UNIVERSITÄT MANNHEIM

Data Mining I
Classification, Part 1

Heiko Paulheim

Outline

1. What is Classification?
2. k Nearest Neighbors and Nearest Centroids
3. Naïve Bayes
4. Evaluating Classification
5. Decision Trees
6. The Overfitting Problem
7. Other Classification Approaches
8. Hyperparameter Tuning

A Couple of Questions

- What is this?
- Why do you know?
- How have you come to that knowledge?

Introductory Example

- Learning a new concept, e.g., "Tree"

"tree"

"tree"

"tree"

"not a tree"

"not a tree"

"not a tree"

Introductory Example

- Example: learning a new concept, e.g., "Tree"
  - we look at (positive and negative) examples
  - ...and derive a *model*
    - e.g., "Trees are big, green plants"





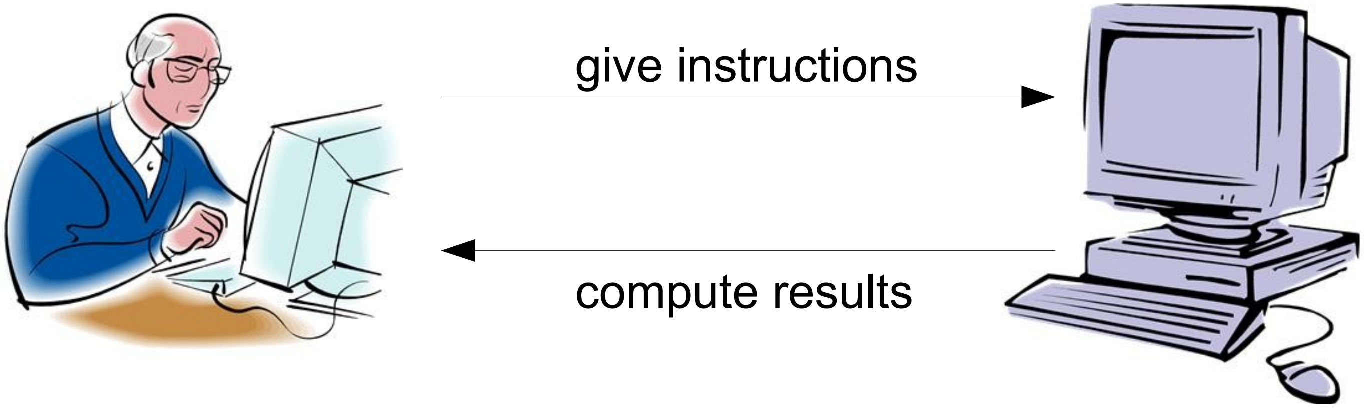

- Goal: Classification of new instances

"tree?"

Warning:
Models are only approximating examples!
Not guaranteed to be correct or complete!

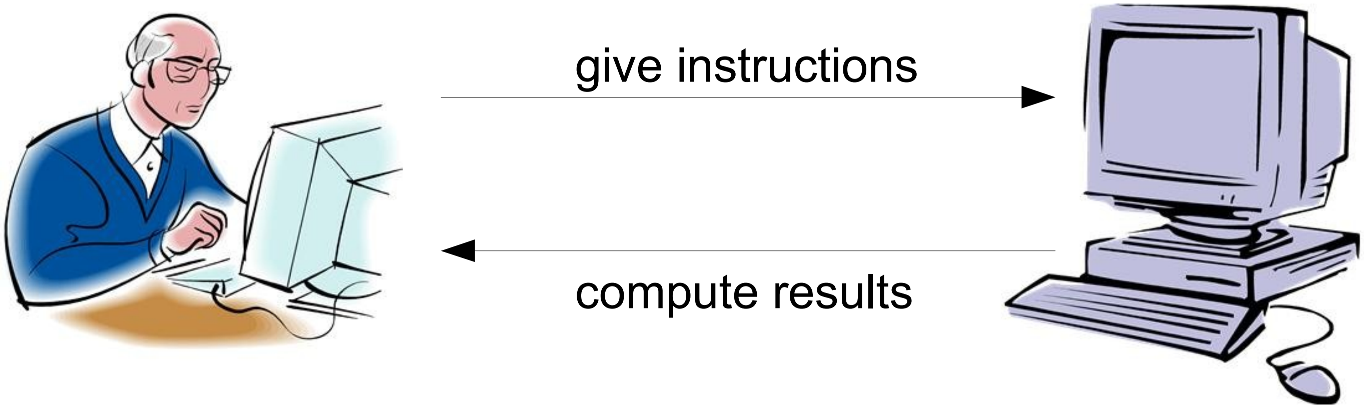
What is Classification?

- Classic programming:
  - `if more than 10 orders/year and more than $100k spent`
    `set customer.isPremiumCustomer = true`
- The prevalent style of programming computers
  - works well as long as we know the rules
  - e.g.: what makes a customer a premium customer?



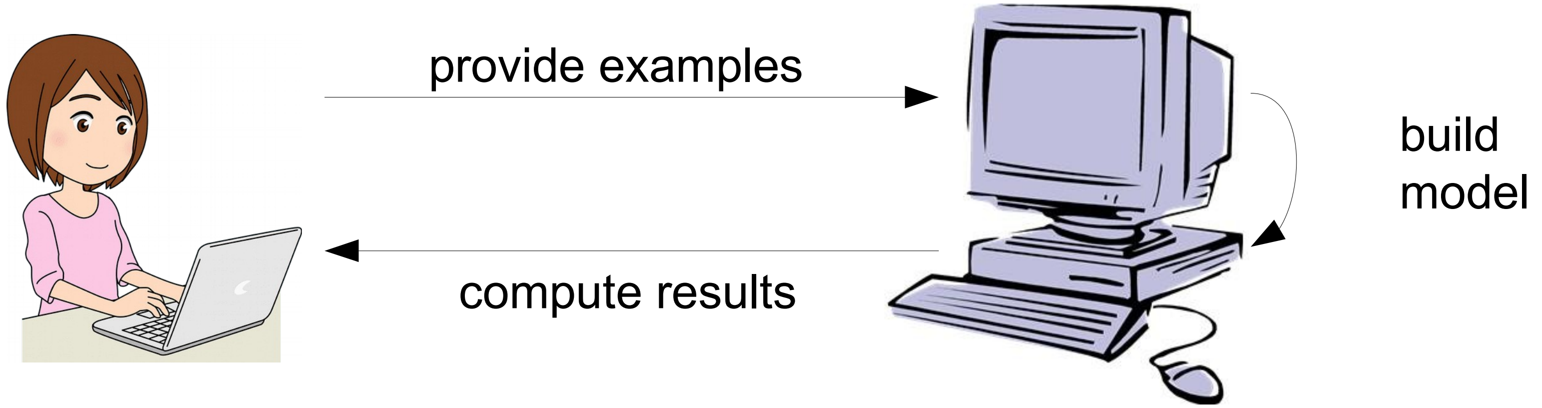
What is Classification?

- Sometimes, it's not so easy
- E.g., due to missing knowledge
  - `if customer is likely to order a new phone`
    `send advertisement for new phones`
- E.g., due to difficult formalization as an algorithm
  - `if customer review is angry`
    `send apology`



What is Classification?

- A different paradigm:
  - User provides computer with examples
  - Computer finds model by itself
  - Notion: the computer *learns* from examples (term: *machine learning)*
- Example
  - labeled examples of angry and non-angry customer reviews
  - computer finds model for telling if a customer is angry



Classification: Formal Definition

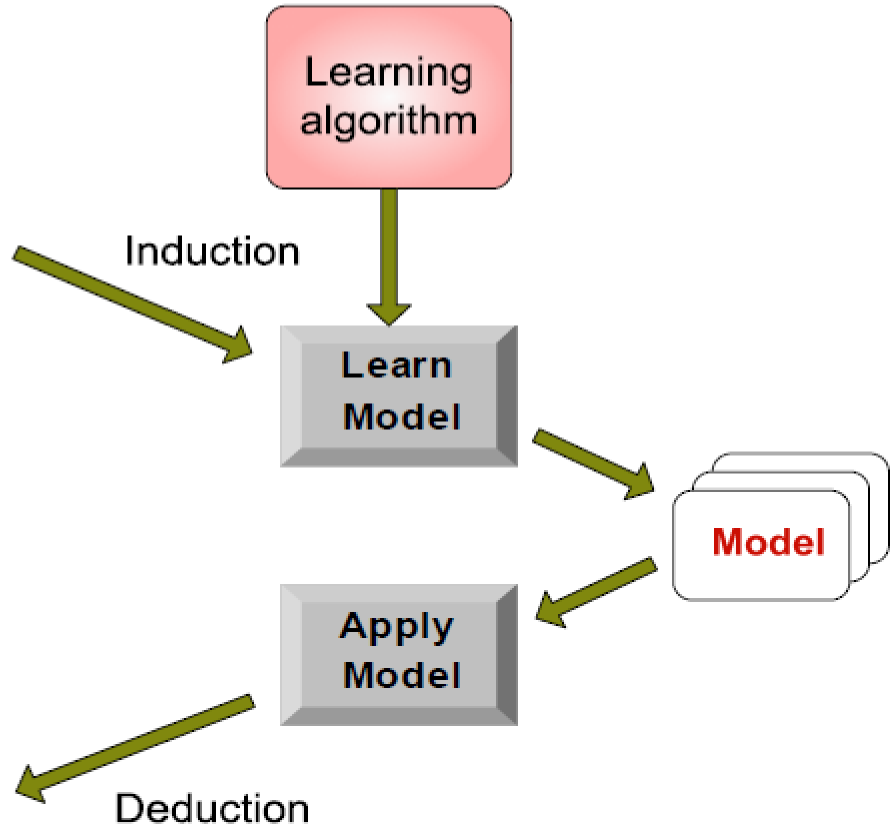
- Given:
  - a set of labeled records, consisting of
    - data fields (a.k.a. attributes or features)
    - a class label (e.g., true/false)
- Generate
  - a function f(r)
    - input: a record
    - output: a class label
  - which can be used for classifying previously unseen records

- Variants:
  - single class problems (e.g., only true/false)
  - multi class problems
  - multi label problems (more than one class per record, not covered in this lecture)
  - hierarchical multi class/label problems (with class hierarchy, e.g., product categories)

The Classification Workflow

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Unseen Records

Classification Applications – Examples

- Attributes: a set of symptoms (cough, sore throat...)
  - class: does the patient suffer from CoViD-19?
- Attributes: the values in your tax declaration
  - class: are you trying to cheat?
- Attributes: your age, income, debts, …
  - class: are you getting credit by your bank?
- Attributes: the countries you phoned with in the last 6 months
  - class: are you a terrorist?

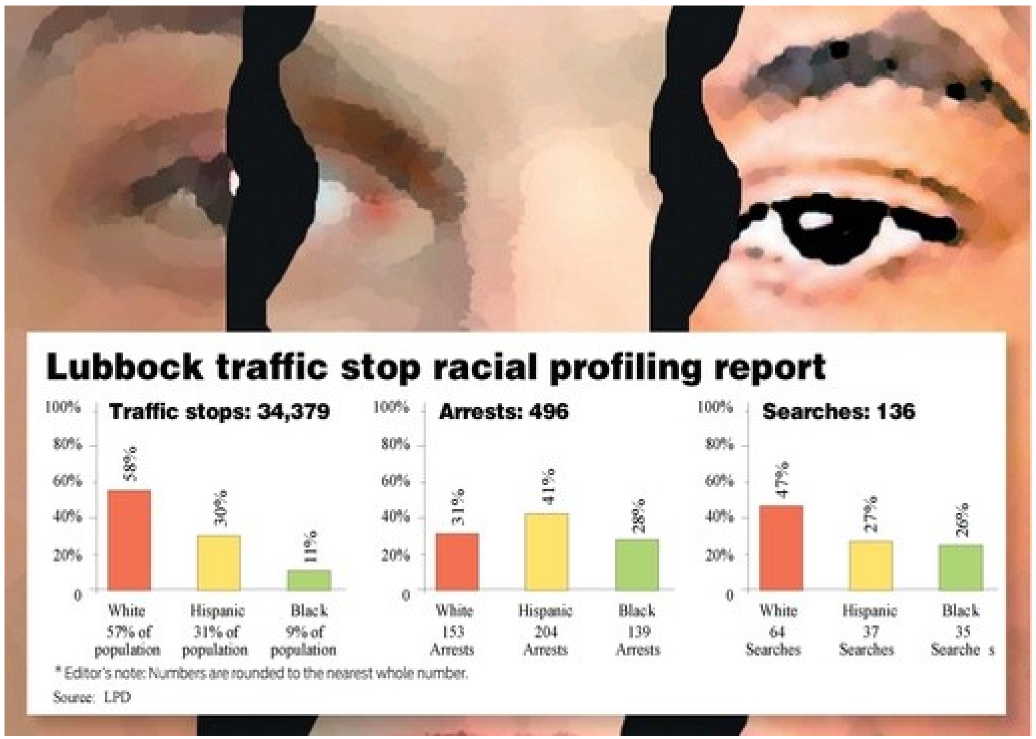
Classification Applications – Examples

- Attributes: words in a product review
  - Class: Is it a fake review?

