DATA MINING

CLASSIFICATION 1

OVERVIEW

Introduction to classification

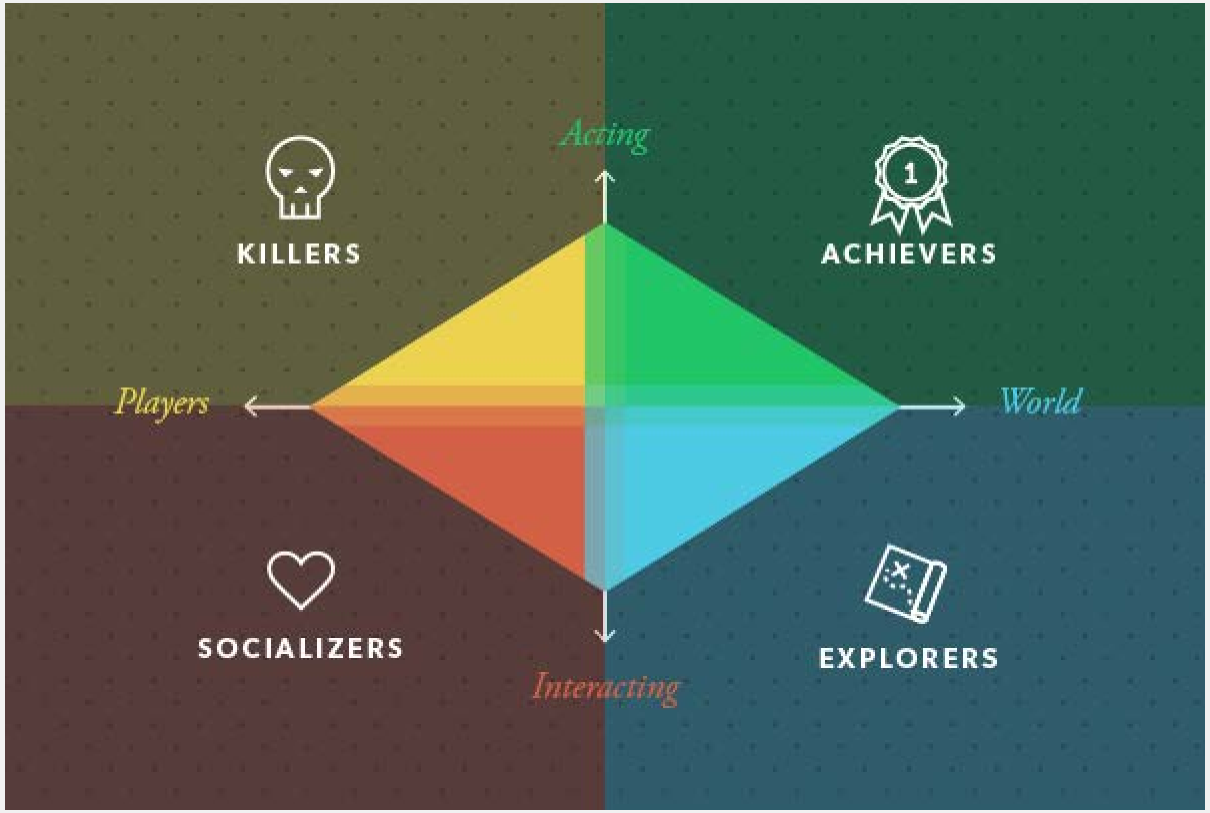
Algorithms

- Decision tree induction
- K-nearest neighbour algorithm

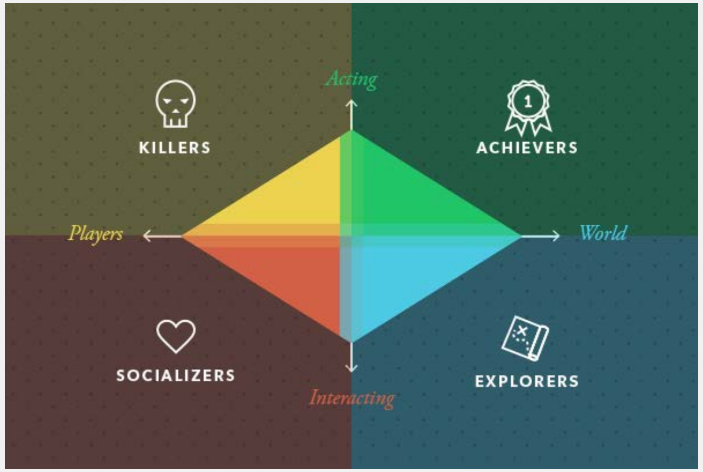
Evaluation

- Basic metrics
- Comparing classification models

EXAMPLE: BARTLE'S TAXONOMY

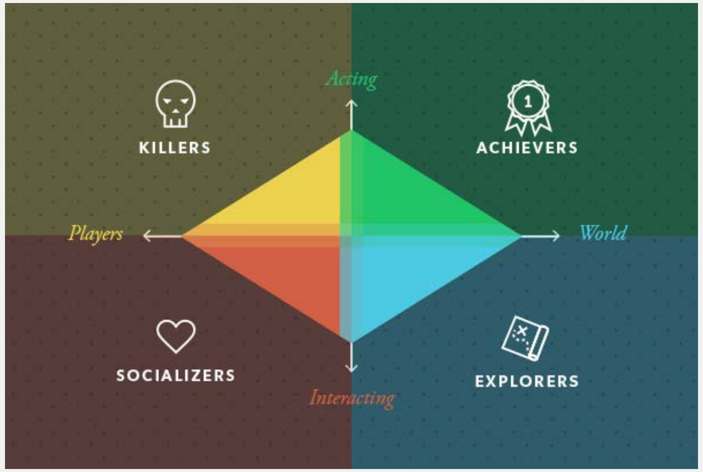


PREDICTION PROBLEM



- Suppose you have a data set of players with basic information (age, occupation, citizenship…)
- These players have been sorted into classes
- Can you predict the class label for a new player?
- Can you predict her expected average playing time?

PREDICTION PROBLEM



- Can you predict the class label for a new player?
  - This is a **classification** problem
- Can you predict her expected average playing time?
  - This is **numeric prediction**

CLASSIFICATION VS CLUSTERING

Supervised learning (classification)

- Training data accompanied by labels
- **Labels** indicate the object's class
- New data classified based on training set

Unsupervised learning (clustering)

- The **class** of training data objects is **unknown**
- Goal: establish the existence of classes
- Later in the course!

