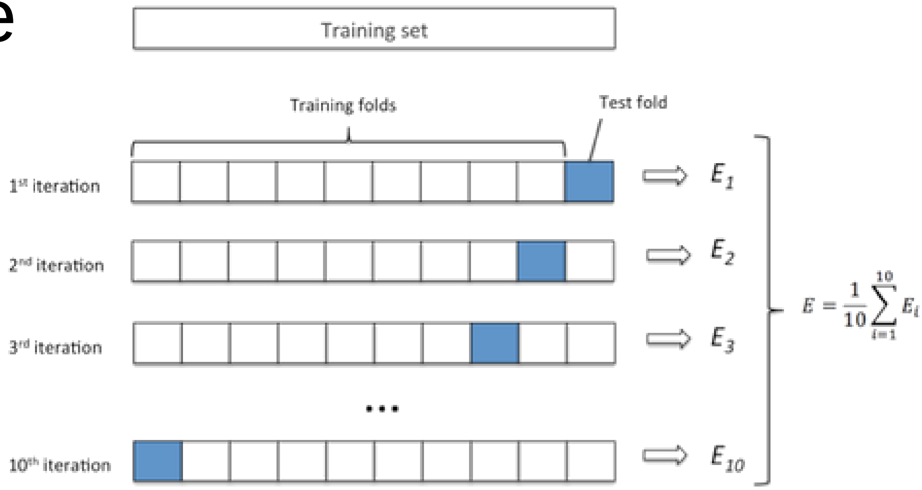
- Really unseen data doesn't have labels
  - "Simulate" "unseen" data
  - *"Holdout method"*
  - Split labelled data into training and test (validation) sets
    - E.g. ~80% training, 20% test, 66% - 33%
    - Linear (first 80%), randomised, …
- Performance on test set is an estimate for generalisation power
- Results can vary a lot according to how split is done

  ➔ *Cross validation*

k-fold Cross validation

- Split data into e.g. 10 parts of equal sizes
- This is called 10-fold cross validation
- repeat 10 times:
  - use 9 parts for training (training set)
  - calculate performance on remaining part (test set)

- Estimate of performance is average (mean) of the validation set performances



k-fold Cross validation

- Estimate of performance is average (mean)
- In addition to mean, compute standard deviation
  - Indication on how stable the results are in the folds

    ➔ lower standard deviation is better ...

  - Standard deviation to be considered when comparing cross-validation performances from different classifiers

Classifier / Fold	1	2
1	91,80	86,7
2	82,30	87
3	84,40	87,1
4	93,00	85,7
5	81,60	86,8
6	87,40	86,4
7	82,40	87,2
8	92,10	86,5
9	91,90	86,5
10	87,40	86,5
Mean	87,4	86,6
Stdv	4,6	0,4

k-fold Cross validation

- Which classifier is better:
  - Average 87,4%, standard deviation 4,8%
    - *(87,4%   4,8)*
  - Average 86,6%, standard deviation 0,4%
    - *(86,6%   0,4)*
    - ➔ *More on that later: significance testing*

k-fold Cross validation

- Results obtained via cross-validation are generally much more reliable
  - Parameter of 10 often used
  - But no theoretical foundation for that
  - Fewer folds on smaller sample size

- Number of folds increases runtime!
  - More-or-less linear with ***n***

- Might not be that critical – why?
  - Can be parallelised

Leave-*p*-out Cross validation

- A type of ***exhaustive*** cross-validation
  - Use *p* observations in test (validation) set
    - Remaining samples are in training set
  - Repeated for all combinations to cut *p* samples
  - Quickly becomes computationally infeasible
  - 100 samples, p=30
    - **3×10^{25}** combinations!

