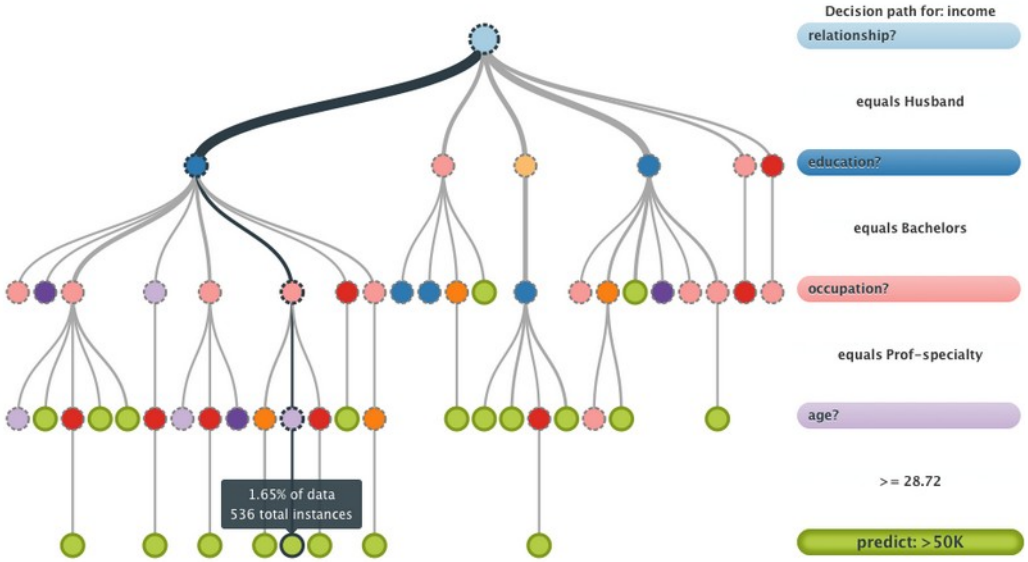
Machine learning

Lecture 4



Marcin Wolter

IFJ PAN

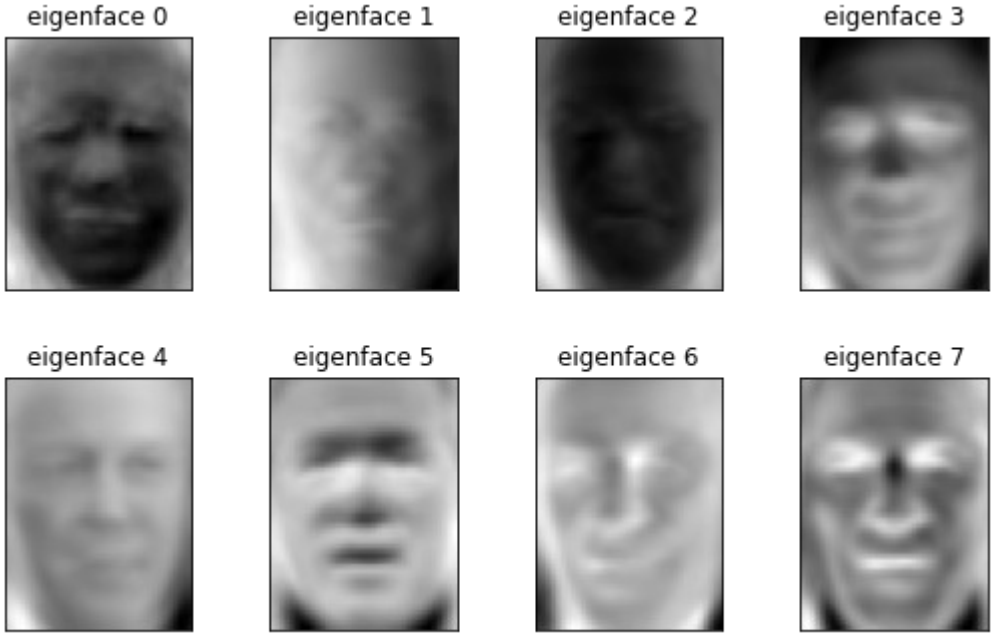
4 November 2020

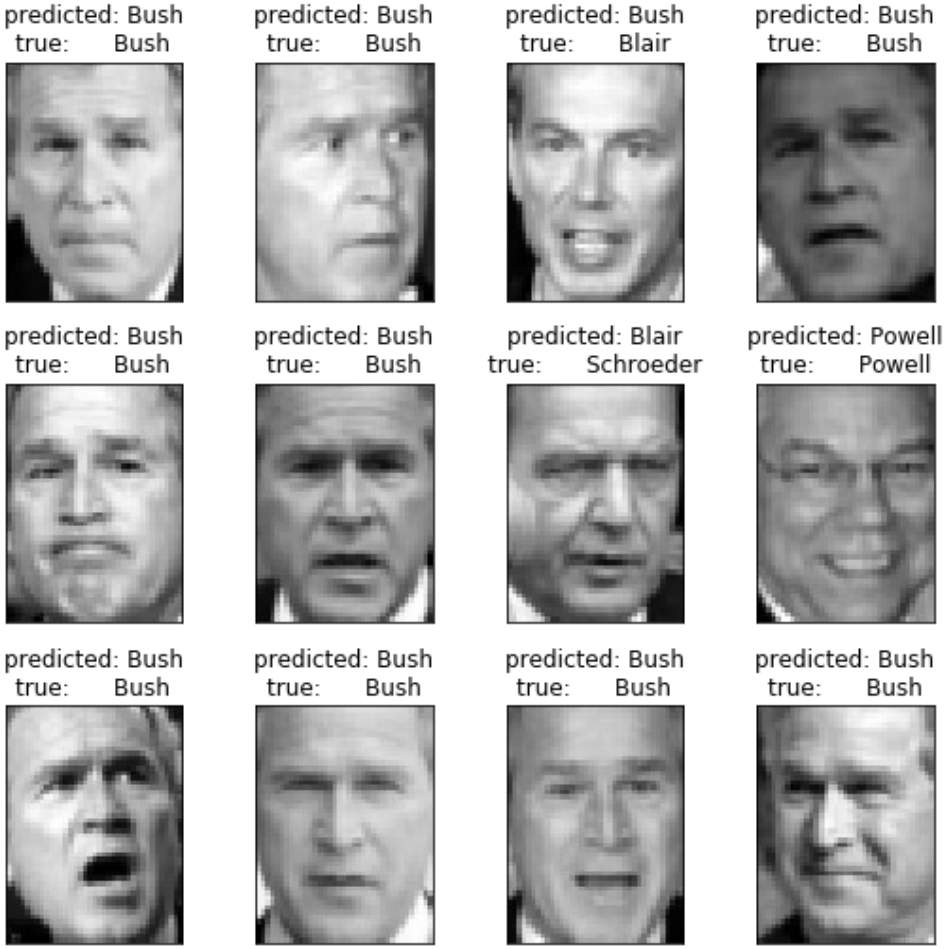
- Simple non-linear methods like k-nearest neighbors and decision trees.
- Independent Component Analysis ICA
- Ensemble learning – Boosted Decision Trees BDT

All slides will be here: https://indico.ifj.edu.pl/event/397/

PCA - classification of faces

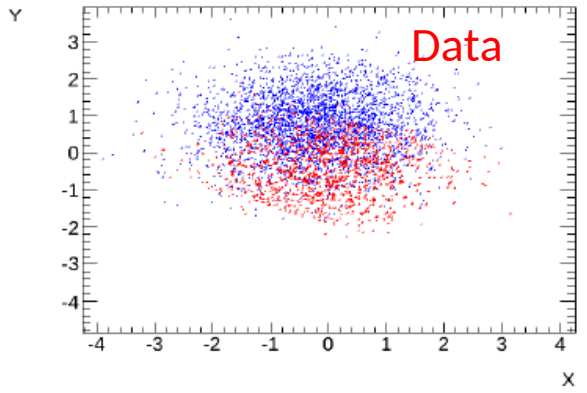
- PCA – each face can be represented as a combination of a limited number of “eigenfaces”
- https://github.com/marcinwolter/MachineLearning2020/blob/main/plot_face_recognition.ipynb

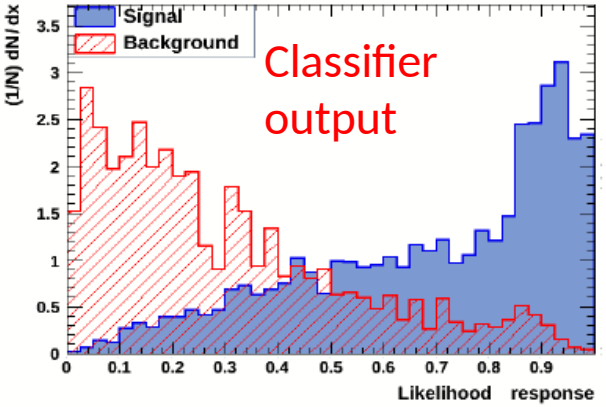


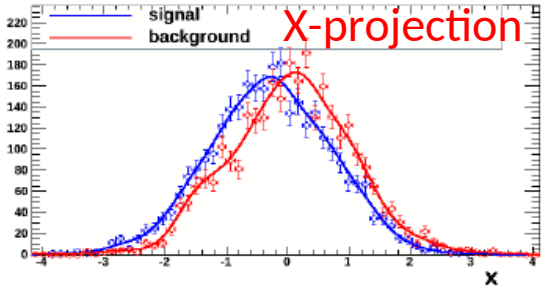


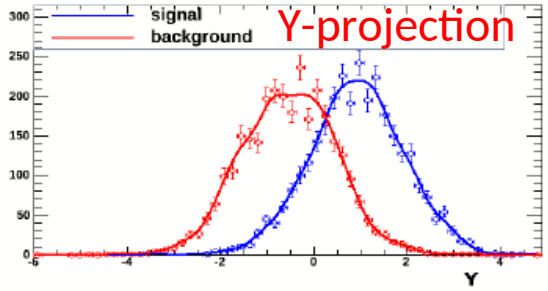
Naive Bayes classifier

(repetition from the previous lecture)









Frequently called **"projected likelihood"** by physicists

- Based on the assumption, that variables are independent (so „naive"):