- Attributes: words and header fields of an e-mail
  - Class: Is it a spam e-mail?

Classification Applications – Examples

- A controversial example
  - Class: whether you are searched by the police
  - Class: whether you are selected at the airport for an extra check

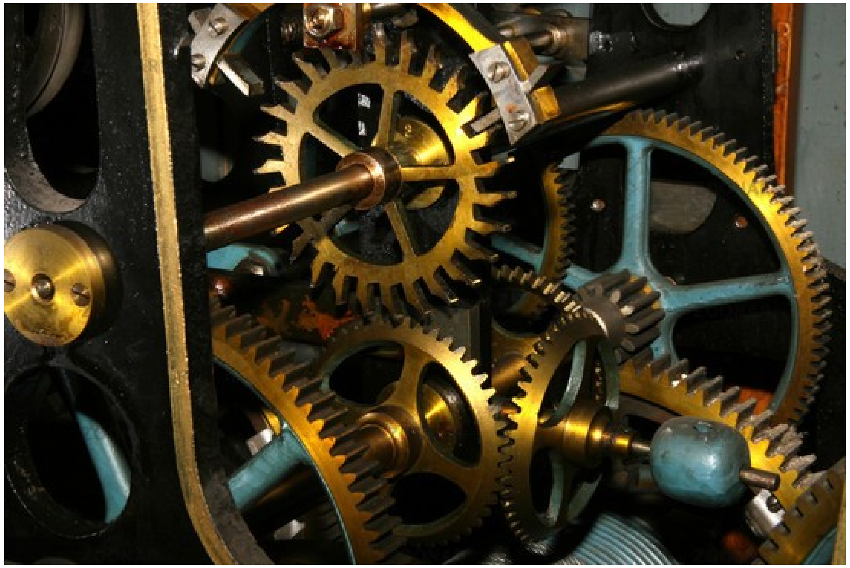


http://lubbockonline.com/stories/030609/loc_405504016.shtml

Classification Algorithms

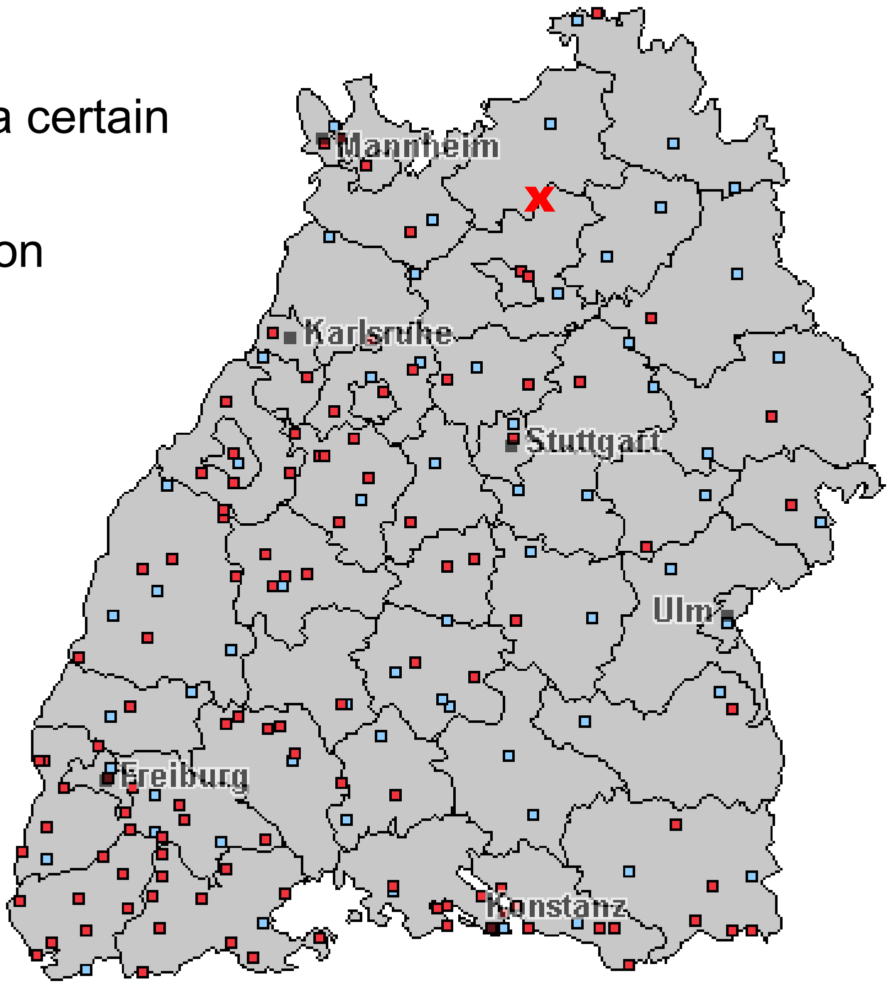
- Recap:
  - we give the computer a set of labeled examples
  - the computer learns to classify new (unlabeled) examples

- How does that work?



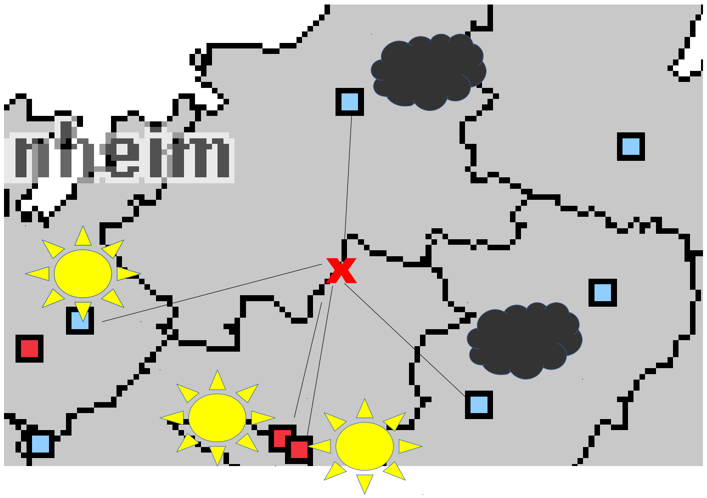
k Nearest Neighbors

- Problem
  - find out what the weather is in a certain place
  - where there is no weather station
  - how could you do that?



k Nearest Neighbors

- Idea: use the average of the nearest stations
- Example:
  - 3x sunny
  - 2x cloudy
  - result: sunny



- Approach is called
  - “k nearest neighbors”
  - where k is the number of neighbors to consider
  - in the example: k=5
  - in the example: “near” denotes geographical proximity

k Nearest Neighbors

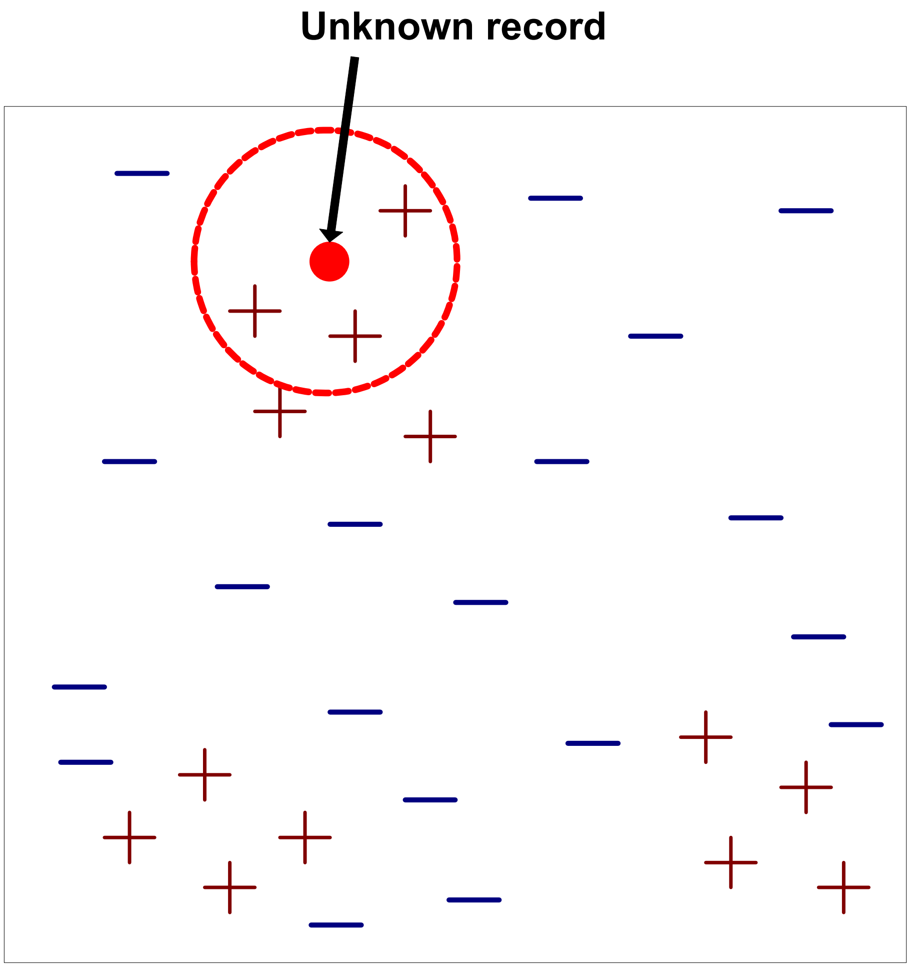
- Further examples:
- Is a customer going to buy a product?
  - → have similar customers bought that product?
- What party are you going to vote for?
  - → what party do your (closest) friends/family members vote for?
- Is a film going to win an oscar?
  - → have similar films won an oscar?

- and so on...

Recap: Similarity and Distance

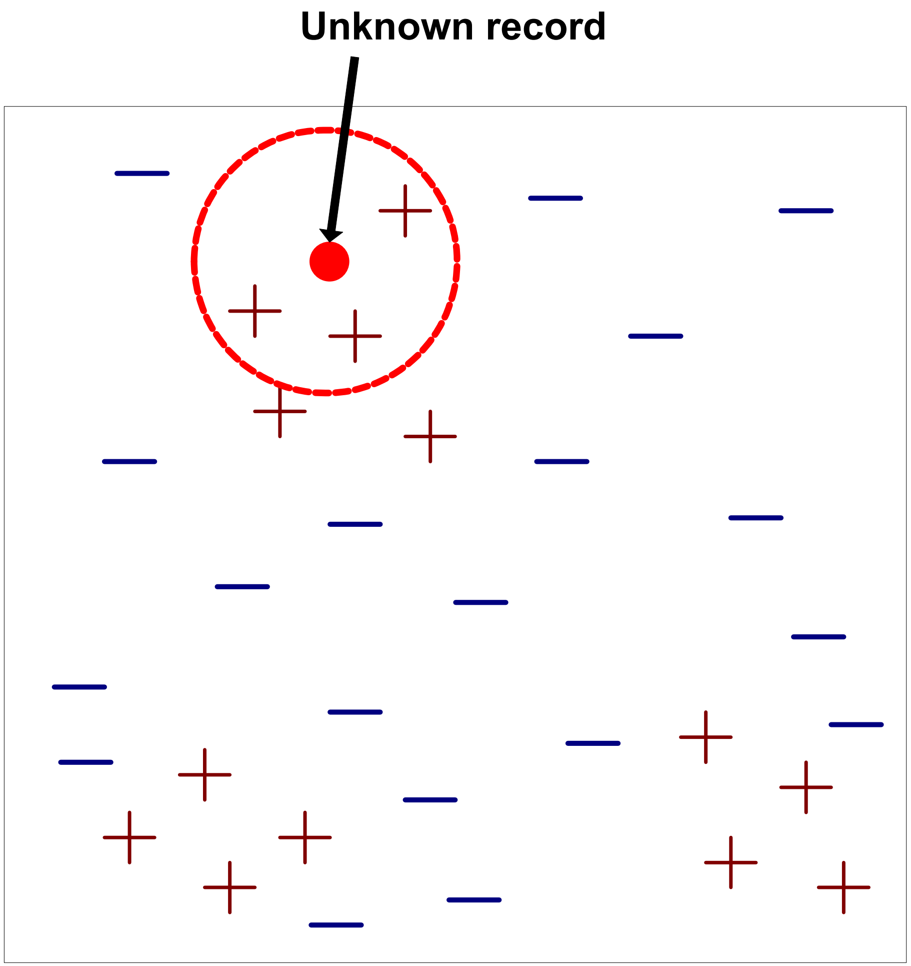
- k *Nearest* Neighbors
  - requires a notion of similarity (i.e., what is “near”?)
- Review: similarity measures for clustering
  - similarity of individual data values
  - similarity of data points
- Think about scales and normalization!