CLASSIFICATION

1. Model construction

   Using the **training set** data

2. Accuracy estimation

   Using the **test set** data

3. Data classification

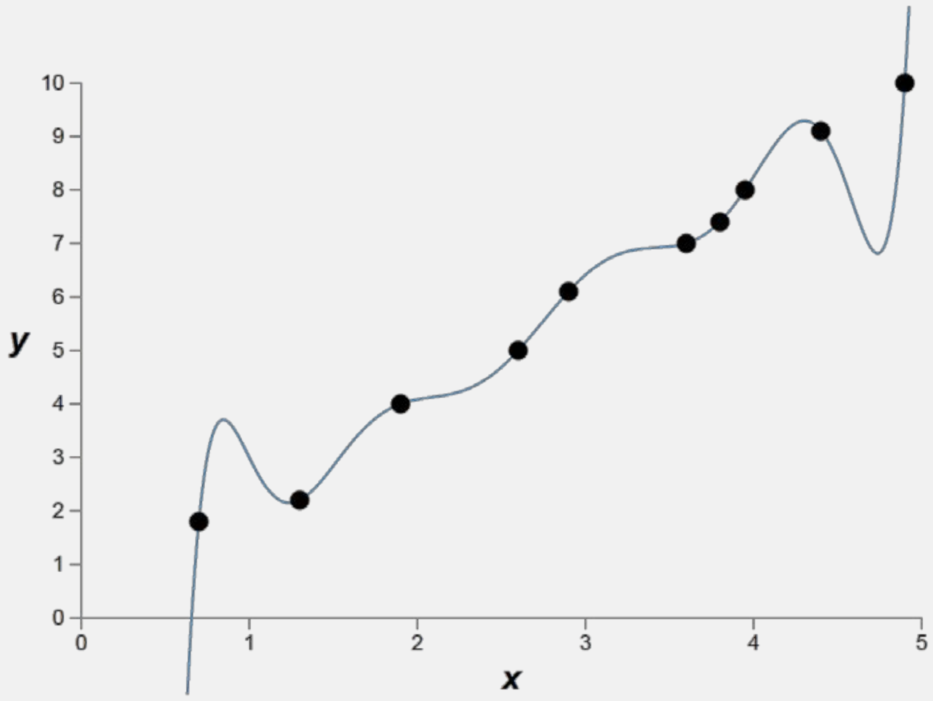
   Using new data

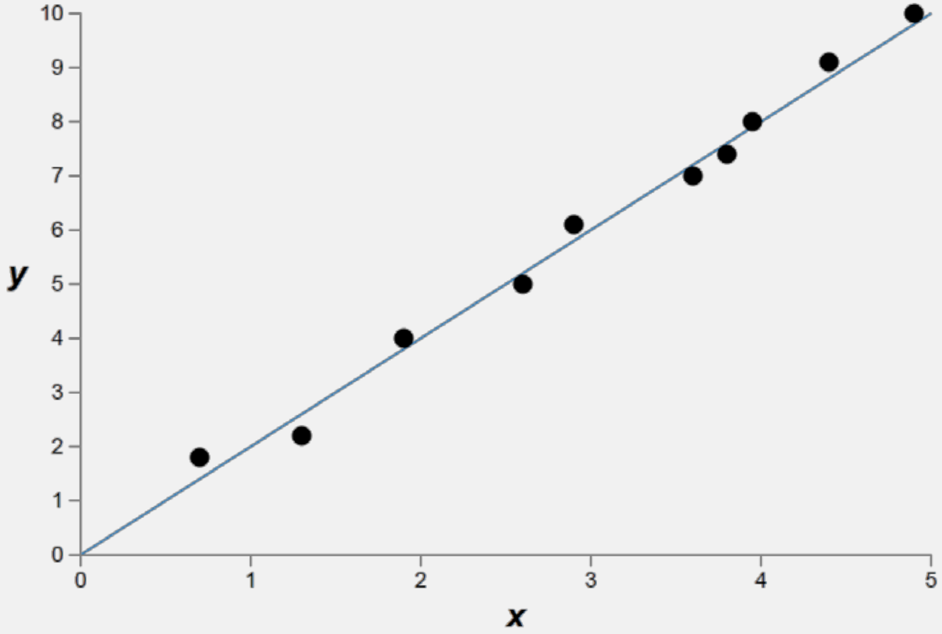
CLASSIFICATION

Why do we need to have different data sets?

Why not train and estimate on the same data?

CLASSIFICATION—OVERFITTING





Check Michael Nielsen's [Neural Networks and Deep Learning]

CLASSIFICATION—DATA SETS

- **Training** data

  Used to tune (train) the algorithm

- **Validation** data

  Used to **choose** best **algorithm** or
  to find the best **hyper-parameters**

- **Test** data

  Used to evaluate the accuracy of the model

ALGORITHMS

EAGER VS LAZY

Eager learners

- Uses training data to build a general model
- Queries have no effect on the model
- Long training time, fast classification
- Deals better with noise

EAGER VS LAZY

Lazy (“instance-based”) learners

- Stores training data (minimal processing)
- Processing only when each query is received
- Can solve multiple problems simultaneously
- Needs to store lots of data
- Slower evaluation
- Useful with large datasets with few attributes
- It works even if not all data is available in the beginning

EAGER VS LAZY

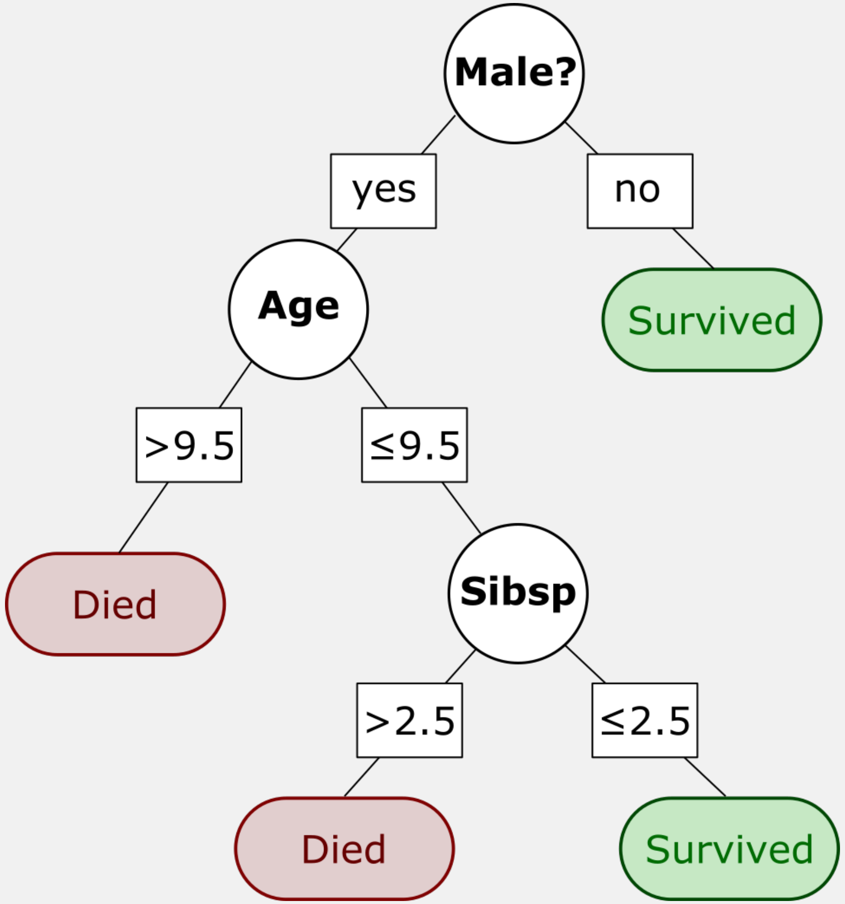
Today:

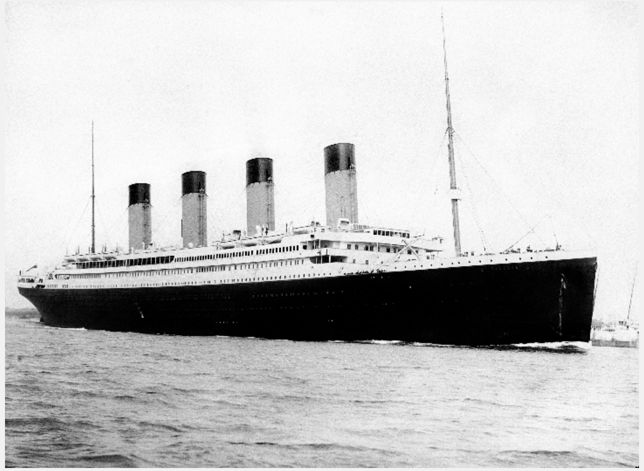
Decision tree induction (eager)

K-nearest neighbours (lazy)

DECISION TREE INDUCTION

DECISION TREE





DECISION TREE

Readable by humans

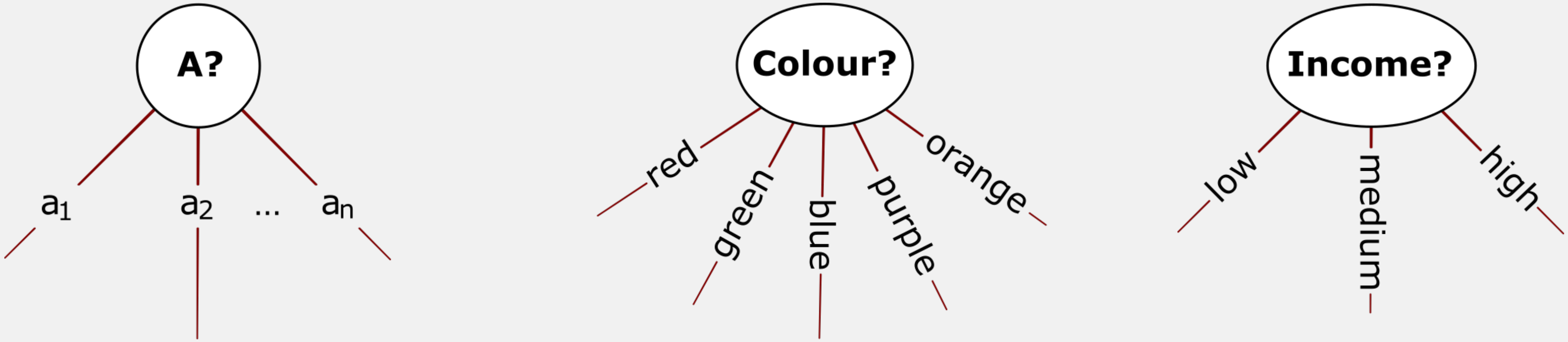
Requires no domain knowledge

Little or no parameters

Knowledge discovery?

