Open for Innovation
KNIME

[L4-ML] Introduction to Machine Learning Algorithms

KNIME AG

Structure of the Course

Session	Topic
Session 1	Introduction & Decision Tree Algorithm
Session 2	Regression Models, Ensemble Models, & Logistic Regression
Session 3	Neural Networks & Recommendation Engines
Session 4	Clustering & Data Preparation
Session 5	Last Exercise and Q&A

- Structure of each session
- Discussion of past exercises (10 minutes)
- Course (60 minutes)
- Introduction of next exercises (5 minutes)

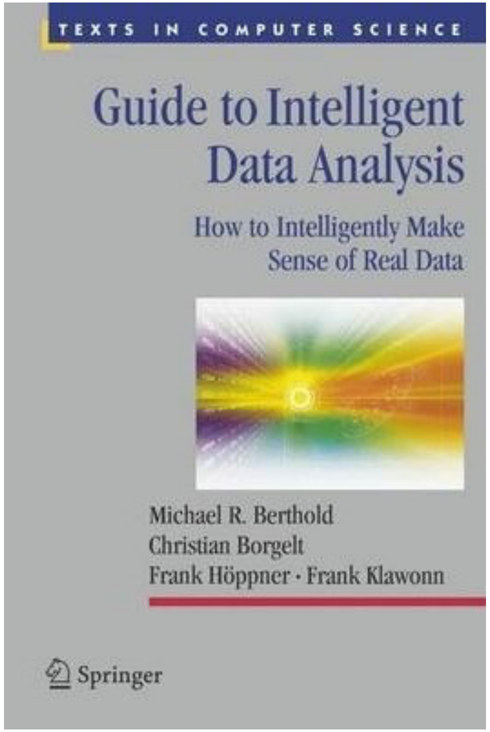
Material

- Michael Berthold, Christian Borgelt, Frank Höppner, Frank Klawonn:
  Guide to Intelligent Data Analysis
  Springer, 2010.

- Tom Mitchell:
  Machine Learning
  McGraw Hill, 1997.

- David Hand, Heikki Mannila, Padhraic Smyth:
  Principles of Data Mining
  MIT Press, 2001.

- Michael Berthold, David Hand (eds):
  Intelligent Data Analysis, An Introduction
  (2nd Edition) Springer Verlag, 2003.


TEXTS IN COMPUTER SCIENCE
Guide to Intelligent Data Analysis
How to Intelligently Make Sense of Real Data
Michael R. Berthold
Christian Borgelt
Frank Höppner · Frank Klawonn
Springer

What is Data Science?

[Wikipedia quoting Dhar 13, Leek 13]

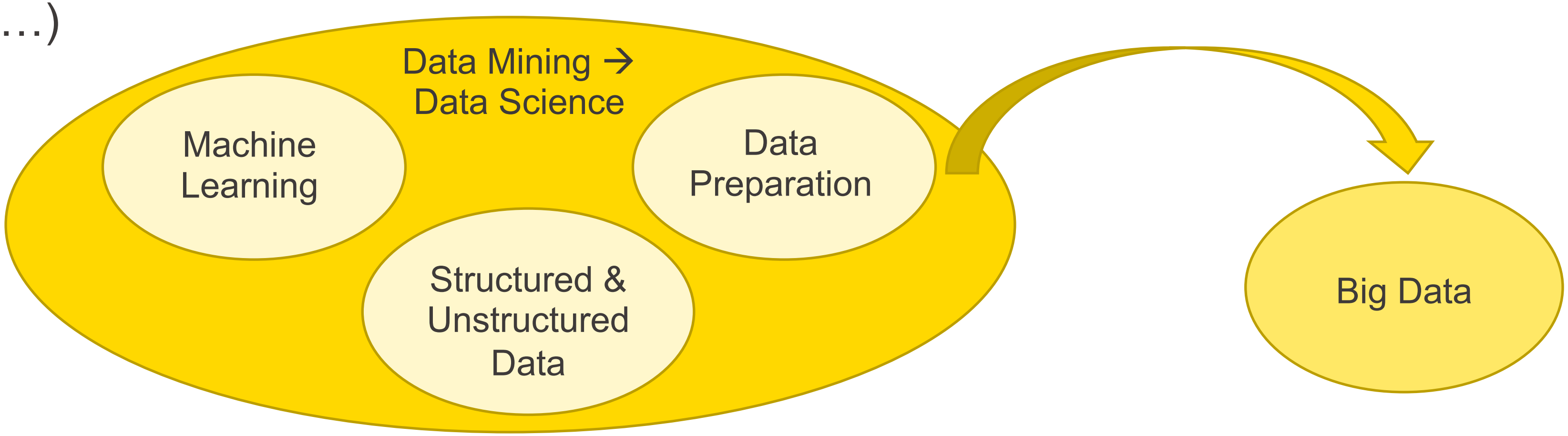
Data science is a multi-disciplinary field that uses scientific methods, processes, algorithms and systems to **extract knowledge and insights** from structured and unstructured data*.*

[Fayyad, Piatetsky-Shapiro & Smyth 96]

Knowledge discovery in databases (*KDD*) is the process of (semi-)automatic **extraction of knowledge** from databases which is *valid*, *previously unknown*, and *potentially useful.*

Some Clarity about Words

- *(semi)-automatic*: no manual analysis, though some user interaction required
- *valid*: in the statistical sense
- *previously unknown*: not explicit, no „common sense knowledge“
- *potentially useful*: for a given application
- *structured data*: numbers
- *unstructured data*: everything else (images, texts, networks, chem. compounds, …)



Use Case Collection

Churn Prediction

CRM System
Data about your customer

- Demographics
- Behavior
- Revenues

Model

- Churn Prediction
- Upselling Likelihood
- Product Propensity /NBO
- Campaign Management
- Customer Segmentation
- …

Customer Segmentation

CRM System
Data about your customer

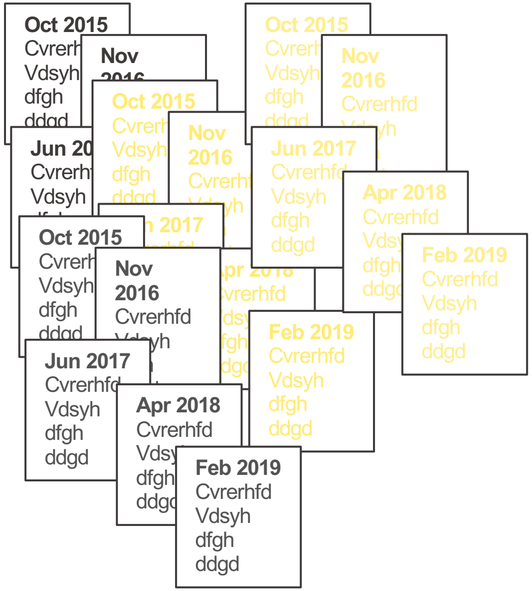
- Demographics
- Behavior
- Revenues

Model

- Churn Prediction
- Upselling Likelihood
- Product Propensity /NBO
- Campaign Management
- Customer Segmentation
- …

Risk Assessment

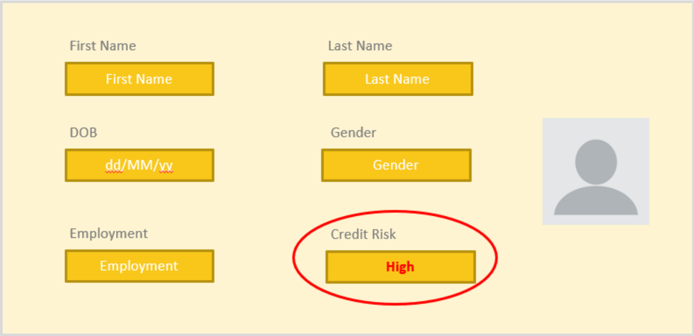
Customer History



Model

Risk Prognosis

- High Risk
- Low Risk
- High Risk
- Very High Risk
- Very Low Risk
- Medium Risk
- …



First Name: First Name

Last Name: Last Name

DOB: dd/MM/yy

Gender: Gender

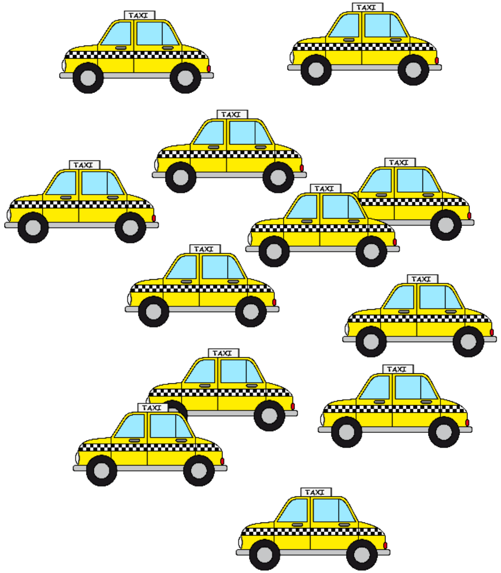
Employment: Employment

Credit Risk: High

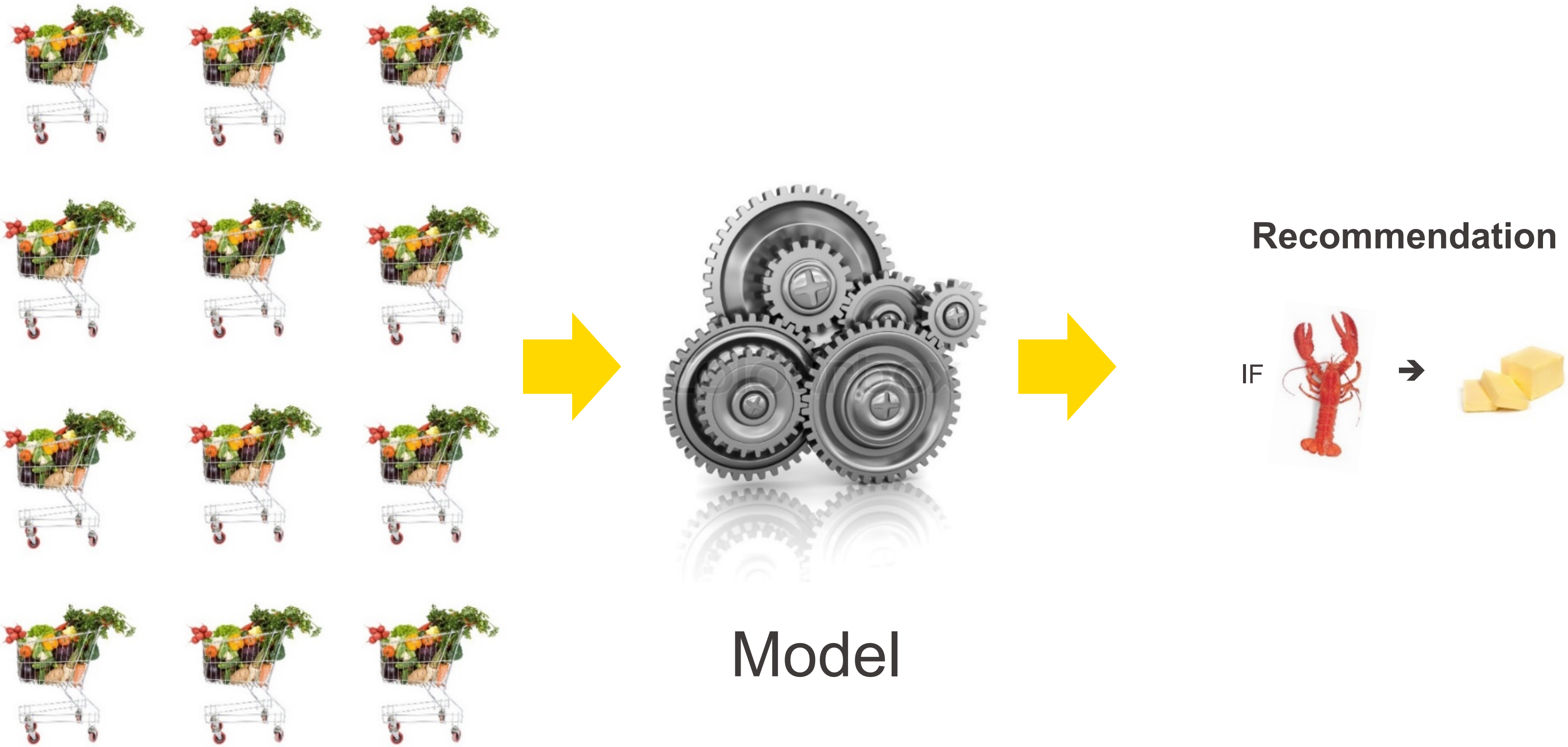
Demand Prediction

- How many taxis do I need in NYC on Wednesday at noon?

Model



Recommendation Engines / Market Basket Analysis



Model

Recommendation

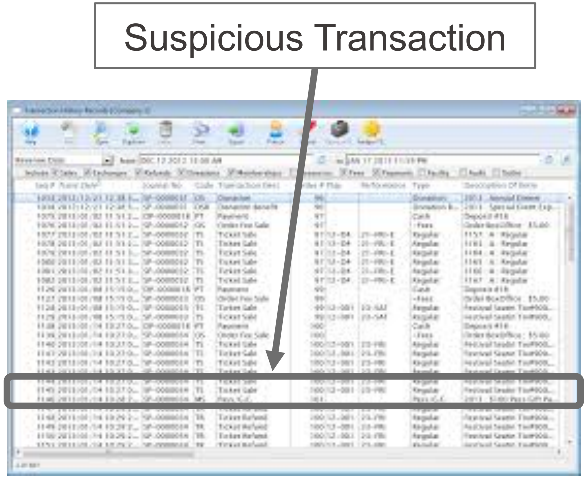
IF →

Fraud Detection

Transactions

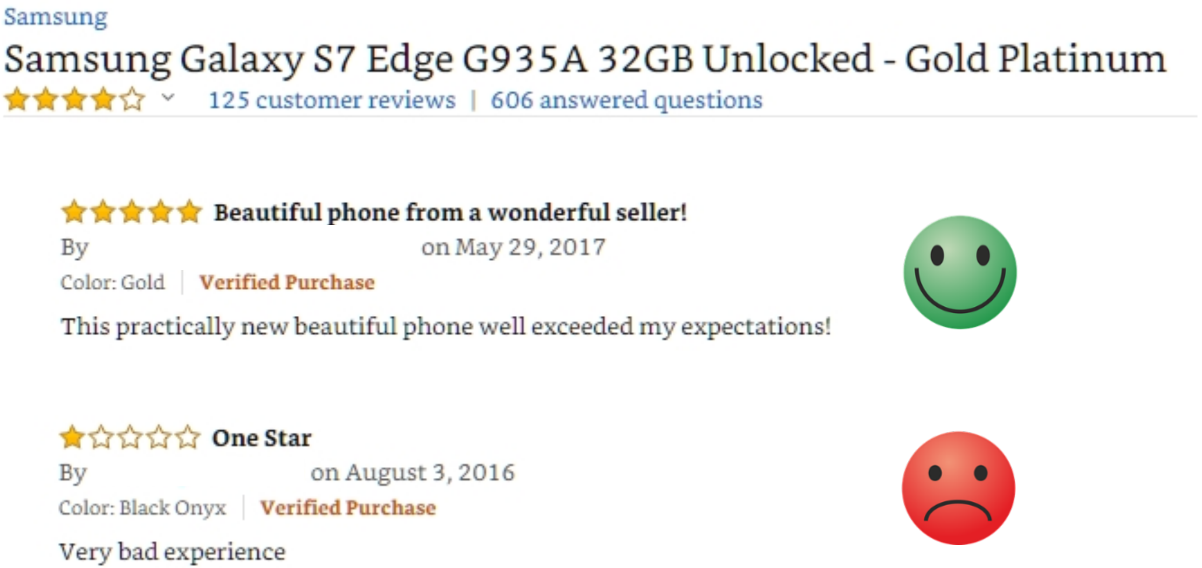
- Trx 1
- Trx 2
- Trx 3
- Trx 4
- Trx 5
- Trx 6
- …

Model



Suspicious Transaction

Sentiment Analysis



Samsung

Samsung Galaxy S7 Edge G935A 32GB Unlocked - Gold Platinum

★★★★☆ 125 customer reviews | 606 answered questions

★★★★★ **Beautiful phone from a wonderful seller!**

By on May 29, 2017

Color: Gold | Verified Purchase

This practically new beautiful phone well exceeded my expectations!

☺

★☆☆☆☆ **One Star**

By on August 3, 2016

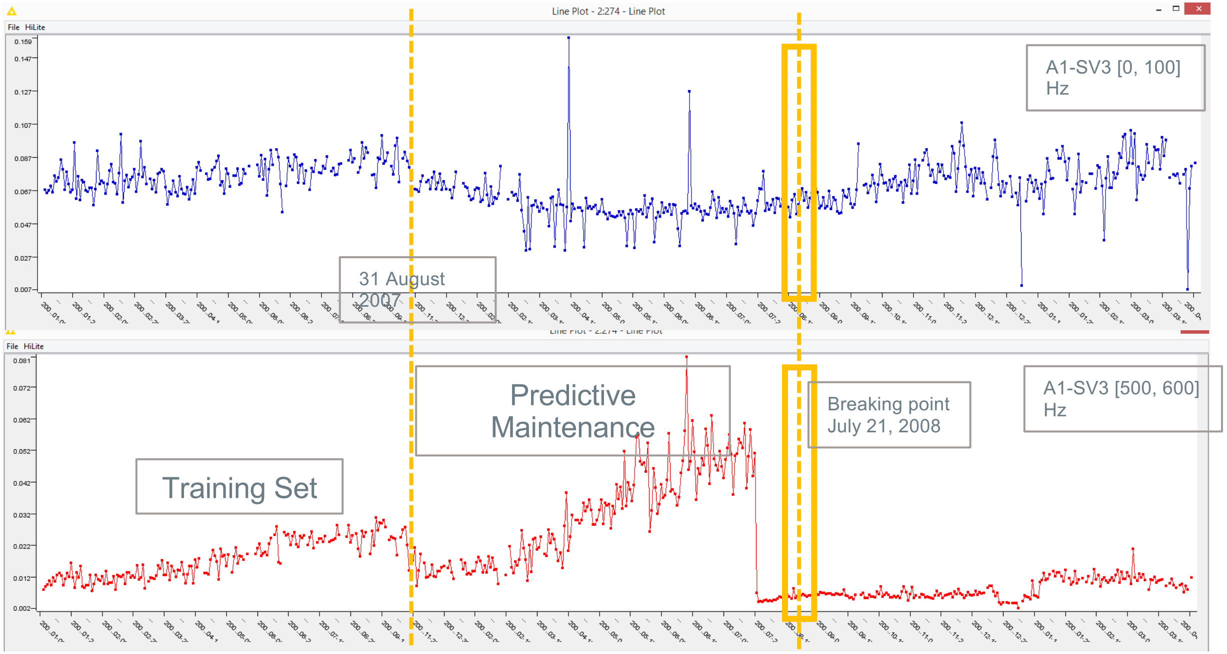
Color: Black Onyx | Verified Purchase

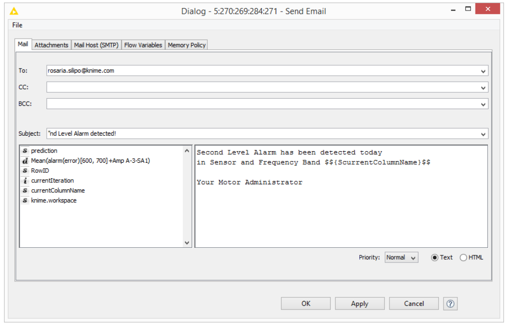
Very bad experience

☹

Anomaly Detection

Predicting mechanical failure as late as possible but before it happens





To: rosaria.silipo@knime.com

CC:

BCC:

Subject: "nd Level Alarm detected!

- prediction
- Mean(alarm(error)[600, 700] +Amp A-3-SA1)
- RowID
- currentIteration
- currentColumnName
- knime.workspace

```
Second Level Alarm has been detected today
in Sensor and Frequency Band $${ScurrentColumnName}$$

Your Motor Administrator
```

Priority: Normal ● Text ○ HTML

via REST

Only some Spectral Time Series shows the break down

Basic Concepts in Data Science

What is a Learning Algorithm?

- Input features
- Input attributes
- Independent variables

$\boldsymbol{X} = (x_1, x_2, \ldots, xn)$

Model

y

- Class
- Label
- Target
- Output feature/attribute
- Dependent variable

Model parameters

$$y = f(\boldsymbol{\beta}, \boldsymbol{X}) \quad \text{with } \boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_m]$$

A learning algorithm adjusts (learns) the model parameters $\boldsymbol{\beta}$ throughout a number of iterations to maximize/minimize a likelihood/error function on y.

Algorithm Training / Learning

- The model ***learns*** / ***is trained*** during the ***learning / training*** phase to produce the right answer *y* (a.k.a., label)

- That is why ***machine learning*** ☺

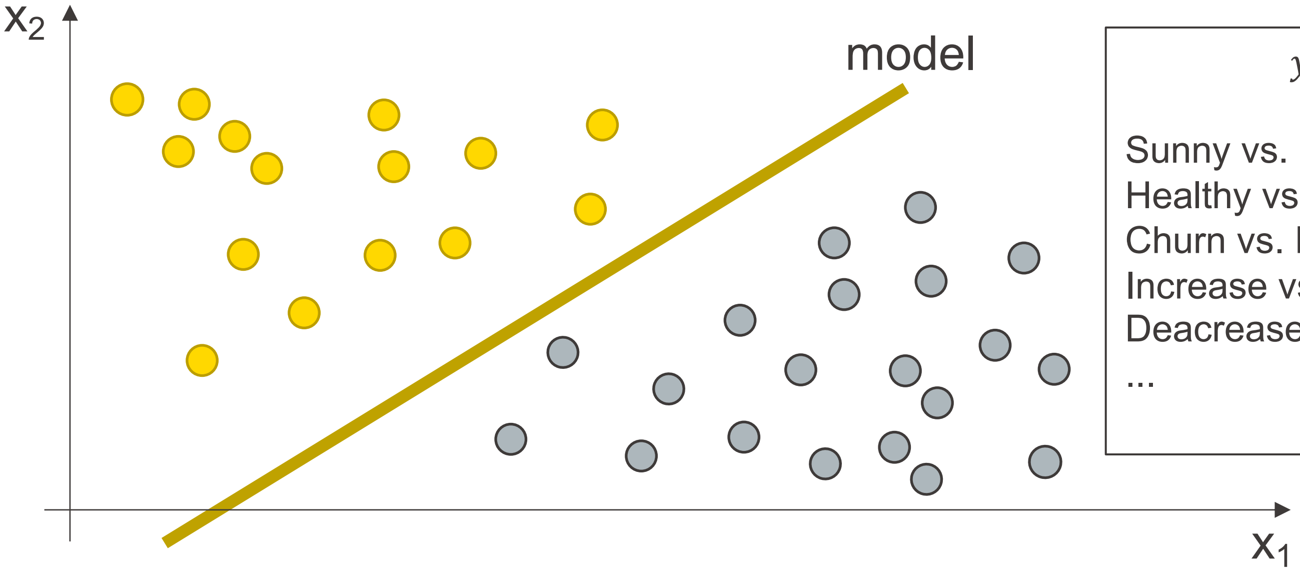
- Many different algorithms for three ways of learning:
  - Supervised
  - Unsupervised
  - Semi-supervised

Supervised Learning

- $\boldsymbol{X} = (x_1, x_2)$ and $y = \{yellow, gray\}$
- A training set with many examples of $(\boldsymbol{X}, y)$
- The model learns on the examples of the training set to produce the right value of y for an input vector $\boldsymbol{X}$

$\boldsymbol{X}$

Age
Money
Temperature
Speed
Number of taxi
...



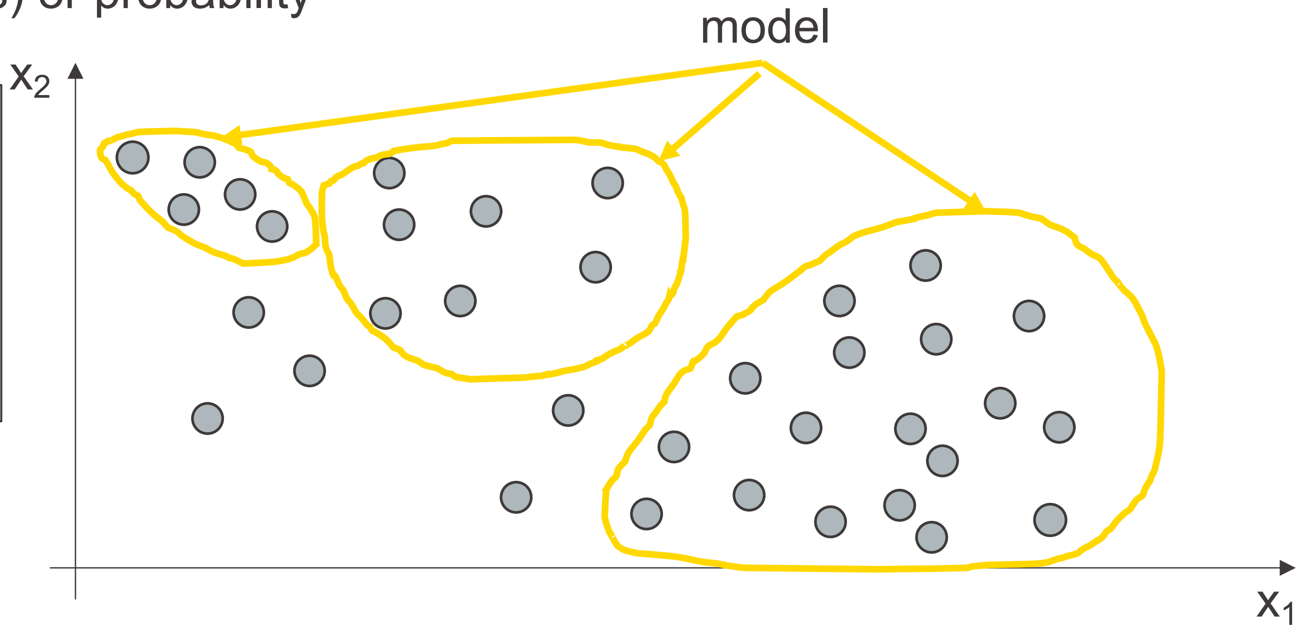
y

Sunny vs. Cloudy
Healthy vs. Sick
Churn vs. Remain
Increase vs.
Deacrease
...

Unsupervised Learning

- $\boldsymbol{X} = (x_1, x_2)$ and ~~$y = \{yellow, gray\}$~~
- A training set with many examples of $(\boldsymbol{X}, \cancel{y})$
- The model learns to group the examples $\boldsymbol{X}$ of the training set based on similarity (closeness) or probability

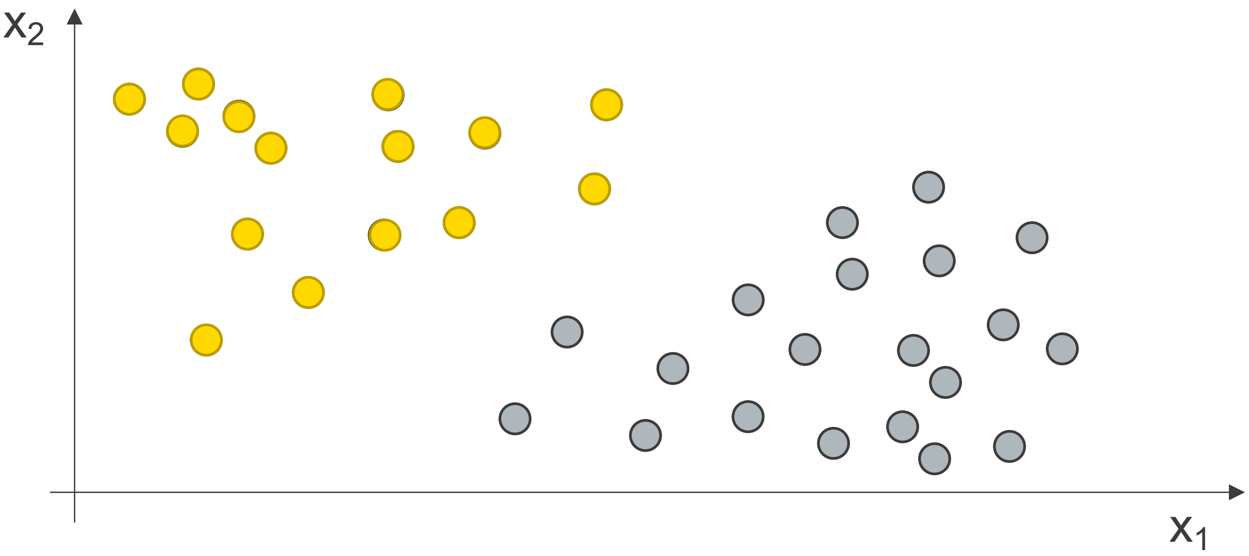
$\boldsymbol{X}$
Age
Money
Temperature
Speed
Number of taxi
...



Semi-Supervised Learning

- $\boldsymbol{X} = (x_1, x_2)$ and $y = \{yellow, gray\}$
- A training set with many examples of $(\boldsymbol{X}, \cancel{y})$ and some samples $(\boldsymbol{X}, y)$
- The model labels the data in the training set using a modified unsupervised learning procedure

X
Age
Money
Temperature
Speed
Number of taxi
...



Supervised Learning: Classification vs. Numerical Predictions

- $\boldsymbol{X} = (x_1, x_2)$ and $y = \boxed{\{label\ 1, \ldots, label\ n\} \text{ or } y \in \mathbb{R}}$
- A training set with many examples of $(\boldsymbol{X}, y)$
- The model learns on the examples of the training set to produce the right value of y for an input vector $\boldsymbol{X}$

Classification

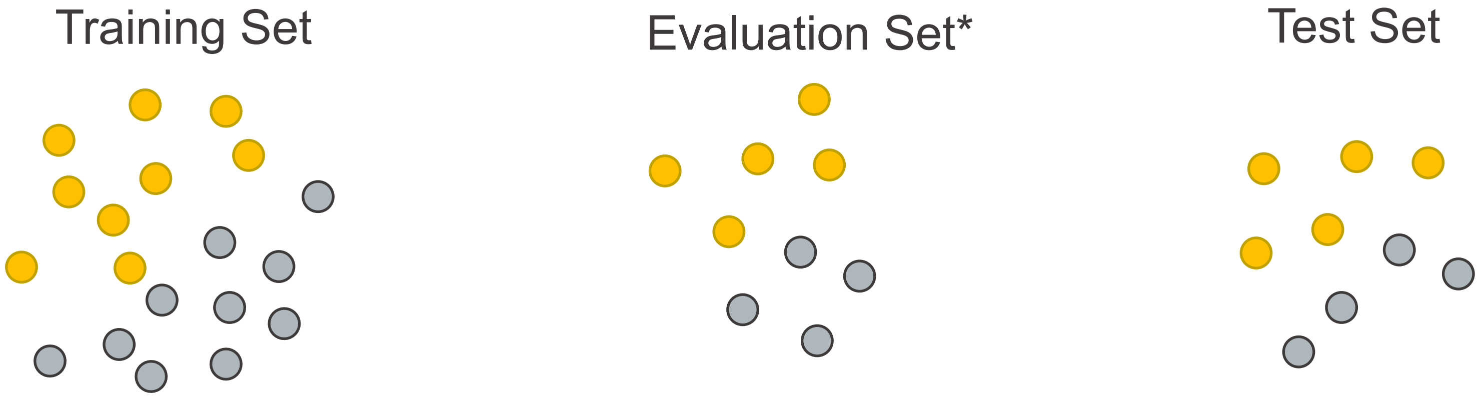
y = {yellow, gray}
y = {churn, no churn}
y = {increase, unchanged, decrease}
y = {blonde, gray, brown, red, black}
y = {job 1, job 2, ... , job n}

Numerical Predictions

y = temperature
y = number of visitors
y = number of kW
y = price
y = number of hours

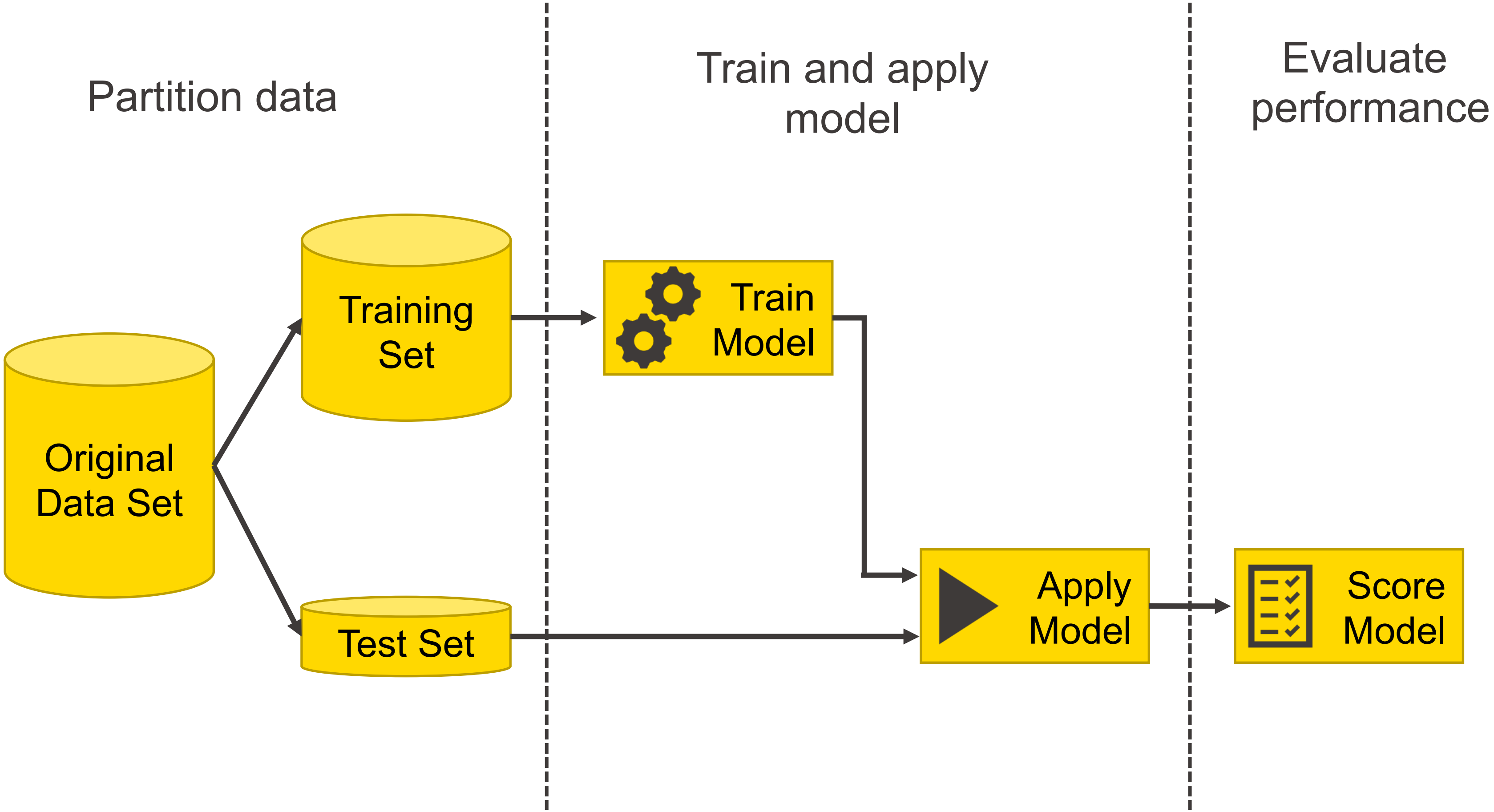
Training vs. Testing: Partitioning

- ***Training phase***: the algorithm trains a model using the data in the training set
- ***Testing phase***: a metric measures how well the model is performing on data in a new dataset (the test set)

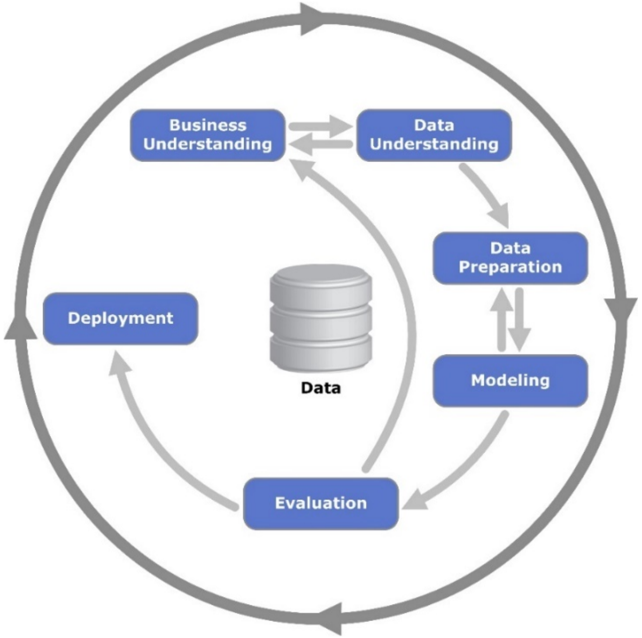


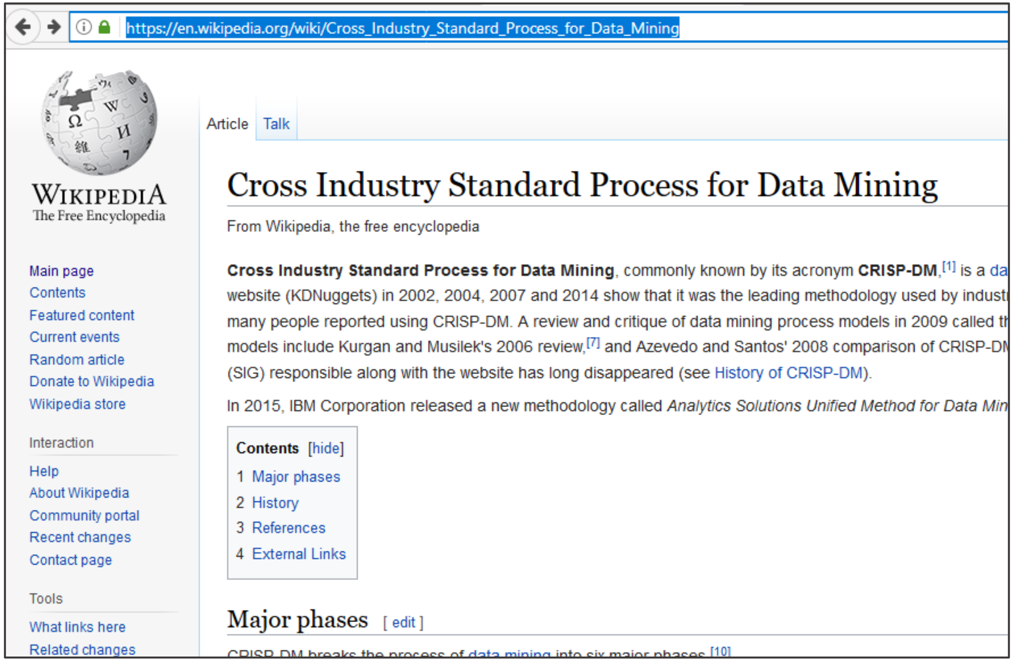
* sometimes

Data Science: Process Overview



The CRISP-DM Cycle





Cross Industry Standard Process for Data Mining

From Wikipedia, the free encyclopedia

Cross Industry Standard Process for Data Mining, commonly known by its acronym **CRISP-DM**,[1] is a da
website (KDNuggets) in 2002, 2004, 2007 and 2014 show that it was the leading methodology used by industr
many people reported using CRISP-DM. A review and critique of data mining process models in 2009 called th
models include Kurgan and Musilek's 2006 review,[7] and Azevedo and Santos' 2008 comparison of CRISP-DM
(SIG) responsible along with the website has long disappeared (see History of CRISP-DM).

In 2015, IBM Corporation released a new methodology called *Analytics Solutions Unified Method for Data Min*

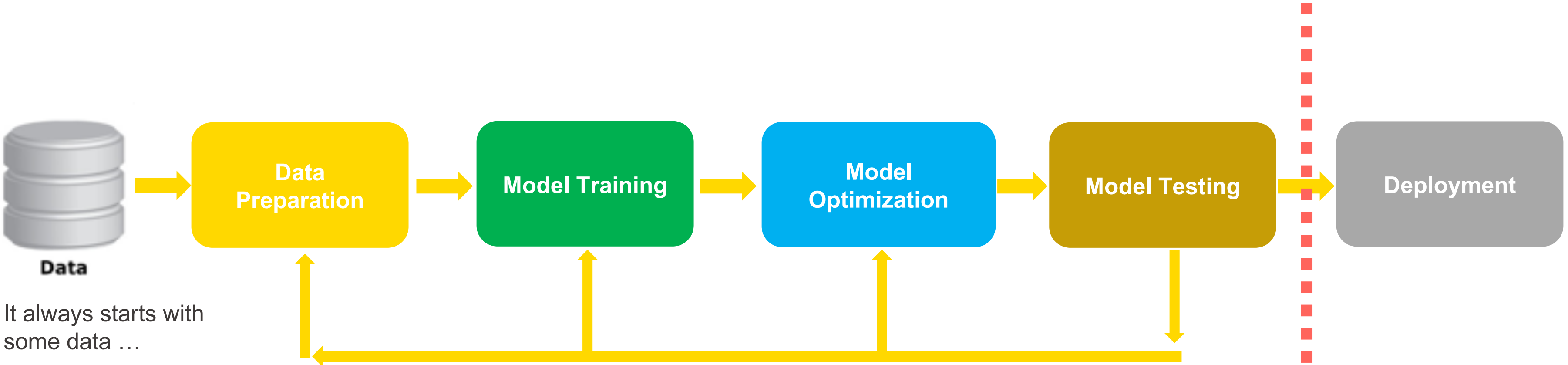
Contents [hide]

Major phases [edit]

CRISP-DM breaks the process of data mining into six major phases.[10]

https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining

A Classic Data Science Project



It always starts with some data …

Data Manipulation
Data Blending
Missing Values Handling
Feature Generation
Dimensionality Reduction
Feature Selection
Outlier Removal
Normalization
Partitioning
…

Model Training
Bag of Models
Model Selection
Ensemble Models
Own Ensemble Model
External Models
Import Existing Models
Model Factory
…

Parameter Tuning
Parameter Optimization
Regularization
Model Size
No. Iterations
…

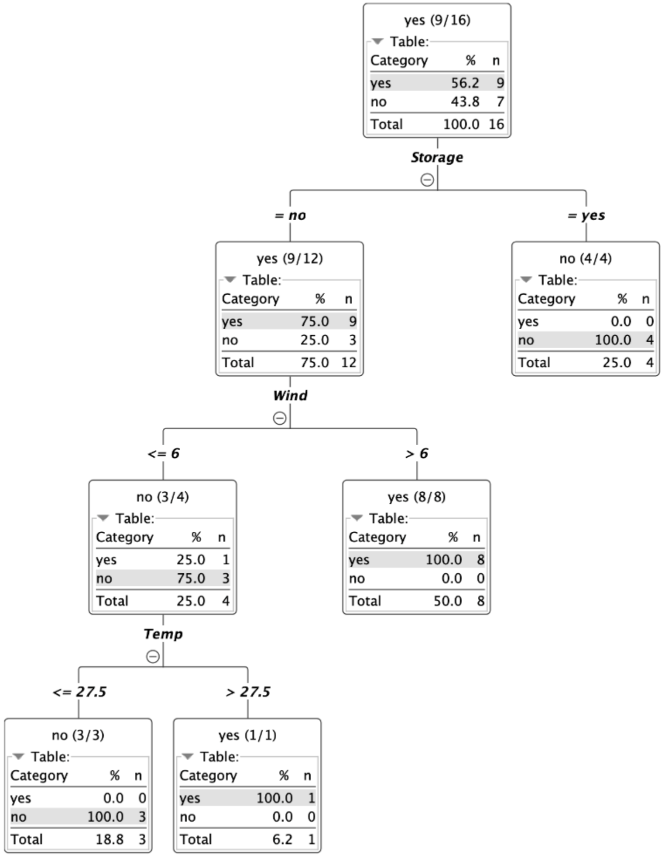
Performance Measures
Accuracy
ROC Curve
Cross-Validation
…

Files & DBs
Dashboards
REST API
SQL Code Export
Reporting
…

Decision Tree Algorithm

Goal: A Decision Tree

Outlook	Wind	Temp	Storage	Sailing
sunny	3	30	no	yes
sunny	3	25	no	no
rain	12	15	no	yes
overcast	15	2	yes	no
rain	16	25	no	yes
sunny	14	18	no	yes
rain	3	5	yes	no
sunny	9	20	no	yes
overcast	14	5	yes	no
sunny	1	7	yes	no
rain	4	25	no	no
rain	14	24	no	yes
sunny	11	20	no	yes
sunny	2	18	no	no
overcast	8	22	no	yes
overcast	13	24	no	yes



yes (9/16)

Table:

Category	%	n
yes	56.2	9
no	43.8	7
Total	100.0	16

yes (9/12)

Table:

Category	%	n
yes	75.0	9
no	25.0	3
Total	75.0	12

no (4/4)

Table:

Category	%	n
yes	0.0	0
no	100.0	4
Total	25.0	4

no (3/4)

Table:

Category	%	n
yes	25.0	1
no	75.0	3
Total	25.0	4

yes (8/8)

Table:

Category	%	n
yes	100.0	8
no	0.0	0
Total	50.0	8

no (3/3)

Table:

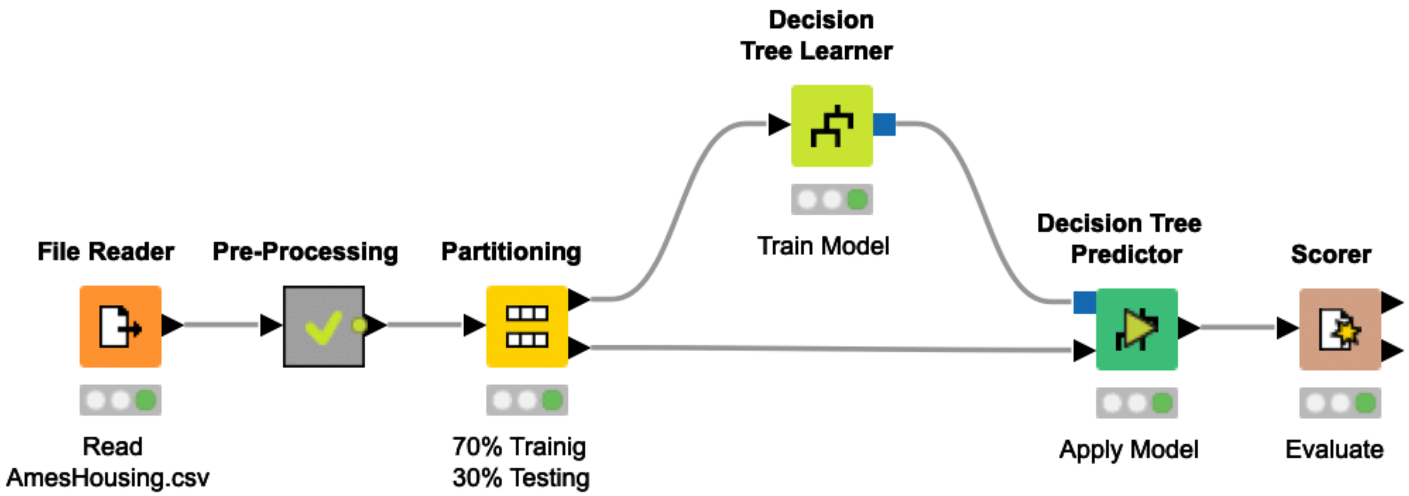
Category	%	n
yes	0.0	0
no	100.0	3
Total	18.8	3

yes (1/1)

Table:

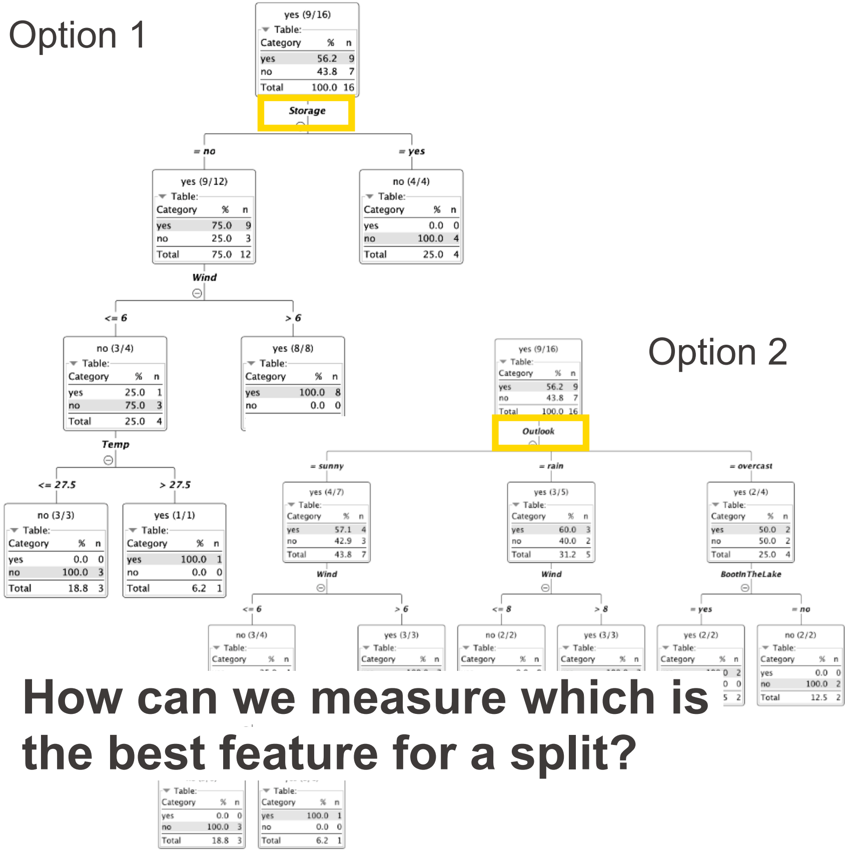
Category	%	n
yes	100.0	1
no	0.0	0
Total	6.2	1

How can we Train a Decision Tree with KNIME Analytics Platform



Goal: A Decision Tree

Outlook	Wind	Temp	Storage	Sailing
sunny	3	30	yes	yes
sunny	3	25	yes	no
rain	12	15	yes	yes
overcast	15	2	no	no
rain	16	25	yes	yes
sunny	14	18	yes	yes
rain	3	5	no	no
sunny	9	20	yes	yes
overcast	14	5	no	no
sunny	1	7	no	no
rain	4	25	yes	no
rain	14	24	yes	yes
sunny	11	20	yes	yes
sunny	2	18	yes	no
overcast	8	22	yes	yes
overcast	13	24	yes	yes



How can we measure which is the best feature for a split?

