



# Naïve Bayes, KNN, & Other Approaches

Data Science for Mobility

- Mining of Massive Datasets Section 12.3, 12.4
- Data Mining and Analysis: Fundamental Concepts and Algorithms
  - Chapter 18 & 19
- Chapter 4 of Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques - Ian H. Witten et al.

# Naïve Bayes

## Bayes Probability

- Conditional Probability:

$$P(C|A) = \frac{P(A \wedge C)}{P(A)}$$

$$P(A|C) = \frac{P(A \wedge C)}{P(C)}$$

- Bayes theorem,

$$P(C|A) = \frac{P(A|C)P(C)}{P(A)}$$

- A priori probability of C,  $P(C)$ , is the probability of event before evidence is seen
- A posteriori probability of C,  $P(C|A)$ , is the probability of event after evidence is seen

- What's the probability of the class given an example?
- An example is represented as a tuple of attributes

$$\vec{x} = \langle x_1, \dots, x_n \rangle$$

- Given the target  $y$  (identifying the class value for the instance) we are looking for the class with the highest probability for  $X$

$$class = \arg \max_y P(y|\vec{x})$$

- Given the target  $y$  and the example  $\vec{x}$  described by  $n$  attributes, Bayes Theorem says that

$$P(y|\vec{x}) = \frac{P(\vec{x}|y)P(y)}{P(\vec{x})}$$

- Naïve Bayes classifiers assume that attributes are statistically independent
- Thus, evidence splits into parts that are independent

$$P(y|\vec{x}) = \frac{P(x_1|y) \cdots P(x_n|y)P(y)}{P(\vec{x})}$$

- **Training**

- Computes the class probability  $P(H)$  and the conditional probability  $P(e_i|H)$  for each attribute value  $e_i$  and each class value  $H$

- **Testing**

- Given an example  $E$ , computes the class as

$$\text{class} = \arg \max_y P(y|\vec{x})$$

$$= \arg \max_y \frac{P(x_1|y) \cdots P(x_n|y) P(y)}{P(\vec{x})}$$

$$= \arg \max_y P(x_1|y) \cdots P(x_n|y) P(y)$$

- Two assumptions
  - Attributes are equally important
  - Attribute are statistically independent
- Statistically independent means that knowing the value of one attribute says nothing about the value of another (if the class is known)
- Independence assumption is almost never correct! But the scheme works well in practice

# The Weather Dataset

9

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Outlook		Temperature		Humidity		Windy		Play			
	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	
Sunny	2	3	Hot	2	2	High	3	4	False	6	2
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3
Rainy	3	2	Cool	3	1						
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5
Rainy	3/9	2/5	Cool	3/9	1/5						

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

What is the assigned class?

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$\begin{aligned}
 P(yes|E) &= P(Sunny|yes)P(Cool|yes)P(High|yes)P(True|yes)P(yes) \\
 &= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} \\
 &= 0.0053
 \end{aligned}$$

$$\begin{aligned}
 P(no|E) &= P(Sunny|no)P(Cool|no)P(High|no)P(True|no)P(no) \\
 &= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} \\
 &= 0.0206
 \end{aligned}$$

- Conversion into a probability by normalization:

$$P(yes) = 0.0053 / (0.0053 + 0.0206) = 0.205$$

$$P(no) = 0.0206 / (0.0053 + 0.0206) = 0.795$$

- What if an attribute value does not occur with every class value? (for instance, “Humidity = high” for class “yes”)

- The corresponding probability will be zero,

$$P(\text{Humidity} = \text{high} | \text{yes}) = 0$$

- A posteriori probability will also be zero!  
(No matter how likely the other values are!)

$$P(\text{yes} | \langle \dots, \text{Humidity} = \text{high}, \dots \rangle) = 0$$

- The typical remedy is to add 1 to the count for every attribute value-class combination (Laplace estimator)
- The probabilities will never be zero! (also: stabilizes probability estimates)

- Sometimes, adding a constant different from one might be more appropriate; for example, we can add the same  $\mu/3$  to each count

$$\frac{2 + \mu/3}{9 + \mu}$$

Sunny

$$\frac{4 + \mu/3}{9 + \mu}$$

Overcast

$$\frac{3 + \mu/3}{9 + \mu}$$

Rainy

- Or different weights to each count

$$\frac{2 + \mu p_1}{9 + \mu}$$

$$\frac{4 + \mu p_2}{9 + \mu}$$

$$\frac{3 + \mu p_3}{9 + \mu}$$

- Weights don't need to be equal (but they must sum to 1)

- Training
  - Instance is not included in frequency count for attribute value-class combination
- Testing
  - The attribute will be omitted from calculation

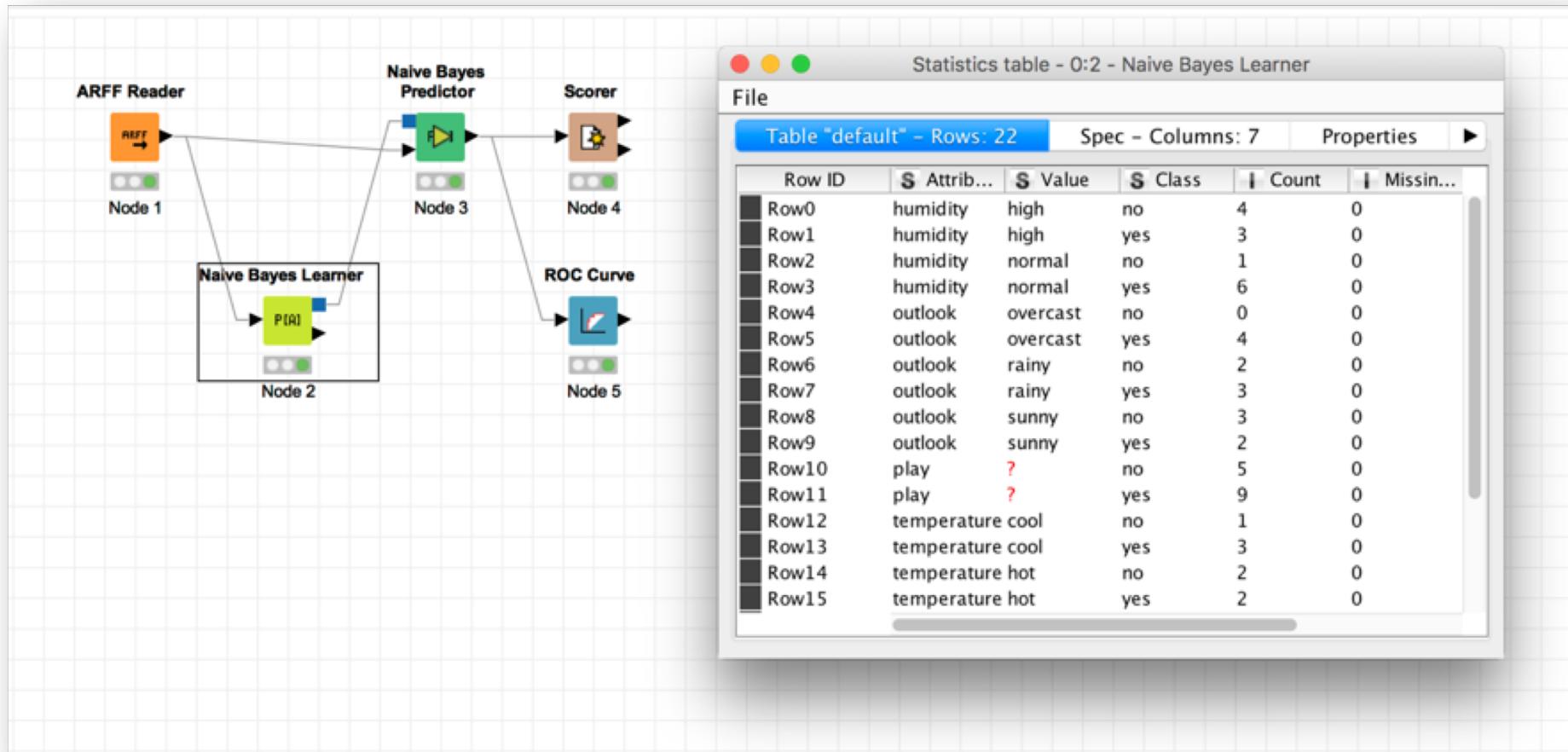
Outlook	Temperature	Humidity	Windy	Play
?	Cool	High	True	?

$$\text{Likelihood of "yes"} = 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$$