High-accuracy

PARTITION SCENARIOS

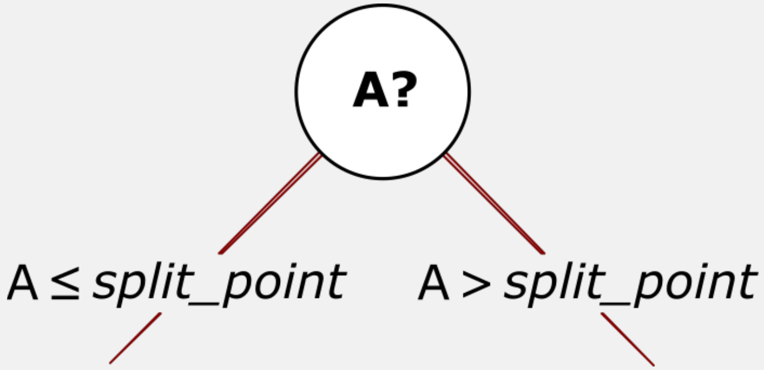


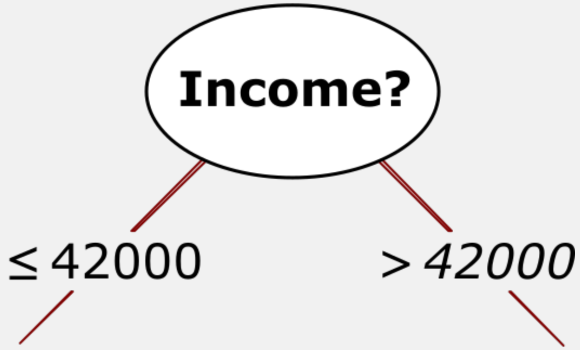
Discrete-valued attribute

The attribute is removed from the list of splitting candidates

One branch for each value (possible empty sets!)

PARTITION SCENARIOS



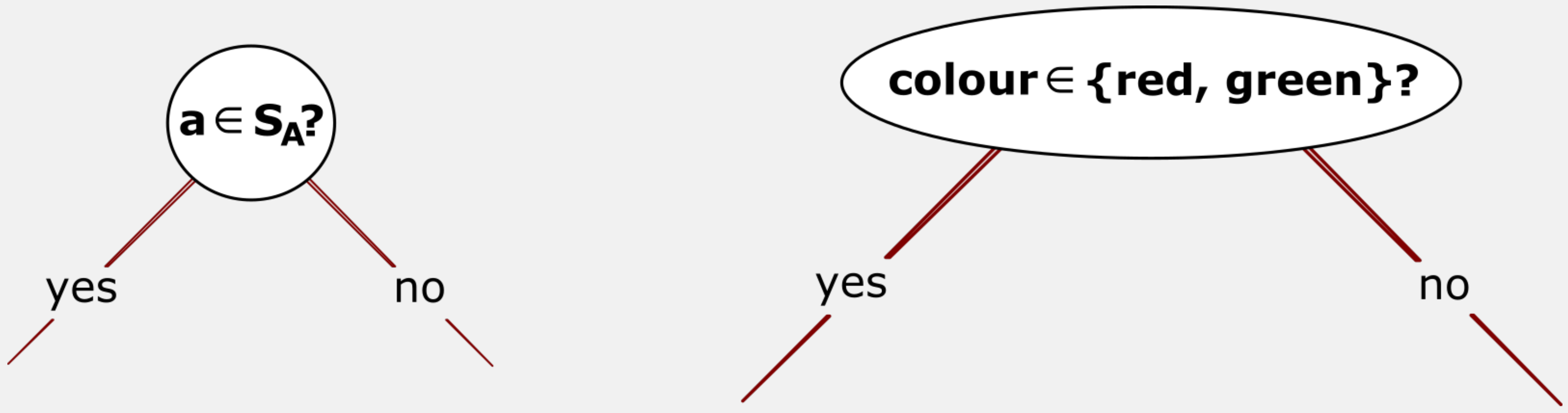


Continuous-valued attribute

The attribute is **not** removed from the list of splitting candidates

Two branches (attributes at either side of the split)

PARTITION SCENARIOS



Discrete-valued attribute (and binary tree)

The attribute is **not** removed from the list of splitting candidates

Two branches (attribute value in the subset or not)

ID3 ALGORITHM

Greedy algorithm:

Makes the locally best decision at every step

Global optimum?

ID3 ALGORITHM

Age	Income	Student	Credit rating	Buys computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
(30, 40]	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
(30, 40]	low	no	excellent	yes
≤ 30	medium	yes	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
(30, 40]	medium	no	excellent	yes
(30, 40]	high	yes	fair	yes
> 40	medium	no	excellent	no

ID3 ALGORITHM

- Create a node
- Do all tuples have same label? (false)
- No more possible splitting criteria left? (false)

Age	Income	Student	Credit rating	Buys computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
(30, 40]	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
(30, 40]	low	no	excellent	yes
≤ 30	medium	yes	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
(30, 40]	medium	no	excellent	yes
(30, 40]	high	yes	fair	yes
> 40	medium	no	excellent	no

ID3 ALGORITHM

- Apply **attribute selection measure** to find splitting criterion
- Returns: **age**
- (How? In a minute)

Age	Income	Student	Credit rating	Buys computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
(30, 40]	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
(30, 40]	low	no	excellent	yes
≤ 30	medium	yes	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
(30, 40]	medium	no	excellent	yes
(30, 40]	high	yes	fair	yes
> 40	medium	no	excellent	no

ID3 ALGORITHM

Age

≤ 30

Income	Student	Credit rating	Buys computer
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

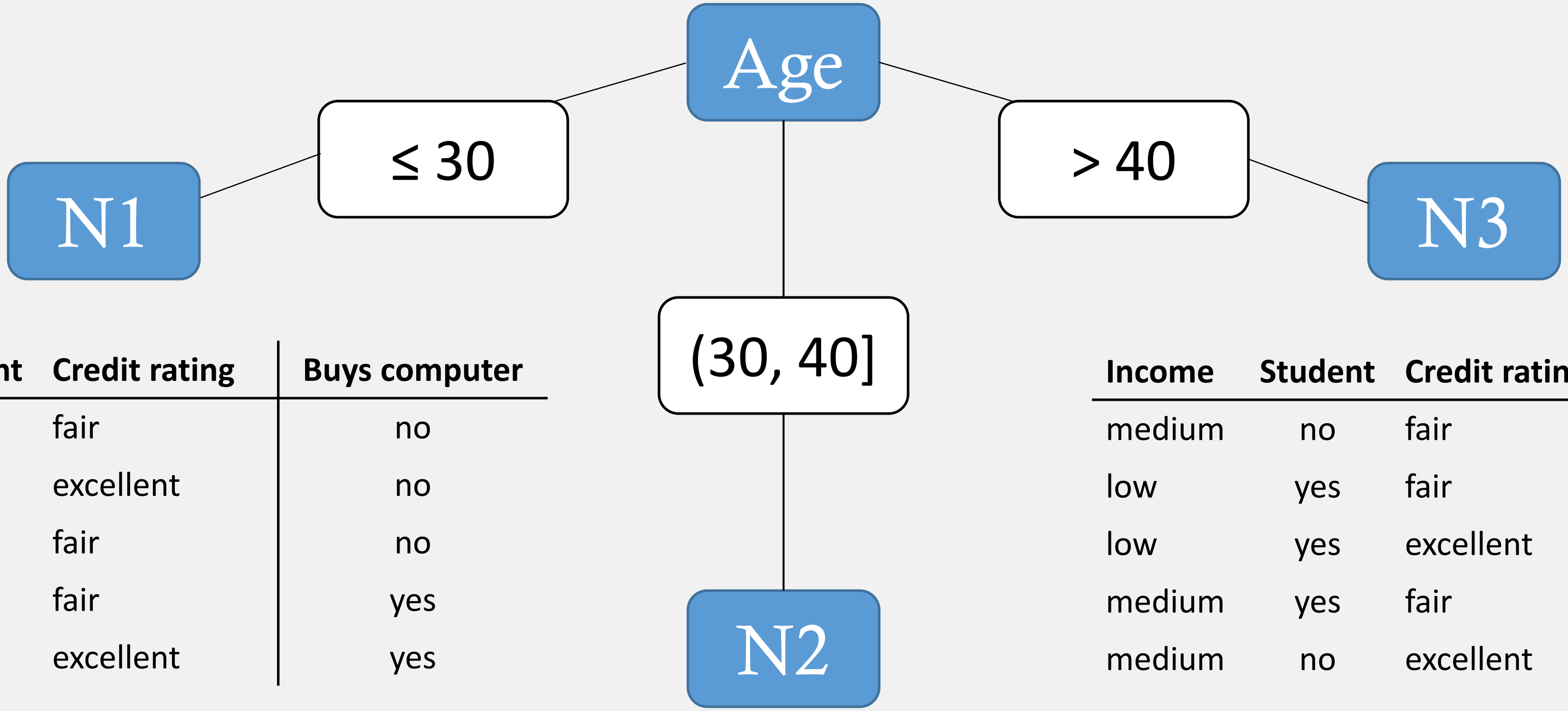
(30, 40]