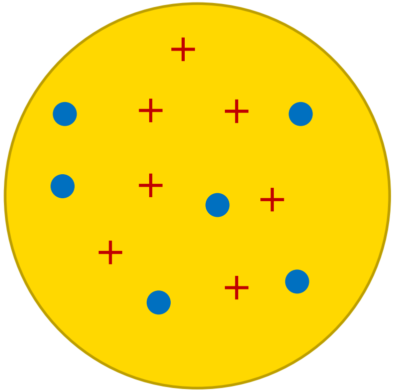
Possible Split Criterion: Gain Ratio

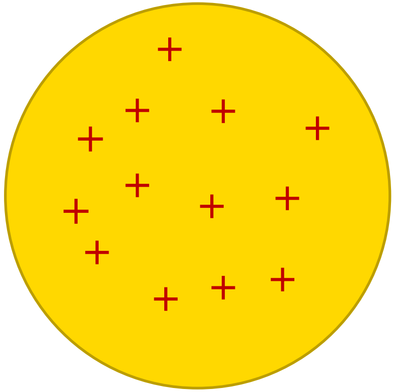
Based on entropy = measure for information / uncertainty

$$Entropy\ (p) = -\sum_{i=0}^{n} p_i\ \log_2 p_i \quad \text{for } p \in \mathbb{Q}^n$$

$p_1 = {}^{7}/_{13}$

$p_2 = {}^{6}/_{13}$





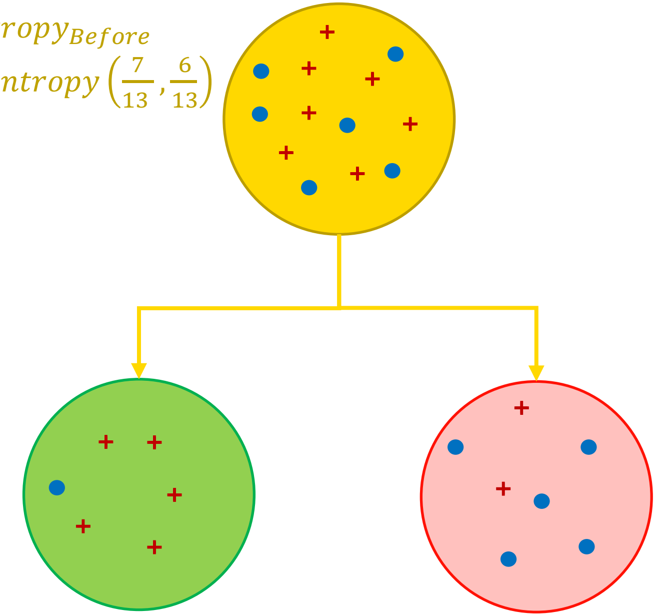
$p_1 = {}^{13}/_{13} = 1$

$p_2 = {}^{0}/_{13} = 0$

$$Entropy\ (p) = -\left({}^{7}/_{13} \log_2({}^{7}/_{13}) + {}^{6}/_{13}\ \log_2({}^{6}/_{13})\right)$$
$$= 0{,}995$$

$$Entropy\ (p) = -\left({}^{13}/_{13} \log_2({}^{13}/_{13}) + {}^{0}/_{13}\ \log_2({}^{0}/_{13})\right)$$
$$= 0$$

Possible Split Criterion: Gain Ratio



$$Entropy_{Before}$$
$$= Entropy\left(\frac{7}{13}, \frac{6}{13}\right)$$

$$Entropy_1 = Entropy\left(\frac{5}{6}, \frac{1}{6}\right)$$
$$w_1 = {}^{6}/_{13}$$

$$Entropy_2 = Entropy\left(\frac{2}{7}, \frac{5}{7}\right)$$
$$w_2 = {}^{7}/_{13}$$

Split criterion:

$$Gain = Entropy_{Before} - Entropy_{After}$$

$$Gain = Entropy_{Before} - \frac{6}{13}\, Entropy_1 - \frac{7}{13} Entropy_2$$

Next splitting feature: Feature with highest $Gain$

Problem: Favors features with many different values

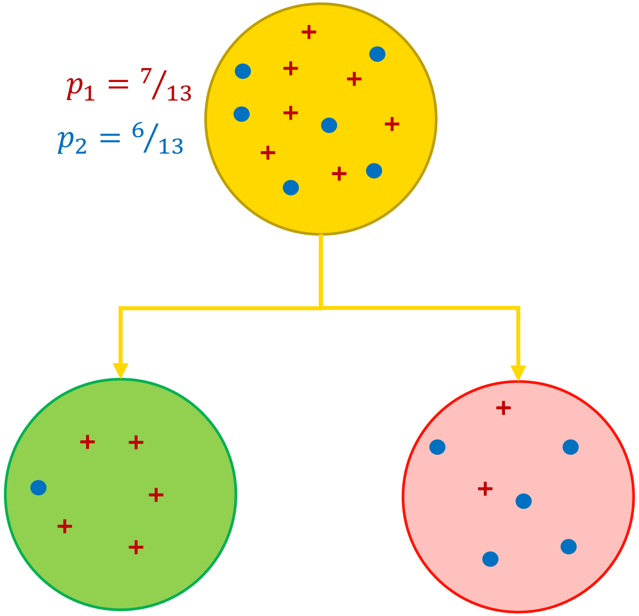
Solution: *Gain Ratio*

$$GainRatio = \frac{Gain}{SplitInfo} = \frac{Entropy_{Before} - \sum_{i=1}^{k} w_i\, Entropy_i}{\sum_{i=1}^{k} w_i \, log_2 \, w_i}$$

Possible Split Criterion: Gini Index

$p_1 = {}^{7}/_{13}$

$p_2 = {}^{6}/_{13}$



$Gini_1 = Gini\left({}^{5}/_{6,}\,{}^{1}/_{6}\right)$

$w_1 = {}^{6}/_{13}$

$Gini_2 = Gini\left({}^{2}/_{7,}\,{}^{5}/_{7}\right)$

$w_2 = {}^{7}/_{13}$

Gini index is based on Gini impurity:

$Gini(p) = 1 - \sum_{i=1}^{n} p_i^2$ for $p \in \mathbb{Q}^n$

$$Gini(p) = 1 - \frac{7^2}{13^2} - \frac{6^2}{13^2}$$

Split criterion:

$$Gini_{Index} = \sum_{i=1}^{n} w_i Gini_i$$

$$Gini_{Index} = \frac{6}{13} Gini_1 + \frac{7}{13} Gini_2$$

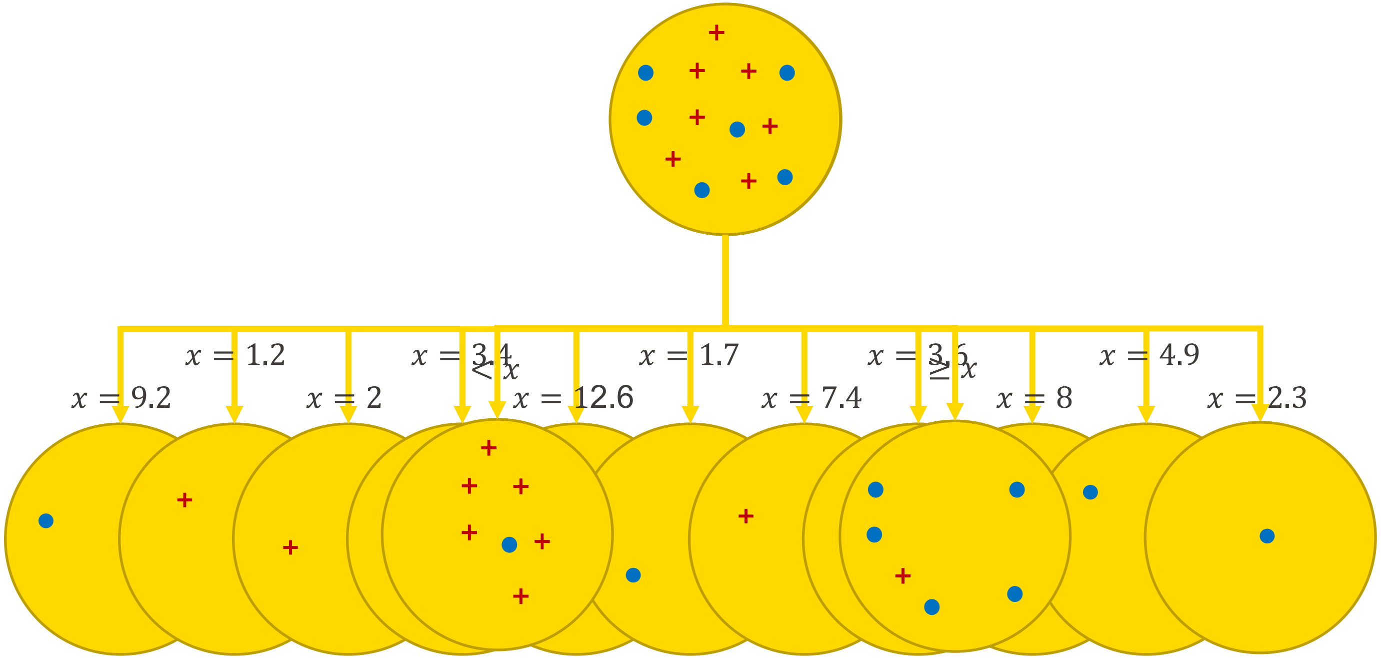
Next splitting feature:

Feature with lowest $Gini_{Index}$

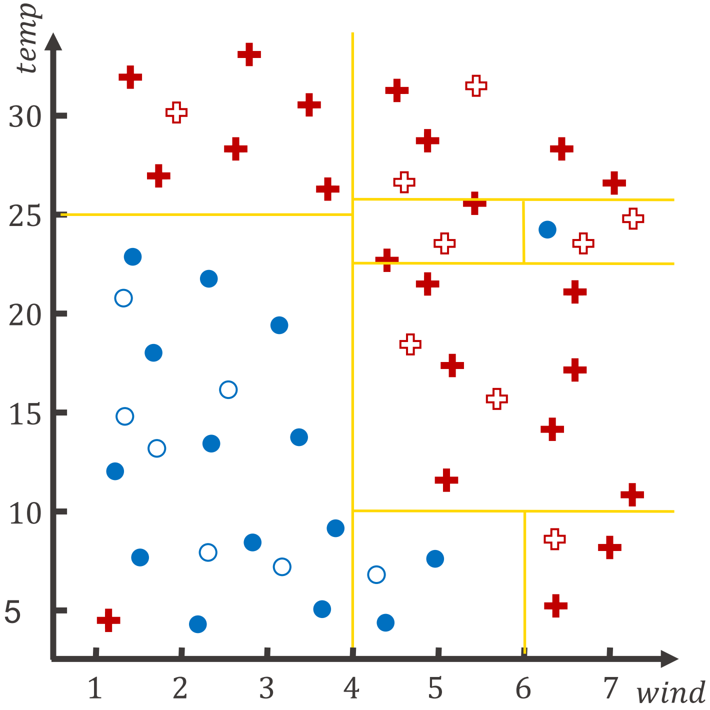
What happens for numerical Input Features?

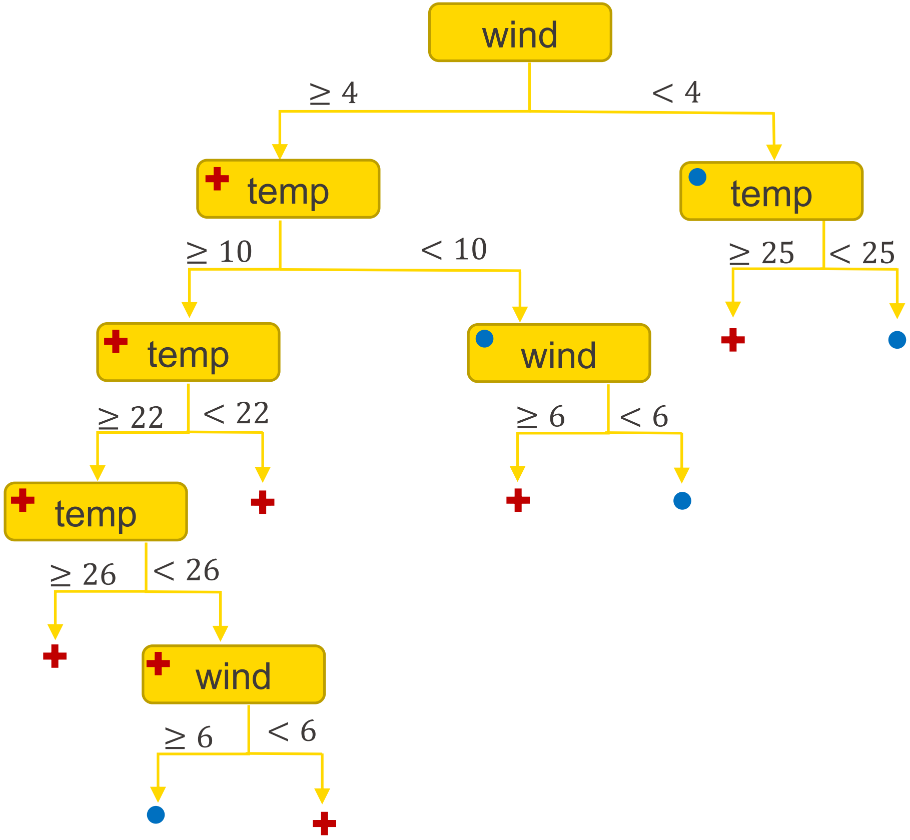
Subset for each value? – NO

Solution: Binary splits



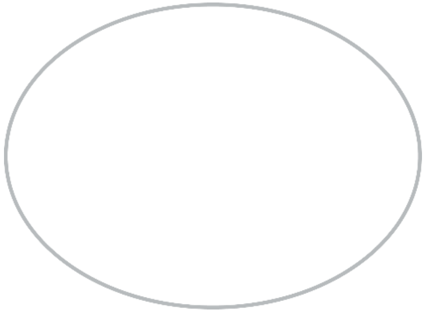
The Deeper the Better?!





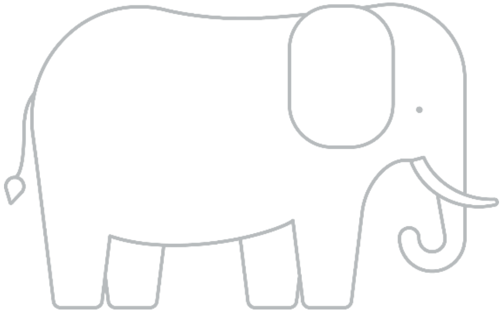
Overfitting vs Underfitting

Underfitted



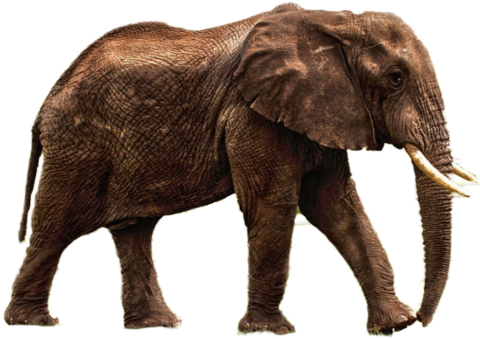
Model overlooks underlying patterns in the training set

Generalized



Model captures correlations in the training set

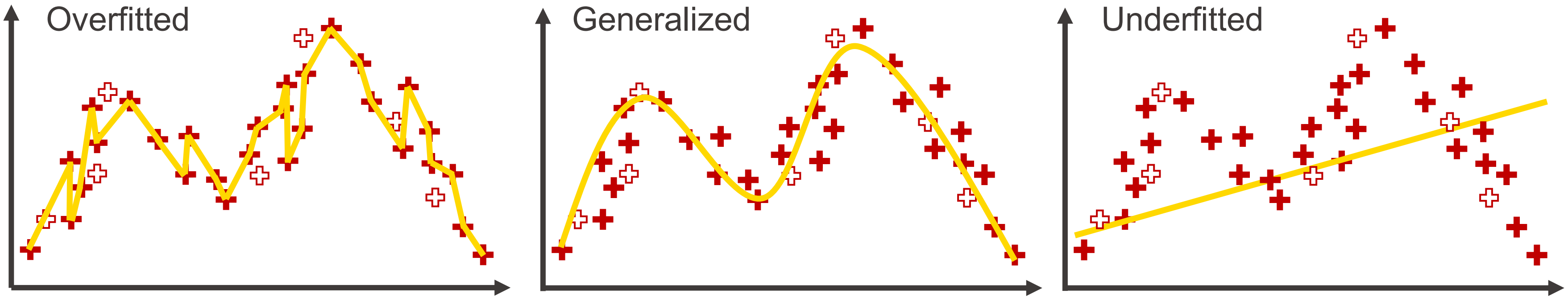
Overfitted



Model memorizes the training set rather then finding underlying patterns

Overfitting vs Underfitting

Overfitting	Underfitting
▪ Model that fits the training data too well, including details and noise ▪ Negative impact on the model's ability to generalize	▪ A model that can neither model the training data nor generalize to new data



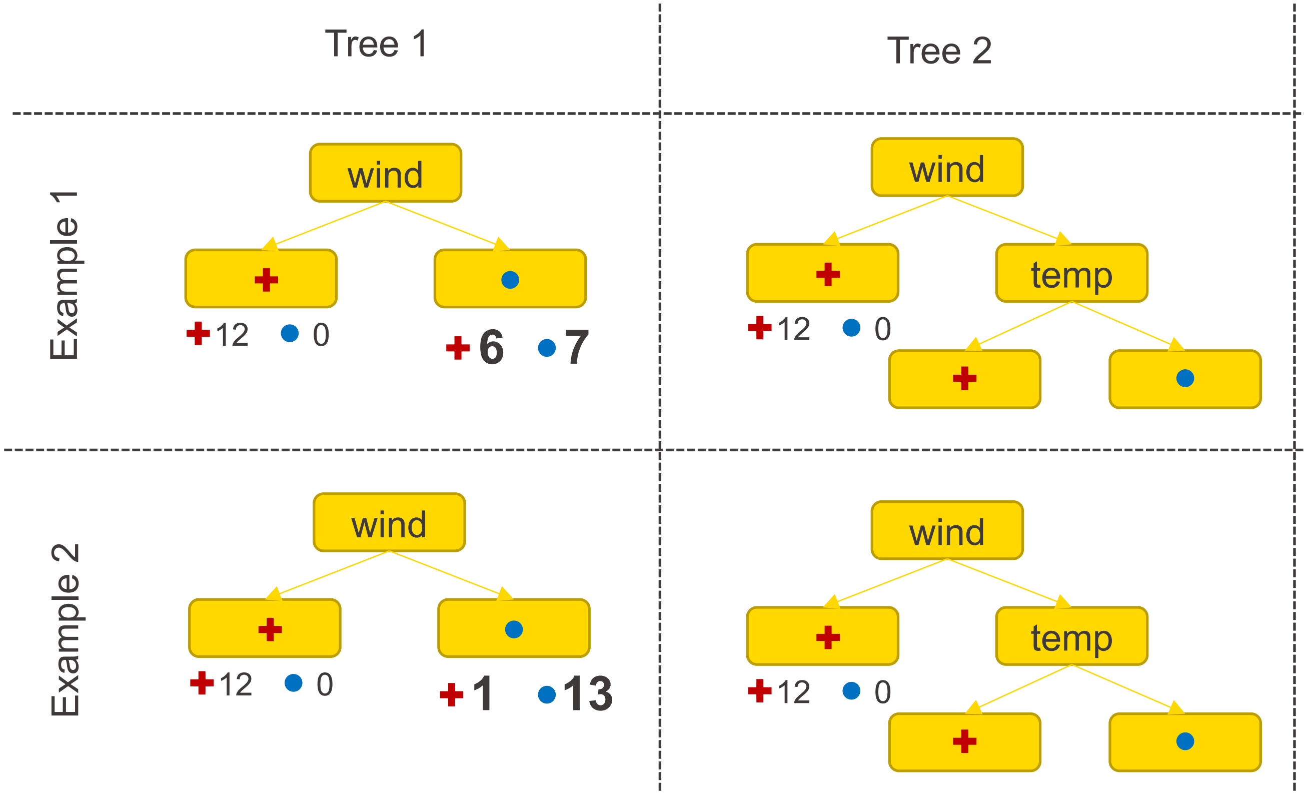
Controlling the Tree Depth

Goal: Tree that generalizes to new data and doesn’t overfit

Pruning	Early stopping
Idea: Cut branches that seem as result from overfitting	**Idea:** Define a minimum size for the tree leaves
Techniques: • Reduced Error Pruning • Minimum description length	

Pruning - Minimum Description Length Pruning (MDL)

Definition: Description length = #bits(tree) + #bits(misclassified samples)



Note

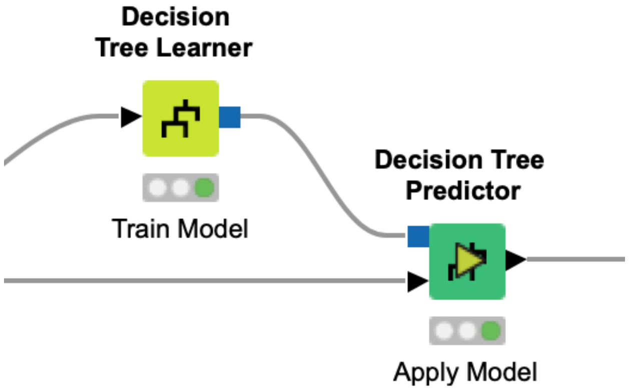
Many misclassified samples in tree 1

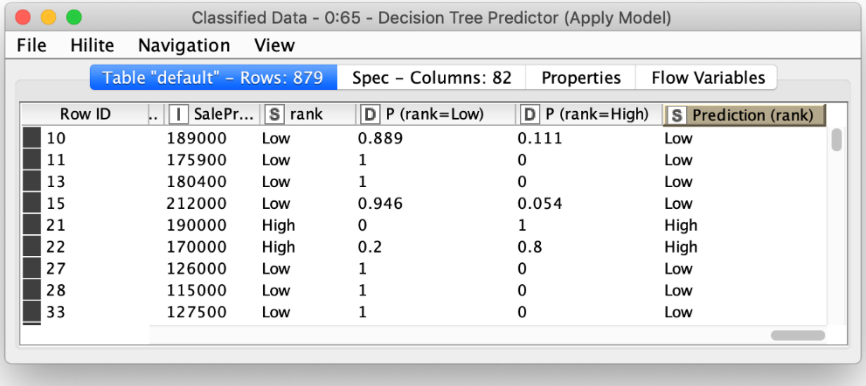
=> DL(Tree 1) > DL(Tree 2)
=> Select Tree 2

Only 1 misclassified sample in tree 1

=> DL(Tree 1) < DL(Tree 2)
=> Select Tree 1

Applying the Model – What are the Outputs?





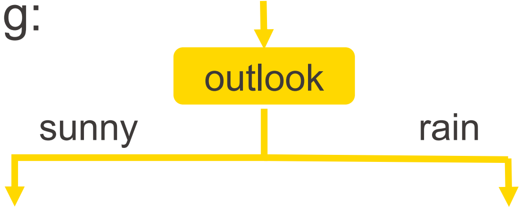
Classified Data - 0:65 - Decision Tree Predictor (Apply Model)
File
Hilite
Navigation
View
Table "default" – Rows: 879
Spec – Columns: 82
Properties
Flow Variables

Row ID	..	SalePr...	rank	P (rank=Low)	P (rank=High)	Prediction (rank)
10		189000	Low	0.889	0.111	Low
11		175900	Low	1	0	Low
13		180400	Low	1	0	Low
15		212000	Low	0.946	0.054	Low
21		190000	High	0	1	High
22		170000	High	0.2	0.8	High
27		126000	Low	1	0	Low
28		115000	Low	1	0	Low
33		127500	Low	1	0	Low

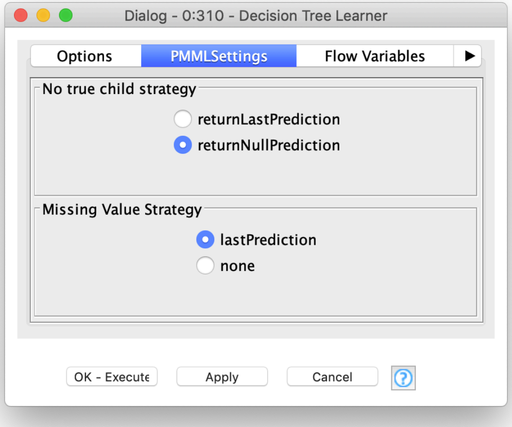
No True Child Strategy

	Outlook	Wind	Temp	Storage	Sailing
Training	sunny	3	30	yes	yes
	sunny	3	25	yes	no
	rain	12	15	yes	yes
	rain	16	25	yes	yes
	sunny	14	18	yes	yes
	rain	3	5	no	no
	sunny	9	20	yes	yes
	sunny	1	7	no	no
Testing	rain	4	25	yes	no
	rain	14	24	yes	yes
	sunny	11	20	yes	yes
	sunny	2	18	yes	no
	overcast	8	22	yes	yes
	overcast	13	24	yes	yes

Training:



What happens with outlook = overcast?


Dialog - 0:310 - Decision Tree Learner

No true child strategy

○ returnLastPrediction
● returnNullPrediction

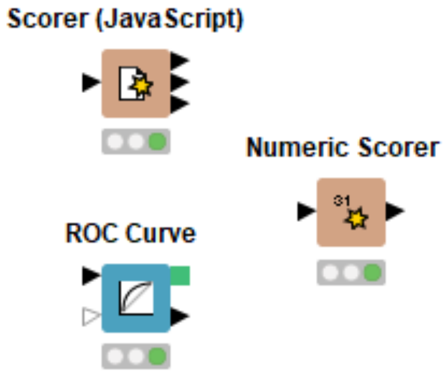
Missing Value Strategy

● lastPrediction
○ none

Evaluation of Classification Models

Evaluation Metrics

- Why evaluation metrics?
  - Quantify the power of a model
  - Compare model configurations and/or models, and select the best performing one
  - Obtain the expected performance of the model for new data
- Different model evaluation techniques are available for
  - Classification/regression models
  - Imbalanced/balanced target class distributions



Overall Accuracy

- Definition:

$$\boldsymbol{Overall\ accuracy} = \frac{\#\ \boldsymbol{Correct\ classifications\ (test\ set)}}{\#\ \boldsymbol{All\ events\ (test\ set)}}$$

- The proportion of correct classifications

- Downsides:
  - Only considers the performance in general and not for the different classes
  - Therefore, not informative when the class distribution is unbalanced

Confusion Matrix for Sailing Example

Sailing yes / no	Predicted class: yes	Predicted class: no
True class: yes	22	3
True class: no	12	328

Sailing yes / no	Predicted class: yes	Predicted class: no
True class: yes	0	25
True class: no	0	340

$$Accuracy = \frac{350}{365} = 0{,}96$$

$$Accuracy = \frac{340}{365} = 0{,}93$$

- Rows – true class values
- Columns – predicted class values
- Numbers on main diagonal – correctly classified samples
- Numbers off the main diagonal – misclassified samples

Confusion matrix

Arbitrarily define one class value as POSITIVE and the remaining class as NEGATIVE

	Predicted class positive	Predicted class negative
True class positive	TRUE POSITIVE	FALSE NEGATIVE
True class negative	FALSE POSITIVE	TRUE NEGATIVE

TRUE POSITIVE (**TP**): Actual and predicted class is positive

TRUE NEGATIVE (**TN**): Actual and predicted class is negative

FALSE NEGATIVE (**FN**): Actual class is positive and predicted negative

FALSE POSITIVE (**FP**): Actual class is negative and predicted positive

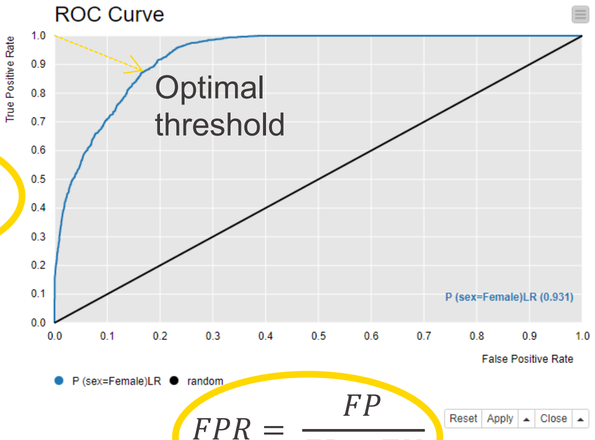
Use these four statistics to calculate other evaluation metrics, such as overall accuracy, true positive rate, and false positive rate

ROC Curve

- The ROC Curve shows the false positive rate and true positive rate for different threshold values
  - False positive rate (FPR)
    - negative events **in**correctly classified as positive
  - True positive rate (TPR)
    - positive events correctly classified as positive

	Predicted class positive	Predicted class negative
True class positive	True Positive (TP)	False Negative (FN)
True class negative	False Positive (FP)	True Negative (TN)

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



Optimal threshold

$$FPR = \frac{FP}{FP + TN}$$

Cohen‘s Kappa (κ) vs. Overall accuracy

	Positive	Negative
Positive	14	6
Negative	5	75

$$p_{e1} = \frac{19}{100} \times \frac{20}{100}$$

$$p_{e2} = \frac{81}{100} \times \frac{80}{100}$$

$$p_e = p_{e1} + p_{e2} = 0.686$$

$$p_0 = \frac{89}{100} = 0.89$$

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = \frac{0.204}{0.314} \approx 0.65$$

Switch TP and FP

	Positive	Negative
Positive	6	14
Negative	5	75

$$p_{e1} = \frac{11}{100} \times \frac{20}{100}$$

$$p_{e2} = \frac{89}{100} \times \frac{80}{100}$$

$$p_e = p_{e1} + p_{e2} = 0.734$$

$$p_0 = \frac{81}{100} = 0.81$$

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = \frac{0.076}{0.266} = 0.29$$

Overall accuracy

κ = 1: perfect model performance
κ = 0: the model performance is equal to a random classifier

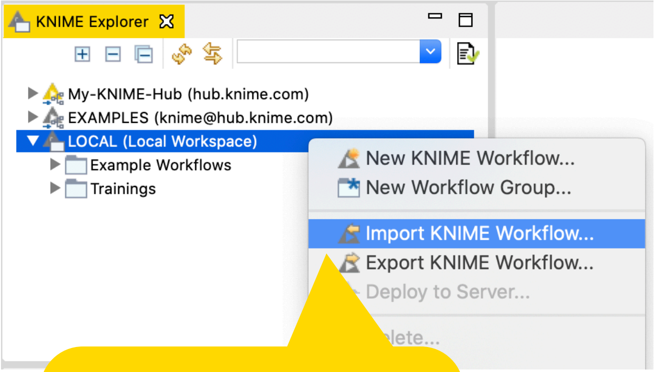
Exercise: 01_Training_a_Decision_Tree_Model

- Dataset: Sales data of individual residential properties in Ames, Iowa from 2006 to 2010.
- One of the columns is the overall condition ranking, with values between 1 and 10.
- Goal: train a binary classification model, which can predict whether the overall condition is high or low.

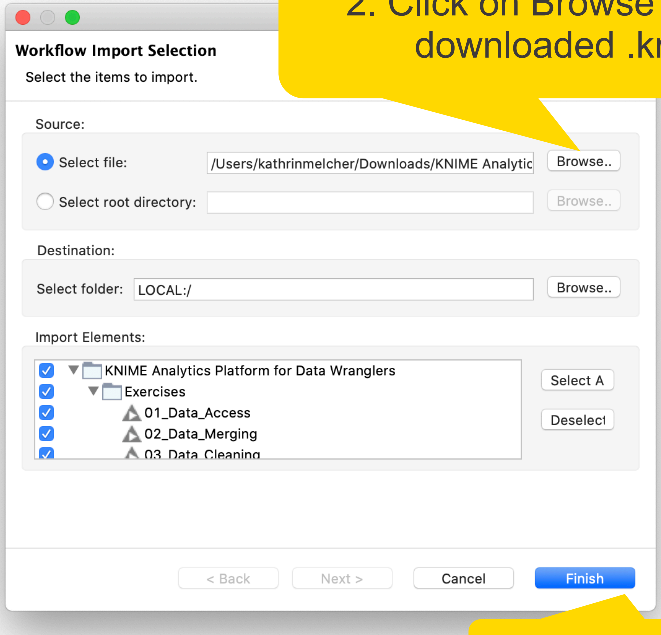
You can download the training workflows from the KNIME Hub:
https://hub.knime.com/knime/spaces/Education/latest/Courses/

Exercise Session 1

- Import the course material to KNIME Analytics Platform



1. Right click on LOCAL and select Import KNIME Workflow….



Workflow Import Selection

Select the items to import.

Source:

● Select file: /Users/kathrinmelcher/Downloads/KNIME Analytic

○ Select root directory:

Destination:

Select folder: LOCAL:/

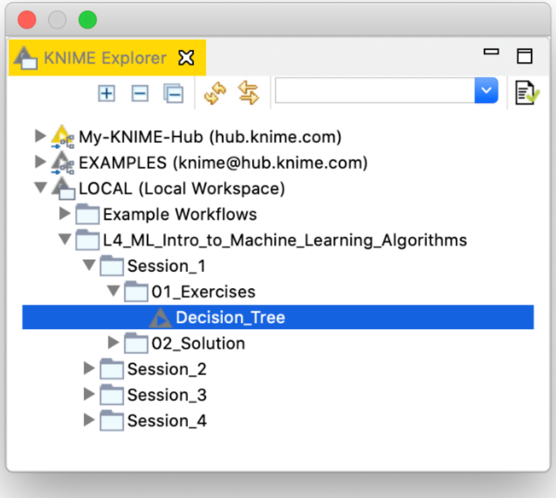
Import Elements:

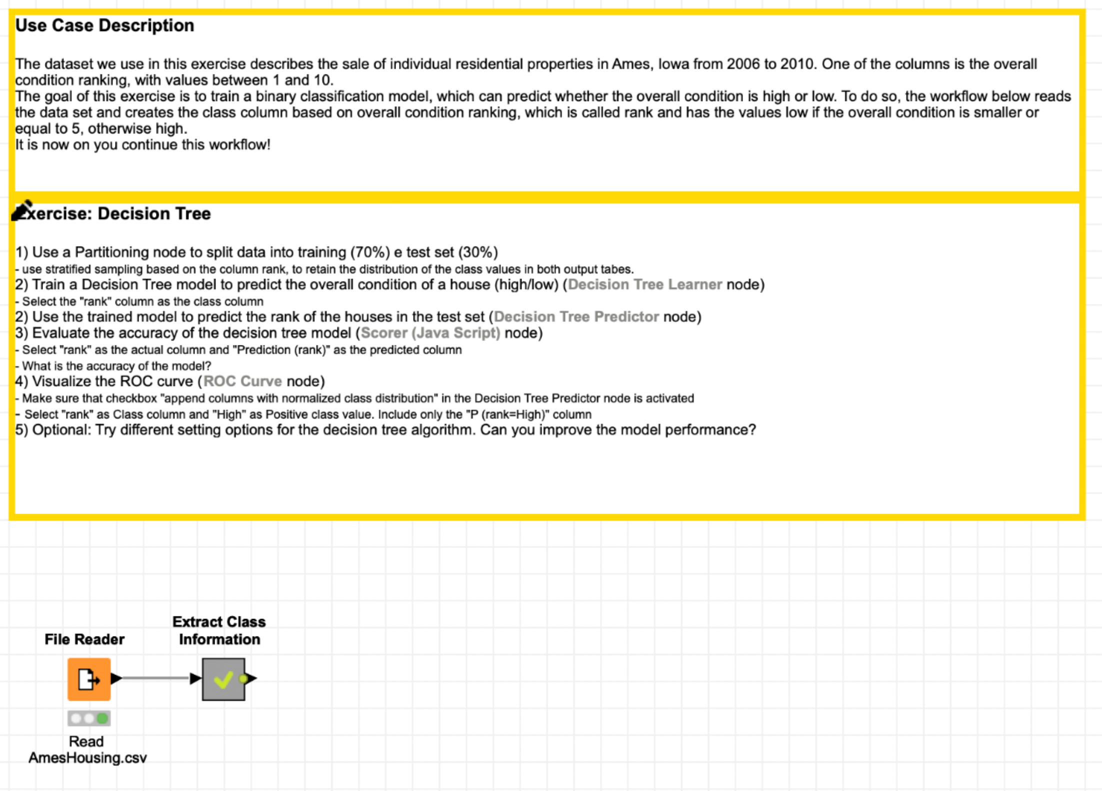
- ☑ KNIME Analytics Platform for Data Wranglers
  - ☑ Exercises
    - ☑ 01_Data_Access
    - ☑ 02_Data_Merging
    - ☑ 03_Data_Cleaning

2. Click on Browse and select downloaded .knar file

3. Click on Finish

Exercise: Decision_Tree





Use Case Description

The dataset we use in this exercise describes the sale of individual residential properties in Ames, Iowa from 2006 to 2010. One of the columns is the overall condition ranking, with values between 1 and 10.
The goal of this exercise is to train a binary classification model, which can predict whether the overall condition is high or low. To do so, the workflow below reads the data set and creates the class column based on overall condition ranking, which is called rank and has the values low if the overall condition is smaller or equal to 5, otherwise high.
It is now on you continue this workflow!

Exercise: Decision Tree

1) Use a Partitioning node to split data into training (70%) e test set (30%)
- use stratified sampling based on the column rank, to retain the distribution of the class values in both output tabes.
2) Train a Decision Tree model to predict the overall condition of a house (high/low) (Decision Tree Learner node)
- Select the "rank" column as the class column
2) Use the trained model to predict the rank of the houses in the test set (Decision Tree Predictor node)
3) Evaluate the accuracy of the decision tree model (Scorer (Java Script) node)
- Select "rank" as the actual column and "Prediction (rank)" as the predicted column
- What is the accuracy of the model?
4) Visualize the ROC curve (ROC Curve node)
- Make sure that checkbox "append columns with normalized class distribution" in the Decision Tree Predictor node is activated
- Select "rank" as Class column and "High" as Positive class value. Include only the "P (rank=High)" column
5) Optional: Try different setting options for the decision tree algorithm. Can you improve the model performance?

Session II: Regression Models, Ensemble Models & Logistic Regression

Regression Problems

Regression Analysis

- Prediction of numerical target values

- Commonality with models for classification
  - First, construct a model
  - Second, use model to predict unknown value
    - Major method for prediction is regression in all its flavors
      - Simple and multiple regression
      - Linear and non-linear regression

- Difference from classification
  - Classification aims at predicting categorical class label
  - Regression models aim at predicting values from continuous-valued functions

Regression

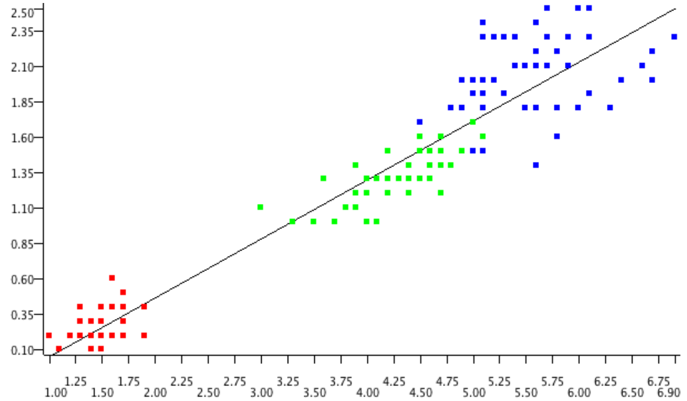
Predict *numeric* outcomes on existing data (supervised)

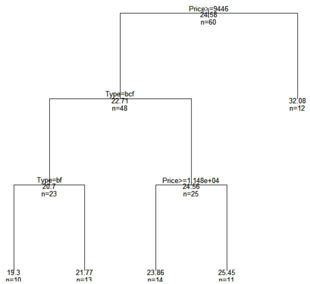
Applications

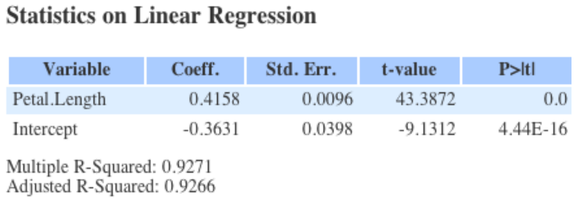
- Forecasting
- Quantitative Analysis

Methods

- Linear
- Polynomial
- Regression Trees
- Partial Least Squares







Statistics on Linear Regression

Variable	Coeff.	Std. Err.	t-value	P>|t|
Petal.Length	0.4158	0.0096	43.3872	0.0
Intercept	-0.3631	0.0398	-9.1312	4.44E-16

Multiple R-Squared: 0.9271
Adjusted R-Squared: 0.9266

Linear Regression Algorithm

Linear Regression

Predicts the values of the target variable y
based on a linear combination of
the values of the input feature(s) x_j

Two input features: $\hat{y} = a_0 + a_1x_1 + a_2x_2$

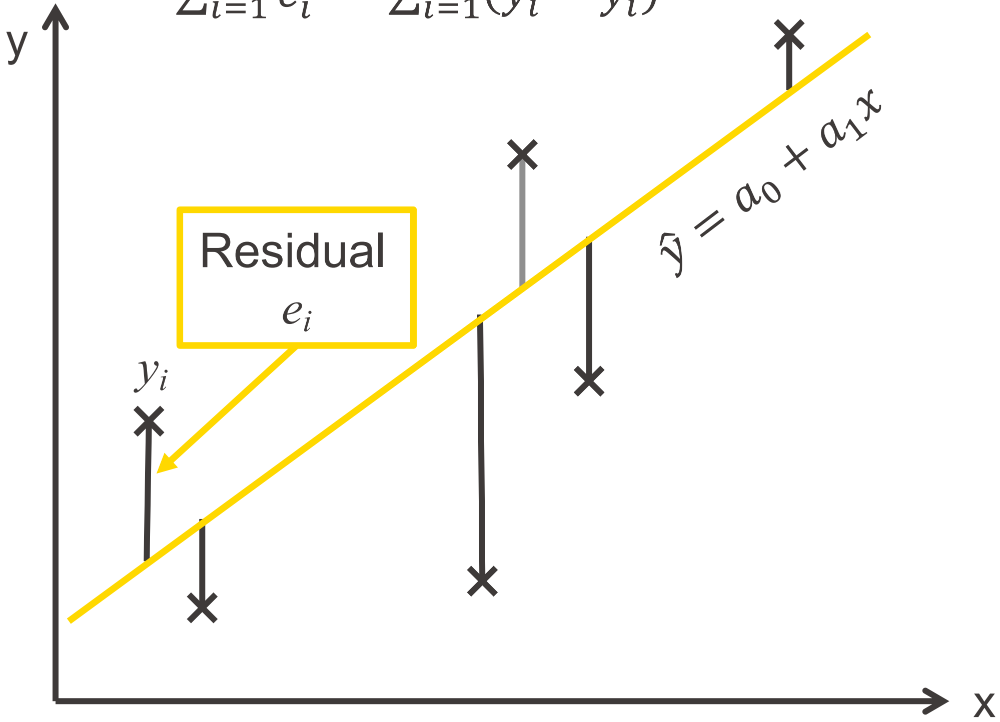
p input features: $\hat{y} = a_0 + a_1x_1 + a_2x_2 + \cdots + a_px_p$

- Simple regression: one input feature → regression line
- Multiple regression: several input features → regression hyper-plane
- Residuals: differences between observed and predicted values (errors)
  Use the residuals to measure the model fit

Simple Linear Regression

Optimization goal: minimize sum of squared residuals

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \widehat{y_i})^2$$



Simple Linear Regression

- Think of a straight line $\hat{y} = f(x) = a + bx$
- Find a and b to model all observations (x_i, y_i) as close as possible
- ➔ SSE $F(a, b) = \sum_{i=1}^{n}(f(x) - y_i)^2 = \sum_{i=1}^{n}(a + bx_i - y_i)^2$ should be minimal
- That is:

$$\frac{\partial F}{\partial a} = \sum_{i=1}^{n} 2(a + bx_i - y_i) = 0$$

$$\frac{\partial F}{\partial b} = \sum_{i=1}^{n} 2(a + bx_i - y_i)\, x_i = 0$$

- ➔ A unique solution exists for a and b

Linear Regression

- Optimization goal: minimize the squared residuals

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{n} a_j x_{j,i}\right)^2 = (y - aX)^T (y - aX)$$

- Solution:

$$\hat{a} = (X^T X)^{-1} X^T y$$

Linear Regression Learner

- Computational issues:
  - $X^T X$ must have full rank, and thus be invertible
    (Problems arise if linear dependencies between input features exist)
  - Solution may be unstable, if input features are almost linearly dependent

Linear Regression: Summary

- Positive:
  - Strong mathematical foundation
  - Simple to calculate and to understand
    (For moderate number of dimensions)
  - High predictive accuracy
    (In many applications)

- Negative:
  - Many dependencies are non-linear
    (Can be generalized)
  - Model is global and cannot adapt well to locally different data distributions
    But: Locally weighted regression, CART

Polynomial Regression

Predicts the values of the target variable y
based on a polynomial combination of degree d of
the values of the input feature(s) x_j

$$\tilde{y} = a_0 + \sum_{j=1}^{p} a_{j,1} x_j + \sum_{j=1}^{p} a_{j,2} x_j^{2} + \cdots + \sum_{j=1}^{p} a_{j,d} x_j^{d}$$

- Simple regression: one input feature → regression curve
- Multiple regression: several input features → regression hypersurface
- Residuals: differences between observed and predicted values (errors)
  Use the residuals to measure the model fit

Evaluation of Regression Models

Numeric Errors: Formulas

Error Metric	Formula	Notes
R-squared	$1-\frac{\sum_{i=1}^{n}(y_i-f(x_i))^2}{\sum_{i=1}^{n}(y_i-\overline{y})^2}$	Universal range: the closer to 1 the better
Mean absolute error (MAE)	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i-f(x_i)\rvert$	Equal weights to all distances Same unit as the target column
Mean squared error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i-f(x_i))^2$	Common loss function
Root mean squared error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-f(x_i))^2}$	Weights big differences more Same unit as the target column
Mean signed difference	$\frac{1}{n}\sum_{i=1}^{n}(y_i-f(x_i))$	Only informative about the direction of the error
Mean absolute percentage error (MAPE)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i-f(x_i)\rvert}{\lvert y_i\rvert}$	Requires non-zero target column values

MAE (Mean Absolute Error) vs. RMSE (Root Mean Squared Error)

MAE	RMSE
Easy to interpret – mean average absolute error	Cannot be directly interpreted as the average error
All errors are equally weighted	Larger errors are weighted more
Generally smaller than RMSE	Ideal when large deviations need to be avoided

Example:

Actual values = [2,4,5,8],

Case 1: Predicted Values = [4, 6, 8, 10]

Case 2: Predicted Values = [4, 6, 8, 14]

	MAE	RMSE
Case 1	2.25	2.29
Case 2	3.25	3.64

R-squared vs. RMSE

R-squared	RMSE
Relative measure: Proportion of variability explained by the model	**Absolute measure**: How much deviation at each point
Range: 0 (no variability explained) to 1 (all variability explained)	Same scale as the target

Example:

Actual values = [2,4,5,8],

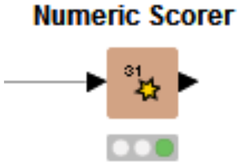
Case 1: Predicted Values = [3, 4, 5, 6]

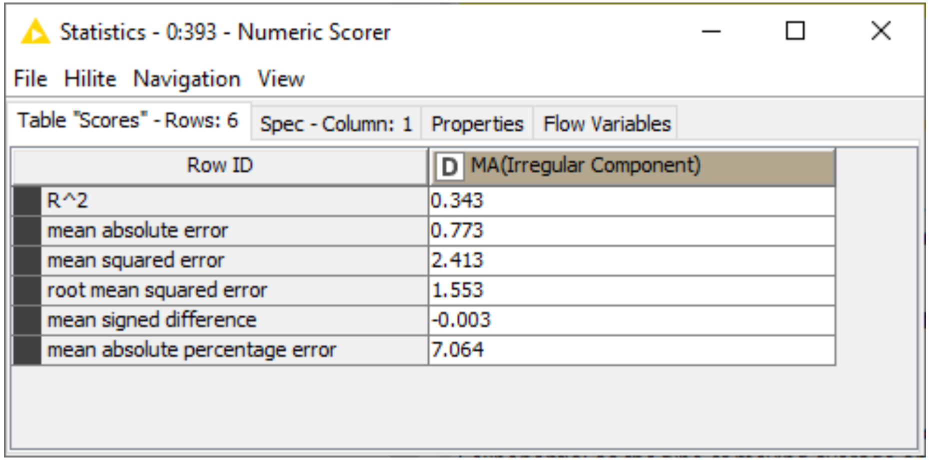
Case 2: Predicted Values = [3, 3, 7, 7]

	R-sq	RMSE
Case 1	0.96	1.12
Case 2	0.65	1.32

Numeric Scorer

- Similar to scorer node, but for nodes with *numeric* predictions
- Compare dependent variable values to predicted values to evaluate model quality.
- Report R^2, RMSE, MAPE, etc.