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

"Naive" assumption: $P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

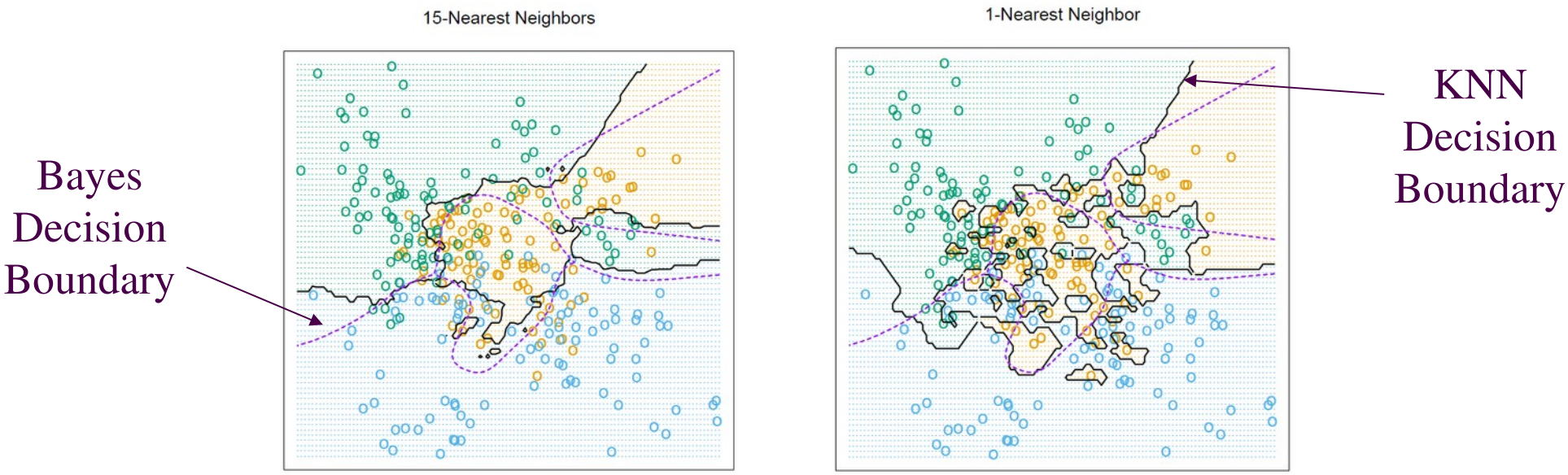
Bayes formula

- Output probability is **a product of probabilities for all variables.**
- Fast and stable, not optimal, but in many cases sufficient.

K-Nearest Neighbors

- The only parameter that can adjust the complexity of KNN is the number of neighbors *k*.

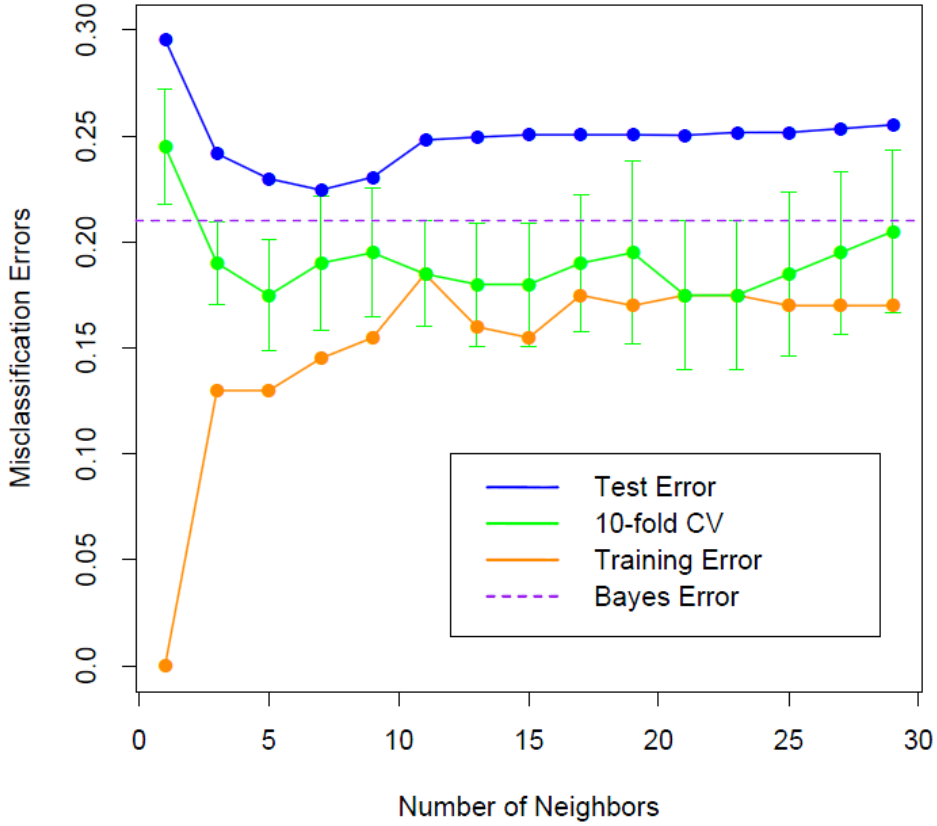
- The larger *k* is, the smoother the classification boundary. Or we can think of the complexity of KNN as lower when *k* increases.



K-Nearest Neighbors

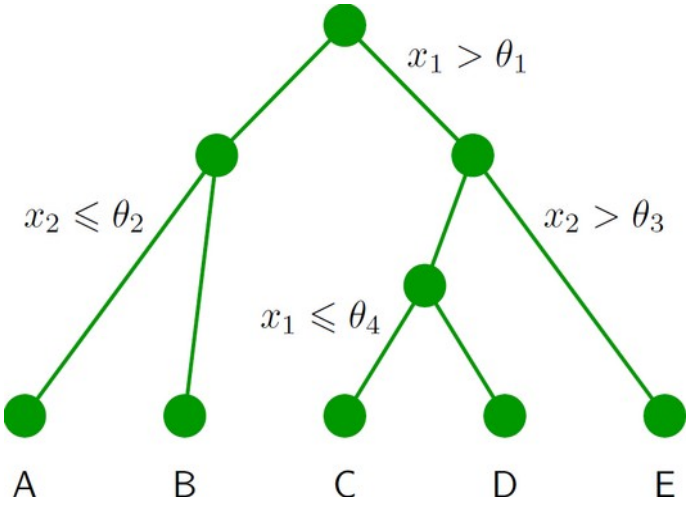
- For another simulated data set, there are two classes. The error rates based on the training data, test data, and *10*-fold cross validation are plotted against *K*, the number of neighbors.

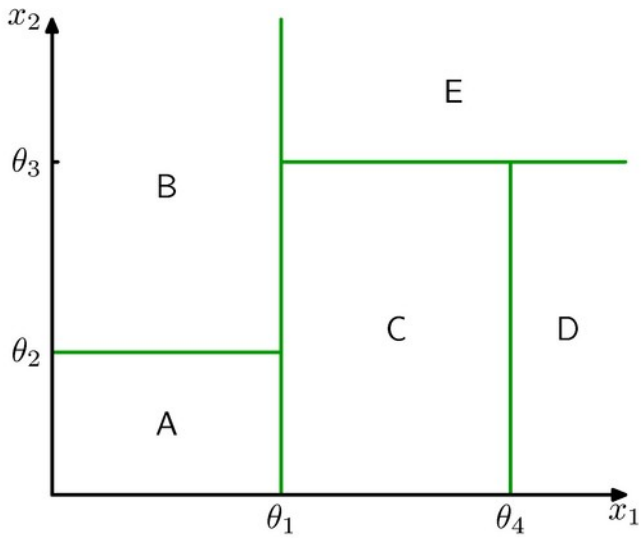
- We can see that the training error rate tends to grow when *k* grows, which is not the case for the error rate based on a separate test data set or cross-validation.



Decision trees

- Decision tree – a series of cuts, each „leaf” (A,B,C,D,E) has a label, for example ”signal” and “background”.

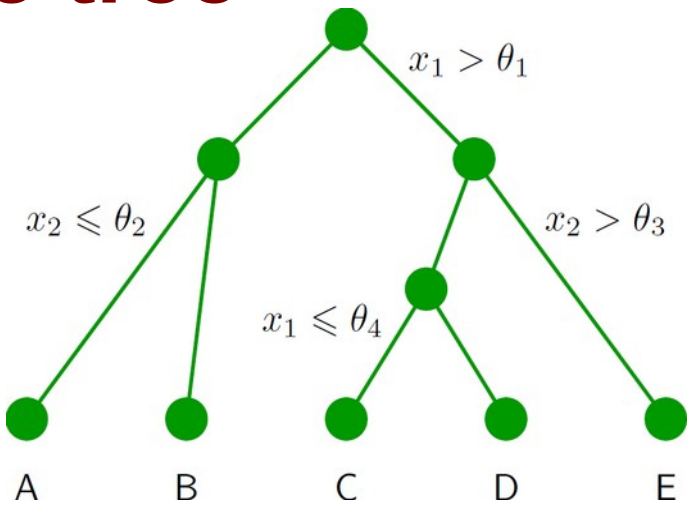




- Easy in visualization and interpretation
- Resistant for *outliers*.
- Weak variables are ignored.
- Fast training and classification.
- Unfortunately: **sensitive for fluctuations, unstable**.

Building the tree

- We start from the root.
- We divide the training sample by the best separating cut on the best variable.
- We repeat the procedure until the stopping conditions are fulfilled, for example: number of leafs, number of events in a leaf etc.
- The ratio S/B in a leaf defines the classification (binary signal or background, or a real number giving the probability, that a given event is a signal).



Definitions of separation:

- Gini inpurity: (*Corrado Gini 1912, invented the Gini index used to measure the inequality of incomes*)

  p (1-p) : p= P(signal), *purity*

- Cross-entropy:

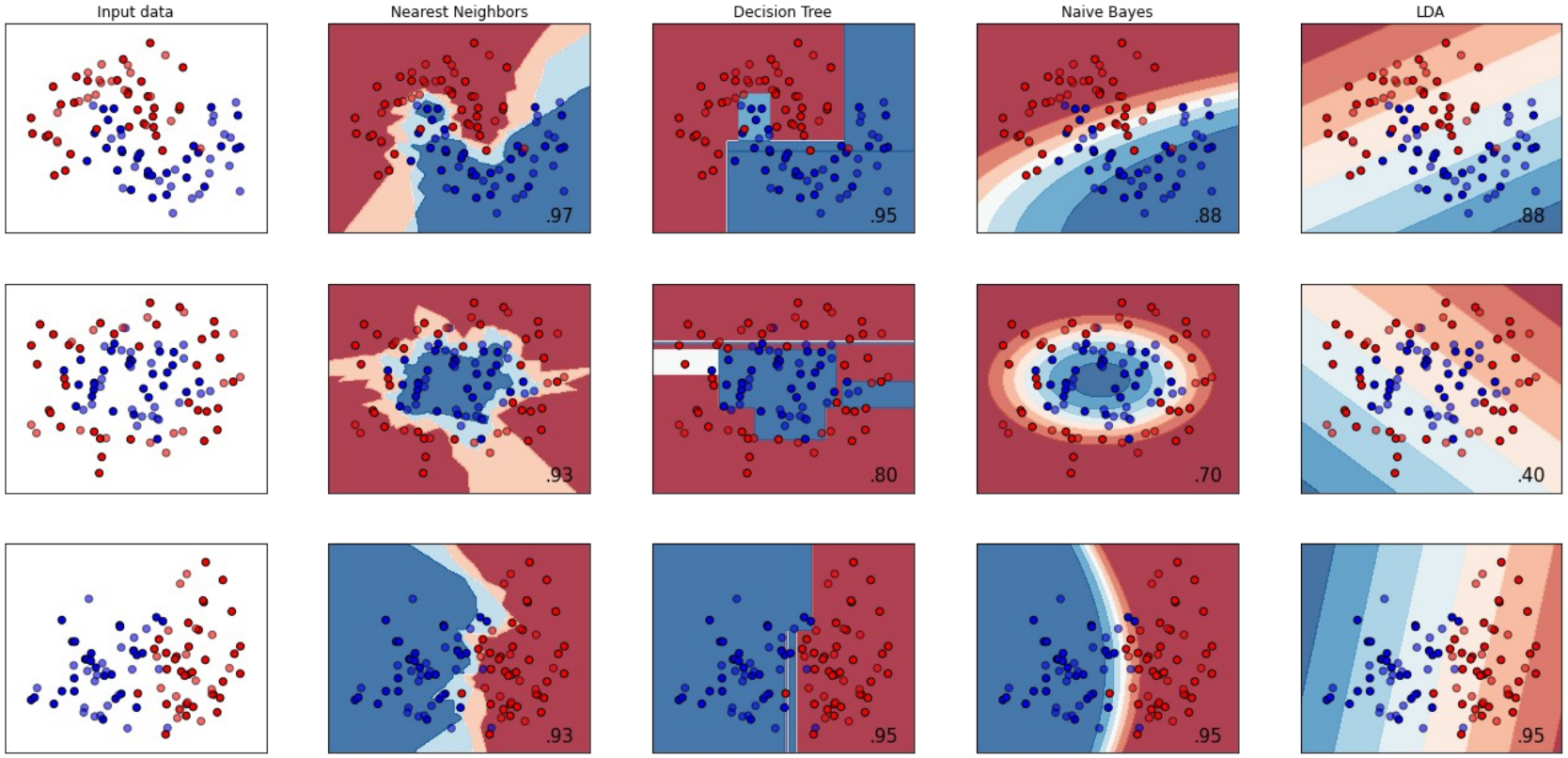
  -(p ln p + (1-p)ln(1-p))

- Missidentification:

  1-max(p,1-p)

Example of simple classifiers

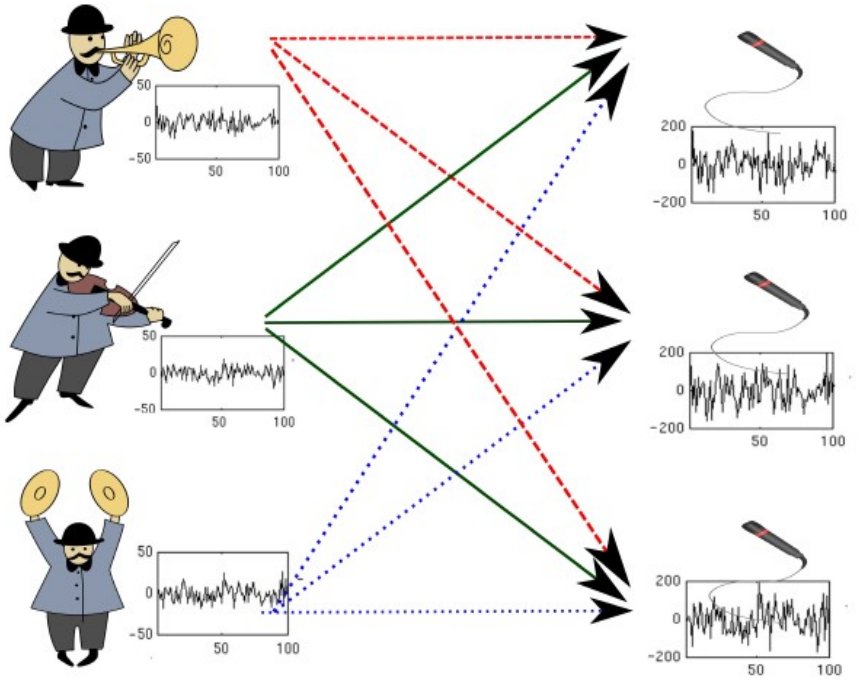
https://github.com/marcinwolter/MachineLearning2020/blob/main/simple_classifier_comparison.ipynb

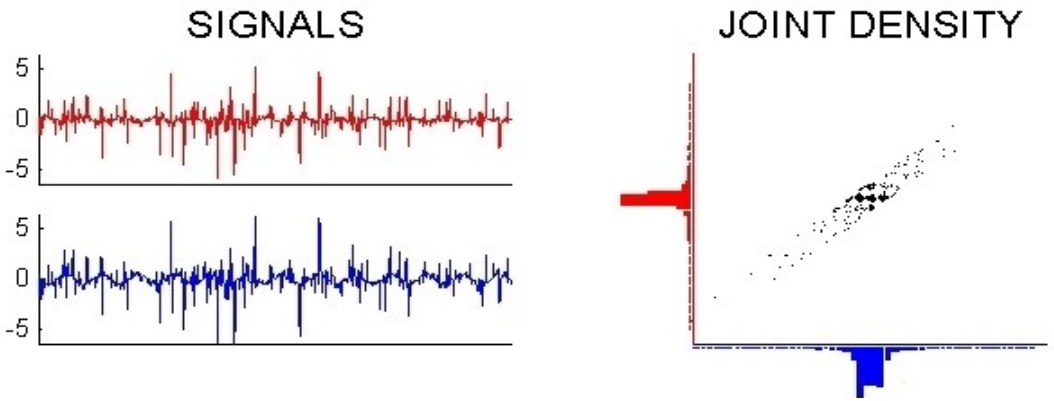


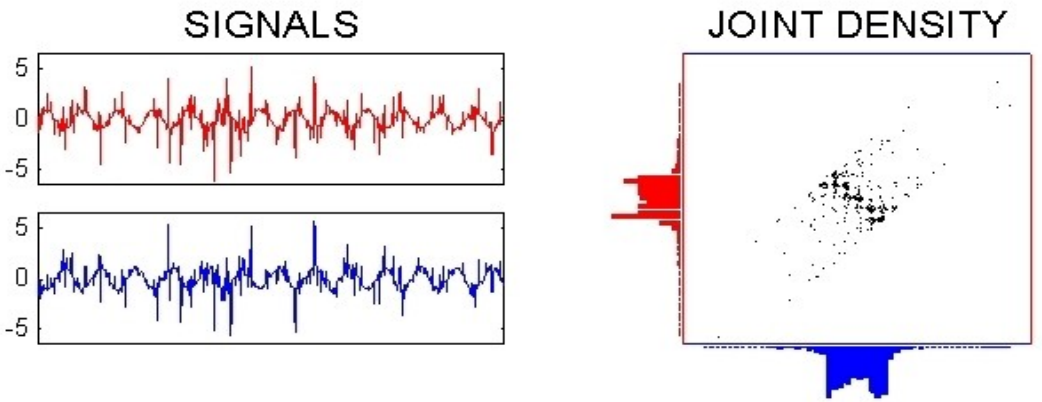
Independent Component Analysis ICA

Developed at Helsinki University of Technology *http://www.cis.hut.fi/projects/ica/*

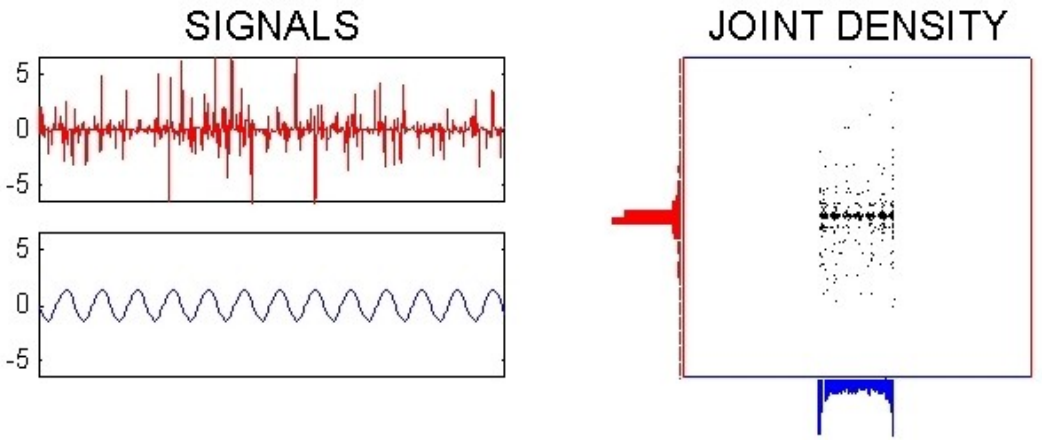
- **Problem:**
  - Assume, that signal **X** is a linear combination **X = AS** of independent sources **S**. The mixing matrix **A** and vector of sources **S** are unknown.
  - **Task:** find a matrix **T (inverted A)**, such that elements of vector **U = TX** are statistically independent. **T** is the matrix returning the original signals.
- **Applications:**
  - Filtering of one source out of many others,
  - Separation of signals in telecommunication,
  - Separation of signals from different regions of brain,
  - Signal separation in astrophysics,
  - Decomposition of signals in accelerator beam analysis in FERMILAB.







Whitened signals and density



Separated signals after 5 steps of FastICA

How does ICA work?

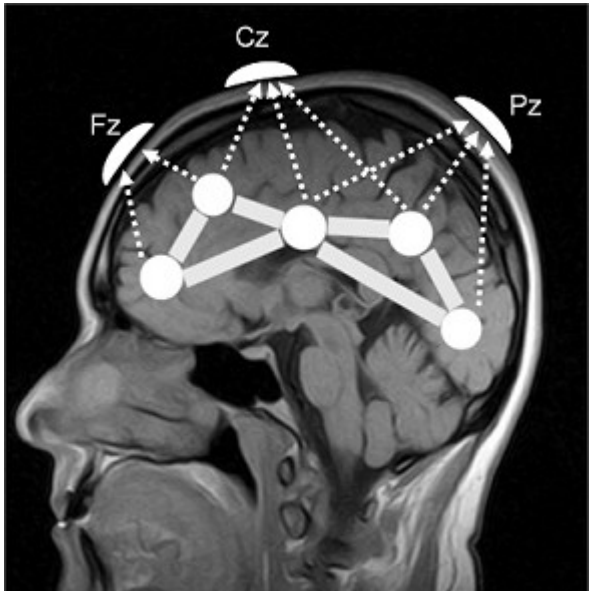
- We have two measured signals and we want to separate them into two independent sources.
- Preparing data - decorrelation (correlation coefficients equal zero, σ=1).