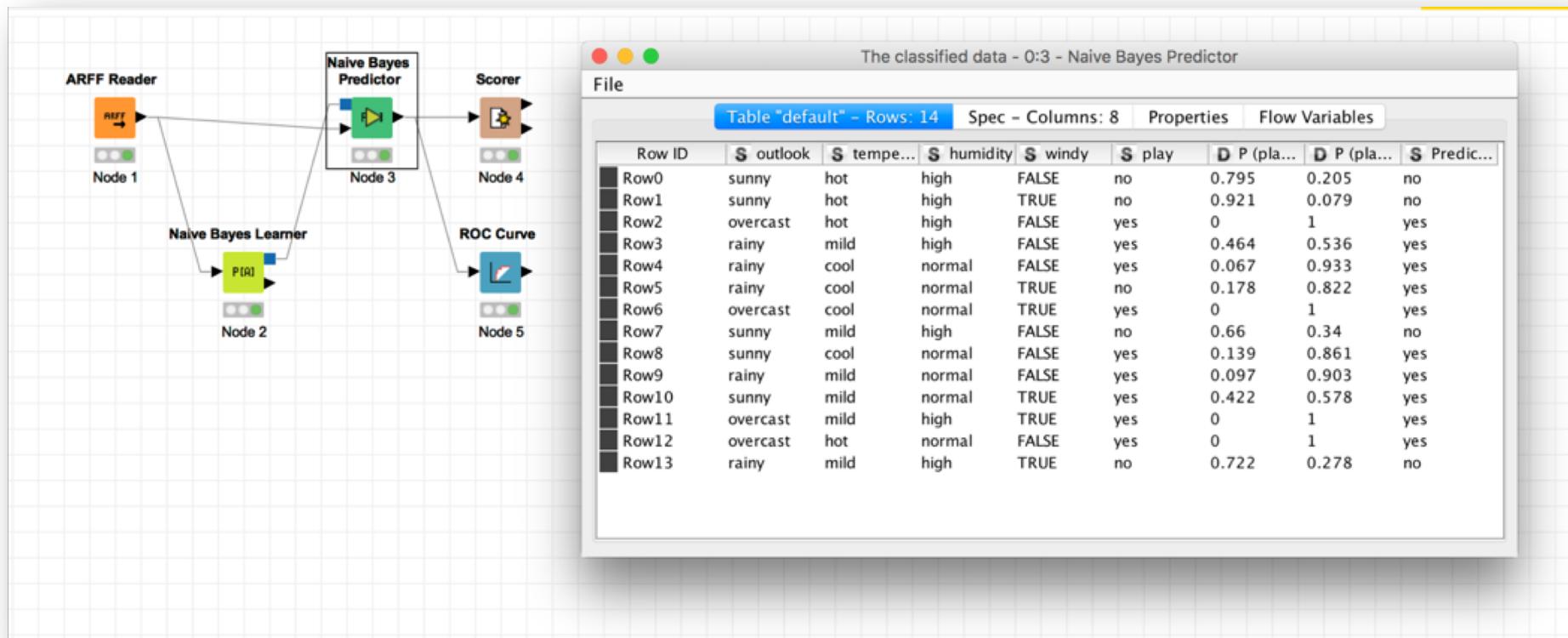
$$\text{Likelihood of "no"} = 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$$

$$P(\text{"yes"}) = 0.0238 / (0.0238 + 0.0343) = 41\%$$

$$P(\text{"no"}) = 0.0343 / (0.0238 + 0.0343) = 59\%$$



Example of Naïve Bayes using KNIME



Example of Naïve Bayes using KNIME

```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool, freeze}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
...
```

weka-3-6-10-oracle-jvm

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayesSimple

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

More options...

(Nom) play

Start Stop

Result list (right-click for options)

10:33:11 - bayes.NaiveBayesSimple

Classifier output

```
== Classifier model (full training set) ==

Naive Bayes (simple)

Class yes: P(C) = 0.625

Attribute outlook
sunny overcast rainy
0.25 0.41666667 0.33333333

Attribute temperature
hot mild cool freeze
0.23076923 0.38461538 0.30769231 0.07692308

Attribute humidity
high normal
0.36363636 0.63636364

Attribute windy
TRUE FALSE
0.36363636 0.63636364

Class no: P(C) = 0.375

Attribute outlook
sunny overcast rainy
0.5 0.125 0.375

Attribute temperature
hot mild cool freeze
0.33333333 0.33333333 0.22222222 0.11111111

Attribute humidity
high normal
0.71428571 0.28571429

Attribute windy
TRUE FALSE
0.57142857 0.42857143
```

Status

OK Log x 0

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

**Classifier**

Choose **NaiveBayesUpdateable**

**Test options**

Use training set  
 Supplied test set Set...  
 Cross-validation Folds 10  
 Percentage split % 66  
More options...

(Nom) play

Start Stop

**Result list (right-click for options)**

18:49:07 - bayes.NaiveBayes  
18:49:35 - bayes.NaiveBayes  
18:52:05 - bayes.NaiveBayesUpdateable

**Classifier output**

Naive Bayes Classifier

Attribute	Class	
	yes (0.63)	no (0.38)
outlook		
sunny	3.0	4.0
overcast	5.0	1.0
rainy	4.0	3.0
[total]	12.0	8.0
temperature		
hot	3.0	3.0
mild	5.0	3.0
cool	4.0	2.0
[total]	12.0	8.0
humidity		
high	4.0	5.0
normal	7.0	2.0
[total]	11.0	7.0
windy		
TRUE	4.0	4.0
FALSE	7.0	3.0
[total]	11.0	7.0

Time taken to build model: 0 seconds

== Evaluation on training set ==

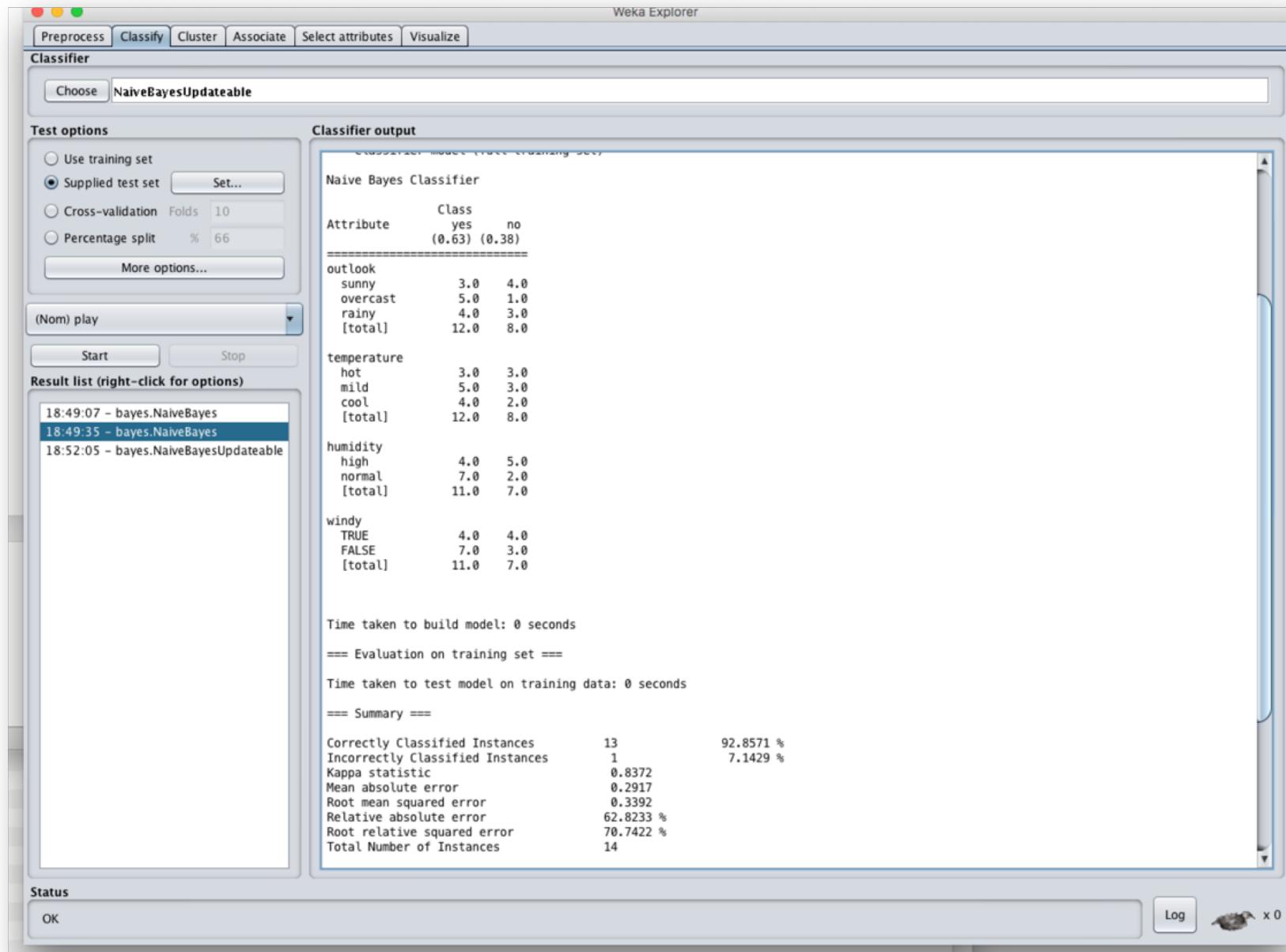
Time taken to test model on training data: 0 seconds

== Summary ==

Correctly Classified Instances	13	92.8571 %
Incorrectly Classified Instances	1	7.1429 %
Kappa statistic	0.8372	
Mean absolute error	0.2917	
Root mean squared error	0.3392	
Relative absolute error	62.8233 %	
Root relative squared error	70.7422 %	
Total Number of Instances	14	

Status

OK Log x 0



Example of Naïve Bayes with Laplace Estimator

- We assume that the attributes have a normal or Gaussian probability distribution (given the class)
- The probability density function for the normal distribution is defined by two parameters, the mean and the standard deviation
- Sample mean,

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

- Standard deviation,

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2}$$

- Then the density function  $f(x)$  is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

# Statistics for Weather data with numeric temperature

Outlook		Temperature		Humidity		Windy		Play			
	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, ...	85, ...	80, ...	95, ...					
Sunny	2/9	3/5	$\mu = 73$	$\mu = 75$	$\mu = 79$	$\mu = 86$	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	$\sigma = 6.2$	$\sigma = 7.9$	$\sigma = 10.2$	$\sigma = 9.7$	True	3/9	3/5		
Rainy	3/9	2/5									

$$f(\text{temperature} = 66 | \text{yes}) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2 \times 6.2^2}} = 0.0340$$

- A new day,

Outlook	Temperature	Humidity	Windy	Play
Sunny	66	90	true	?