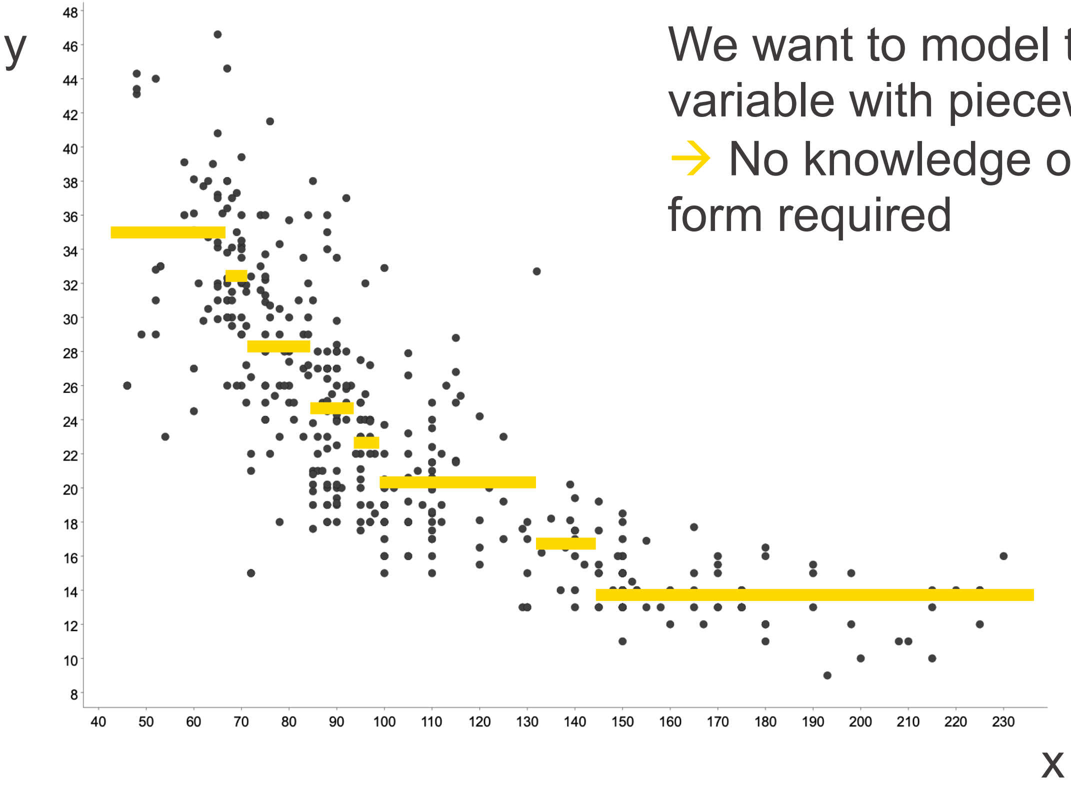




Row ID	D MA(Irregular Component)
R^2	0.343
mean absolute error	0.773
mean squared error	2.413
root mean squared error	1.553
mean signed difference	-0.003
mean absolute percentage error	7.064

Regression Tree

Regression Tree: Goal



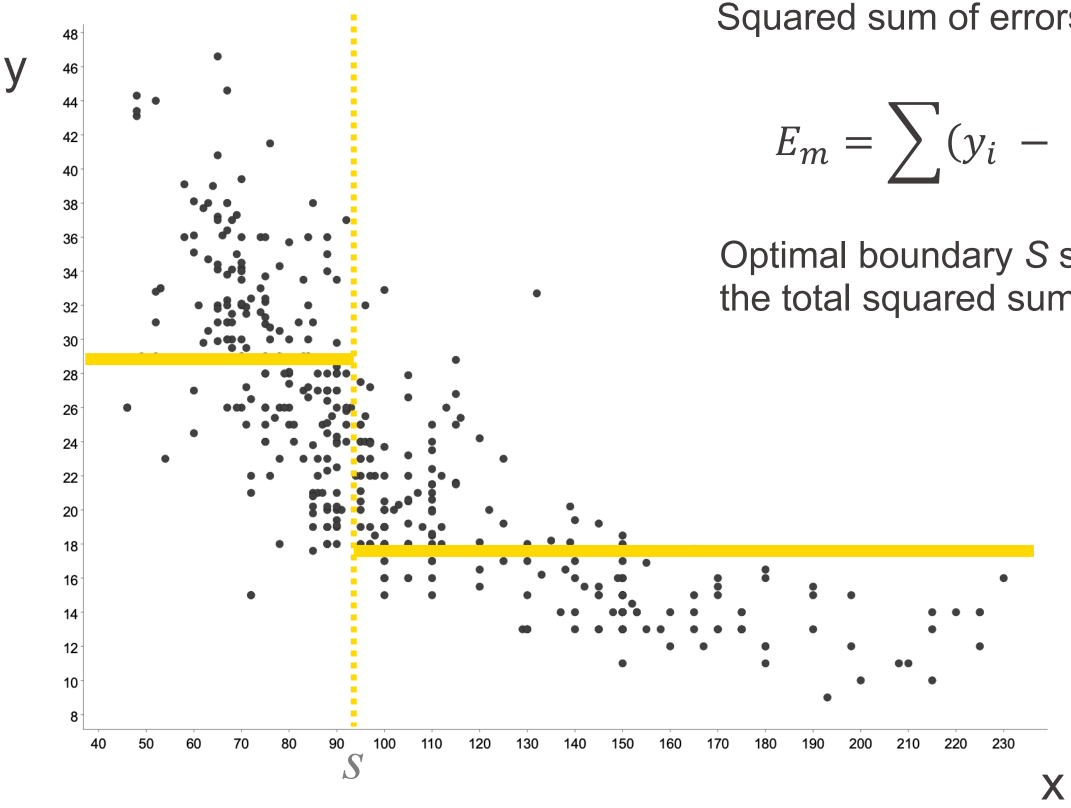
We want to model the target variable with piecewise lines
→ No knowledge of functional form required

Regression Tree: Initial Split

Local mean:

$$c_m = \frac{1}{n}\sum y_i$$

For observations in segment *m*



Squared sum of errors:

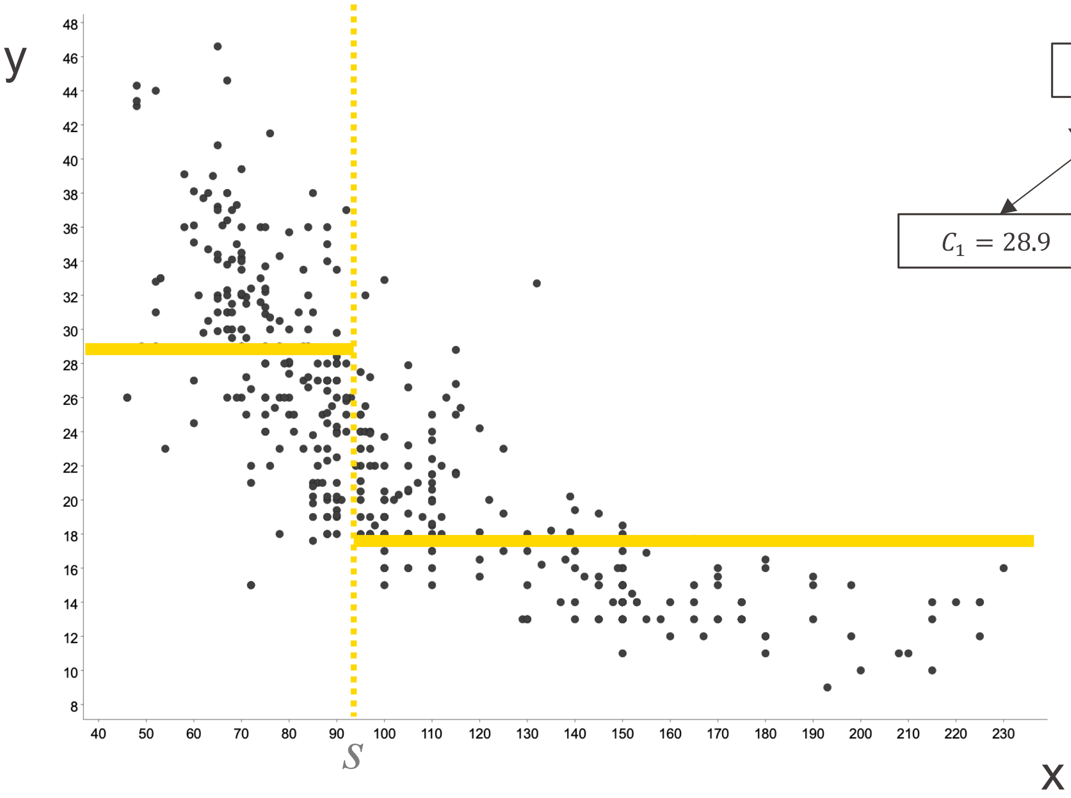
$$E_m = \sum(y_i - c_m)^2$$

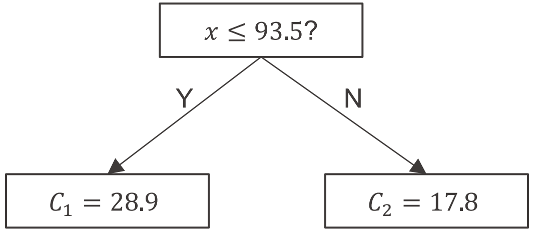
Optimal boundary *S* should minimize the total squared sum:

$$\sum E_m$$

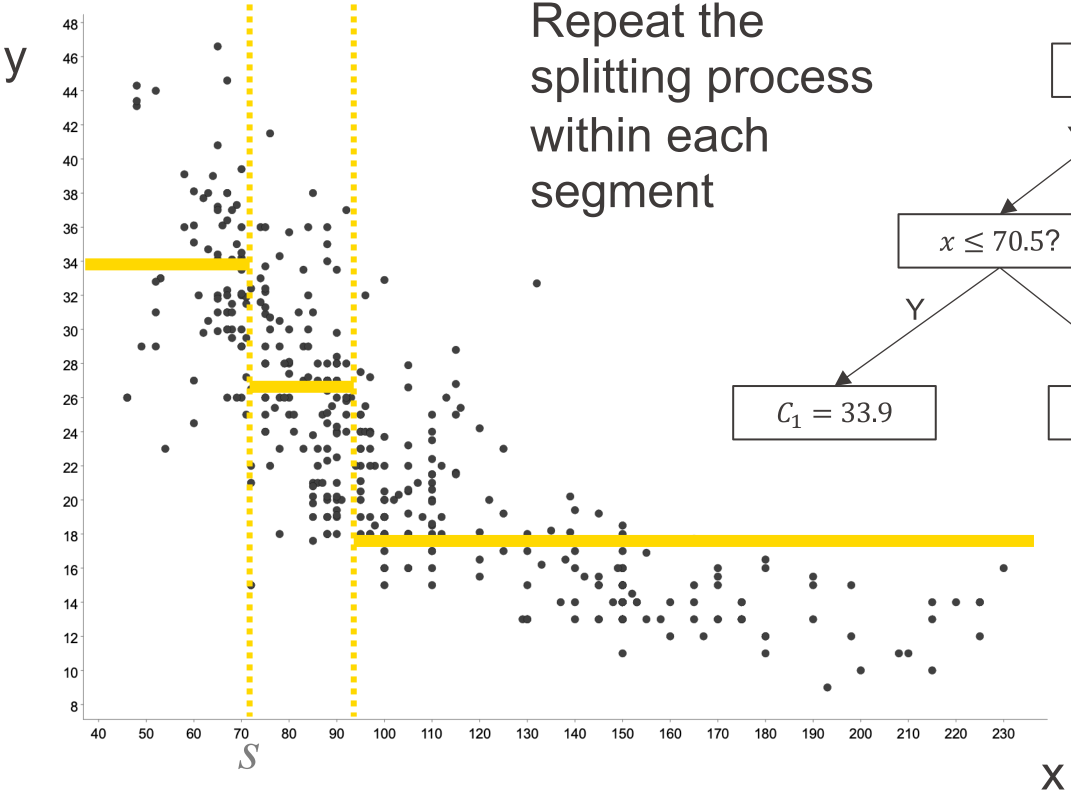
For all segments *m*

Regression Tree: Initial Split

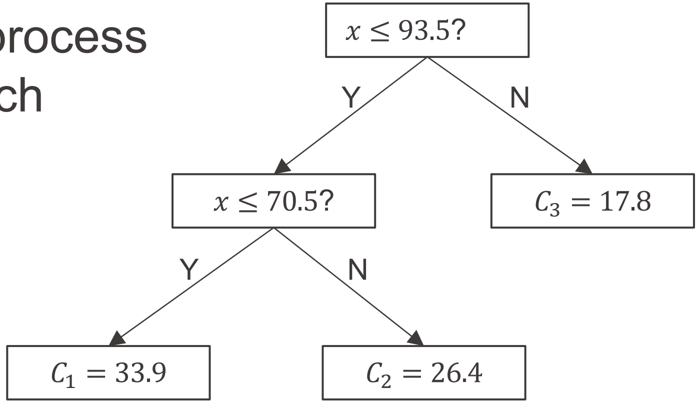




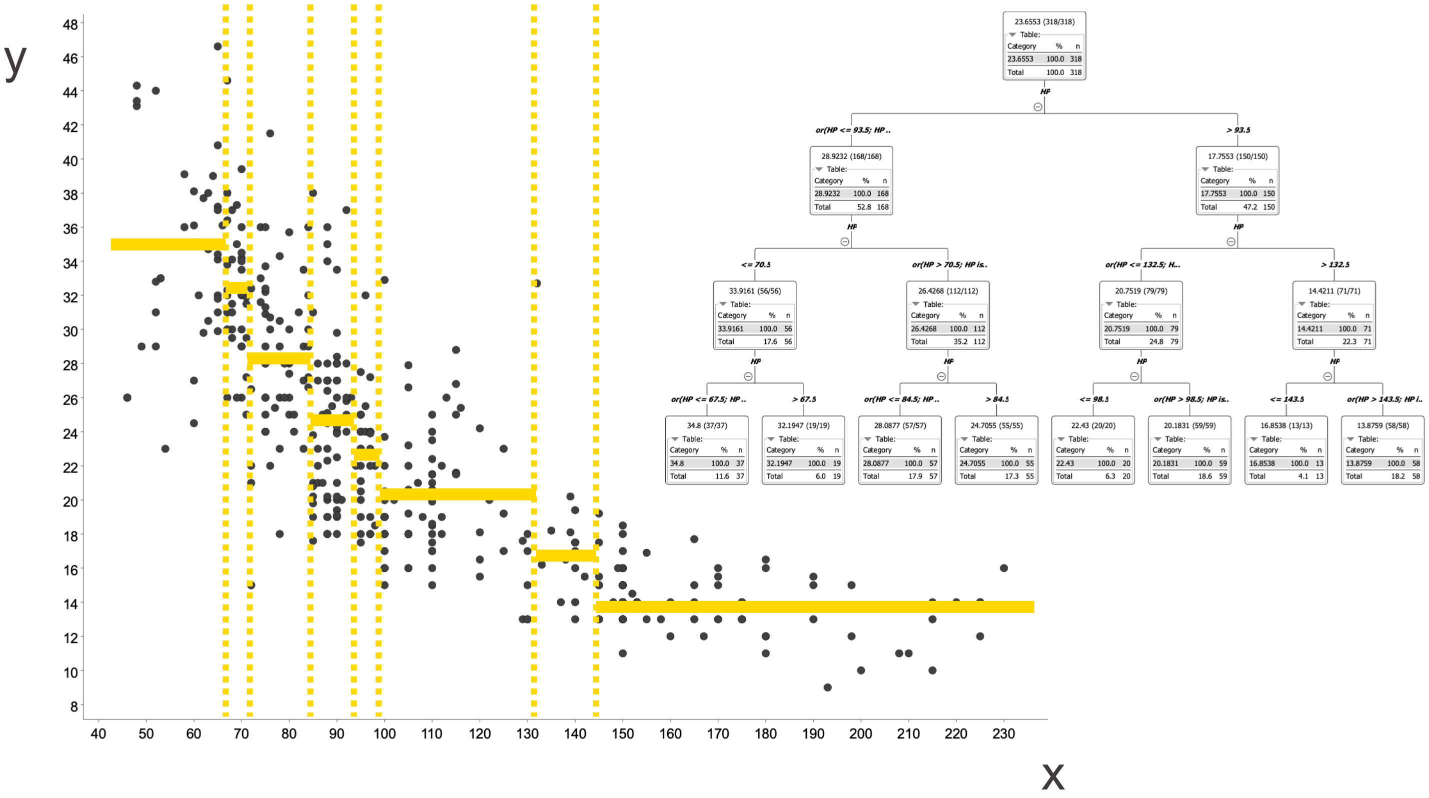
Regression Tree: Growing the Tree



Repeat the splitting process within each segment



Regression Tree: Final Model



23.6553 (318/318)

Table:

Category	%	n
23.6553	100.0	318
Total	100.0	318

28.9232 (168/168)

Table:

Category	%	n
28.9232	100.0	168
Total	52.8	168

17.7553 (150/150)

Table:

Category	%	n
17.7553	100.0	150
Total	47.2	150

33.9161 (56/56)

Table:

Category	%	n
33.9161	100.0	56
Total	17.6	56

26.4268 (112/112)

Table:

Category	%	n
26.4268	100.0	112
Total	35.2	112

20.7519 (79/79)

Table:

Category	%	n
20.7519	100.0	79
Total	24.8	79

14.4211 (71/71)

Table:

Category	%	n
14.4211	100.0	71
Total	22.3	71

34.8 (37/37)

Table:

Category	%	n
34.8	100.0	37
Total	11.6	37

32.1947 (19/19)

Table:

Category	%	n
32.1947	100.0	19
Total	6.0	19

28.0877 (57/57)

Table:

Category	%	n
28.0877	100.0	57
Total	17.9	57

24.7055 (55/55)

Table:

Category	%	n
24.7055	100.0	55
Total	17.3	55

22.43 (20/20)

Table:

Category	%	n
22.43	100.0	20
Total	6.3	20

20.1831 (59/59)

Table:

Category	%	n
20.1831	100.0	59
Total	18.6	59

16.8538 (13/13)

Table:

Category	%	n
16.8538	100.0	13
Total	4.1	13

13.8759 (58/58)

Table:

Category	%	n
13.8759	100.0	58
Total	18.2	58

Regression Tree: Algorithm

Start with a single node containing all points.

1. Calculate c_i and E_i.
2. If all points have the same value for feature x_j, stop.
3. Otherwise, find the best binary splits that reduces $E_{j,s}$ as much as possible.
   - $E_{j,s}$ doesn’t reduce as much → stop
   - A node contains less than the minimum node size → stop
   - Otherwise, take that split, creating two new nodes.
   - In each new node, go back to step 1.

Regression Trees: Summary

- Differences to decision trees:
  - Splitting criterion: minimizing intra-subset variation (error)
  - Pruning criterion: based on numeric error measure
  - Leaf node predicts average target values of training instances reaching that node

- Can approximate piecewise constant functions
- Easy to interpret

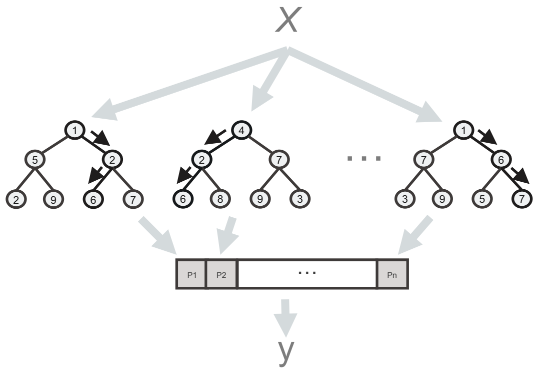
Regression Trees: Pros & Cons

- Finding of (local) regression values (average)
- Problems:
  - No interpolation across borders
  - Heuristic algorithm: unstable and not optimal.
- Extensions:
  - Fuzzy trees (better interpolation)
  - Local models for each leaf (linear, quadratic)

Ensemble Models

Tree Ensemble Models

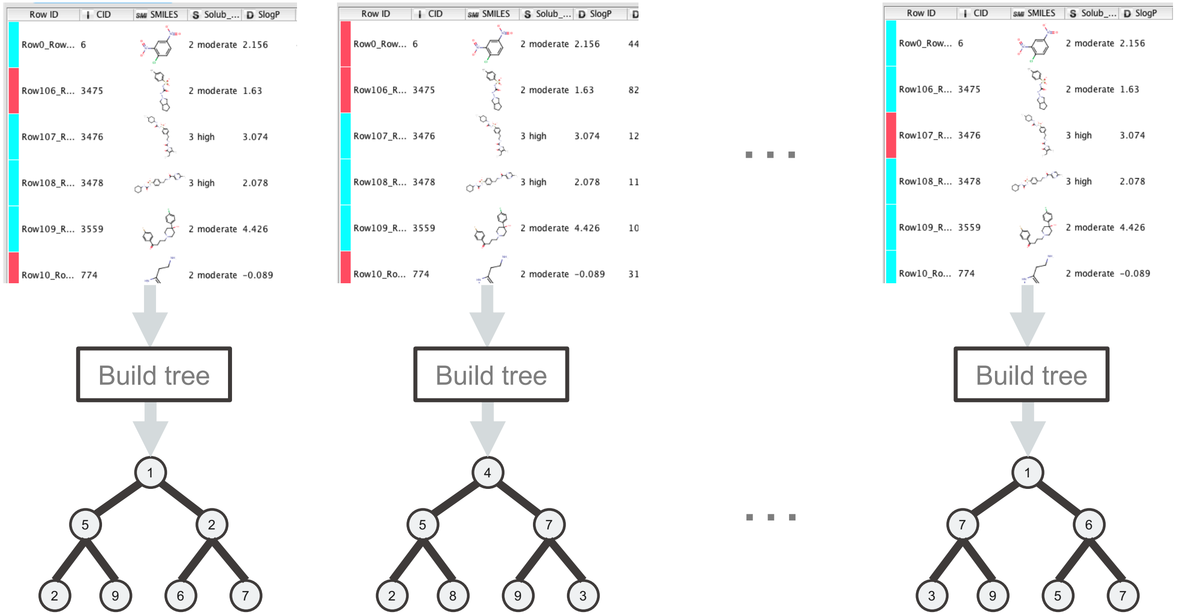
- General idea: take advantage of the "wisdom of the crowd"
- Ensemble models: Combining predictions from many predictors, e.g. decision trees
- Leads to a more accurate and robust model
- Model is difficult to interpret
  - There are multiple trees in the model



Typically for classification, the individual models vote and the majority wins; for regression, the individual predictions are averaged

Bagging - Idea

- One option is ”bagging” (Bootstrap AGGregatING)
- For each tree / model a training set is generated by sampling uniformly with replacement from the standard training set



	Row ID	CID	SMILES	Solub_...	SlogP
🟦	Row0_Row...	6		2 moderate	2.156
🟥	Row106_R...	3475		2 moderate	1.63
🟦	Row107_R...	3476		3 high	3.074
🟦	Row108_R...	3478		3 high	2.078
🟦	Row109_R...	3559		2 moderate	4.426
🟥	Row10_Ro...	774		2 moderate	-0.089

	Row ID	CID	SMILES	Solub_...	SlogP	D
🟥	Row0_Row...	6		2 moderate	2.156	44
🟥	Row106_R...	3475		2 moderate	1.63	82
🟦	Row107_R...	3476		3 high	3.074	12
🟦	Row108_R...	3478		3 high	2.078	11
🟦	Row109_R...	3559		2 moderate	4.426	10
🟥	Row10_Ro...	774		2 moderate	-0.089	31

	Row ID	CID	SMILES	Solub_...	SlogP
🟦	Row0_Row...	6		2 moderate	2.156
🟦	Row106_R...	3475		2 moderate	1.63
🟥	Row107_R...	3476		3 high	3.074
🟦	Row108_R...	3478		3 high	2.078
🟦	Row109_R...	3559		2 moderate	4.426
🟦	Row10_Ro...	774		2 moderate	-0.089

Example for Bagging

Full training set

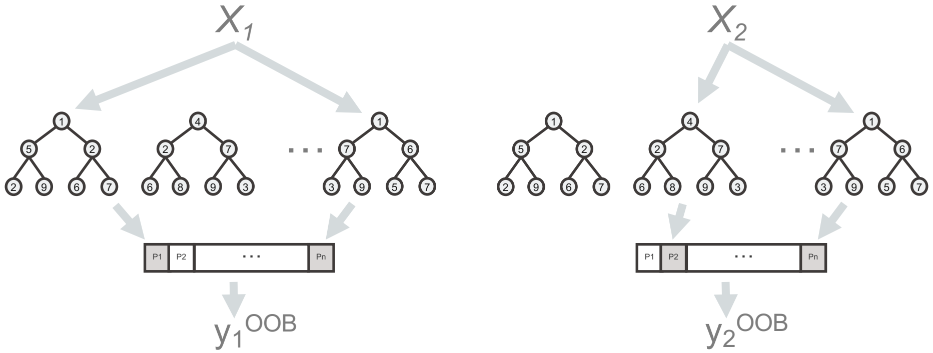
RowID	x_1	x_2	y
Row_1	2	6	Class 1
Row_2	4	1	Class 2
Row_3	9	3	Class 2
Row_4	2	7	Class 1
Row_5	8	1	Class 2
Row_6	2	6	Class 1
Row_7	5	2	Class 2

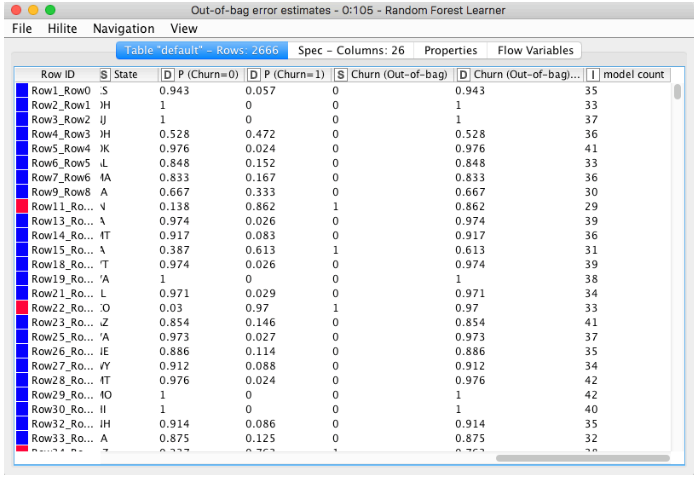
Sampled training set

RowID	x_1	x_2	y
Row_3	9	3	Class 2
Row_6	2	6	Class 1
Row_1	2	6	Class 1
Row_3	9	3	Class 2
Row_5	8	1	Class 2
Row_6	2	6	Class 1
Row_1	2	6	Class 1

An Extra Benefit of Bagging: Out of Bag Estimation

- Able to evaluate the model using the training data
- Apply trees to samples that haven’t been used for training



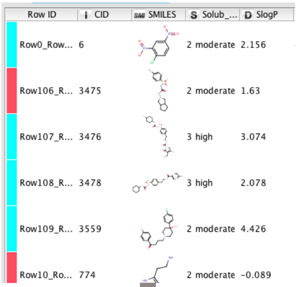


Out-of-bag error estimates - 0:105 - Random Forest Learner
File
Hilite
Navigation
View
Table "default" – Rows: 2666
Spec – Columns: 26
Properties
Flow Variables

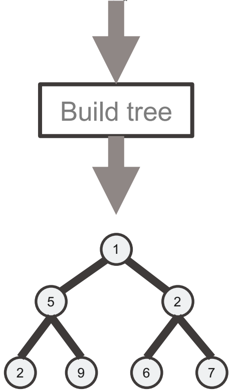
Row ID	S State	D P (Churn=0)	D P (Churn=1)	S Churn (Out-of-bag)	D Churn (Out-of-bag)...	I model count
🟦 Row1_Row0	:S	0.943	0.057	0	0.943	35
🟦 Row2_Row1	)H	1	0	0	1	33
🟦 Row3_Row2	IJ	1	0	0	1	37
🟦 Row4_Row3	)H	0.528	0.472	0	0.528	36
🟦 Row5_Row4	)K	0.976	0.024	0	0.976	41
🟦 Row6_Row5	\L	0.848	0.152	0	0.848	33
🟦 Row7_Row6	ΙA	0.833	0.167	0	0.833	36
🟦 Row9_Row8	A	0.667	0.333	0	0.667	30
🟥 Row11_Ro...	N	0.138	0.862	1	0.862	29
🟦 Row13_Ro...	\	0.974	0.026	0	0.974	39
🟦 Row14_Ro...	IT	0.917	0.083	0	0.917	36
🟦 Row15_Ro...	\	0.387	0.613	1	0.613	31
🟦 Row18_Ro...	'T	0.974	0.026	0	0.974	39
🟦 Row19_Ro...	'A	1	0	0	1	38
🟦 Row21_Ro...	L	0.971	0.029	0	0.971	34
🟥 Row22_Ro...	:O	0.03	0.97	1	0.97	33
🟦 Row23_Ro...	\Z	0.854	0.146	0	0.854	41
🟦 Row25_Ro...	'A	0.973	0.027	0	0.973	37
🟦 Row26_Ro...	IE	0.886	0.114	0	0.886	35
🟦 Row27_Ro...	VY	0.912	0.088	0	0.912	34
🟦 Row28_Ro...	IT	0.976	0.024	0	0.976	42
🟦 Row29_Ro...	IO	1	0	0	1	42
🟦 Row30_Ro...	II	1	0	0	1	40
🟦 Row32_Ro...	IH	0.914	0.086	0	0.914	35
🟦 Row33_Ro...	A	0.875	0.125	0	0.875	32

Random Forest

- Bag of decision trees, with an extra element of randomization
- **Each node** in the decision tree only “sees” **a subset of the input features**, typically $\sqrt{N}$ to pick from
- Random forests tend to be very robust w.r.t. overfitting



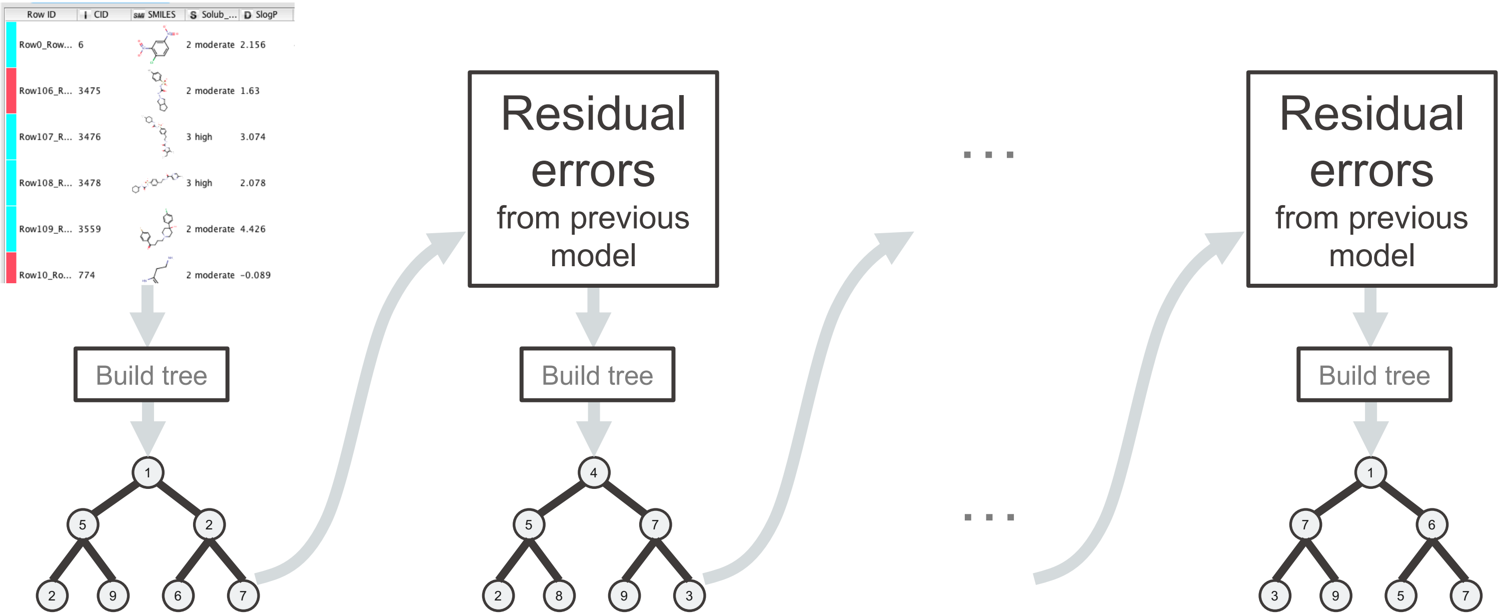
Row ID	CID	SMILES	Solub_...	SlogP
Row0_Row...	6		2 moderate	2.156
Row106_R...	3475		2 moderate	1.63
Row107_R...	3476		3 high	3.074
Row108_R...	3478		3 high	2.078
Row109_R...	3559		2 moderate	4.426
Row10_Ro...	774		2 moderate	-0.089



Build tree

Boosting - Idea

- Starts with a single tree built from the data
- Fits a tree to residual errors from the previous model to refine the model sequentially



Row ID	CID	SMILES	Solub_...	SlogP
Row0_Row...	6		2 moderate	2.156
Row106_R...	3475		2 moderate	1.63
Row107_R...	3476		3 high	3.074
Row108_R...	3478		3 high	2.078
Row109_R...	3559		2 moderate	4.426
Row10_Ro...	774		2 moderate	-0.089

Build tree
1
5
2
2
9
6
7
Residual errors from previous model
Build tree
4
5
7
2
8
9
3
...
...
Residual errors from previous model
Build tree
1
7
6
3
9
5
7

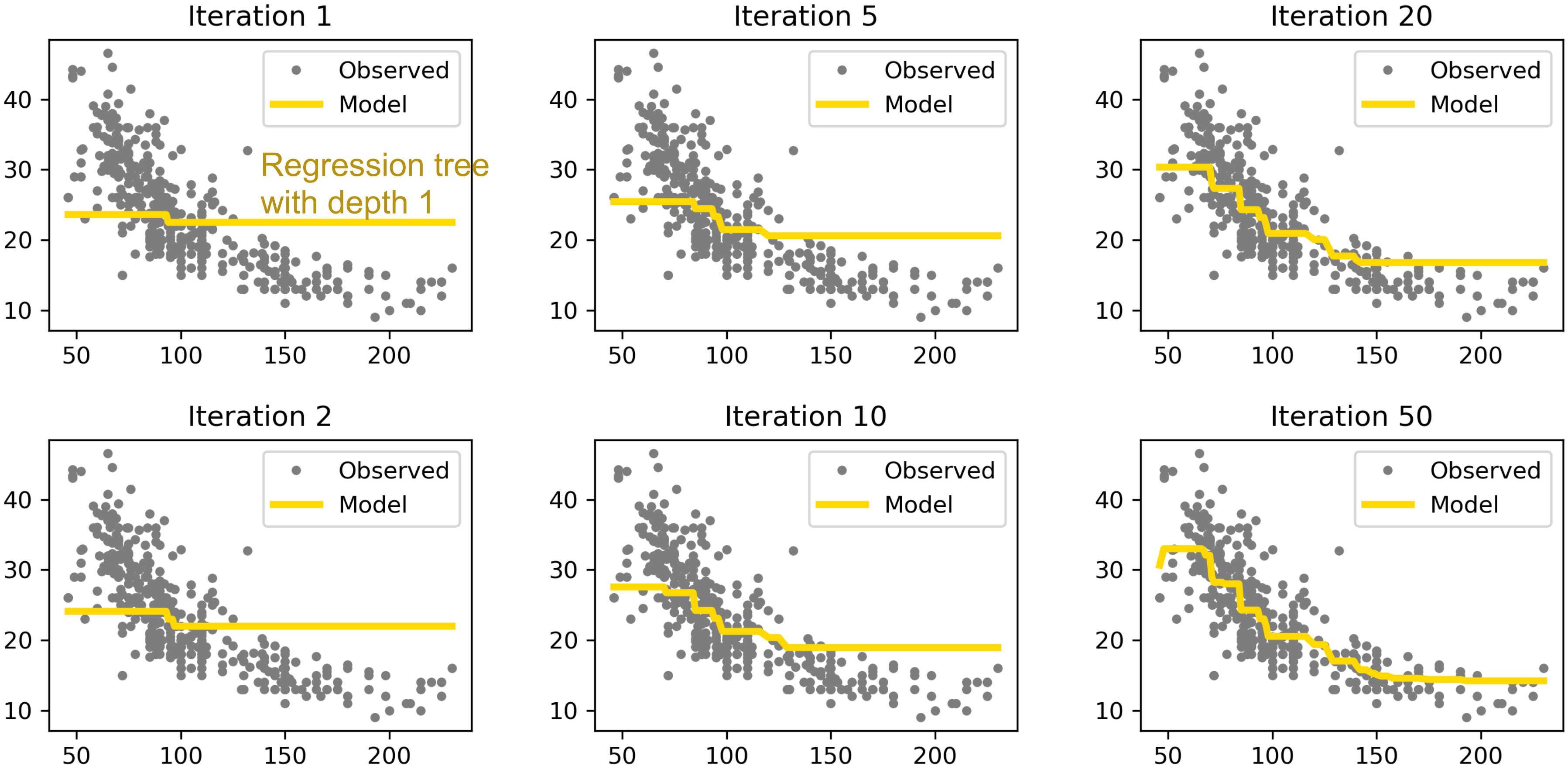
Boosting - Idea

- **Gradient boosting** method
- A shallow tree (depth 4 or less) is built at each step
  - To fit residual errors from the previous step
  - Resulting in a tree $h_m(x)$
- The resulting tree is added to the latest model to update

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

  - Where $F_{m-1}(x)$ is the model from the previous step
  - The weight γ_m is chosen to minimize the loss function
    - Loss function: quantifies the difference between model predictions and data

Gradient Boosting Example – Regression



Iteration 1

Iteration 5

Iteration 20

Iteration 2

Iteration 10

Iteration 50

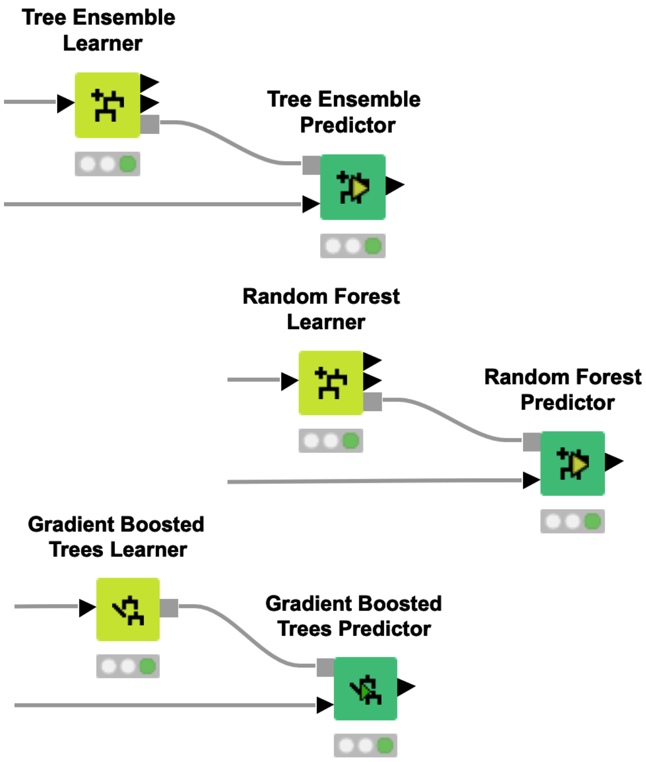
Regression tree with depth 1

Gradient Boosted Trees

- Can be used for classification and regression
- Large number of iterations – prone to overfitting
  - ~100 iterations are sufficient
- Can introduce randomness in choice of data subsets (“stochastic gradient boosting”) and choice of input features

Ensemble Tree Nodes in KNIME Analytics Platform

Classification Problems



Tree Ensemble Learner

Tree Ensemble Predictor

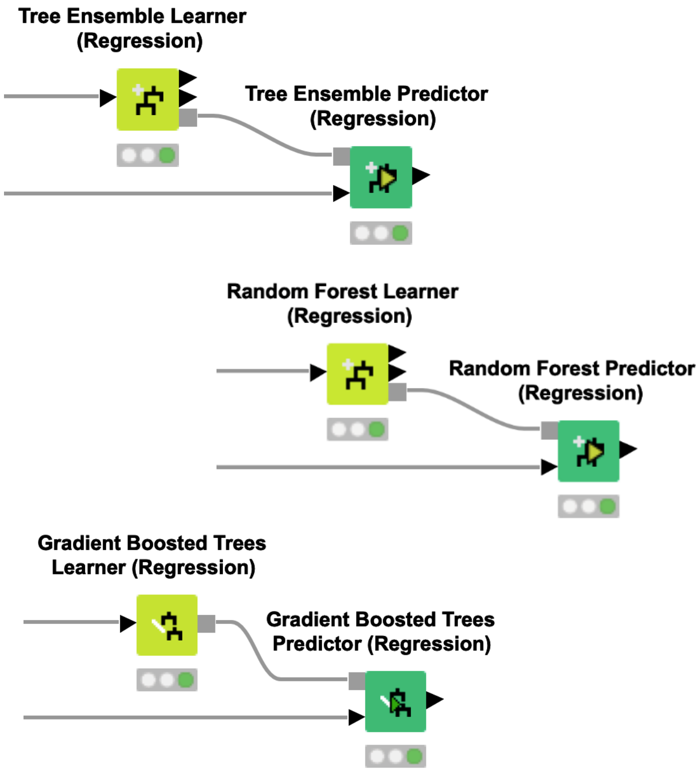
Random Forest Learner

Random Forest Predictor

Gradient Boosted Trees Learner

Gradient Boosted Trees Predictor

Regression Problems



Tree Ensemble Learner (Regression)

Tree Ensemble Predictor (Regression)

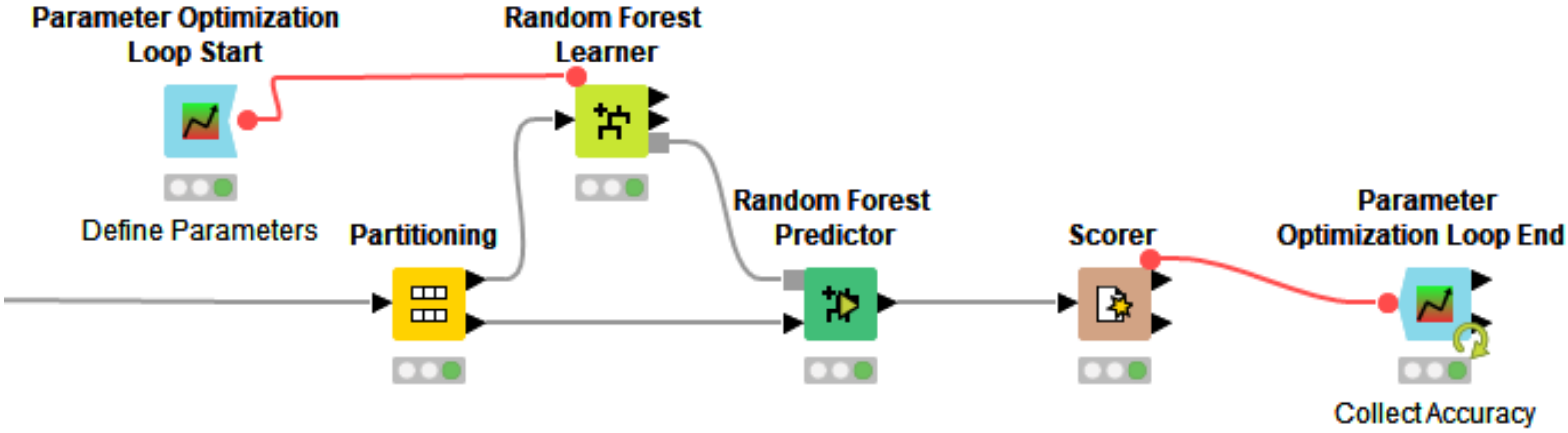
Random Forest Learner (Regression)

Random Forest Predictor (Regression)

Gradient Boosted Trees Learner (Regression)

Gradient Boosted Trees Predictor (Regression)

Parameter Optimization



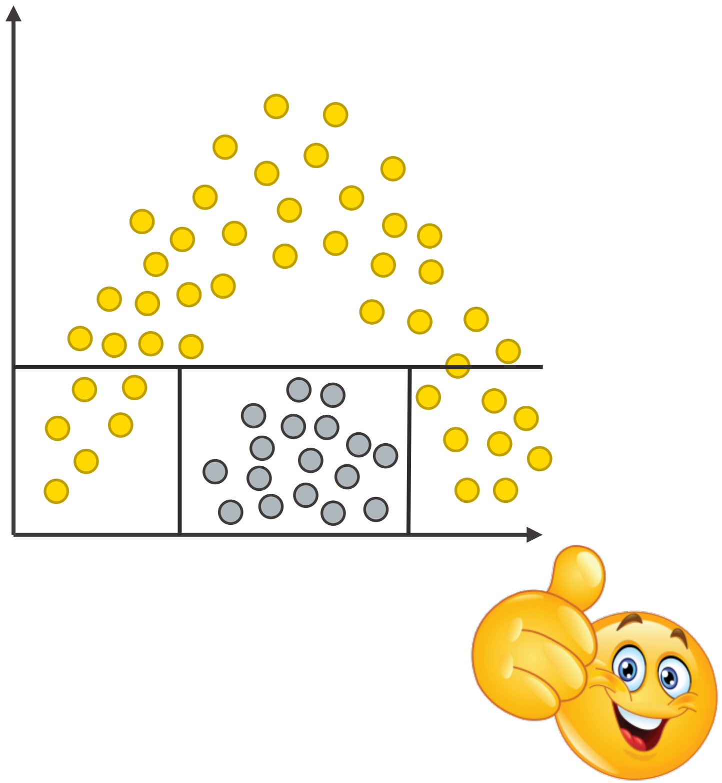
Logistic Regression

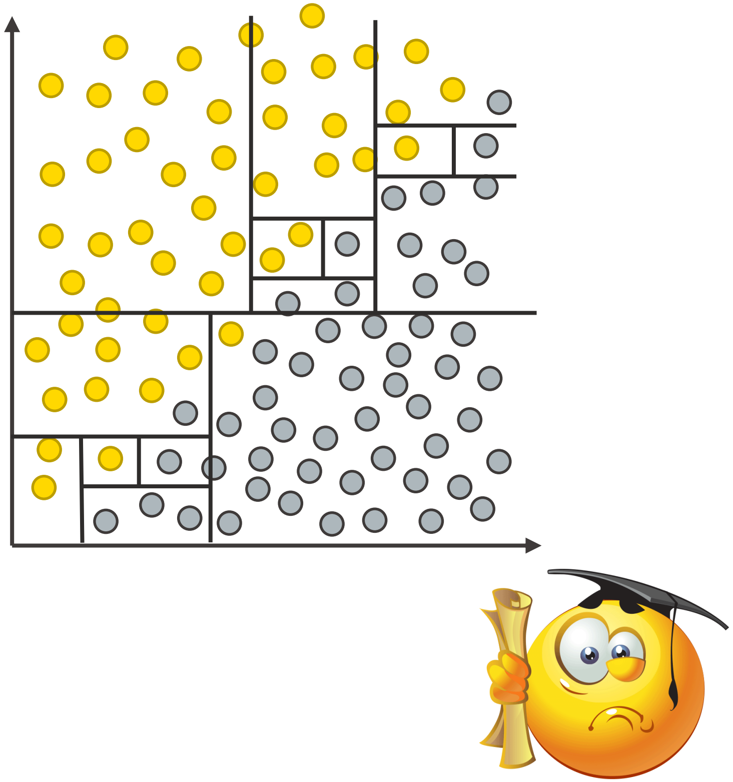
What is a Logistic Regression (algorithm)?