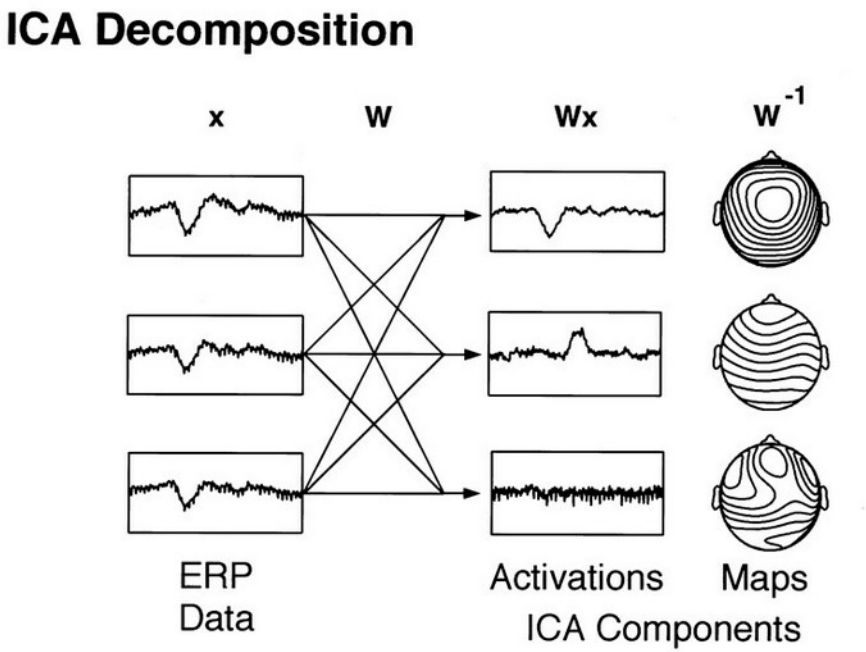
Superposition of many independent distributions gives Gaussian in the limit.

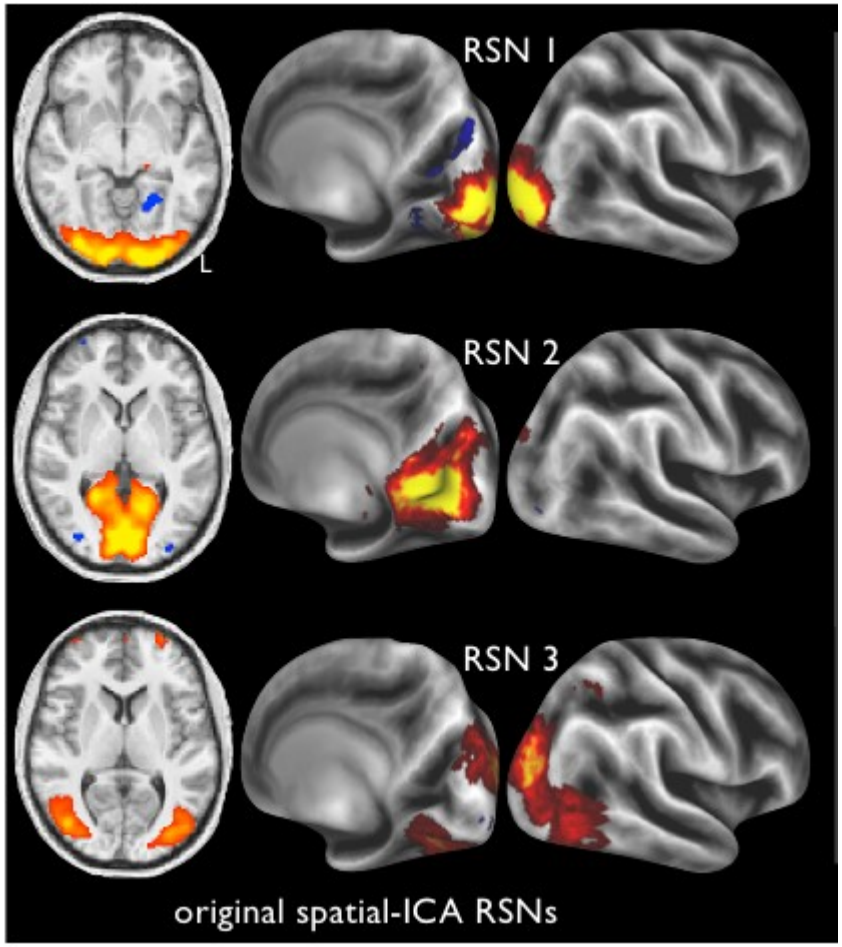
- ICA – rotation, signals should be maximally non-Gaussian (measure of non-Gaussianity might be curtosis).
- *Curtosis:* $\mathrm{Kurt} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^4}{\sigma^4} - 3$

where μ is the mean of the distribution and σ is a standard deviation.

ICA – brain research, signal separation



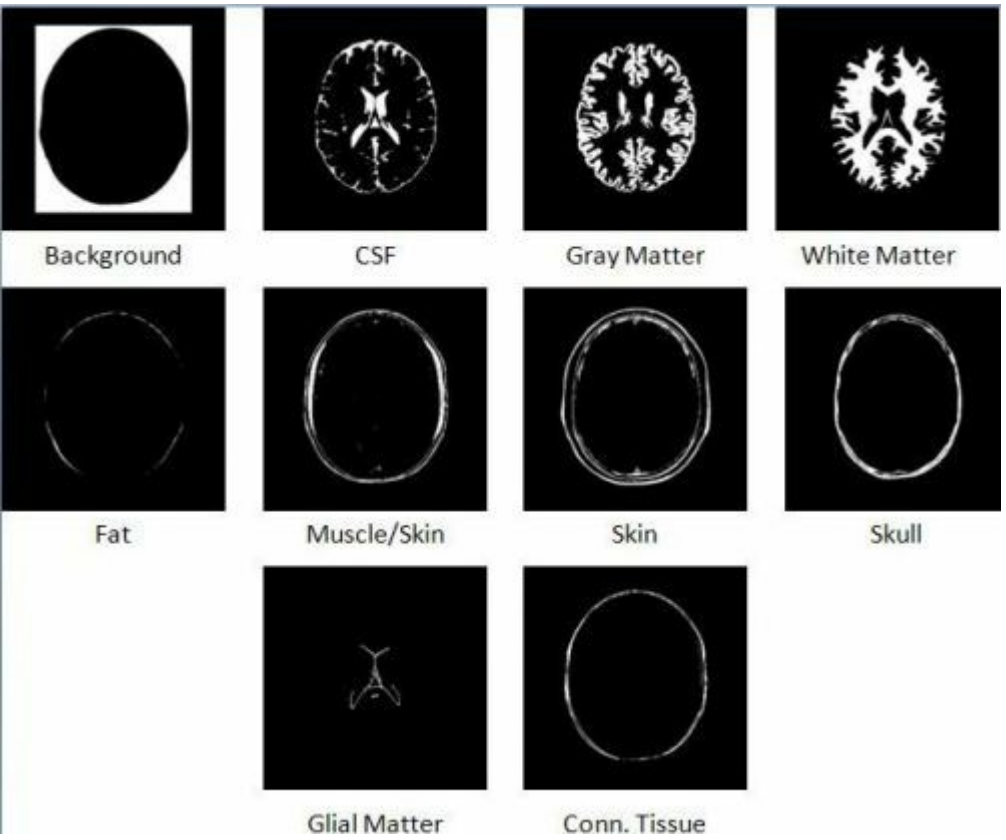




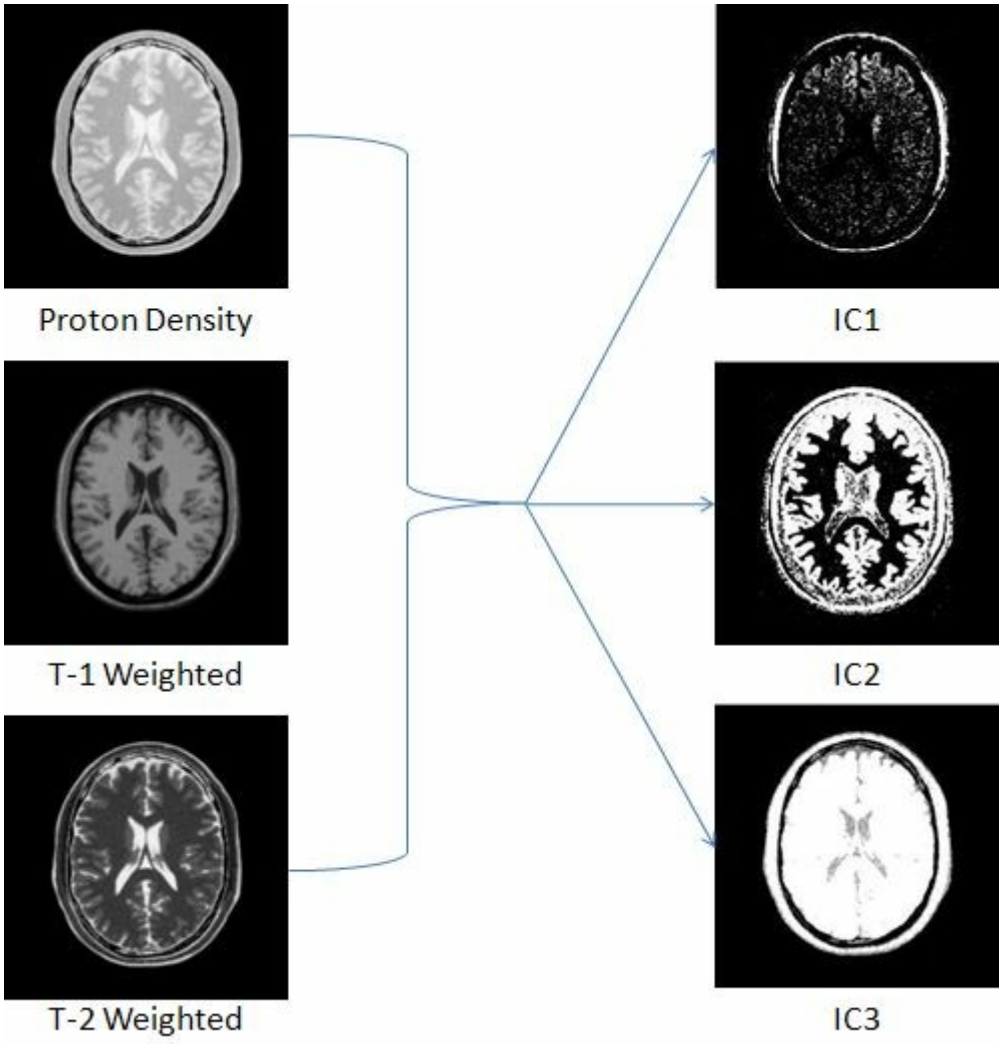
3 components from 21-dimensional decomposition using the “spatial-ICA'' algorithm.

