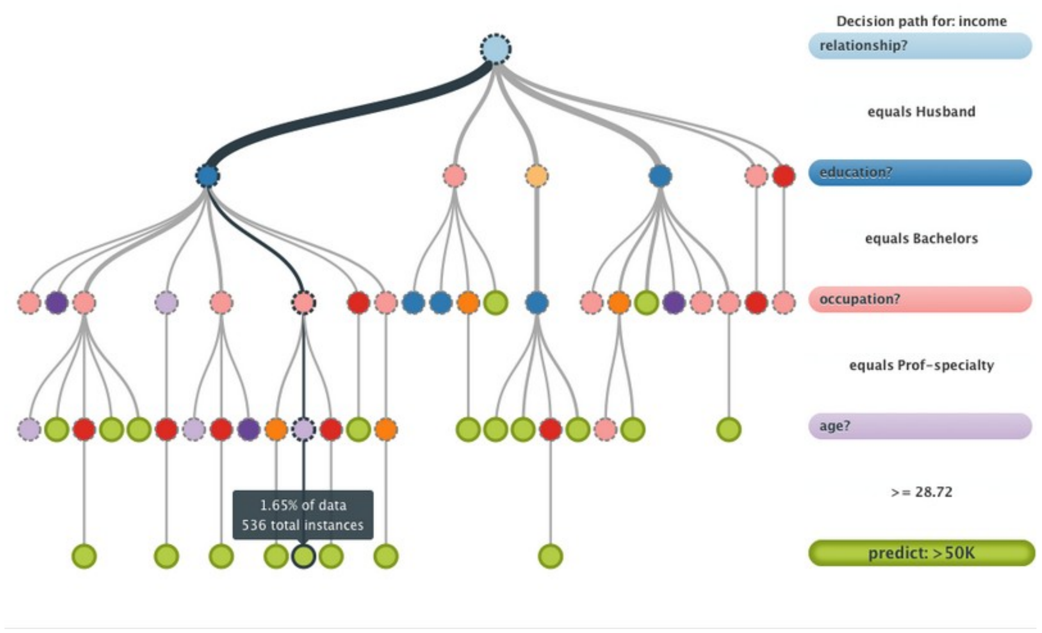
Machine learning
Lecture 2



Marcin Wolter

IFJ PAN

21 October 2020

- Decorrelation and Principal Component Analysis (PCA)
- Overtraining
- Simple non-linear methods like k-nearest neighbors and decision trees.

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

Glen Cowan book and slides on statistics

- Fantastic Glen Cowan's courses on statistics can be found here:
  - http://www.pp.rhul.ac.uk/~cowan/stat_cern.html
  - https://www.pp.rhul.ac.uk/~cowan/stat_course.html
- and his book can be found here:
  - http://www.sherrytowers.com/cowan_statistical_data_analysis.pdf

Online python course

If you want to take one of the online python programming or machine learning courses on the https://www.datacamp.com/ platform Leszek Grzanka leszek.grzanka@ifj.edu.pl can give you a free 6 months access to this e-learning platform.
Just write him an e-mail!

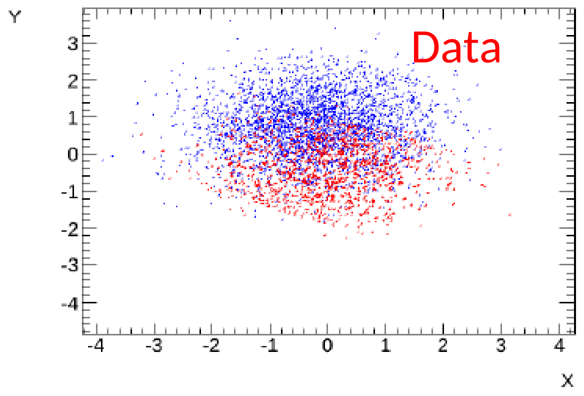
THE SMARTEST WAY TO

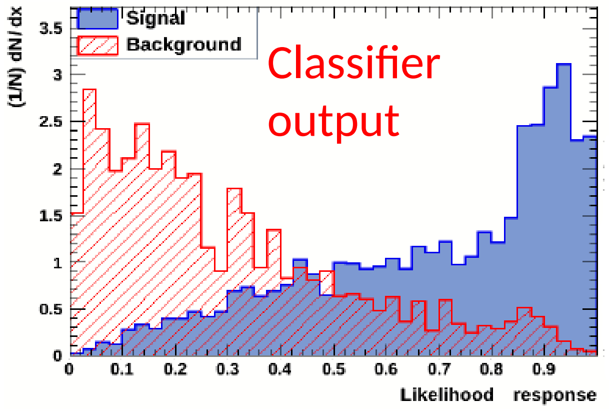
Learn Data Science Online

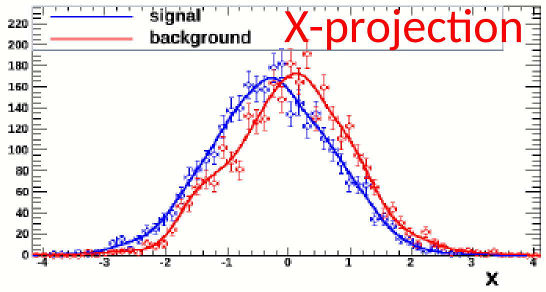
The skills people and businesses need to succeed are changing. No matter where you are in your career or what field you work in, you will need to understand the language of data. With DataCamp, you learn data science today and apply it tomorrow.

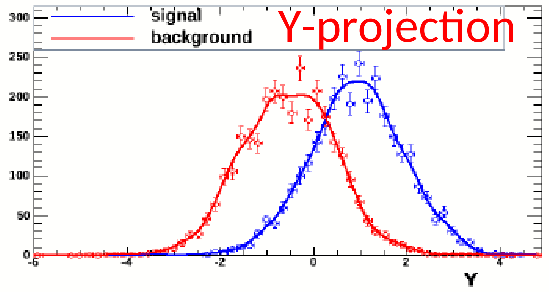
Naive Bayes classifier

(repetition from the previous lecture)