- Another algorithm to train a classification model

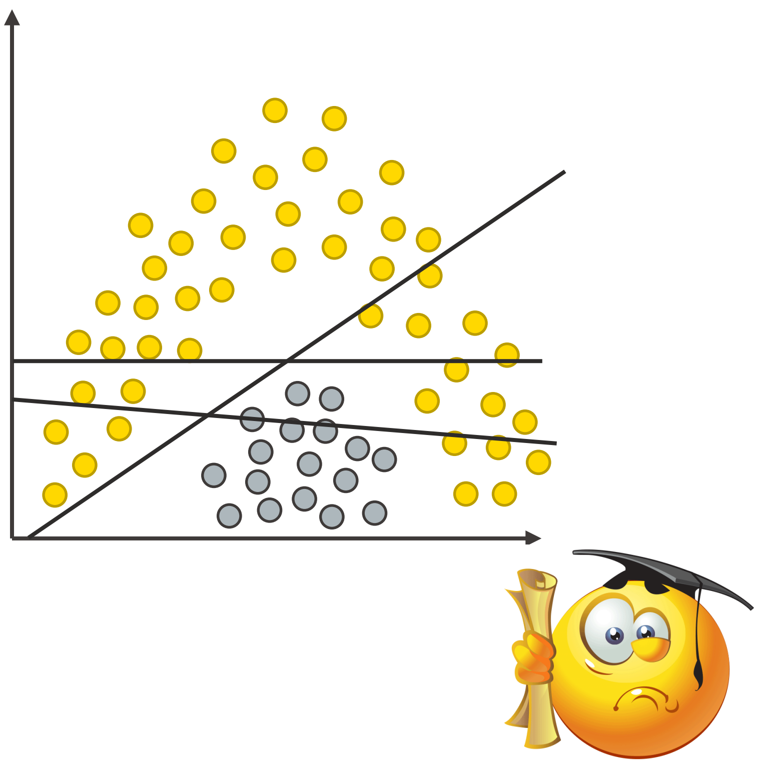
I know already the decision tree algorithm and tree ensembles. Why do I need another one?

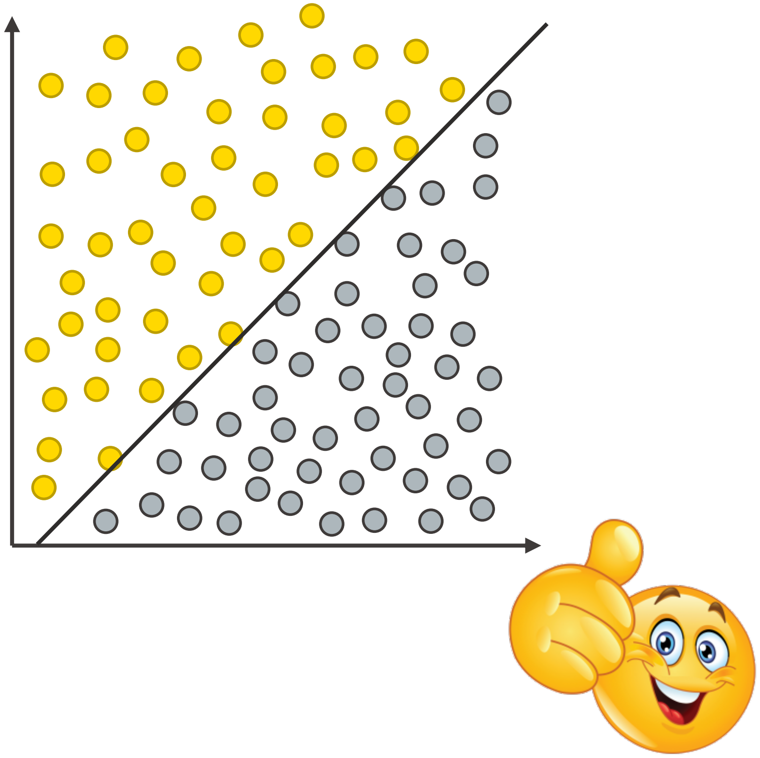
Why Shouldn’t we Always use the Decision Tree?





Decision Boundary of a Logistic Regression





Linear Regression vs. Logistic Regression

	Linear Regression	Logistic Regression
Target variable y	Numeric $y \in (-\infty, \infty)/[a, b]$	***Nominal*** $y \in \{0, 1, 2, 3\}/\{red, white\}$
Functional relationship between features and…	… target value y $y = f(x_1, \dots, x_n, \beta_0, \dots, \beta_n)$ $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$	… ***class probability P (y = class i)*** $P(y = c_i) = f(x_1, \dots, x_n, \beta_0, \dots, \beta_n)$

Goal: Find the regression coefficients $\beta_0, \dots, \beta_n$

Let’s find out how Binary Logistic Regression works!

- Idea: Train a function, which gives us the probability for each class (0 and 1) based on the input features

- Recap on probabilities
  - Probabilities are always between 0 and 1
  - The probability of all classes sum up to 1

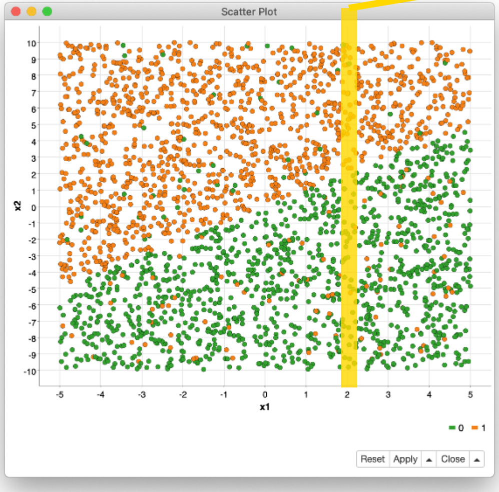
$$P(y = 1) = p_1 \implies P(y = 0) = 1 - p_1$$

➔ It’s sufficient to model the probability for one class

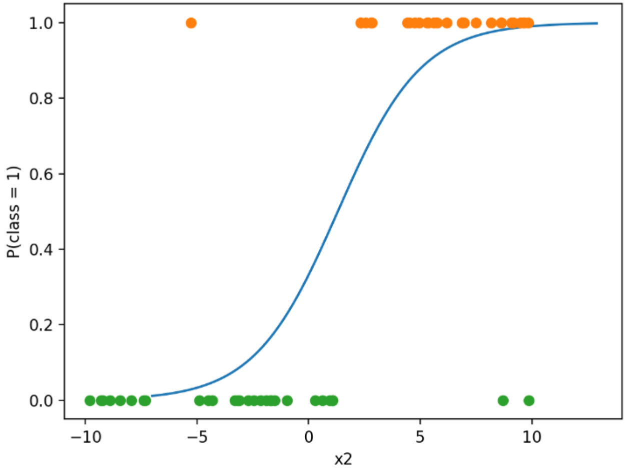
Let’s Find Out How Binary Logistic Regression Works!

$$P(y = 1) = f(x_1, x_2; \beta_0, \beta_1, \beta_2) := \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

Feature space



Probability function given $x_1 = 2$



More General: Binary Logistic Regression

- Model:

$$\pi = P(y = 1) = \frac{1}{1+\exp(-z)}$$

With $z = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n = \boldsymbol{X\beta}$.

- Goal: Find the regression coefficients $\boldsymbol{\beta} = (\beta_0, \dots, \beta_n)$
- Notation:
  - y_i is the class value for sample i
  - $x_1, \dots, x_n$ is the set of input features, $\boldsymbol{X} = (1, x_1, \dots, x_n)$
  - The training data set has m observations $(y_i, x_1^i, \dots, x_n^i)$

How can we Find the Best Coefficients β?

Maximize the product of the probabilities ➔ Likelihood function

$$L(\beta; y, X) = \prod_{i=1}^{m} P(y = y_i) = \prod_{i=1}^{m} \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

Why does it make sense to maximize this function?

$$P(y = y_i) = \begin{cases} \pi_i & if\ y_i = 1 \\ 1-\pi_i & if\ y_i = 0 \end{cases}$$

$$= \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

Remember:
$\pi_i = \mathrm{P}(y = 1)$
$u^0 = 1$ for $u \in \mathbb{R}$
$u^1 = u$ for $u \in \mathbb{R}$

Max Likelihood and Log Likelihood Functions

- Maximize the Likelihood function $L(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X})$

$$\max_{\beta} L(\beta; y, X) = \max_{\beta} \prod_{i=1}^{m} \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

- Equivalent to maximizing the Log Likelihood function $LL(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X})$

$$\max_{\beta} LL(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X}) = \max_{\beta} \sum_{i=1}^{n} y_i \ln(\pi_i) + (1 - y_i) \ln(1 - \pi_i)$$

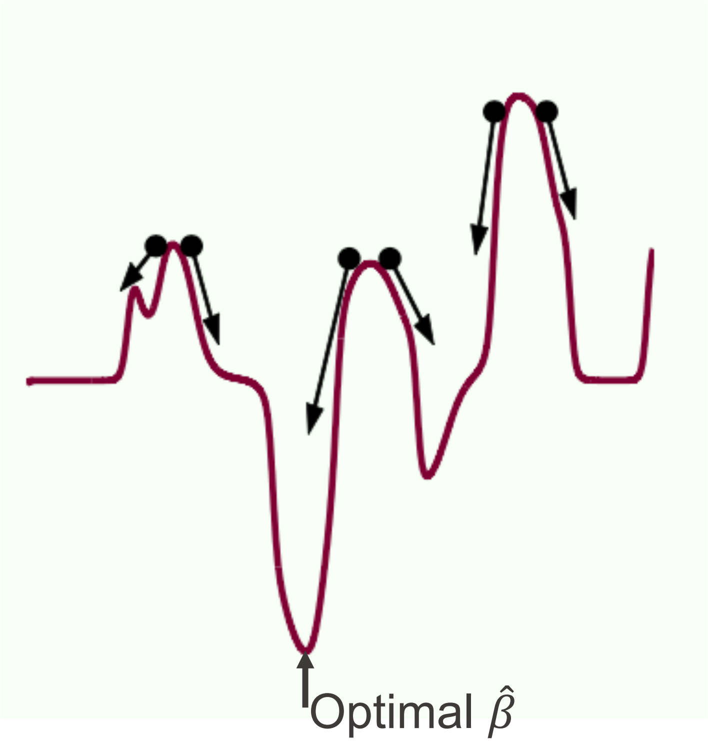
How can we find this Coefficients?

- To find the coefficients of our model we want to find $\boldsymbol{\beta}$ so that the value of the function $LL(\boldsymbol{\beta}; \boldsymbol{y}, \boldsymbol{X})$ is maximal

- KNIME Analytics Platform provides two algorithms
  - Iteratively re-weighted least squares
    - Uses the idea of the newton method
  - Stochastic average gradient descent

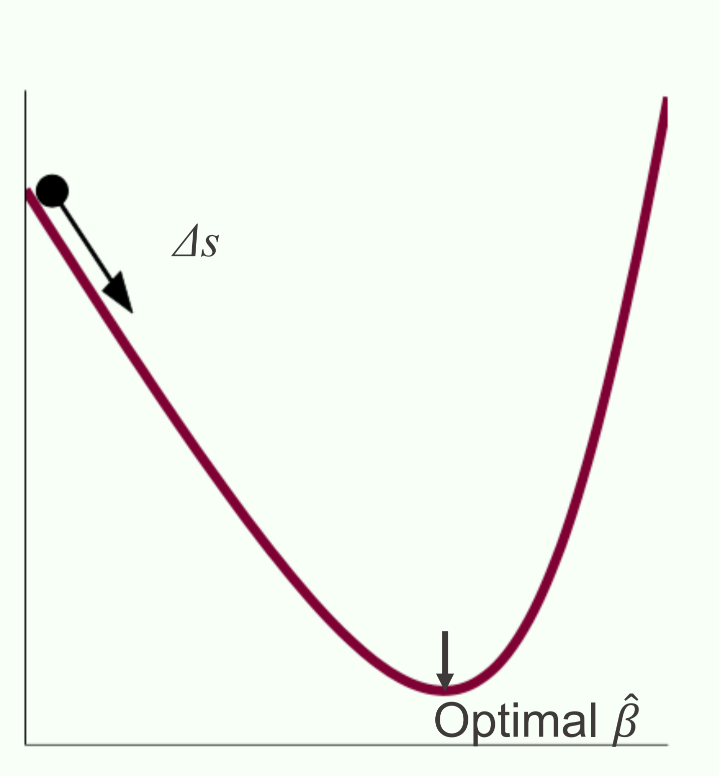
Idea: Gradient Descent Method

$$\max LL(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{y}) \Leftrightarrow \min -LL(\boldsymbol{\beta};\boldsymbol{X},\boldsymbol{y})$$



Idea: Gradient Descent Method

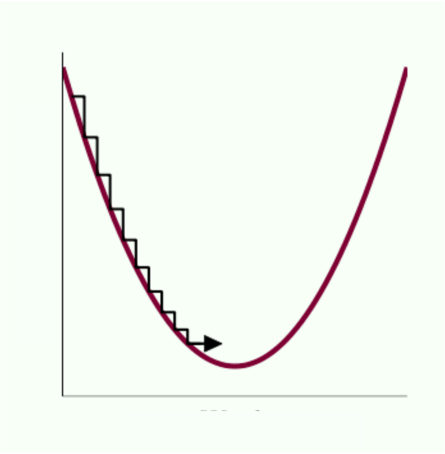
$$\max LL(\boldsymbol{\beta}; \boldsymbol{X}, \boldsymbol{y}) \Leftrightarrow \min -LL(\boldsymbol{\beta}; \boldsymbol{X}, \boldsymbol{y})$$



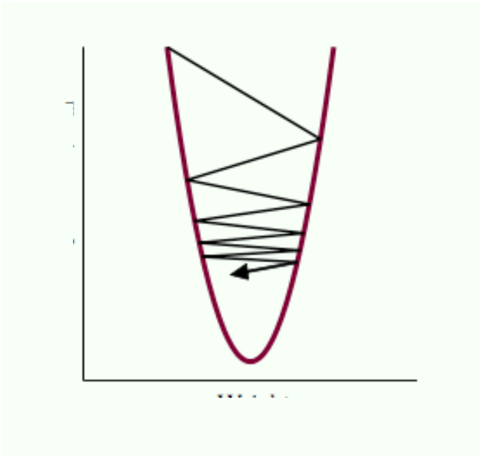
- Example: $\min\ -LL(\beta) \coloneqq f(\beta)$
- Start from an arbitrary point
- Move towards the minimum
- With step size Δs
- If $f(\beta)$ is strictly convex
  ➔ Only one global minimum exists
- Z normalization of the input data lead to better convergence

Learning Rate / Step Length Δs

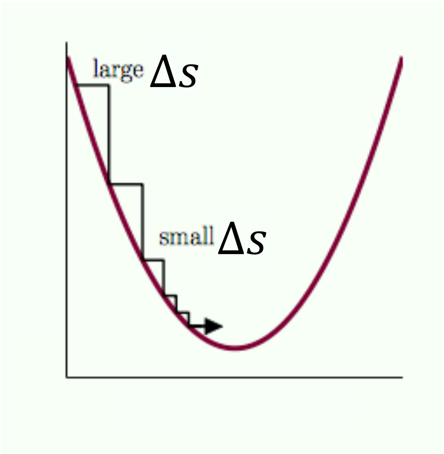
Δs too small



Δs too large



Just right



Learning Rate Δs

- Fixed:

$$\Delta s_k = \Delta s_0$$

- Annealing:

$$\Delta s_k = \frac{\Delta s_0}{1 + \frac{\alpha}{k}}$$

with iteration number k and decay rate α

- Line Search: Learning rate strategy that tries to find the optimal learning rate

Is there a way to handle Overfitting as well? (optional)

- To avoid overfitting: add regularization by penalizing large weights
  - L_2 regularizations = Coefficients are Gauss distributed with $\sigma = \frac{1}{\lambda}$

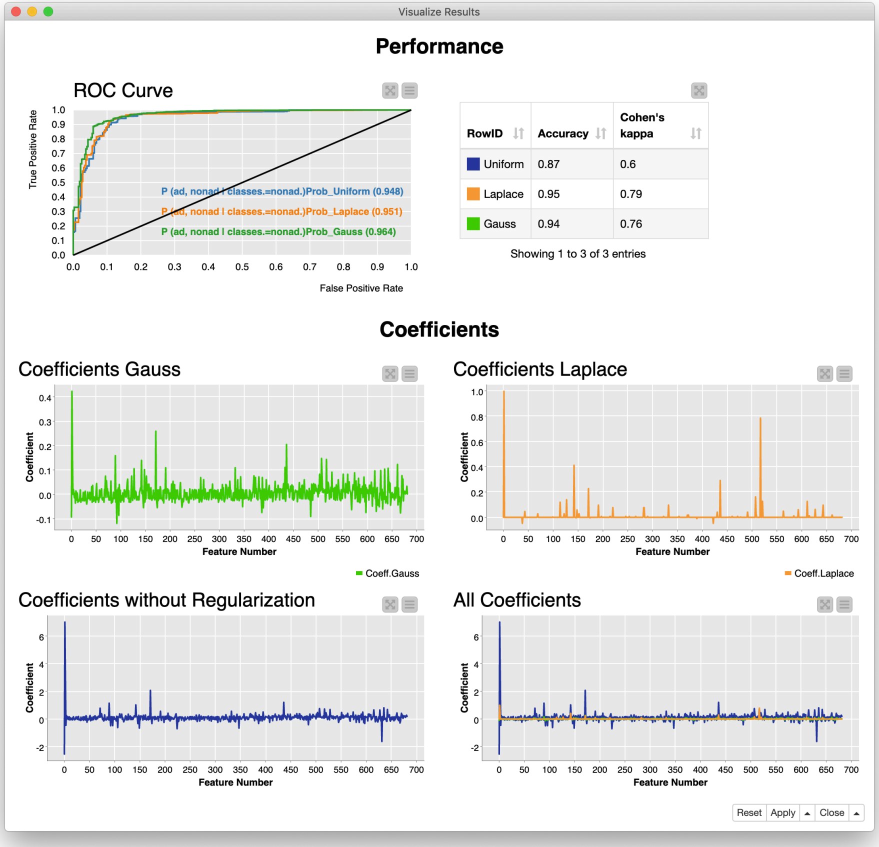
$$l(\hat{\beta}; y, X) := -LL(\hat{\beta}; y, X) + \frac{\lambda}{2} ||\hat{\beta}||_2^2$$

  - L_1 regularizations = Coefficients are Laplace distributed with $\sigma = \frac{\sqrt{2}}{\lambda}$

$$l(\hat{\beta}; y, X) := -LL(\hat{\beta}; y, X) + \lambda ||\hat{\beta}||_1$$

=> The smaller σ, the smaller the coefficients $\hat{\beta}$

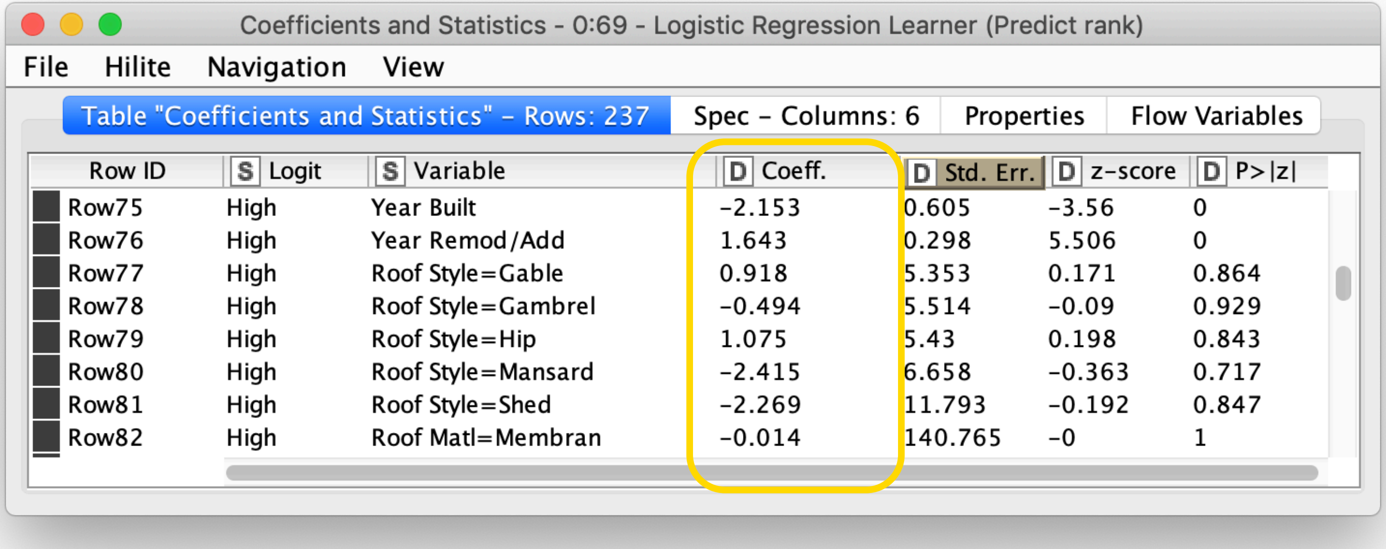
Impact of Regularization



RowID	Accuracy	Cohen's kappa
🟦 Uniform	0.87	0.6
🟧 Laplace	0.95	0.79
🟩 Gauss	0.94	0.76

Showing 1 to 3 of 3 entries

Interpretation of the Coefficients

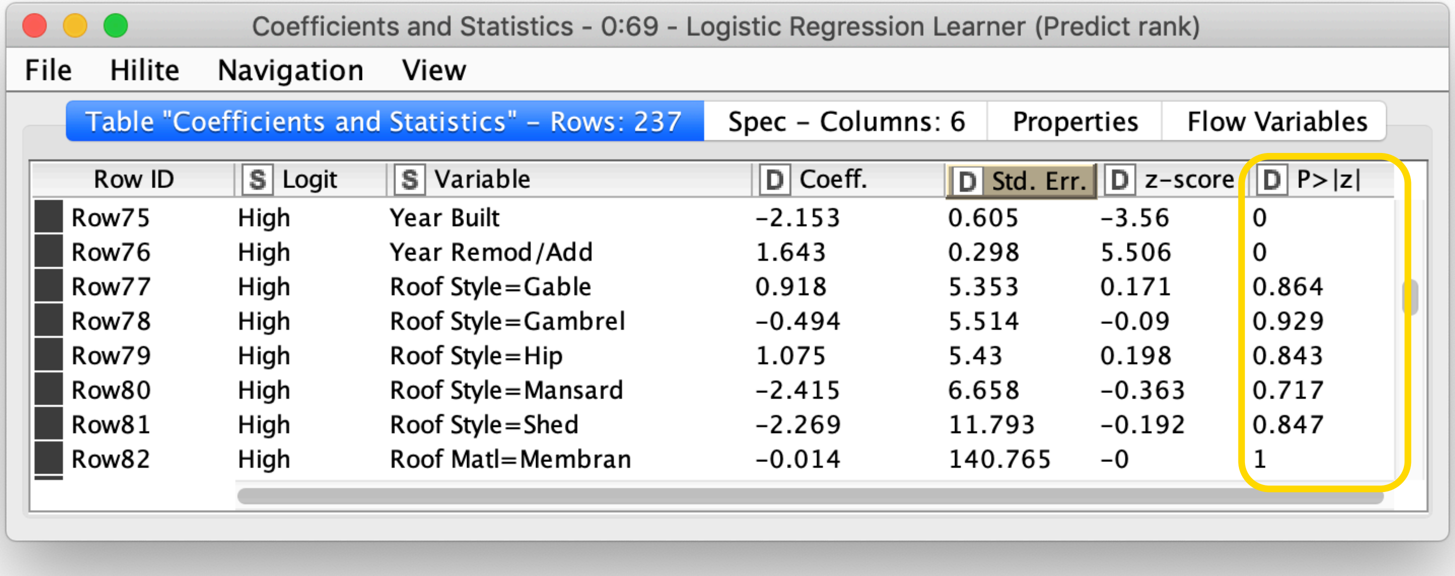


Coefficients and Statistics - 0:69 - Logistic Regression Learner (Predict rank)
File
Hilite
Navigation
View
Table "Coefficients and Statistics" – Rows: 237
Spec – Columns: 6
Properties
Flow Variables

Row ID	S Logit	S Variable	D Coeff.	D Std. Err.	D z-score	D P>|z|
Row75	High	Year Built	-2.153	0.605	-3.56	0
Row76	High	Year Remod/Add	1.643	0.298	5.506	0
Row77	High	Roof Style=Gable	0.918	5.353	0.171	0.864
Row78	High	Roof Style=Gambrel	-0.494	5.514	-0.09	0.929
Row79	High	Roof Style=Hip	1.075	5.43	0.198	0.843
Row80	High	Roof Style=Mansard	-2.415	6.658	-0.363	0.717
Row81	High	Roof Style=Shed	-2.269	11.793	-0.192	0.847
Row82	High	Roof Matl=Membran	-0.014	140.765	-0	1

- Interpretation of the sign
  - $\beta_i > 0$: Bigger x_i lead to higher probability
  - $\beta_i < 0$: Bigger x_i lead to smaller probability

Interpretation of the p Value



Coefficients and Statistics - 0:69 - Logistic Regression Learner (Predict rank)

Row ID	S Logit	S Variable	D Coeff.	D Std. Err.	D z-score	D P>|z|
Row75	High	Year Built	-2.153	0.605	-3.56	0
Row76	High	Year Remod/Add	1.643	0.298	5.506	0
Row77	High	Roof Style=Gable	0.918	5.353	0.171	0.864
Row78	High	Roof Style=Gambrel	-0.494	5.514	-0.09	0.929
Row79	High	Roof Style=Hip	1.075	5.43	0.198	0.843
Row80	High	Roof Style=Mansard	-2.415	6.658	-0.363	0.717
Row81	High	Roof Style=Shed	-2.269	11.793	-0.192	0.847
Row82	High	Roof Matl=Membran	-0.014	140.765	-0	1

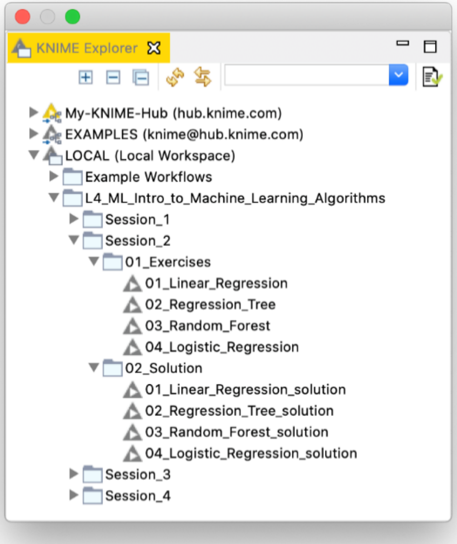
- p- value < α: input feature has a significant impact on the dependent variable.

Summary Logistic Regression

- Logistic regression is used for classification problems
- The regression coefficients are calculated by maximizing the likelihood function, which has no closed form solution, hence iterative methods are used.
- Regularization can be used to avoid overfitting.
- The p-value shows us whether an independent variable is significant

Exercises

- Regression Exercises:
  - Goal: Predicting the house price
  - 01_Linear_Regression
  - 02_Regression_Tree
- Classification Exercises:
  - Goal: Predicting the house condition (high /low)
  - 03_Radom_Forest (with optional exercise to build a parameter optimization loop)
  - 04_Logistic_Regression



KNIME Explorer

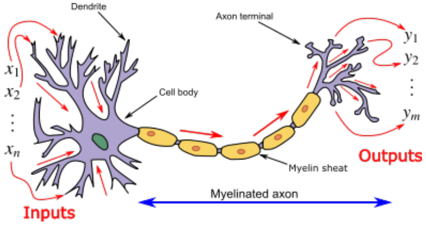
- My-KNIME-Hub (hub.knime.com)
- EXAMPLES (knime@hub.knime.com)
- LOCAL (Local Workspace)
  - Example Workflows
  - L4_ML_Intro_to_Machine_Learning_Algorithms
    - Session_1
    - Session_2
      - 01_Exercises
        - 01_Linear_Regression
        - 02_Regression_Tree
        - 03_Random_Forest
        - 04_Logistic_Regression
      - 02_Solution
        - 01_Linear_Regression_solution
        - 02_Regression_Tree_solution
        - 03_Random_Forest_solution
        - 04_Logistic_Regression_solution
    - Session_3
    - Session_4

Session 3: Neural Networks and Recommendation Engines

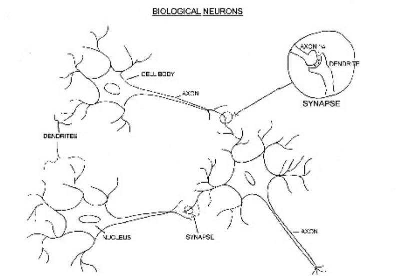
Artificial Neurons and Networks

Biological vs. Artificial

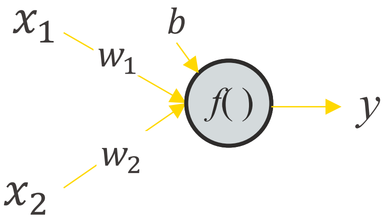
Biological Neuron



Biological Neural Networks



Artificial Neuron (Perceptron)



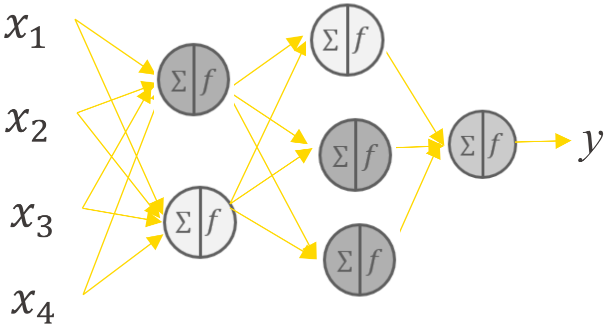
$$y = f(x_1 w_1 + x_2 w_2 + b)$$

$$b = w_0$$

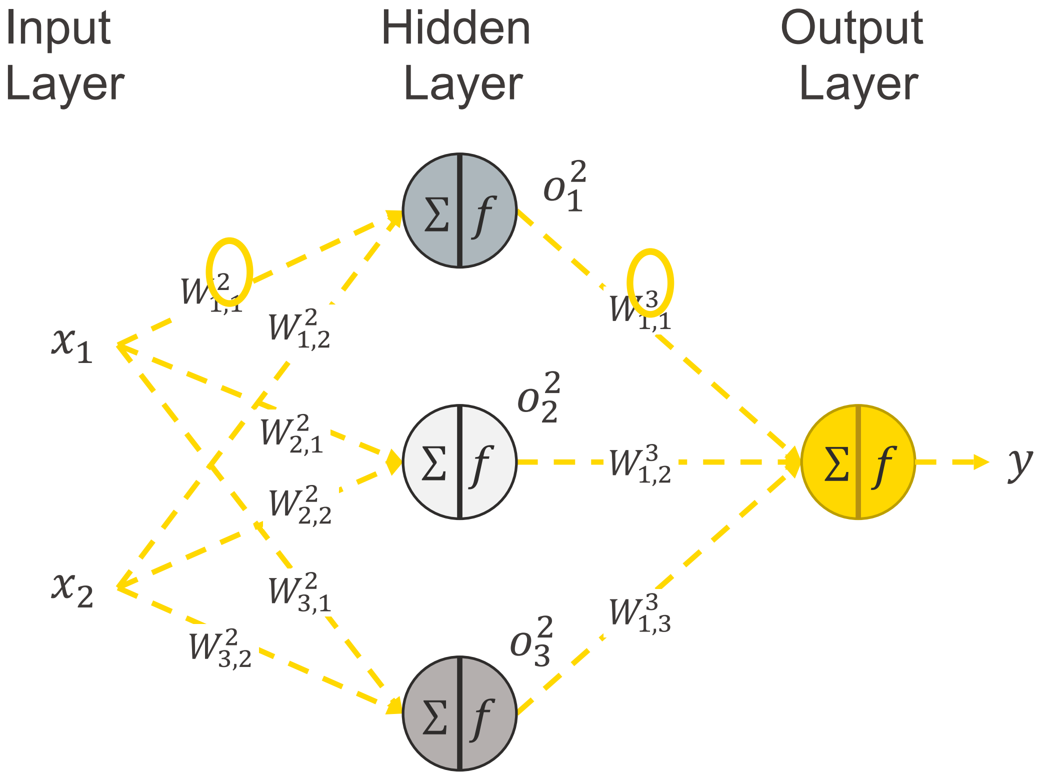
$$y = f(\sum_{i=0}^{n} x_i w_i)$$

Artificial Neural Networks

(Multilayer Perceptron, MLP)



Architecture / Topology



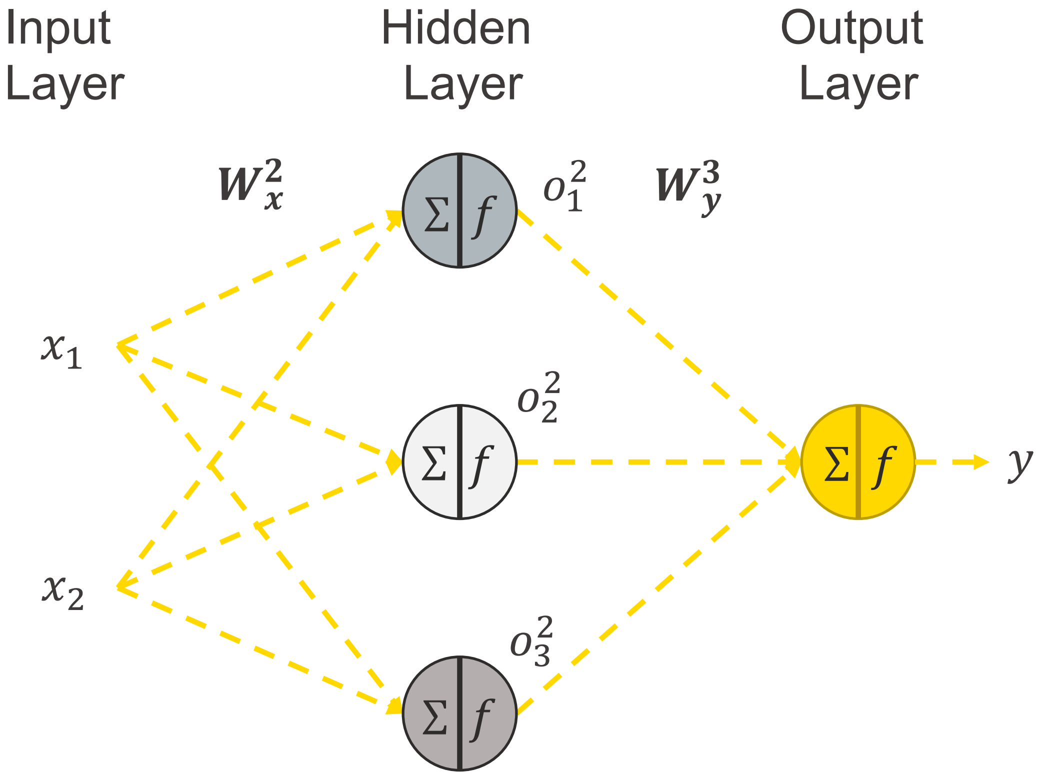
Forward pass:

$$\boldsymbol{o} = f(W_x^2 \boldsymbol{x})$$

$$y = f(W_y^3 \boldsymbol{o})$$

fully connected
feed forward

Same with Matrix Notations



Forward pass:

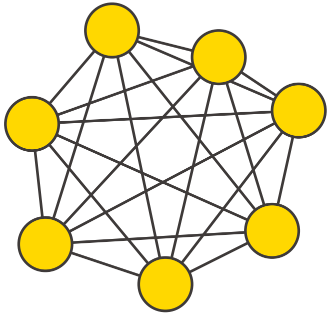
$$\boldsymbol{o} = f(W_x^2 \boldsymbol{x})$$

$$y = f(W_y^3 \boldsymbol{o})$$

f( ) = activation function

Neural Architectures

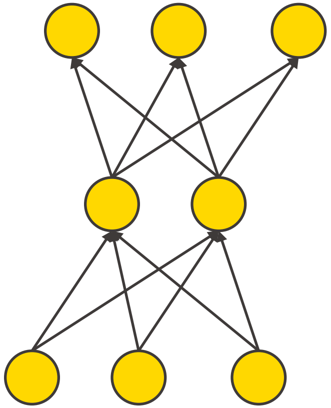
completely
connected



example:

- Associative neural network
- Hopfield

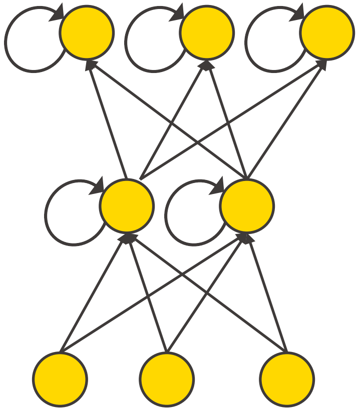
feedforward
(directed, a-cyclic)



example:

- auto associative neural network
- Multi Layer Perceptron

recurrent
(feedback connections)

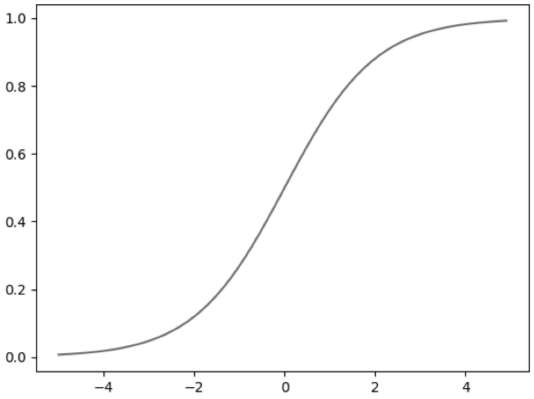


example:

- recurrent neural network (for time series recognition)

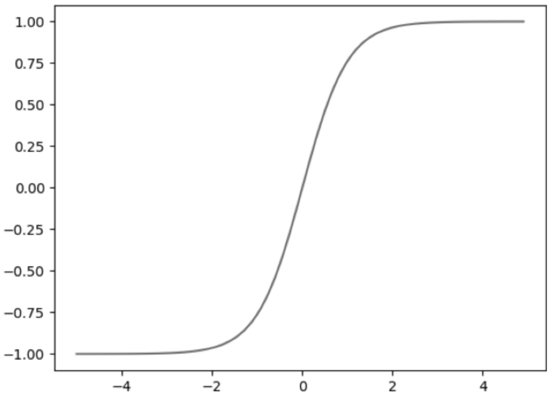
Frequently used activation functions

Sigmoid



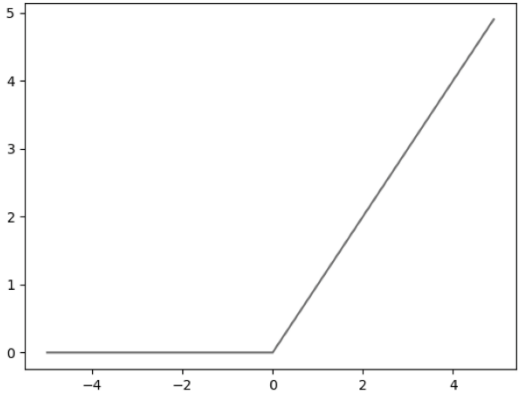
$$f(a) = \frac{1}{1 + e^{-ha}}$$

Tanh



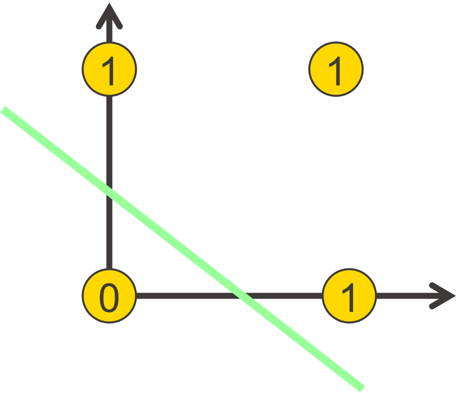
$$f(a) = \frac{e^{2ha} - 1}{e^{2ha} + 1}$$

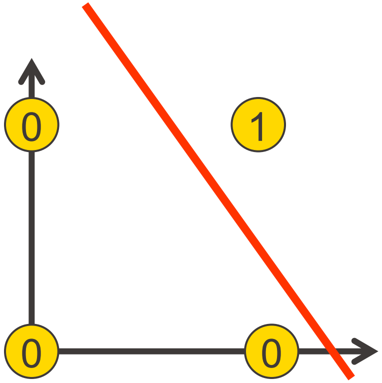
Rectified Linear Unit (ReLU)

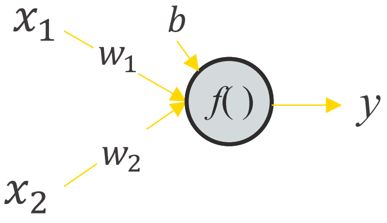


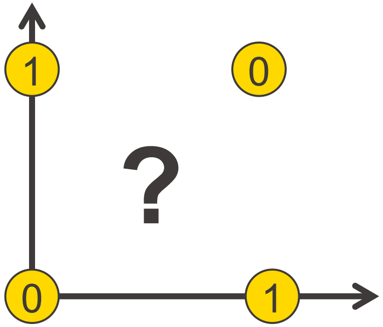
$$f(a) = max\{0, ha\}$$

What can a single Perceptron do?

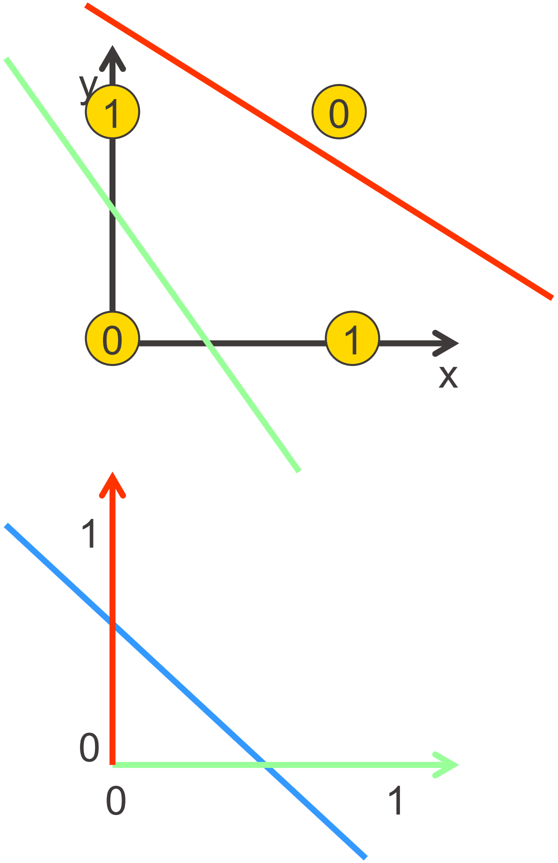


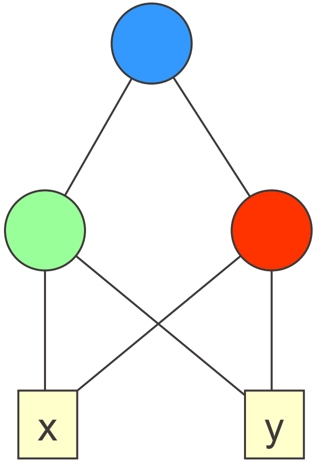




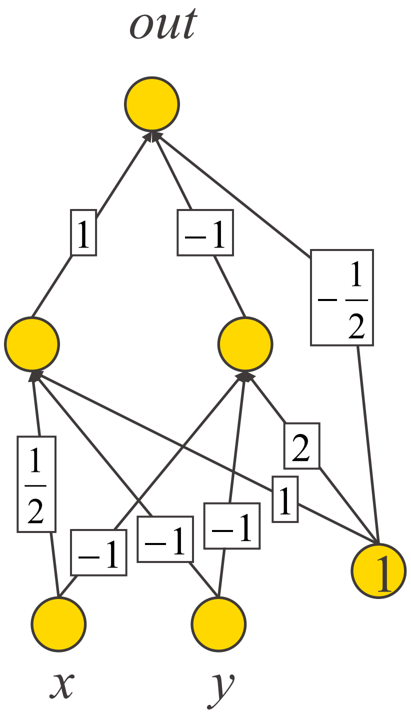


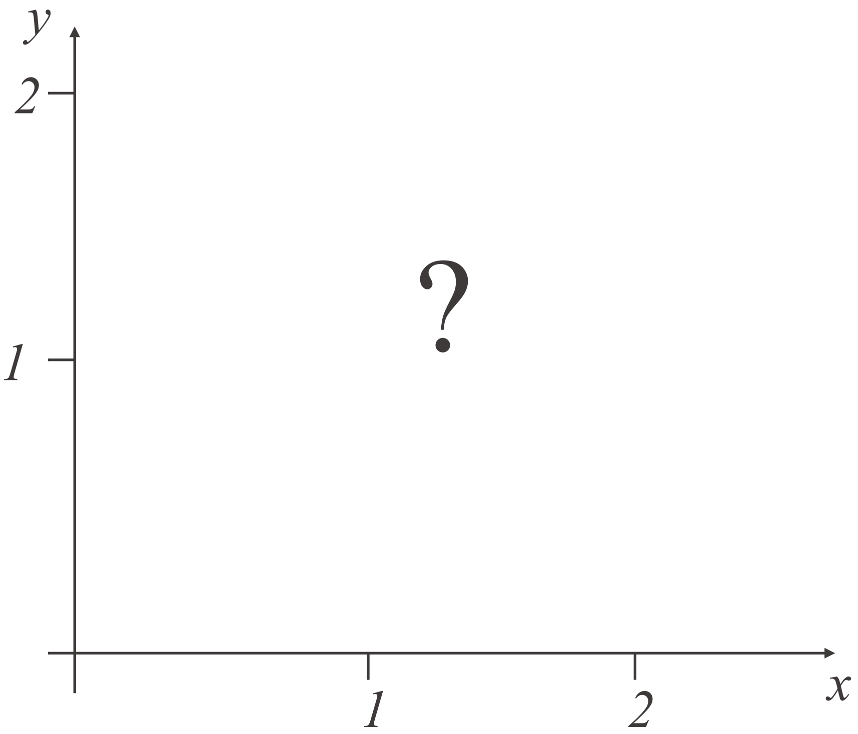
What can a 3-neuron MLP do?