PNAS February 21, 2012 vol. 109 no. 8 3131-3136

ICA and magnetic resonance



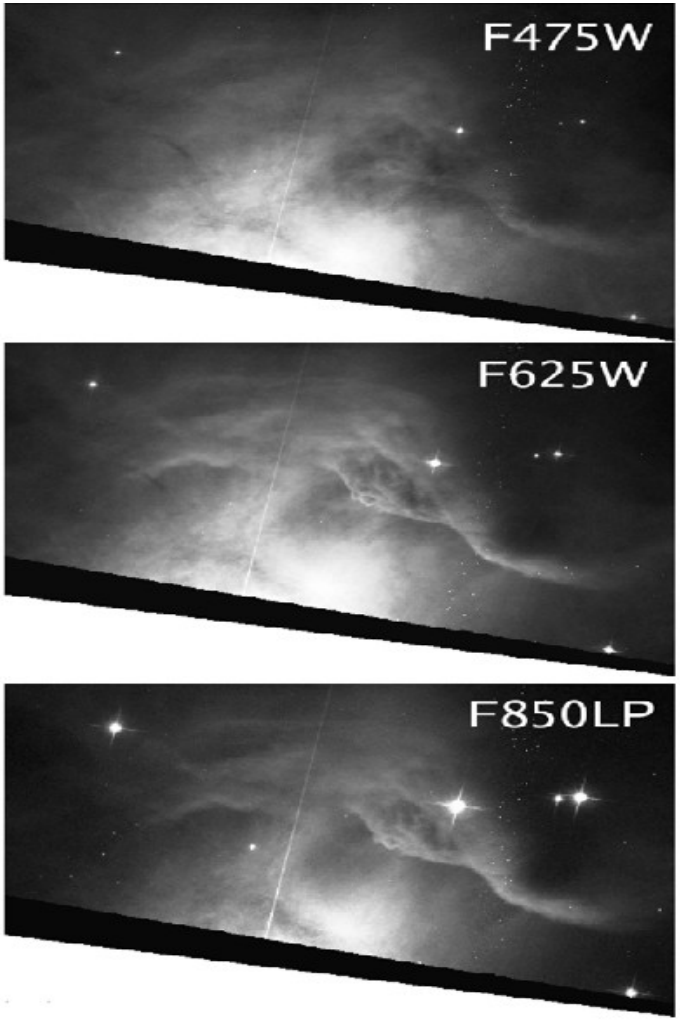
Sources of signals

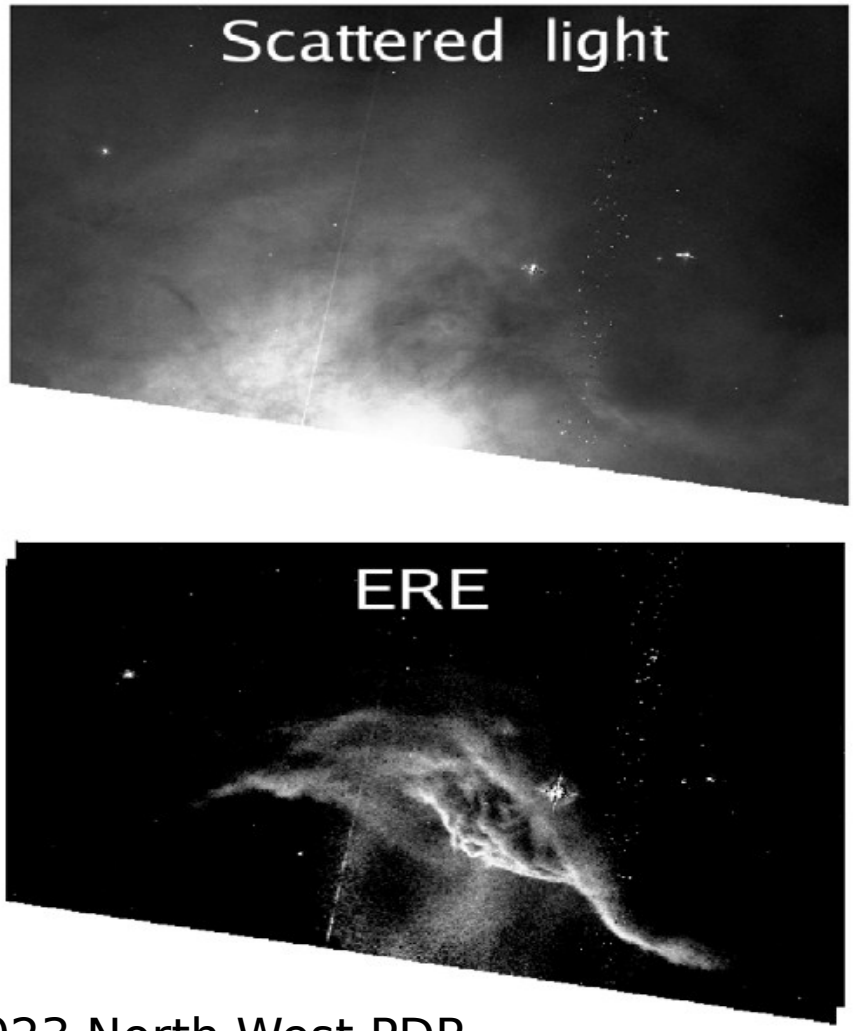


Measured **Separated components**

Blind Source Separation in Magnetic Resonance Images January 30, 2010 by Shubhendu Trivedi

ICA – astronomy





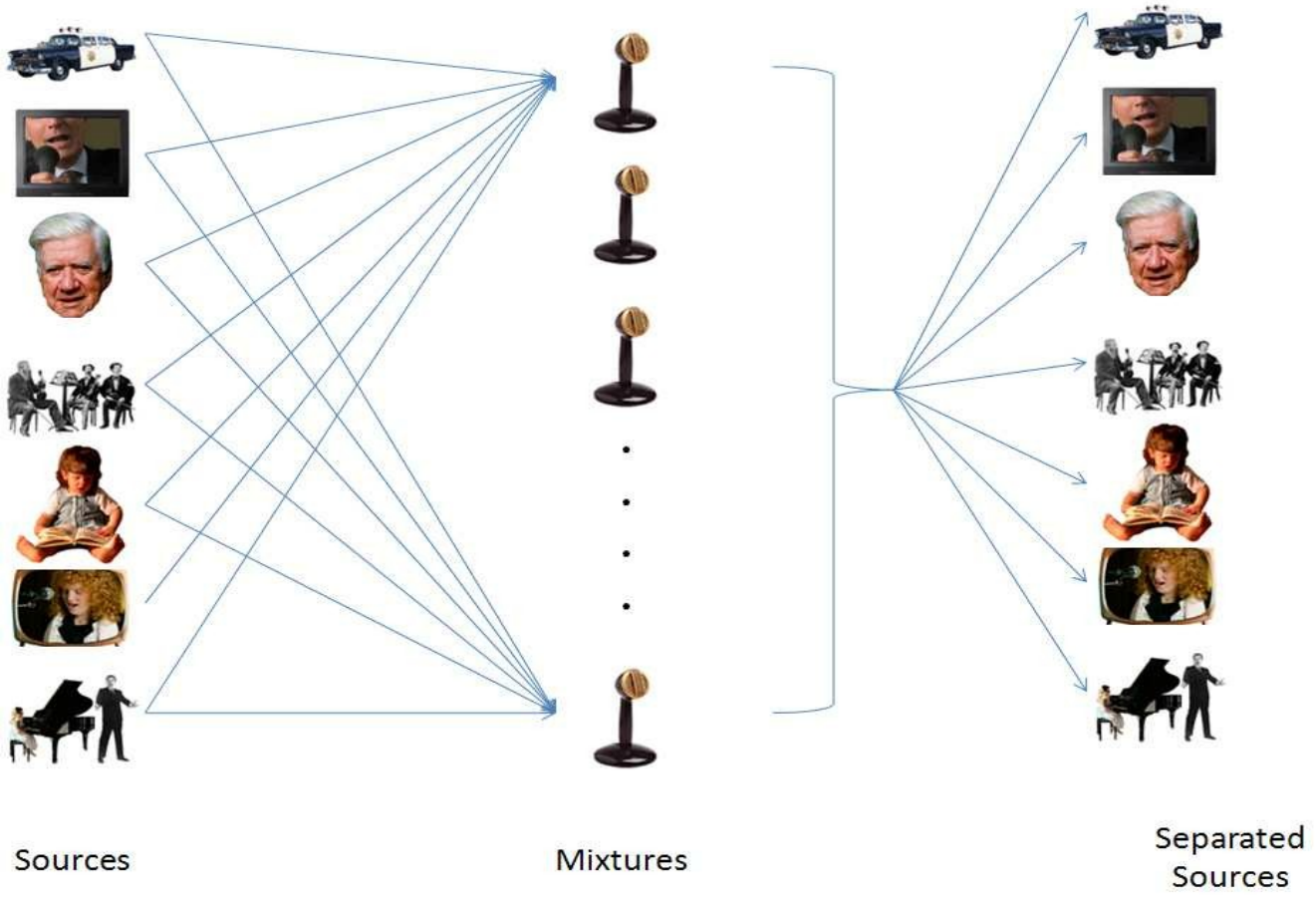
On the left: HST images of the NGC 7023 North-West PDR in three SDSS wide-band filters. On the right: scattered light and ERE (Extended Red Emission) images extracted with FastICA from the observations.