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - A distance metric to compute distance between records
  - The value of k, the number of nearest neighbors to retrieve

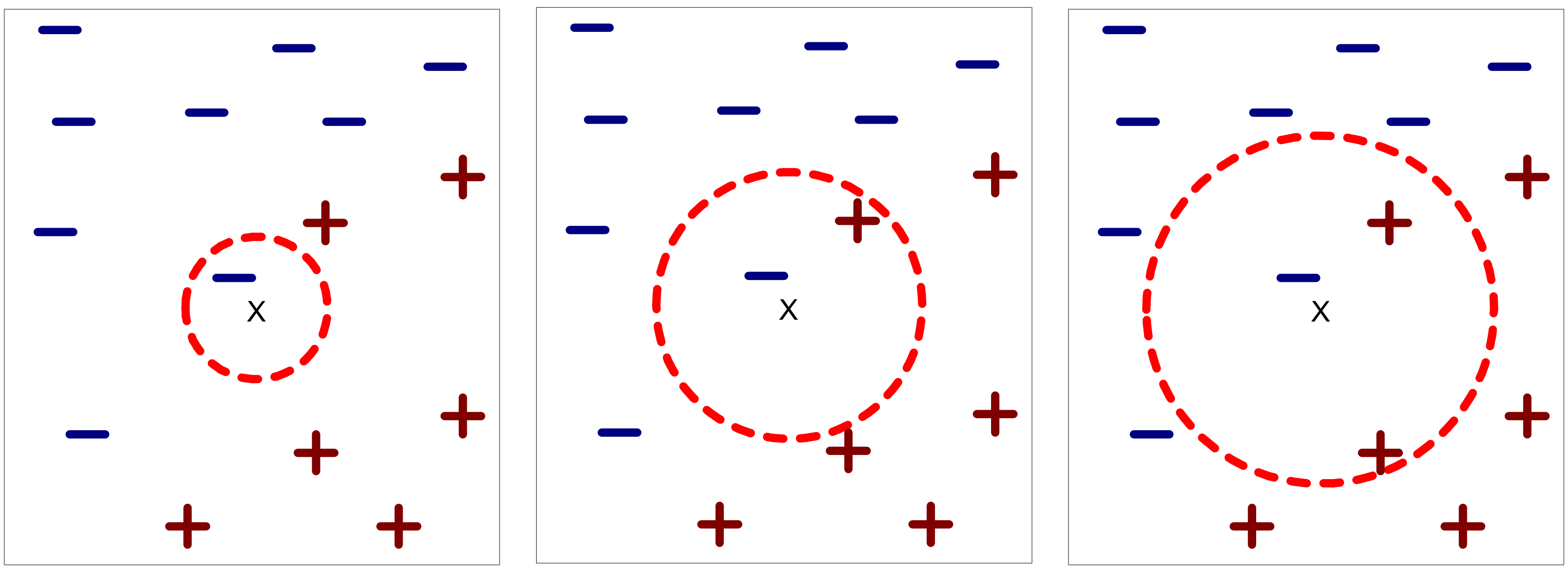
Nearest-Neighbor Classifiers



- To classify an unknown record:
  - Compute distance to each training record
  - Identify k nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record
    - by taking majority vote
    - by weighing the vote according to distance

Definition of the k Nearest Neighbors

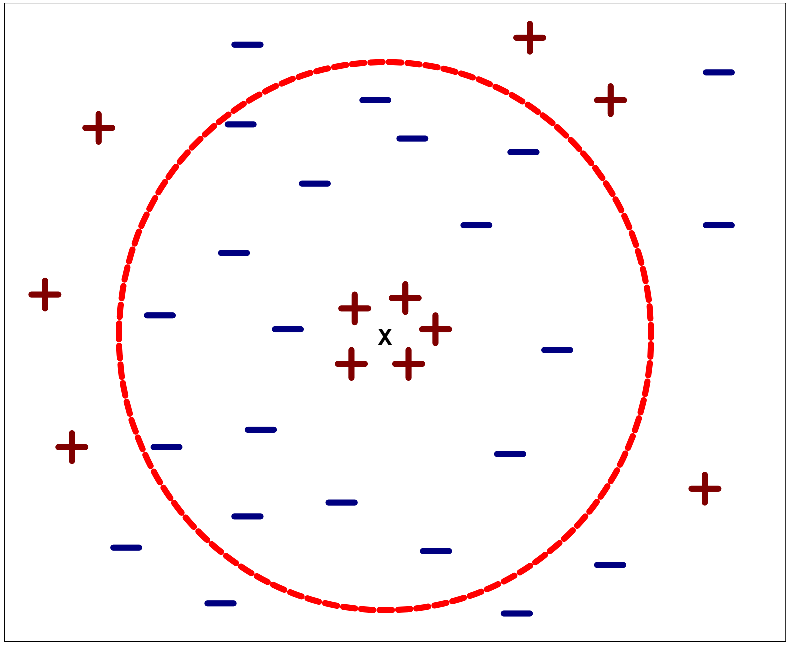
The k nearest neighbors of a record x are data points that have the k smallest distance to x.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Choosing a Good Value for k

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes



- Rule of thumb: Test k values between 1 and 10.

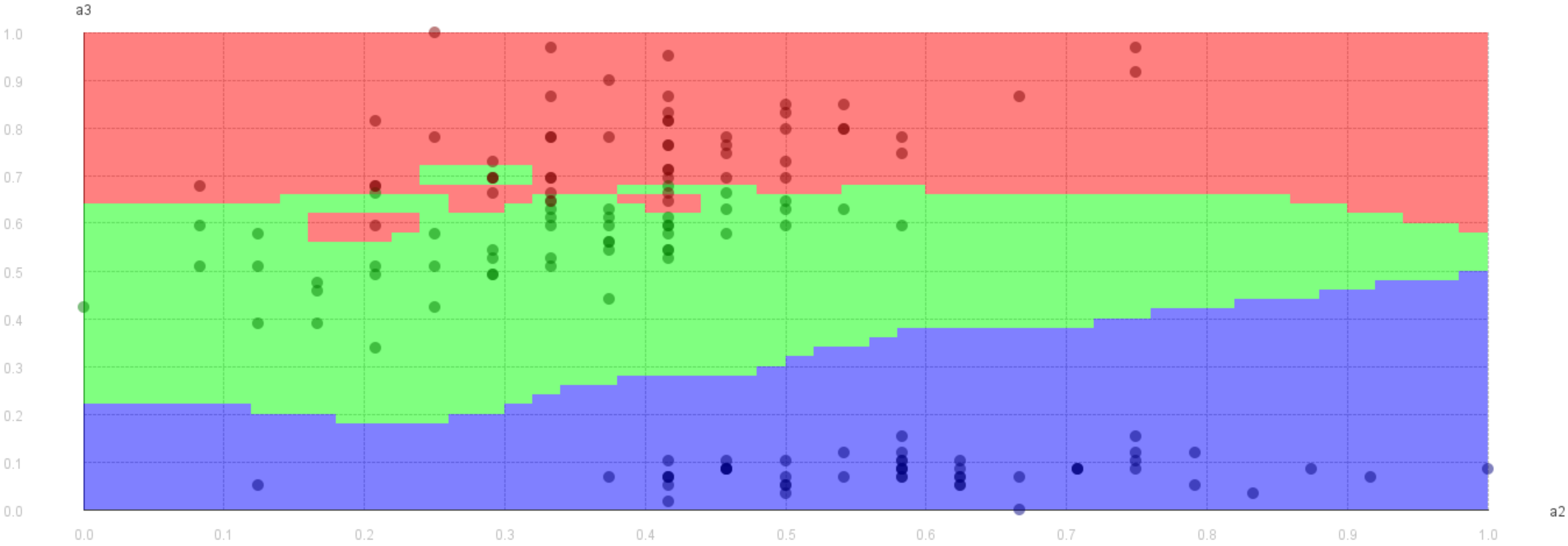
Discussion of K-Nearest Neighbor

- Often very accurate
- … but slow as training data needs to be searched

- Can handle decision boundaries which are not parallel to the axes
- Assumes all attributes are equally important
  - Remedy: Attribute selection or using attribute weights

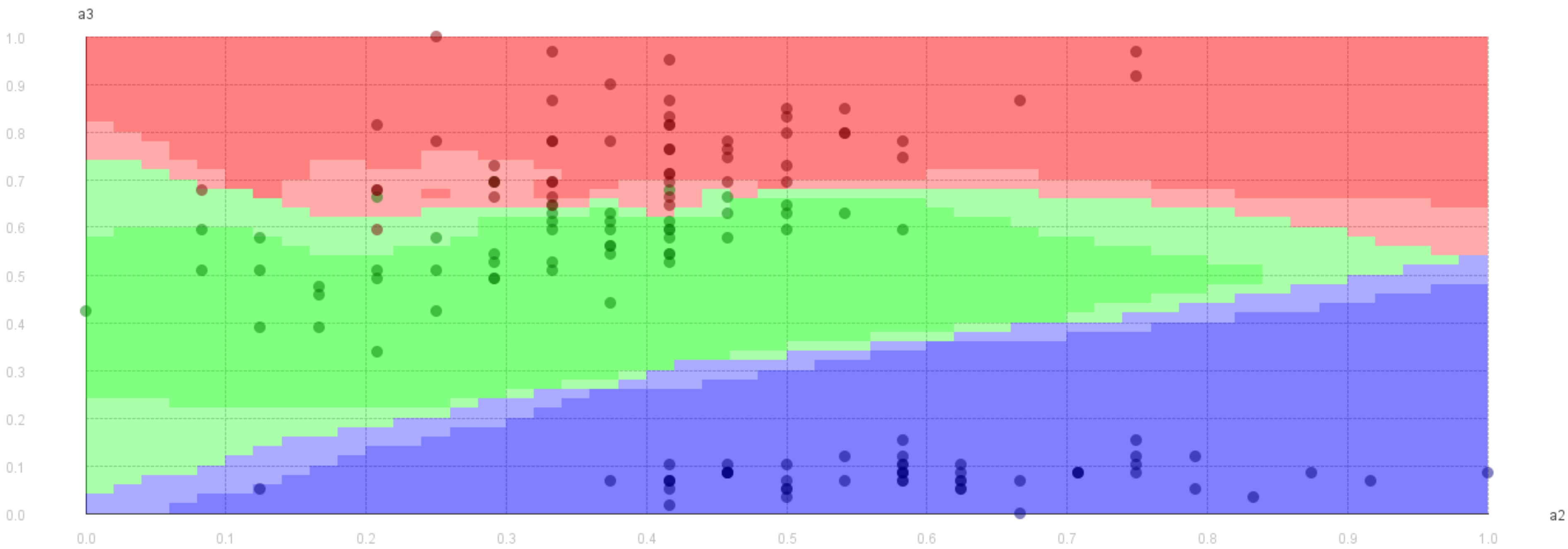
Decision Boundaries of a k-NN Classifier