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

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

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

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

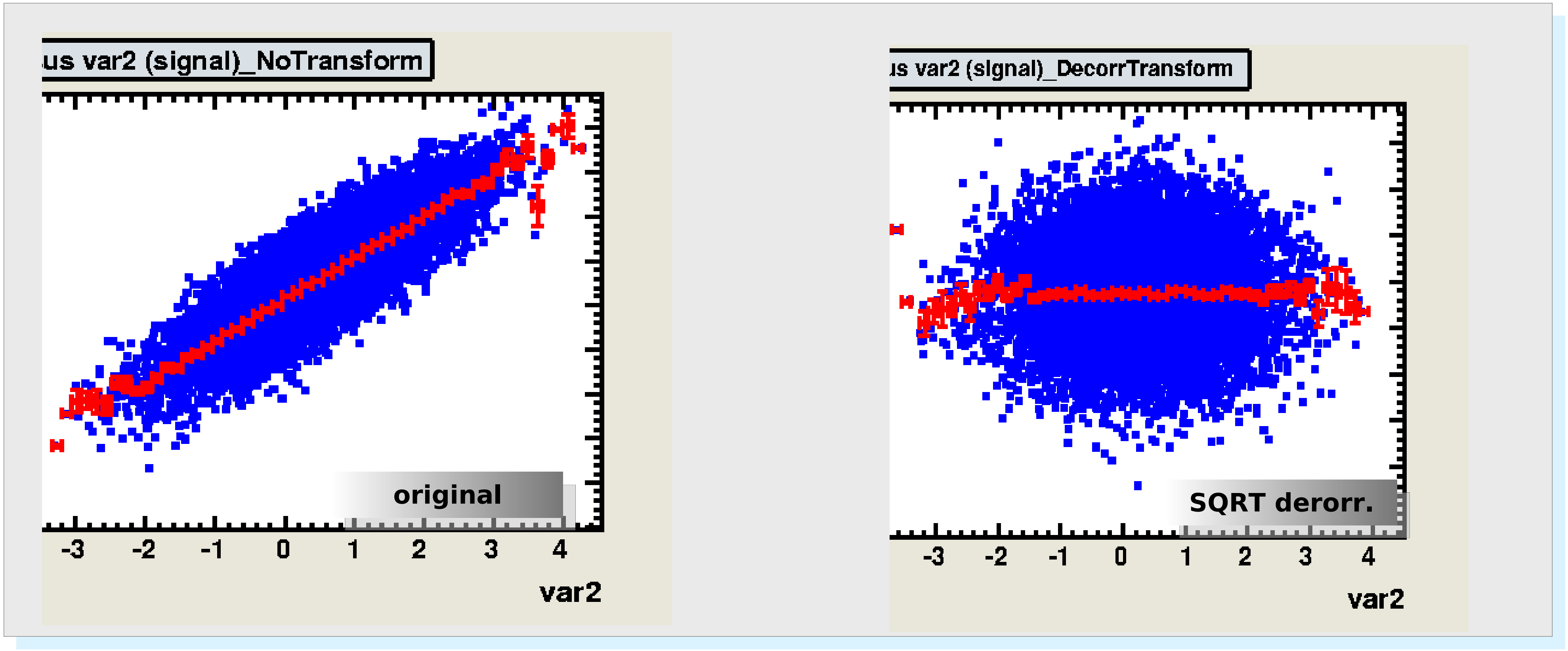
Bayes formula

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

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

Decorrelation

- **Removes correlation between variables by a rotation in the space of variables**

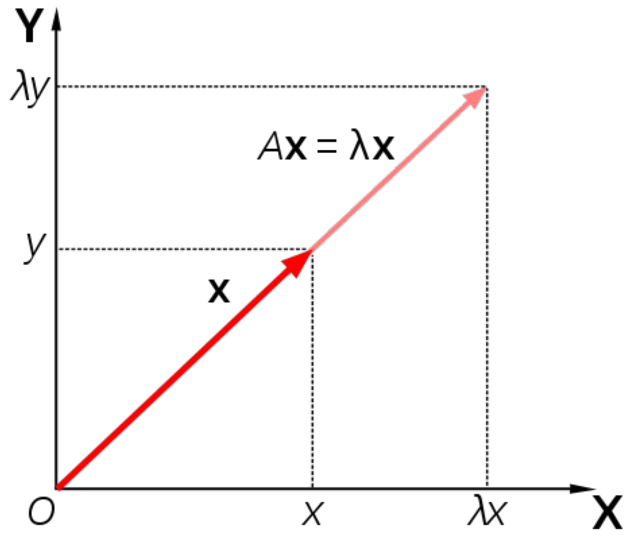


Eigenvalues and eigenvectors

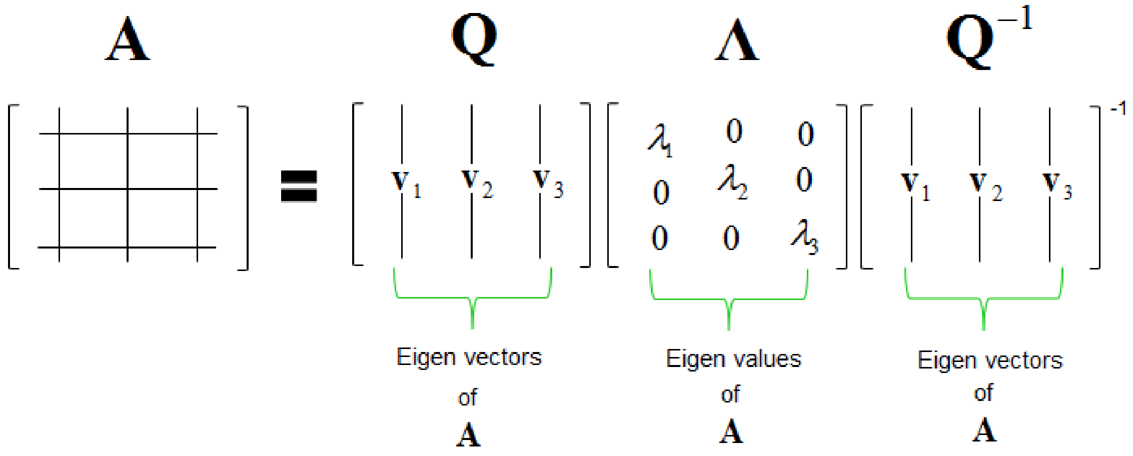
In essence, an eigenvector v of a linear transformation A is a non-zero vector that, when A is applied to it, does not change direction. Applying A to the eigenvector only scales the eigenvector by the scalar value λ, called an eigenvalue. This condition can be written as the equation

$$\mathbf{A}(\mathbf{v}) = \lambda\,\mathbf{v}$$

referred to as the eigenvalue equation or eigenequation. In general, λ may be any scalar. For example, λ may be negative, in which case the eigenvector reverses direction as part of the scaling, or it may be zero or even complex.



Matrix A acts by stretching the vector x, not changing its direction, so x is an eigenvector of A.