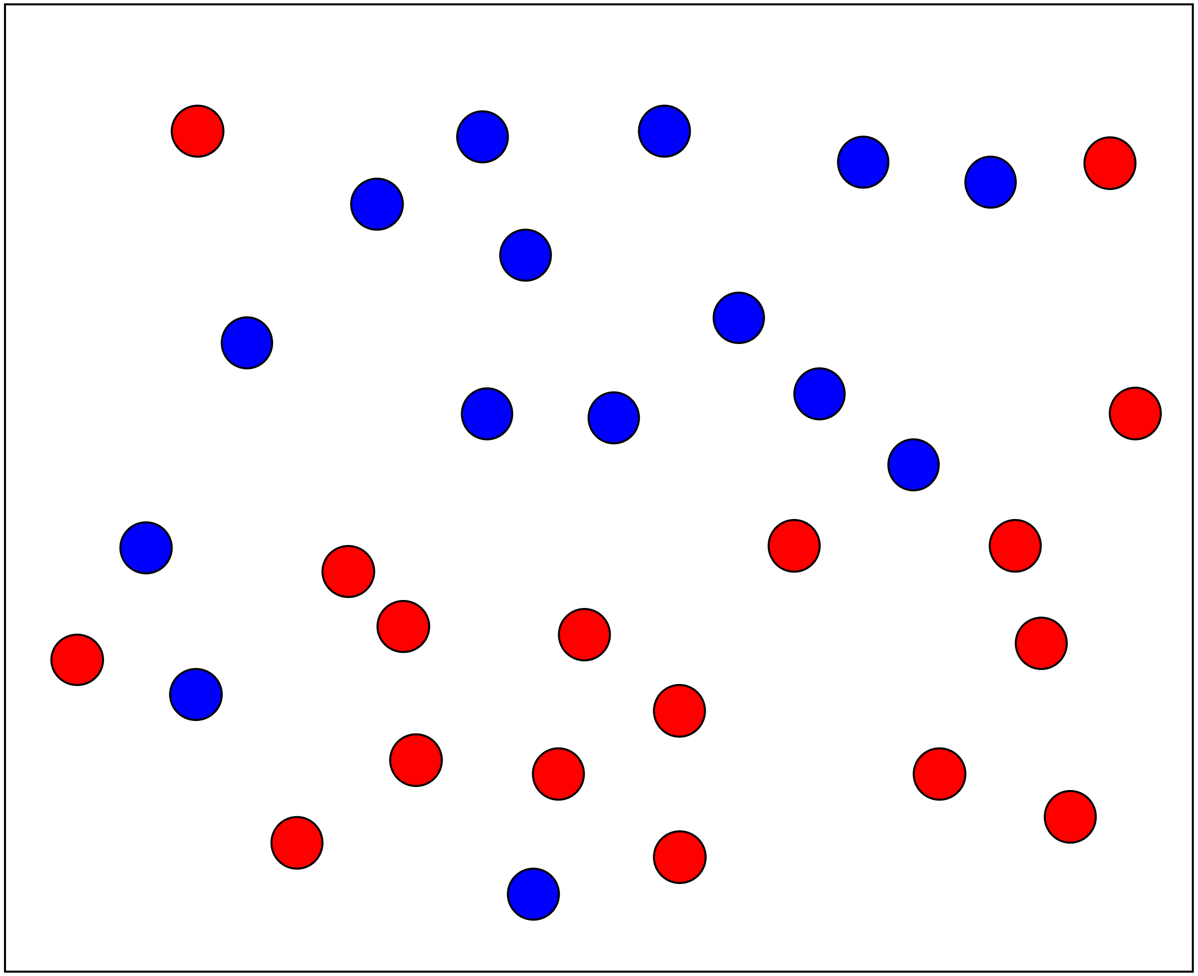
- Special case: p=1, leave-one-out cross validation
  - Test/validation set contains one sample
  - Number of combinations?
    - **n**

Bootstrapping

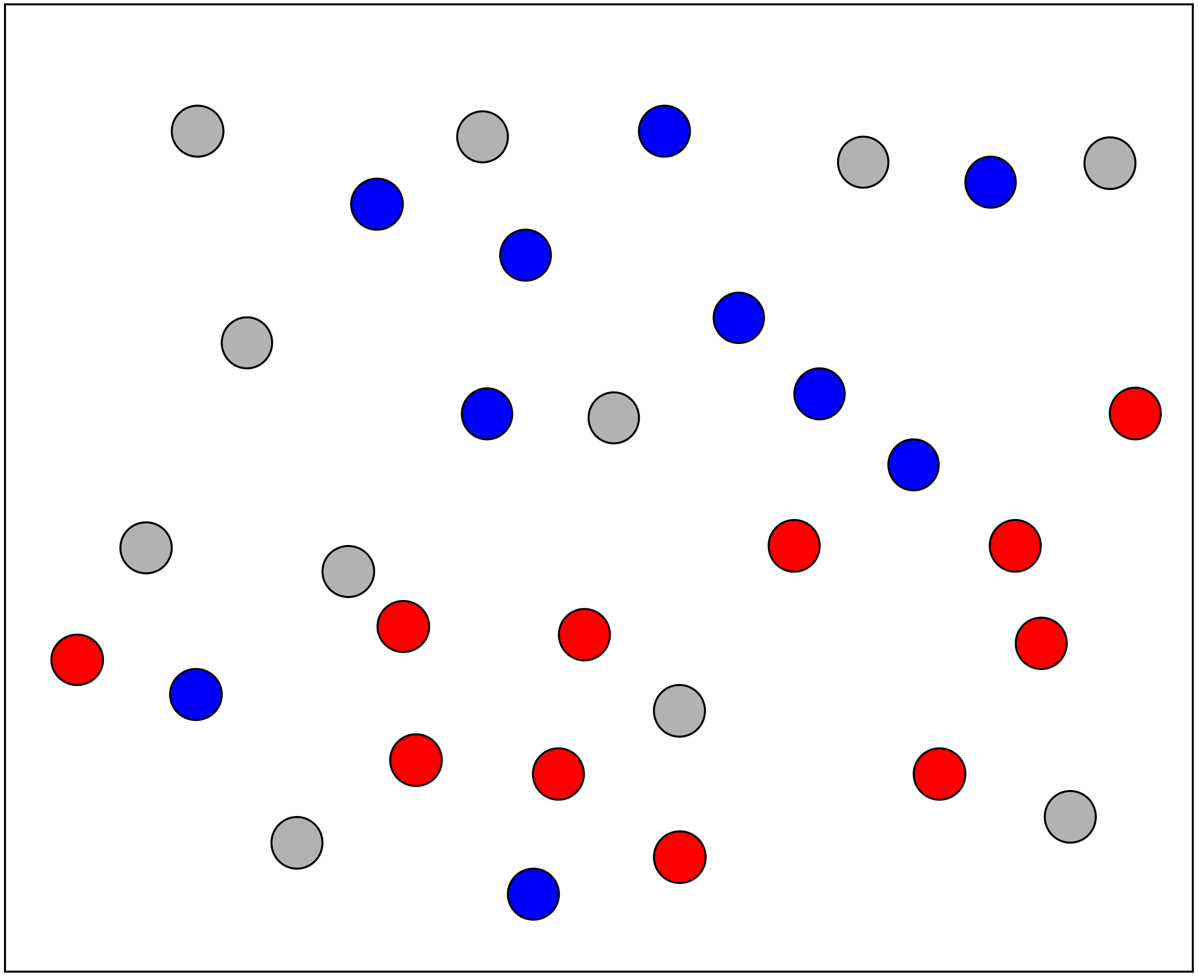
- A bootstrap sample is a random subset of the data sample
  - Test set is also random sample
- Data points may be selected repeatedly
  - i.e. selection with replacement
- An arbitrary number of bootstrap samples may be used