Income	Student	Credit rating	Buys computer
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

> 40

Income	Student	Credit rating	Buys computer
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

ID3 ALGORITHM

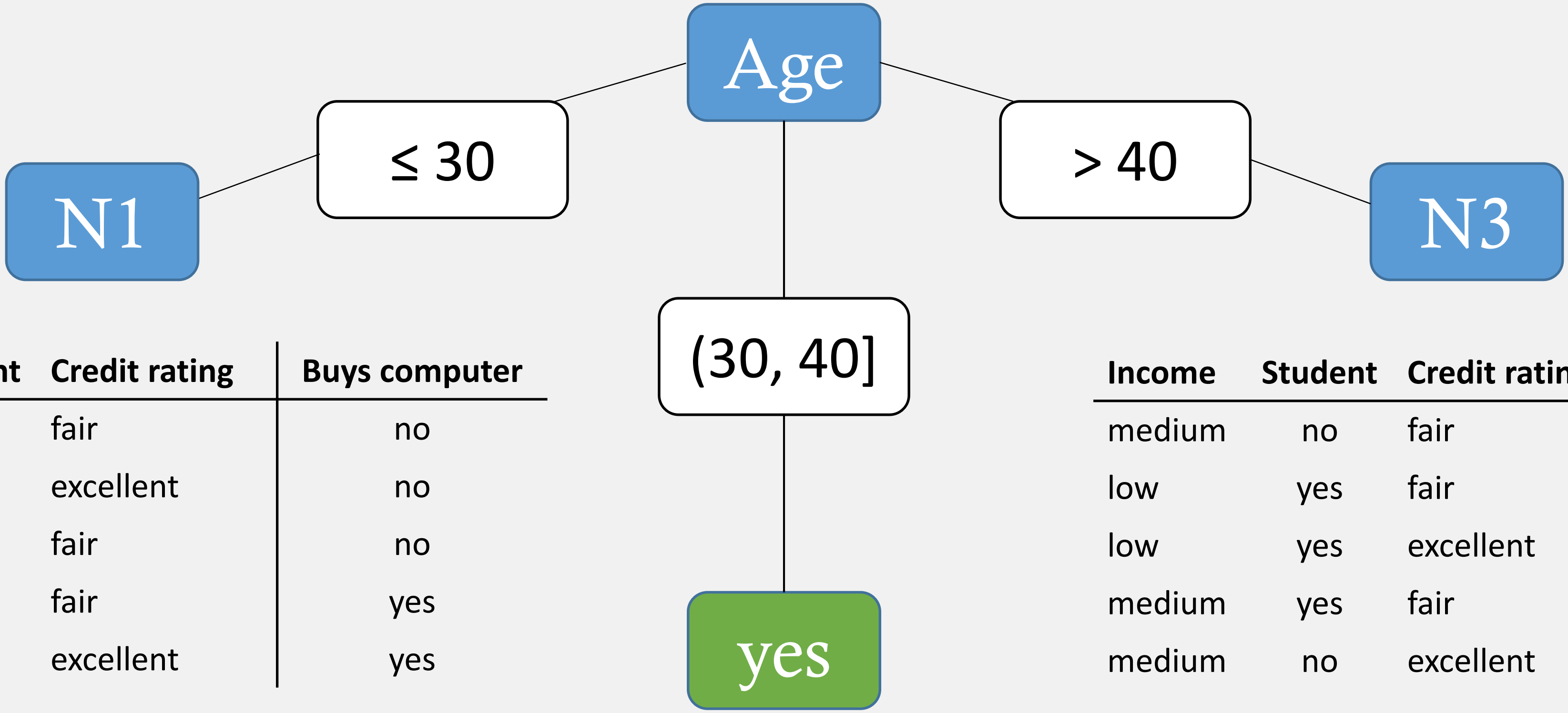


Income	Student	Credit rating	Buys computer
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

Income	Student	Credit rating	Buys computer
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Income	Student	Credit rating	Buys computer
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

ID3 ALGORITHM

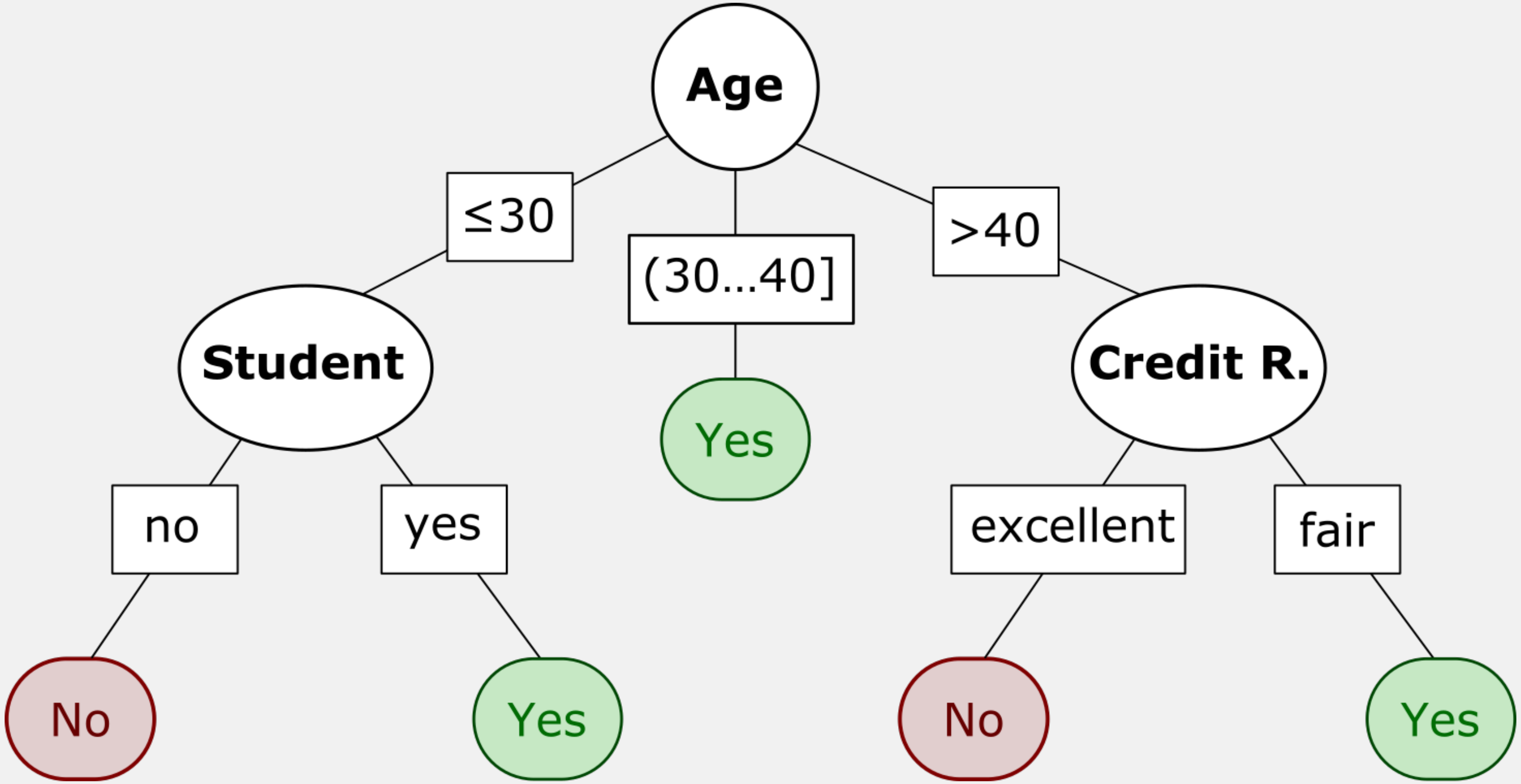


Income	Student	Credit rating	Buys computer
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