A&A 479, L41-L44 (2008)
DOI: 10.1051/0004-6361:20079158

ICA example – cocktail party

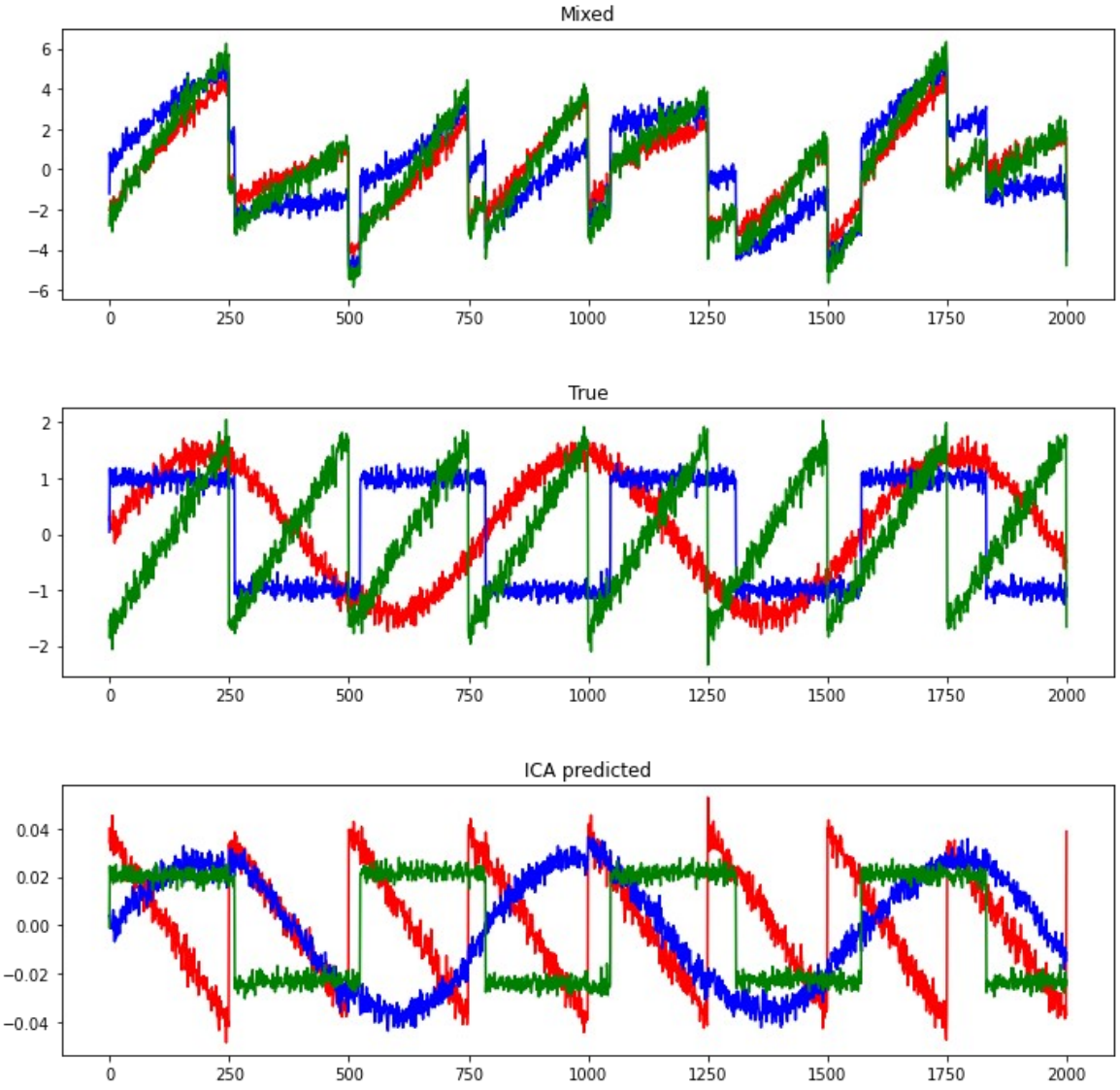
http://cnl.salk.edu/~tewon/Blind/blind_audio.html

- Cocktail Party Demo - applet showing how the the ICA algorithm works – blind separation of sound sources.



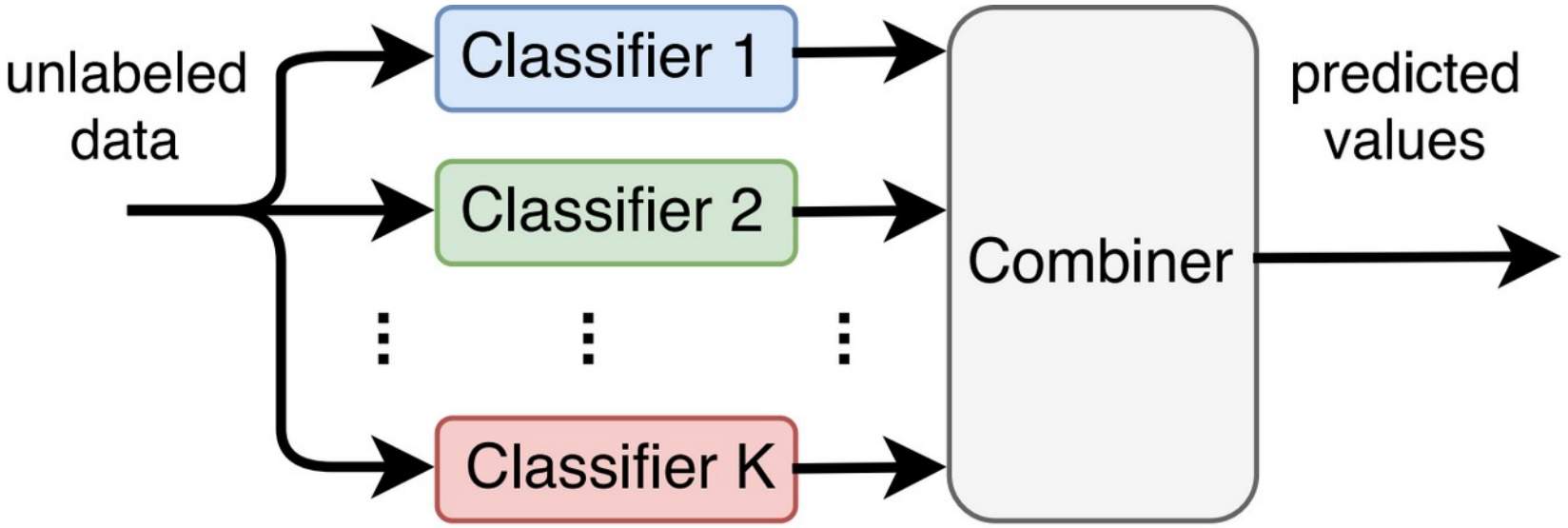
ICA example

- https://github.com/marcinwolter/MachineLearning2020/blob/main/ICA_sklearn.ipynb



Boosting, bagging, BDT…

Ensemble learning



What does BDT mean???

- **BDT – Boosted Decision Tree:**
  - **Decision Tree** – an algorithm know for a long time, used in most of the expert systems. For example, the first aid manual is frequently written in the form of a decision tree: if (condition) do something, else do something different, then check another condition...
  - **Boosted** - a method of joining many weak classifiers in an ensemble to get one strong classifiers. It is not limited to decision trees, however with them it is most commonly used.

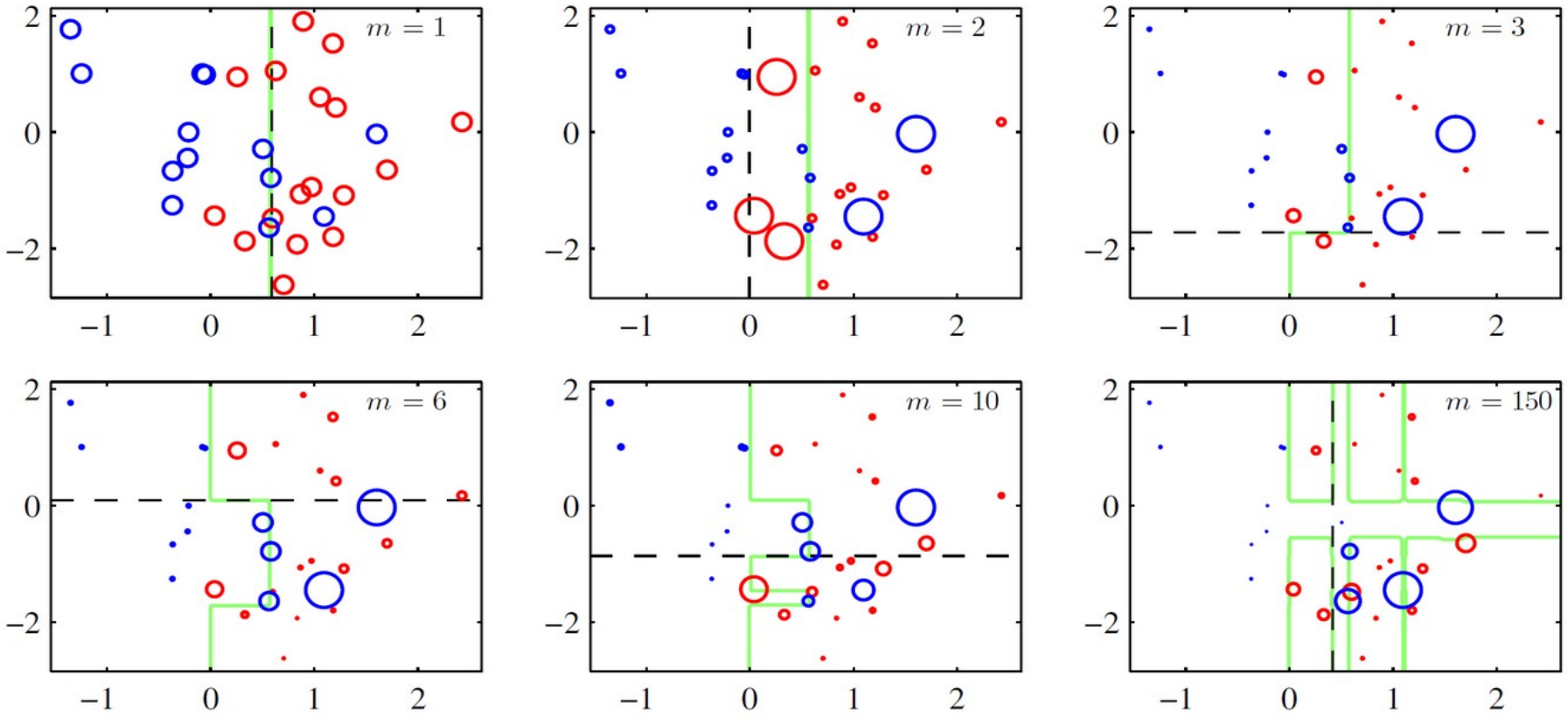
AdaBoost – ensemble of classifiers

Problem: could we improve a weak classifier?

