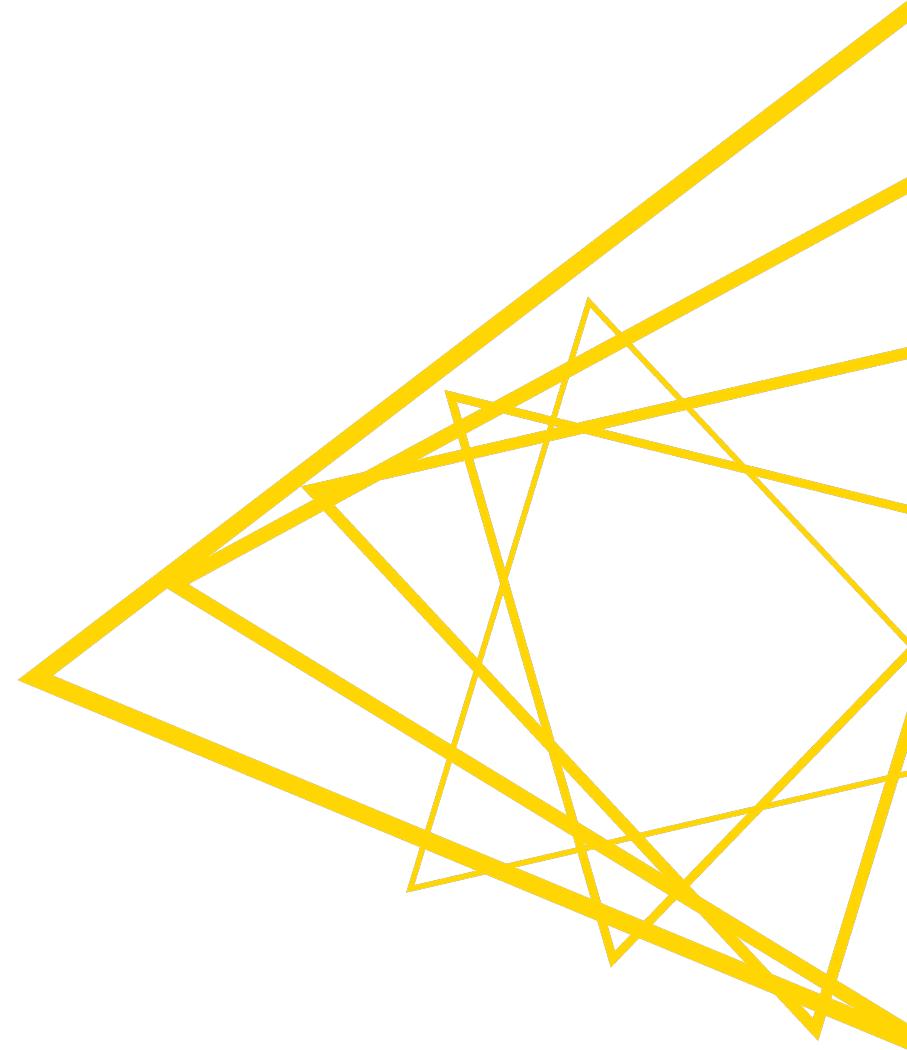




# [L4-ML] Introduction to Machine Learning Algorithms

KNIME AG



# Structure of the Course

---

Time	Session
09:00-10:30	Introduction & Decision Tree Algorithm
10:45-12:15	Regression Models, Ensemble Models, & Logistic Regression
12:15-13:15	Lunch
13:15-14:45	Neural Networks & Recommendation Engines
15:00-17:00	Clustering & Data Preparation
17:00-17:30	Q&A and Feedback

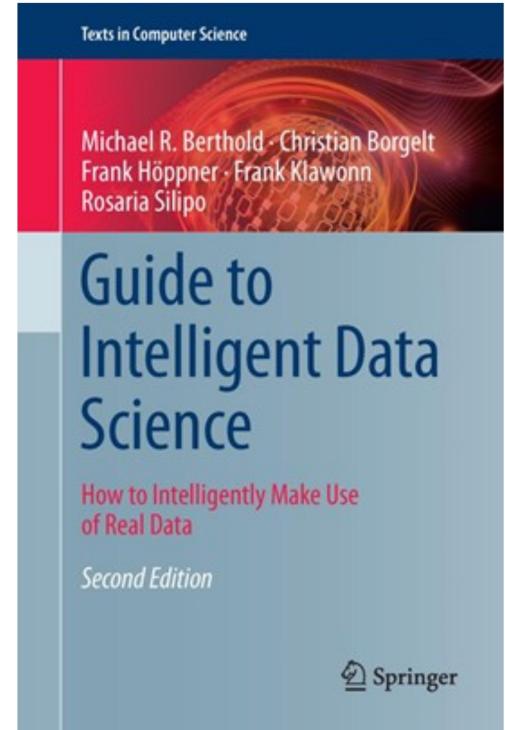
## Structure of each session

- Theory (60 min)
- Work on exercise (15 min)
- Exercise discussion (15 min)
- Break (15 min)

# Material

---

- Michael Berthold, Christian Borgelt, Frank Höppner, Frank Klawonn:  
Guide to Intelligent Data Science  
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.



# What is Data Science?

---

[Wikipedia quoting Dhar 13, Leek 13]