- Bootstrapping is an alternative to cross validation and training-test split

Example: Bootstrapping



Example: Bootstrapping



Evaluation: recap & outlook

- Table of confusion → accuracy, precision, ...
- Cross-validation
- Bootstrapping
- Micro vs. macro averaging
  - Confusion matrix
  - Cost functions
- Overfitting & Generalisation
- Evaluation measures for regression
- ROC curves, kappa statistic
- Bootstrapping
- Significance testing

Outline

- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

Random Forests

- Combination of Decision Trees and bootstrapping concepts
- Proposed ~1995 by Leo Breiman & Adele Cutler

- Basic Idea & Name: a large number of decision trees is “grown in the forest”, each on a different bootstrap sample

Random Forests

- For each tree: use bootstrap sample
- For each tree, only a random number of the original variables is available
  - i.e. small selection of columns
  - much smaller than original number
  - Change at each tree node!
- Grow trees to maximal extend
  - No stopping
  - No pruning
  - (they are rather small anyhow)

Example: Tree in Random Forests

Input Output

12	A	0.1	501	red	I
8	B	1.2	499	red	II
9	B	1.1	504	blue	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
-2	C	0.7	512	green	II
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
10	A	1.5	500	cyan	I
0	C	0.3	505	blue	II
9	B	1.9	502	blue	II

Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II

Bootstrap sample

Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II

?

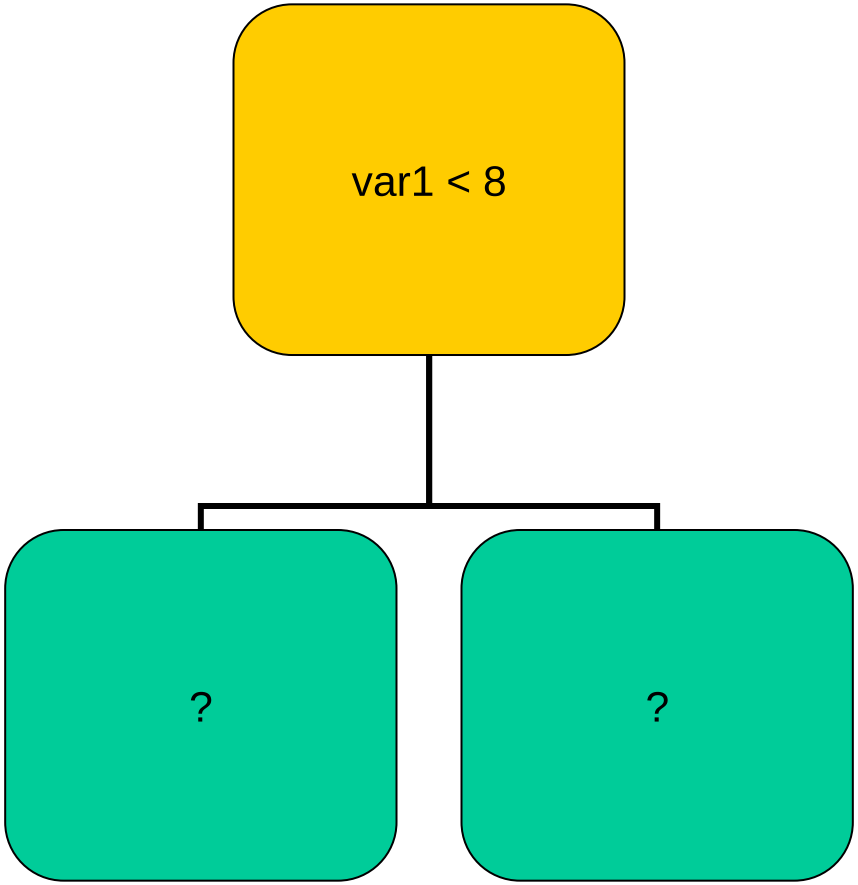
Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II

?