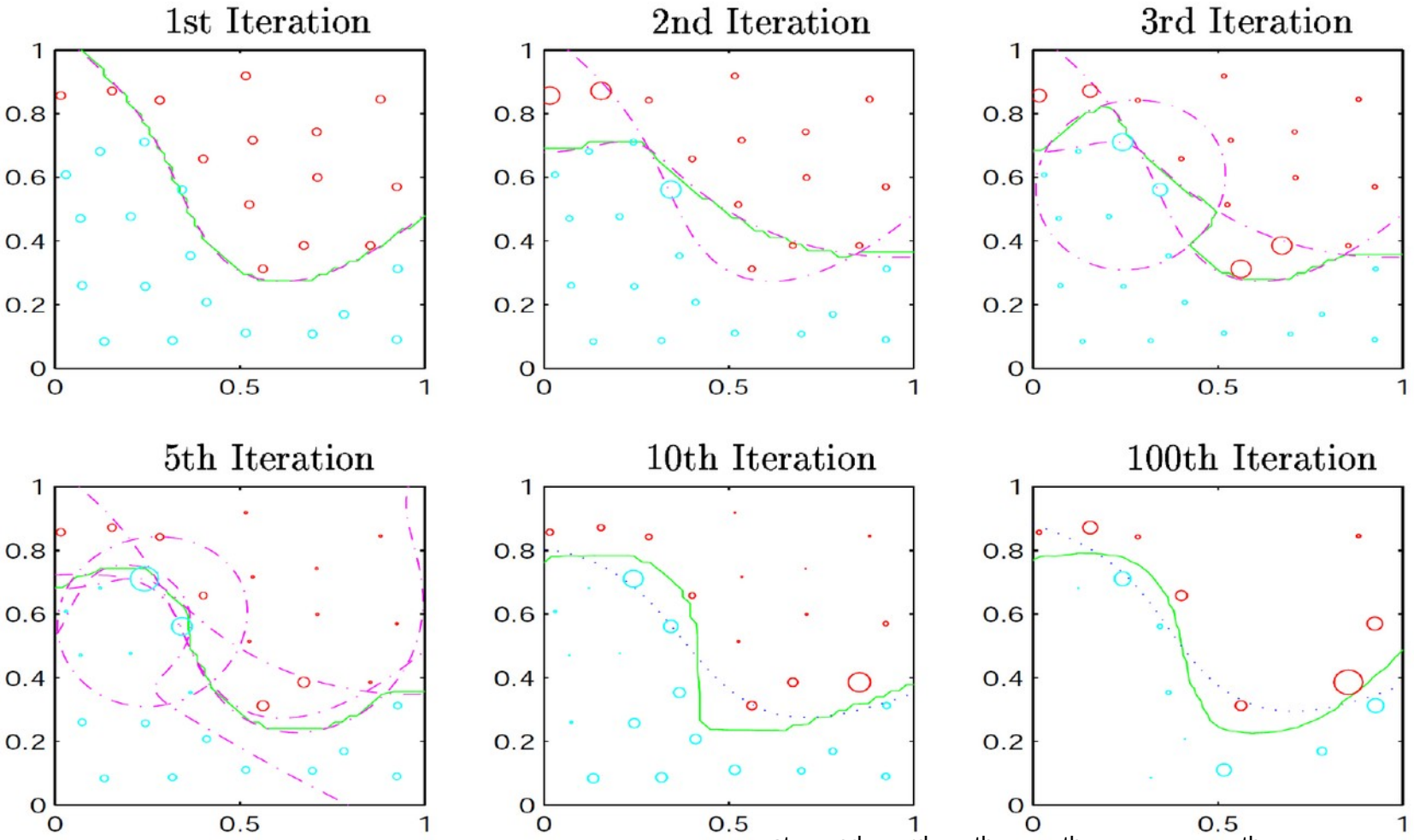
Answer: yes, by applying it many times.

- An algorithm used most frequently: **AdaBoost** (Freund & Schapire 1996 – Gödel prize)

- Build a decision tree.
- Increase the weights of wrongly classified events.
- Repeat many times (typically 100-1000)
- Classify the events by “voting” of all the trees.



AdaBoost



AdaBoost for 2-dimensional data – results after 1st, 2nd, 3rd, 5th, 10th and 100th iteration. The solid green line shows the results of the combined classifier, the dashed line the borders between classes obtained after each step. For the last two plots the dotted line marks the class borders obtained from the bagging algorithm (we will talk about bagging soon).

AdaBoost

- Five years after publication of AdaBoost Friedman proved, that the algorithm in fact minimizes the exponential loss function:

$$E = \sum_{n=1}^{N} \exp\left(-t_n f_m(x_n)\right)$$

where *f(x)* is the answer of the algorithm

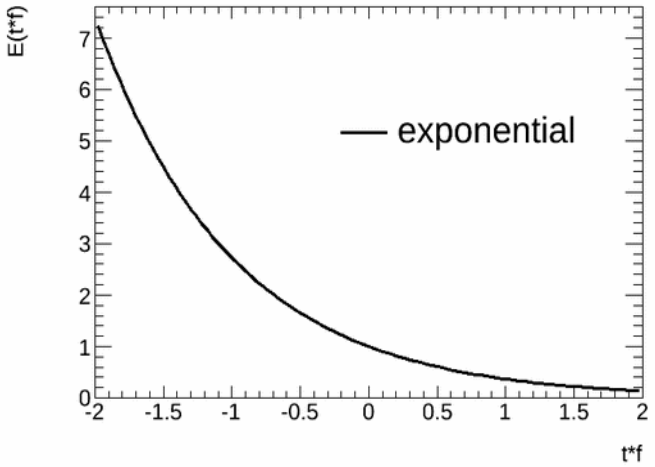
t = 1 signal, *t = -1* background

$t_n \cdot f_m(x_n) > 0$ – correctly classified

$t_n \cdot f_m(x_n) < 0$ – incorrectly classified

- Exponential function goes up quickly => huge punishment for wrongly classified events, so the algorithm is sensitive for single, outstanding points. Classification becomes worse, when data are hardly separable.

  - Another loss function?
  - Friedman in year 2000 proposed few other loss functions, but AdaBoost is still the most popular (we will talk about).



AdaBoost in action

AdaBoost algorithm in detail

1. Give all vectors from the training set a weight w_i=1/N.

2. For m = 1,...,M:

a) Train classifiers $y_m(x)$ on the training sample minimizing:

$$J_m = \sum_{n=1}^{N} w_n^m I\left(y_m(x_n) \neq t_n\right)$$

b) Evaluate the quantities:

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^m I\left(y_m(x_n) \neq t_n\right)}{\sum_{n=1}^{N} w_n^m}$$

$$\alpha_m = \frac{1}{2} \ln \frac{1-\epsilon_m}{\epsilon_m}$$

c) Update the weights of the vectors in the training sample:

$$w_n^{m+1} = \begin{cases} \frac{w_n^m}{Z_m} e^{\alpha_m} & y_m(x_n) \neq t_n \\ \frac{w_n^m}{Z_m} e^{-\alpha_m} & y_m(x_n) = t_n \end{cases}$$

3.The result of voting is given as:

$$Y_M(\mathbf{x}) = sign\left[\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right]$$

All the classifiers are trained on the same training sample, but with different weights. These weights depend on the results of the preceding training, so it's difficult to train many classifiers in parallel.

AdaBoost

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \tag{14.15}$$

   where $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$ and 0 otherwise.

   (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{14.16}$$

   and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}. \tag{14.17}$$

   (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\} \tag{14.18}$$

3. Make predictions using the final model, which is given by

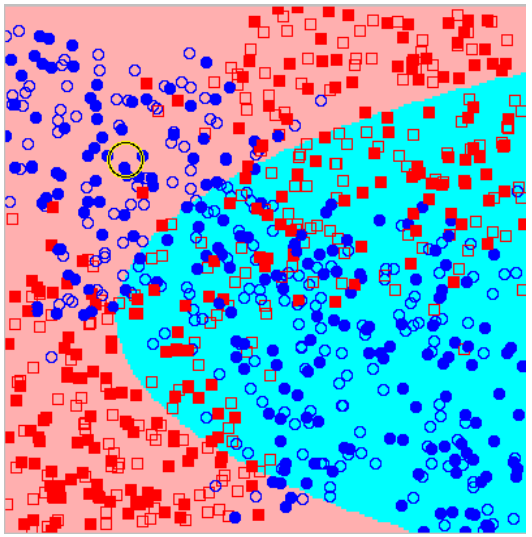
$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) \right). \tag{14.19}$$

C. Bishop, Pattern recognition and machine learning. Springer, 2009.

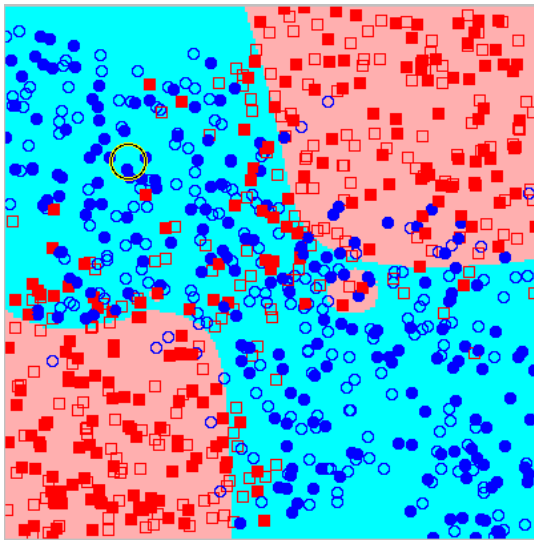
Classifier boosting

- Boosting, bagging – in a “magical” way we get a strong classifier out of weak ones
- **Typically boosting used on decision trees – Boosted Decision Trees BDT.**
- Good results without time consuming tuning of parameters:

  „*the best out-of-box classification algorithm*”.
- Relatively resistant on overtraining.
- Quite frequently used nowadays. And with good results!