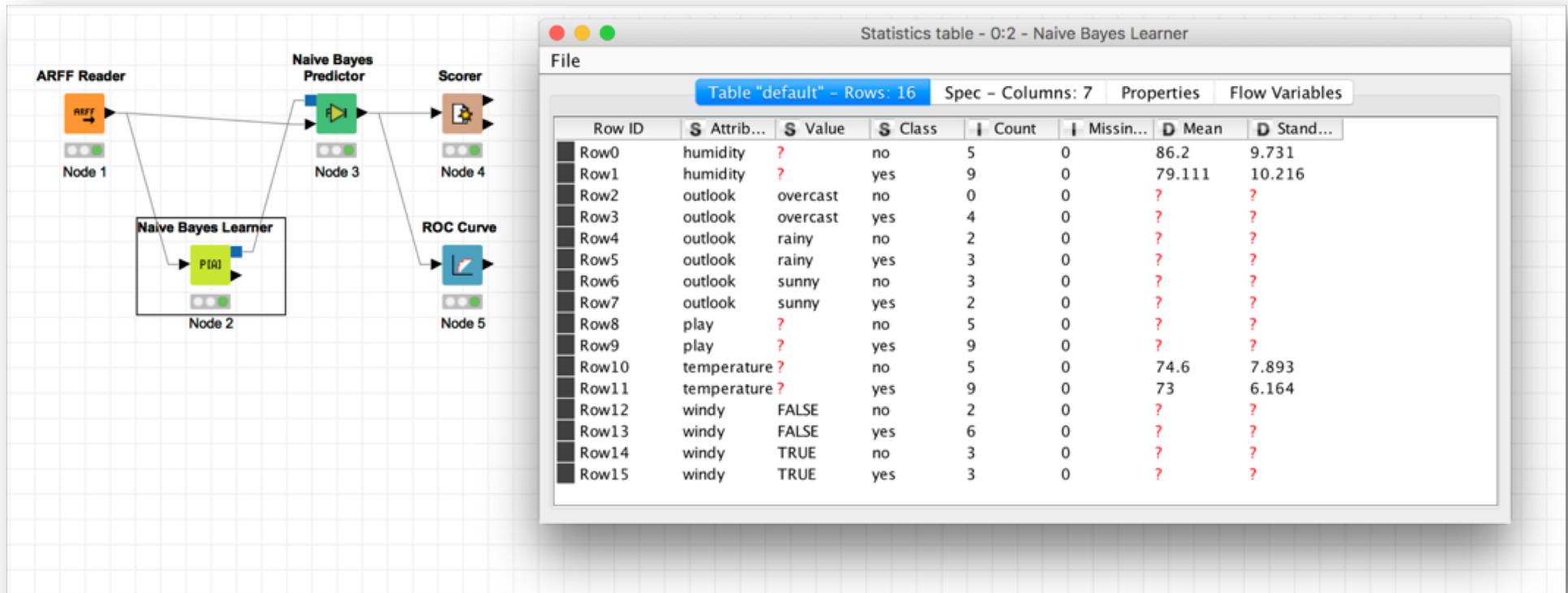
- Missing values during training are not included in calculation of mean and standard deviation

Likelihood of "yes" =  $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of "no" =  $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

$P(\text{"yes"}) = 0.000036 / (0.000036 + 0.000136) = 20.9\%$

$P(\text{"no"}) = 0.000136 / (0.000036 + 0.000136) = 79.1\%$



Example of Naïve Bayes with nominal and continuous attributes using KNIME

weka-3-6-10-oracle-jvm

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayesSimple

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

More options...

(Nom) play

Start Stop

Result list (right-click for options)

10:33:11 - bayes.NaiveBayesSimple  
10:45:46 - bayes.NaiveBayesSimple

Classifier output

```
== Classifier model (full training set) ==

Naive Bayes (simple)

Class yes: P(C) = 0.625

Attribute outlook
sunny    overcast    rainy
0.25      0.41666667   0.33333333

Attribute temperature
Mean: 73      Standard Deviation: 6.164414

Attribute humidity
Mean: 79.11111111      Standard Deviation: 10.21572861

Attribute windy
TRUE    FALSE
0.36363636     0.63636364

Class no: P(C) = 0.375

Attribute outlook
sunny    overcast    rainy
0.5       0.125      0.375

Attribute temperature
Mean: 74.6      Standard Deviation: 7.8930349

Attribute humidity
Mean: 86.2      Standard Deviation: 9.7313925

Attribute windy
TRUE    FALSE
0.57142857     0.42857143
```

Status OK Log x 0

# weka-3-6-10-oracle-jvm

## Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

### Classifier

Choose NaiveBayesSimple

#### Test options

- Use training set
  - Supplied test set [Set...](#)
  - Cross-validation Folds 10
  - Percentage split % 66
- [More options...](#)

(Nom) play

Start

Stop

#### Result list (right-click for options)

10:33:11 - bayes.NaiveBayesSimple

10:45:46 - bayes.NaiveBayesSimple

### Classifier output

== Predictions on training set ==

inst#	actual	predicted	error	probability distribution
1	2:no	2:no	0.316	*0.684
2	2:no	2:no	0.208	*0.792
3	1:yes	1:yes	*0.791	0.209
4	1:yes	1:yes	*0.545	0.455
5	1:yes	1:yes	*0.77	0.23
6	2:no	1:yes	+ *0.736	0.264
7	1:yes	1:yes	*0.938	0.062
8	2:no	2:no	0.414	*0.586
9	1:yes	1:yes	*0.809	0.191
10	1:yes	1:yes	*0.757	0.243
11	1:yes	1:yes	*0.623	0.377
12	1:yes	1:yes	*0.733	0.267
13	1:yes	1:yes	*0.915	0.085
14	2:no	2:no	0.41	*0.59

== Evaluation on training set ==

== Summary ==

Correctly Classified Instances	13	92.8571 %
Incorrectly Classified Instances	1	7.1429 %
Kappa statistic	0.8372	
Mean absolute error	0.3003	
Root mean squared error	0.3431	
Relative absolute error	64.6705 %	
Root relative squared error	71.5605 %	
Total Number of Instances	14	

== Detailed Accuracy By Class ==

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	1	0.2	0.9	1	0.947	0.933	yes
0.8	0	0	1	0.8	0.889	0.933	no
Weighted Avg.	0.929	0.129	0.936	0.929	0.926	0.933	