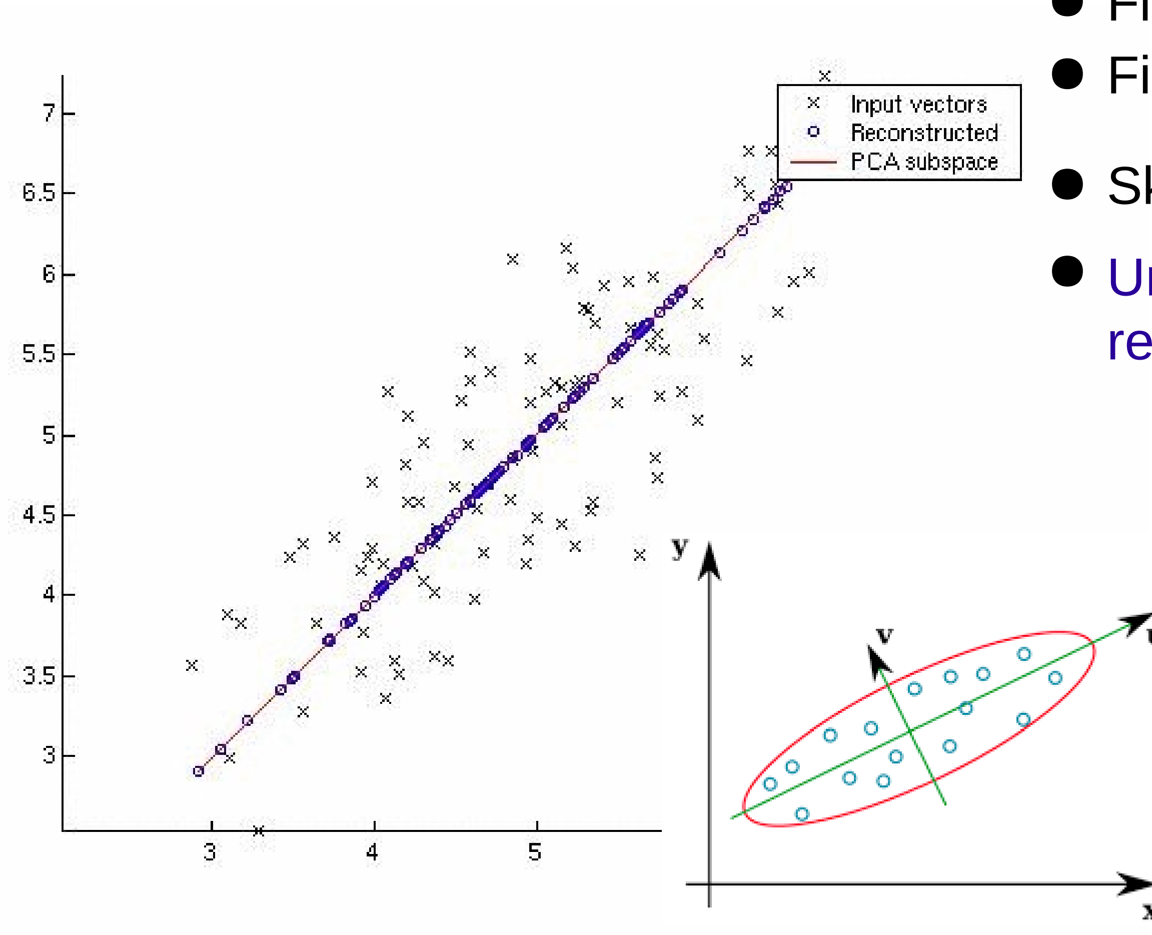
Eigendecomposition of matrix

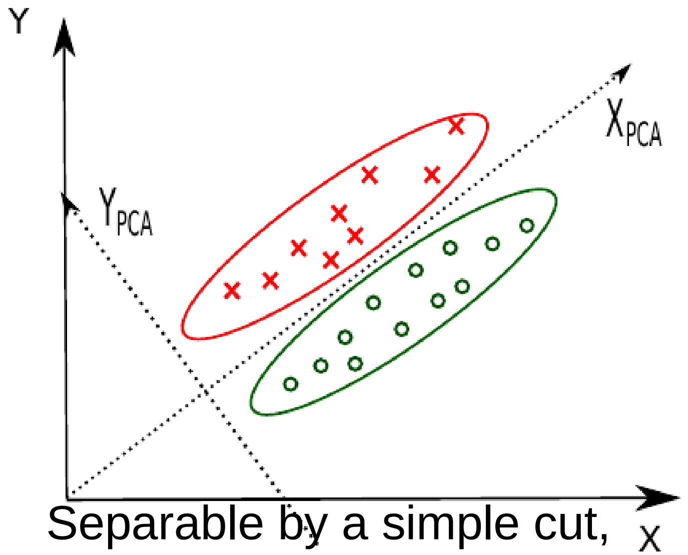
Principal Component Analysis - PCA

- Task: reduce the number of dimensions minimizing the loss of information
- Finds the orthogonal base of the covariance matrix, the eigenvectors with the smallest eigenvalues might be skipped

Procedure:

- Find the covariance matrix Cov(X)
- Find eigenvalues λ_i and eigenvectors v_i
- Skip smallest λ_i
- Unsupervised learning & dimensionality reduction



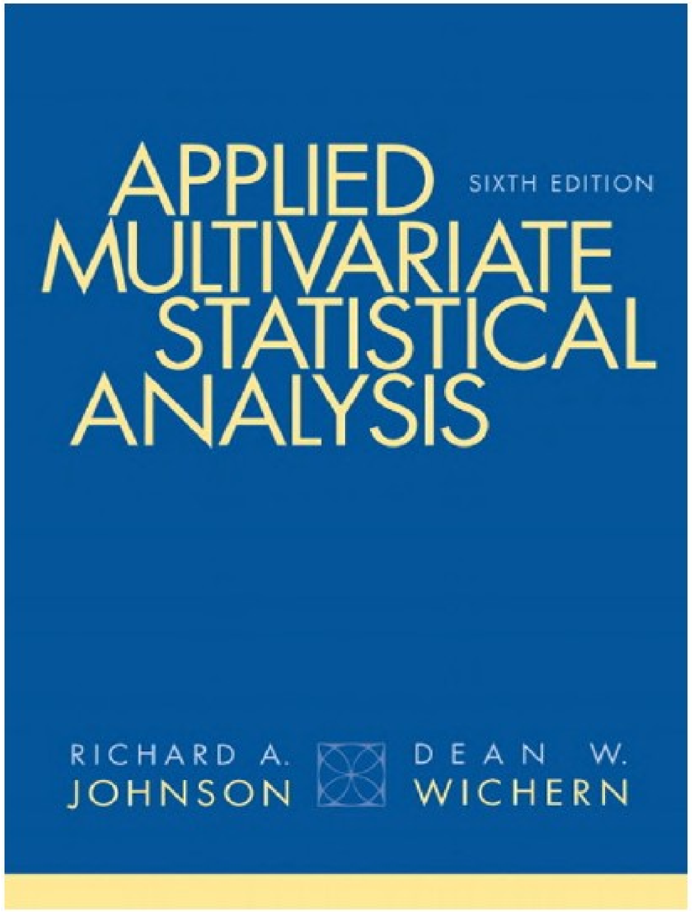


Separable by a simple cut, but in this case PCA doesn't help...

Principal Component Analysis (PCA)

Nicely explained in:

http://docshare04.docshare.tips/files/12598/125983744.pdf



Correlation or covariance matrix?

- Mean-centering is unnecessary if performing a principal components analysis on a correlation matrix, as the data are already centered after calculating correlations.
- We tend to use the covariance matrix when the variable scales are similar and the correlation matrix when variables are on different scales.

$$\underbrace{\mathrm{Corr}(X,Y)}_{\text{Correlation between X and Y}} = \left.\frac{\mathrm{Cov}(X,Y)}{\underbrace{\sigma_x}_{\text{Standard deviation of X}}\underbrace{\sigma_y}_{\text{Standard deviation of Y}}}\right\} \text{Covarianced normalized by Standard Deviation}$$

Covariance Matrix

- Let X be a p-variate random vector. The covariance matrix of X is defined as:

$$\Sigma_{XX} = Var(X) = E\{(X-\mu)^T(X-\mu)\}$$
$$= \begin{pmatrix} Var(X_1) & Cov(X_1,X_2) & \dots & Cov(X_1,X_p) \\ Cov(X_2,X_1) & Var(X_2) & \dots & Cov(X_2,X_p) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_p,X_1) & Cov(X_p,X_2) & \dots & Var(X_p) \end{pmatrix}.$$

Where:

$$Var(X_i) = E\{(X_i - \mu(X_i)) \cdot (X_i - \mu(X_i))\}$$
$$Cov(X_i, Y_j) = E\{(X_i - \mu(X_i)) \cdot (Y_j - \mu(Y_j))\}$$

Correlation matrix

The correlation matrix refers to the symmetric array of numbers

$$\mathbf{R} = \begin{pmatrix} 1 & r_{12} & r_{13} & \cdots & r_{1p} \\ r_{21} & 1 & r_{23} & \cdots & r_{2p} \\ r_{31} & r_{32} & 1 & \cdots & r_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & r_{p3} & \cdots & 1 \end{pmatrix}$$