Naive Bayes classifier

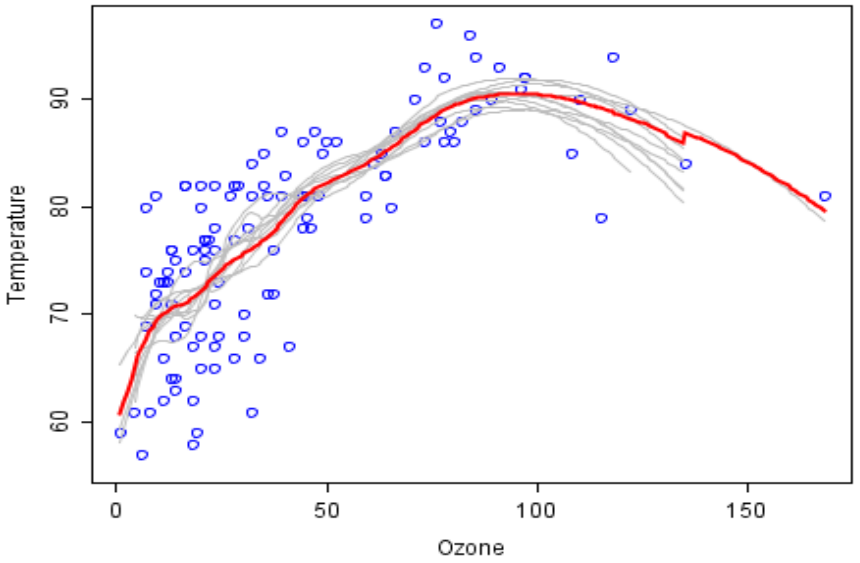


Boosted Naive Bayes classifier

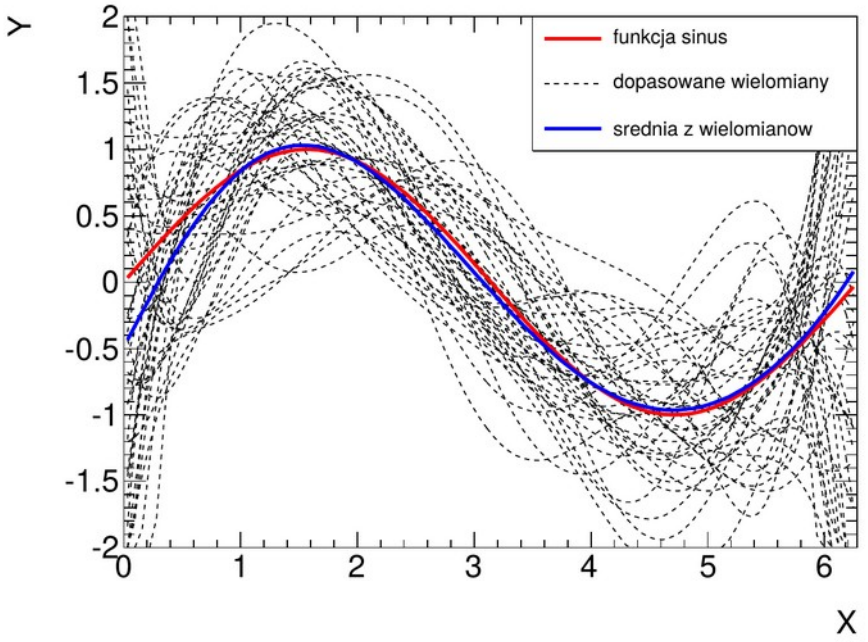
Application of a boosting algorithm (5 iterations) to the Naive Bayes classifier.

Bagging (Bootstrap AGGregatING)

- **Algorithm proposed by Leo Breiman in 1994:**
  - Take N events from N-element training set, but with repetitions.
  - Train the classifier on this set.
  - Repeat many times
  - Classify new events by voting of all the classifiers.

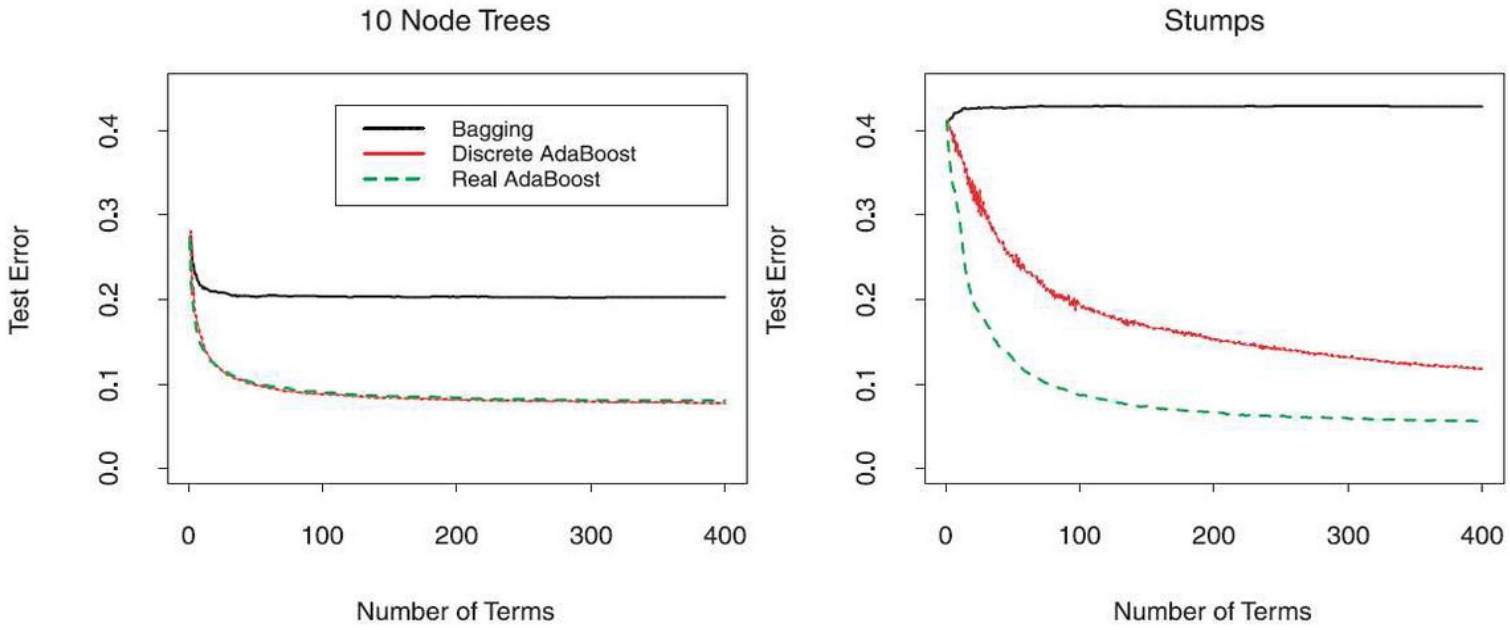


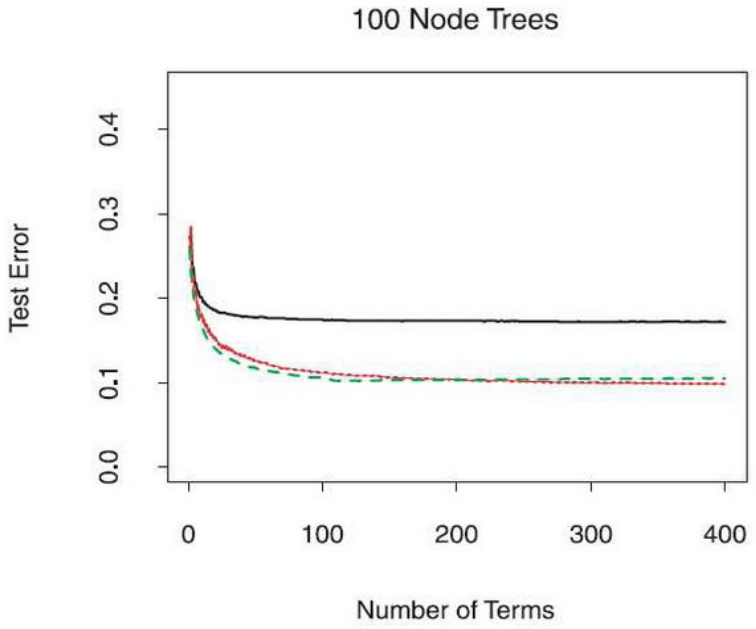
For this data the red line (mean of 100 classifiers) is smoother, more stable and more resistant for overtraining the any of the single classifiers.



Analogy: mean of many poorly fitting functions gives a good fitting function.

Bagging vs. boosting



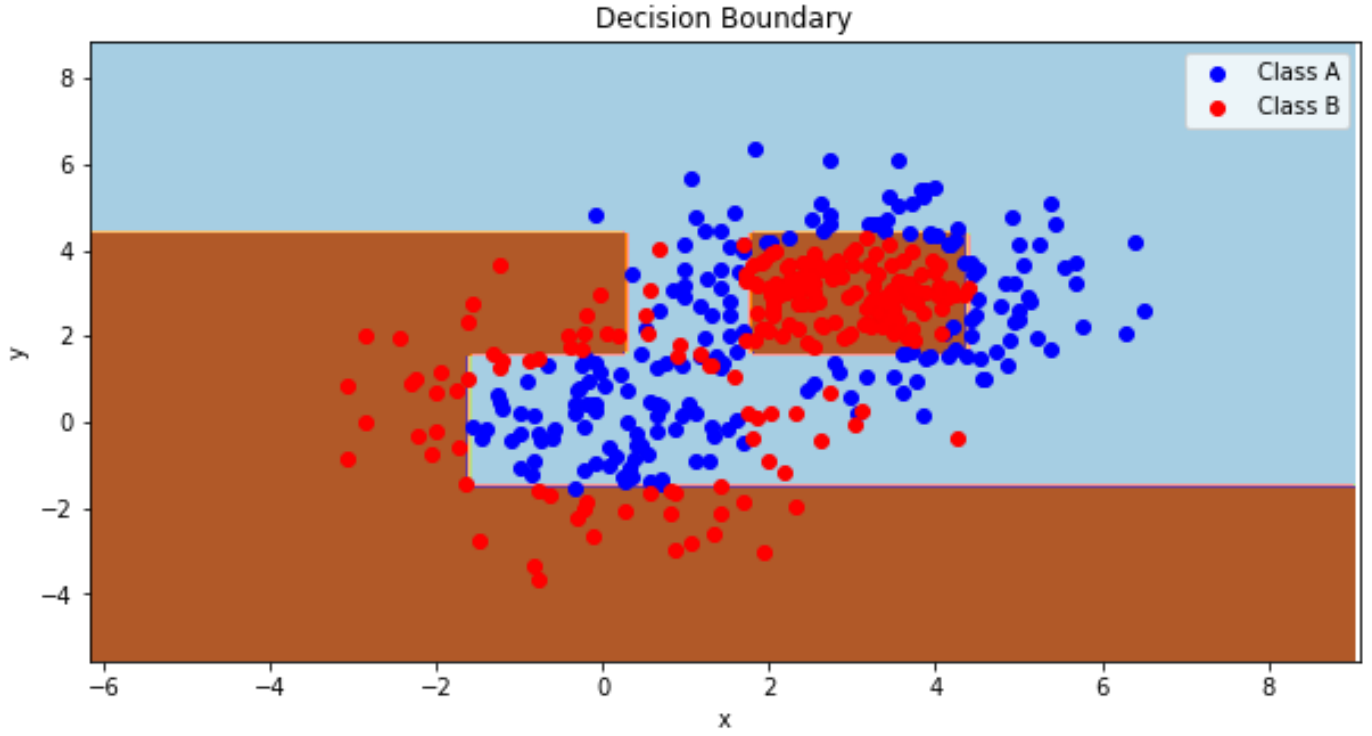


Test error for Bagging, Discrete AdaBoost and Real AdaBoost on a simulated two-class nested spheres problem. There are 2000 training data points in ten dimensions.
[Friedman 2000].

Ensemble Learning example

Example code to run various classification tasks, also with ensemble learning:

https://github.com/marcinwolter/MachineLearning2020/blob/main/EnsambleLearningExample.ipynb



Summary

- Boosting, bagging – in a magic way we can build a strong classifier out of weak ones.
- Commonly used for decision trees, because they are simple and fast.
- Gives good results:

  - „the best out-of-box classification algorithm”.
- Very popular