Income	Student	Credit rating	Buys computer
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Income	Student	Credit rating	Buys computer
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

ID3 ALGORITHM



ATTRIBUTE SELECTION MEASURE

Needed to compare different split options

ID3: **information gain**

C4.5: **gain ratio**

INFORMATION GAIN

Idea:

Select the attribute that **minimizes the information needed** to classify tuples in the resulting data partitions

INFORMATION GAIN

Entropy gives the expected **information needed to classify a tuple** in D

$$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

m: number of class labels

p_i: probability that a tuple in D belongs to class C_i

INFORMATION GAIN

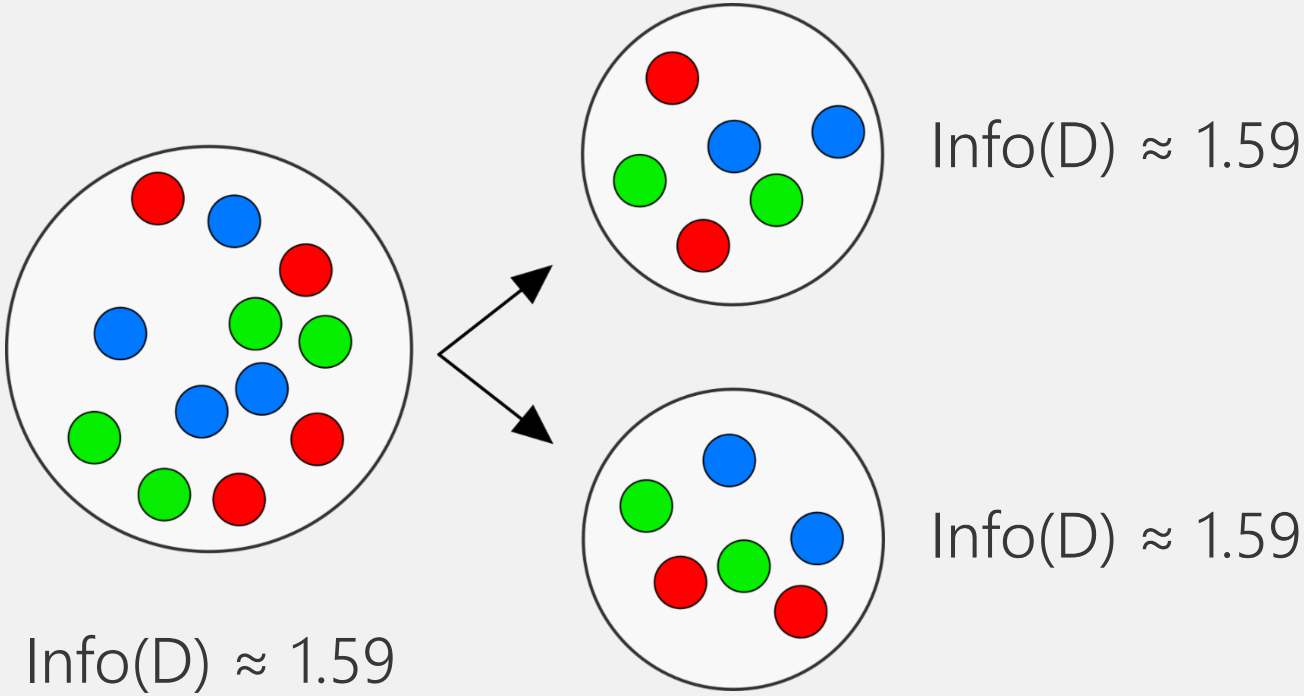
Suppose we split D using attribute A:

- We can apply the same formula to the subsets!
- We want the expected **information** needed **to classify** a tuple taken **from** any of the *v* **subsets** $\{D_1, D_2, \ldots D_v\}$

$$\mathrm{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \mathrm{Info}(D_j)$$

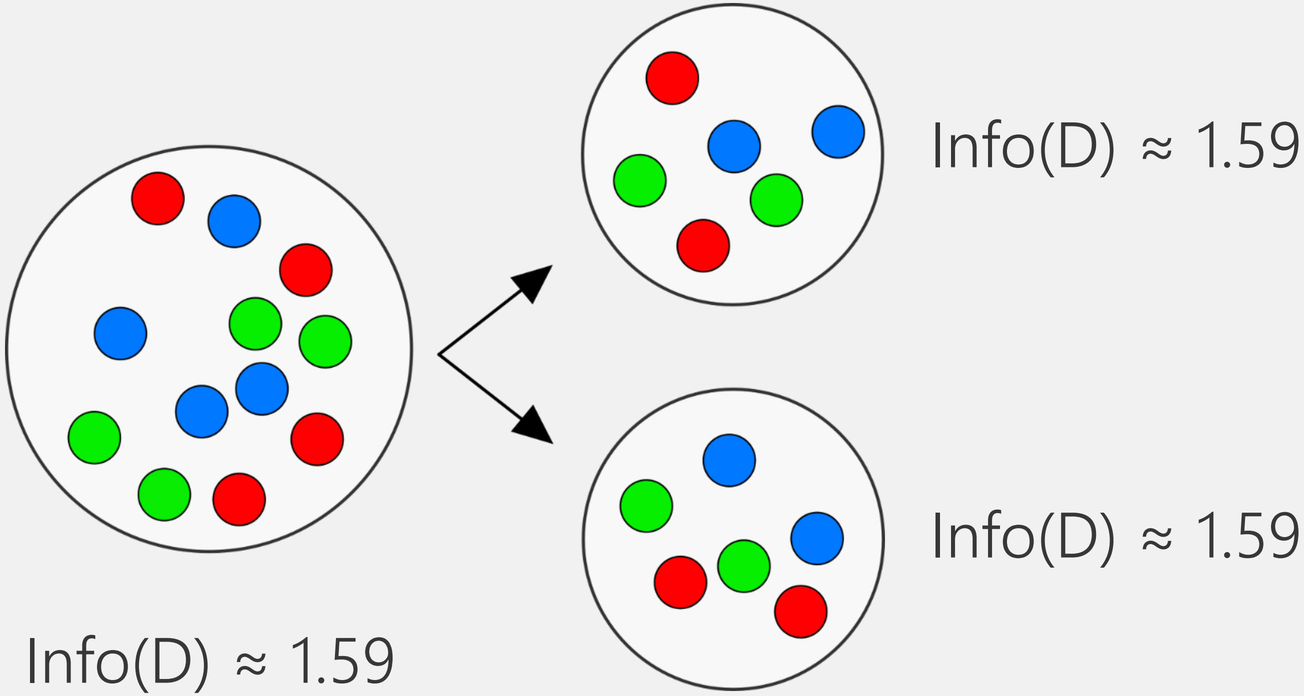
EXPECTED INFORMATION—NOTE

$$\text{Info}\left(D\right) = -\sum_{i=1}^{m} p_i \log_2\left(p_i\right)$$



EXPECTED INFORMATION—NOTE

$$\text{Info}_A(D) = \sum_{j=1}^{v} \boxed{\frac{|D_j|}{|D|}} \times \text{Info}(D_j)$$



INFORMATION GAIN

- We have the information needed to classify a tuple before and after a set partition.
- Then we have the information gain for splitting using the attribute A:

$$\mathrm{Gain}\left(A\right) = \mathrm{Info}\left(D\right) - \mathrm{Info}_A\left(D\right)$$

INFORMATION GAIN

Let’s calculate the information gain for the age split in our example!