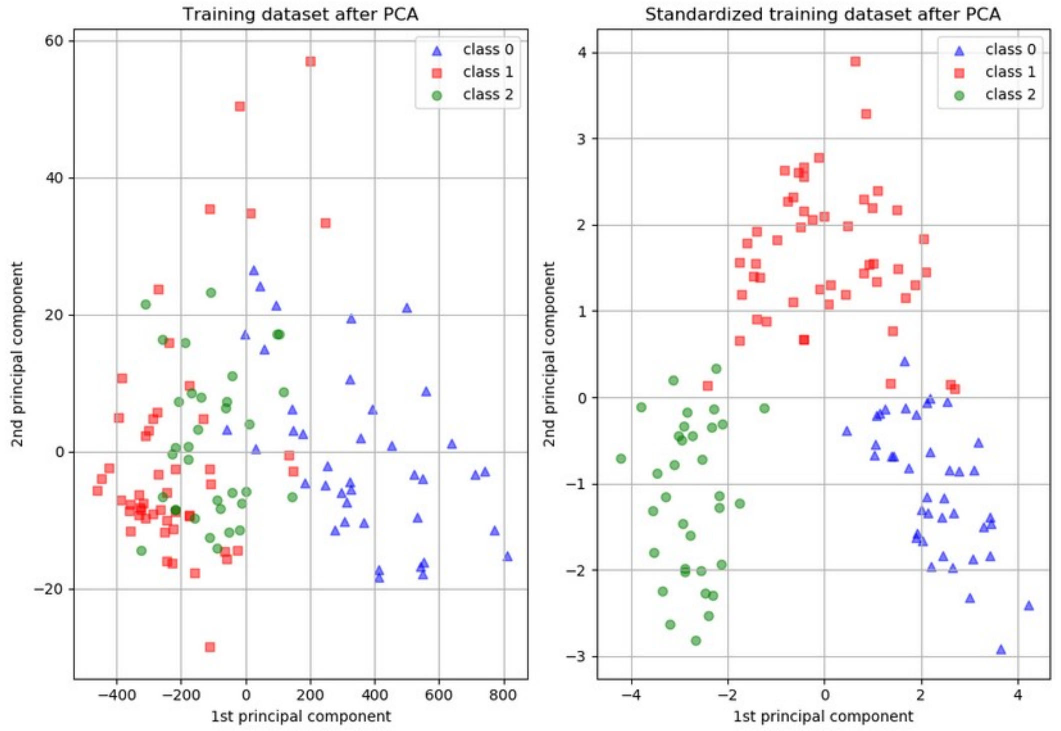
where

$$r_{jk} = \frac{s_{jk}}{s_j s_k} = \frac{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)}{\sqrt{\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}\sqrt{\sum_{i=1}^{n}(x_{ik} - \bar{x}_k)^2}}$$

is the Pearson correlation coefficient between variables $\mathbf{x}_j$ and $\mathbf{x}_k$.

What happens if we perform PCA without normalization? Why do we normalize data?

In PCA we are interested in the components that maximize the variance. If one component (e.g. human height) varies less than another (e.g. weight) because of their respective scales (meters vs. kilos), PCA might determine that the direction of maximal variance more closely corresponds with the 'weight' axis, if those features are not scaled. As a change in height of one meter can be considered much more important than the change in weight of one kilogram, this is clearly incorrect.



The dataset used is the Wine Dataset available at UCI. This dataset has continuous features that are heterogeneous in scale due to differing properties that they measure (i.e alcohol content, and malic acid).

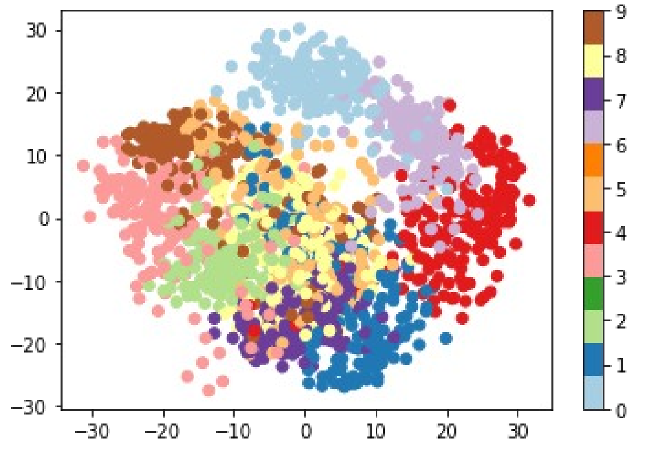
Applications

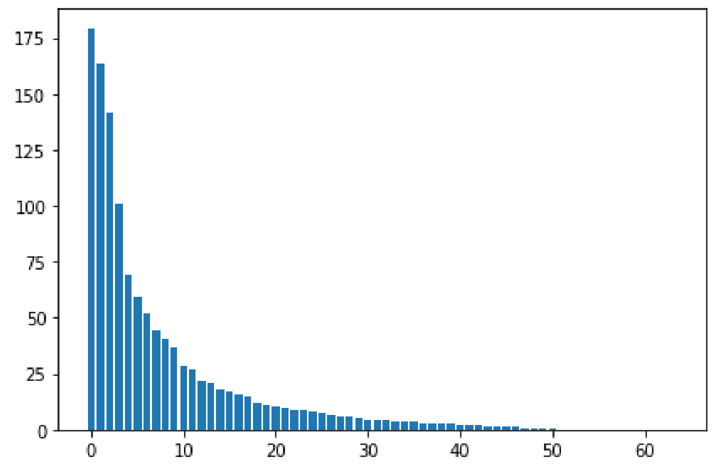
- **Uses:**
  - Data Visualization
  - Data Reduction
  - Data Classification
  - Noise Reduction

- **Examples**:
  - How many unique “sub-sets” are in the sample?
  - How are they similar / different?
  - Which measurements are needed to differentiate?
  - How to best present what is “interesting”?
  - Which “sub-set” does this new sample rightfully belong?

Example

- All examples will be available here:
  https://github.com/marcinwolter/MachineLearning2020
- https://github.com/marcinwolter/MachineLearning2020/blob/main/plot_digits_classif.ipynb
  - iPython notebook prepared to run on Google Colaboratory
  https://colab.research.google.com/
  - Reads handwritten digits
  - Performs PCA
  - Displays two first principal components:





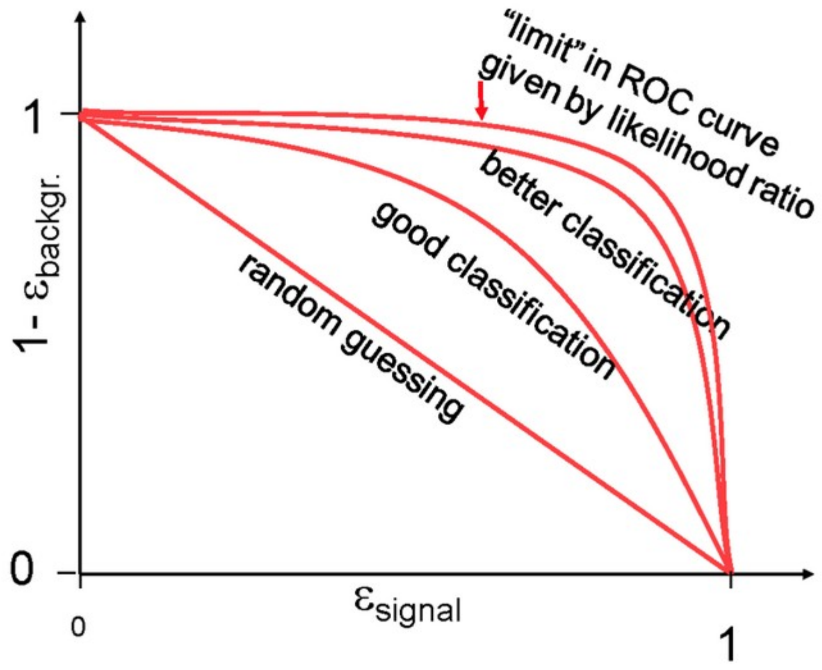
  - Classification using Naive Bayes and LDA

Simple classificators

- We have learned about:
  - Disciminants:
    - Cuts
    - Fisher Linear Discriminant
    - Naive Bayes
  - Unsupervised method
    - Decorrelation
    - Principal Component Analysis
- Today we will talk about:
  - Overtraining problem.
  - Other discrimimants: k-nearest neighbors, decission tree
  - Example how to use them and how to check their performance (ROC curve).
  - Example of face classification – eigenfaces.