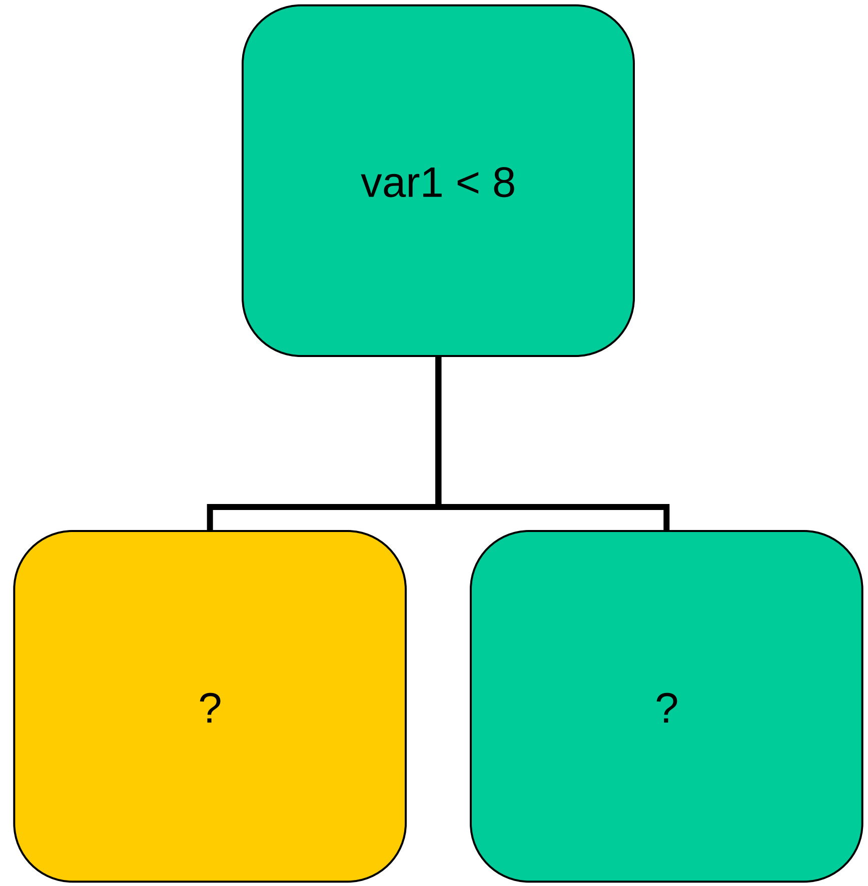
Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II

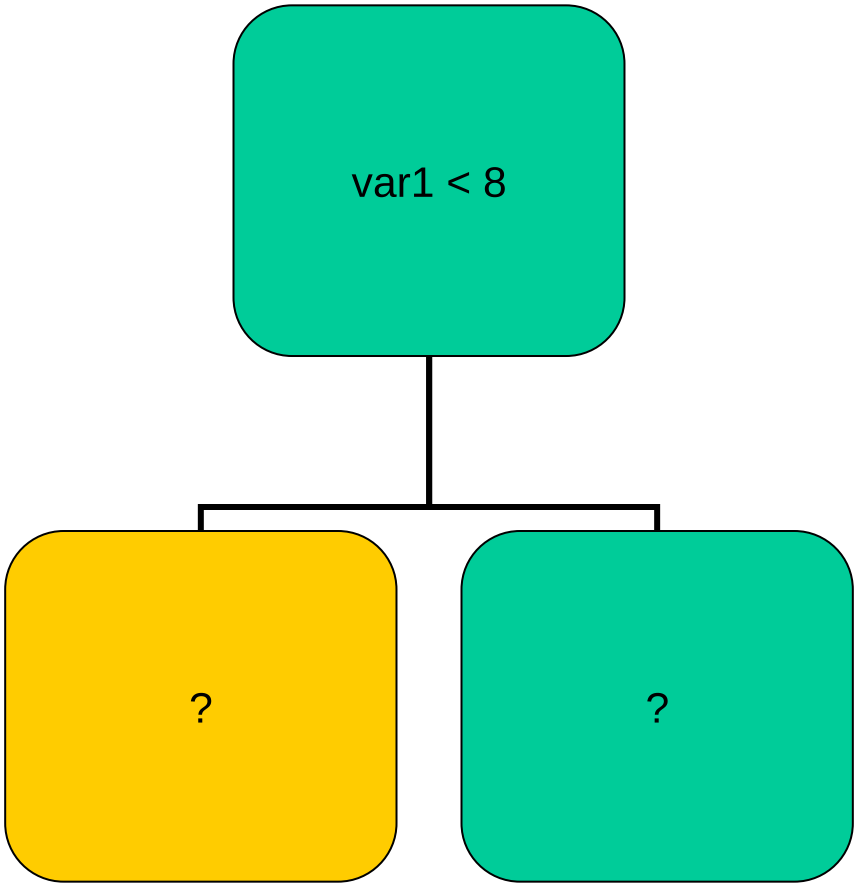


Example: Tree in Random Forests

!! Select new attributes at each tree node !!