== Confusion Matrix ==

a	b	<-- classified as
9	0	a = yes
1	4	b = no

### Status

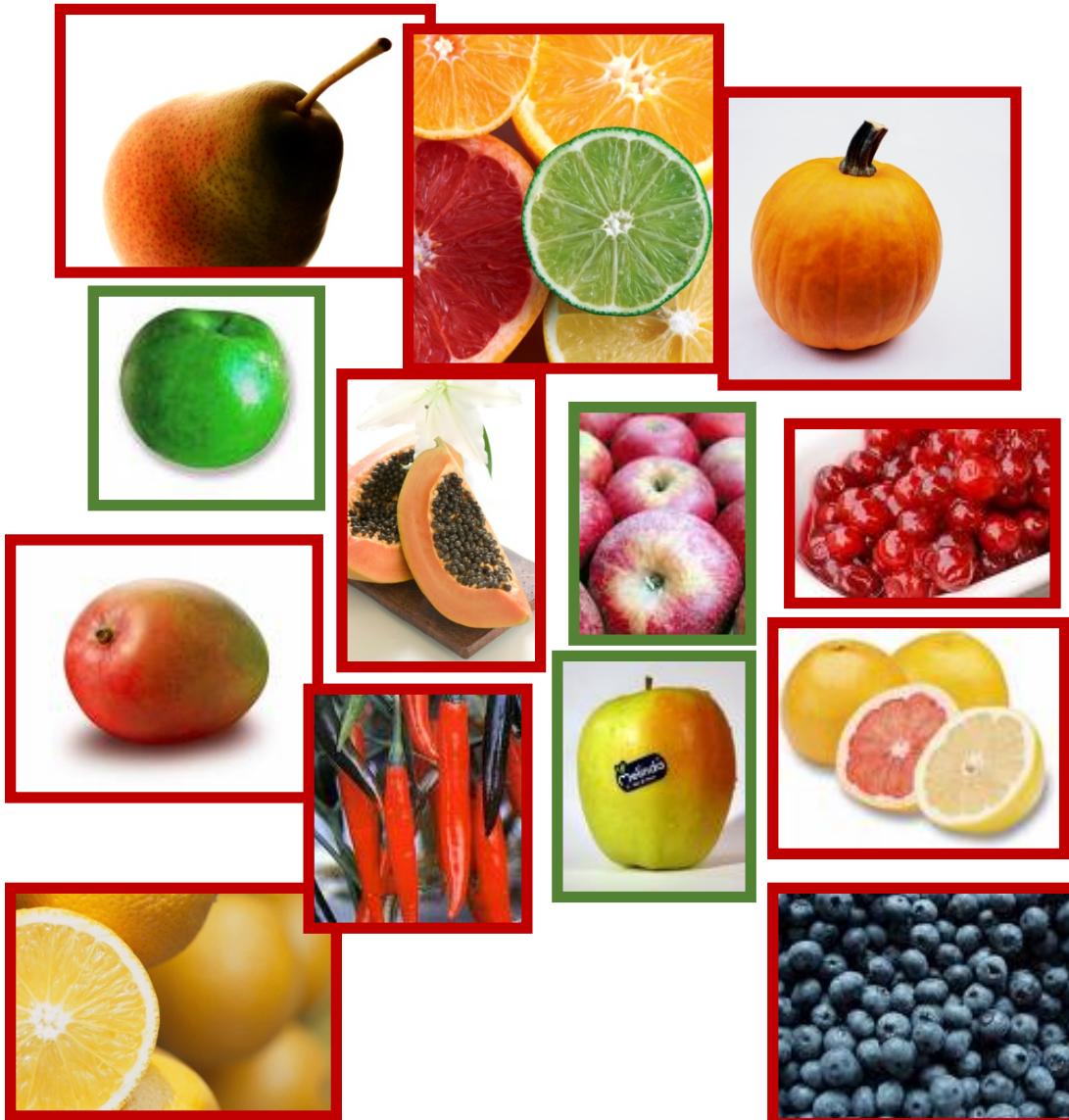
OK

Log



- Naïve Bayes works surprisingly well, even if independence assumption is clearly violated
- Why? Because classification doesn't require accurate probability estimates as long as maximum probability is assigned to correct class
- However, adding too many redundant attributes will cause problems (e.g. identical attributes)
- Also, many numeric attributes are not normally distributed

# Nearest Neighbors (Instance-Based Learning)

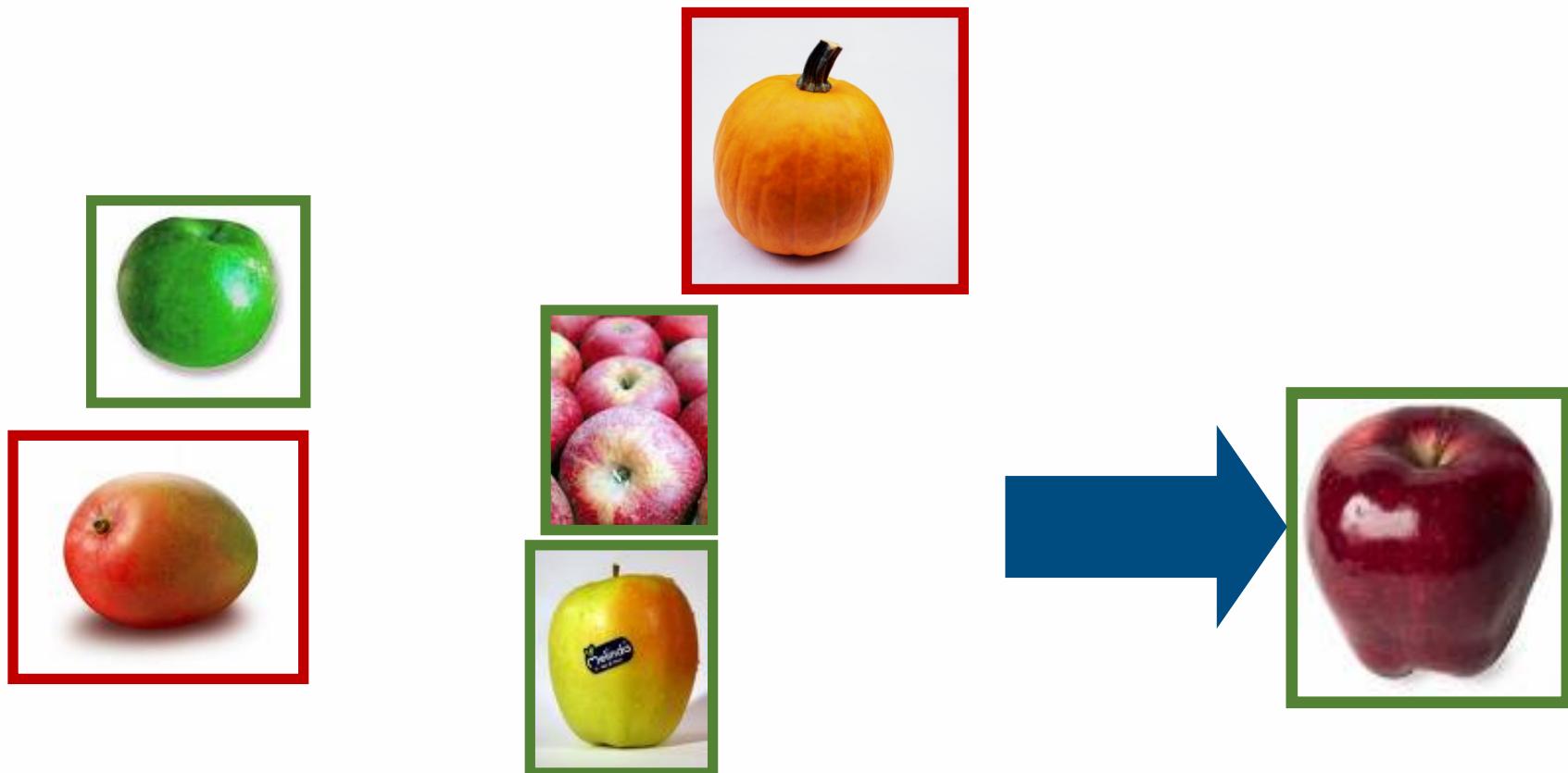


is this an apple?



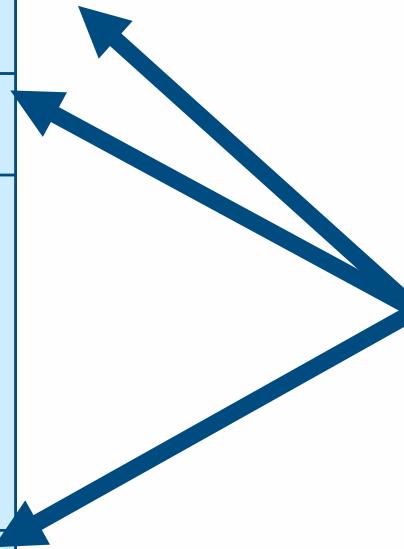
- To decide the label for an unseen example, consider the  $k$  examples from the training data that are more similar to the unknown one
- Classify the unknown example using the most frequent class

- If  $k=5$ , these 5 fruits are the most similar ones to the unclassified example
- Since, the majority of the fruits are labeled as positive examples, we decide that the unknown fruit is an apple



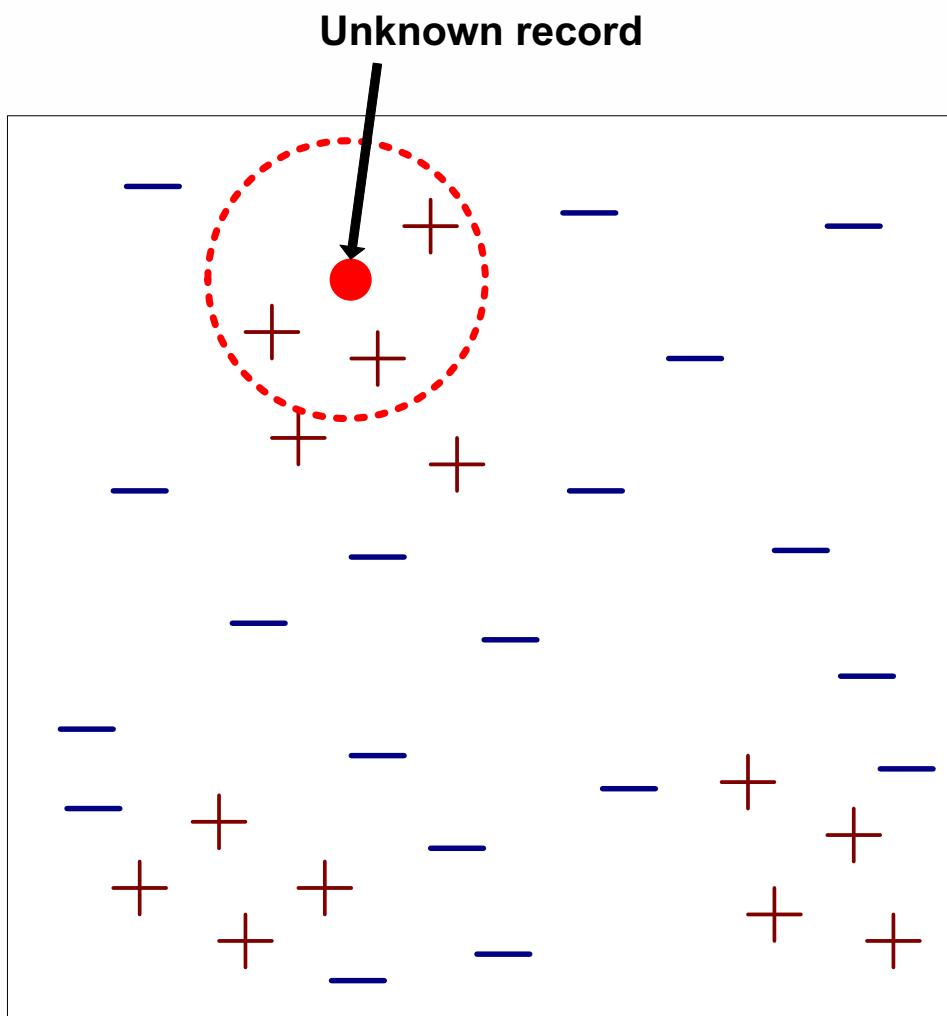
- Store the training records only, no model is computed
- Use the training records to predict an unknown class label