ROC curve
(see previous lecture)

- ROC (Receiver Operation Characteristic) curve was first used to calibrate radars.
- Shows the background rejection ($1-\varepsilon_B$) vs signal efficiency ε_B. Shows how good the classifier is.
- The integral of ROC could be a measure of the classifier quality:



Integral(ROC) = ½ – random

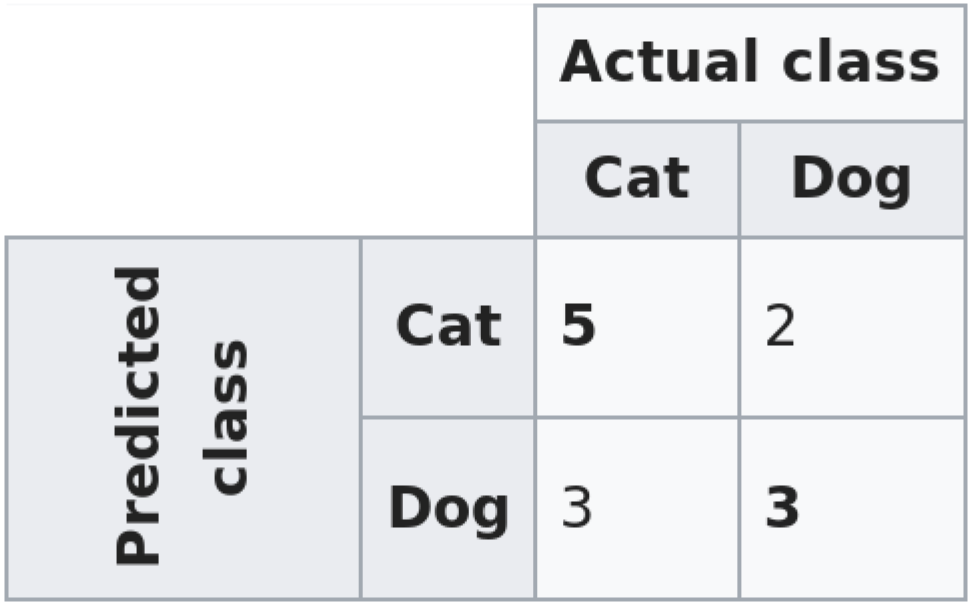
Integral(ROC) = 1 - ideal

Classification Accuracy

- Classification accuracy is the ratio of correct predictions to total predictions made.
  - **classification accuracy = correct predictions / total predictions**
- It is often presented as a percentage by multiplying the result by 100.
  - **classification accuracy = correct predictions / total predictions * 100**
- Classification accuracy can also easily be turned into a misclassification rate or error rate by inverting the value, such as:
  - **error rate = (1 - (correct predictions / total predictions)) * 100**

Confusion matrix

- Example confusion matrix (recognition of dogs vs. cats)



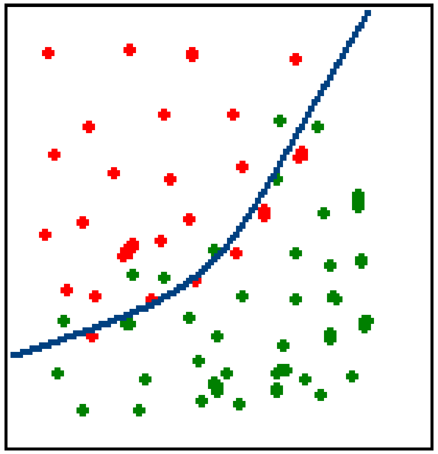
		Actual class	
		Cat	Dog
Predicted class	Cat	**5**	2
	Dog	3	**3**

Confusion Matrix

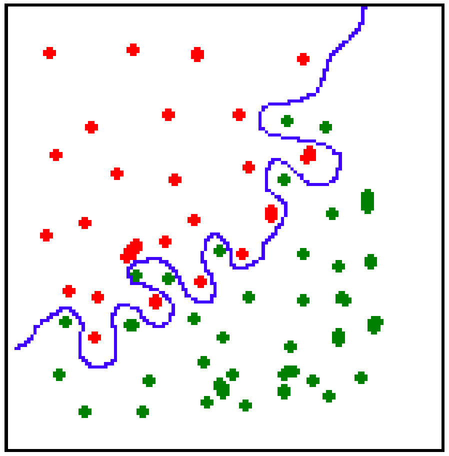
		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error
	Predicted condition negative	**False negative**, Type II error	**True negative**
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power $= \frac{\Sigma\ \text{True positive}}{\Sigma\ \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma\ \text{False positive}}{\Sigma\ \text{Condition negative}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma\ \text{False negative}}{\Sigma\ \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma\ \text{True negative}}{\Sigma\ \text{Condition negative}}$

Overtraining

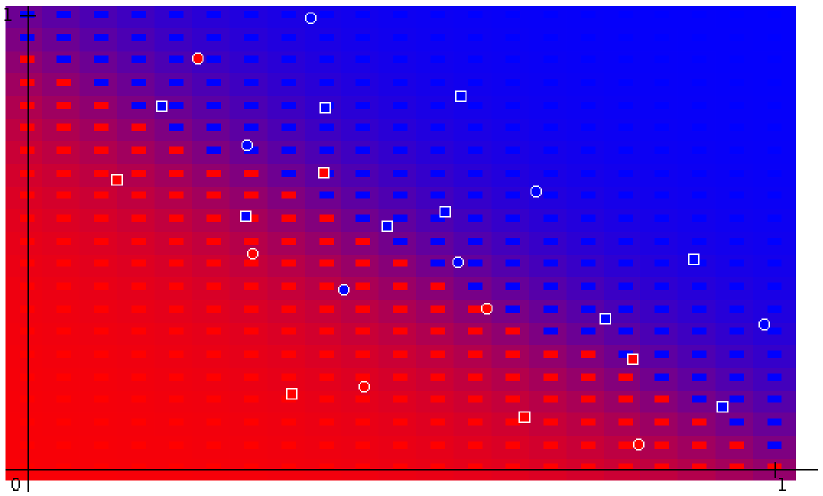
- **Overtraining** – algorithm “learns” the particular events, not the rules.
- This effect important for all ML algorithms.
- **Remedy – checking with another, independent dataset.**