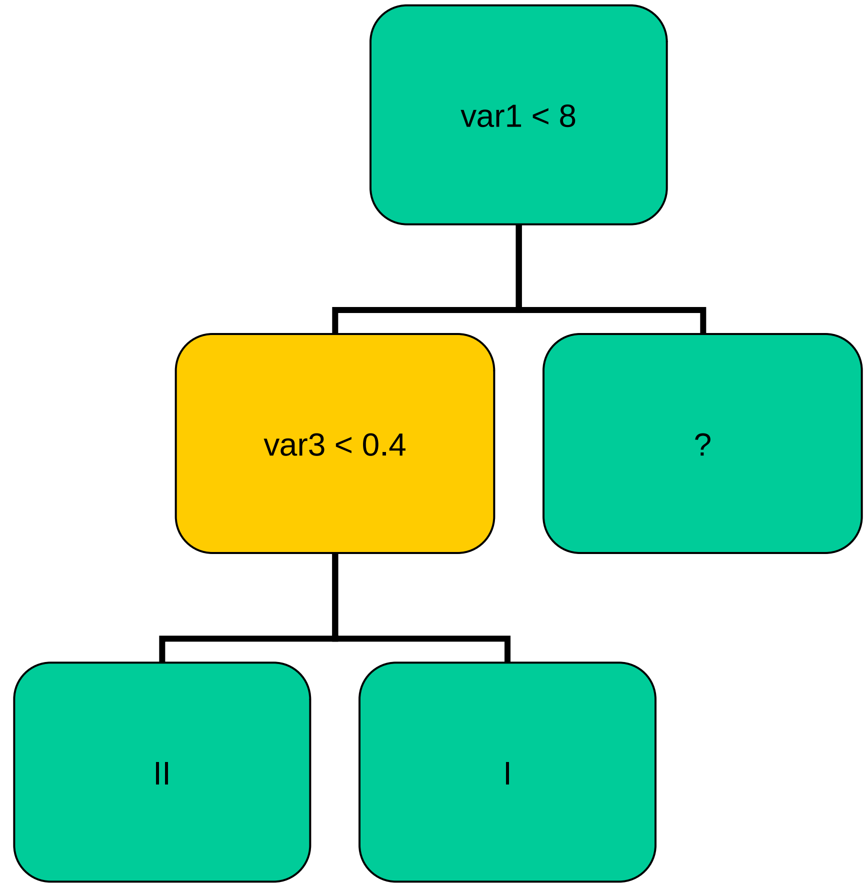
Input | Output

12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



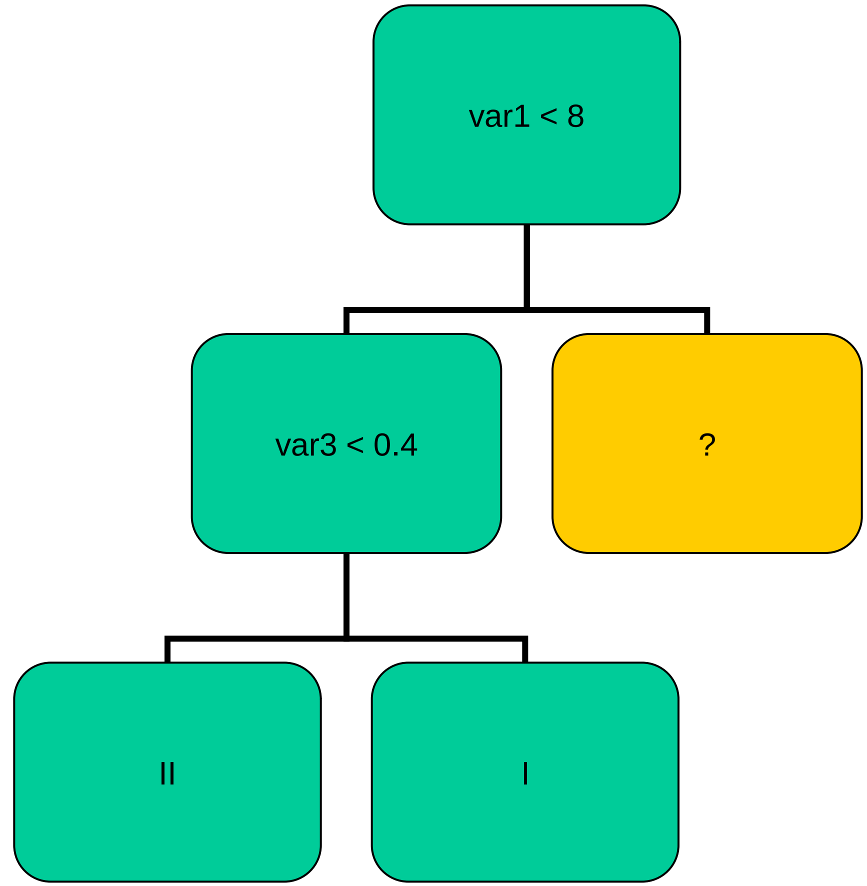
Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