Age	Income	Student	Credit rating	Buys computer
≤ 30	high	no	fair	no
≤ 30	high	no	excellent	no
(30, 40]	high	no	fair	yes
> 40	medium	no	fair	yes
> 40	low	yes	fair	yes
> 40	low	yes	excellent	no
(30, 40]	low	no	excellent	yes
≤ 30	medium	yes	fair	no
≤ 30	low	yes	fair	yes
> 40	medium	yes	fair	yes
≤ 30	medium	yes	excellent	yes
(30, 40]	medium	no	excellent	yes
(30, 40]	high	yes	fair	yes
> 40	medium	no	excellent	no

INFORMATION GAIN

We are classifying for the {**Buys computer**} label

Out of 14 tuples, 9 buy computers, 5 do not.

$$\text{Info}\left(D\right) = -\sum_{i=1}^{m} p_i \log_2\left(p_i\right)$$

$$\text{Info}\left(D\right) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) \simeq 0.940 \text{ bits}$$

INFORMATION GAIN

$$\mathrm{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \mathrm{Info}(D_j)$$

After the split

For partition D_1 → 2 tuples buy, 3 don't, 5 total (out of 14):

$$\mathrm{Info}_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) \right) +$$

INFORMATION GAIN

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

After the split

For partition D_2 → 4 tuples buy, 0 don't, 5 total (out of 14):

$$\text{Info}_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right) +$$

$$+\frac{4}{14} \times \left(-\frac{4}{4}\log_2\left(\frac{4}{4}\right)\right) +$$

INFORMATION GAIN

$$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$

After the split

For partition D_3 → 3 tuples buy, 2 don't, 5 total (out of 14):

$$\text{Info}_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right) +$$

$$+\frac{4}{14} \times \left(-\frac{4}{4}\log_2\left(\frac{4}{4}\right)\right) +$$

$$+\frac{5}{14} \times \left(-\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) \simeq 0.694 \text{ bits}$$

INFORMATION GAIN—NOTE

What about the term…?

$$-\frac{0}{4}\log_2\left(\frac{0}{4}\right)$$

Remember that

$$\lim_{x \to 0^+} x \cdot \ln(x) = 0$$

(Try L'Hôpital)

INFORMATION

Finally, the information gain for splitting on Age is:

$$\mathrm{Gain}\,(age) = \mathrm{Info}\,(D) - \mathrm{Info}_{age}\,(D) \simeq 0.940 - 0.694 = 0.246 \text{ bits}$$

If we try the other available attributes we find

- Gain(*income*) = 0.029 bits
- Gain(*student)* = 0.151 bits
- Gain(*credit rating*) = 0.048 bits

CONTINUOUS ATTRIBUTES

- Consider the **midpoint** between each **pair** of adjacent (sorted) **values** as possible split point
- Compute the information gain for each case with
  - Subset D_1 for tuples where A ≤ *split point*
  - Subset D_2 for tuples where A > *split point*
- Computationally demanding!

INFORMATION GAIN VS GAIN RATIO

- **Information gain** is **biased** towards attributes with **many values**!
- *Student ID* creates trivial subsets of one student
  (so they are “perfectly” classified)

GAIN RATIO

Expected information to determine the subset D_i of a tuple in D:

$$\mathrm{SplitInfo}\,(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2 \left(\frac{|D_j|}{|D|} \right)$$

SplitInfo(2 equal subsets) = 1 bit

SplitInfo(8 equal subsets) = 3 bits

GAIN RATIO

We weight the information gain with the entropy resulting from subdividing the data into the subsets $\{D_1, D_2, \ldots D_v\}$

$$\mathrm{GainRatio}(A) = \frac{\mathrm{Gain}(A)}{\mathrm{SplitInfo}(A)}$$

Choose the attribute with highest gain ratio

GAIN RATIO

$$\mathrm{GainRatio}\left(A\right) = \frac{\mathrm{Gain}\left(A\right)}{\mathrm{SplitInfo}\left(A\right)}$$

- Gain ratio becomes unstable for low split info!
- Constraint: the information gain must be at least as large as the average gain in all tests examined

OTHER SELECTION MEASURES

Gini, CHAID (χ^2), C-SEP…

See the book for more!

ID3 PSEUDO CODE

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

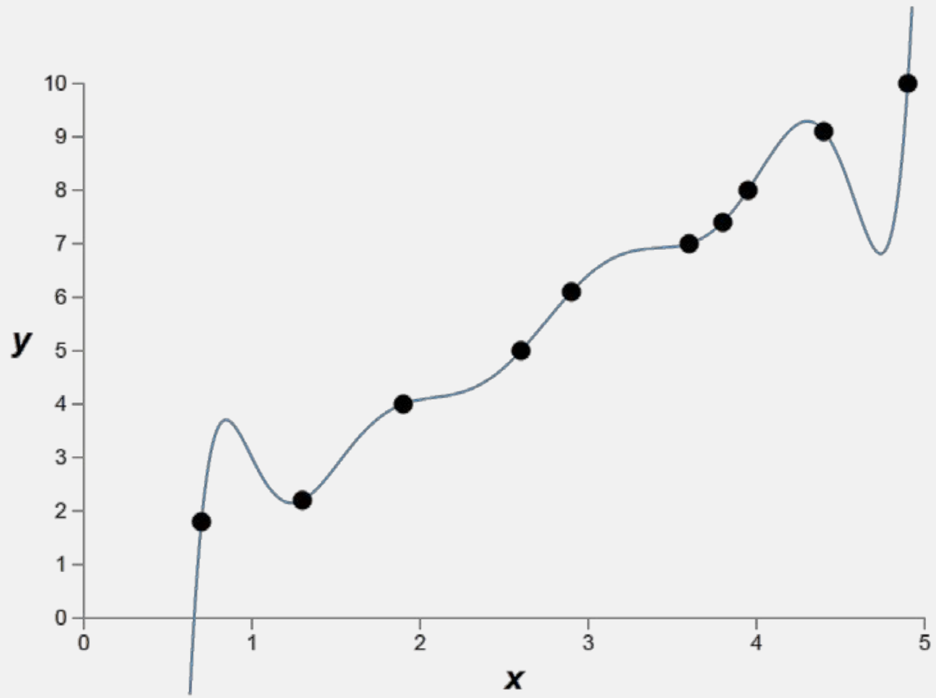
Output: A decision tree.

ID3 PSEUDO CODE

Method:

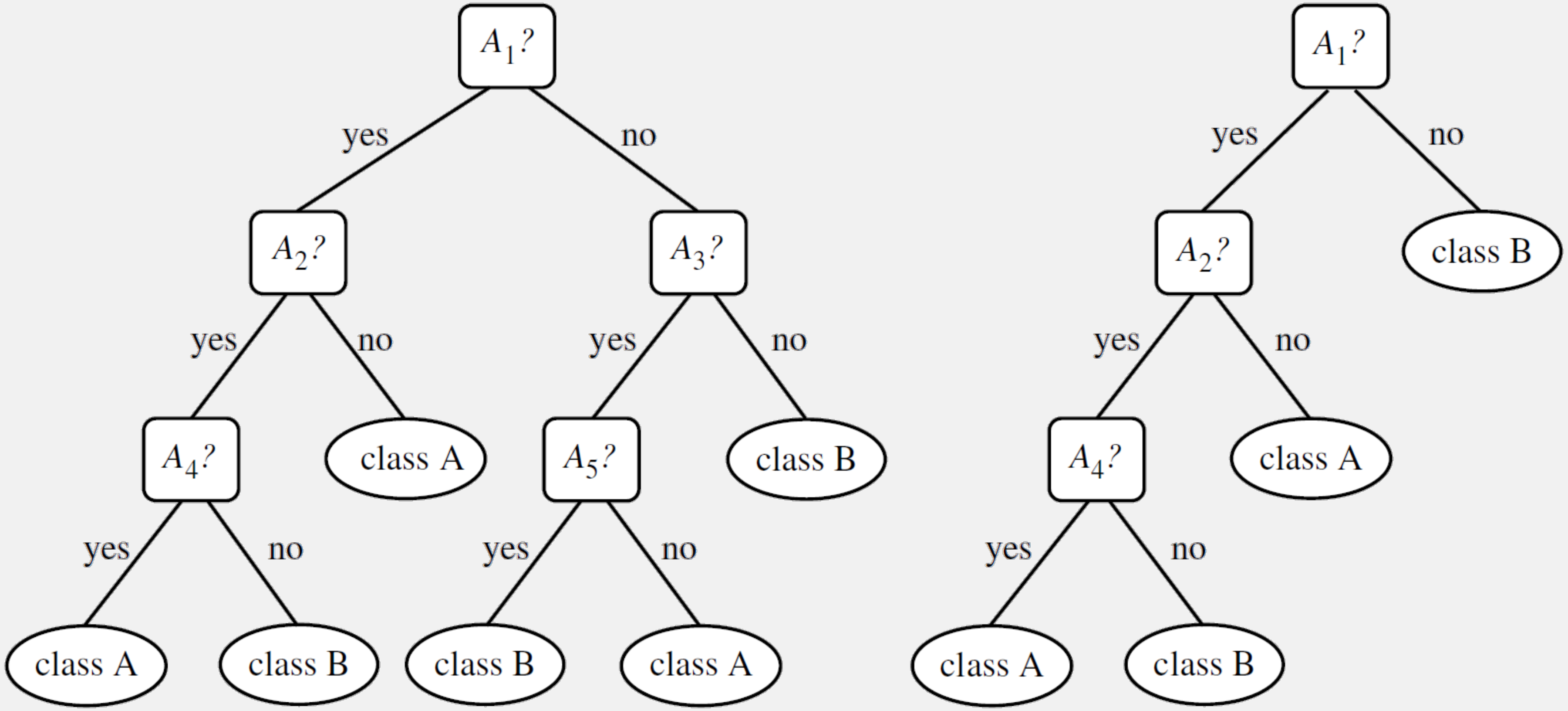
```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12)     if D_j is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(D_j, attribute_list) to node N;
     endfor
(15) return N;
```