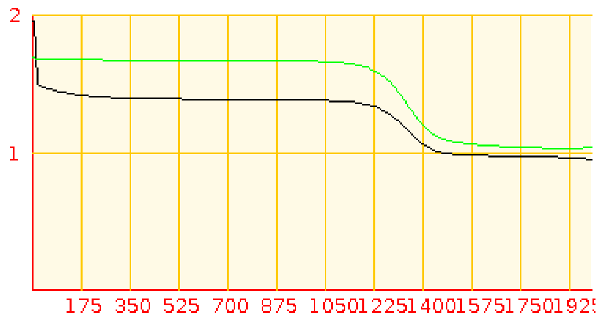


Correct



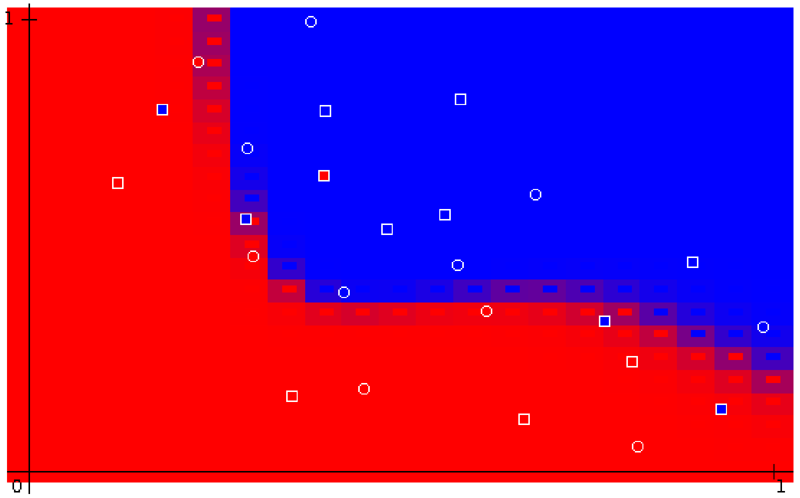
Overtraining

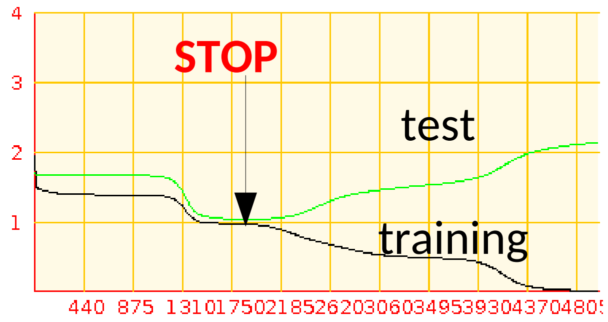




● ● Training sample

■ ■ Test sample



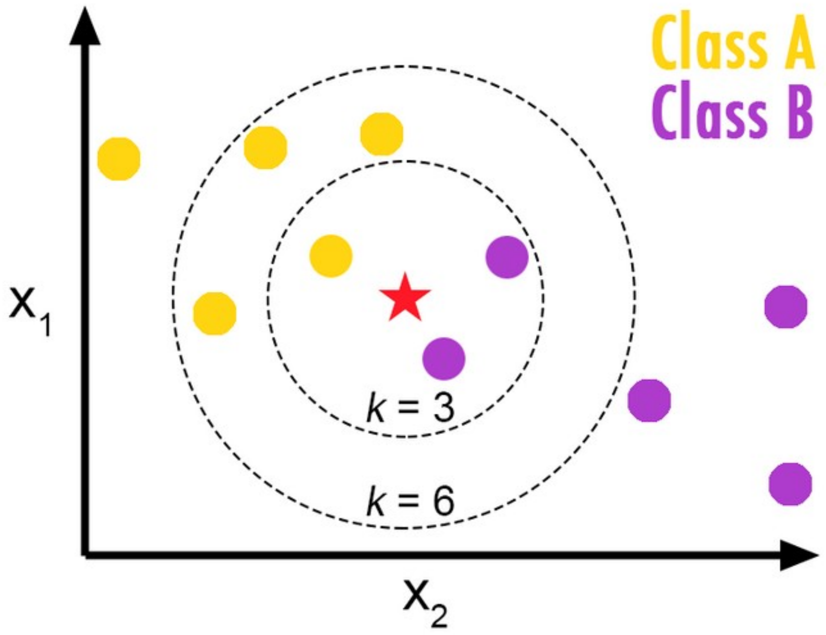


Example of using Neural Network.

KNN - k nearest neighbors

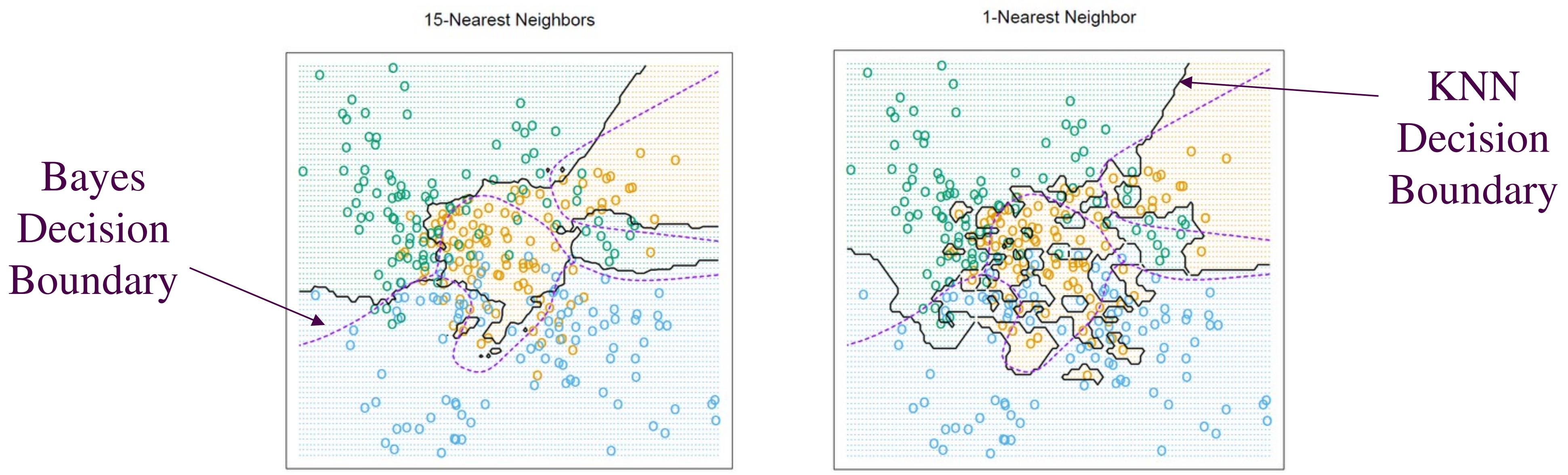
- Proposed already in 1951
- An event is qualified to the class, to which belongs the majority of it's k nearest neighbors,
- or we calculate the probability of belonging to the class "signal" as:

$$p(S|x) = \frac{k_S}{k}$$



K-Nearest Neighbors

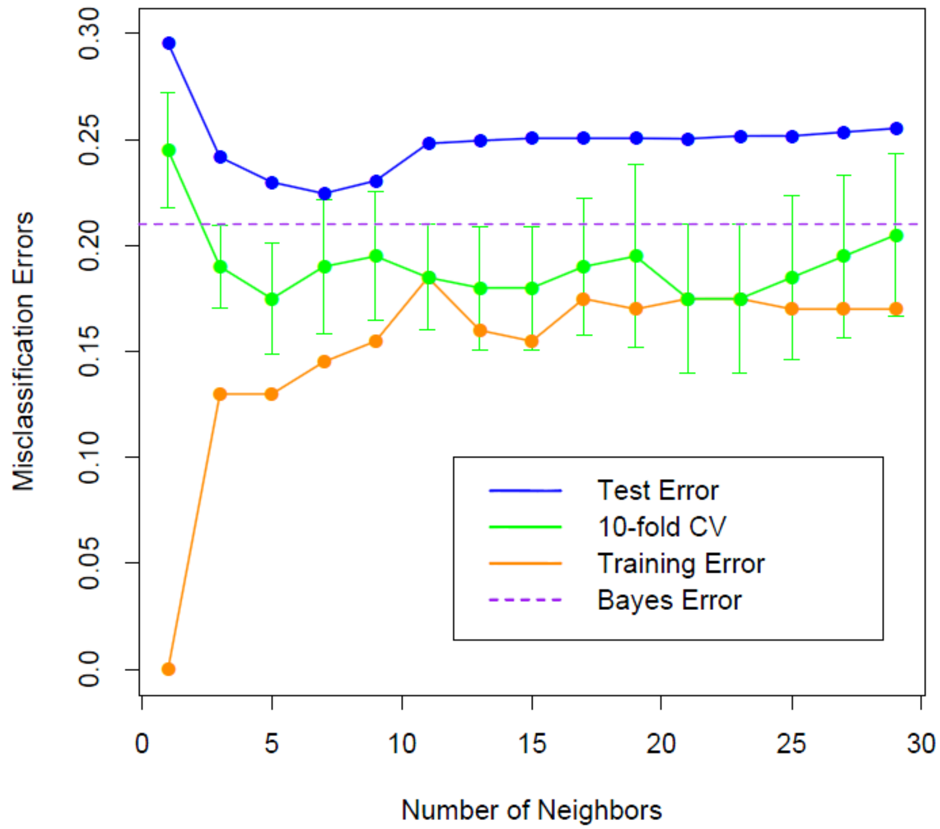
- The only parameter that can adjust the complexity of KNN is the number of neighbors *k*.
- The larger *k* is, the smoother the classification boundary. Or we can think of the complexity of KNN as lower when *k* increases.