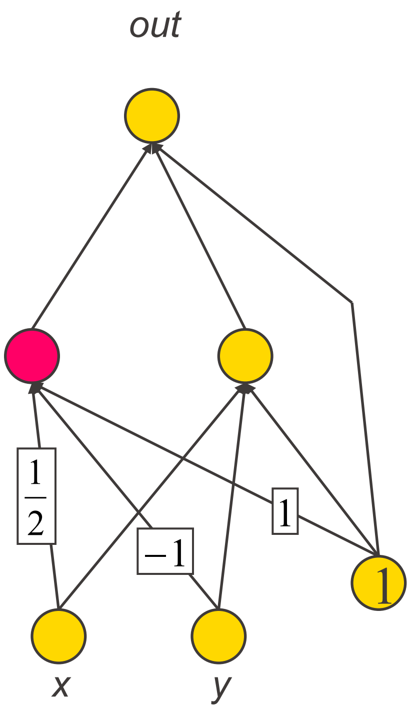


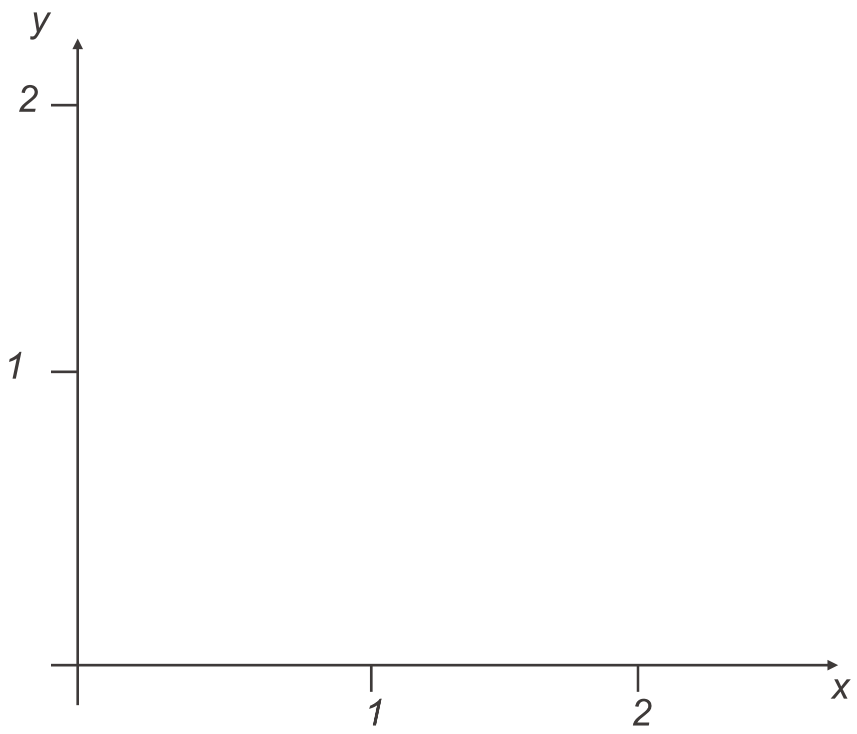
MLP: Example



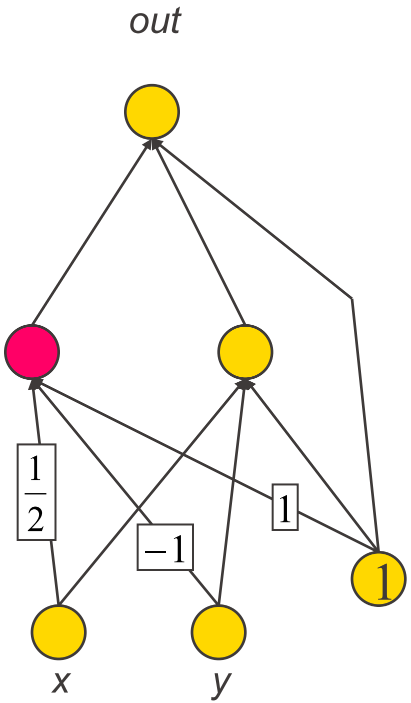


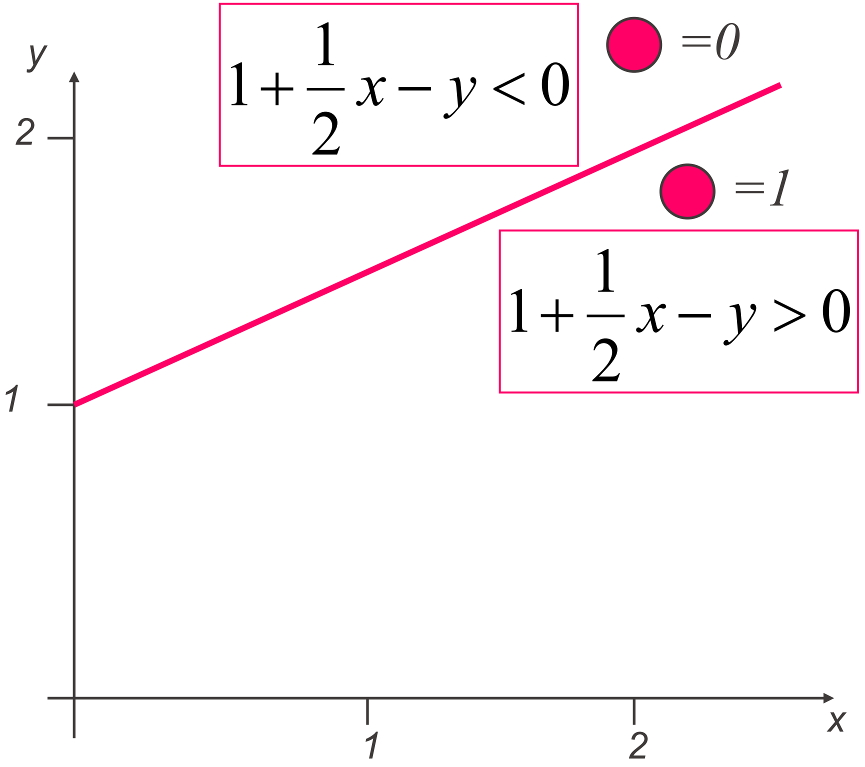
MLP: Example





MLP: Example





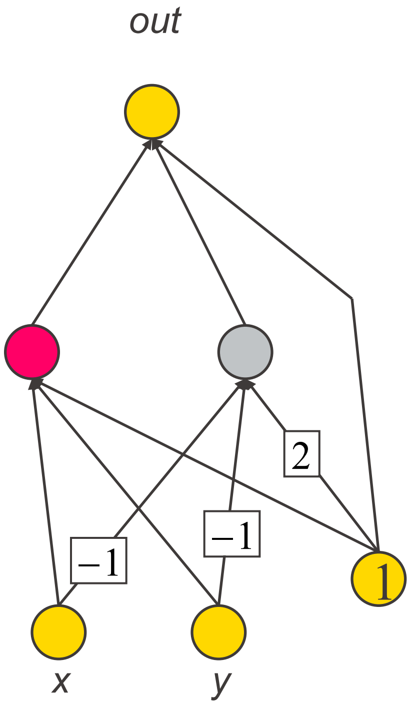
$$1+\frac{1}{2}x-y<0$$

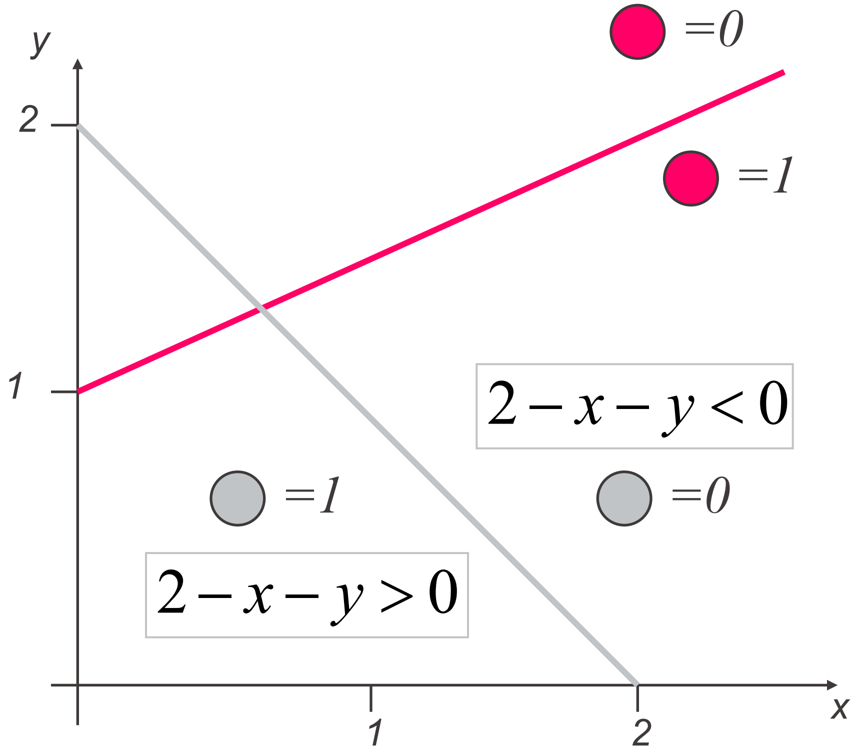
● $=0$

● $=1$

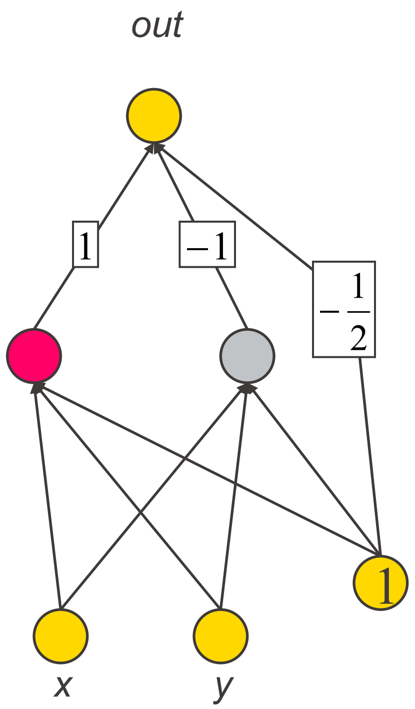
$$1+\frac{1}{2}x-y>0$$

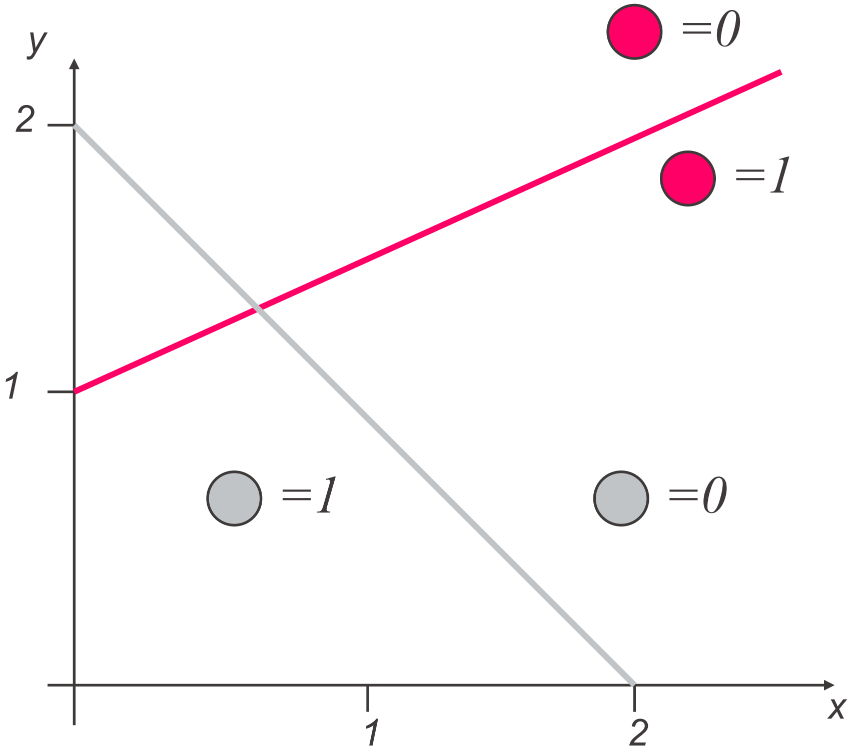
MLP: Example



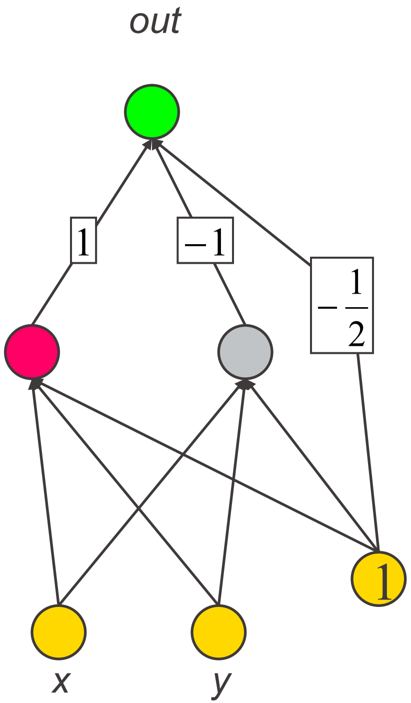


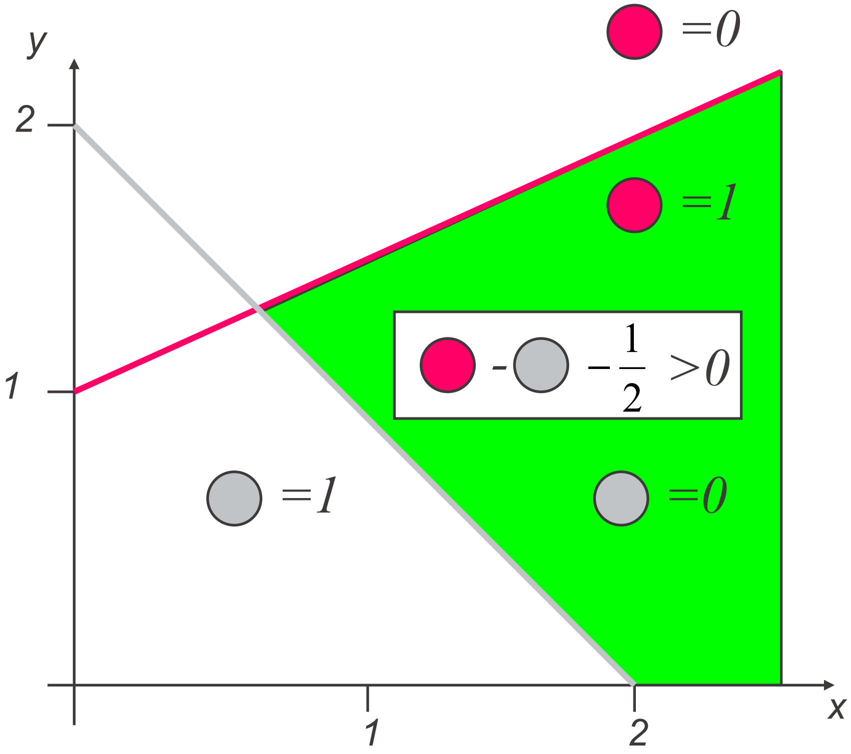
MLP: Example



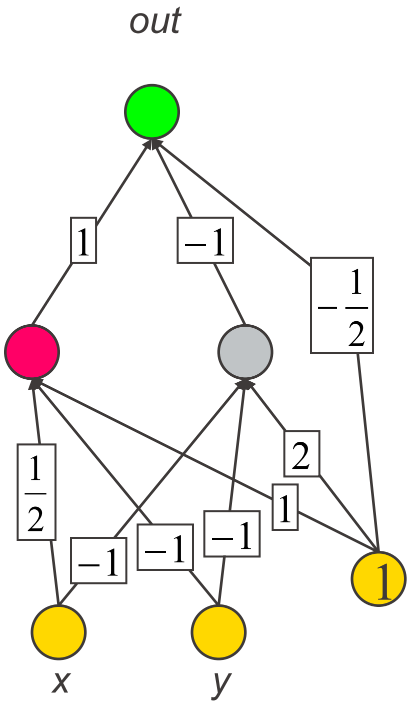


MLP: Example

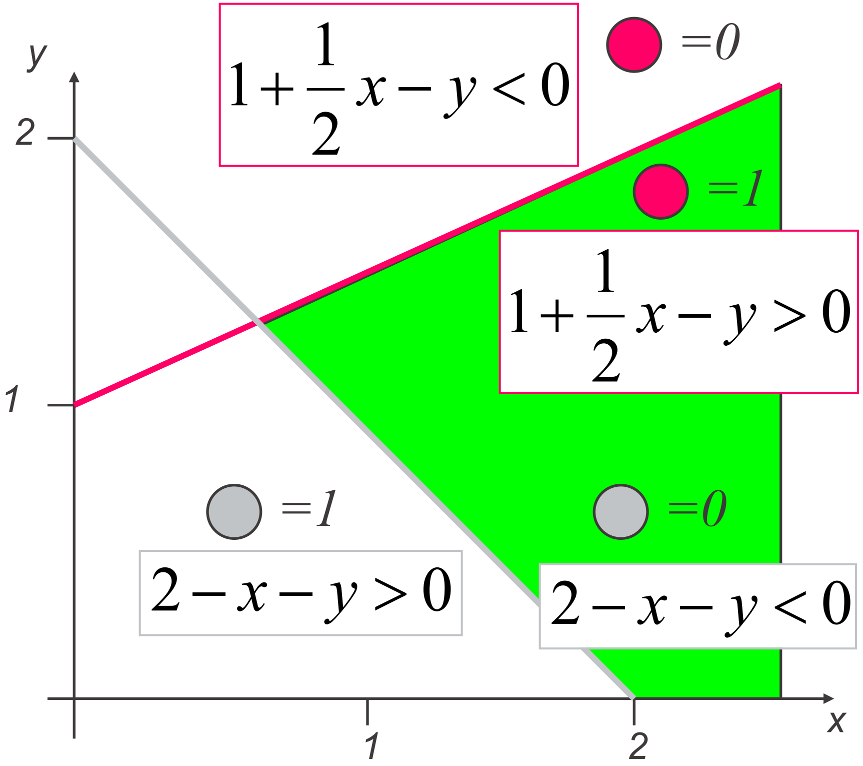




MLP: Example



out
1
-1
$-\frac{1}{2}$
$\frac{1}{2}$
2
1
-1
-1
-1
1
x
y



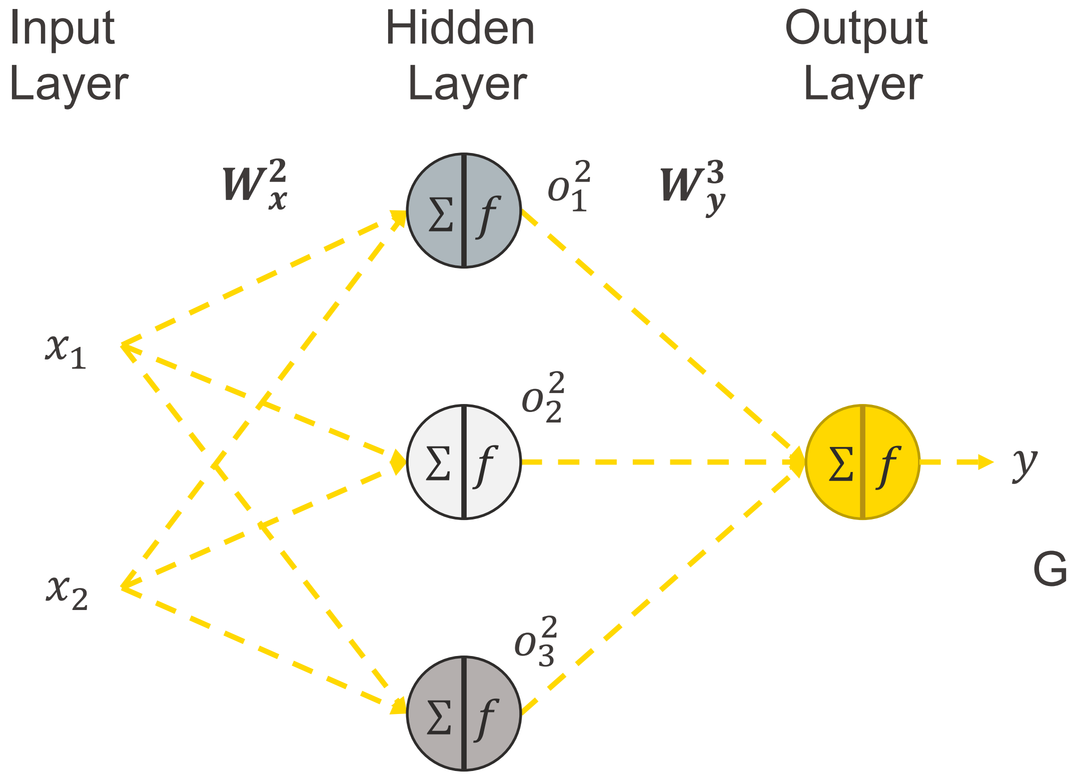
y
2
1
x
1
2
$1+\frac{1}{2}x-y<0$
● $=0$
● $=1$
$1+\frac{1}{2}x-y>0$
● $=1$
$2-x-y>0$
● $=0$
$2-x-y<0$

Back-Propagation

Training of a Feed Forward Neural Network - MLP

- Teach (ensemble of) neuron(s) a desired input-output behavior.
- Show examples from the training set repeatedly
- Networks adjusts parameters to fit underlying function
  - topology
  - weights
  - internal functional parameters

Training of a Feed Forward Neural Network - MLP



Forward pass:

$$\boldsymbol{o} = f(W_x^2 \boldsymbol{x})$$

$$y = f(W_y^3 \boldsymbol{o})$$

$$E = \sum_j \frac{1}{2} (t_j - y_j)^2$$

Target (j) Network output (j)

Gradient descent

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

... Some Calculations for the Output Layer ....

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial\left(\frac{1}{2}(t_j - y_j)^2\right)}{\partial w_{ji}} = \frac{\partial\left(\frac{1}{2}(t_j - y_j)^2\right)}{\partial y_j}\frac{\partial y_j}{\partial w_{ji}} = -\left(t_j - y_j\right)\frac{\partial y_j}{\partial w_{ji}}$$

$$= -\left(t_j - y_j\right)\frac{\partial y_j}{\partial h_j}\frac{\partial h_j}{\partial w_{ji}} = -\left(t_j - y_j\right)g'(h_j)\frac{\partial h_j}{\partial w_{ji}} = -\left(t_j - y_j\right)g'(h_j)\frac{\partial\left(\sum_k x_k w_{jk}\right)}{\partial w_{ji}}$$

$$= -\left(t_j - y_j\right)g'(h_j)x_i$$

$$\Delta w_{ji} = -\eta\left(t_j - y_j\right)g'\left(h_j\right)x_i = -\eta\ \delta_j^{out}\ x_i$$

… some Calculations for the Hidden Layer …

$$\Delta w_{ij}^{hidden} = \frac{\partial \frac{1}{2}\sum_{x\in T}\sum_{k=1}^{c}(f(a_k^{out}(x)) - y_k(x))^2}{\partial w_{ij}^{hidden}} = -\frac{\eta}{2}\sum_{x\in T}\sum_{k=1}^{c}\frac{\partial(f(a_k^{out}(x)) - y_k(x))^2}{\partial w_{ij}^{hidden}}$$

$$\dots = -\frac{\eta}{2}\sum_{x\in T}\sum_{k=1}^{c} 2(f(a_k^{out}(x)) - y_k(x)) \frac{\partial\left(f\left(\sum_{j'=1}^{h} w_{j'k}^{out} f\left(\sum_{i'=1}^{m} w_{i'j'}^{hidden}\cdot x_{i'}\right)\right) - y_k(x)\right)}{\partial w_{ij}^{hidden}}$$

$$\dots = -\eta\sum_{x\in T}\sum_{k=1}^{c}(f(a_k^{out}(x)) - y_k(x)) f'\left(\sum_{j'=1}^{h} w_{j'k}^{out}\, f\left(\sum_{i'=1}^{m} w_{i'j'}^{hidden}\cdot x_{i'}\right)\right)\frac{\partial\sum_{j'=1}^{h} w_{j'k}^{out} f\left(\sum_{i'=1}^{m} w_{i'j'}^{hidden}\cdot x_{i'}\right)}{\partial w_{ij}^{hidden}}$$

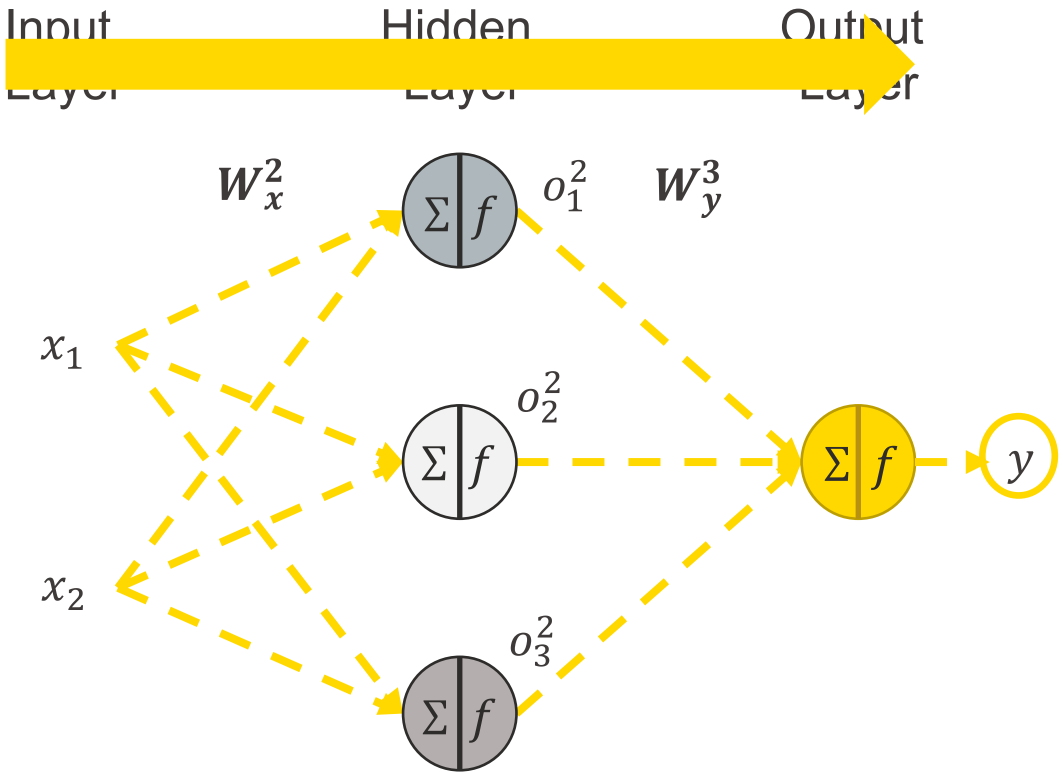
$$\dots = -\eta\sum_{x\in T}\sum_{k=1}^{c}\delta_k^{out}\frac{\partial\sum_{j'=1}^{h} w_{j'k}^{out} f\left(\sum_{i'=1}^{m} w_{i'j'}^{hidden}\cdot x_{i'}\right)}{\partial w_{ij}^{hidden}} = -\eta\sum_{x\in T}\sum_{k=1}^{c}\delta_k^{out} w_{jk}^{out}\frac{\partial f\left(\sum_{i'=1}^{m} w_{i'j}^{hidden}\cdot x_{i'}\right)}{\partial w_{ij}^{hidden}}$$

$$\dots = -\eta\sum_{x\in T}\sum_{k=1}^{c}\delta_k^{out} w_{jk}^{out} f'\left(\sum_{i'=1}^{m} w_{i'j}^{hidden}\cdot x_{i'}\right)\cdot x_i = -\eta\sum_{x\in T}\sum_{k=1}^{c}\delta_k^{out} w_{jk}^{out} f'\left(a_j^{hidden}\right)\cdot x_i$$

$$\dots = \sum_{x\in T} -\eta\cdot\delta_j^{hidden}\cdot x_i$$

Do you understand now why the sigmoid is a commonly used activation function?

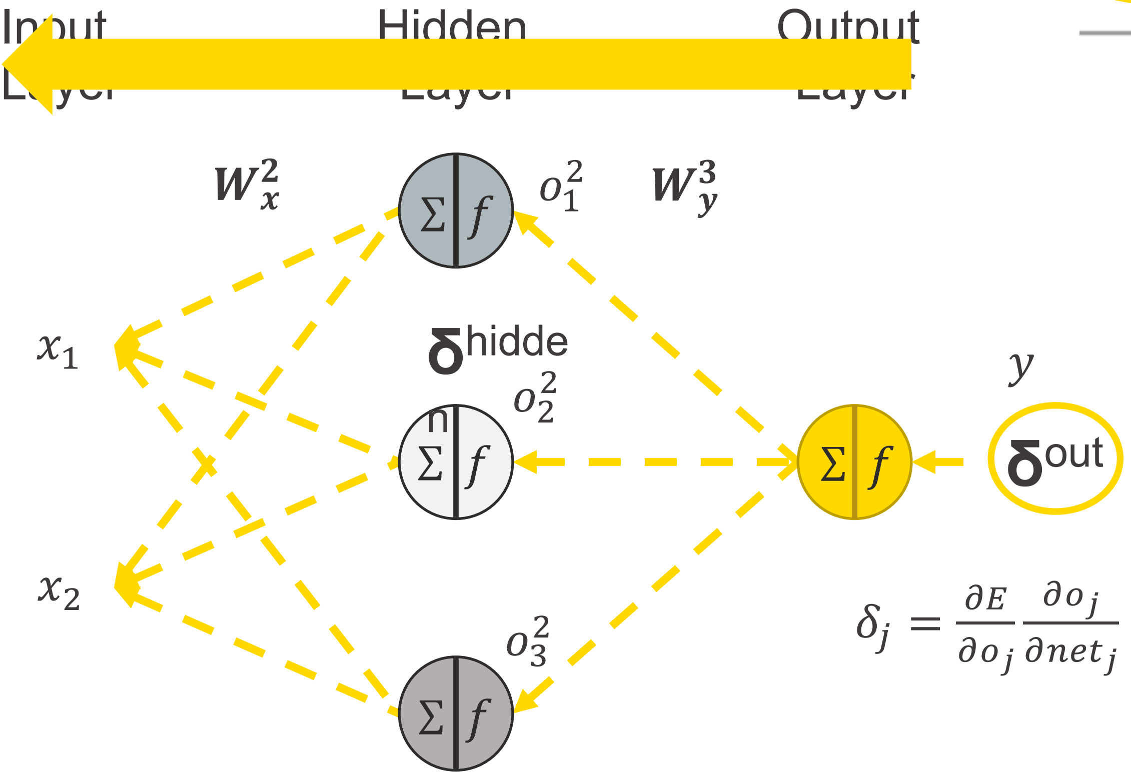
Step 1. Forward Pass

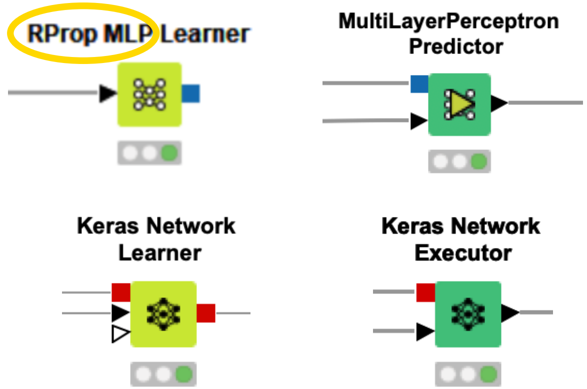


1. Forward pass:

$$\boldsymbol{o} = f(W_x^2 \boldsymbol{x})$$
$$y = f(W_y^3 \boldsymbol{o})$$

Step 1. Backward Pass





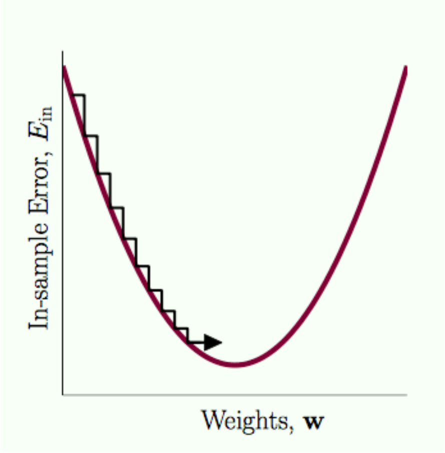
2. Backward pass:

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial net_j} = \begin{cases} (o_j - t_j)o_j(1 - o_j) \\ \left(\sum_{k \in L} w_{jk}\delta_k\right)o_j(1 - o_j) \end{cases}$$

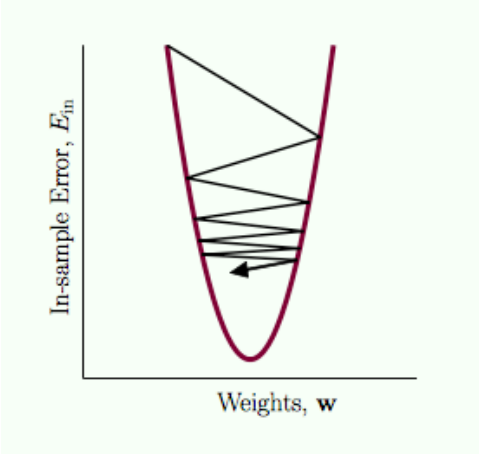
$$\Delta w_{ij} = -\eta\, o_i\, \delta_j$$

Learning Rate η

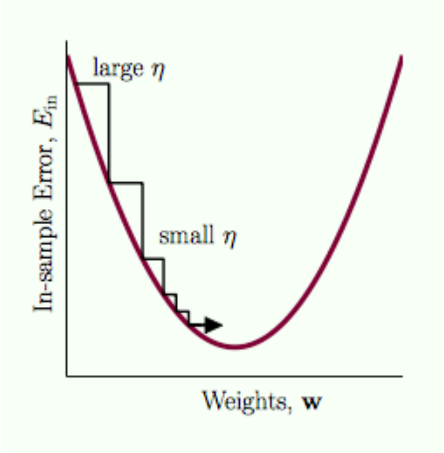
η too small



η too large



η just right



Training: Batch vs. Online

- Batch Training: Weight update after all patterns
  - correct
  - computationally expensive and slow
  - works with reasonably large learning rates (fewer updates!)
- Online Training: Weight update after each pattern
  - Approximation
    - can (in theory) run into oscillations
  - faster (fewer epochs!)
  - smaller learning rates necessary

Back-Propagation: Optimizations

- Weight Decay:
  - try to keep weights small
- Momentum:
  - increase weight updates as long as they have the same sign
- Resilient Backpropagation:
  - estimate optimum for weight based on assumption that error surface is a polynomial.

Overfitting

- MLP describe potentially very complex relationships
- Danger of fitting training data too well: Overfitting
  - Modeling of training data instead of underlying concept
    - Modeling of artifacts or outliers

Knowledge Extraction and MLPs

- MLPs are powerful but black boxes
- Rule extraction only possible in some cases
  - VI-Analysis (interval propagation)
  - extraction of decision trees
- Problems:
  - Global influence of each neuron
  - Interpretation of hidden layer(s) complicated
- Possible Solution:
  - Local activity of neurons in hidden layer: Local Basis Function Networks

Deep Learning

Recurrent Neural Networks

What are Recurrent Neural Networks?

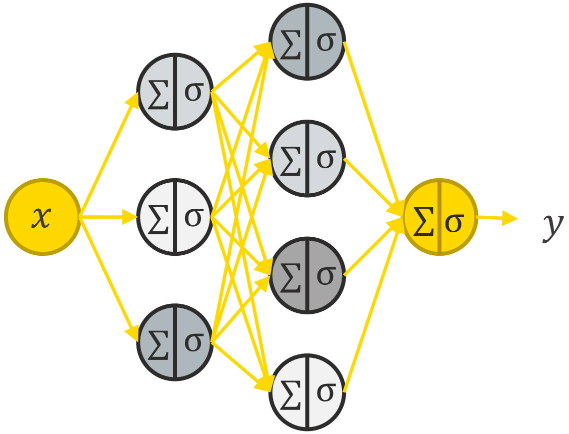
- **R**ecurrent **N**eural **N**etwork (RNN) are a family of neural networks used for processing of sequential data
- RNNs are used for all sorts of tasks:
  - Language modeling / Text generation
  - Text classification
  - Neural machine translation
  - Image captioning
  - Speech to text
  - Numerical time series data, e.g. sensor data

Why do we need RNNs for Sequential Data?

- Goal: Translation network from German to English

  “Ich mag Schokolade”

  => *“I like chocolate”*



- One option: Use feed forward network to translate word by word

Input x	Output y
Ich	I
mag	like
Schokolade	chocolate

- But what happens with this question?

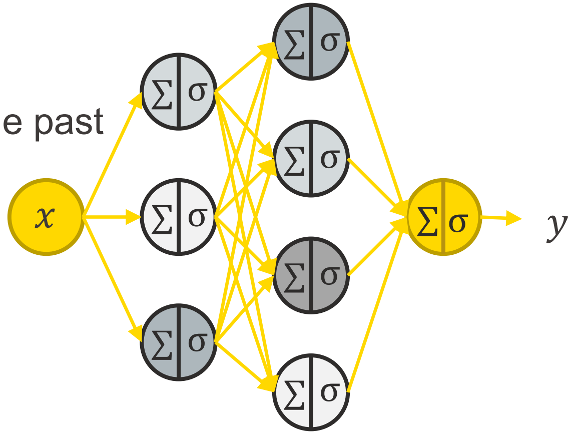
  “Mag ich Schokolade?”

  => *“Do I like chocolate?”*

Why do we need RNNs for Sequential Data?

- Problems:
  - Each time step is completely independent
  - For translations we need context
  - More general: we need a network that remembers inputs from the past

- Solution: Recurrent neural networks



Input x	Output y
Ich	I
mag	like
Schokolade	chocolate

What are RNNs?

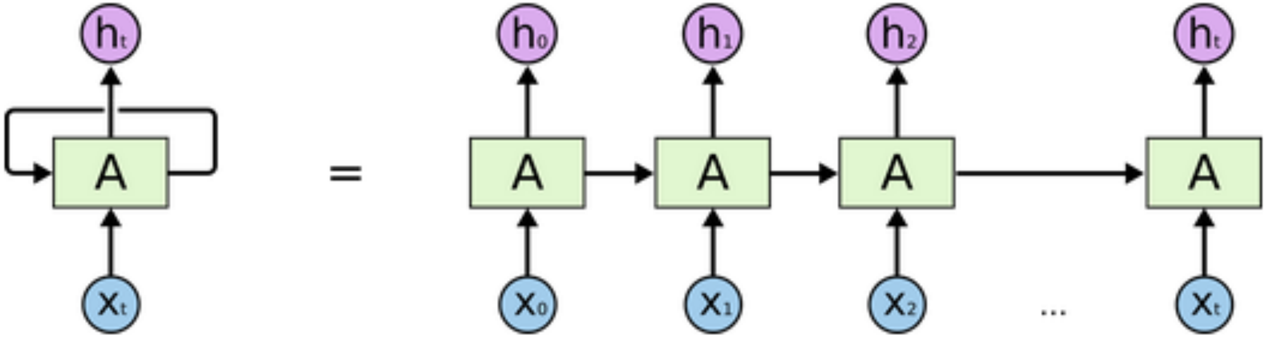
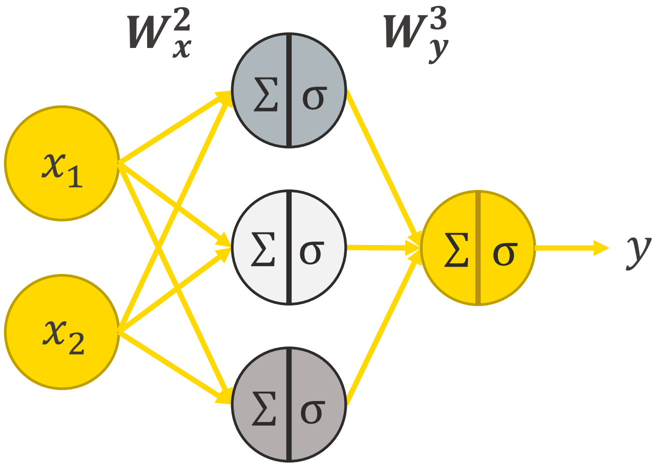
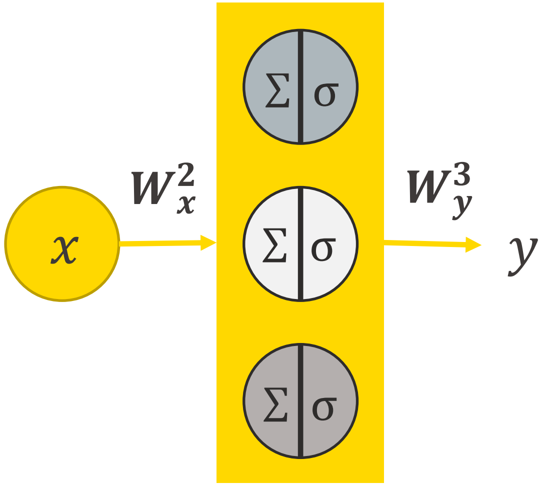


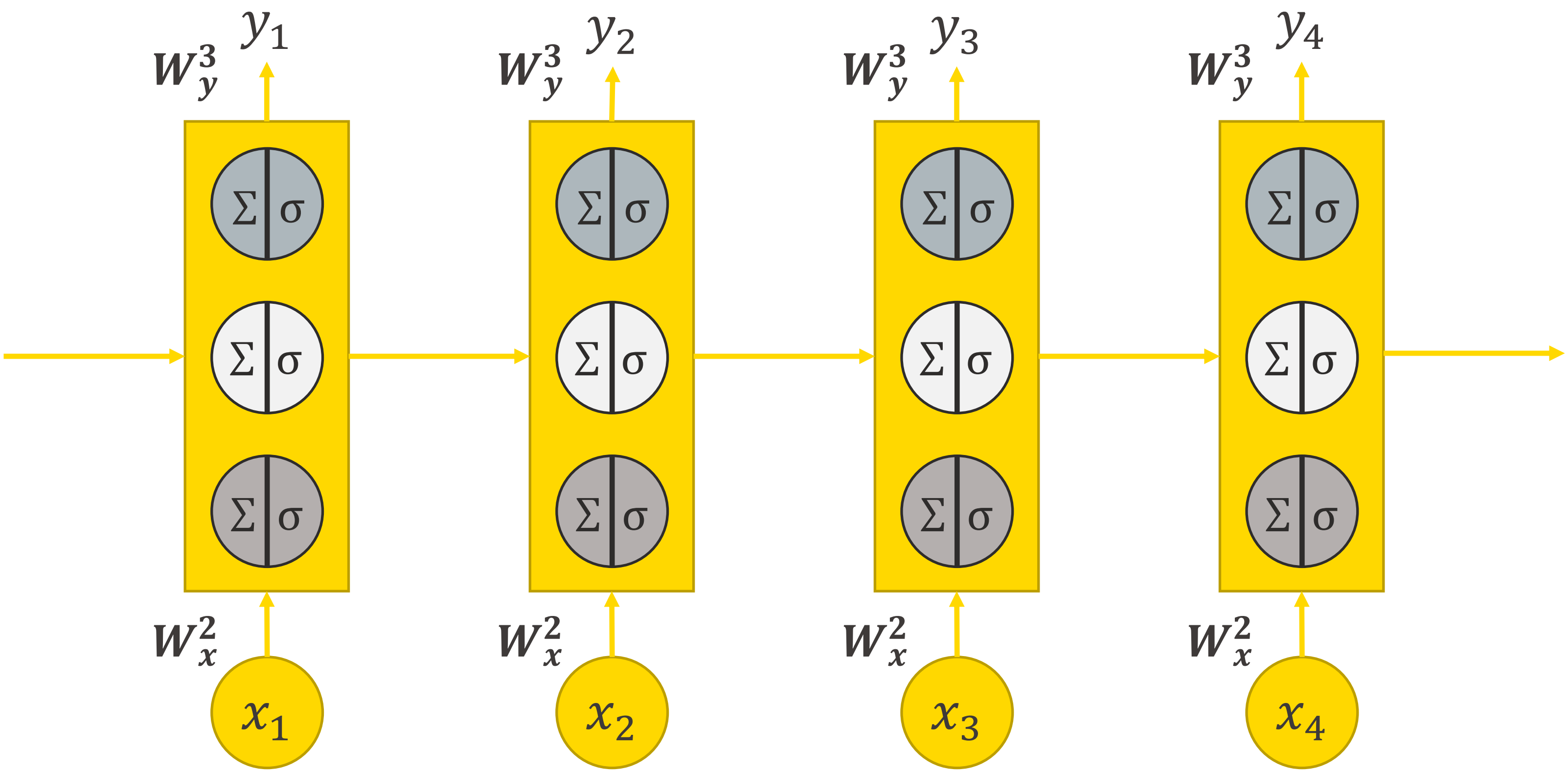
Image Source: Christopher Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

From Feed Forward to Recurrent Neural Networks





From Feed Forward to Recurrent Neural Networks



Simple RNN unit

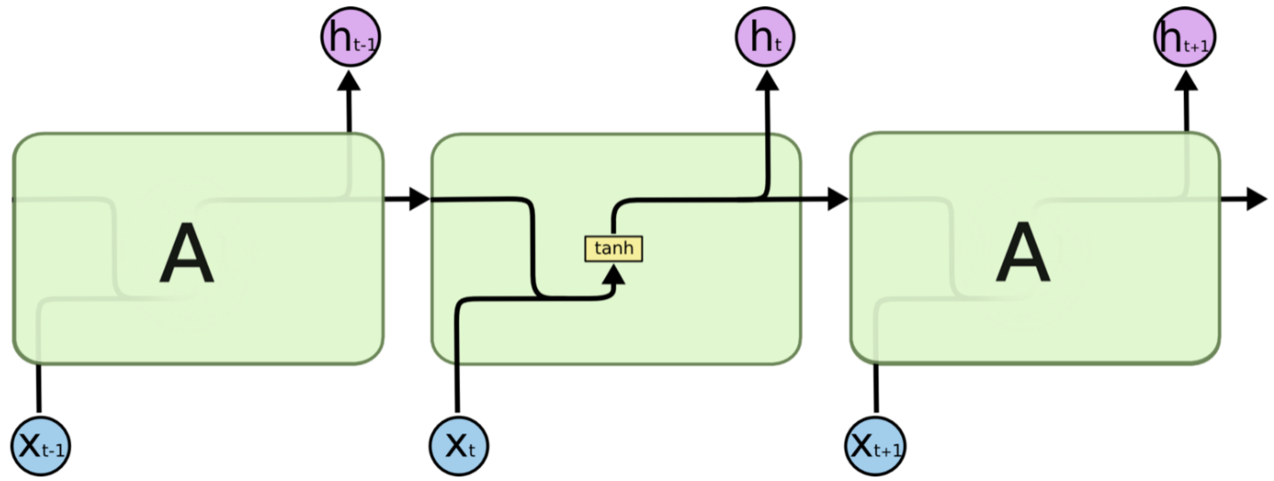


Image Source: Christopher Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Limitations of Simple Layer Structures

The “memory” of simple RNNs is sometimes too limited to be useful

- “Cars drive on the ____” (road)
- “I love the beach – my favorite sound is the crashing of the _____“ (cars? glass? waves?)