Att <sub>1</sub>	...	Att <sub>n</sub>	Class

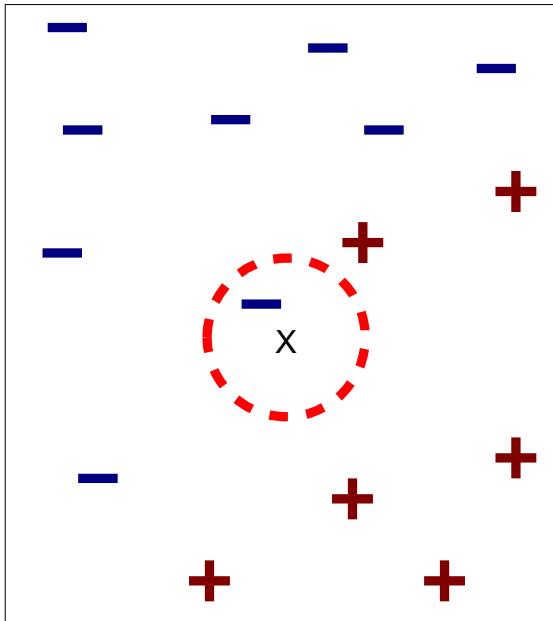


Att <sub>1</sub>	...	Att <sub>n</sub>

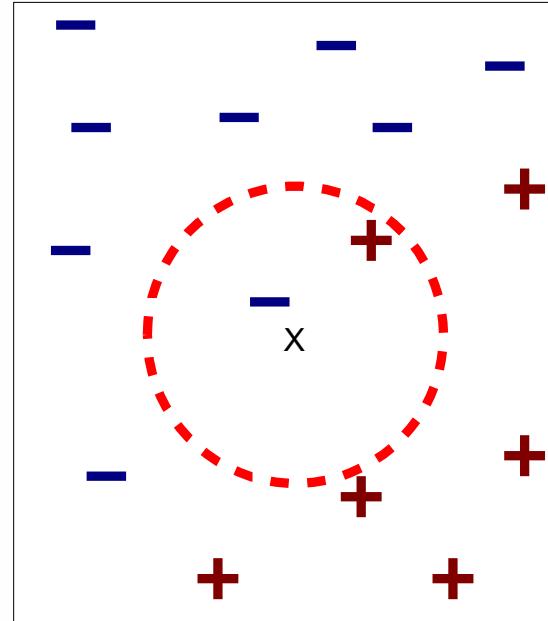
- They are the simplest form of learning
- The training dataset is searched for the instances that are more similar to the unlabeled instance
- The training dataset is the model itself, it is the knowledge
- The similarity function defines what's "learned"
- They implement a "lazy evaluation" (or lazy learning) scheme, nothing happens until a new unlabeled instance must be classified
- Known methods include "Rote Learning", "Case Base Reasoning", "k-Nearest Neighbors"



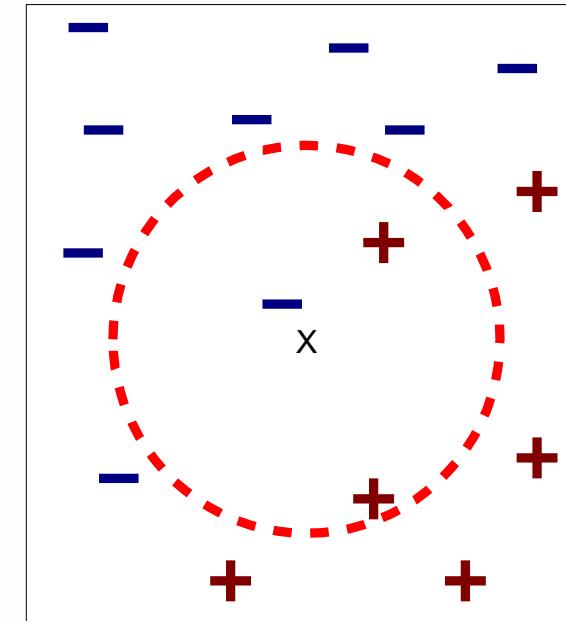
- Three elements
  - The training dataset
  - Similarity function (or distance metric)
  - The value of  $k$ , the number of nearest neighbors to retrieve
- Classification
  - Compute distance to other training records
  - Identify the  $k$  nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

K-nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distance to  $x$

# How many neighbors?

If  $k$  is too small, classification might be sensitive to noise points

If  $k$  is too large, neighborhood may include quite dissimilar examples

**weka-3-6-10-oracle-jvm**

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

**Classifier**

Choose **IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\"weka.core.EuclideanDistance -R first-last\\"**

**Test options**

Use training set  
 Supplied test set Set...  
 Cross-validation Folds 10  
 Percentage split % 66  
 More options...

(Nom) play

Start Stop

**Result list (right-click for options)**

- 09:19:25 - bayes.NaiveBayes
- 09:25:55 - lazy.IBk**

**Classifier output**

```

windy
play
Test mode:evaluate on training data

== Classifier model (full training set) ==

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

== Evaluation on training set ==
== Summary ==

Correctly Classified Instances      14          100      %
Incorrectly Classified Instances    0           0        %
Kappa statistic                      1
Mean absolute error                 0.0625
Root mean squared error              0.0625
Relative absolute error              13.4615 %
Root relative squared error         13.0347 %
Total Number of Instances           14

== Detailed Accuracy By Class ==

      TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
      1          0          1          1          1          1          yes
      1          0          1          1          1          1          no
Weighted Avg.   1          0          1          1          1          1

== Confusion Matrix ==

a b  <-- classified as
9 0 | a = yes
0 5 | b = no

```

**Status**

OK

Log x 0

**weka-3-6-10-oracle-jvm**

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

**Classifier**

Choose **IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\"weka.core.EuclideanDistance -R first-last\\""**

**Test options**

Use training set  
 Supplied test set Set...  
 Cross-validation Folds 10  
 Percentage split % 66  
 More options...

(Nom) play

Start Stop

**Result list (right-click for options)**

- 09:19:25 - bayes.NaiveBayes
- 09:25:55 - lazy.IBk
- 09:28:25 - lazy.IBk**

**Classifier output**

```

windy
play
Test mode: evaluate on training data

== Classifier model (full training set) ==

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

== Evaluation on training set ==
== Summary ==

Correctly Classified Instances      12          85.7143 %
Incorrectly Classified Instances   2           14.2857 %
Kappa statistic                   0.6585
Mean absolute error               0.352
Root mean squared error          0.3936
Relative absolute error          75.8217 %
Root relative squared error     82.0865 %
Total Number of Instances        14

== Detailed Accuracy By Class ==

      TP Rate    FP Rate    Precision    Recall    F-Measure    ROC Area    Class
      1          0.4        0.818       1          0.9        0.844      yes
      0.6        0          1          0.6        0.75       0.844      no
Weighted Avg.    0.857    0.257      0.883      0.857      0.846      0.844

== Confusion Matrix ==

a b    <-- classified as
9 0 | a = yes
2 3 | b = no

```

**Status**

OK

Log x 0

weka-3-6-10-oracle-jvm

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose **IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\"weka.core.EuclideanDistance -R first-last\\""**

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66

More options...