K-Nearest Neighbors

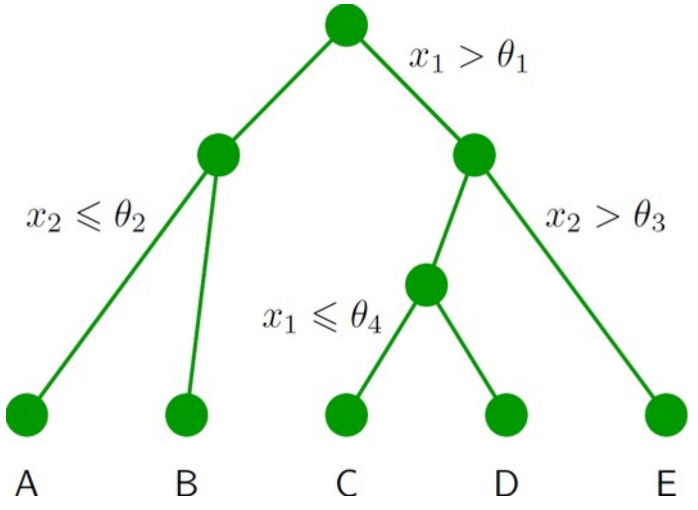
- For another simulated data set, there are two classes. The error rates based on the training data, test data, and *10*-fold cross validation are plotted against *K*, the number of neighbors.

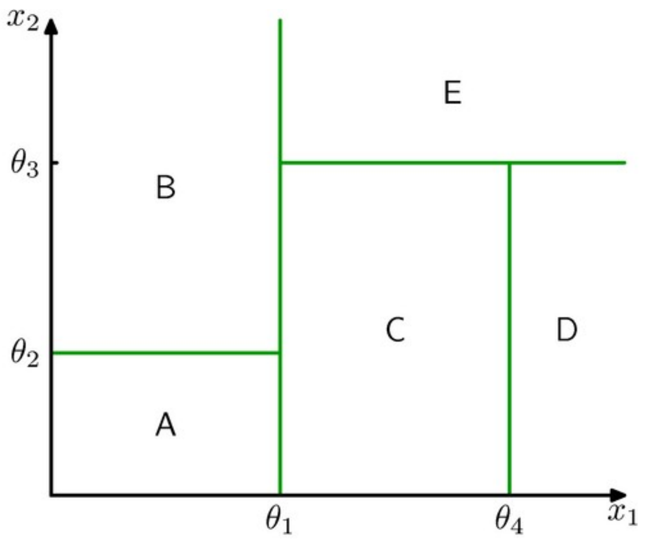
- We can see that the training error rate tends to grow when *k* grows, which is not the case for the error rate based on a separate test data set or cross-validation.



Decision trees

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

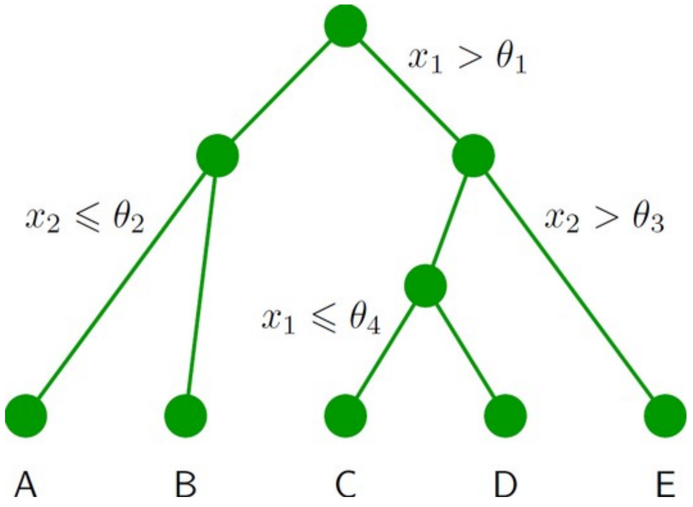




- Easy in visualization and interpretation
- Resistant for *outliers*.
- Weak variables are ignored.
- Fast training and classification.
- Unfortunately: **sensitive for fluctuations, unstable**.

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*)

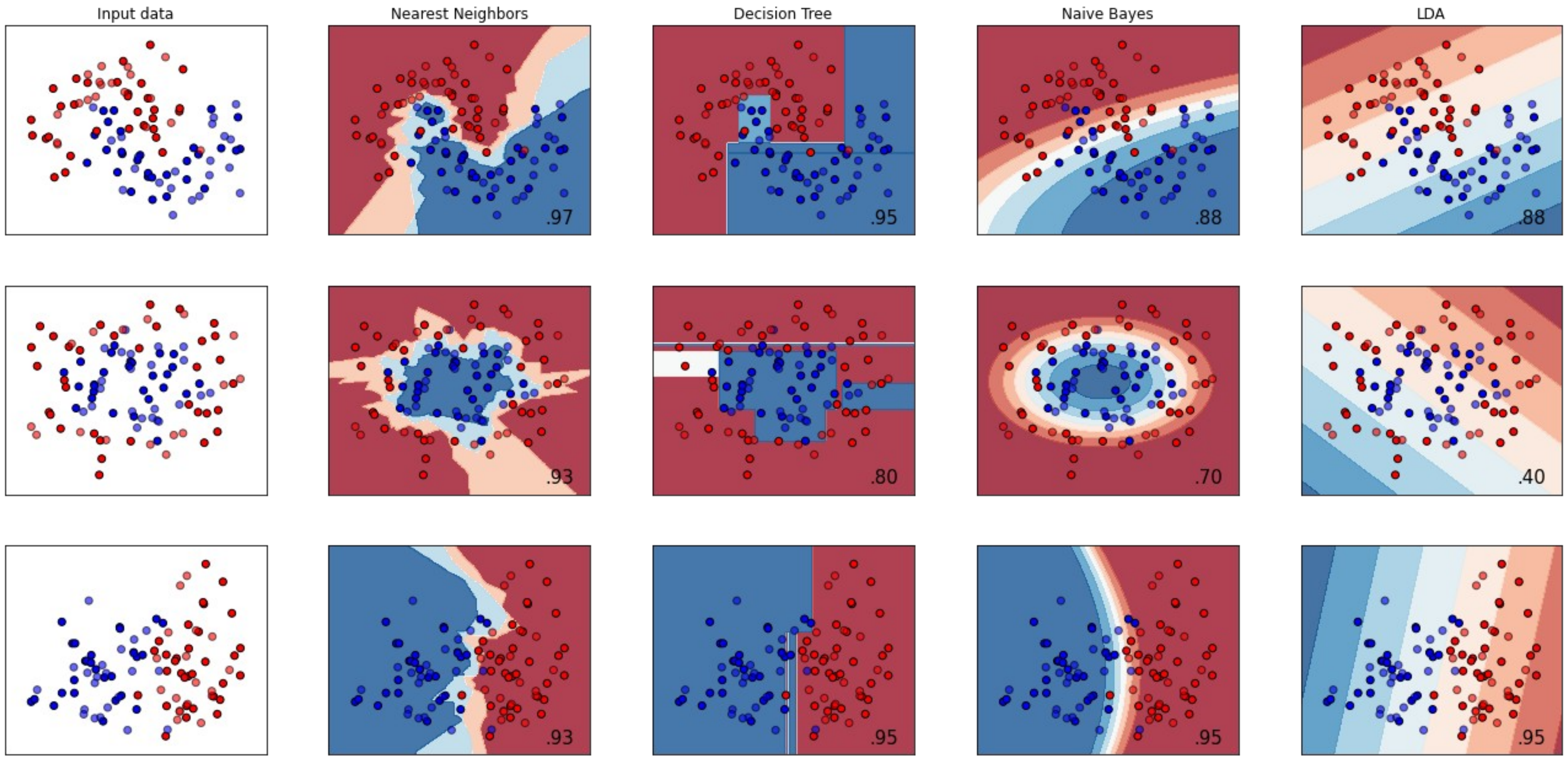
  p (1-p) : p= P(signal), *purity*
- Cross-entropy:

  -(p ln p + (1-p)ln(1-p))
- Missidentification:

  1-max(p,1-p)