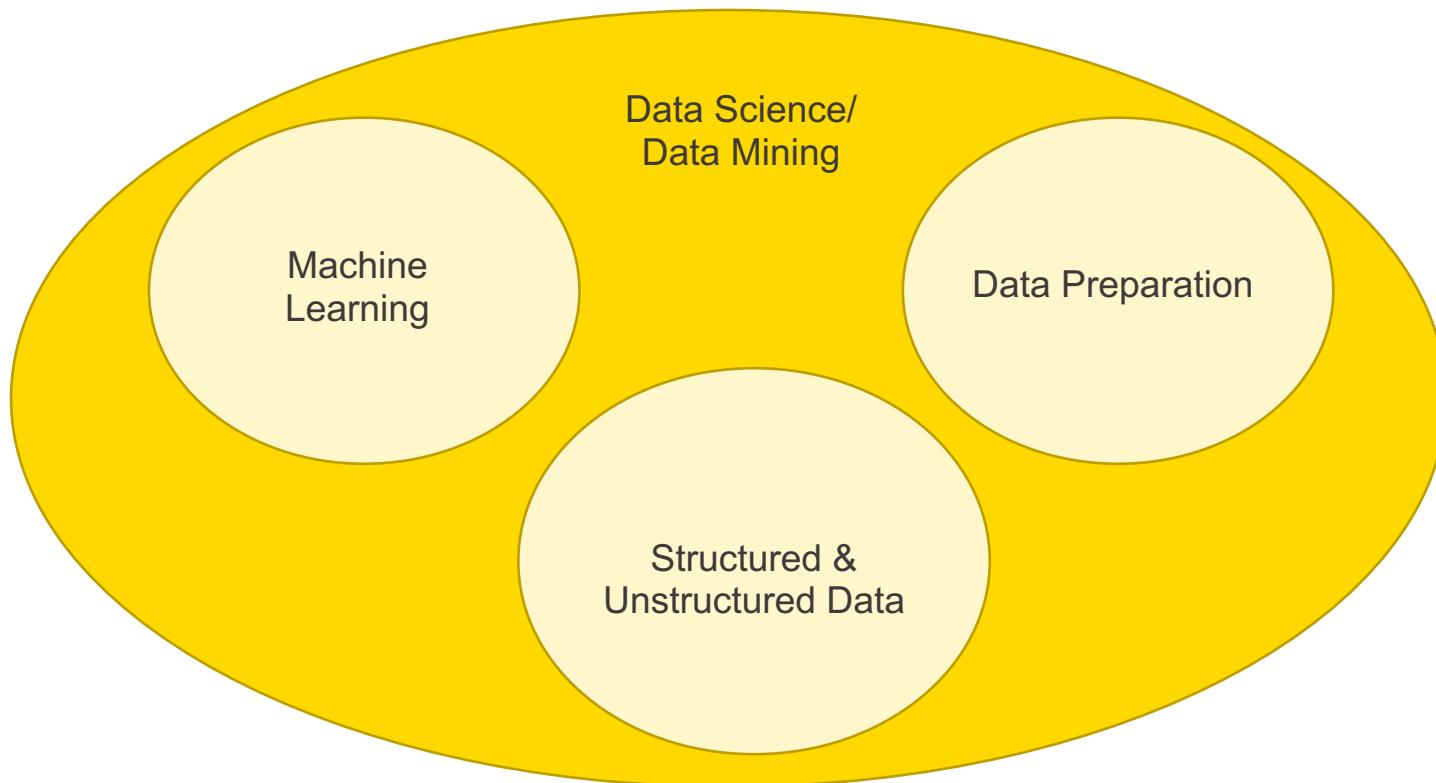
**Data science** is a multi-disciplinary field that uses scientific methods, processes, algorithms and systems to **extract knowledge and insights** from structured and unstructured data.

[Fayyad, Piatetsky-Shapiro & Smyth 96]

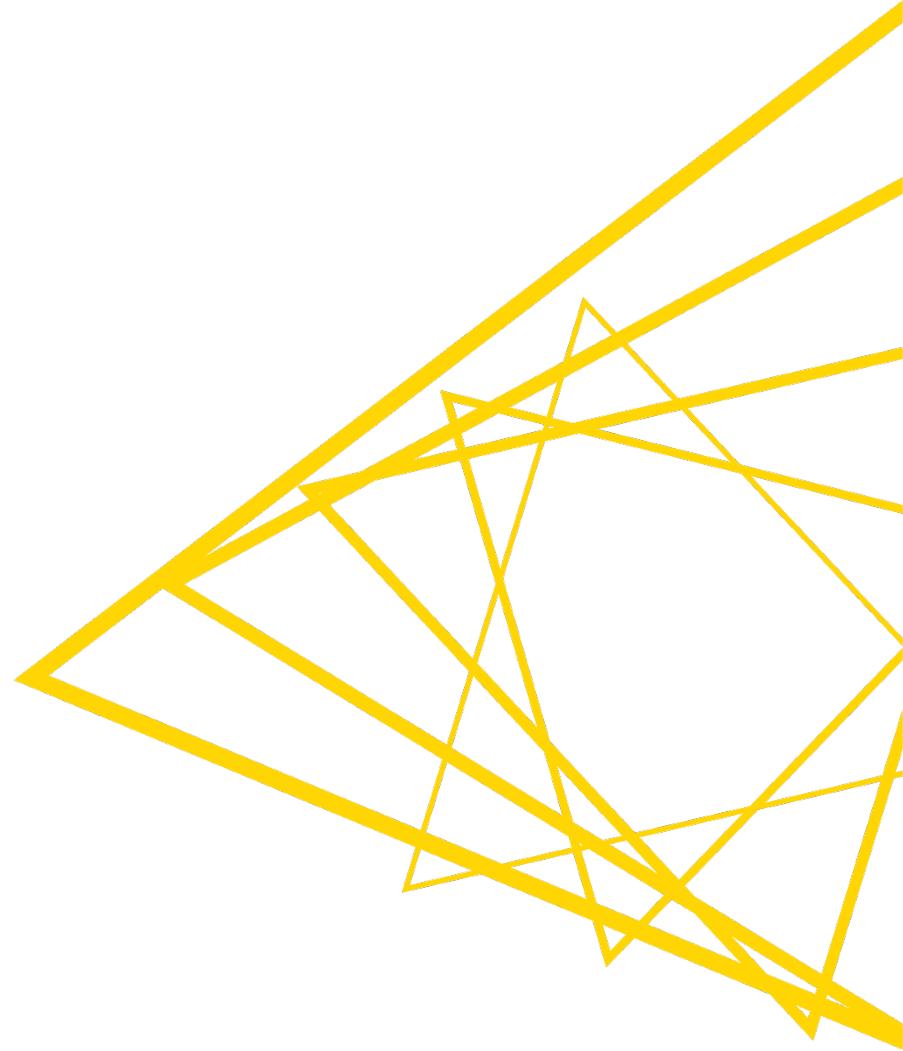
*Knowledge discovery in databases (KDD) is the process of (semi-)automatic **extraction of knowledge** from databases which is *valid, previously unknown, and potentially useful*.*

# What is Data Science?

---



# 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



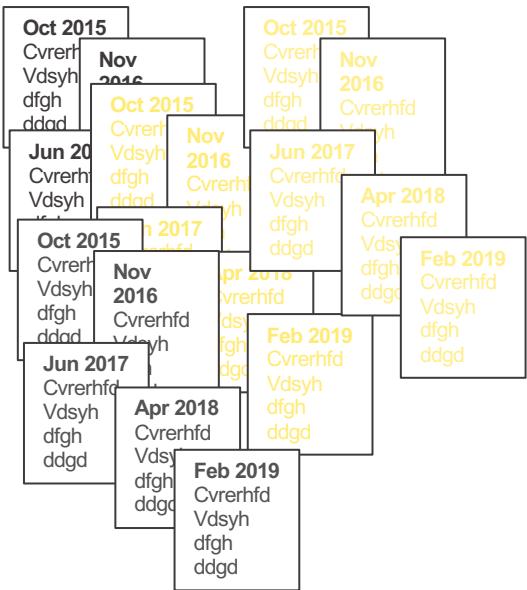
Model



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

# Risk Assessment

## Customer History



## Risk Prognosis

- High Risk
- Low Risk
- High Risk
- Very High Risk
- Very Low Risk
- Medium Risk
- ...