(Nom) play

Start Stop

Result list (right-click for options)

- 09:19:25 - bayes.NaiveBayes
- 09:25:55 - lazy.IBk
- 09:28:25 - lazy.IBk
- 09:30:29 - lazy.IBk**

Classifier output

windy  
play

Test mode:10-fold cross-validation

== Classifier model (full training set) ==

IB1 instance-based classifier  
using 1 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

== Stratified cross-validation ==

== Summary ==

Correctly Classified Instances	8	57.1429 %
Incorrectly Classified Instances	6	42.8571 %
Kappa statistic	0.0667	
Mean absolute error	0.4911	
Root mean squared error	0.5985	
Relative absolute error	103.137 %	
Root relative squared error	121.313 %	
Total Number of Instances	14	

== Detailed Accuracy By Class ==

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.667	0.6	0.667	0.667	0.667	0.5	yes
0.4	0.333	0.4	0.4	0.4	0.456	no
Weighted Avg.	0.571	0.505	0.571	0.571	0.484	

== Confusion Matrix ==

a b <- classified as  
6 3 | a = yes  
3 2 | b = no

Status OK Log x 0

**weka-3-6-10-oracle-jvm**

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

**Classifier**

Choose **IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \\"weka.core.EuclideanDistance -R first-last\\""**

**Test options**

- Use training set
- Supplied test set
- Cross-validation Folds
- Percentage split %

(Nom) play

**Result list (right-click for options)**

- 09:19:25 - bayes.NaiveBayes
- 09:25:55 - lazy.IBk
- 09:28:25 - lazy.IBk
- 09:30:29 - lazy.IBk
- 09:31:56 - lazy.IBk**

**Classifier output**

```

temperature
humidity
windy
play

Test mode:10-fold cross-validation

== Classifier model (full training set) ==

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification

Time taken to build model: 0 seconds

== Stratified cross-validation ==
== Summary ==

Correctly Classified Instances      9          64.2857 %
Incorrectly Classified Instances    5          35.7143 %
Kappa statistic                      0.1026
Mean absolute error                  0.4414
Root mean squared error              0.4747
Relative absolute error              92.699 %
Root relative squared error         96.2242 %
Total Number of Instances           14

== Detailed Accuracy By Class ==

      TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
      0.889     0.8       0.667     0.889     0.762     0.689   yes
      0.2       0.111     0.5       0.2       0.286     0.644   no
Weighted Avg.   0.643     0.554     0.607     0.643     0.592     0.673

== Confusion Matrix ==

a b  <-- classified as
8 1 | a = yes
4 1 | b = no

```

Status OK   x 0

# What similarity measures?

Distance and similarity identify orthogonal yet related concepts

When distance is larger  
similarity is smaller

Viceversa, when similarity is larger,  
distance is smaller

- Euclidian distance is the typical function used to compute the similarity between two examples

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- To determine the class from nearest neighbor list
  - Take the majority vote of class labels among the k neighbors
  - Or weight the vote according to distance (e.g.,  $w = 1/d^2$ )
- Another popular metric is city-block (Manhattan) metric, distance is the sum of absolute differences

- $L_r$ -norm

$$d([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \left( \sum_{i=1}^n |x_i - y_i|^r \right)^{1/r}$$

- Euclidean distance ( $r=2$ )

$$d([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Manhattan distance ( $r=1$ )

$$d([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \sum_{i=1}^n |x_i - y_i|$$

- $L_\infty$ -norm

$$d([x_1, x_2, \dots, x_n], [y_1, y_2, \dots, y_n]) = \max_{i=1}^n |x_i - y_i|$$

- The cosine distance between  $x, y$  is the angle that the vectors to those points make

$$d(\vec{x}, \vec{y}) = \arccos \frac{\sum_1^n x_i y_i}{\sqrt{\sum_i^n x_i^2} \sqrt{\sum_i^n y_i^2}}$$

- This angle will be in the range 0 to 180 degrees, regardless of how many dimensions the space has.
- Example: given  $x = (1, 2, -1)$  and  $y = (2, 1, 1)$  the angle between the two vectors is 60

- Jaccard distance is a measure of how dissimilar two sets are.  
Examples are represented by binary variables and are viewed as representations of set
- Jaccard distance is defined as

$$d(\vec{x}, \vec{y}) = 1 - J(\vec{x}, \vec{y})$$

- $J$  is the Jaccard similarity which is computed as the number of

$$J(\vec{x}, \vec{y}) = \frac{|\vec{x} \cap \vec{y}|}{|\vec{x} \cup \vec{y}|}$$

- Which can also be interpreted as the percentage of identical attributes

- Hamming distance between two vectors is the number of components in which they differ
- Or equivalently, given the number of variables  $n$ , and the number  $m$  of matching components, we define

$$d(\vec{x}, \vec{y}) = \frac{n - m}{n}$$

- Example: the Hamming distance between the vectors 10101 and 11110 is 3/5.

- Different attributes are measured on different scales, thus we often need to normalize them, using for instance range normalization o standard score normalization

$$x'_i = \frac{x_i - \min_i x_i}{\max_i x_i - \min_i x_i}$$

$$x'_i = \frac{x_i - \mu}{\sigma}$$

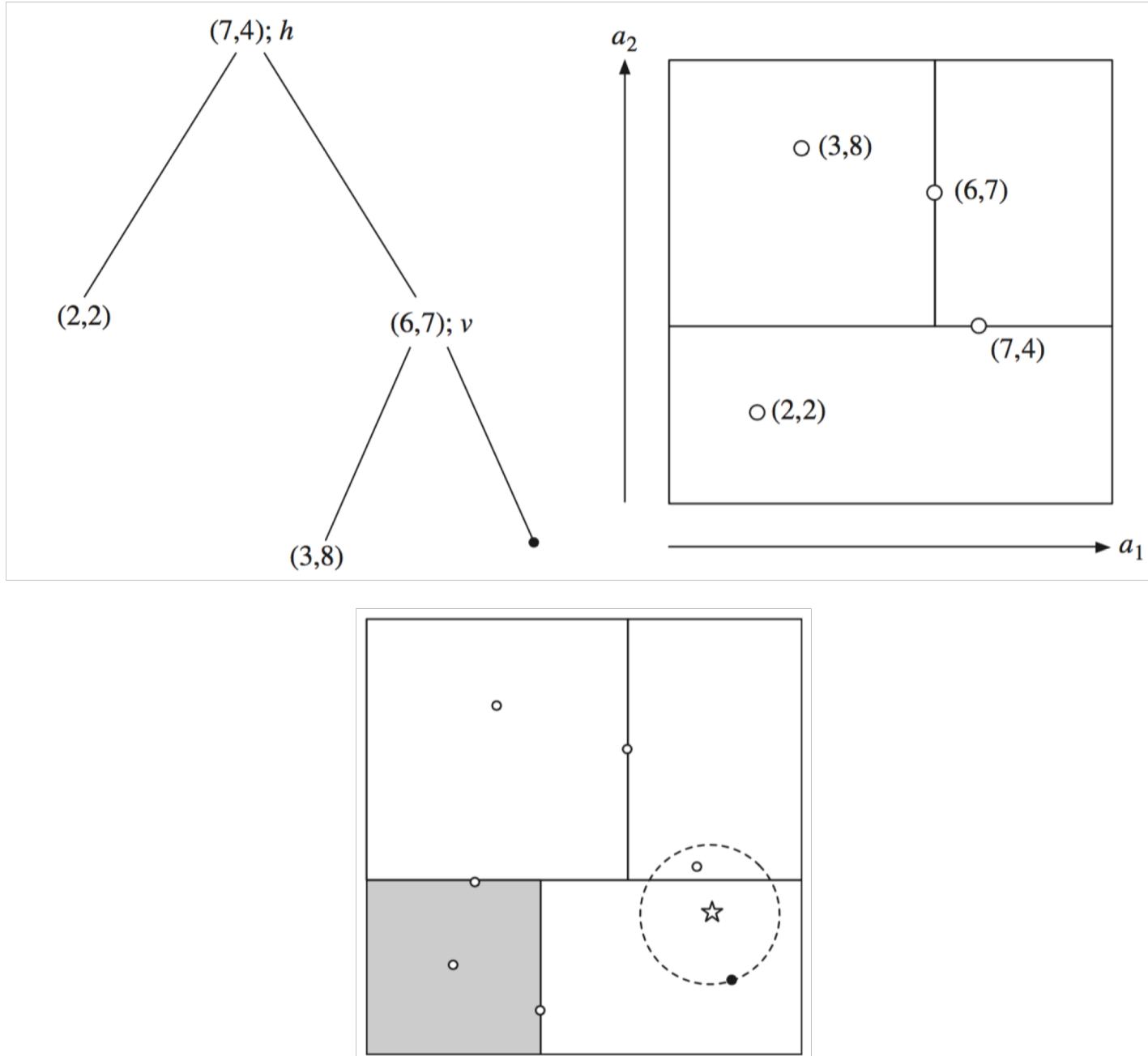
- For nominal attributes, the distance either 0 (the value is the same) or 1 (the value is different)
- Missing values are usually assumed to be maximally distant (given normalized attributes)

- Basic Approach
  - Linear scan of the data
  - Classification time is proportional to the product of the number of instances in training and test sets (and the
- Nearest-neighbor search can be speeded up by using
  - KD-Trees
  - Ball-Trees
  - ...

# KD-Trees

Split the space hierarchically using  
a tree generated from the data

To find the neighbor of a specific example,  
navigate the tree using the example



Example of KD-tree. The tree works like a decision tree and splits the space in a hierarchy. When the class of one example is needed, the tree is navigated until a leaf is reached and only the nearby areas are actually checked by backtracking on the tree.