LSTM = Long Short Term Memory Unit

- Special type of unit with three gates
  - Forget gate
  - Input gate
  - Output gate

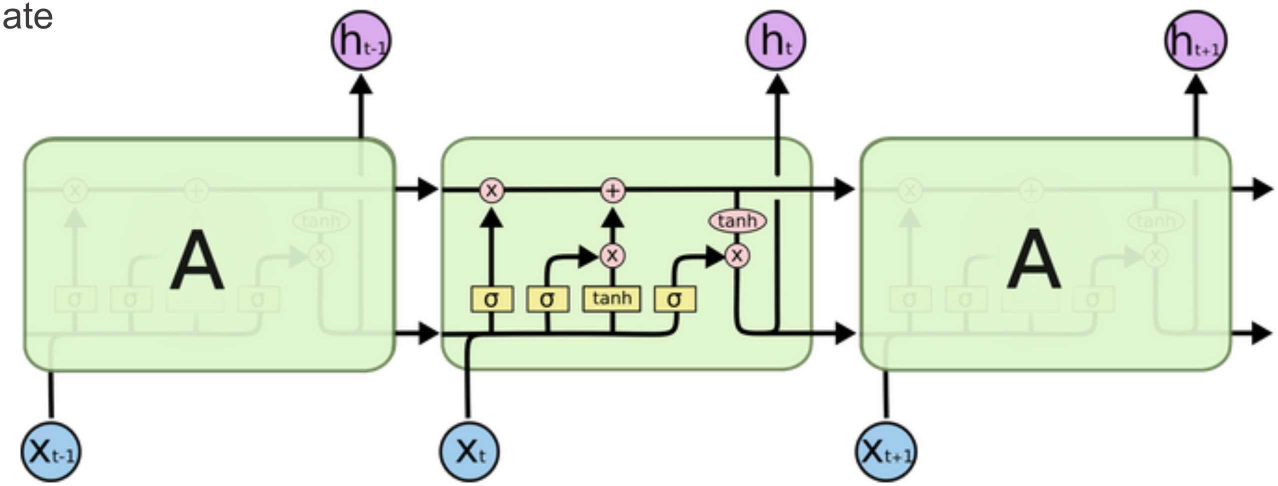
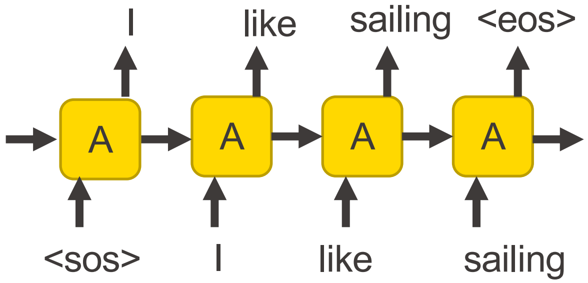


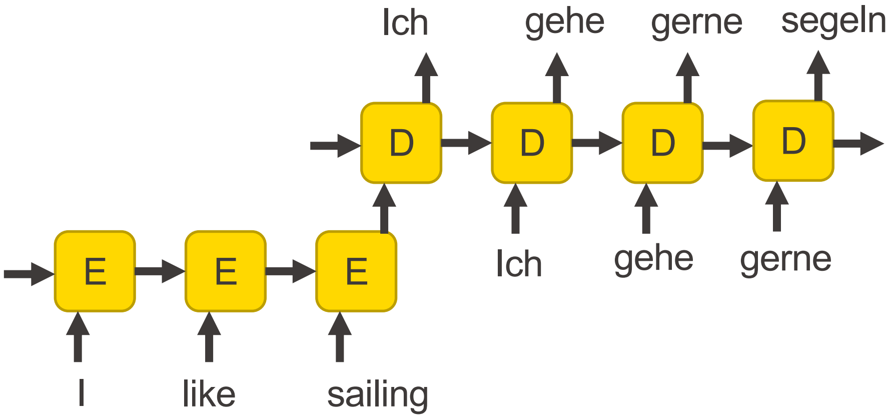
Image Source: Christopher Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Different Network-Structures and Applications

Many to Many



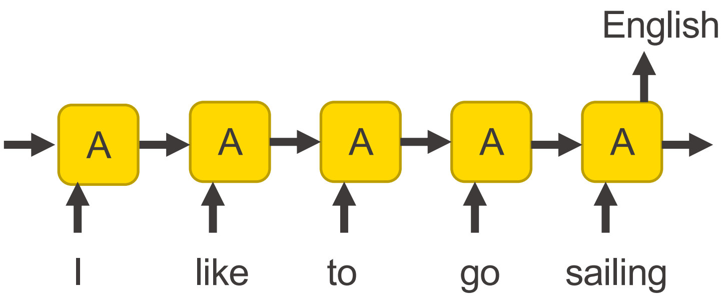
Language model



Neural machine translation

Different Network-Structures and Applications

Many to one



Language classification

Text classification

One to many

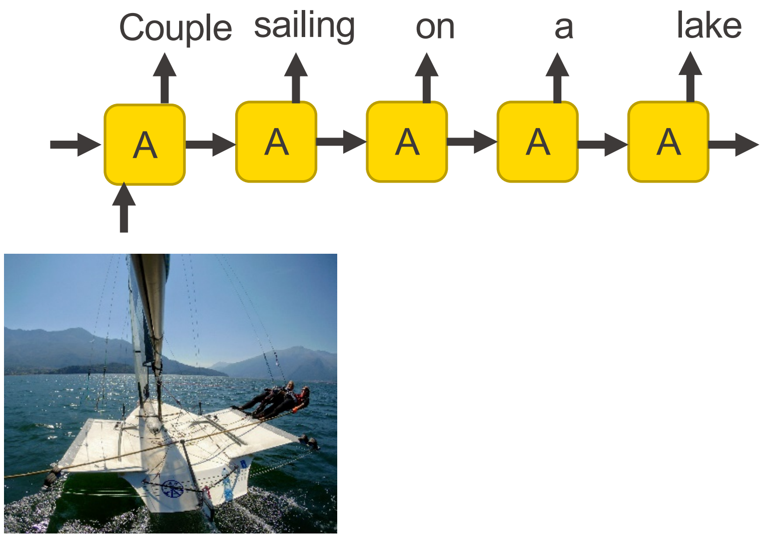
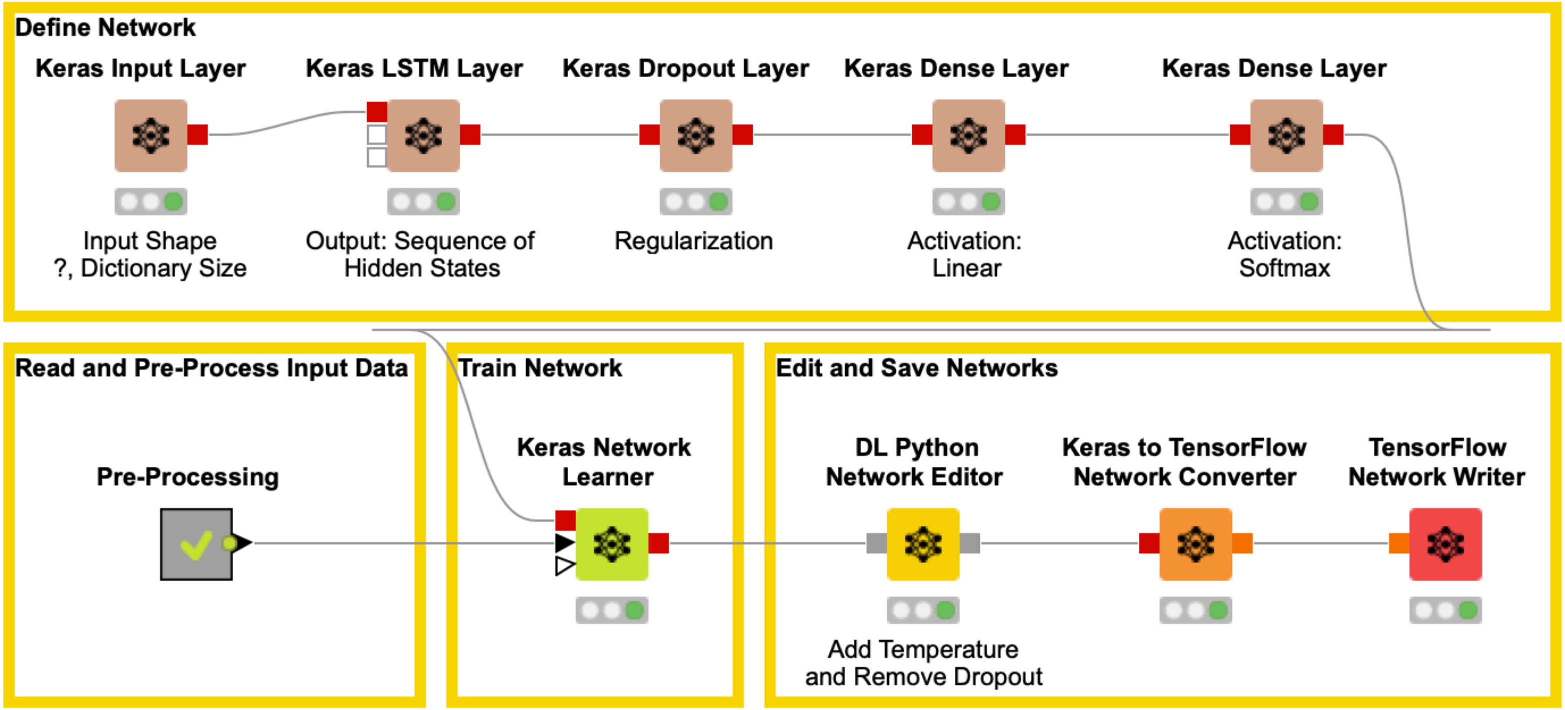


Image captioning

Neural Network: Code-free



Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN)

- Used when data has spatial relationships, e.g. images
- Instead of connecting every neuron to the new layer a sliding window is used
- Some convolutions may detect edges or corners, while others may detect cats, dogs, or street signs inside an image

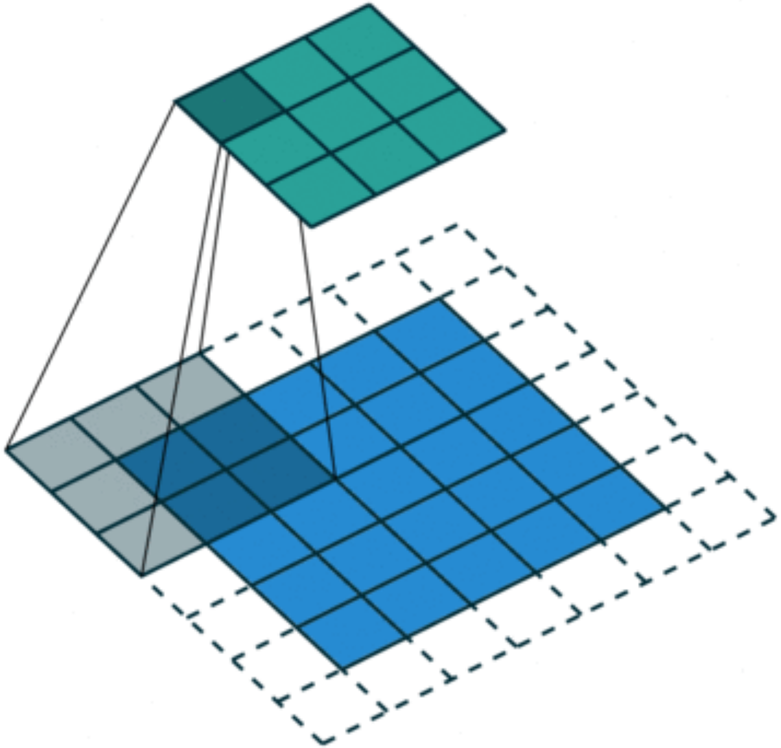
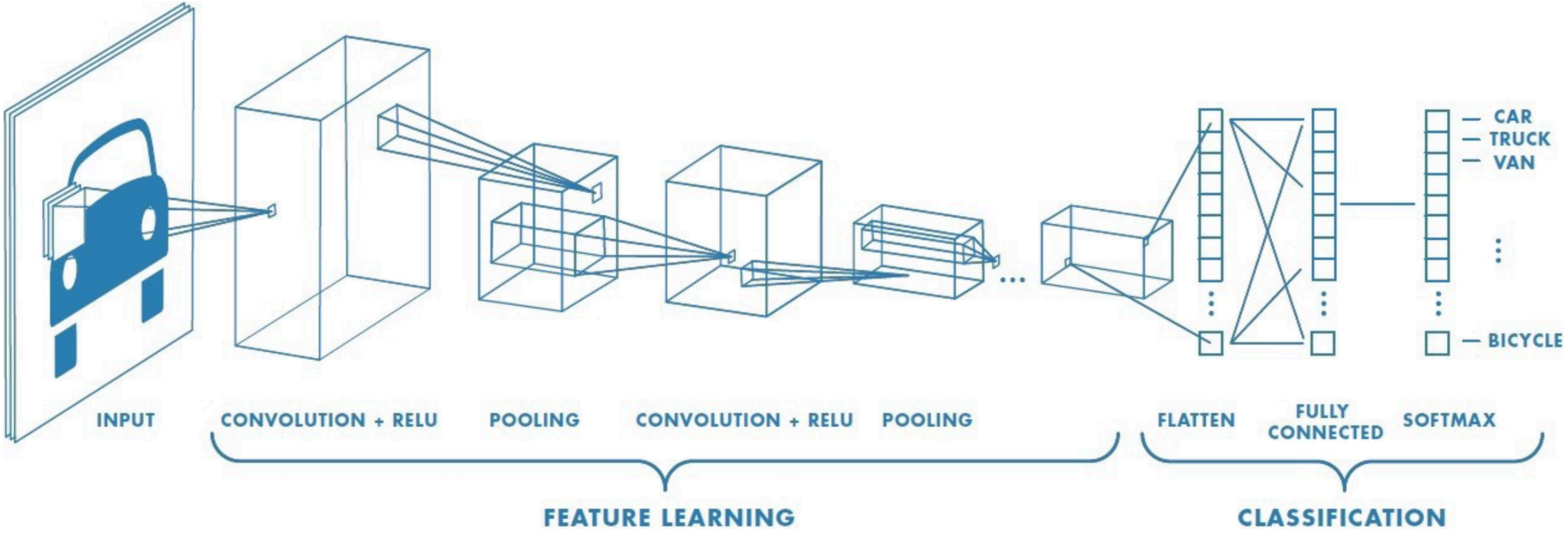
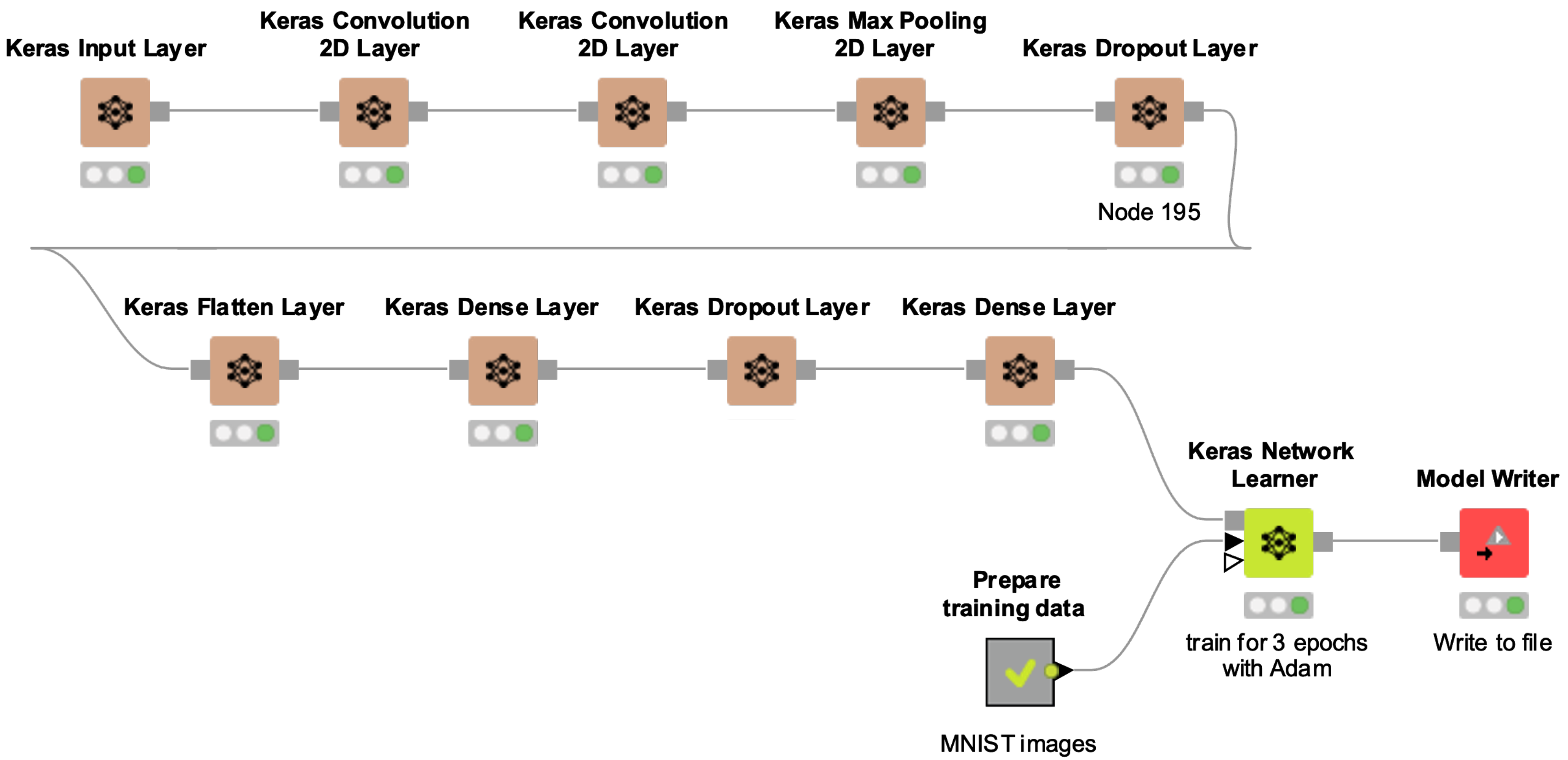


Image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional Neural Networks



Building CNNs with KNIME



Recommendation Engines

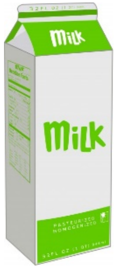
Recommendation Engines and Market Basket Analysis

From the analysis of many shopping baskets ...

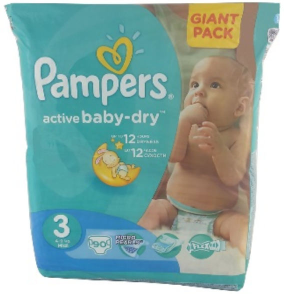
A-priori algorithm

Recommendation

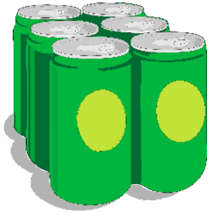
IF



+

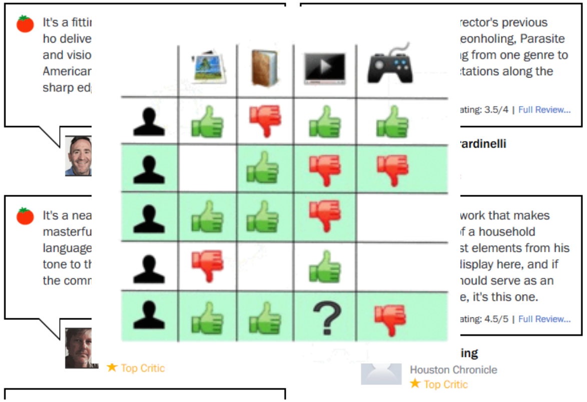


THEN



Recommendation Engines or Market Basket Analysis

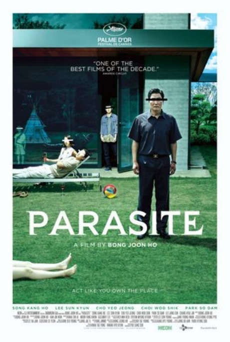
From the analysis of the reactions of many people to the same item ...



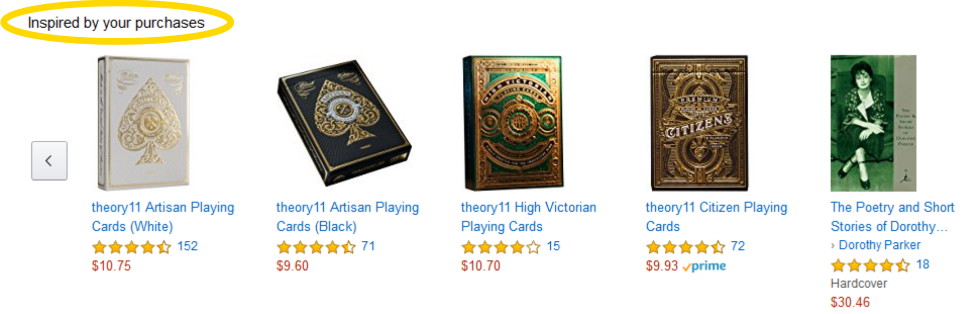
	🖼	📖	▶	🎮
👤	👍	👎	👍	👍
👤		👍	👎	👎
👤	👍	👍	👎	
👤	👎		👍	
👤	👍	👍	?	👎

Collaborative Filtering

Recommendation



PALME D'OR
"ONE OF THE BEST FILMS OF THE DECADE."
PARASITE
A FILM BY BONG JOON HO
ACT LIKE YOU OWN THE PLACE



Inspired by your purchases

- theory11 Artisan Playing Cards (White) ★★★★☆ 152 $10.75
- theory11 Artisan Playing Cards (Black) ★★★★☆ 71 $9.60
- theory11 High Victorian Playing Cards ★★★★☆ 15 $10.70
- theory11 Citizen Playing Cards ★★★★☆ 72 $9.93 ✓prime
- The Poetry and Short Stories of Dorothy... › Dorothy Parker ★★★★☆ 18 Hardcover $30.46

IF *A* has the same opinion as *B* on an item,
THEN A is more likely to have B's opinion on another item than that of a randomly chosen person

A-priori Algorithm: the Association Rule

IF

+

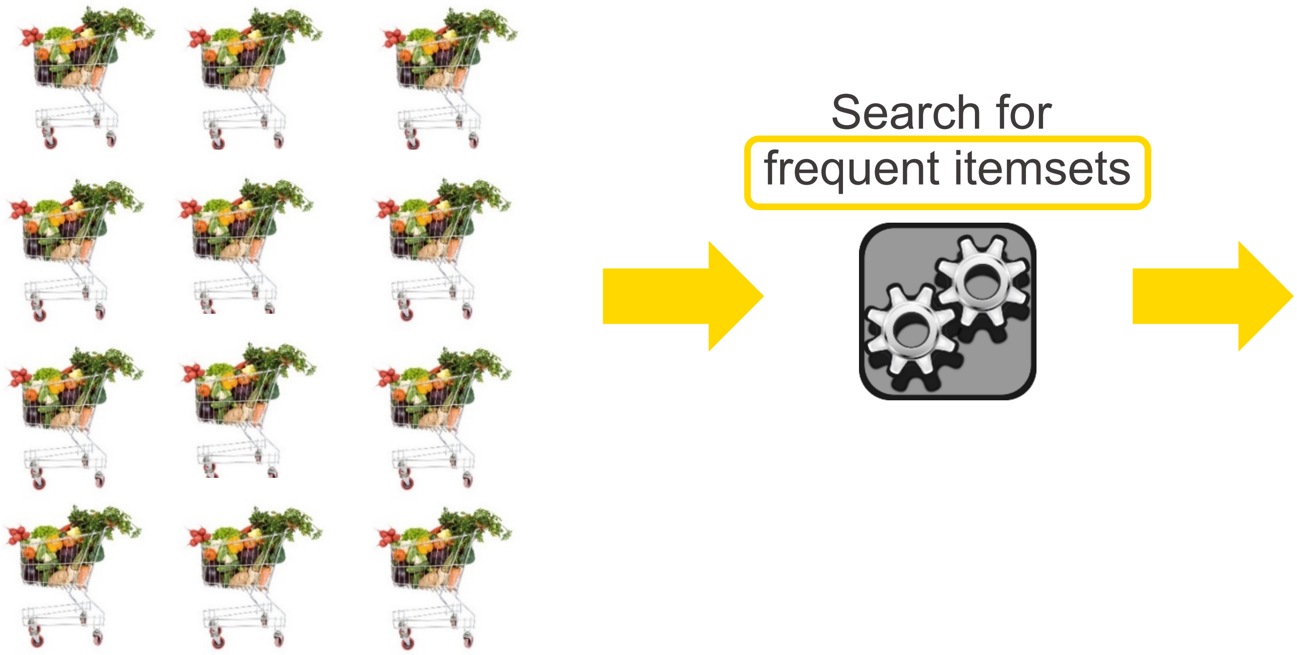
THEN

Antecedent

Consequent

Building the Association Rule

N shopping baskets



{A, B, F, H}
{A, B, C}
{B, C, H}
{D, E , F}
{D, E}
{A, B}
{A, C}
{H, F}
…

From “Frequent Itemsets“ to “Rules“

{A, B, F, H} →

{A, B, F} → H

{A, B, H} → F

{A, F, H} → B

{B, F, H} → A

Which rules shall I choose?

Support, Confidence, and Lift

$$\{A, B, F\} \rightarrow H$$

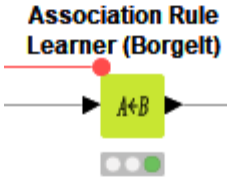
- Item set support $s = \frac{freq(A,B,F,H)}{N}$ ← How often these items are found together

- Rule confidence $c = \frac{freq(A,B,F,H)}{freq(A,B,F)}$ ← How often the antecedent is together with the consequent

- Rule lift $= \frac{support\ (\{A,B,F\} \Rightarrow H)}{support\ (A,B,F) \times support(H)}$ ← How often antecedent and consequent happen together compared with random chance

The rules with support, confidence and lift above a threshold → most reliable ones

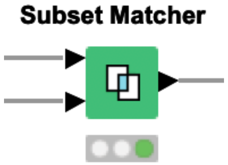
Association Rule Mining (ARM): Two Phases



Discover all frequent and strong association rules

X ⇒ Y → "if X then Y"

with sufficient support and confidence



Two phases:

1. find all frequent itemsets (FI) ← Most of the complexity
   - Select itemsets with a minimum support

$$FI = \{\{X,Y\}, X,Y \subset I | s(X,Y) \geq S_{min}\}$$

2. build strong association rules
   - Select rules with a minimum confidence:

$$Rules: \{X \Rightarrow Y, X,Y \subset FI, | c(X \Rightarrow Y) \geq C_{min}\}$$

User parameters

A-Priori Algorithm: Example

- Let‘s consider milk, diaper, and beer: $\{milk, diaper\} \Rightarrow beer$

- How often are they found together across all shopping baskets?
- How often are they found together across all shopping baskets containing the antecedents?

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

support

$$s(milk, diaper, beer) = \frac{P(milk, diaper, beer)}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{P(milk, diaper, beer)}{P(milk, diaper)} = \frac{2}{3} = 0.67$$

confidence

A-priori algorithm: an example

- Let‘s consider milk, diaper, and beer: $\{milk, diaper\} \Rightarrow beer$
- How often are they found together across all shooping baskets?
- How often are they found together across all shopping baskets containing the antecedents?

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$s(milk, diaper) = \frac{P(milk, diaper)}{|T|} = \frac{3}{5} = 0.6$$

$$s(beer) = \frac{P(beer)}{|T|} = \frac{3}{5} = 0.6$$

$$Rule\ lift = \frac{s(milk, diaper, beer)}{s(milk, diaper) \times s(beer)}$$

$$= \frac{0.4}{0.6 \times 0.6} = 1.11$$

Association Rule Mining: Is it Useful?

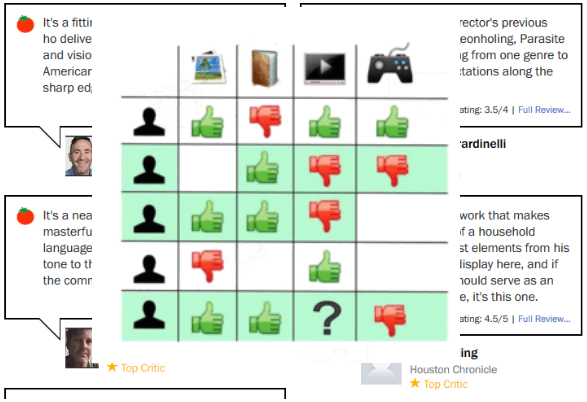
- David J. Hand (2004):
  „Association Rule Mining is likely the field with the highest ratio of number of published papers per reported application.“

- KNIME Blog post:

https://www.knime.com/knime-applications/market-basket-analysis-and-recommendation-engines

Recommendation Engines or Market Basket Analysis

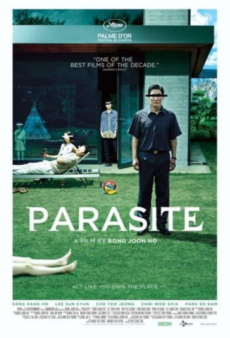
From the analysis of the reactions of many people to the same item ...



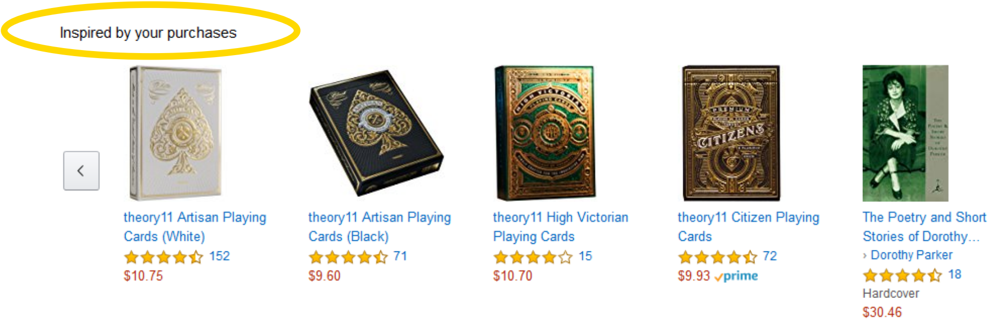
	🖼	📖	🎬	🎮
👤	👍	👎	👍	👍
👤		👍	👎	👎
👤	👍	👍	👎	
👤	👎		👍	
👤	👍	👍	?	👎

Collaborative Filtering

Recommendation



PALME D'OR
"ONE OF THE BEST FILMS OF THE DECADE."
PARASITE
A FILM BY BONG JOON HO
ACT LIKE YOU OWN THE PLACE



Inspired by your purchases

theory11 Artisan Playing Cards (White)
★★★★☆ 152
$10.75

theory11 Artisan Playing Cards (Black)
★★★★☆ 71
$9.60

theory11 High Victorian Playing Cards
★★★★☆ 15
$10.70

theory11 Citizen Playing Cards
★★★★☆ 72
$9.93 ✓prime

The Poetry and Short Stories of Dorothy...
› Dorothy Parker
★★★★☆ 18
Hardcover
$30.46

IF *A* has the same opinion as *B* on an item,
THEN A is more likely to have B's opinion on another item than that of a randomly chosen person

Collaborative Filtering (CF)

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

1. Look for users who share the same rating patterns with the active user (the user whom the recommendation is for)
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user
3. Implemented in Spark

https://www.knime.com/blog/movie-recommendations-with-spark-collaborative-filtering

Collaborative Filtering: Memory based approach

- User u to give recommendations to
- U = set of top N users most similar to user u
- Rating of user u on item i calculated as average of ratings of all similar users in U:

$$r_{u,i} = \frac{1}{N}\sum_{u'\in U} r_{u',i} \quad \text{or weighted} \quad r_{u,i} = \frac{1}{N}\sum_{u'\in U} simil(u,u')\; r_{u',i}$$

Pearson correlation

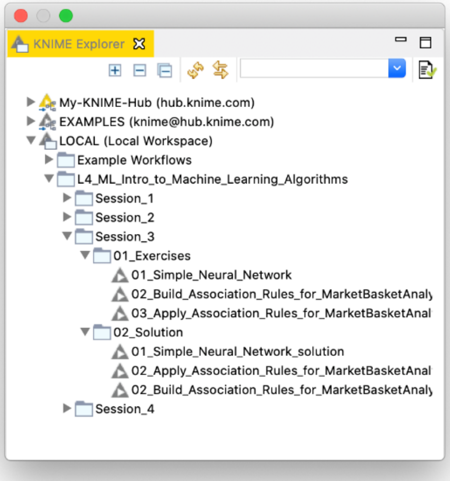
$$simil(u,u') = \frac{\sum_{i\in I_{xy}}\left(r_{u,i}-\overline{r_u}\right)\left(r_{u',i}-\overline{r_{u'}}\right)}{\sqrt{\sum_{i\in I_{xy}}\left(r_{u,i}-\overline{r_u}\right)^2}\sqrt{\sum_{i\in I_{xy}}\left(r_{u',i}-\overline{r_{u'}}\right)^2}}$$

Set of items rated by both user x and y

Exercises:

- Neural Network
  - Goal: Train an MLP to solve our classification problem (rank: high/low)
  - 01_Simple_Neural_Network

- Market Basket Analysis
  - 02_Build_Association_Rules_for_MarketBasketAnalysis
  - 03_Apply_Association_Rules_for_MarketBasketAnalysis



KNIME Explorer

- My-KNIME-Hub (hub.knime.com)
- EXAMPLES (knime@hub.knime.com)
- LOCAL (Local Workspace)
  - Example Workflows
  - L4_ML_Intro_to_Machine_Learning_Algorithms
    - Session_1
    - Session_2
    - Session_3
      - 01_Exercises
        - 01_Simple_Neural_Network
        - 02_Build_Association_Rules_for_MarketBasketAnaly
        - 03_Apply_Association_Rules_for_MarketBasketAnal
      - 02_Solution
        - 01_Simple_Neural_Network_solution
        - 02_Apply_Association_Rules_for_MarketBasketAnal
        - 02_Build_Association_Rules_for_MarketBasketAnaly
    - Session_4

Session 4: Clustering & Data Preparation

Unsupervised Learning: Clustering

Goal of Cluster Analysis

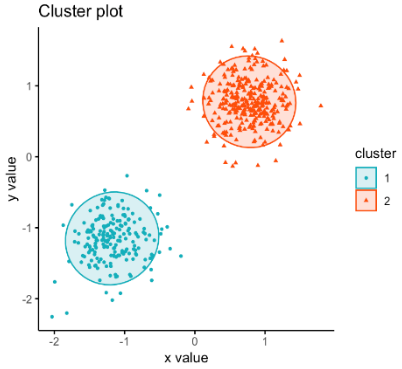
Discover hidden structures in unlabeled data (unsupervised)

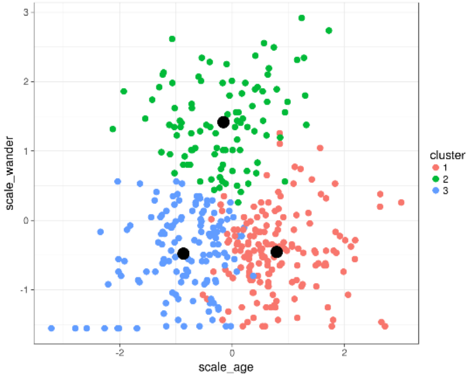
Clustering identifies a finite set of groups (*clusters*) $C_1, C_2 \cdots, C_k$ in the dataset such that:

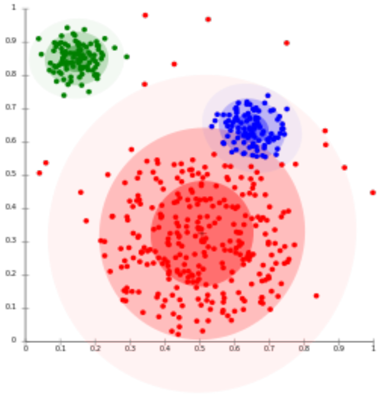
- Objects within the *same* cluster C_i shall be as similar as possible
- Objects of *different* clusters C_i, C_j $(i \neq j)$ shall be as dissimilar as possible

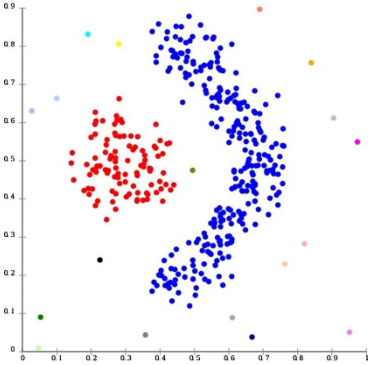
Cluster Properties

- Clusters may have different sizes, shapes, densities
- Clusters may form a hierarchy
- Clusters may be overlapping or disjoint









Clustering Applications

- Find “natural” clusters and desc
  - Data understanding
- Find useful and suitable groups
  - Data Class Identification
- Find representatives for homogenous groups
  - Data Reduction
- Find unusual data objects
  - Outlier Detection
- Find random perturbations of the data
  - Noise Detection

Methods

- K-means
- Hierarchical
- DBScan

Examples

- Customer segmentation
- Molecule search
- Anomaly detection

Clustering as Optimization Problem

Definition:
Given a data set $D, |D| = n$. Determine a *clustering* C *of* D with:

$$C = \{C_1, C_2, \cdots, C_{\mathrm{k}}\} \quad \text{where} \quad C_i \subseteq D \quad \text{and} \bigcup_{1 \le i \le k} C_i = D$$

that best fits the given data set D.

Inside the space

Cover the whole space

Clustering Methods:

1. partitioning
2. hierarchical (linkage based)
3. density-based

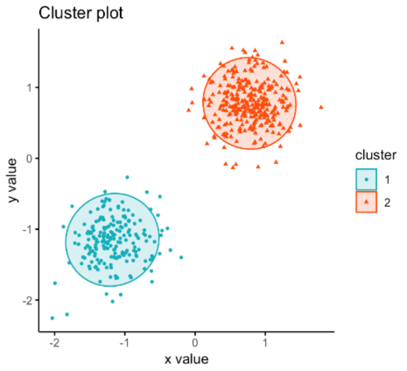
Clustering: Partitioning
k-Means

Partitioning

Goal:

A (disjoint) partitioning into k clusters with minimal costs

- Local optimization method:
  - choose k initial cluster representatives
  - optimize these representatives iteratively
  - assign each object to its most similar cluster representative
- Types of cluster representatives:
  - Mean of a cluster (*construction of central points*)
  - Median of a cluster (*selection of representative points*)
  - Probability density function of a cluster (*expectation maximization*)



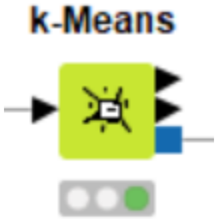
k-Means-Algorithm

Given k, the k-Means algorithm is implemented in four steps:

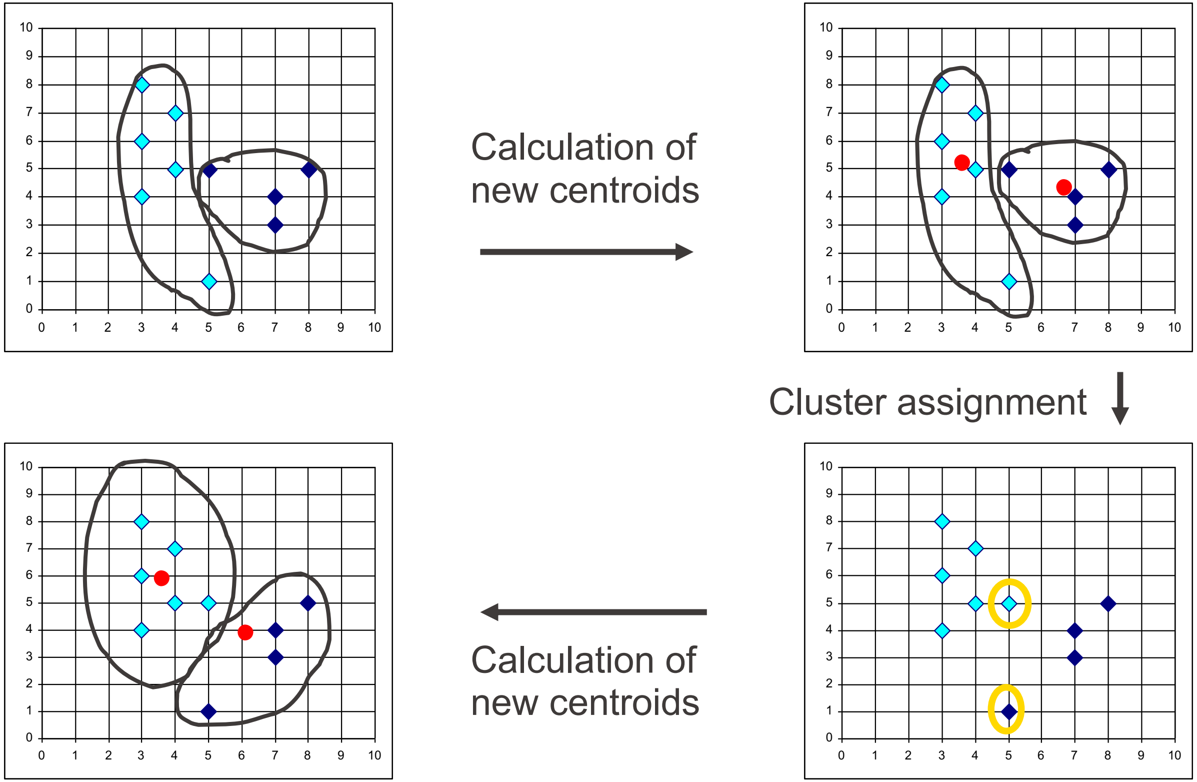
1. Partition objects into k non-empty subsets, calculate their **centroids** (i.e., **mean point**, of the cluster)
2. Assign each object to the cluster with the **nearest** centroid

Euclidean distance

3. Compute the centroids from the current partition
4. Go back to Step 2, repeat until the updated centroids stop moving significantly



k-Means Algorithm



Calculation of new centroids

Cluster assignment

Calculation of new centroids

Comments of the k-Means Method

- Advantages:
  - Relatively efficient
  - Simple implementation
- Weaknesses:
  - Often terminates at a local optimum
  - Applicable only when mean is defined (what about categorical data?)
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

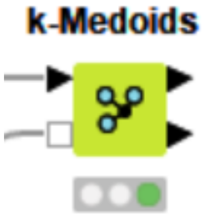
Outliers: k-Means vs k-Medoids

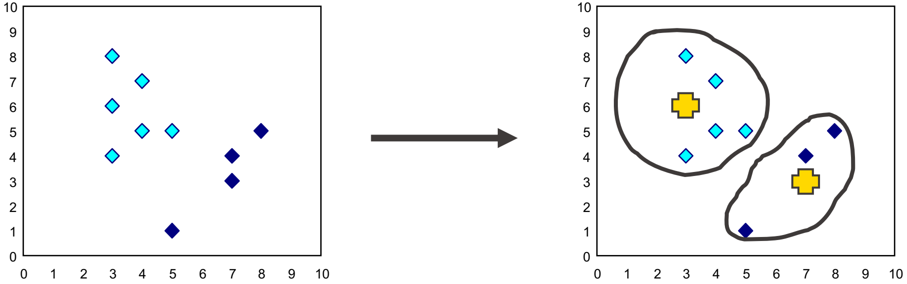
Problem with K-Means

An object with an extremely large value can substantially distort the distribution of the data.

One solution: K-Medoids

Instead of taking the **mean** value of the objects in a cluster as a reference point, **medoids** can be used, which are the most centrally located objects in a cluster.