First Name

Last Name

DOB

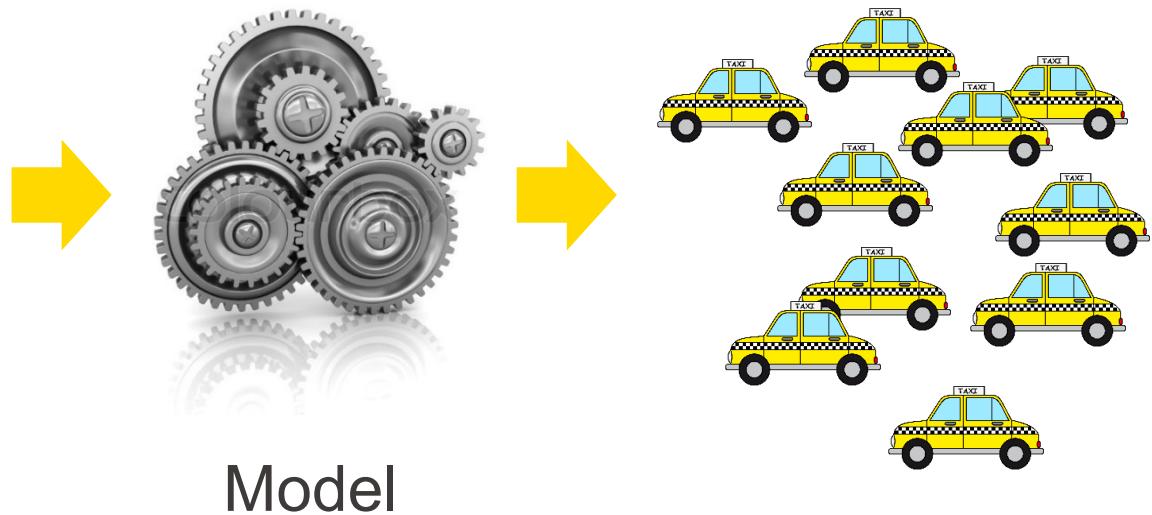
Gender

Employment

Credit Risk

# Demand Prediction

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



# Recommendation Engines / Market Basket Analysis



Recommendation



Model

# Fraud Detection



Transactions

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



Model



Suspicious Transaction

A screenshot of a computer screen displaying a software application titled 'Bank Transaction History Records Company'. The window shows a list of transactions with various columns: Date, Transaction ID, Description, Type, and Reference. A specific transaction is highlighted with a black rectangle and an arrow pointing to it from the 'Suspicious Transaction' text above. The highlighted transaction is dated 11.07.2012, has a transaction ID of 1001, and is described as 'Ticket Sale'. It is categorized as a 'Suspicious' type and has a reference of '11011 - Suspicious'.

# 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

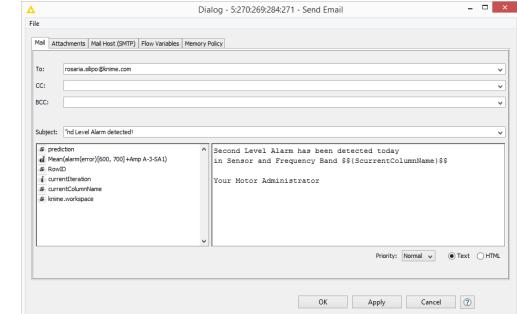
Color: Black Onyx | Verified Purchase

Very bad experience



# Anomaly Detection

Predicting mechanical failure as late as possible but before it happens



Only some Spectral Time Series shows the break down

via REST

# Basic Concepts in Data Science



# What is a Learning Algorithm?

$X = (x_1, x_2, \dots, x_n)$

- Input features
- Input attributes
- Independent variables



Model

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

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

Model parameters

A black arrow points from the text "Model parameters" down to the symbol  $\beta$  in the equation.