- Complexity depends on depth of the tree, given by the logarithm of number of nodes for a balanced tree
- Amount of backtracking required depends on quality of tree (“square” vs. “skinny” nodes)
- How to build a good tree?
  - Need to find good split point and split direction
  - Possible split direction: direction with greatest variance
  - Possible split point: median value along that direction
- Using value closest to mean (rather than median) can be better if data is skewed
- Can apply this split selection strategy recursively just like in the case of decision tree learning

# Example using the loan dataset

Evaluation using 10-fold crossvalidation

kd-tree

t=11.163 Accuracy=0.806 Std=0.00

Brute force

t=567.895 Accuracy=0.806 Std=0.001

Run the Python notebooks  
for k-Nearest-Neighbor

- The regression methods we previously discussed are parametric
  - They assume an approximation function parametrized by a weight vector
  - The regression algorithm search for the best weight vector
  - E.g., linear regression assumes that the target is computed as a linear combination of the weight vector  $w$  and the feature vector  $h(x)$
- Nearest Neighbor regression fits each point locally
- Basic Algorithm
  - Given a dataset  $(x_1, y_1) \dots (x_N, y_N)$
  - Given a query point  $x_q$
  - The value  $y_q$  associated to  $x_q$  is computed as a local interpolation of the targets associated to neighbor points
- Prediction can use the plain average of the  $k$  nearest targets or a weighted average of the same targets
- Prediction can use kernel functions that take the distance as input and return a weight (e.g., a Gaussian function of the distance)

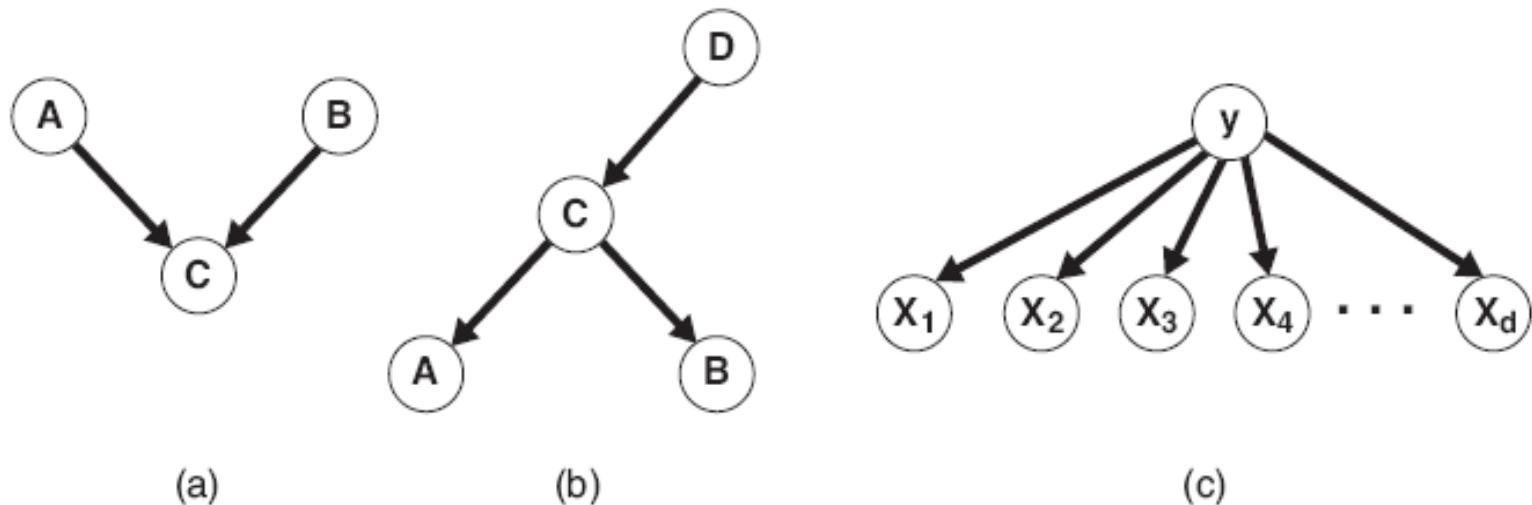
- K-Nearest Neighbor is often very accurate but slow, since simple version scans entire training data to derive a prediction
- Assumes all attributes are equally important, thus may need attribute selection or weights
- Possible remedies against noisy instances:
  - Take a majority vote over the  $k$  nearest neighbors
  - Remove noisy instances from dataset (difficult!)
- Statisticians have used k-NN since early 1950s,  
If  $n \rightarrow \infty$  and  $k/n \rightarrow 0$ , error approaches minimum
- Data Structures
  - kD-trees can become inefficient when the number of attributes is too large
  - Ball trees may help

# Bayesian Belief Networks

- The conditional independence assumption,
  - makes computation possible
  - yields optimal classifiers when satisfied
  - but is seldom satisfied in practice, as attributes (variables) are often correlated
- Bayesian Belief Networks (BBN) allows us to specify which pair of attributes are conditionally independent
- They provide a graphical representation of probabilistic relationships among a set of random variables

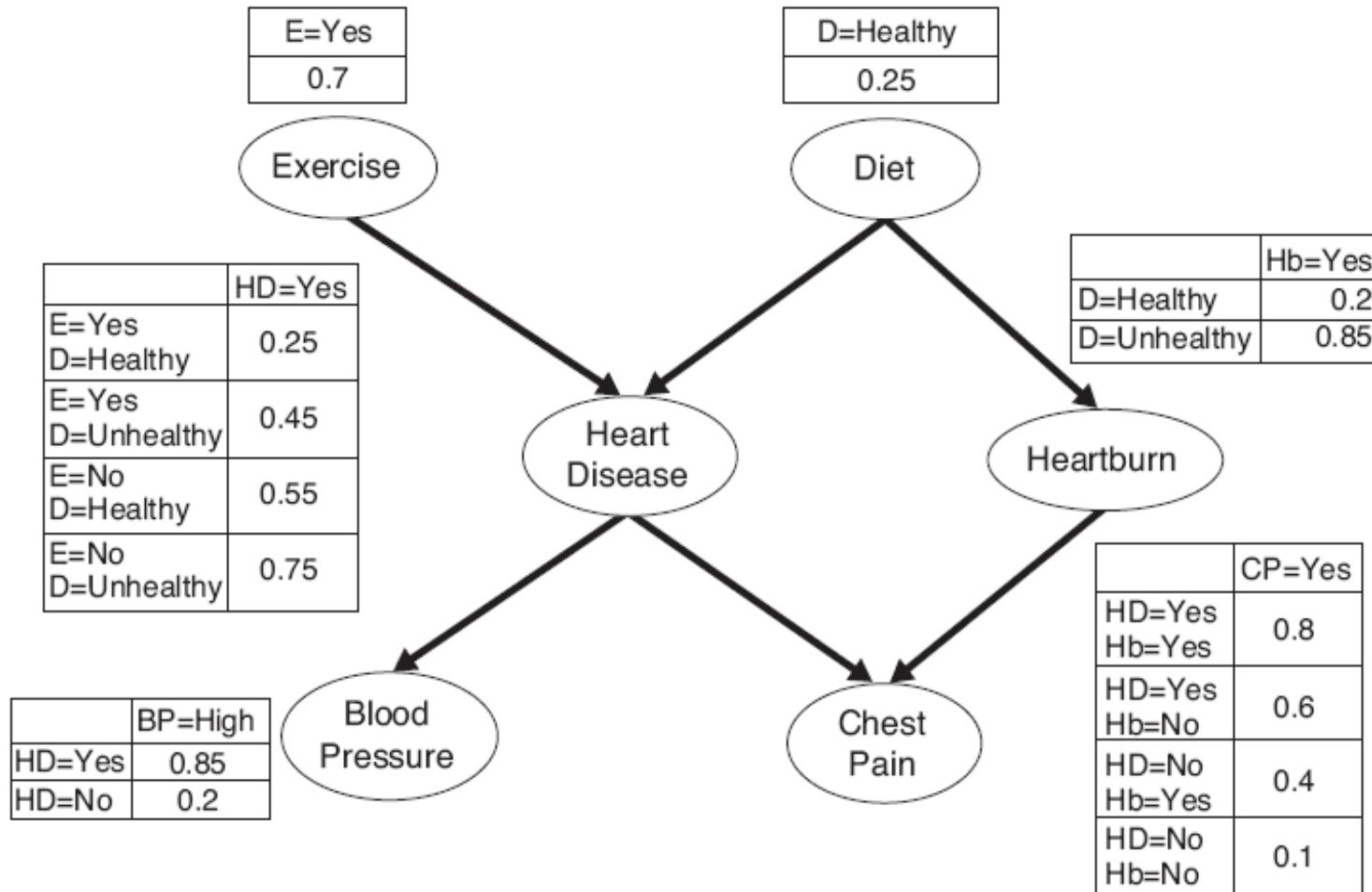
- Describe the probability distribution governing a set of variables by specifying
  - Conditional independence assumptions that apply on subsets of the variables
  - A set of conditional probabilities
- Two key elements
  - A direct acyclic graph, encoding the dependence relationships among variables
  - A probability table associating each node to its immediate parents node

- Variable A and B are independent, but each one of them has an influence on the variable C
- A is conditionally independent of both B and D, given C
- The configuration of the typical naïve Bayes classifier