- k=1
- Single noise points have influence on model

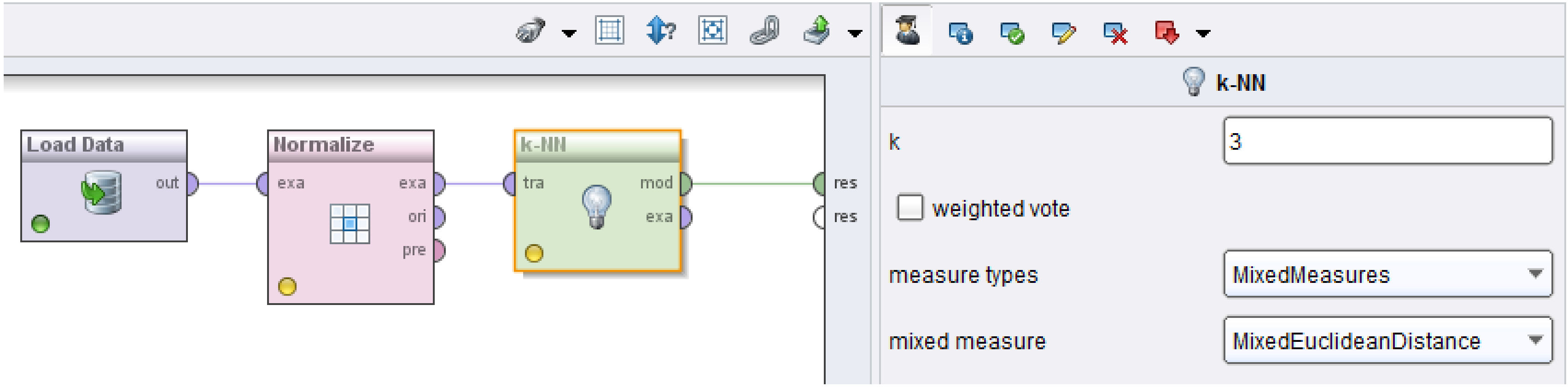


Decision Boundaries of a k-NN Classifier

- k=3
- Boundaries become smoother
- Influence of noise points is reduced

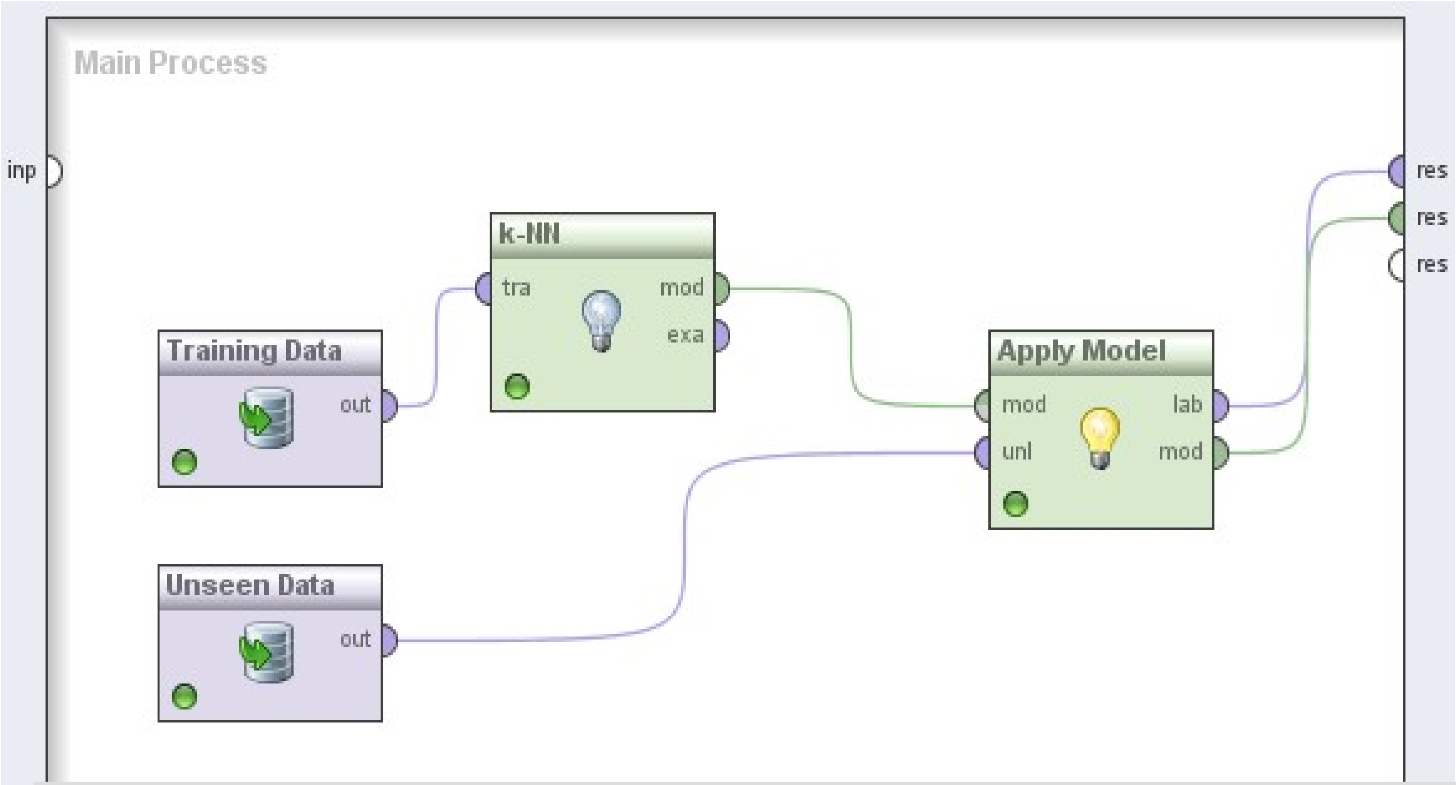


KNN in RapidMiner & Python




```
scaler = MinMaxScaler()
features_norm = scaler.fit_transform(features)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(features_norm,label)
```

Applying the Model




```
test_norm = scaler.transform(test)
model.predict(test_norm)
```

Contrast: Nearest Centroids

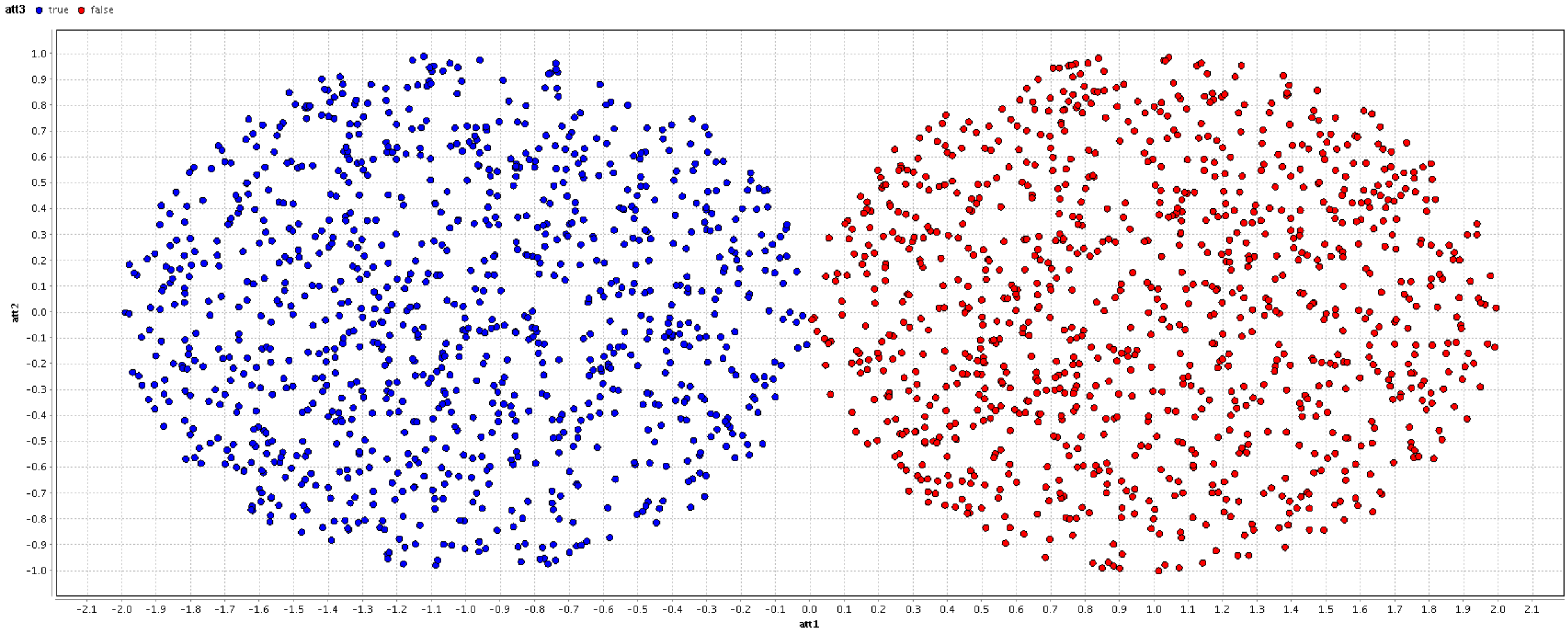
- a.k.a. Rocchio classifier
- Training: compute centroid for each class
  - center of all points of that class
  - like: centroid for a cluster
- Classification:
  - assign each data point to nearest centroid

Sounds pretty much just like k-NN, huh?

- RapidMiner:
  - available in Mannheim RapidMiner Toolbox Extension
- Python:
  - scikit_learn.neighbors.NearestCentroid

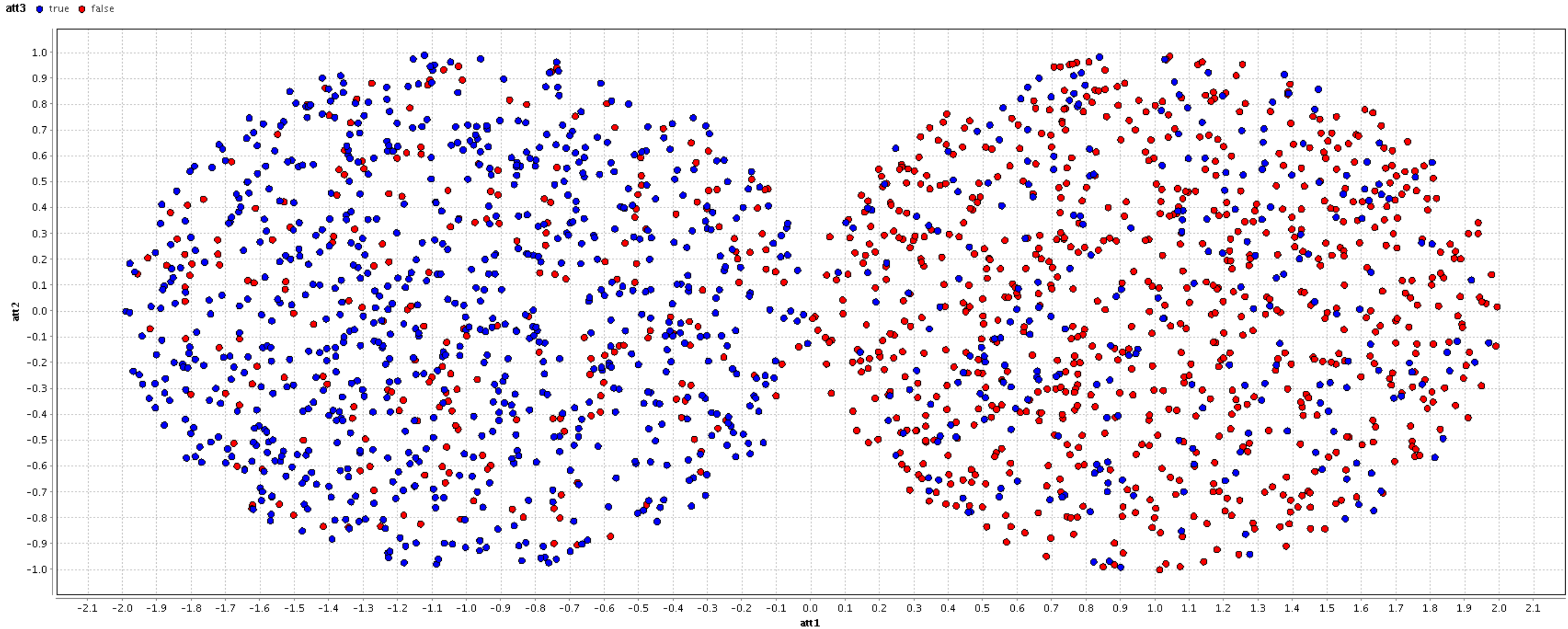
k-NN vs. Nearest Centroid

- Basic problem: two circles
  - Both k-NN and Nearest Centroid are rather perfect



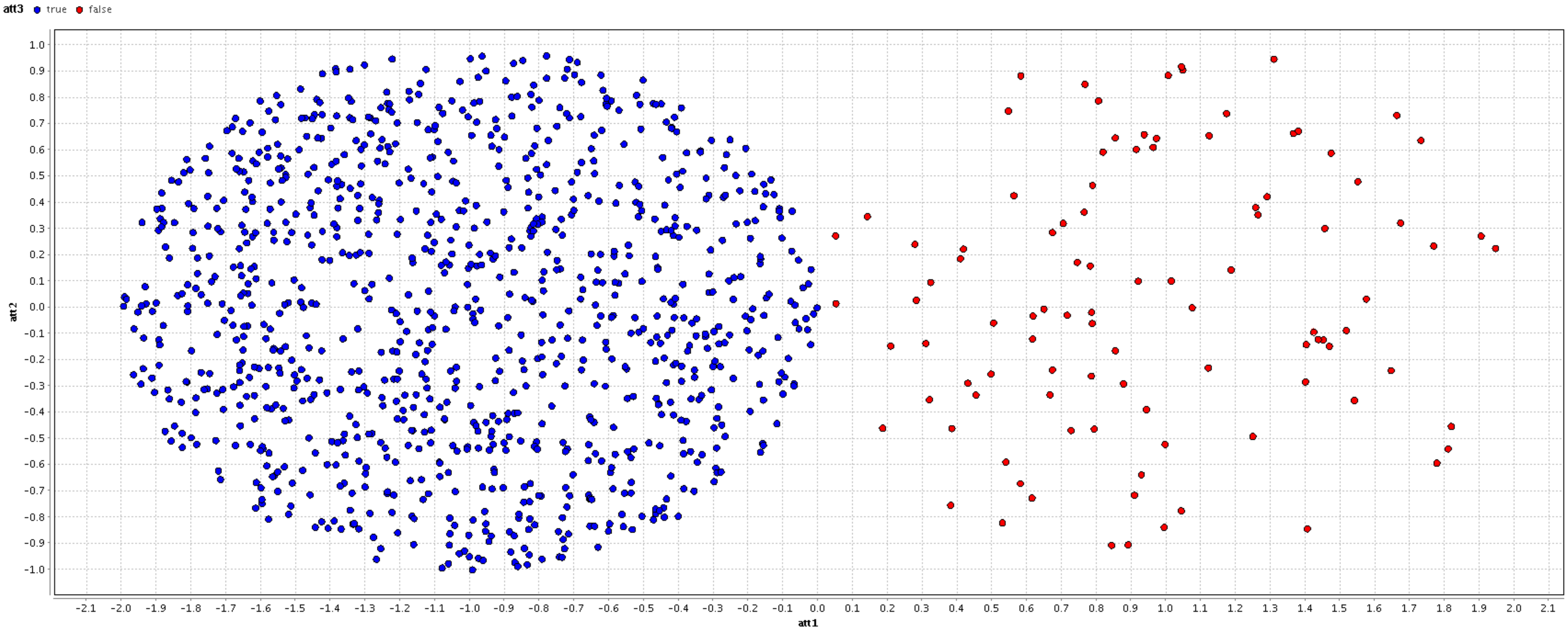
k-NN vs. Nearest Centroid

- Some data points are mislabeled
  - k-NN loses performance
  - Nearest Centroid is stable