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

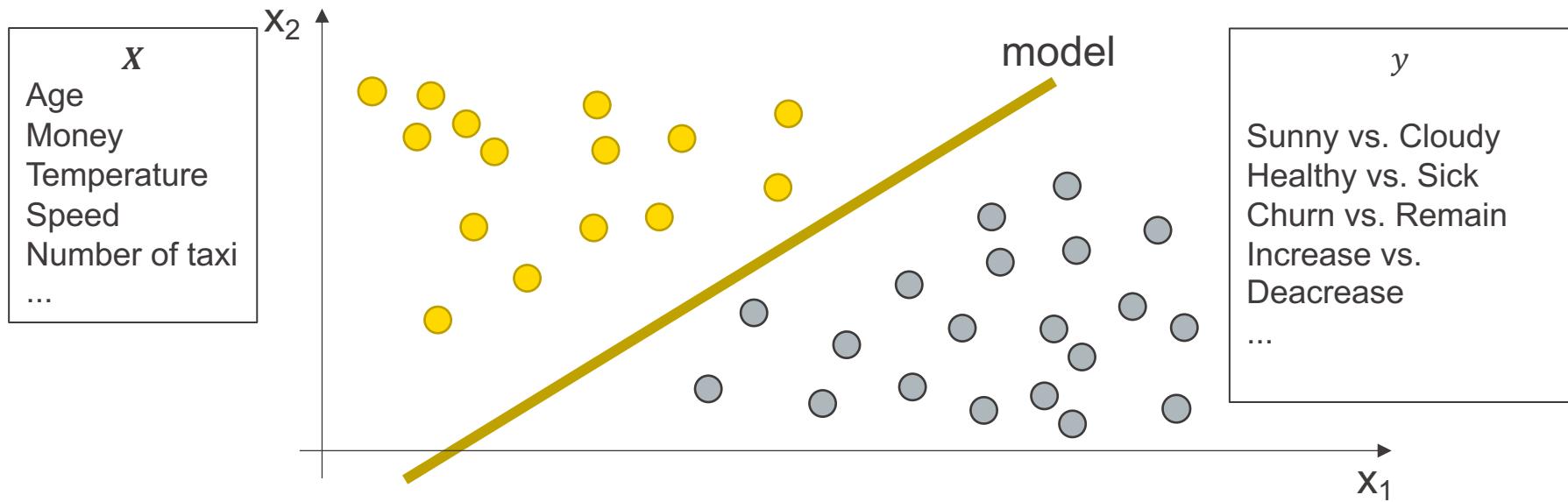
# Algorithm Training / Learning

---

- 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

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



# Supervised Learning: Classification vs. Regression

---

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

## Classification

$y = \{\text{yellow, gray}\}$

$y = \{\text{churn, no churn}\}$

$y = \{\text{increase, unchanged, decrease}\}$

$y = \{\text{blonde, gray, brown, red, black}\}$

$y = \{\text{job 1, job 2, \dots, job n}\}$

## Numerical Predictions (Regression)

$y = \text{temperature}$

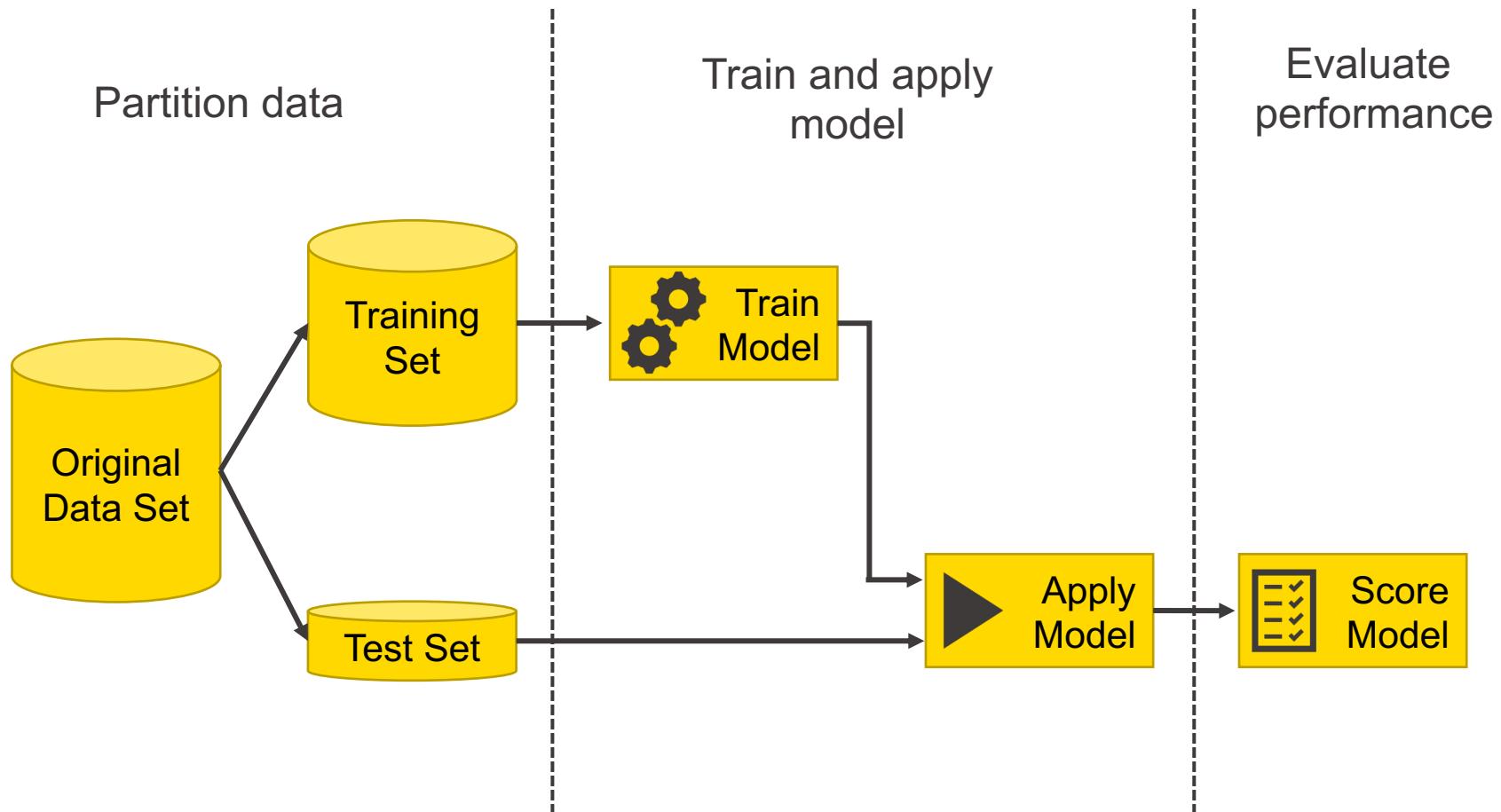
$y = \text{number of visitors}$

$y = \text{number of kW}$