TREE PRUNNING



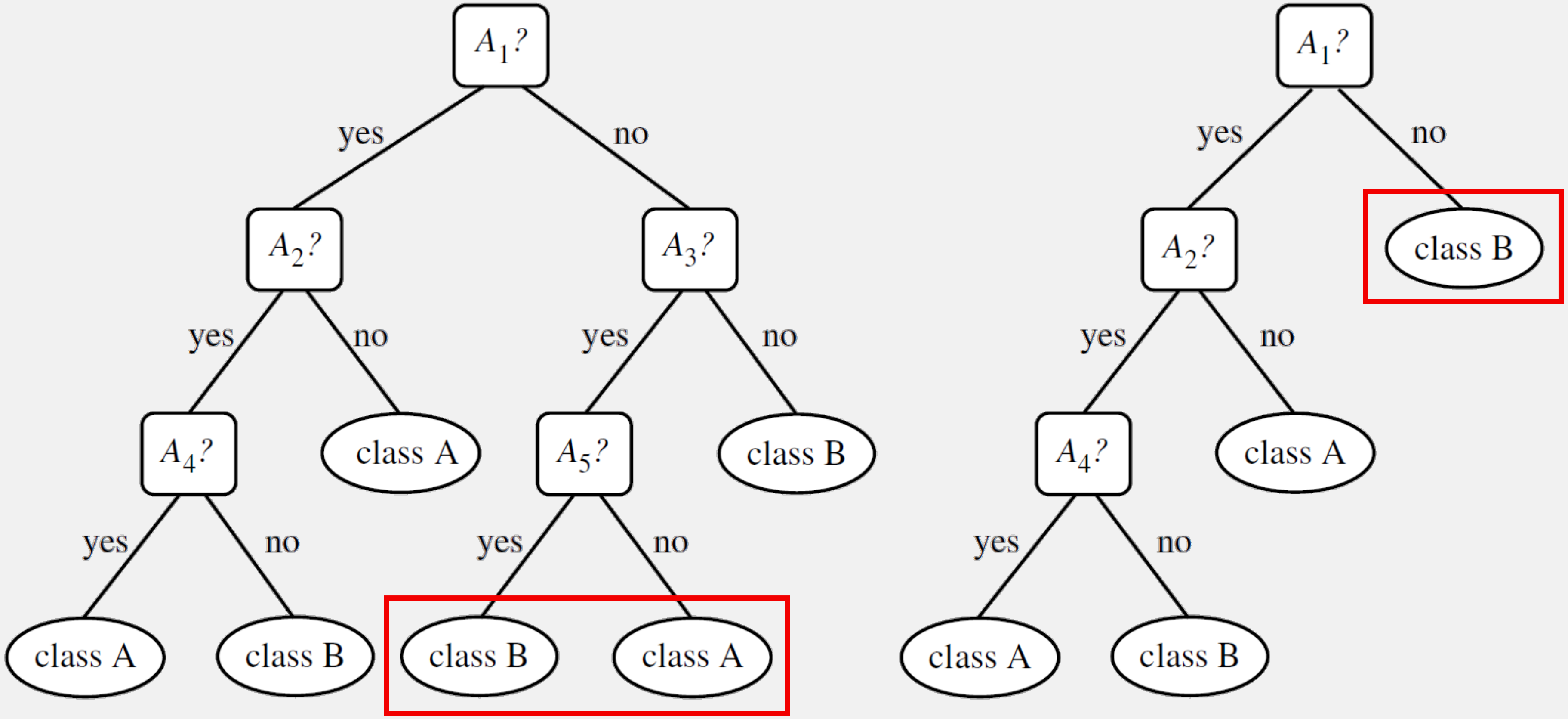
- **Overfitting**: the decision tree reflects noise or particularities in the training data
- Overfitted models generalize poorly

TREE PRUNNING



Remove least-reliable branches to increase the quality of the tree

TREE PRUNNING



Remove least-reliable branches to increase the quality of the tree

TREE PRUNING

Pre-pruning

- Do not split a node if the benefit measure (e.g. gain ratio) is less than a threshold
- Create leaf with most frequent class
- Hard to find a good threshold!

Post-pruning

- Remove branches from a grown tree
- Use pruning set data (not test/training data) to decide which tree is best

POSTPRUNING EXAMPLE

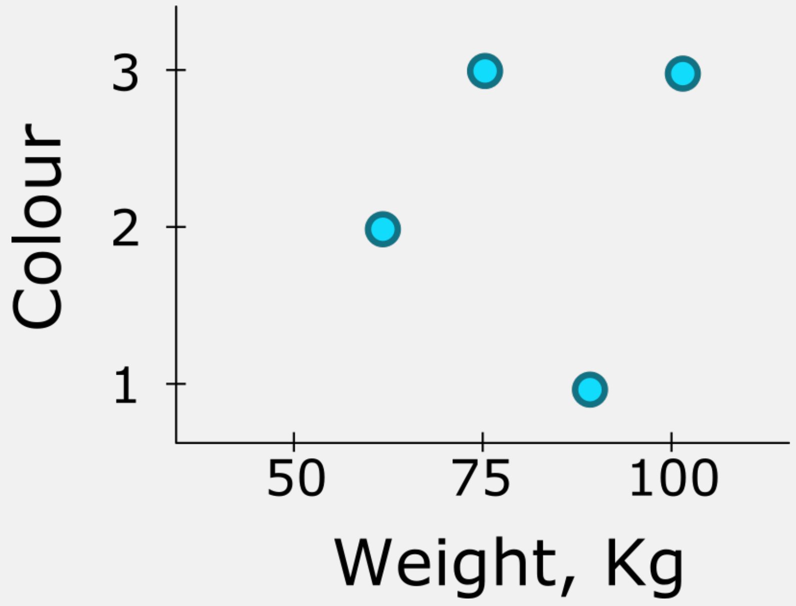
- CART uses **cost-complexity**
- Cost-complexity is a function of the number of leaves and the error rate
- Starting from the bottom:
  - Compute subtree cost-complexity at current node
  - Compute cost-complexity assuming pruning at the node
- Other approaches: minimum description length, pessimistic pruning, etc.

NOTE ON MISSING DATA

- The original version of ID3 cannot handle missing data (at least "unknown" label required)
- C4.5 can!
- Some links:
  - http://research.ijcaonline.org/volume70/number13/pxc3888063.pdf
  - https://goo.gl/1dvwwM
  - https://goo.gl/cje8zw

K-NEAREST NEIGHBOURS

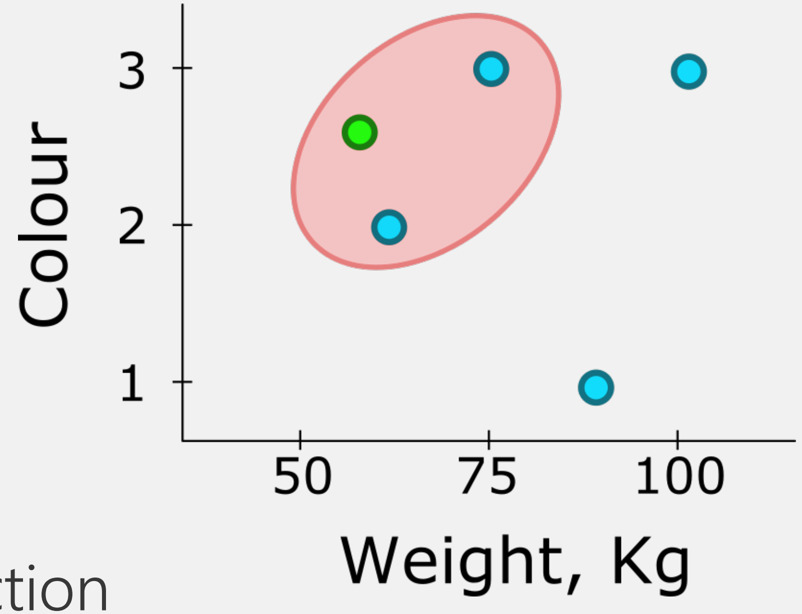
K-NEAREST NEIGHBOURS



All tuples have a position in a N-dimensional space

Each attribute is the value of a dimension!