Example: Tree in Random Forests

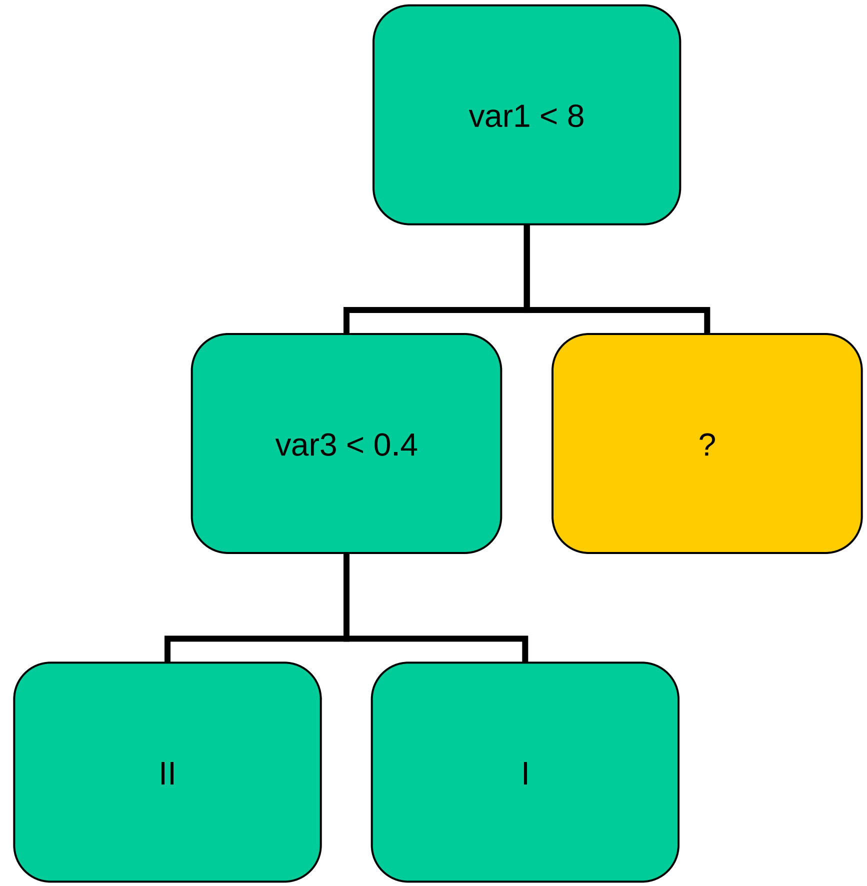
Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



Example: Tree in Random Forests

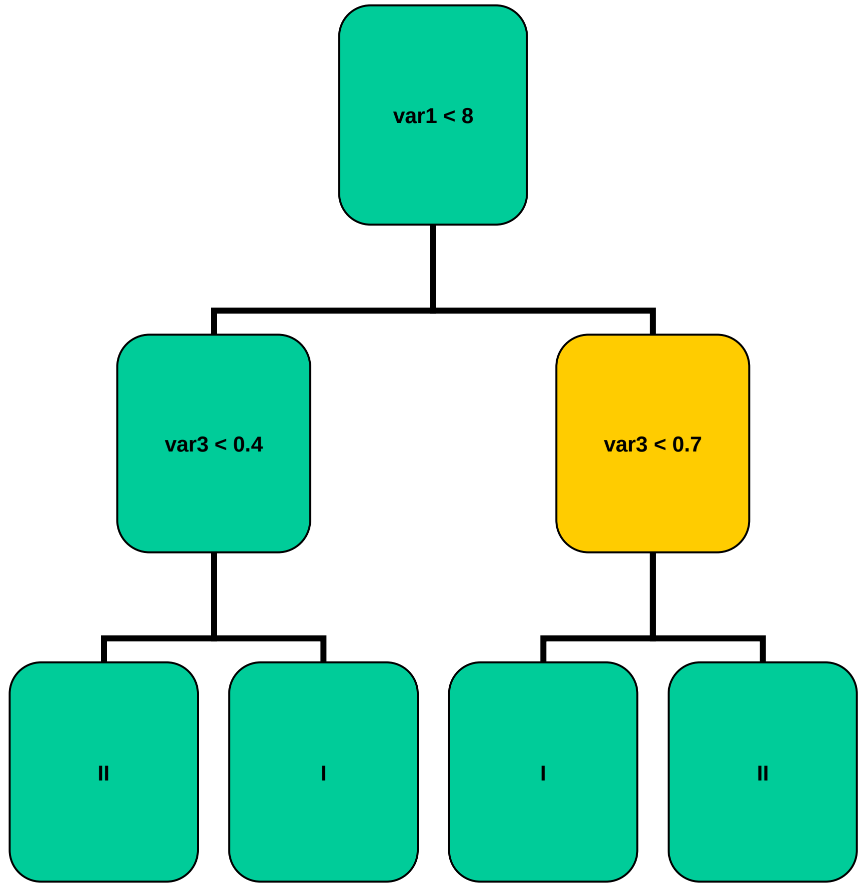
!! Select new attributes at each tree node !!