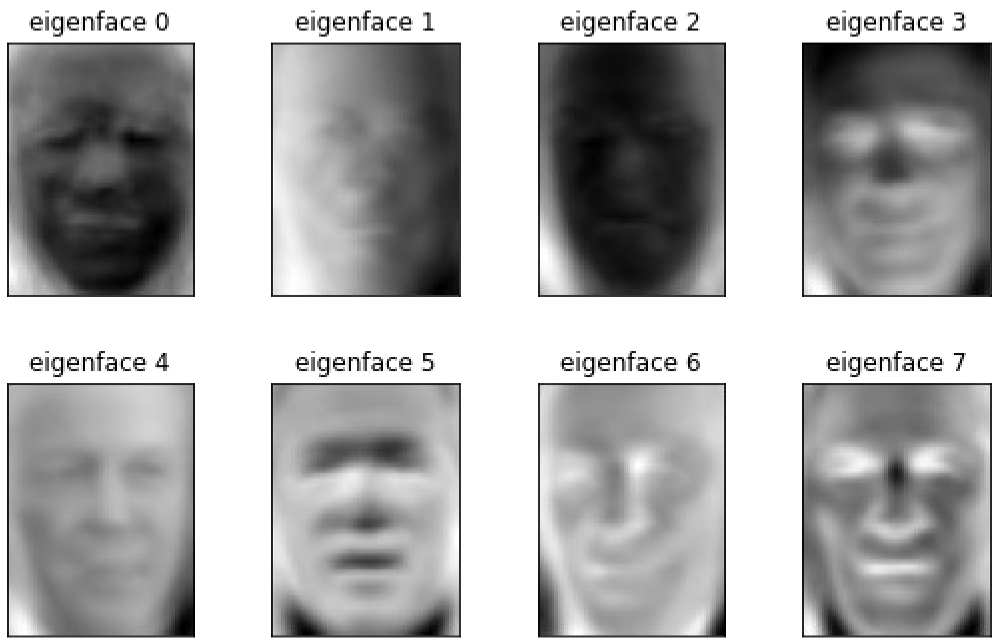
Example
# of simple classifiers

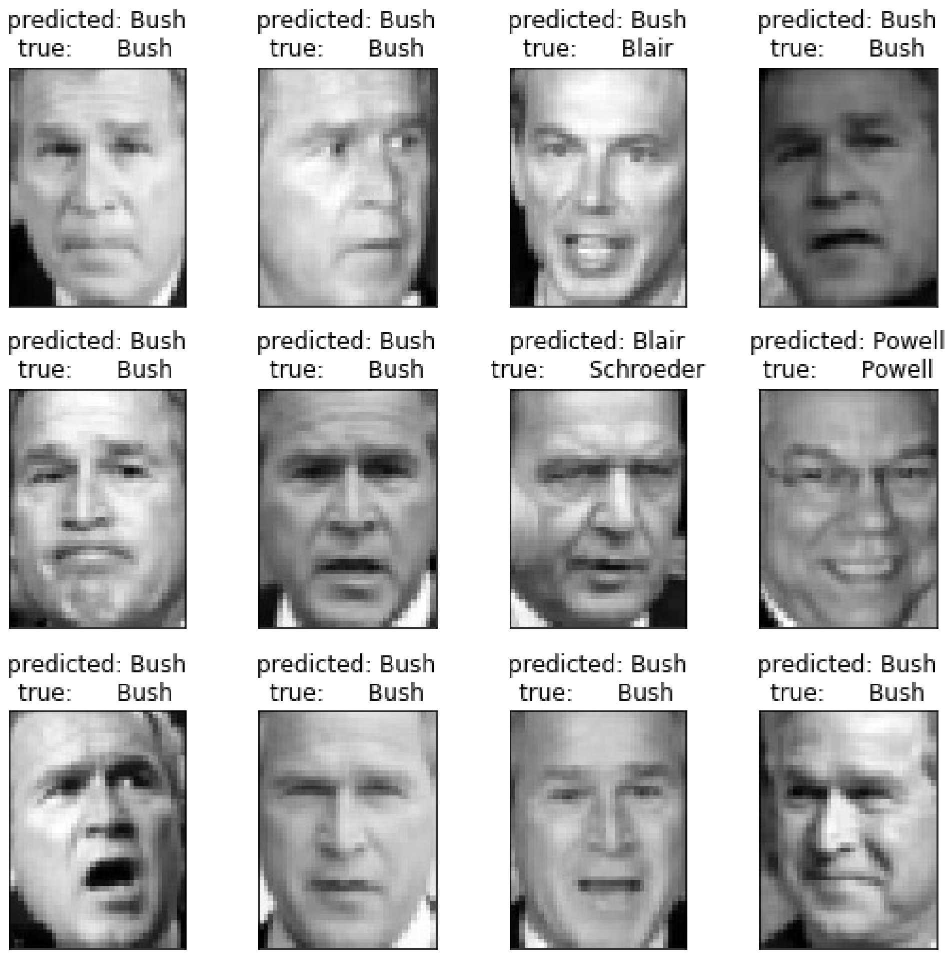
https://github.com/marcinwolter/MachineLearning2020/blob/main/simple_classifier_comparison.ipynb



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





Summary

- We have learned about simple classifiers….
- ...how to train them, how to optimize them, how to measure their performance...
- ...and about unsupervised methods like Principal Component Analysis.

- Next time we will talk about:
  - Independent Component Analysis ICA
  - Boosted Decision Trees BDT

Backup – PCA in detail

Steps of PCA

Calculate the empirical mean

Find the empirical mean along each column j = 1, ..., p.
Place the calculated mean values into an empirical mean vector u of dimensions p × 1.

$$u_j = \frac{1}{n} \sum_{i=1}^{n} X_{ij}$$

Calculate the deviations from the mean
Subtract the mean from each data point

Find the covariance matrix

Find the p × p empirical covariance matrix **C** from matrix **B** (data):

$$\mathbf{C} = \frac{1}{N} \mathbf{B}^T \mathbf{B}$$

Find eigenvalues by solving:

$$det(\mathbf{C} - \lambda \mathbb{1}) = 0$$

This means solving a characteristic polynomial.

Steps of PCA

Find the eigenvectors and eigenvalues of the covariance matrix

Compute the matrix **V** of eigenvectors which diagonalizes the covariance matrix **C**:

$$\mathbf{C}\overrightarrow{v_i} = \lambda_i \overrightarrow{v_i}$$

Rearrange the eigenvectors and eigenvalues

Sort the columns of the eigenvector matrix V and eigenvalue matrix D in order of decreasing eigenvalue.

Select a subset of the eigenvectors as basis vectors

The goal is to choose a value of L as small as possible while achieving a reasonably high value of g on a percentage basis.

Project the z-scores of the data onto the new basis

You have reduced the dimensionality of your explaining as much variance as possible.

Steps of PCA

Find how much of variance is explained by n first principal components

In case of PCA, "variance" means *summative variance* or *multivariate variability* or *overall variability* or *total variability*. Below is the covariance matrix of some 3 variables. Their variances are on the diagonal, and the sum of the 3 values (3.448) is the overall variability.

```
    1.343730519   -.160152268    .186470243
    -.160152268    .619205620   -.126684273
     .186470243   -.126684273   1.485549631
```

Now, PCA replaces original variables with new variables, called principal components, which are orthogonal (i.e. they have zero covariations) and have variances (called eigenvalues) in decreasing order. So, the covariance matrix between the principal components extracted from the above data is this:

```
    1.651354285    .000000000    .000000000
     .000000000   1.220288343    .000000000
     .000000000    .000000000    .576843142
```

Note that the diagonal sum is still 3.448, which says that all 3 components account for all the multivariate variability. The 1st principal component accounts for or "explains" 1.651/3.448 = 47.9% of the overall variability; the 2nd one explains 1.220/3.448 = 35.4% of it; the 3rd one explains .577/3.448 = 16.7% of it.