K-NEAREST NEIGHBOURS



Idea

For unseen elements use the values of the nearest neighbours for classification or prediction

- Discrete-value classification: majority voting
- Continuous-value prediction: return average value

K-NEAREST NEIGHBOURS

Different metrics for distance. Typical: Euclidian

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}$$

Good practice: normalize

Nominal and mixed types? Remember data similarity measures!

K-NEAREST NEIGHBOURS

Distance-weighted nearest neighbor

- Weigh contribution of each neighbour according to distance:

$$w = \frac{1}{d\left(x_q, x_i\right)^2}$$

K-NEAREST NEIGHBOURS

- Robust to noisy data (averaging)
- Distance may be dominated by irrelevant attributes
- Finding best number of neighbours (k) requires experimentation (remember: validation data)

EVALUATING CLASSIFICATION MODELS

TERMINOLOGY

Positive tuple

Most interesting class

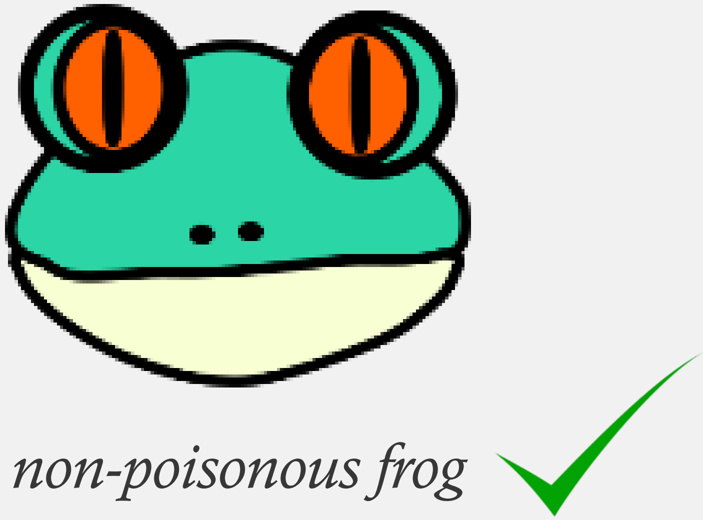
poisonous

Negative tuple

Other classes

non-poisonous

TERMINOLOGY



True positive (TP)

Positive tuple

Correct classification

True negative (TN)

Negative tuple

Correct classification

TERMINOLOGY

non-poisonous frog ✗

False negative (FN)

Positive tuple

Incorrect classification

poisonous frog ✗

False positive (FP)

Negative tuple

Incorrect classification

TERMINOLOGY

The number of true positives, true negatives, false positives and false negatives are essential values to measure performance.

CONFUSION MATRIX

		Predicted class		
		yes	no	Total
Actual class	yes	TP	FN	P
	no	FP	TN	N
	Total	P'	N'	$P + N$

EVALUATION MEASURES

- Accuracy, recognition rate $$\frac{TP + TN}{P + N}$$
- Error rate, misclassification rate $$\frac{FP + FN}{P + N}$$
- Sensitivity, true positive rate, recall $$\frac{TP}{P}$$
- Specificity, true negative rate $$\frac{TN}{N}$$

EVALUATION MEASURES

- Accuracy, recognition rate $$\frac{TP + TN}{P + N}$$
- Error rate, misclassification rate $$\frac{FP + FN}{P + N}$$
- Sensitivity, true positive rate, recall $$\frac{TP}{P}$$
- Specificity, true negative rate $$\frac{TN}{N}$$

EVALUATION MEASURES

- Precision

$$\frac{TP}{TP + FP}$$

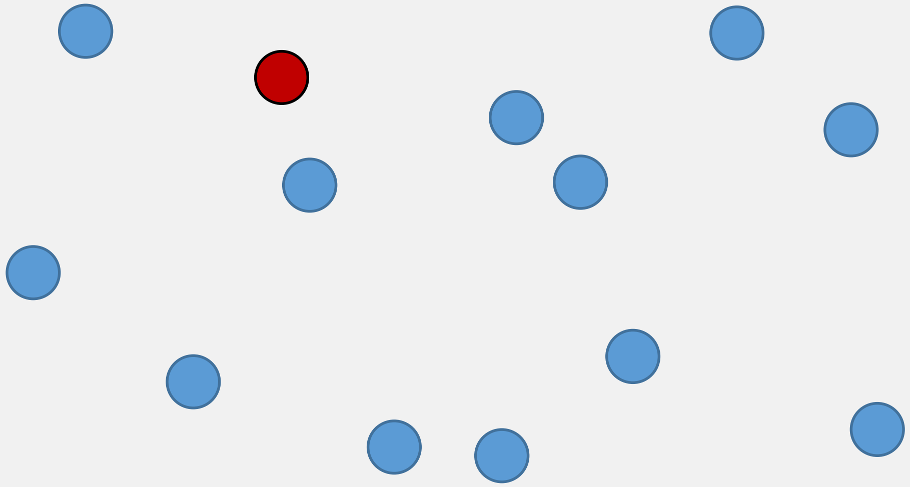
- F, F_1, F-score, harmonic mean of precision and recall

$$\frac{2 \times precision \times recall}{precision + recall}$$

- F_β, where β is a non-negative real number

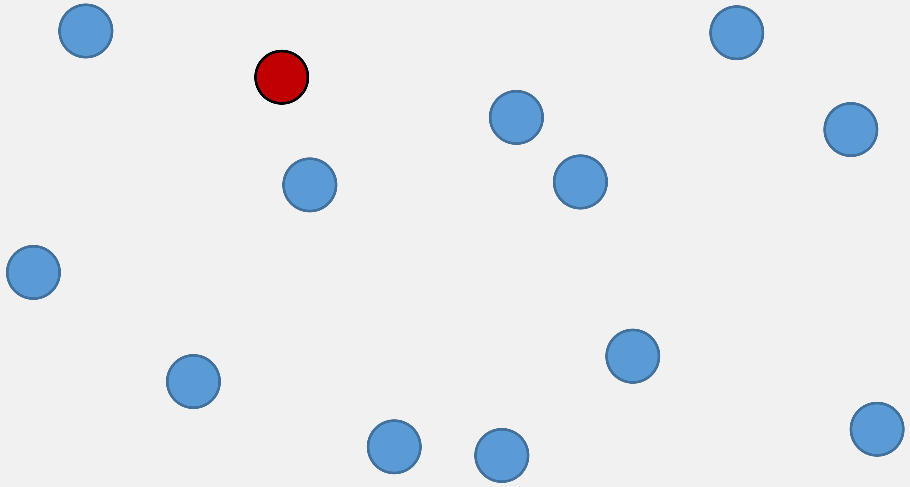
$$\frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

CLASS IMBALANCE



- Significant majority of negative class (positive class elements rare)
- Accuracy is misleading… **why**?

CLASS IMBALANCE



- Significant majority of negative class (positive class elements rare)
- Accuracy is misleading… **why**?
- Use sensitivity (proportion of positive tuples correctly classified) and specificity (proportion of negative tuples correctly classified)

EVALUATING ACCURACY

Holdout method

Training set (e.g. 2/3)—classifier construction

Test set (1/3) —accuracy estimation

Random subsampling

Repeat holdout k times, taking the average accuracy

K-FOLD CROSS-VALIDATION

- Data into k subsets $\{D_1, D_2, \ldots D_k\}$ of similar size
- At iteration "i" use D_i as test set, the rest as training set
- **Leave-one-out**

  k equals the number of tuples in the data set (usually small sets)

K-FOLD CROSS-VALIDATION

- **Stratified cross-validation**
  Each subset is stratified (class label distribution similar as in complete data set)
- Stratified 10-folds cross-validation most popular

COMPARING MODELS

- Accuracy and error measures are estimates
- What if the difference between models was just chance?
- Statistical significance test: student t-test
- Assume both models are equal: can we disprove this?

THANKS FOR LISTENING!