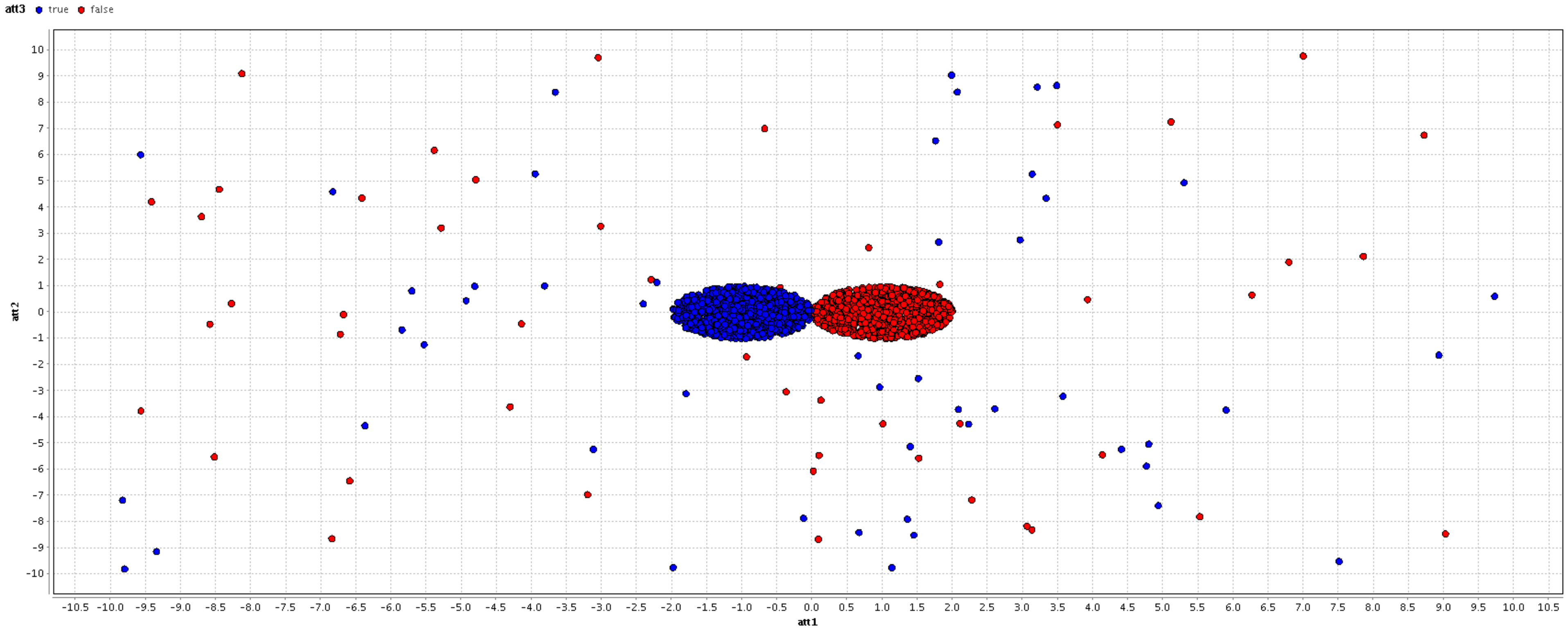
k-NN vs. Nearest Centroid

- One class is significantly smaller than the other
  - k-NN loses performance
  - Nearest Centroid is stable



k-NN vs. Nearest Centroid

- Outliers are contained in the dataset
  - k-NN is stable
  - Nearest Centroid loses performance



k-NN vs. Nearest Centroid

- k-NN
  - slow at classification time (linear in number of data points)
  - requires much memory (storing all data points)
  - robust to outliers
- Nearest Centroid
  - fast at classification time (linear in number of classes)
  - requires only little memory (storing only the centroids)
  - robust to label noise
  - robust to class imbalance

- Which classifier is better?
  - that strongly depends on the problem at hand!

Bayes Classifier

- Based on Bayes Theorem
- Thomas Bayes (1701-1761)
  - British mathematician and priest
  - tried to formally prove the existence of God
- Bayes Theorem
  - important theorem in probability theory
  - was only published after Bayes' death

Conditional Probability and Bayes Theorem

- A probabilistic framework for solving classification problems
- Conditional Probability:

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

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

- Bayes theorem:

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

Conditional Probability and Bayes Theorem

- Bayes Theorem
  - Computes one conditional probability P(C|A) out of another P(A|C)
  - given that the base probabilities P(A) and P(C) are known

- Useful in situations where P(C|A) is unknown
  - while P(A|C), P(A) and P(C) are known or easy to determine/estimate

- Example:
  - Given a symptom, what's the probability that I have a certain disease?

Example of Bayes Theorem

- PCR test for SaRS-CoV-2
  - exact quality is unknown
- Optimistic estimates[1]
  - If you're infected, PCR shows a positive result with p=95% (called “sensitivity”)
  - If you're not infected, PCR shows a negative result also with p=95% (called “specificity”)
- Assume you have a positive test
  - What's the probability that you're infected with SARS-CoV-2?

[1]see https://www.nejm.org/doi/full/10.1056/NEJMp2015897

Example of Bayes Theorem

- We want to know P(Corona|pos)
  - Bayes theorem:

~3% in Germany

$$P(Cor \mid pos) = \frac{P(pos \mid Cor)\, P(Cor)}{P(pos)}$$

- We still need P(pos)
  - i.e., the probability that a test is positive

$$P(pos) = P(pos \mid Cor \vee \neg Cor)$$
$$= P(pos \mid Cor) \cdot P(Cor) + P(pos \mid \neg Cor) \cdot P(\neg Cor)$$

Example of Bayes Theorem

- Now: numbers

$$P(Corona \mid pos) = \frac{P(pos \mid Corona)\,P(Corona)}{P(pos)}$$

$$= \frac{P(pos \mid Corona)\,P(Corona)}{P(pos \mid Cor)\cdot P(Cor) + P(pos \mid \neg Cor)\cdot P(\neg Cor)}$$

$$= \frac{0.95\cdot 0.03}{0.95\cdot 0.03 + 0.05\cdot 0.97} = 0.37$$

- That means:
  - at more than 65% probability, you are still healthy, given a positive test!

- Caveat:
  - numbers P(Cor) and (P¬Cor) are different due to non-random testing!

Estimating the Prior Probability P(C)

- The prior probability $P(C_j)$ for each class is estimated by
  1. counting the records in the training
     set that are labeled with class C_j
  2. dividing the count by the overall number of records
- Example:
  - P(Play=no) = 5/14
  - P(Play=yes) = 9/14

Training Data

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

Estimating the Conditional Probability P(A | C)

- Naïve Bayes assumes that all attributes are statistically independent
  - knowing the value of one attribute says nothing about the value of another
  - this independence assumption is almost never correct!
  - but … this scheme works well in practice
- The independence assumption allows the joint probability P(A | C) to be reformulated as the product of the individual probabilities $P(A_i| C_j)$:

$$P(A_1, A_2, \ldots, A_n \mid C_j) = \prod P(A_n| C_j) = P(A_1| C_j)\times P(A_2| C_j)\times \ldots \times P(A_n| C_j)$$

P(Outlook=rainy, Temperature=cool | Play=yes)
= P(Outlook=rainy | Play=yes) × P(Temperature=cool | Play=yes)

- Result: The probabilities $P(A_i| C_j)$ for all A_i and C_j can be estimated directly from the training data

Estimating the Probabilities $P(A_i \mid C_j)$

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	9	5
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	9/14	5/14
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								

1.1. count how often an attribute value co-occurs with class C_j
2. divide by the overall number of instances in class C_j

Example:
"Outlook=sunny" occurs on 2/9 examples in class "Yes"
➔ p(Outlook=sunny|Yes) = 2/9

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

Classifying a New Record

Unseen record

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Class-conditional probability of the evidence

$$P(yes \mid E) = P(Outlook = Sunny \mid yes)$$
$$\times P(Temperature = Cool \mid yes)$$
$$\times P(Humidity = High \mid yes)$$
$$\times P(Windy = True \mid yes)$$
$$\times \frac{P(yes)}{P(E)}$$

Probability of class "yes" given the evidence

Prior probability of class "yes"

Prior probability of evidence

$$= \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{P(E)}$$

Classifying a New Record (ctd.)

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	9	5
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	9/14	5/14
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								

– A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Prior probability
Evidence

Likelihood of the two classes

For "yes" = 2/9 x 3/9 x 3/9 x 3/9 x 9/14 = 0.0053

For "no" = 3/5 x 1/5 x 4/5 x 3/5 x 5/14 = 0.0206

Conversion into a probability by normalization:

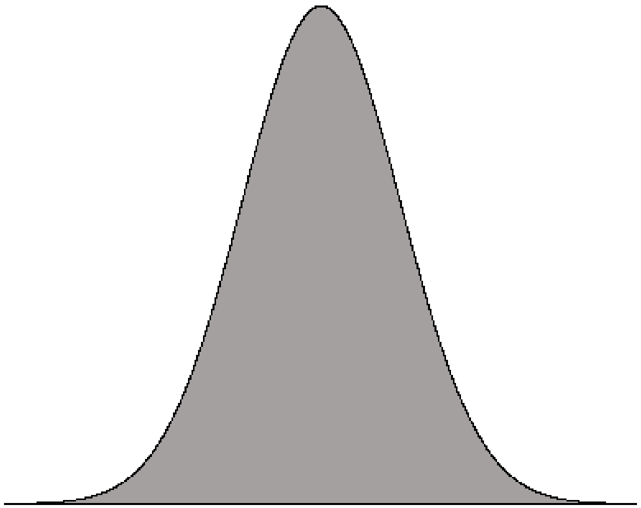
P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

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

Choose Maximum

Handling Numerical Attributes

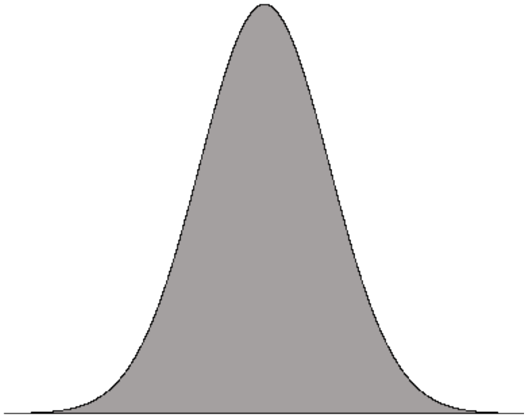
- Option 1:
  Discretize numerical attributes before learning classifier.
  - Temp= 37°C ➔ "Hot"
  - Temp= 21°C ➔ "Mild"
- Option 2:
  Make assumption that numerical attributes have a normal distribution given the class.
  - use training data to estimate parameters of the distribution
    (e.g., mean and standard deviation)
  - once the probability distribution is known, it can be used to estimate the conditional probability $P(A_i|C_j)$



Handling Numerical Attributes

– The probability density function for the normal distribution is

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



– It is defined by two parameters:

- *Sample mean μ* $\quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
- *Standard deviation σ* $\quad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2}$

– Both parameters can be estimated from the training data

Statistics for the Weather Data

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	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5		
Rainy	3/9	2/5									

Example calculation:

$$f(temp = 66 \mid yes) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.0340$$

Classifying a New Record

Unseen record

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = 2/9 × 0.0340 × 0.0221 × 3/9 × 9/14 = 0.000036

Likelihood of "no" = 3/5 × 0.0291 × 0.0380 × 3/5 × 5/14 = 0.000136

P("yes") = 0.000036 / (0.000036 + 0. 000136) = 20.9%

P("no") = 0.000136 / (0.000036 + 0. 000136) = 79.1%

Caveat: Some numeric attributes are not normally distributed and you may thus need to choose a different probability density function or use discretization

Handling Missing Values

– Missing values may occur in training and in unseen classification records

– Training: Record is not included into frequency count for attribute value-class combination

– Classification: Attribute will be omitted from calculation

- Example:

Unseen record

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

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

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

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

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

Zero Frequency Problem

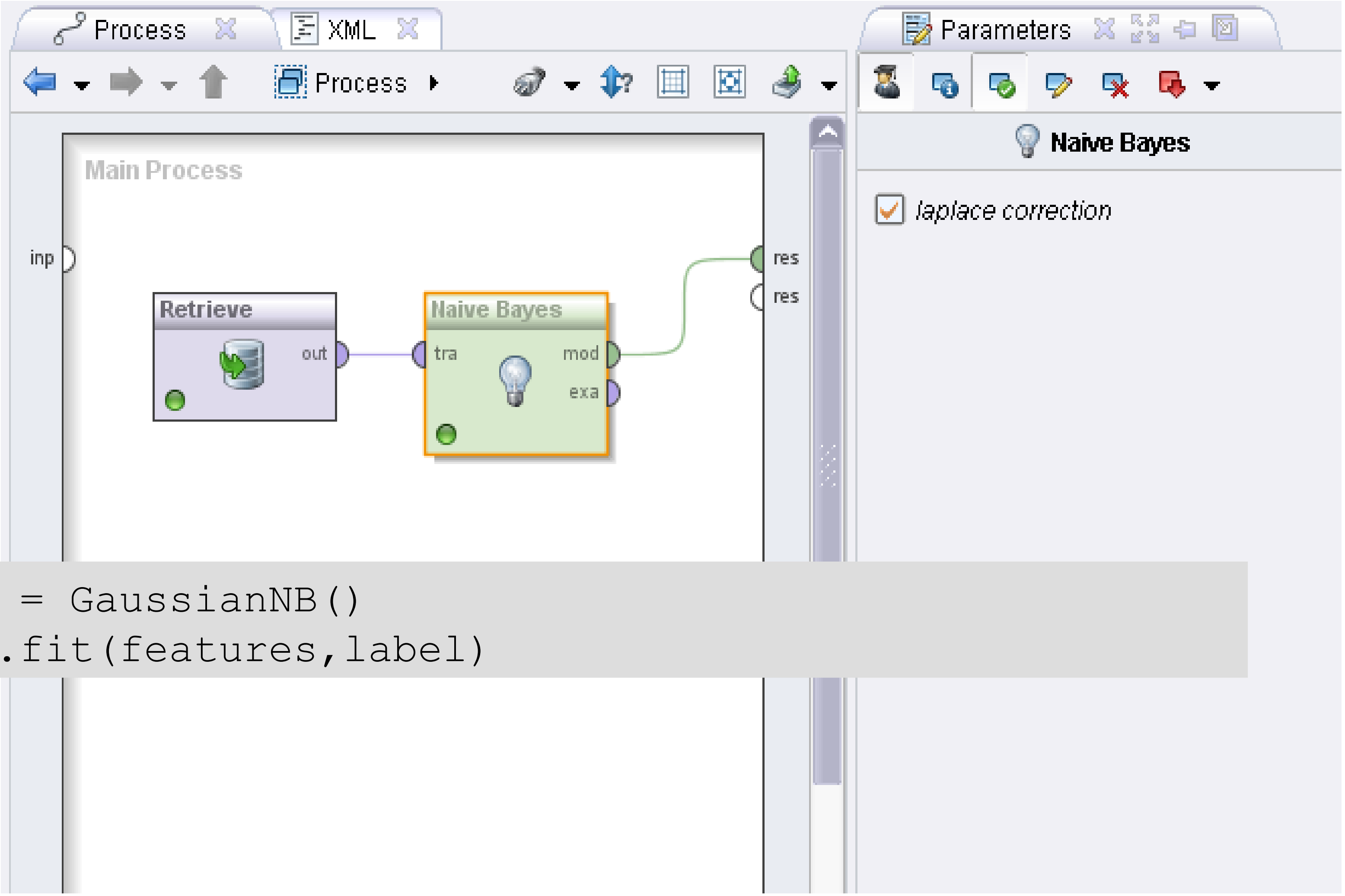
- If one of the conditional probabilities is zero, then the entire expression becomes zero
- And it is not unlikely that an exactly same data point has not yet been observed
- Probability estimation:

$$\text{Original: } P(A_i|C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace: } P(A_i|C) = \frac{N_{ic}+1}{N_c+c}$$

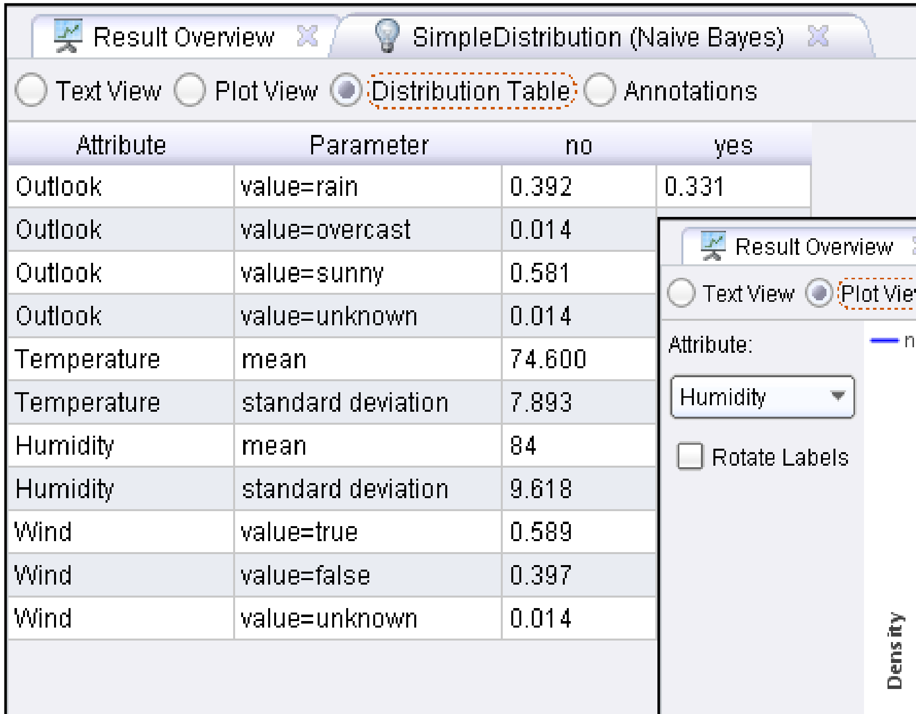
c: number of attribute values of A

Naïve Bayes in RapidMiner & Python

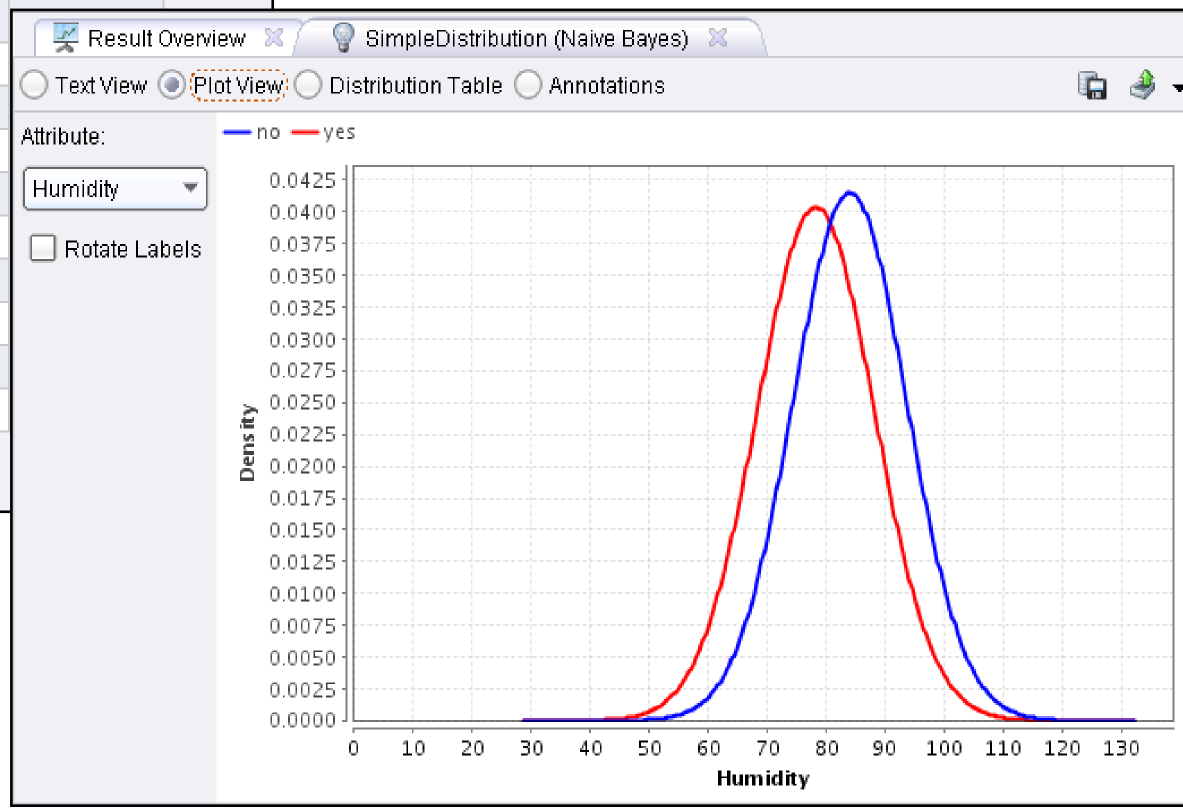



```
model = GaussianNB()
model.fit(features,label)
```

Anatomy of a Naïve Bayes Model



Attribute	Parameter	no	yes
Outlook	value=rain	0.392	0.331
Outlook	value=overcast	0.014	
Outlook	value=sunny	0.581	
Outlook	value=unknown	0.014	
Temperature	mean	74.600	
Temperature	standard deviation	7.893	
Humidity	mean	84	
Humidity	standard deviation	9.618	
Wind	value=true	0.589	
Wind	value=false	0.397	
Wind	value=unknown	0.014	



Using Conditional Probabilities for Naïve Bayes

Result Overview | ExampleSet (Retrieve Golf-Testset)

Data View | Meta Data View | Plot View | Advanced Charts | Annotations

ExampleSet (14 examples, 4 special attributes, 4 regular attributes) View Filter (14 / 14):

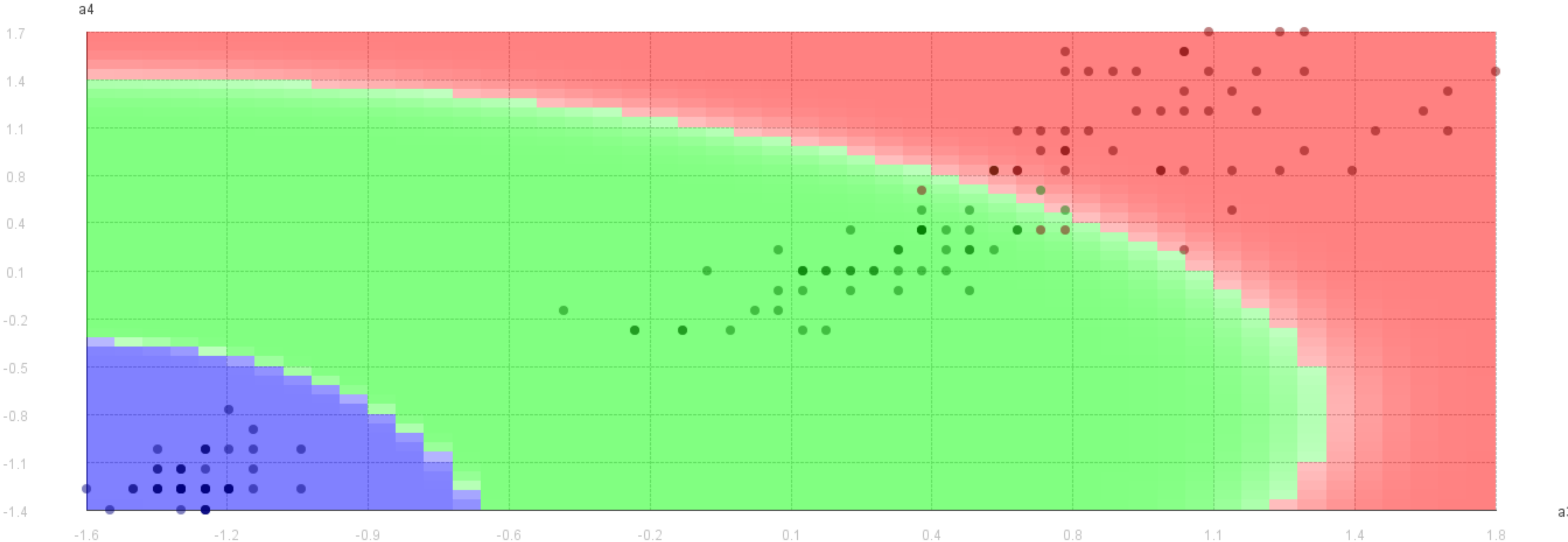
Row No.	Play	confidence(no)	confidence(yes)	prediction(Play)	Outlook	Temperature	Humidity	Wind
1	yes	0.711	0.289	no	sunny	85	85	false
2	no	0.058	0.942	yes	overcast	80	90	true
3	yes	0.014	0.986	yes	overcast	83	78	false
4	yes	0.412	0.588	yes	rain	70	96	false
5	yes	0.460	0.540	yes	rain	68	80	true
6	no	0.336	0.664	yes	rain	65	70	true
7	yes	0.010	0.990	[illegible]	[illegible]	[illegible]	[illegible]	true
8	no	0.596	0.404	no	[illegible]	[illegible]	[illegible]	false
9	yes	0.248	0.752	yes	sunny	69	70	false
10	no	0.407	0.593	yes	sunny	75	80	false
11	yes	0.496	0.504	[illegible]	[illegible]	[illegible]	[illegible]	true
12	yes	0.038	0.962	yes	[illegible]	[illegible]	[illegible]	true
13	no	0.027	0.973	yes	overcast	81	75	true
14	yes	0.453	0.547	yes	rain	71	80	true

classifier is quite sure

classifier is not sure

Decision Boundary of Naive Bayes Classifier

- Usually larger coherent areas
- Soft margins with uncertain regions
- Arbitrary (often curved) shapes



Naïve Bayes (Summary)

- Robust to isolated noise points
  - they have a small impact on the probabilities
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

Why *Naïve* Bayes?