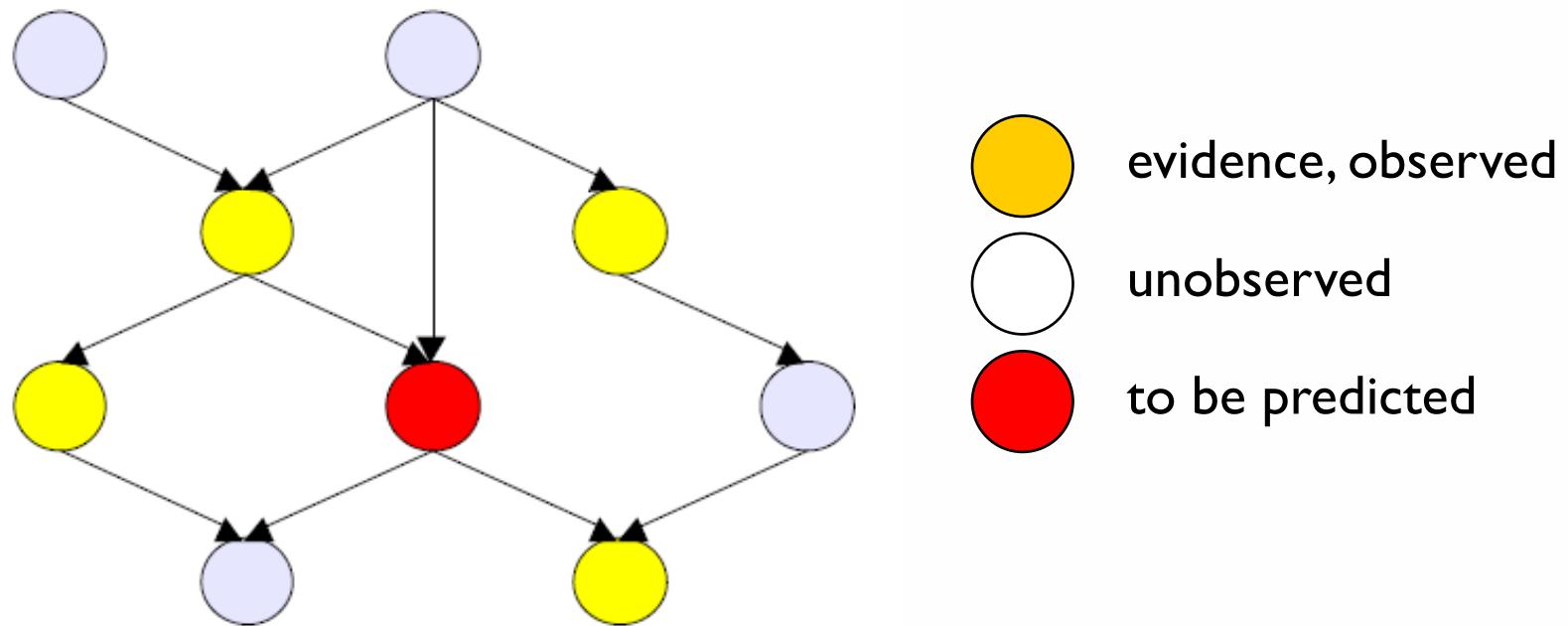
# An Example of Bayesian Belief Network

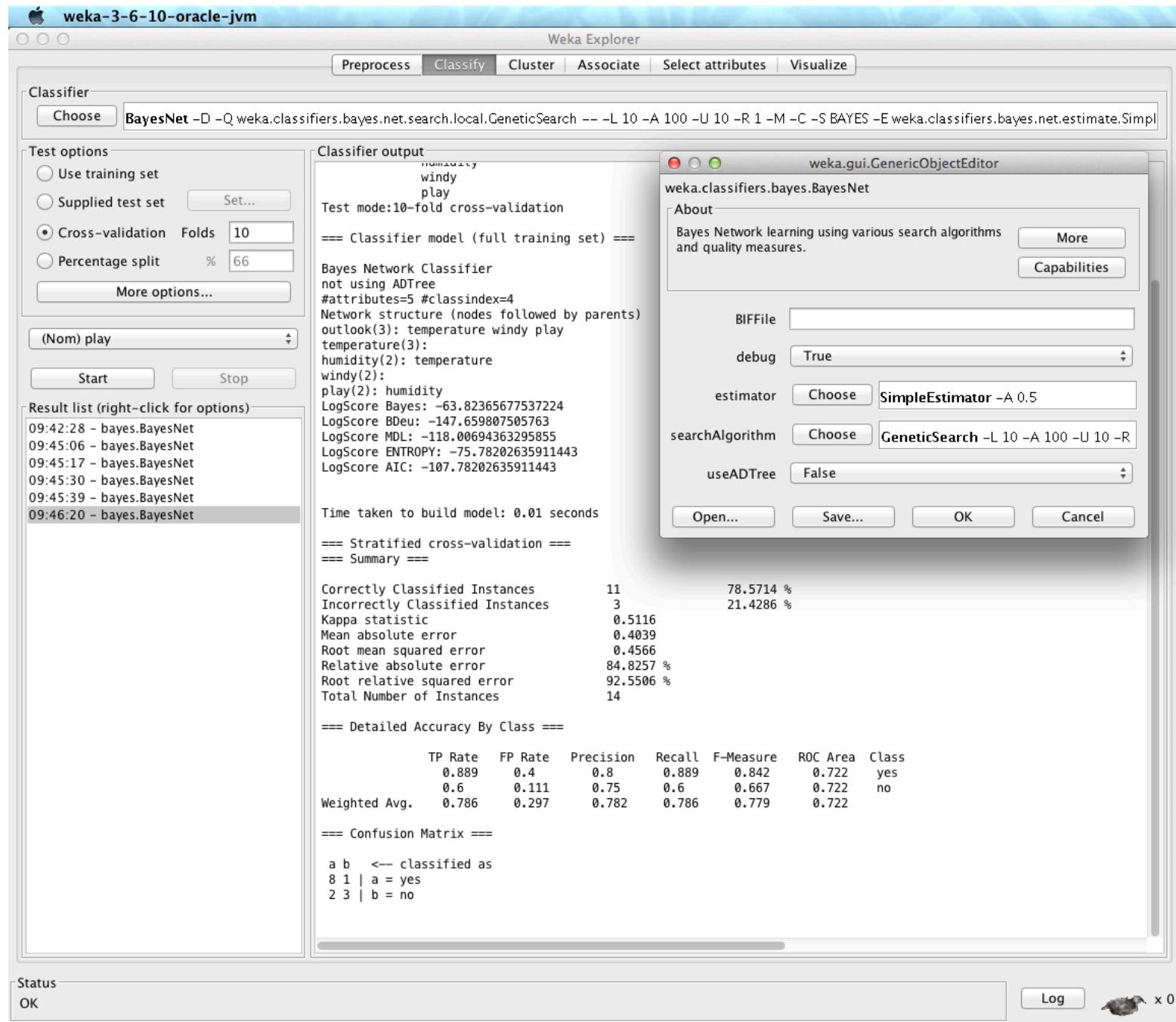
62



- The network topology imposes conditions regarding the variable conditional independence
- Each node is associated with a probability table
  - If a node  $X$  does not have any parents, then the table contains only the prior probability  $P(X)$
  - If a node  $X$  has only one parent  $Y$ , then the table contains only the conditional probability  $P(X|Y)$
  - If a node  $X$  has multiple parents,  $Y_1, \dots, Y_k$  the table contains the conditional probability  $P(X|Y_1 \dots Y_k)$

- In general the inference is NP-complete but there are approximating methods, e.g. Monte-Carlo





**weka-3-6-10-oracle-jvm**

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

**Classifier**

Choose BayesNet -D -Q weka.classifiers.bayes.net.search.local.GeneticSearch -- -L 10 -A 100 -U 10 -R 1 -M -C -S BAYES -E weka.classifiers.bayes.net.estimate.Simpl

**Test options**

- Use training set
- Supplied test set Set...
- Cross-validation Folds 10
- Percentage split % 66

More options...

(Nom) play

Start Stop

**Result list (right-click for options)**

```
09:42:28 - bayes.BayesNet
09:45:06 - bayes.BayesNet
09:45:17 - bayes.BayesNet
09:45:30 - bayes.BayesNet
09:45:39 - bayes.BayesNet
```

**Classifier output**

```
humidity
windy
play
Test mode:10-fold cross-validation
== Classifier model (full training set) ==
Bayes Network Classifier
not using ADTree
#attributes=5 #classindex=4
Network structure (nodes followed by parents)
outlook(3): temperature windy play
temperature(3):
humidity(2): temperature
windy(2):
play(2): humidity
LogScore Bayes: -63.82365677537224
LogScore BDeu: -147.659807505763
LogScore MDL: -118.00694363295855
LogScore ENTROPY: -75.78202635911443
LogScore AIC: -107.78202635911443

Time taken to build model: 0.01 seconds

== Stratified cross-validation ==
== Summary ==

Correctly Classified Instances      11          78.5714 %
Incorrectly Classified Instances    3           21.4286 %
Kappa statistic                     0.5116
Mean absolute error                 0.4039
Root mean squared error             0.4566
Relative absolute error              84.8257 %
Root relative squared error        92.5506 %
Total Number of Instances          14

== Detailed Accuracy By Class ==

      TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
      0.889     0.4       0.8       0.889     0.842      0.722    yes
      0.6       0.111     0.75      0.6       0.667      0.722    no
Weighted Avg.   0.786     0.297     0.782      0.786     0.779      0.722

== Confusion Matrix ==

a b  <-- classified as
8 1 | a = yes
2 3 | b = no
```

**Weka Classifier Graph Visualizer: 09:45:39 - bayes.BayesNet**

```
graph TD
    temperature((temperature)) --> humidity((humidity))
    humidity --> play((play))
    windy((windy)) --> outlook((outlook))
    play --> outlook
```

Status OK Log x 0

- Bayesian belief network allows a subset of the variables conditionally independent
- A graphical model of causal relationships
- Several cases of learning Bayesian belief networks
  - Given both network structure and all the variables is easy
  - Given network structure but only some variables
  - When the network structure is not known in advance

- They provide an approach for capturing prior knowledge of a particular domain using a graphical model
- Building the network is time consuming, but adding variables to a network is straightforward
- They can encode causal dependencies among variables
- They are well suited to dealing with incomplete data
- They are robust to overfitting