$y = \text{price}$

$y = \text{number of hours}$

# Process Overview for Supervised Learning



# Training vs. Testing: Partitioning

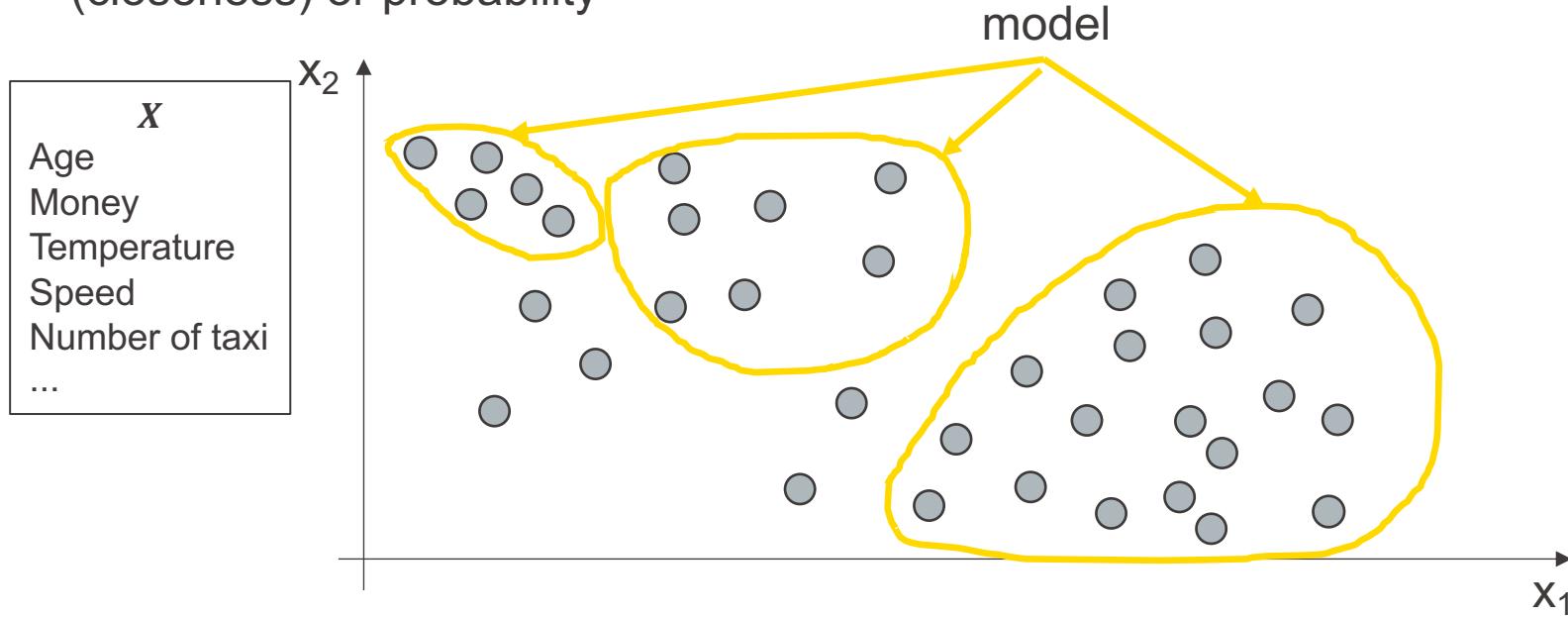
- **Training phase:** the algorithm trains a model using the data in the training set
- **Testing phase:** a metric measures how well the model is performing on data in a new dataset (the test set)



\* sometimes

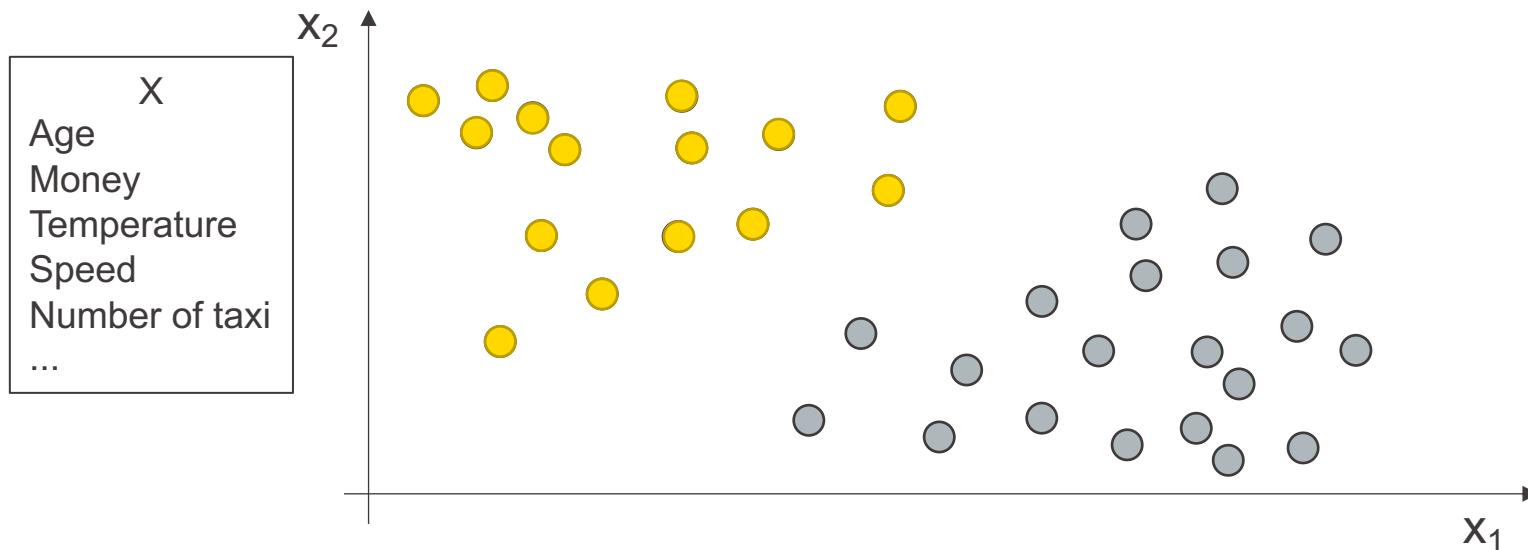
# Unsupervised Learning

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

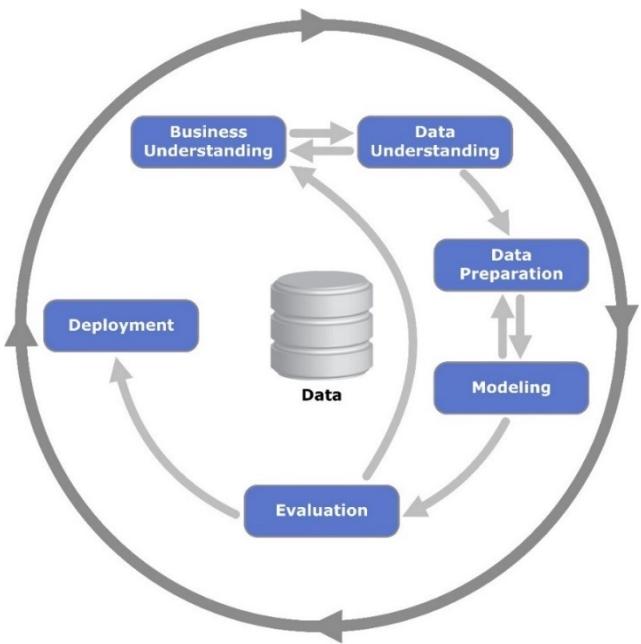


# Semi-Supervised Learning

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



# The CRISP-DM Cycle



Screenshot of the Wikipedia page for the Cross Industry Standard Process for Data Mining:

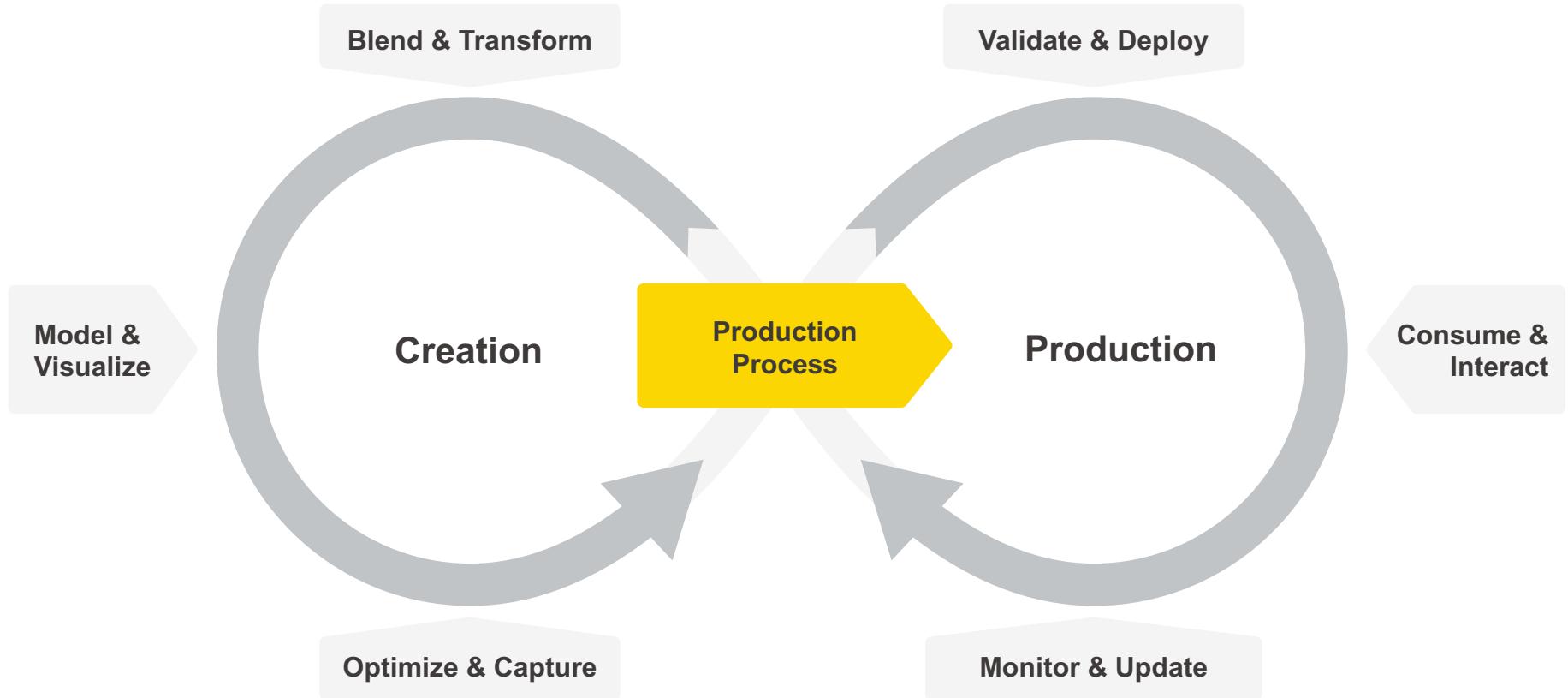
[https://en.wikipedia.org/wiki/Cross\\_Industry\\_Standard\\_Process\\_for\\_Data\\_Mining](https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining)

The page title is "Cross Industry Standard Process for Data Mining". The main content discusses the CRISP-DM methodology, its history, and its replacement by IBM's Analytics Solutions Unified Method for Data Mining. A sidebar contains a table of contents for the major phases.

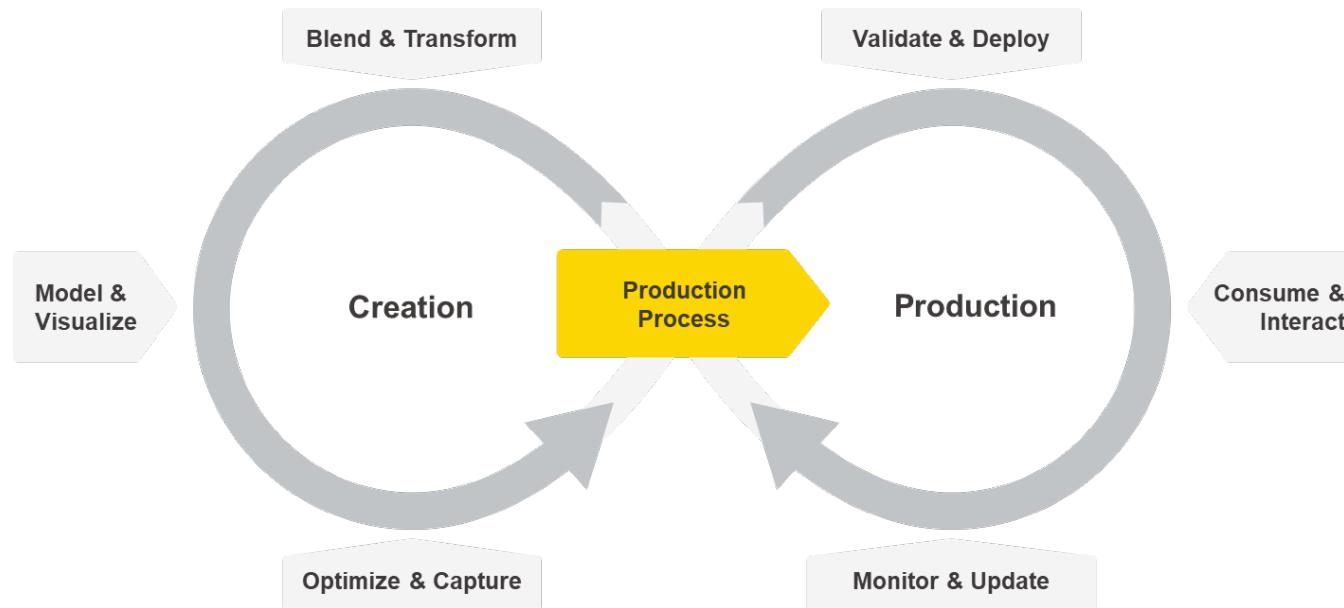
<b>Contents [hide]</b>
<a href="#">1 Major phases</a>
<a href="#">2 History</a>
<a href="#">3 References</a>
<a href="#">4 External Links</a>