- Recap:
  - we assume that all the attributes are independent

- This does not hold for many real world datasets
  - e.g., persons: sex, weight, height
  - e.g., cars: weight, fuel consumption
  - e.g., countries: population, area, GDP
  - e.g., food: ingredients
  - e.g., text: word occurrences (“Donald”, “Trump”, “Duck”)
  - ...

Naïve Bayes Discussion

- Naïve Bayes works surprisingly well
  - even if independence assumption is clearly violated
  - Classification doesn't require accurate probability estimates as long as maximum probability is assigned to correct class
- *Too many* redundant attributes will cause problems
  - Solution: Select attribute subset as Naïve Bayes often works as well or better with just a fraction of all attributes
- Technical advantages:
  - Learning Naïve Bayes classifiers is computationally cheap (probabilities are estimated in one pass over the training data)
  - Storing the probabilities does not require a lot of memory

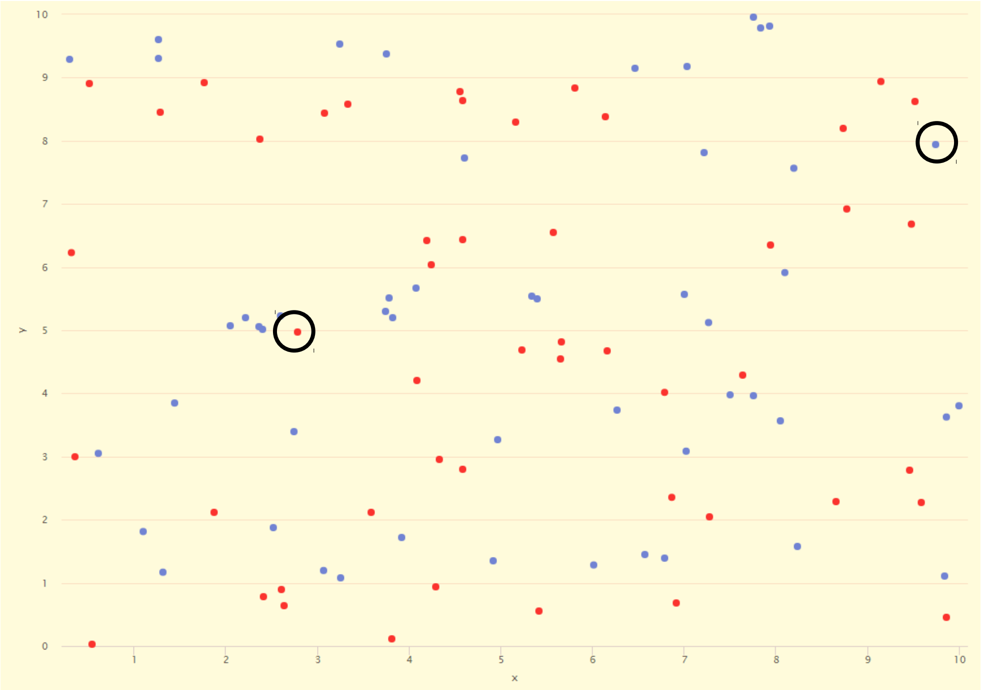
Redundant Variables

- Consider two variables which are perfectly correlated
  - i.e., one is redundant
  - e.g.: a measurement in different units
- Violate independence assumption in Naive Bayes
  - Can, at large scale, skew the result
  - Consider, e.g., a price attribute in 20 currencies

    → price variable gets 20 times more influence
- May also skew the distance measures in k-NN
  - But the effect is not as drastic
  - Depends on the distance measure used

Irrelevant Variables

- Consider a random variable x, and two classes A and B
  - For Naive Bayes: p(x=v|A) = p(x=v|B) for any value v
  - Since it is random, it does not depend on the class variable
  - The overall result does not change
- For kNN:



Comparison kNN and Naïve Bayes

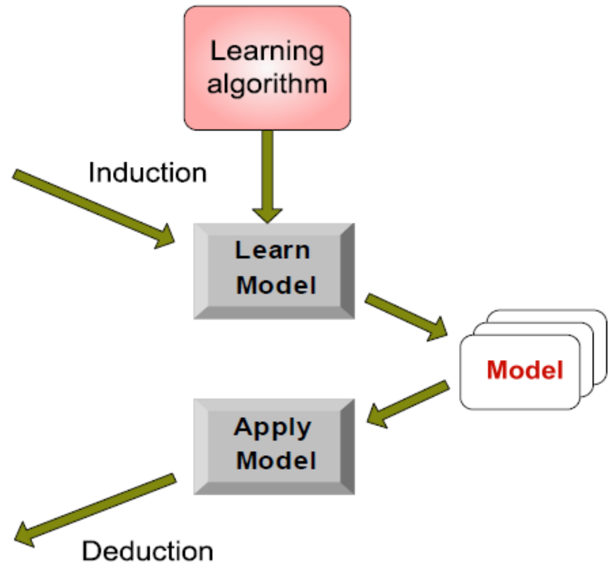
- Computation
  - Naïve Bayes is often faster
- Naïve Bayes uses *all* data points
  - Naive Bayes is less sensitive to label noise
  - k-NN is less sensitive to outliers
- *Redundant* attributes
  - are less problematic for kNN
- *Irrelevant* attributes
  - are less problematic for Naïve Bayes
  - attribute values equally distributed across classes
    → same factor for each class
- In both cases
  - attribute pre-selection makes sense (see Data Mining II)

Lazy vs. Eager Learning

- k-NN, and Naïve Bayes are all “lazy” methods
- They do not build an explicit model!
  – “learning” is only performed on demand for unseen records
- Nearest Centroid is a simple “eager” method

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Unseen Records

Lazy vs. Eager Learning

- We have seen three of the most common techniques for lazy learning
  - k nearest neighbors
  - Naïve Bayes
- ...and a very simple technique for eager learning
  - Nearest Centroids

- We will see more eager learning in the next lectures
  - where explicit models are built
  - e.g., decision trees
  - e.g., rule sets

Model Evaluation

- This week: metrics
  - how to measure performance?
  - here: quality of predictions, not: training time
- Next week: evaluation methods
  - how to obtain meaningful and reliable estimates?

Metrics for Performance Evaluation

- Looking at correctly/incorrectly classified instances
- Two class problem (positive/negative class):
  - true positives, false positives, true negatives, false negatives
- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	TP	FN
	Class=No	FP	TN

Metrics for Performance Evaluation

- Most frequently used metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Error Rate} = 1 - Accuracy$$

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	TP	FN
	Class=No	FP	TN

What is a Good Accuracy?

- i.e., when are you done?
  - at 75% accuracy?
  - at 90% accuracy?
  - at 95% accuracy?

- Depends on difficulty of the problem!
- Baseline: naive guessing
  - always predict majority class

- Compare
  - Predicting coin tosses with accuracy of 50%
  - Predicting dice roll with accuracy of 50%
  - Predicting lottery numbers (6 out of 49) with accuracy of 50%

Limitation of Accuracy: Unbalanced Data

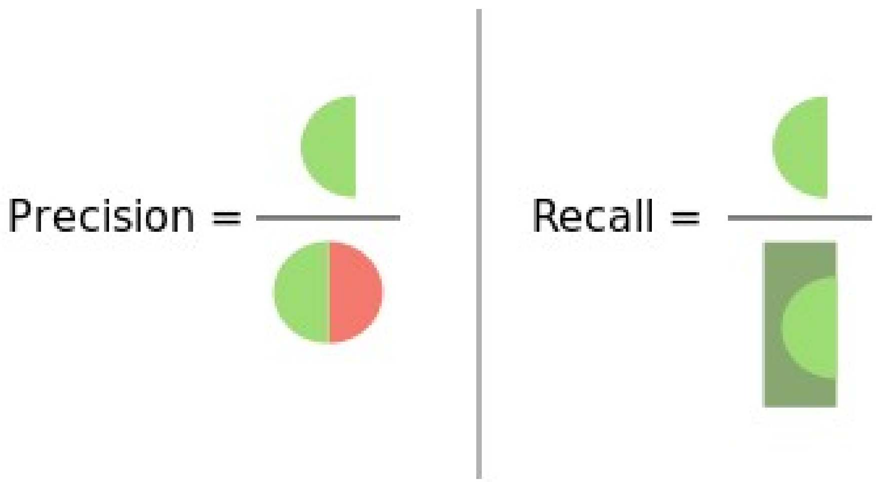
- Sometimes, classes have very unequal frequency
  - Fraud detection: 98% transactions OK, 2% fraud
  - eCommerce: 99% don’t buy, 1% buy
  - Intruder detection: 99.99% of the users are no intruders
  - Security: >99.99% of Americans are not terrorists

- Consider a 2-class problem:
  - Number of Class 0 examples = 9990, Number of Class 1 examples = 10
  - If model predicts everything to be class 0,
    accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect
    any class 1 example

Precision and Recall

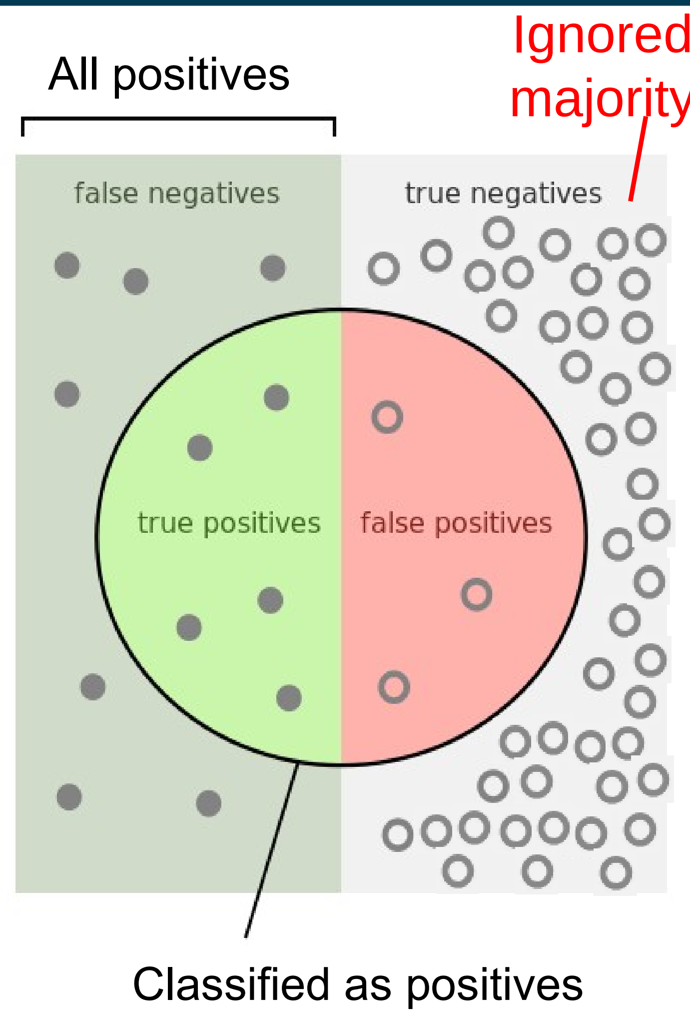
How many examples that are classified positive are actually positive?

Which fraction of all positive examples is classified correctly?