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



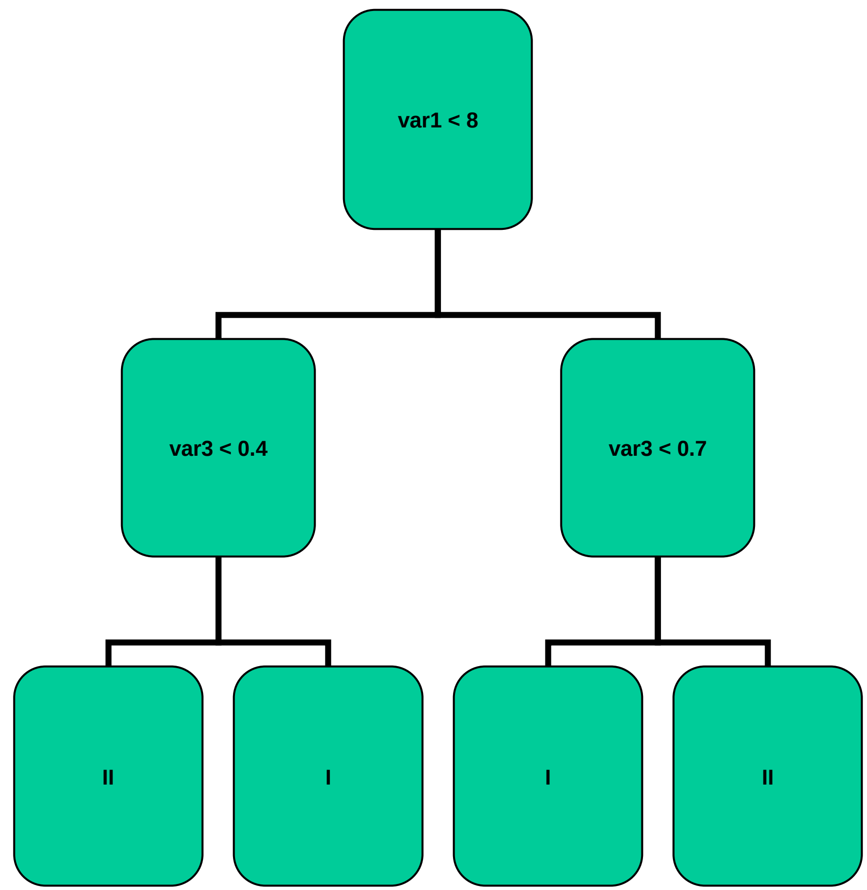
Example: Tree in Random Forests

Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



Example: Tree in Random Forests

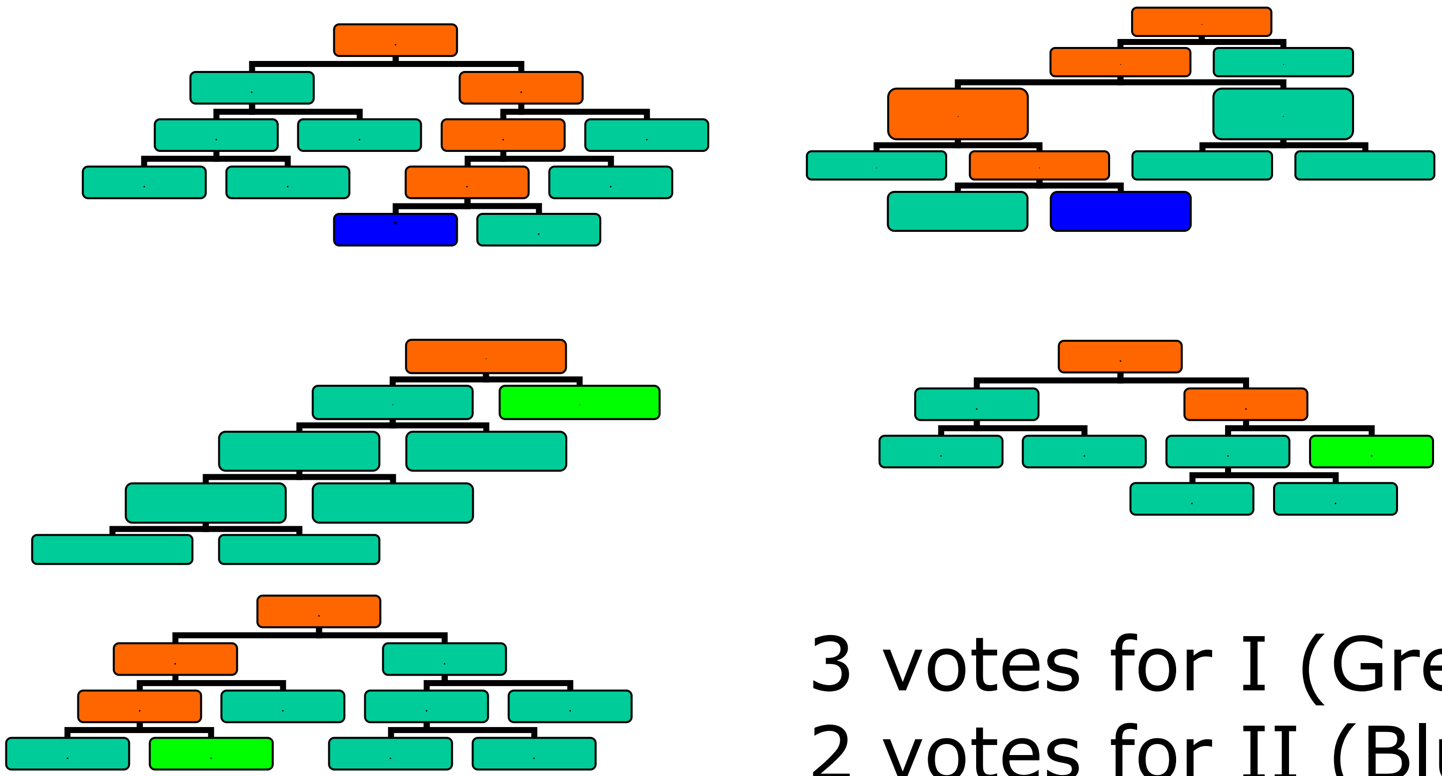
Input					Output
12	A	0.1	501	red	I
8	B	1.2	499	red	II
15	A	1.8	480	green	II
2	C	1.0	511	red	I
7	C	0.4	488	cyan	I
7	A	0.6	491	cyan	I
0	C	0.3	505	blue	II