[https://en.wikipedia.org/wiki/Cross\\_Industry\\_Standard\\_Process\\_for\\_Data\\_Mining](https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining)

# The Data Science Life Cycle



# KNIME Software for the entire Data Science Life Cycle



## KNIME Analytics Platform

KNIME Extensions

KNIME Integrations

Community Extensions

Partner Extensions

## KNIME Server

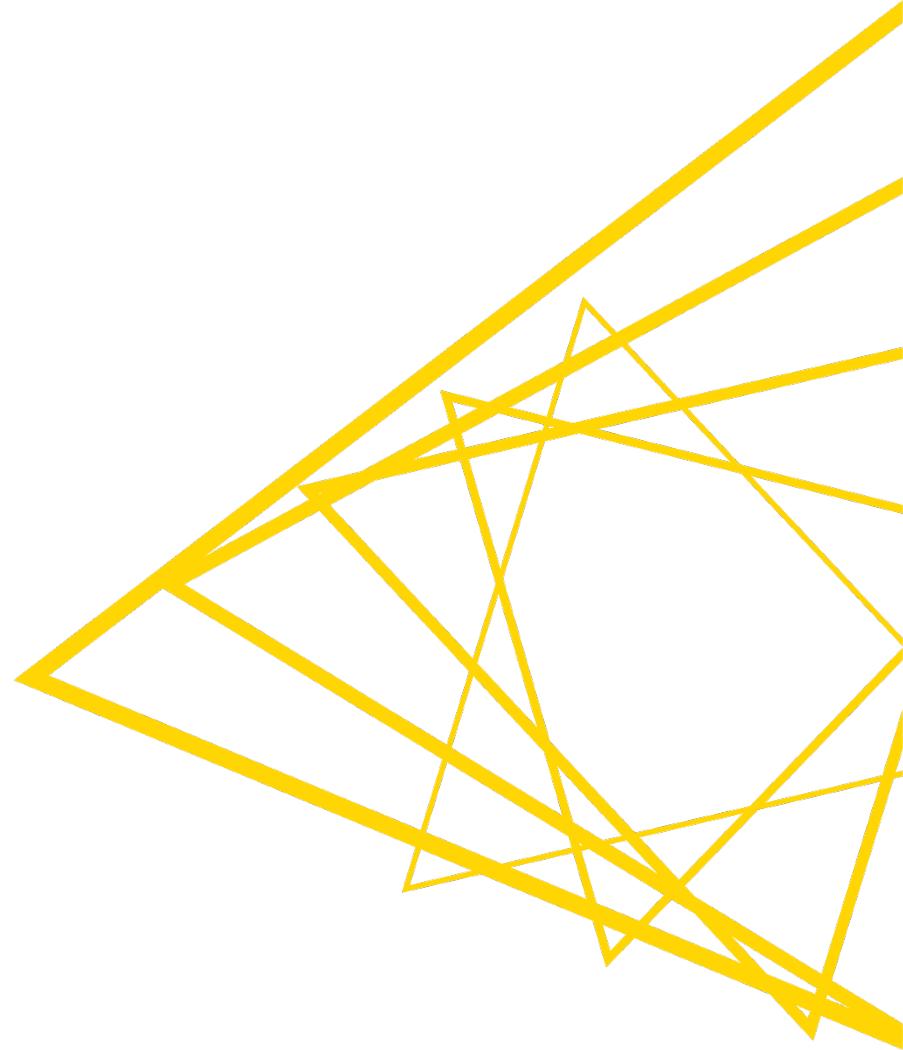
Team Collaboration

End User Applications

API Services

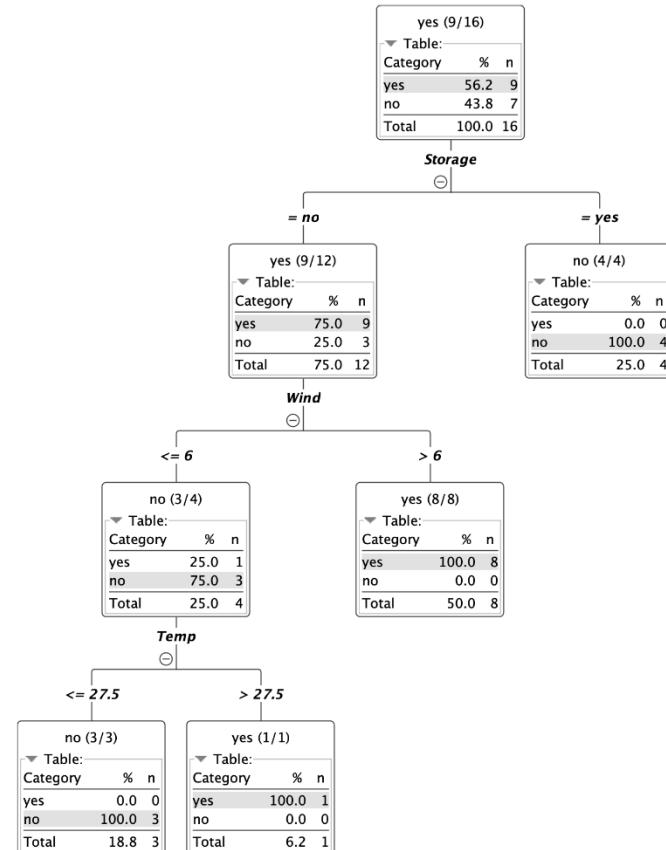
Managed Execution

# Decision Tree Algorithm

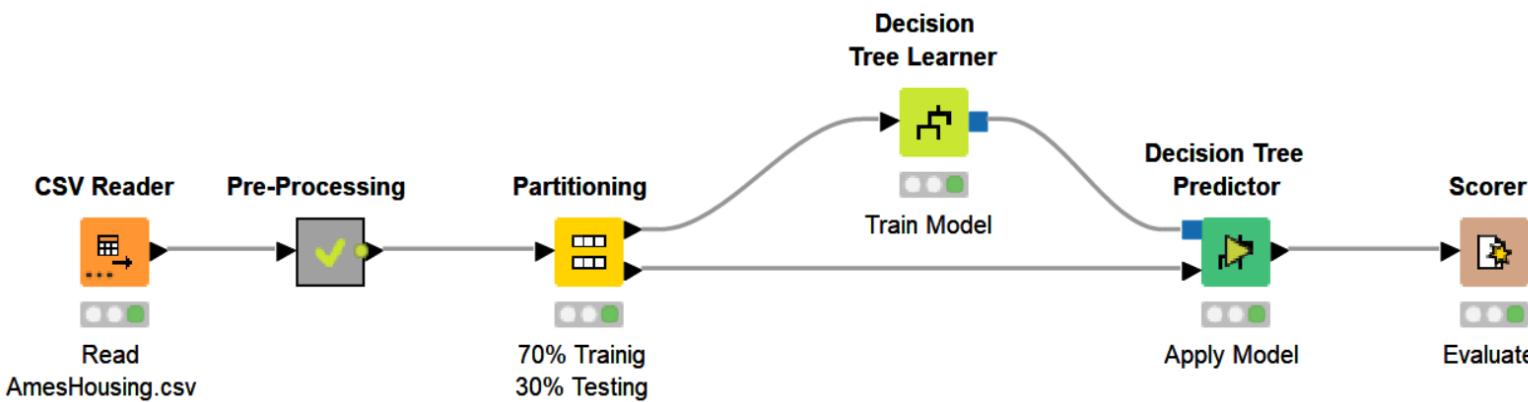


# Goal: A Decision Tree

Outlook	Wind	Temp	(Winter) 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



# 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



Option 1

yes (9/16)
▼ Table:
Category % n
yes 56.2 9
no 43.8 7
Total 100.0 16

Storage

= no

= yes

yes (9/12)
▼ Table:
Category % n
yes 75.0 9
no 25.0 3
Total 75.0 12

Wind

= <= 6

= > 6