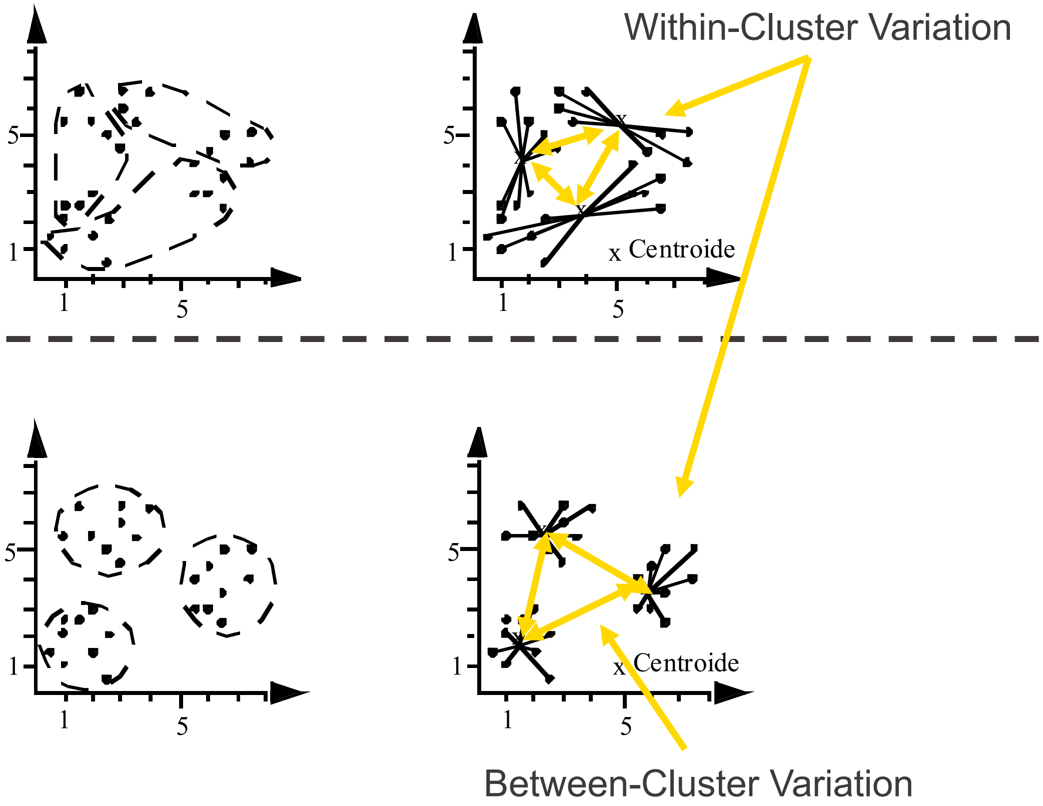


Clustering: Quality Measures
Silhouette

Optimal Clustering: Example

Bad Clustering

Good Clustering



Within-Cluster Variation

Between-Cluster Variation

Cluster Quality Measures

Centroid μ_C: mean vector of all objects in clustering C

- Within-Cluster Variation:

$$TD^2 = \sum_{i=1}^{k} \sum_{p \in \boldsymbol{C_i}} dist(p, \mu_{C_i})^2$$

- Between-Cluster Variation:

$$BC^2 = \sum_{j=1}^{k} \sum_{i=1}^{k} dist(\mu_{C_j}, \mu_{C_i})^2$$

- Clustering Quality (one possible measure):

$$CQ = \frac{BC^2}{TD^2}$$

Silhouette-Coefficient for object x

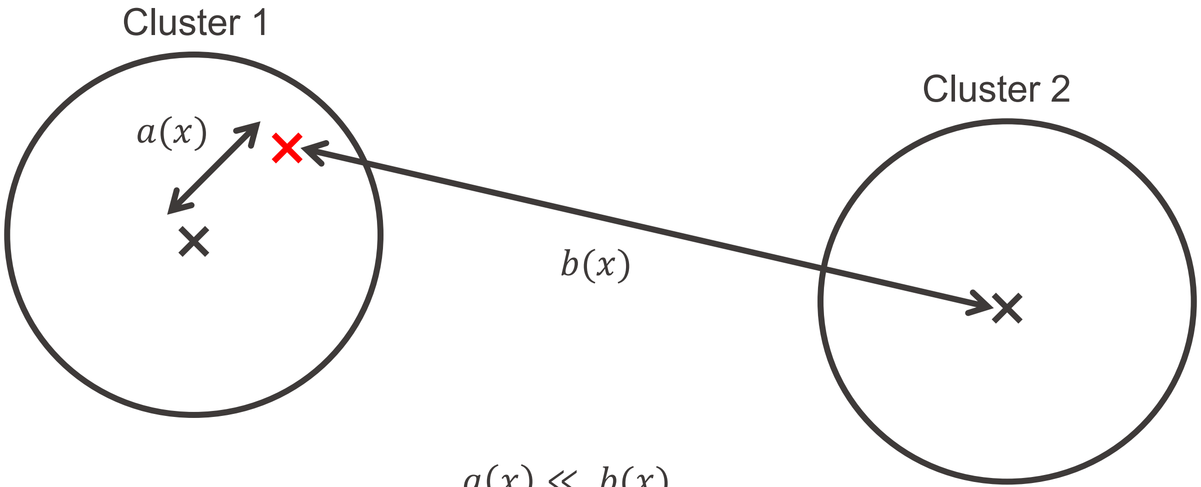
Silhouette-Coefficient [Kaufman & Rousseeuw 1990] measures the quality of clustering

- $a(x)$: distance of object x to its cluster representative
- $b(x)$: distance of object x to the representative of the „second-best“ cluster
- **Silhouette** $s(x)$ of x

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}}$$

Silhouette-Coefficient

Good clustering…

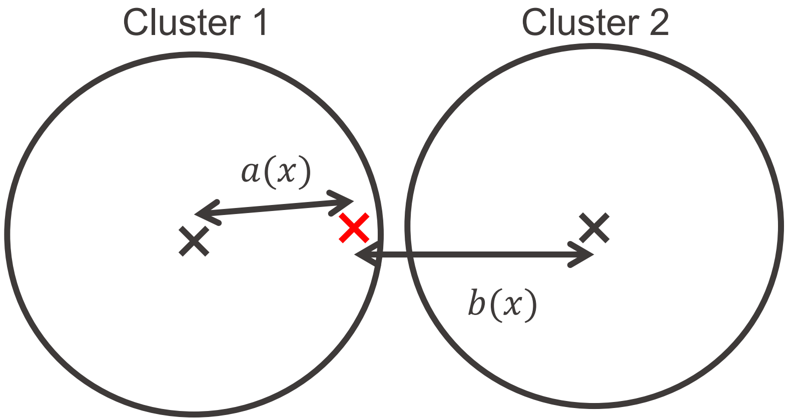


$$a(x) \ll \; b(x)$$

$$s(x) = \frac{b(x) - a(x)}{\max\{\, a(x), b(x)\}} \approx \frac{b(x)}{b(x)} = 1$$

Silhouette-Coefficient

…not so good…

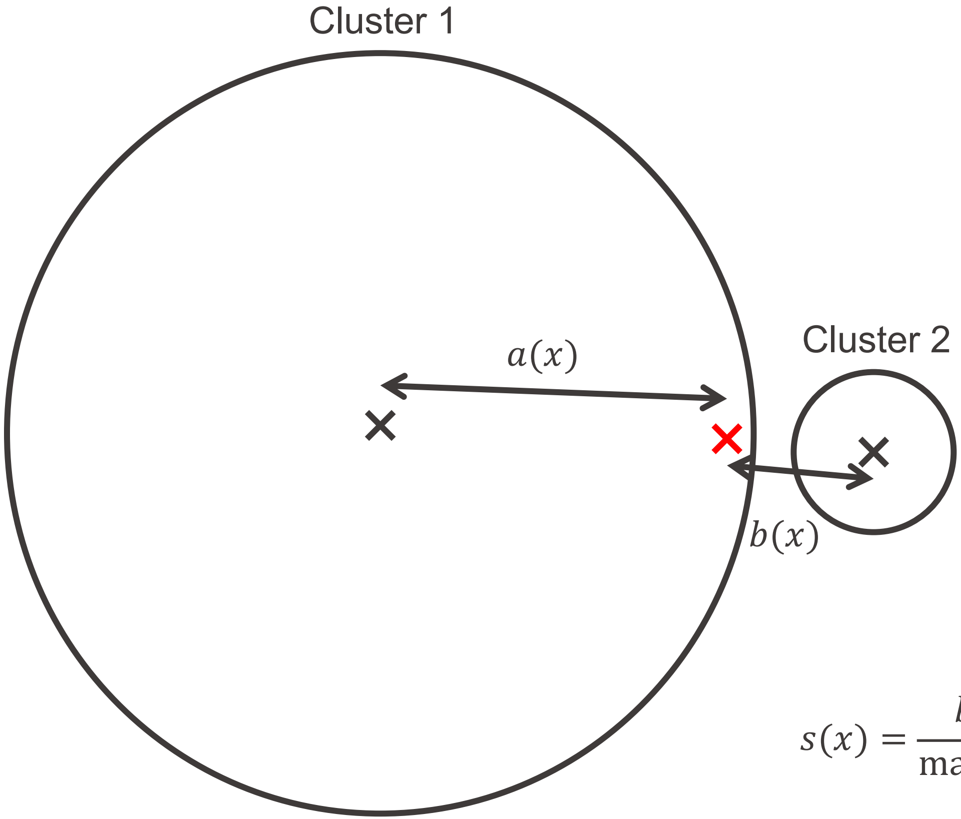


$$a(x) \approx b(x)$$

$$s(x) = \frac{b(x) - a(x)}{\max\{ a(x), b(x)\}} \approx \frac{0}{b(x)} = 0$$

Silhouette-Coefficient

…bad clustering.



$$a(x) \gg b(x)$$

$$s(x) = \frac{b(x) - a(x)}{\max\{\, a(x), b(x)\}} \approx \frac{-a(x)}{a(x)} = -1$$

Silhouette-Coefficient for Clustering C

- Silhouette coefficient s_C for clustering C is the average silhouette over all objects $x \in C$

$$s_C = \frac{1}{n} \sum_{x \in C} s(x)$$

- Interpretation of silhouette coefficient:
  - $s_C > 0.7$: strong cluster structure,
  - $s_C > 0.5$: reasonable cluster structure,
  - $\ldots$

Choice of Parameter k

Method

- For k=2, 3, $\cdots$, n−1, determine one clustering each
- Choose k resulting in the highest clustering quality

Measure of clustering quality

- Uncorrelated with k
- for k-means and k-medoid:

  TD^2 and TD decrease monotonically with increasing k

Summary: Clustering by Partitioning

- Scheme always similar:
  - Find (random) starting clusters
  - Iteratively improve cluster positions
    (compute new mean, swap medoids, compute new distribution parameters,…)
- Important:
  - Number of clusters k
  - Initial cluster position influences (heavily):
    - quality of results
    - speed of convergence
- Problems for iterative clustering methods:
  - Clusters of varied size, density and shape

Clustering: Distance Functions

Distance Functions for Numeric Attributes

For two objects $x = (x_1, x_2, \cdots, x_d)$ and $y = (y_1, y_2, \cdots, y_d)$:

- *L_p-Metric (Minkowski-Distance)*

$$dist(x, y) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

- *Euclidean Distance* $(p = 2)$

$$dist(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- *Manhattan-Distance* $(p = 1)$

$$dist(x, y) = \sum_{i=1}^{d} |x_i - y_i|$$

- *Maximum-Distance* $(p = \infty)$

$$dist(x, y) = \max_{1 \le i \le d} \{|x_i - y_i|\}$$

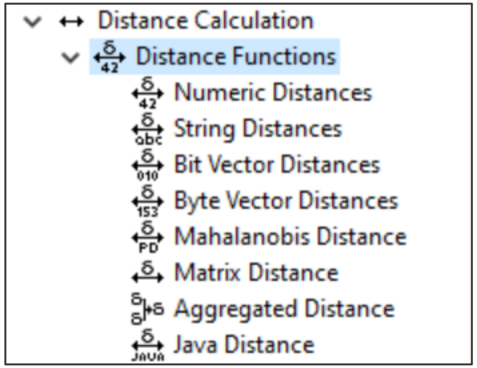
Influence of Distance Function / Similarity

- Clustering vehicles:
  - red Ferrari
  - green Porsche
  - red Bobby car

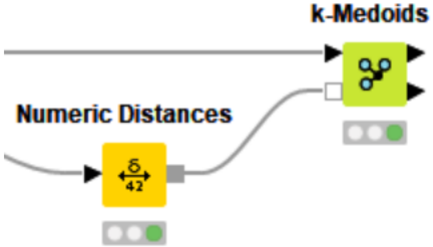
- Distance Function based on maximum speed (numeric distance function):
  - Cluster 1: Ferrari & Porsche
  - Cluster 2: Bobby car

- Distance Function based on color (nominal attributes):
  - Cluster 1: Ferrari and Bobby car
  - Cluster 2: Porsche

The distance function affects the shape of the clusters



- Distance Calculation
  - Distance Functions
    - Numeric Distances
    - String Distances
    - Bit Vector Distances
    - Byte Vector Distances
    - Mahalanobis Distance
    - Matrix Distance
    - Aggregated Distance
    - Java Distance

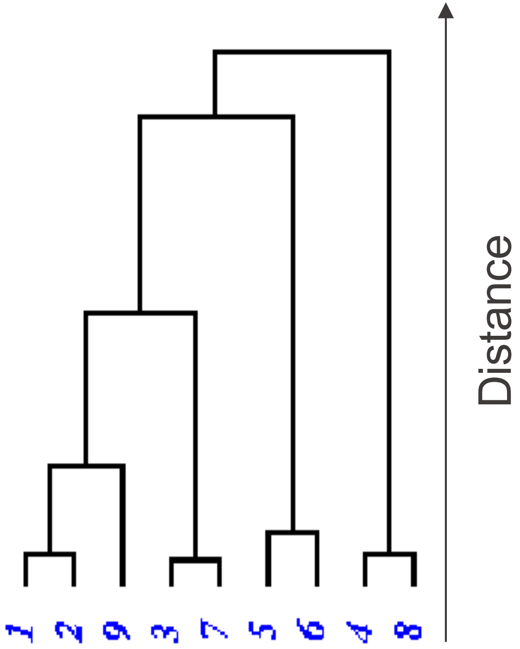


Clustering: Linkage Hierarchical Clustering

Linkage Hierarchies: Basics

Goal

- Construction of a hierarchy of clusters (*dendrogram*) by merging/separating clusters with minimum/maximum distance

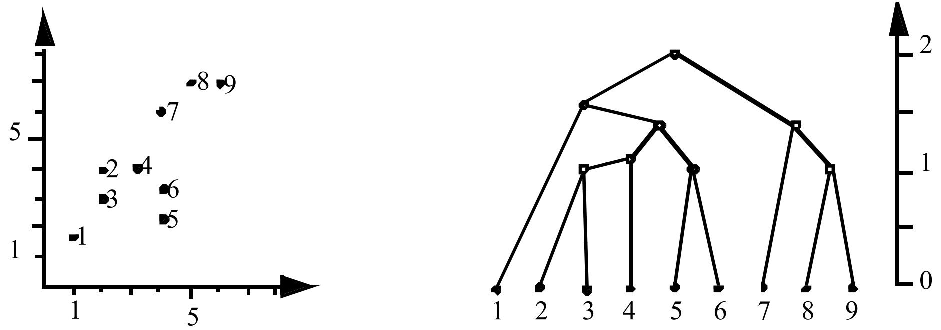


Dendrogram:

- A tree representing hierarchy of clusters, with the following properties:
  - Root: single cluster with the whole data set.
  - Leaves: clusters containing a single object.
  - Branches: merges / separations between larger clusters and smaller clusters / objects

Linkage Hierarchies: Basics

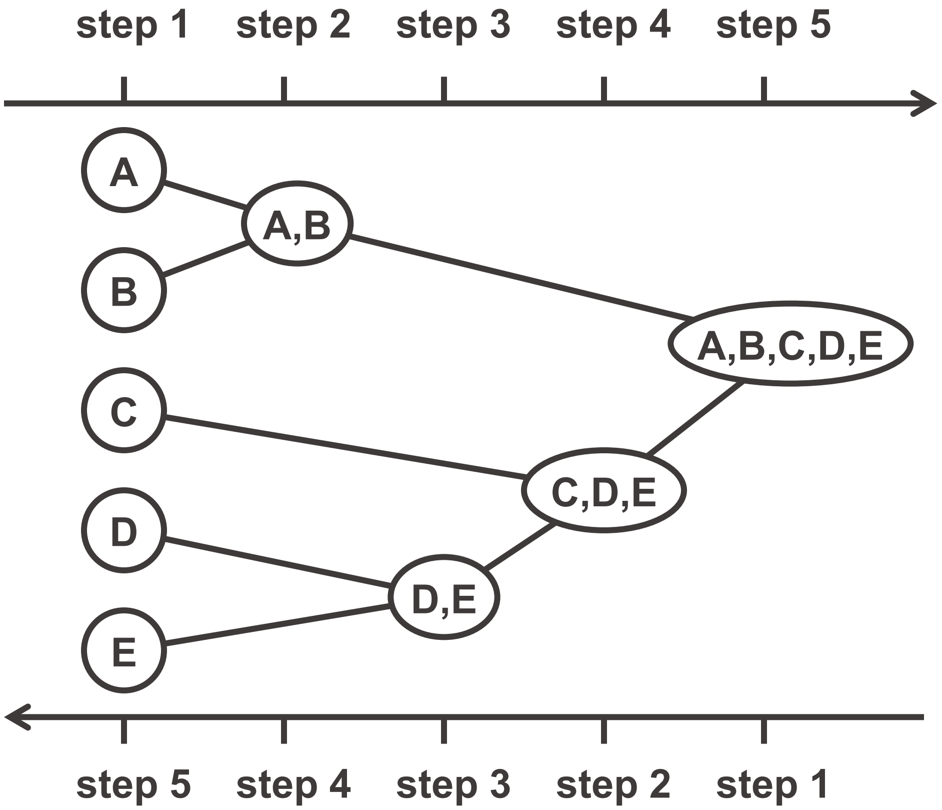
- Example dendrogram



distance between clusters

- Types of hierarchical methods
  - Bottom-up construction of dendrogram (*agglomerative*)
  - Top-down construction of dendrogram (*divisive*)

Agglomerative vs. Divisive Hierarchical Clustering



AGlomerative NESting
(AGNES)

DIvisive ANAlysis
(DIANA)

Base Algorithm

1. Form initial clusters consisting of a single object, and compute the distance between each pair of clusters.
2. Merge the two clusters having minimum distance.
3. Calculate the distance between the new cluster and all other clusters.
4. If there is only one cluster containing all objects:
   Stop, otherwise go to step 2.

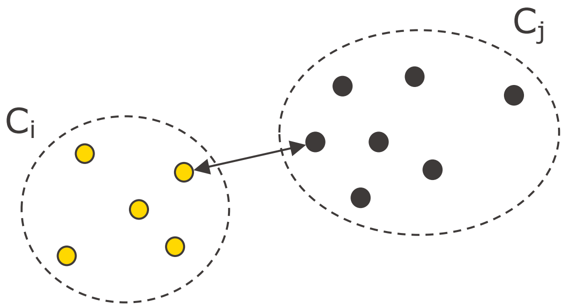
Single Linkage

- Distance between clusters (nodes):

$$Dist(C_1, C_2) = \min_{p \in C_1, q \in C_2} \{dist(p, q)\}$$

Distance of the closest two points, one from each cluster

- Merge Step: Union of two subsets of data points



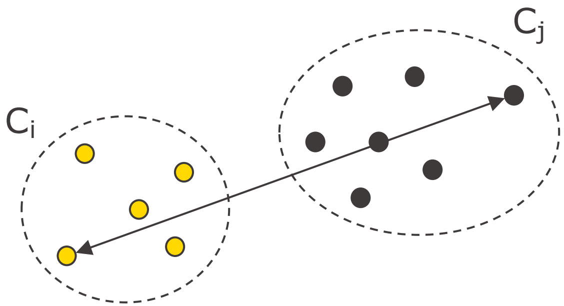
Complete Linkage

- Distance between clusters (nodes):

$$Dist(C_1, C_2) = \max_{p \in C_1, q \in C_2} \{dist(p,q)\}$$

Distance of the farthest two points, one from each cluster

- Merge Step: Union of two subsets of data points



Average Linkage / Centroid Method

- Distance between clusters (nodes):

$$Dist_{avg}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p \in C_1} \sum_{p \in C_2} dist(p, q)$$

Average distance of all possible pairs of points between C_1 and C_2

$$Dist_{mean}(C_1, C_2) = dist\big(mean(C_1), mean(C_2)\big)$$

Distance between two centroids

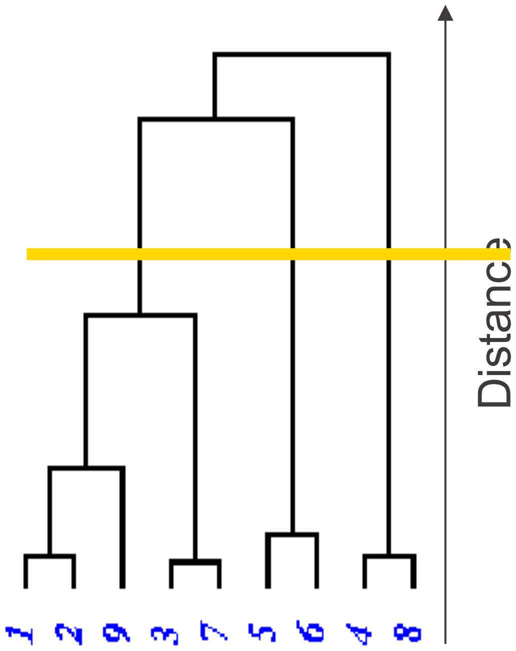
- Merge Step:
  - union of two subsets of data points
  - construct the mean point of the two clusters

Comments on Single Linkage and Variants

+ Finds not only a „flat“ clustering, but a hierarchy of clusters (dendrogram)

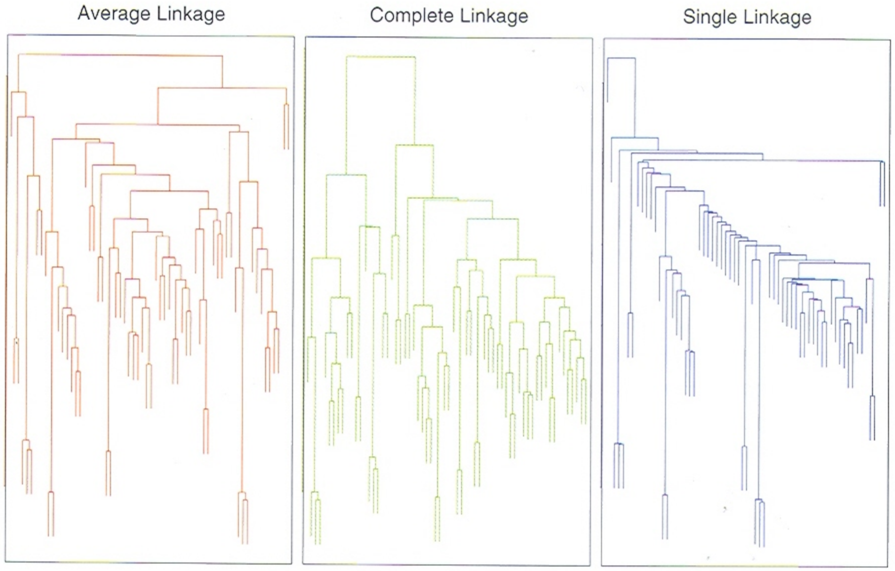
+ A single clustering can be obtained from the dendrogram (e.g., by performing a horizontal cut)

- Decisions (merges/splits) cannot be undone
- Sensitive to noise (Single-Link) (a „line“ of objects can connect two clusters)
- Inefficient
  → Runtime complexity at least $O(n^2)$ for n objects



Linkage Based Clustering

- Single Linkage:
  - Prefers well-separated clusters
- Complete Linkage:
  - Prefers small, compact clusters
- Average Linkage:
  - Prefers small, well-separated clusters…



Clustering: Density
DBSCAN

Clustering: DBSCAN

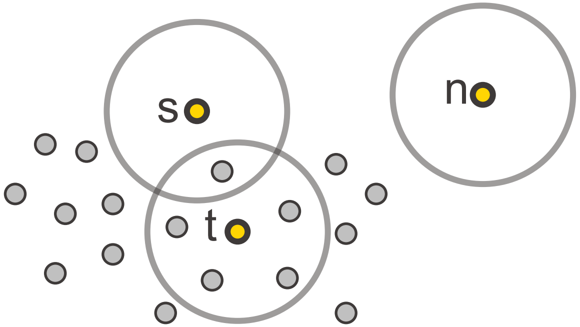
DBSCAN - a density-based clustering algorithm - defines five types of points in a dataset.

- **Core Points** are points that have at least a minimum number of neighbors (**MinPts**) within a specified distance (ε).
- **Noise Points** are neither core points nor border points.
- **Border Points** are points that are within ε of a core point, but have less than **MinPts** neighbors.
- **Directly Density Reachable Points** are within ε of a core point.
- **Density Reachable Points** are reachable with a chain of Directly Density Reachable points.

Clusters are built by joining core and density-reachable points to one another.

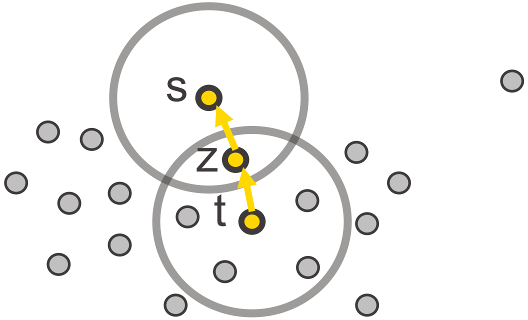
Example with MinPts = 3

Core Point
vs. Border Point
vs. Noise



- t = Core point
- s = Boarder point
- n = Noise point

Directly Density Reachable
vs. Density Reachable

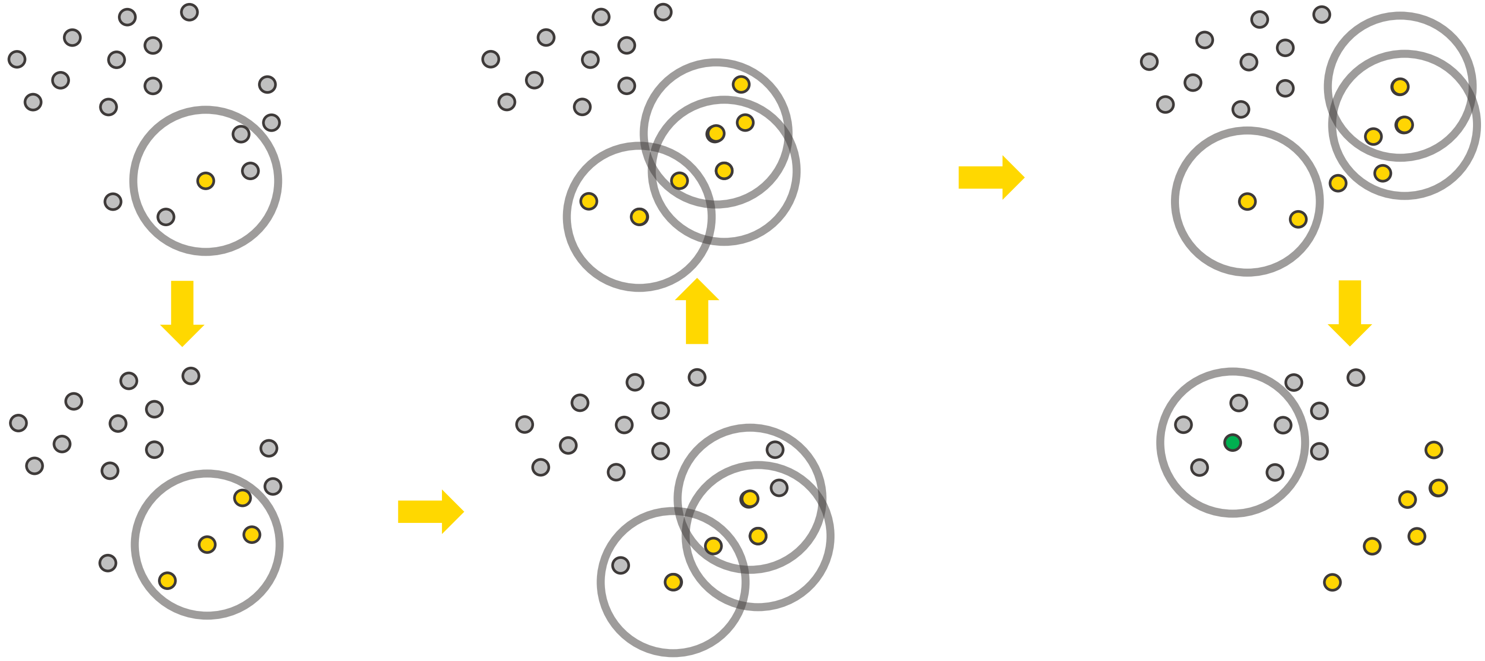


- z is directly density reachable from t
- s is not directly density reachable from t, but density reachable via z

Note: But t is not density reachable from s, because s is not a Core point

DBSCAN [Density Based Spatial Clustering of Applications with Noise]

- For each point, DBSCAN determines the ε-environment and checks whether it contains more than $MinPts$ data points → **core** point
- Iteratively increases the cluster by adding density-reachable points



Summary: DBSCAN

Clustering:

- A density-based clustering C of a dataset D w.r.t. ε and MinPts is the set of all density-based clusters C_i w.r.t. ε and MinPts in D.
- The set $NoiseCL$ („noise“) is defined as the set of all objects in D which do not belong to any of the clusters.

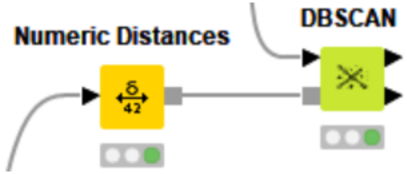
Property:

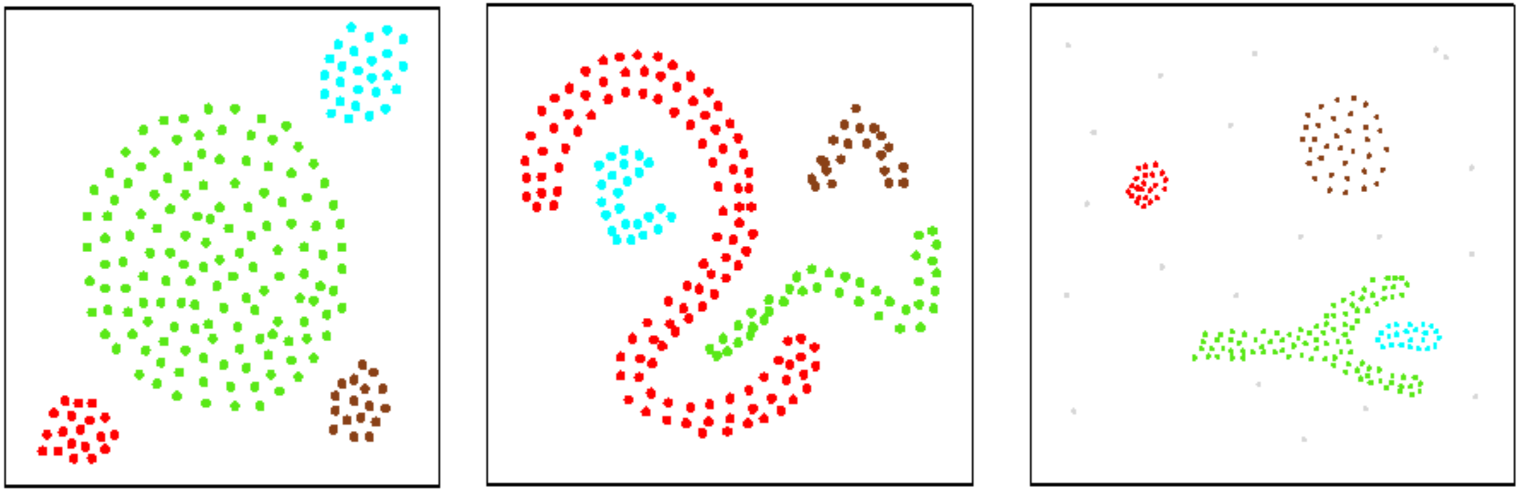
- Let C_i be a density-based cluster and $p \in C_i$ be a core object.

$$C_i = \{o \in D \mid o \text{ density-reachable from } p \text{ w.r.t. } \varepsilon \text{ and } MinPts\}.$$

DBSCAN [Density Based Spatial Clustering of Applications with Noise]

- DBSCAN uses (spatial) index structures for determining the ε-environment:
  → computational complexity $O(n \log n)$ instead of $O(n^2)$
- Arbitrary shape clusters found by DBSCAN
- Parameters: ε and $MinPts$





Data Preparation

Motivation

- **Real world data is „dirty“**
  - → Contains missing values, noises, outliers, inconsistencies
- **Comes from different information sources**
  - → Different attribute names, values expressed differently, related tuples
- **Different value ranges and hierarchies**
  - → One attribute range may overpower another
- **Huge amount of data**
  - → Makes analyis difficult and time consuming

Data Preparation

- Data Cleaning & Standardization (domain dependent)
- Aggregations (often domain dependent)
- Normalization
- Dimensionality Reduction
- Outlier Detection
- Missing Value Imputation
- Feature Selection
- Feature Engineering
- Sampling
- Integration of multiple Data Sources

Data Preparation: Normalization

Normalization: Motivation

Example:

- Lengths in cm (100 – 200) and weights in kilogram (30 – 150) fall both in approximately the same scale
- What about lengths in m (1-2) and weights also in gram (30000 – 150000)?
→ The weight values in mg dominate over the length values for the similarity of records!

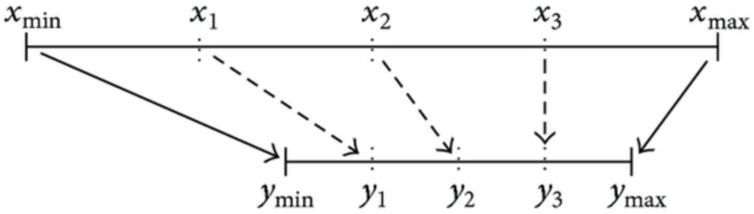
Goal of normalization:

- Transformation of attributes to make record ranges comparable

Normalization: Techniques

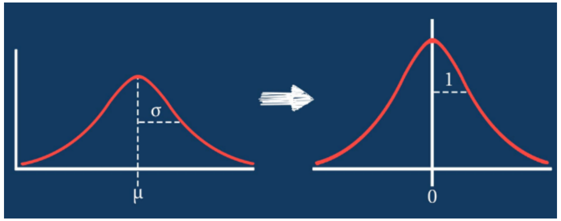
- **min-max normalization**

$$y = \frac{x - x_{min}}{x_{max} - x_{min}} (y_{max} - y_{min}) + y_{min}$$



- **z-score normalization**

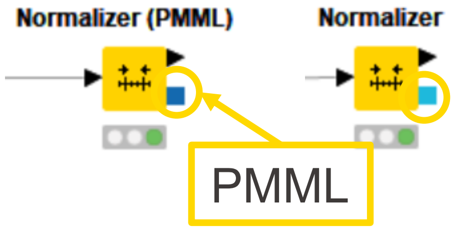
$$y = \frac{x - mean(x)}{stddev(x)}$$



- **normalization by decimal scaling**

$$y = \frac{x}{10^j}$$ where j is the smallest integer for $\max(y) < 1$

Here $[ymin, ymax]$ is $[0,1]$



PMML

- **Predictive Model Mark-up Language** (**PMML**) standard XML-based interchange format for predictive models.
- **Interchange.** PMML provides a way to describe and exchange predictive models produced by machine learning algorithms
- **Standard**. In theory, a PMML model exported from KNIME can be read by PMML compatible functions in other tools
- It does not work that well for the modern / ensemble algorithms, such as random forest or deep learning. In this case, other formats have been experimented.

Data Preparation: Missing Value Imputation

Missing Value Imputation: Motivation

Data is not always available

- E.g., many tuples have no recorded value for several attributes, such as weight in a people database

Missing data may be due to

- Equipment malfunctioning
- Inconsistency with other recorded data and thus deleted
- Data not entered (manually)
- Data not considered important at the time of collection
- Data format / contents of database changes

Missing Values: Types

Types of missing values:

Example: Suppose you are modeling weight Y as a function of sex X

- **Missing Completely At Random** (MCAR): reason does not depend on its value or lack of value.
  There may be no particular reason why some people told you their weights and others didn't.
- **Missing At Random** (MAR): the probability that Y is missing depends only on the value of X.
  One sex X may be less likely to disclose its weight Y.
- **Not Missing At Random** (NMAR): the probability that Y is missing depends on the unobserved value of Y itself.
  Heavy (or light) people may be less likely to disclose their weight.

Missing Values Imputation

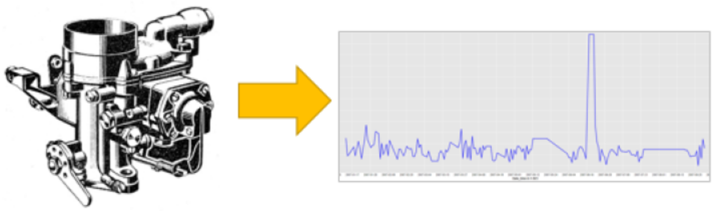
How to handle missing values?

- Ignore the record
- Remove the record
- Fill in missing value as:
  - Fixed value: e.g., “unknown”, -9999, etc.
  - Attribute mean / median / max. / min.
  - Attribute most frequent value
  - Next / previous /avg interpolation / moving avg value (in time series)
  - A predicted value based on the other attributes (inference-based such as Bayesian, Decision Tree, ...)

Data Preparation: Outlier Detection

Outlier Detection

- An outlier could be, for example, rare behavior, system defect, measurement error, or reaction to an unexpected event



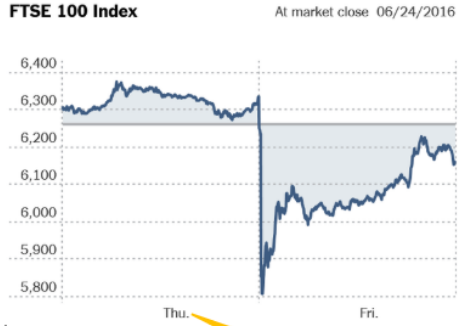
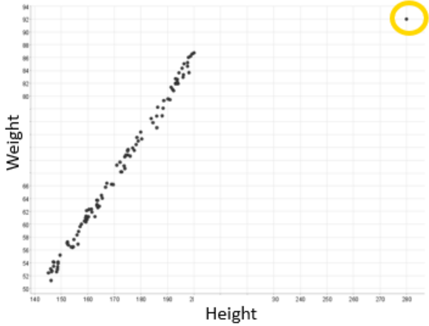


Image: https://www.nytimes.com/2016/06/25/business/international/brexit-financial-economic-impact-leave.html

Brexit referendum

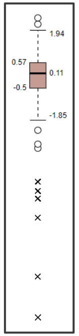


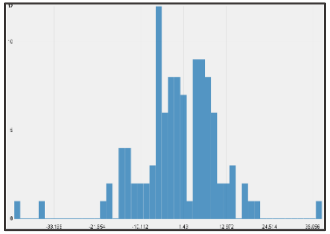
Outlier Detection: Motivation

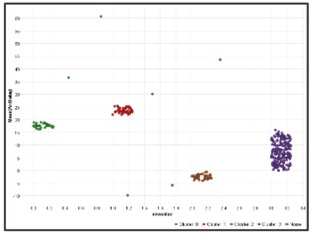
- Why finding outliers is important?
  - Summarize data by statistics that represent the majority of the data
  - Train a model that generalizes to new data
  - Finding the outliers can also be the focus of the analysis and not only data cleaning

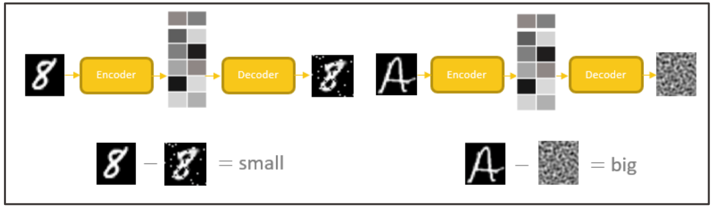
Outlier Detection Techniques

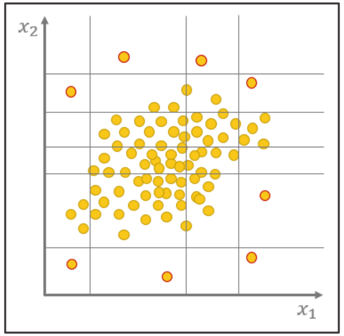
- Knowledge-based
- Statistics-based
  - Distance from the median
  - Position in the distribution tails
  - Distance to the closest cluster center
  - Error produced by an autoencoder
  - Number of random splits to isolate a data point from other data



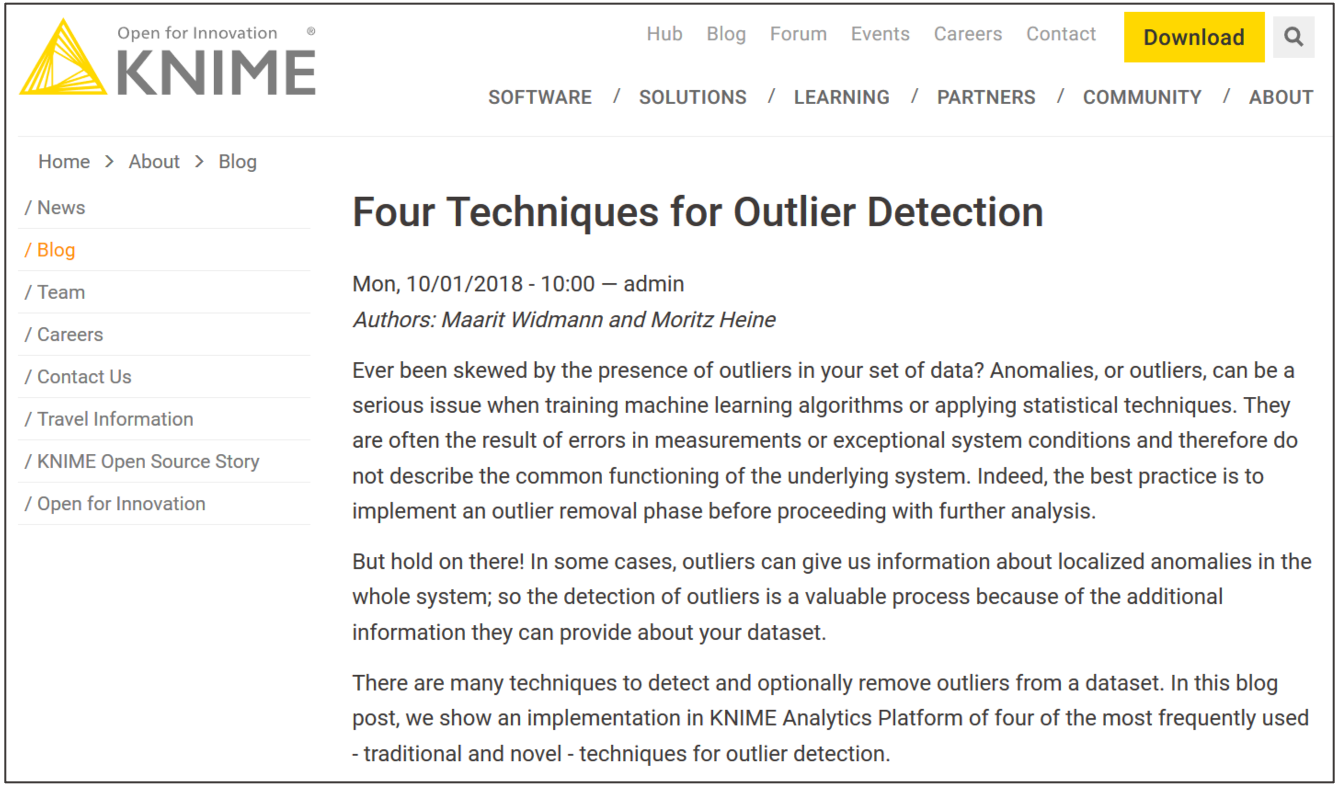








Material



Four Techniques for Outlier Detection

Mon, 10/01/2018 - 10:00 — admin

Authors: Maarit Widmann and Moritz Heine

Ever been skewed by the presence of outliers in your set of data? Anomalies, or outliers, can be a serious issue when training machine learning algorithms or applying statistical techniques. They are often the result of errors in measurements or exceptional system conditions and therefore do not describe the common functioning of the underlying system. Indeed, the best practice is to implement an outlier removal phase before proceeding with further analysis.

But hold on there! In some cases, outliers can give us information about localized anomalies in the whole system; so the detection of outliers is a valuable process because of the additional information they can provide about your dataset.

There are many techniques to detect and optionally remove outliers from a dataset. In this blog post, we show an implementation in KNIME Analytics Platform of four of the most frequently used - traditional and novel - techniques for outlier detection.

https://www.knime.com/blog/four-techniques-for-outlier-detection

Data Preparation:
Dimensionality Reduction

Is there such a thing as "too much data"?

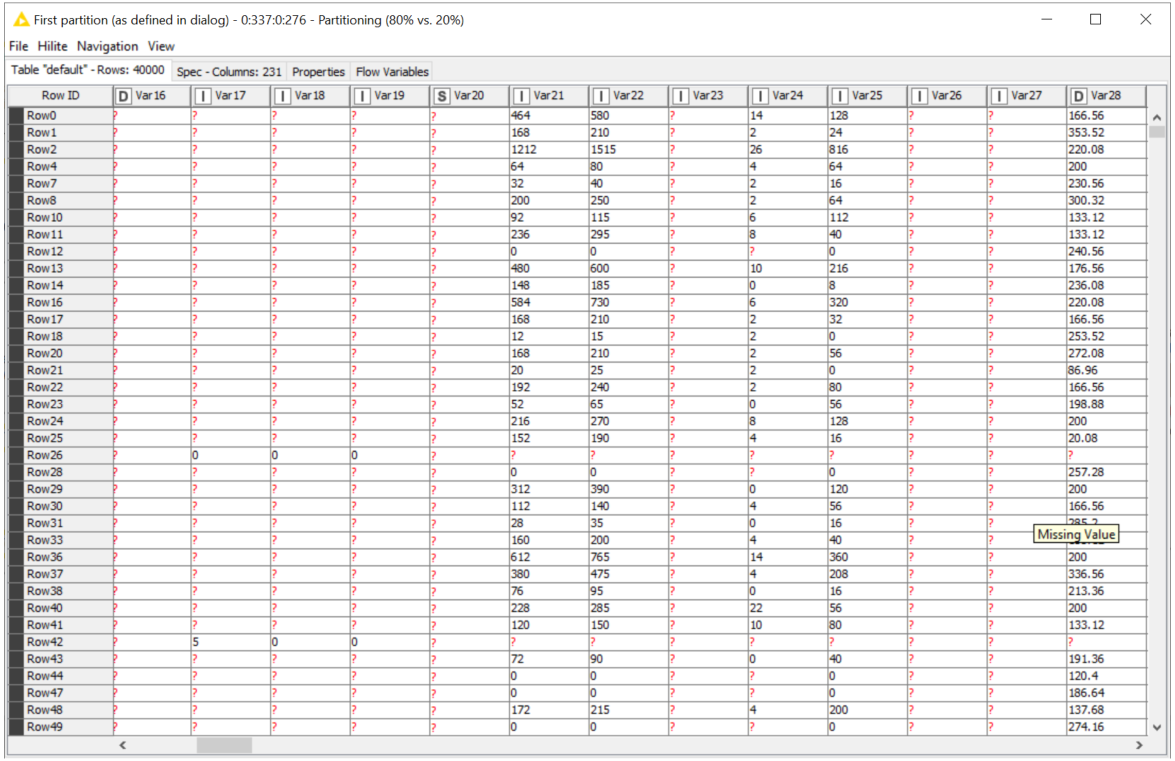
"Too much data":

- Consumes storage space
- Eats up processing time
- Is difficult to visualize
- Inhibits ML algorithm performance
- Beware of the model: Garbage in → Garbage out

Dimensionality Reduction Techniques

- Measure based
  - Ratio of missing values
  - Low variance
  - High Correlation
- Transformation based
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)
  - t-SNE
- Machine Learning based
  - Random Forest of shallow trees
  - Neural auto-encoder

Missing Values Ratio



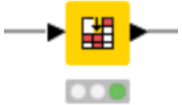
First partition (as defined in dialog) - 0:337:0:276 - Partitioning (80% vs. 20%)

Table "default" - Rows: 40000 | Spec - Columns: 231 | Properties | Flow Variables

Row ID	D Var16	I Var17	I Var18	I Var19	S Var20	I Var21	I Var22	I Var23	I Var24	I Var25	I Var26	I Var27	D Var28
Row0	?	?	?	?	?	464	580	?	14	128	?	?	166.56
Row1	?	?	?	?	?	168	210	?	2	24	?	?	353.52
Row2	?	?	?	?	?	1212	1515	?	26	816	?	?	220.08
Row4	?	?	?	?	?	64	80	?	4	64	?	?	200
Row7	?	?	?	?	?	32	40	?	2	16	?	?	230.56
Row8	?	?	?	?	?	200	250	?	2	64	?	?	300.32
Row10	?	?	?	?	?	92	115	?	6	112	?	?	133.12
Row11	?	?	?	?	?	236	295	?	8	40	?	?	133.12
Row12	?	?	?	?	?	0	0	?	?	0	?	?	240.56
Row13	?	?	?	?	?	480	600	?	10	216	?	?	176.56
Row14	?	?	?	?	?	148	185	?	0	8	?	?	236.08
Row16	?	?	?	?	?	584	730	?	6	320	?	?	220.08
Row17	?	?	?	?	?	168	210	?	2	32	?	?	166.56
Row18	?	?	?	?	?	12	15	?	2	0	?	?	253.52
Row20	?	?	?	?	?	168	210	?	2	56	?	?	272.08
Row21	?	?	?	?	?	20	25	?	2	0	?	?	86.96
Row22	?	?	?	?	?	192	240	?	2	80	?	?	166.56
Row23	?	?	?	?	?	52	65	?	0	56	?	?	198.88
Row24	?	?	?	?	?	216	270	?	8	128	?	?	200
Row25	?	?	?	?	?	152	190	?	4	16	?	?	20.08
Row26	?	0	0	0	?	?	?	?	?	?	?	?	?
Row28	?	?	?	?	?	0	0	?	?	0	?	?	257.28
Row29	?	?	?	?	?	312	390	?	0	120	?	?	200
Row30	?	?	?	?	?	112	140	?	4	56	?	?	166.56
Row31	?	?	?	?	?	28	35	?	0	16	?	?	285.2
Row33	?	?	?	?	?	160	200	?	4	40	?	?	[illegible]
Row36	?	?	?	?	?	612	765	?	14	360	?	?	200
Row37	?	?	?	?	?	380	475	?	4	208	?	?	336.56
Row38	?	?	?	?	?	76	95	?	0	16	?	?	213.36
Row40	?	?	?	?	?	228	285	?	22	56	?	?	200
Row41	?	?	?	?	?	120	150	?	10	80	?	?	133.12
Row42	?	5	0	0	?	?	?	?	?	?	?	?	?
Row43	?	?	?	?	?	72	90	?	0	40	?	?	191.36
Row44	?	?	?	?	?	0	0	?	?	0	?	?	120.4
Row47	?	?	?	?	?	0	0	?	?	0	?	?	186.64
Row48	?	?	?	?	?	172	215	?	4	200	?	?	137.68
Row49	?	?	?	?	?	0	0	?	?	0	?	?	274.16

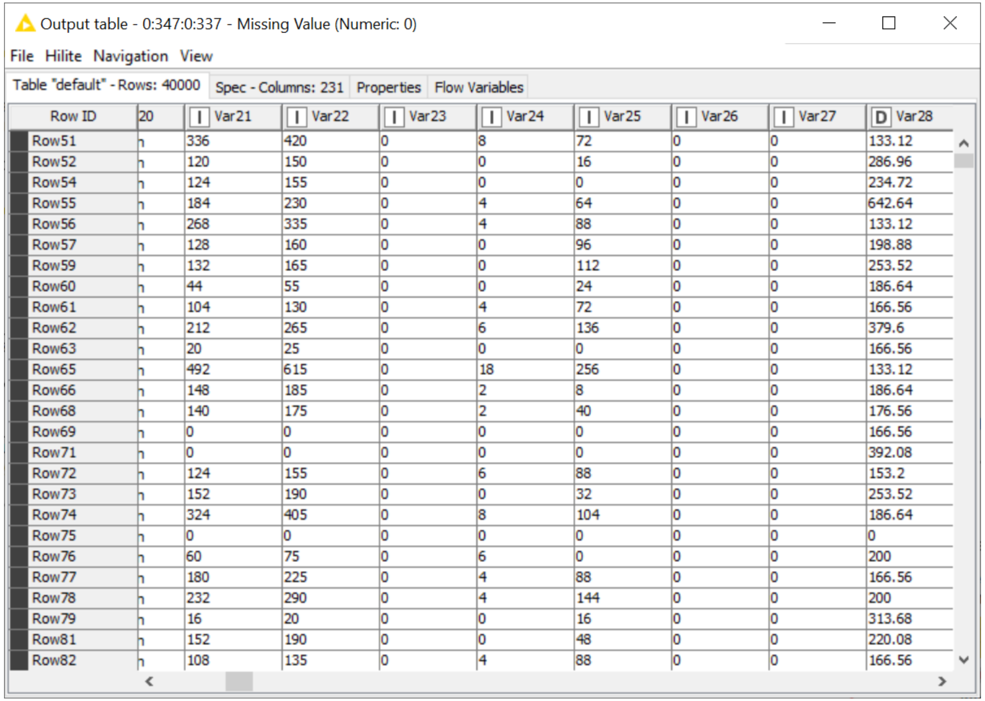
Missing Value

Missing Value Column Filter



IF (% missing value > threshold) THEN remove column

Low Variance



Output table - 0:347:0:337 - Missing Value (Numeric: 0)

Table "default" - Rows: 40000 | Spec - Columns: 231 | Properties | Flow Variables

Row ID	20	I Var21	I Var22	I Var23	I Var24	I Var25	I Var26	I Var27	D Var28
Row51	h	336	420	0	8	72	0	0	133.12
Row52	h	120	150	0	0	16	0	0	286.96
Row54	h	124	155	0	0	0	0	0	234.72
Row55	h	184	230	0	4	64	0	0	642.64
Row56	h	268	335	0	4	88	0	0	133.12
Row57	h	128	160	0	0	96	0	0	198.88
Row59	h	132	165	0	0	112	0	0	253.52
Row60	h	44	55	0	0	24	0	0	186.64
Row61	h	104	130	0	4	72	0	0	166.56
Row62	h	212	265	0	6	136	0	0	379.6
Row63	h	20	25	0	0	0	0	0	166.56
Row65	h	492	615	0	18	256	0	0	133.12
Row66	h	148	185	0	2	8	0	0	186.64
Row68	h	140	175	0	2	40	0	0	176.56
Row69	h	0	0	0	0	0	0	0	166.56
Row71	h	0	0	0	0	0	0	0	392.08
Row72	h	124	155	0	6	88	0	0	153.2
Row73	h	152	190	0	0	32	0	0	253.52
Row74	h	324	405	0	8	104	0	0	186.64
Row75	h	0	0	0	0	0	0	0	0
Row76	h	60	75	0	6	0	0	0	200
Row77	h	180	225	0	4	88	0	0	166.56
Row78	h	232	290	0	4	144	0	0	200
Row79	h	16	20	0	0	16	0	0	313.68
Row81	h	152	190	0	0	48	0	0	220.08
Row82	h	108	135	0	4	88	0	0	166.56

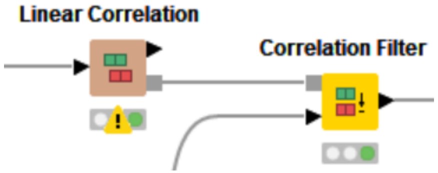
Low Variance Filter

Note: requires min-max-normalization, and only works for numeric columns

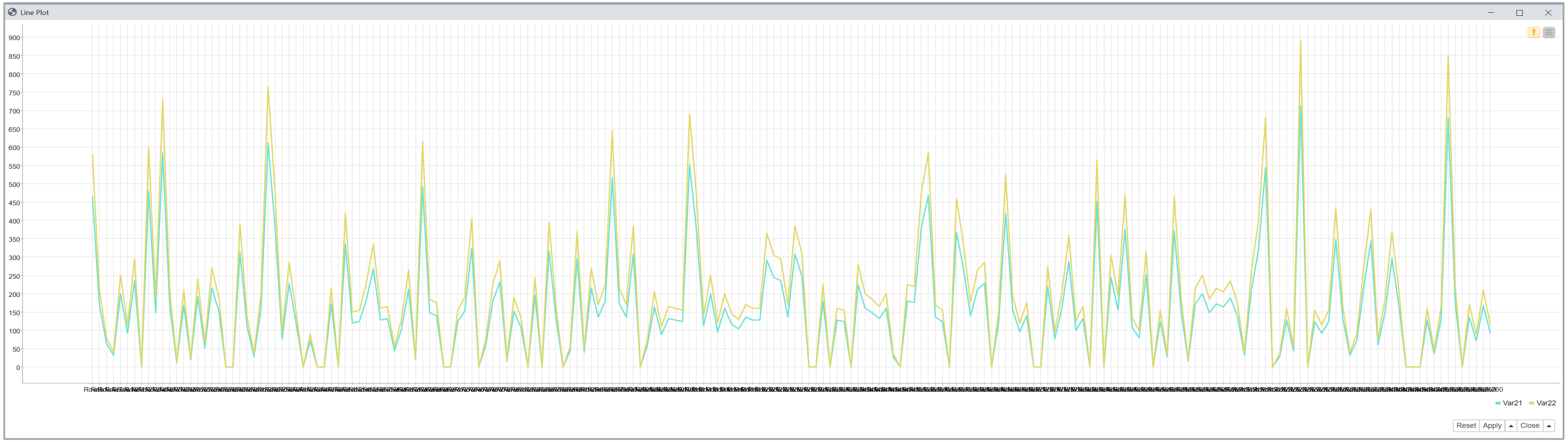
- If column has **constant** value (variance = **0**), it contains no useful information
- In general: IF (variance < threshold) THEN remove column

High Correlation

- Two **highly correlated** input variables probably carry similar information
- IF (**corr(var1, var2)** > threshold) => remove var1



Note: requires min-max-normalization of numeric columns



Principal Component Analysis (PCA)

- PCA is a statistical procedure that **orthogonally** transforms the original n coordinates of a data set into a new set of n coordinates, called principal components.

$$(PC_1, PC_2, \cdots PC_n) = PCA(X_1, X_2, \cdots X_n)$$

- The first principal component PC_1 follows the direction (eigenvector) of the **largest possible variance** (largest eigenvalue of the covariance matrix) in the data.