Random Forests

- Train a number of trees
  - Tens, hundreds – or sometimes even more
- Classify new data by majority voting of the individual trees
  - Count which class is predicted by most trees

Classification with Random Forests



3 votes for I (Green)
2 votes for II (Blue)
=> classify as I

Properties of Random Forests

- Only few parameters (number of trees, number of variables for split)
  - Good default values, rather robust
- Still mostly simple concepts
- Very high accuracy for many data sets
- No over-fitting when selecting large number of trees

- Can be memory consuming, slow
  - Can be parallelised

Outline

- Recap
  - Perceptron & k-NN, continued
- Decision trees
- Evaluation, continued
- Random Forests

Confusion Matrix

- Matrix of classification results per class
  - Size (# classes) x (# classes)
- For each actual class plot the predicted classes
- Shows accuracy for single classes
- Indicates which classes are confused

Confusion Matrix: Example

	Grey	Black	Red	*Accuracy*
Grey	5	3	0	*0.625*
Black	2	3	1	*0.500*
Red	0	1	12	*0.920*
				0.740

- Ideally: numbers only in the diagonal
  - In other cells: indicates misclassification

Confusion Matrix

- Important to analyse mistake patterns
  - *Which classes get mixed up?*

classified as											
a	b	c	d	e	f	g	h	i	j	k	genre
34	3	0	0	2	8	0	0	2	10	1	a = Country
9	39	0	1	1	4	0	0	0	5	1	b = Folk
0	2	47	0	1	4	1	0	1	4	0	c = Grunge
0	2	0	39	0	3	1	6	8	0	1	d = Hip-Hop
2	3	3	0	34	4	10	0	0	4	0	e = Metal
10	3	9	4	4	11	3	2	1	11	2	f = Pop
5	2	5	0	10	2	36	0	0	0	0	g = Punk Rock
2	0	0	10	0	3	0	40	2	1	2	h = R&B
0	1	0	7	0	1	0	2	45	0	4	i = Reggae
8	1	8	1	3	5	1	1	1	27	4	j = Slow Rock
1	0	0	0	0	1	0	1	3	2	52	k = Children’s

Confusion Matrix

- Important to analyse mistake patterns
  - Which classes get mixed up

classified as											
a	b	c	d	e	f	g	h	i	j	k	genre
34	3	0	0	2	8	0	0	2	10	1	a = Country
9	39	0	1	1	4	0	0	0	5	1	b = Folk
0	2	47	0	1	4	1	0	1	4	0	c = Grunge
0	2	0	39	0	3	1	6	8	0	1	d = Hip-Hop
2	3	3	0	34	4	10	0	0	4	0	e = Metal
10	3	9	4	4	11	3	2	1	11	2	f = Pop
5	2	5	0	10	2	36	0	0	0	0	g = Punk Rock
2	0	0	10	0	3	0	40	2	1	2	h = R&B
0	1	0	7	0	1	0	2	45	0	4	i = Reggae
8	1	8	1	3	5	1	1	1	27	4	j = Slow Rock
1	0	0	0	0	1	0	1	3	2	52	k = Children's
47	69	65	63	62	23	69	76	71	42	77	Precision
57	65	78	65	57	18	6	67	75	45	87	Recall

Confusion Matrix: Example

	BigClass	SmallClass	*Accuracy*
BigClass	490	0	*100*
SmallClass	10	0	*0*
			0.98

Evaluation measures – averages

- Previous measures are ***micro-averaged***
- Do not indicate issues with imbalanced classes
- Alternative: macro-averaged measures
  - Compute precision, recall, ... ***per class***
  - Average class-results

Evaluation – micro average

	BigClass	SmallClass	*Accuracy*
BigClass	490	0	*100*
SmallClass	10	0	*0*
			0.98

- Accuracy: $$\frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\#\ samples}$$

Evaluation – macro average

	BigClass	SmallClass	*Accuracy*
BigClass	490	0	*100*
SmallClass	10	0	*0*
			0.5

- Accuracy: $$\frac{1}{|C|}\sum_{i=1}^{|C|}\frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i}$$

Performance per class

- Important to consider when
  - imbalanced classes
  - Performance of a particular class is more important

- *Examples ?*
  - Health prediction
  - Classify sensitive documents, …
  - Spam filter
  - Identify malicious software

Costs of misclassification

- Cost / loss functions
  - Measures per class with weighted averages
  - Higher weight to classes where errors are more severe

  ➔ Requires expert knowledge to identify weights

Effectiveness & Efficiency

- Effectiveness: quality of classification
  - Accuracy, precision, recall, F1, …
- Efficiency: computational efficiency (speed, runtime) of a classification
- Performance: often used as *synonym* for either *effectiveness OR efficiency* !

Effectiveness & Efficiency

- What is more important?

- Trade-off between effectiveness & efficiency
- Differentiate between efficiency on
  - Training (learning) a model
  - Classification

- Efficiency is more relevant if model needs to be (re-)trained frequently

Questions?

Upcoming topics:

- Neural Networks
- Support Vector Machines
- Ensemble Learning
- Evaluation, continued (e.g. significance testing)