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

$$r = \frac{TP}{TP + FN}$$



Precision and Recall Example

	Predicted positive	Predicted negative
Actual positive	1	99
Actual negative	0	1000

- This confusion matrix gives us
  - precision p = 100% and
  - recall r = 1%
- because we only classified one positive example correctly and no negative examples wrongly
- We want a measure that combines precision and recall

F_1-Measure

- It is hard to compare two classifiers using two measures
- F_1-Score combines precision and recall into one measure
  - by using the *harmonic mean*

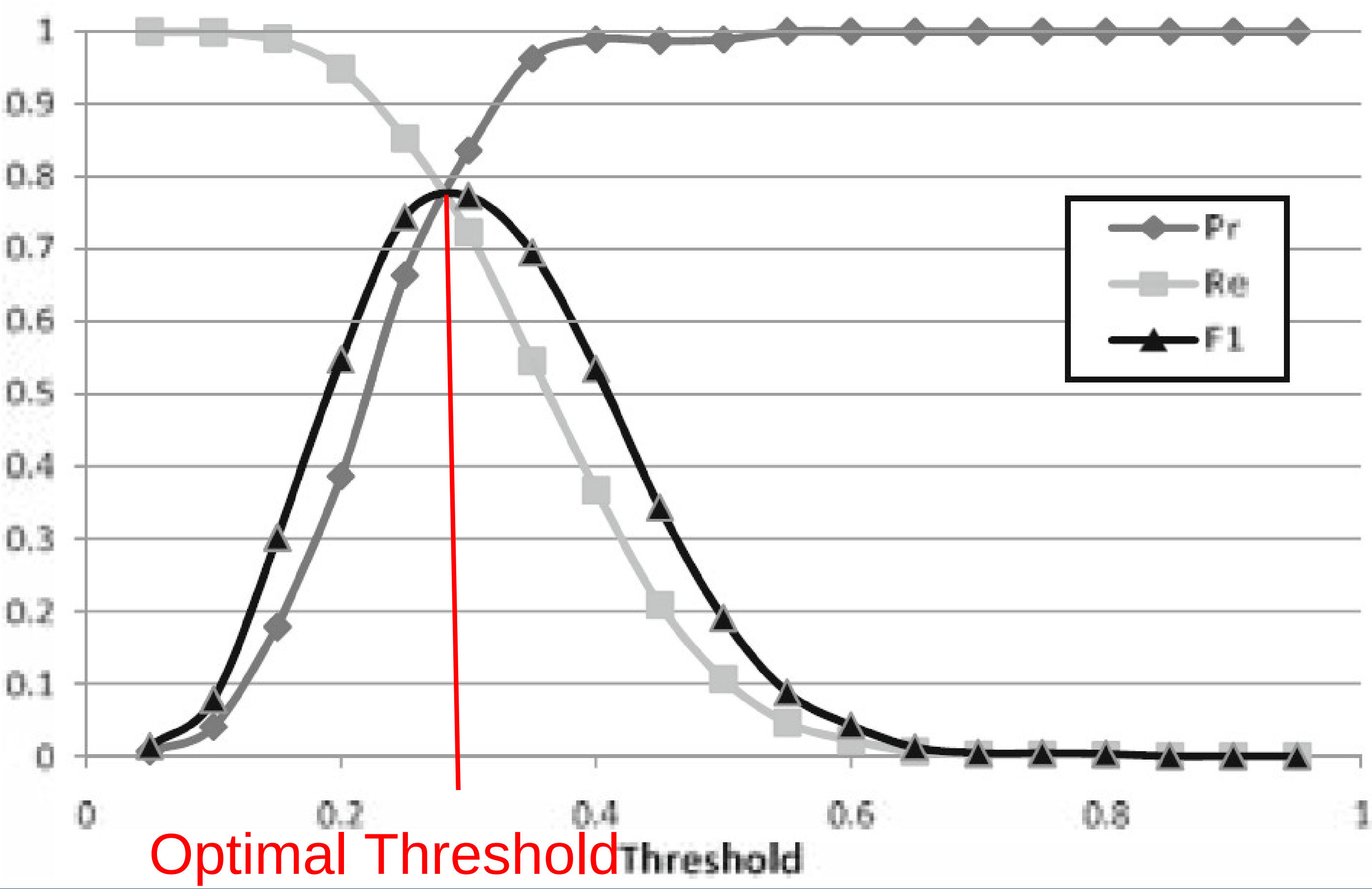
$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}} = \frac{2pr}{p+r}$$

- The harmonic mean of two numbers tends to be closer to the smaller of the two
- For F_1-value to be large, both *p* and *r* must be large

F_1-Measure Graph

Low threshold: Low precision, high recall

Restrictive threshold: High precision, low recall



Alternative for Unbalanced Data: Cost Matrix

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	0	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	162	38
	-	160	240

Accuracy = 67%

Cost = 3960

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	155	45
	-	5	395

Accuracy = 92%

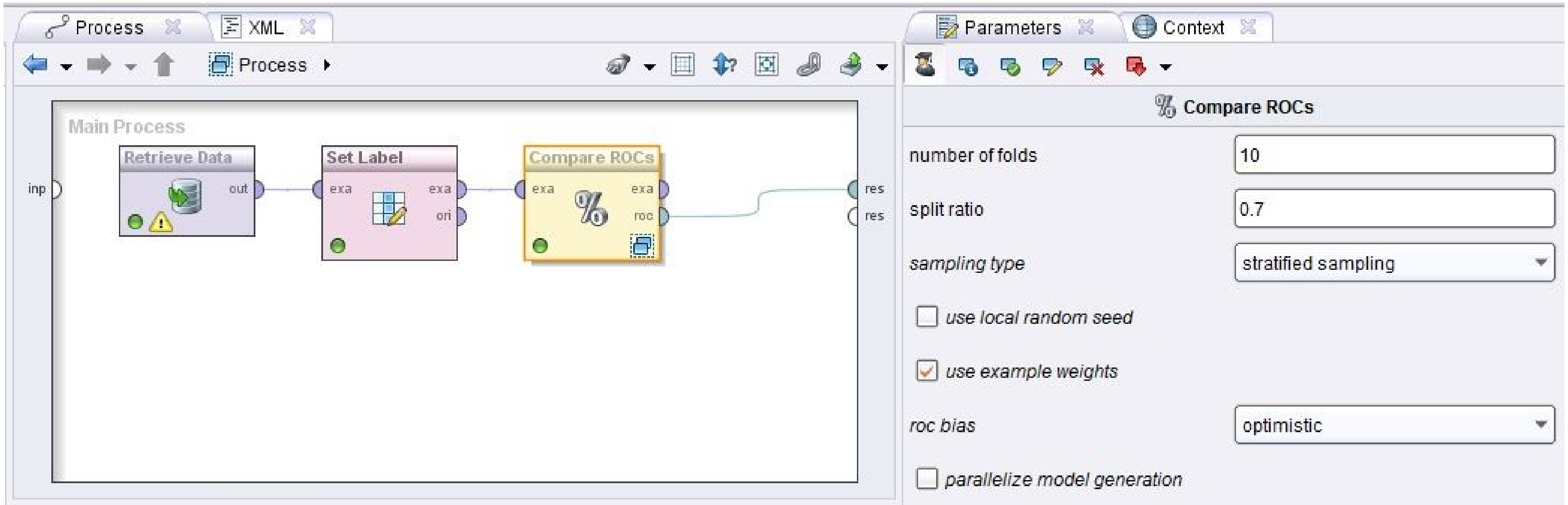
Cost = 4505

ROC Curves

- Some classification algorithms provide confidence scores
  - how sure the algorithms is with its prediction
  - e.g., Naive Bayes: the probability
  - e.g., Decision Trees: the purity of the respective leaf node
- Drawing a ROC Curve
  - Sort classifications according to confidence scores (e.g.: predicted probabilities in Naive Bayes)
  - Evaluate
    - correct prediction: draw one step up
    - incorrect prediction: draw one step to the right

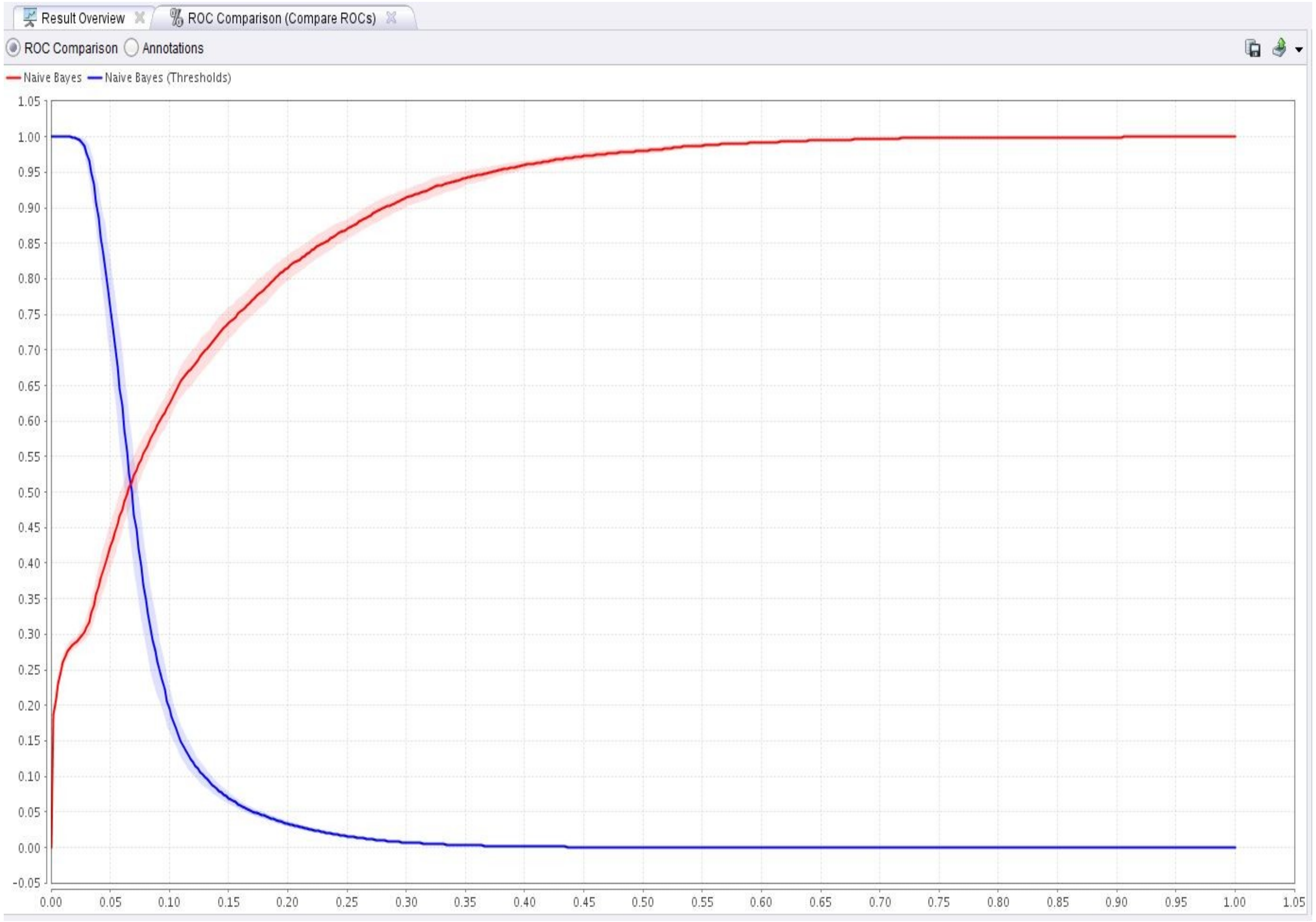
ROC Curves

- Drawing ROC Curves in RapidMiner & Python



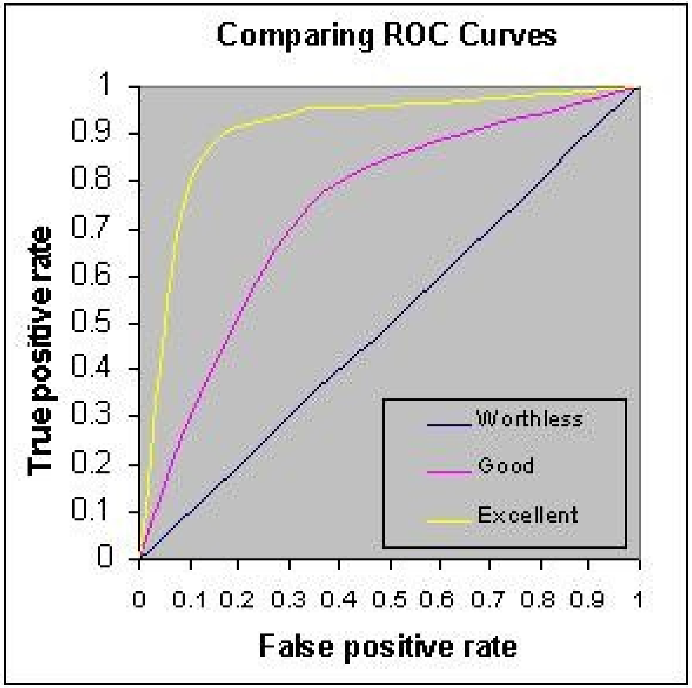

```
fpr, tpr, thresholds = roc_curve(actual, predictions)
plt.plot(fpr, tpr)
```

Example ROC Curve of Naive Bayes



Interpreting ROC Curves

- Best possible result:
  - all correct predictions have higher confidence than all incorrect ones
- The steeper, the better
  - random guessing results in the diagonal
  - so a decent algorithm should result in a curve significantly above the diagonal
- Comparing algorithms:
  - Curve A above curve B means algorithm A better than algorithm B



- Frequently used criterion
  - area under curve (aka ROC AUC)
  - normalized to 1

Questions?