- Each succeeding component PC_k follows the direction of the **next largest possible variance** under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components $(PC_1, PC_2, \cdots PC_{k-1})$.

If you're still curious, there's LOTS of different ways to think about PCA: https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

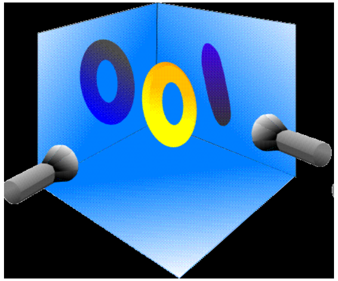
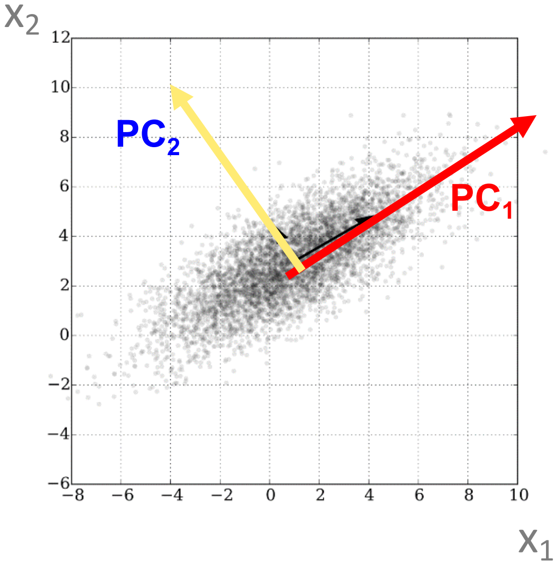


Image from Wikipedia



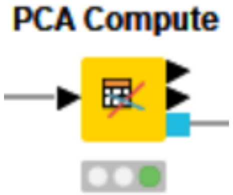
Principal Component Analysis (PCA)

- PC_1 describes most of the variability in the data, PC_2 adds the next big contribution, and so on. In the end, the last PCs do not bring much more information to describe the data.

- Thus, to describe the data we could use only the top $m < n$ (i.e., $PC_1, PC_2, \cdots PC_m$) components with little - if any - loss of information

Dimensionality Reduction

- Caveats:
  - Results of PCA are quite difficult to interpret
  - Normalization required
  - Only effective on numeric columns



Linear Discriminant Analysis (LDA)

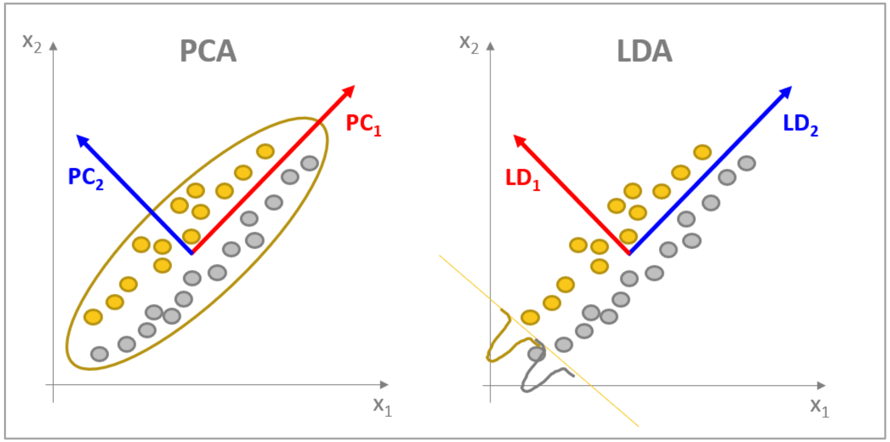
- LDA is a statistical procedure that **orthogonally** transforms the original n coordinates of a data set into a new set of n coordinates, called linear discriminants.

$$(LD_1, LD_2, \cdots LD_n) = LDA(X_1, X_2, \cdots X_n)$$

Linear Discriminant Analysis Compute

- Here, however, discriminants (components) **maximize the separation between classes**

- PCA : unsupervised
- LDA : supervised



Linear Discriminant Analysis (LDA)

- LD_1 describes best the class separation in the data, LD_2 adds the next big contribution, and so on. In the end, the last LDs do not bring much more information to separate the classes.

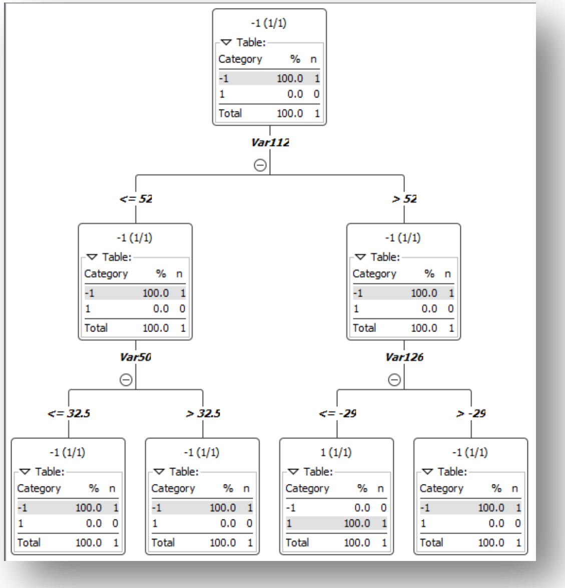
- Thus, for our classification problem we could use only the top $m < n$ (i.e., $LD_1, LD_2, \cdots LD_m$) discriminants with little - if any - loss of information

Dimensionality Reduction

- Caveats:
  - Results of LDA are quite difficult to interpret
  - Normalization required
  - Only effective on numeric columns

Ensembles of Shallow Decision Trees

- Often used for classification, but can be used for feature selection too
- Generate a large number (we used 2000) of trees that are very shallow (2 levels, 3 sampled features)
- Calculate the statistics of candidates and selected features. The more often a feature is selected in such trees, the more likely it contains predictive information
- Compare the same statistics with a forest of trees trained on a random dataset.



-1 (1/1) (root)

Category	%	n
-1	100.0	1
1	0.0	0
Total	100.0	1

-1 (1/1) (<= 52)

Category	%	n
-1	100.0	1
1	0.0	0
Total	100.0	1

-1 (1/1) (> 52)

Category	%	n
-1	100.0	1
1	0.0	0
Total	100.0	1

-1 (1/1) (<= 32.5)

Category	%	n
-1	100.0	1
1	0.0	0
Total	100.0	1

-1 (1/1) (> 32.5)

Category	%	n
-1	100.0	1
1	0.0	0
Total	100.0	1

1 (1/1) (<= -29)

Category	%	n
-1	0.0	0
1	100.0	1
Total	100.0	1

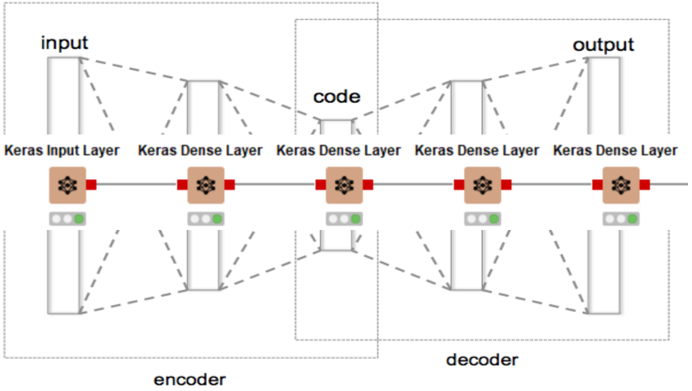
-1 (1/1) (> -29)

Category	%	n
-1	100.0	1
1	0.0	0
Total	100.0	1

Autoencoder

- Feed-Forward Neural Network architecture with encoder / decoder structure. The network is trained to reproduce the input vector onto the output layer.

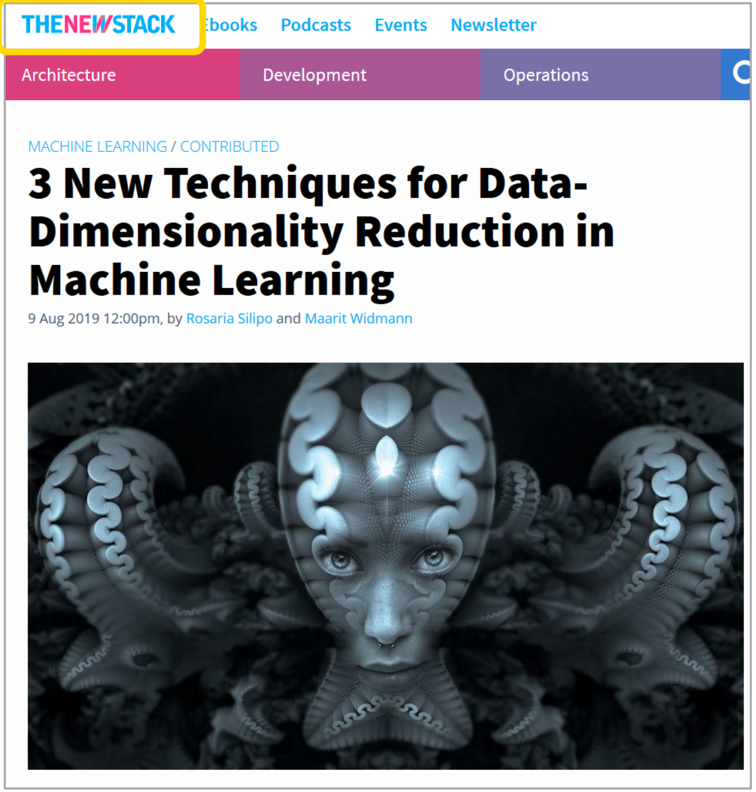
Image: Wikipedia



- That is, it compresses the input vector (dimension n) into a smaller vector space on layer “code” (dimension m<n) and then it reconstructs the original vector onto the output layer.

- If the network was trained well, the reconstruction operation happens with minimal loss of information.

Material



MACHINE LEARNING / CONTRIBUTED

3 New Techniques for Data-Dimensionality Reduction in Machine Learning

9 Aug 2019 12:00pm, by Rosaria Silipo and Maarit Widmann

https://thenewstack.io/3-new-techniques-for-data-dimensionality-reduction-in-machine-learning/

Data Preparation:
Feature Selection

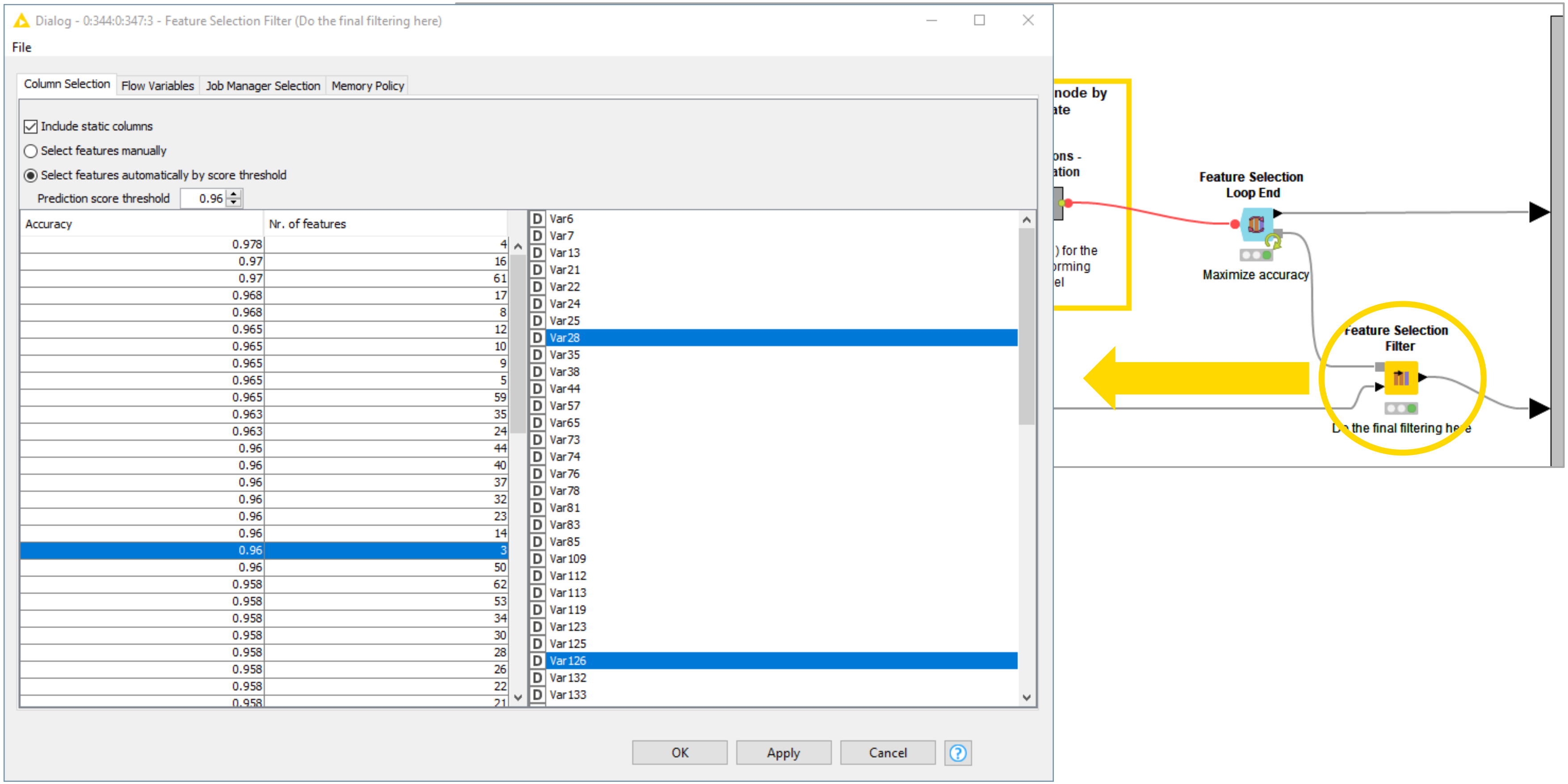
Feature Selection vs. Dimensionality Reduction

- Both methods are used for reducing the number of features in a dataset. However:
- Feature selection is simply selecting and excluding given features **without changing** them.
- Dimensionality reduction **might transform** the features into a lower dimension.
- Feature selection is often a somewhat more aggressive and more computationally expensive process.
  - Backward Feature Elimination
  - Forward Feature Construction

Backward Feature Elimination (greedy top-down)

1. First train one model on n input features
2. Then train n separate models each on $n - 1$ input features and remove the feature whose removal produced the least disturbance
3. Then train $n - 1$ separate models each on $n - 2$ input features and remove the feature whose removal produced the least disturbance
4. And so on. Continue until desired maximum error rate on *training* data is reached.

Backward Feature Elimination



Dialog - 0:344:0:347:3 - Feature Selection Filter (Do the final filtering here)

File

Column Selection | Flow Variables | Job Manager Selection | Memory Policy

☑ Include static columns

○ Select features manually

● Select features automatically by score threshold

Prediction score threshold 0.96

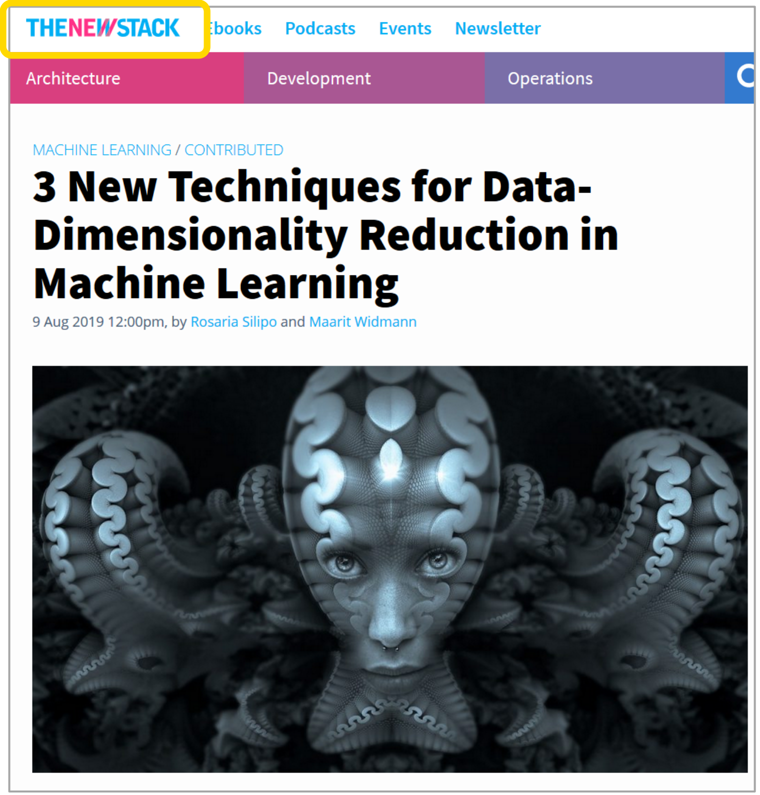
Accuracy	Nr. of features
0.978	4
0.97	16
0.97	61
0.968	17
0.968	8
0.965	12
0.965	10
0.965	9
0.965	5
0.965	59
0.963	35
0.963	24
0.96	44
0.96	40
0.96	37
0.96	32
0.96	23
0.96	14
0.96	3
0.96	50
0.958	62
0.958	53
0.958	34
0.958	30
0.958	28
0.958	26
0.958	22
0.958	21

D Var6
D Var7
D Var13
D Var21
D Var22
D Var24
D Var25
D Var28
D Var35
D Var38
D Var44
D Var57
D Var65
D Var73
D Var74
D Var76
D Var78
D Var81
D Var83
D Var85
D Var109
D Var112
D Var113
D Var119
D Var123
D Var125
D Var126
D Var132
D Var133

Forward Feature Construction (greedy bottom-up)

1. First, train n separate models on one single input feature and keep the feature that produces the best accuracy.
2. Then, train $n-1$ separate models on 2 input features, the selected one and one more. At the end keep the additional feature that produces the best accuracy.
3. And so on … Continue until an acceptable error rate is reached.

Material



MACHINE LEARNING / CONTRIBUTED

3 New Techniques for Data-Dimensionality Reduction in Machine Learning

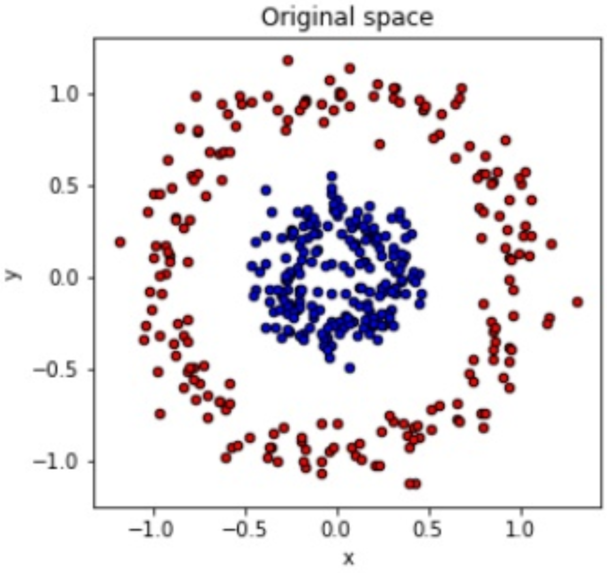
9 Aug 2019 12:00pm, by Rosaria Silipo and Maarit Widmann

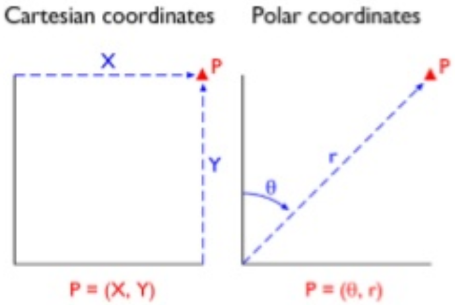
https://thenewstack.io/3-new-techniques-for-data-dimensionality-reduction-in-machine-learning/

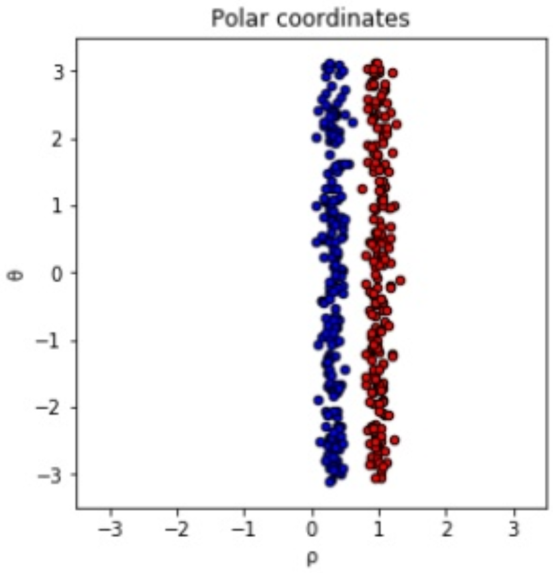
Data Preparation: Feature Engineering

Feature Engineering: Motivation

Sometimes transforming the original data allows for better discrimination by ML algorithms.





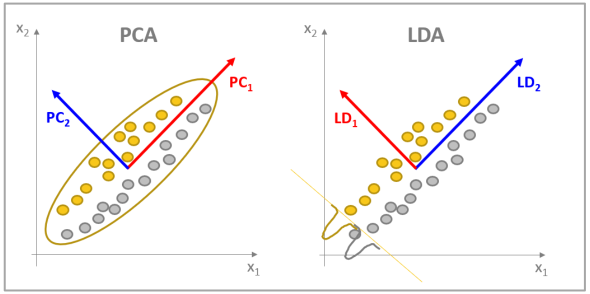


Feature Engineering: Techniques

- Coordinate Transformations

  Remember PCA and LDA?

  Polar coordinates , …



- Distances to cluster centres, after data clustering
- Simple math transformations on single columns

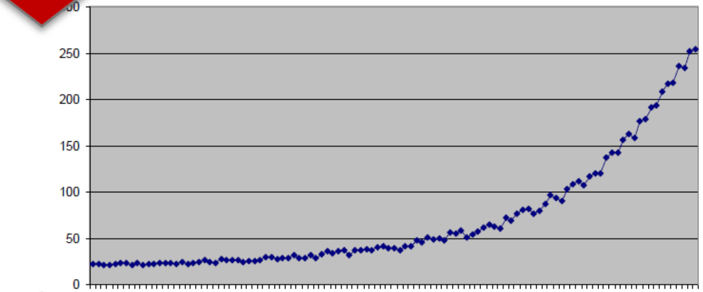
  $(e^x, x^2, x^3, \tanh(x), \log(x)\ , \ldots)$
- Combining together multiple columns in math functions

  $(f(x_1, x_2, \ldots\ xn), x_2 - x_1, \ldots)$
- The whole process is domain dependent

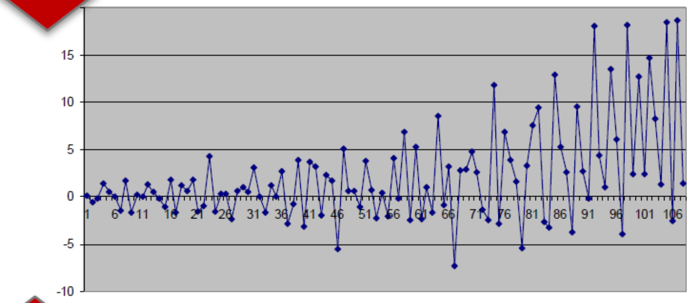
Feature Engineering in Time Series Analysis

- Second order differences: $y = x(t) – x(t-1)$ & $y‘(t) = y(t) – y(t-1)$
- Logarithm: $\log(y‘(t))$

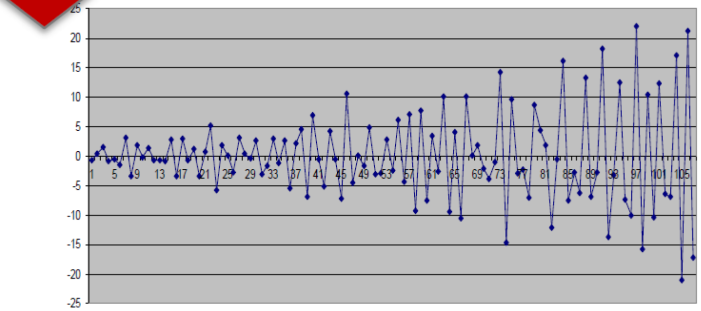
1


1 - Original Time Series: non-stationary (mean and variance)

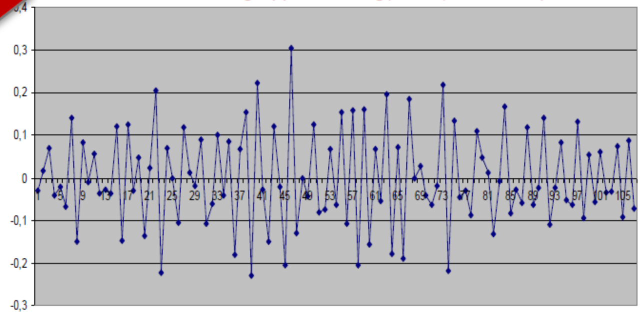
2


2 – First Order Differencing: non-stationary (mean and variance)

3


3 – Second Order Diff.: stationary in mean, but not in variance

4


4 – Double Differencing applied to Log(Series): stationary series

Confirmation of Attendance and Survey

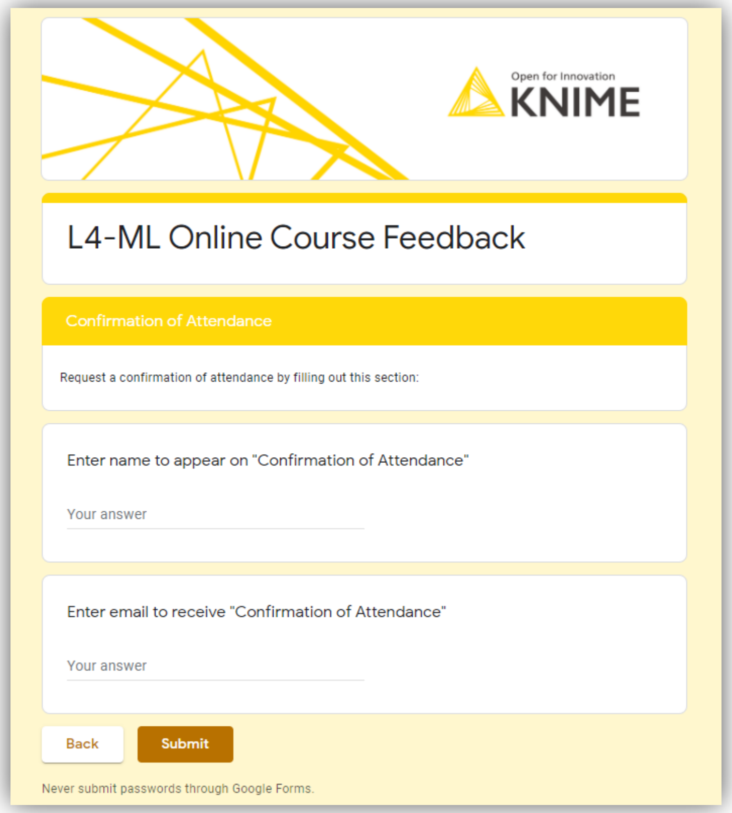
- If you would like to get a “Confirmation of Attendance” please click on the link below*

  Confirmation of Attendance and Survey

- The link also takes you to our course feedback survey. Filling it in is optional but highly appreciated!

Thank you!

*Please send your request within the next 3 days



L4-ML Online Course Feedback

Confirmation of Attendance

Request a confirmation of attendance by filling out this section:

Enter name to appear on "Confirmation of Attendance"

Your answer

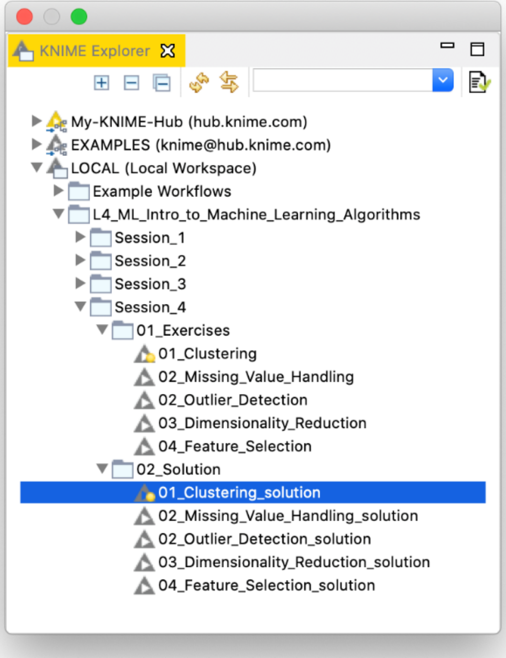
Enter email to receive "Confirmation of Attendance"

Your answer

Never submit passwords through Google Forms.

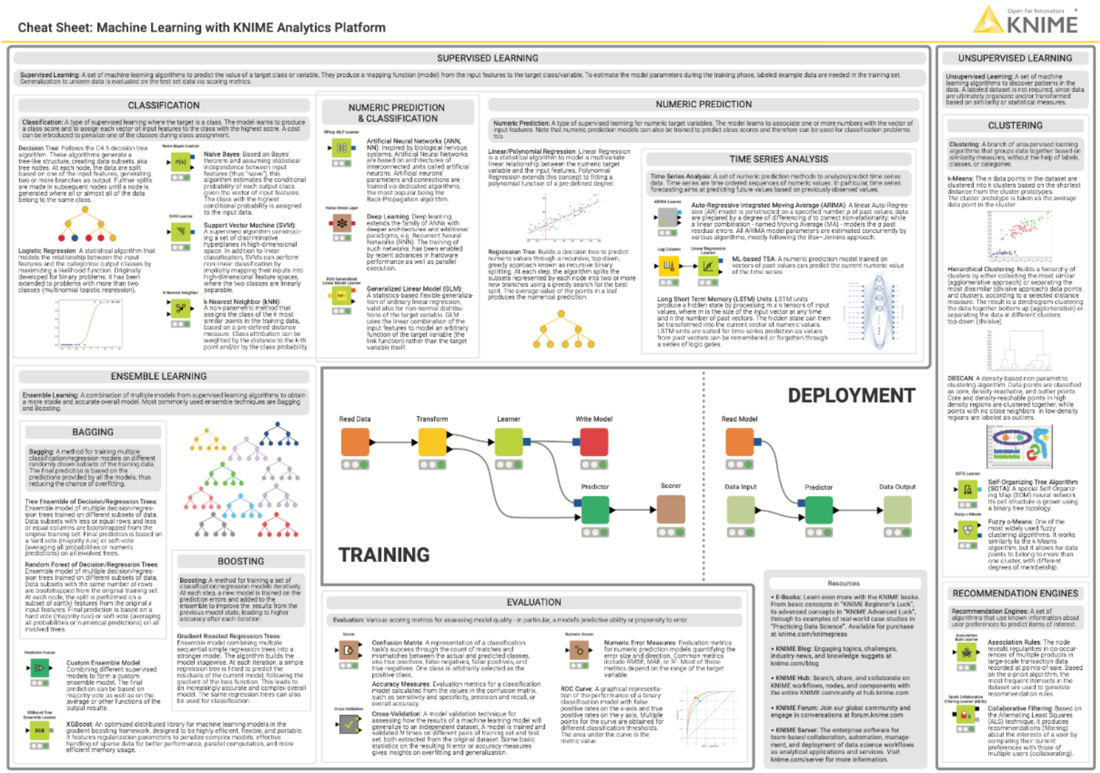
Exercises

- Clustering
  - Goal: Cluster location data from California
  - 01_Clustering
- Data Preparation
  - 02_Missing_Value_Handling
  - 03_Outlier_Detection
  - 04_Dimensionality_Reduction
  - 05_Feature_Selection



```
My-KNIME-Hub (hub.knime.com)
EXAMPLES (knime@hub.knime.com)
LOCAL (Local Workspace)
  Example Workflows
  L4_ML_Intro_to_Machine_Learning_Algorithms
    Session_1
    Session_2
    Session_3
    Session_4
      01_Exercises
        01_Clustering
        02_Missing_Value_Handling
        02_Outlier_Detection
        03_Dimensionality_Reduction
        04_Feature_Selection
      02_Solution
        01_Clustering_solution
        02_Missing_Value_Handling_solution
        02_Outlier_Detection_solution
        03_Dimensionality_Reduction_solution
        04_Feature_Selection_solution
```

Machine Learning Cheat Sheet



Cheat Sheet: Machine Learning with KNIME Analytics Platform

SUPERVISED LEARNING

Supervised Learning: A set of machine learning algorithms to predict the value of a target class or variable. They produce a mapping function (model) from the input features to the target class/variable. To estimate the model parameters during the training phase, labeled example data are needed in the training set. Generalization to unseen data is evaluated on the test set data via scoring metrics.

CLASSIFICATION

Classification: A type of supervised learning where the target is a class. The model learns to produce a class score and to assign each vector of input features to the class with the highest score. A cost can be introduced to penalize one of the classes during class assignment.

Decision Tree: Follows the C4.5 decision tree algorithm. These algorithms generate a tree-like structure, creating data subsets, aka tree nodes. At each node, the data are split based on one of the input features, generating two or more branches as output. Further splits are made in subsequent nodes until a node is generated where all or almost all of the data belong to the same class.

Logistic Regression: A statistical algorithm that models the relationship between the input features and the categorical output classes by maximizing a likelihood function. Originally developed for binary problems, it has been extended to problems with more than two classes (multinomial logistic regression).

Naive Bayes: Based on Bayes' theorem and assuming statistical independence between input features (thus "naive"), this algorithm estimates the conditional probability of each output class given the vector of input features. The class with the highest conditional probability is assigned to the input data.

Support Vector Machine (SVM): A supervised algorithm constructing a set of discriminative hyperplanes in high-dimensional space. In addition to linear classification, SVMs can perform non-linear classification by implicitly mapping their inputs into high-dimensional feature spaces, where the two classes are linearly separable.

k-Nearest Neighbor (kNN): A non-parametric method that assigns the class of the *k* most similar points in the training data, based on a pre-defined distance measure. Class attribution can be weighted by the distance to the *k*-th point and/or by the class probability.

NUMERIC PREDICTION & CLASSIFICATION

Artificial Neural Networks (ANN, NN): Inspired by biological nervous systems, Artificial Neural Networks are based on architectures of interconnected units called artificial neurons. Artificial neurons' parameters and connections are trained via dedicated algorithms, the most popular being the Back-Propagation algorithm.

Deep Learning: Deep learning extends the family of ANNs with deeper architectures and additional paradigms, e.g. Recurrent Neural Networks (RNN). The training of such networks, has been enabled by recent advances in hardware performance as well as parallel execution.

Generalized Linear Model (GLM): A statistics-based flexible generalization of ordinary linear regression, valid also for non-normal distributions of the target variable. GLM uses the linear combination of the input features to model an arbitrary function of the target variable (the link function) rather than the target variable itself.

NUMERIC PREDICTION

Numeric Prediction: A type of supervised learning for numeric target variables. The model learns to associate one or more numbers with the vector of input features. Note that numeric prediction models can also be trained to predict class scores and therefore can be used for classification problems too.

Linear/Polynomial Regression: Linear Regression is a statistical algorithm to model a multivariate linear relationship between the numeric target variable and the input features. Polynomial Regression extends this concept to fitting a polynomial function of a pre-defined degree.

Regression Tree: Builds a decision tree to predict numeric values through a recursive, top-down, greedy approach known as recursive binary splitting. At each step, the algorithm splits the subsets represented by each node into two or more new branches using a greedy search for the best split. The average value of the points in a leaf produces the numerical prediction.

TIME SERIES ANALYSIS

Time Series Analysis: A set of numeric prediction methods to analyze/predict time series data. Time series are time ordered sequences of numeric values. In particular, time series forecasting aims at predicting future values based on previously observed values.

Auto-Regressive Integrated Moving Average (ARIMA): A linear Auto-Regressive (AR) model is constructed on a specified number *p* of past values; data are prepared by a degree of differencing *d* to correct non-stationarity; while a linear combination - named Moving Average (MA) - models the *q* past residual errors. All ARIMA model parameters are estimated concurrently by various algorithms, mostly following the Box–Jenkins approach.

ML-based TSA: A numeric prediction model trained on vectors of past values can predict the current numeric value of the time series.

Long Short Term Memory (LSTM) Units: LSTM units produce a hidden state by processing *m x n* tensors of input values, where *m* is the size of the input vector at any time and *n* the number of past vectors. The hidden state can then be transformed into the current vector of numeric values. LSTM units are suited for time series prediction as values from past vectors can be remembered or forgotten through a series of logic gates.

ENSEMBLE LEARNING

Ensemble Learning: A combination of multiple models from supervised learning algorithms to obtain a more stable and accurate overall model. Most commonly used ensemble techniques are Bagging and Boosting.

BAGGING

Bagging: A method for training multiple classification/regression models on different randomly drawn subsets of the training data. The final prediction is based on the predictions provided by all the models, thus reducing the chance of overfitting.

Tree Ensemble of Decision/Regression Trees: Ensemble model of multiple decision/regression trees trained on different subsets of data. Data subsets with less or equal rows and less or equal columns are bootstrapped from the original training set. Final prediction is based on a hard vote (majority rule) or soft-vote (averaging all probabilities or numeric predictions) on all involved trees.

Random Forest of Decision/Regression Trees: Ensemble model of multiple decision/regression trees trained on different subsets of data. Data subsets with the same number of rows are bootstrapped from the original training set. At each node, the split is performed on a subset of *sqrt(x)* features from the original *x* input features. Final prediction is based on a hard vote (majority rule) or soft-vote (averaging all probabilities or numerical predictions) on all involved trees.

BOOSTING

Boosting: A method for training a set of classification/regression models iteratively. At each step, a new model is trained on the prediction errors and added to the ensemble to improve the results from the previous model state, leading to higher accuracy after each iteration.

Gradient Boosted Regression Trees: Ensemble model combining multiple sequential simple regression trees into a stronger model. The algorithm builds the model stagewise. At each iteration, a simple regression tree is fitted to predict the residuals of the current model, following the gradient of the loss function. This leads to an increasingly accurate and complex overall model. The same regression trees can also be used for classification.

Custom Ensemble Model: Combining different supervised models to form a custom ensemble model. The final prediction can be based on majority vote as well as on the average or other functions of the output results.

XGBoost: An optimized distributed library for machine learning models in the gradient boosting framework, designed to be highly efficient, flexible, and portable. It features regularization parameters to penalize complex models, effective handling of sparse data for better performance, parallel computation, and more efficient memory usage.

TRAINING

DEPLOYMENT

EVALUATION

Evaluation: Various scoring metrics for assessing model quality - in particular, a model's predictive ability or propensity to error.

Confusion Matrix: A representation of a classification task's success through the count of matches and mismatches between the actual and predicted classes, aka true positives, false negatives, false positives, and true negatives. One class is arbitrarily selected as the positive class.

Accuracy Measures: Evaluation metrics for a classification model calculated from the values in the confusion matrix, such as sensitivity and specificity, precision and recall, or overall accuracy.

Cross-Validation: A model validation technique for assessing how the results of a machine learning model will generalize to an independent dataset. A model is trained and validated *N* times on different pairs of training set and test set, both extracted from the original dataset. Some basic statistics on the resulting *N* error or accuracy measures gives insights on overfitting and generalization.

Numeric Error Measures: Evaluation metrics for numeric prediction models quantifying the error size and direction. Common metrics include RMSE, MAE, or R^2. Most of these metrics depend on the range of the target variable.

ROC Curve: A graphical representation of the performance of a binary classification model with false positive rates on the x-axis and true positive rates on the y axis. Multiple points for the curve are obtained for different classification thresholds. The area under the curve is the metric value.

Resources

- **E-Books:** Learn even more with the KNIME books. From basic concepts in "KNIME Beginner's Luck", to advanced concepts in "KNIME Advanced Luck", through to examples of real-world case studies in "Practicing Data Science". Available for purchase at knime.com/knimepress
- **KNIME Blog:** Engaging topics, challenges, industry news, and knowledge nuggets at knime.com/blog
- **KNIME Hub:** Search, share, and collaborate on KNIME workflows, nodes, and components with the entire KNIME community at hub.knime.com
- **KNIME Forum:** Join our global community and engage in conversations at forum.knime.com
- **KNIME Server:** The enterprise software for team-based collaboration, automation, management, and deployment of data science workflows as analytical applications and services. Visit knime.com/server for more information.

UNSUPERVISED LEARNING

Unsupervised Learning: A set of machine learning algorithms to discover patterns in the data. A labeled dataset is not required, since data are ultimately organized and/or transformed based on similarity or statistical measures.

CLUSTERING

Clustering: A branch of unsupervised learning algorithms that groups data together based on similarity measures, without the help of labels, classes, or categories.

k-Means: The *n* data points in the dataset are clustered into *k* clusters based on the shortest distance from the cluster prototypes. The cluster prototype is taken as the average data point in the cluster.

Hierarchical Clustering: Builds a hierarchy of clusters by either collecting the most similar (agglomerative approach) or separating the most dissimilar (divisive approach) data points and clusters, according to a selected distance measure. The result is a dendrogram clustering the data together bottom-up (agglomerative) or separating the data in different clusters top-down (divisive).

DBSCAN: A density-based non-parametric clustering algorithm. Data points are classified as core, density-reachable, and outlier points. Core and density-reachable points in high density regions are clustered together, while points with no close neighbors in low-density regions are labeled as outliers.

Self-Organizing Tree Algorithm (SOTA): A special Self-Organizing Map (SOM) neural network. Its cell structure is grown using a binary tree topology.

Fuzzy c-Means: One of the most widely used fuzzy clustering algorithms. It works similarly to the k-Means algorithm, but it allows for data points to belong to more than one cluster, with different degrees of membership.

RECOMMENDATION ENGINES

Recommendation Engines: A set of algorithms that use known information about user preferences to predict items of interest.

Association Rules: The node reveals regularities in co-occurrences of multiple products in large-scale transaction data recorded at points-of-sale. Based on the a-priori algorithm, the most frequent itemsets in the dataset are used to generate recommendation rules.

Collaborative Filtering: Based on the Alternating Least Squares (ALS) technique, it produces recommendations (filtering) about the interests of a user by comparing their current preferences with those of multiple users (collaborating).

https://www.knime.com/sites/default/files/110519_KNIME_Machine_Learning_Cheat%20Sheet.pdf

Thank You!