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

Outlook

= yes

= no

= rain

= overcast

= BoothTheLake

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

=

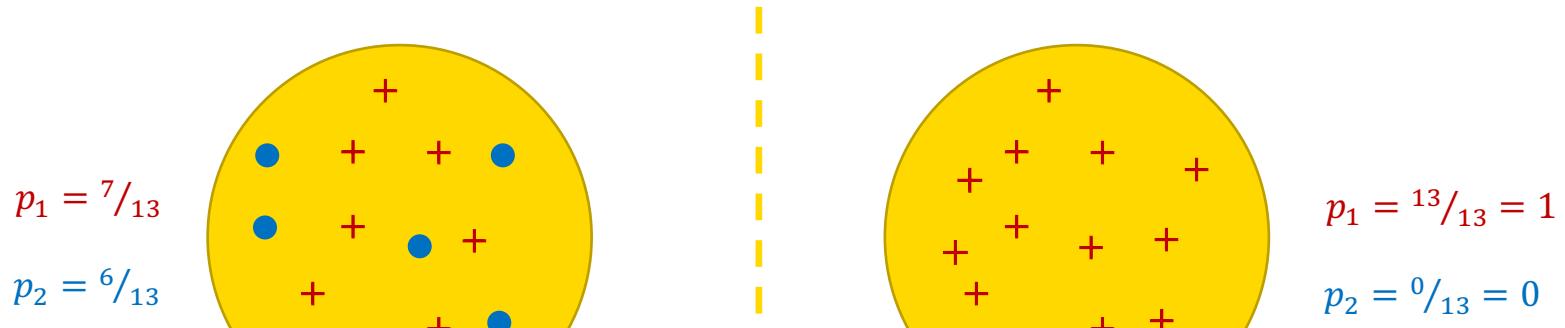
=

=

# Possible Split Criterion: Gain Ratio

Based on entropy = measure for information / uncertainty

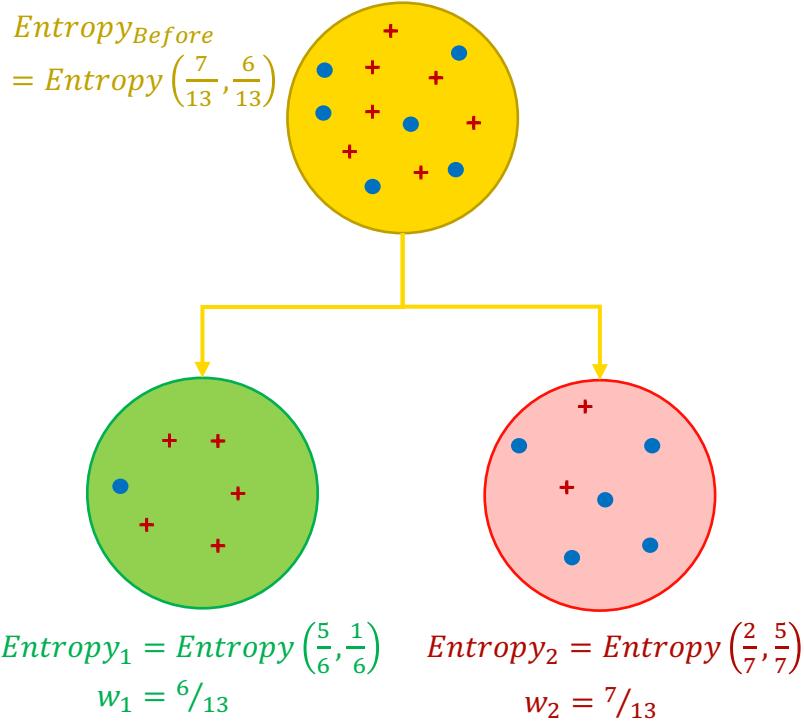
$$\text{Entropy } (p) = - \sum_{i=0}^n p_i \log_2 p_i \text{ for } p \in \mathbb{Q}^n$$



$$\begin{aligned}\text{Entropy } (p) &= -\left(\frac{7}{13} \log_2\left(\frac{7}{13}\right) + \frac{6}{13} \log_2\left(\frac{6}{13}\right)\right) \\ &= 0,995\end{aligned}$$

$$\begin{aligned}\text{Entropy } (p) &= -\left(\frac{13}{13} \log_2\left(\frac{13}{13}\right) + \frac{0}{13} \log_2\left(\frac{0}{13}\right)\right) \\ &= 0\end{aligned}$$

# Possible Split Criterion: Gain Ratio



**Split criterion:**

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

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

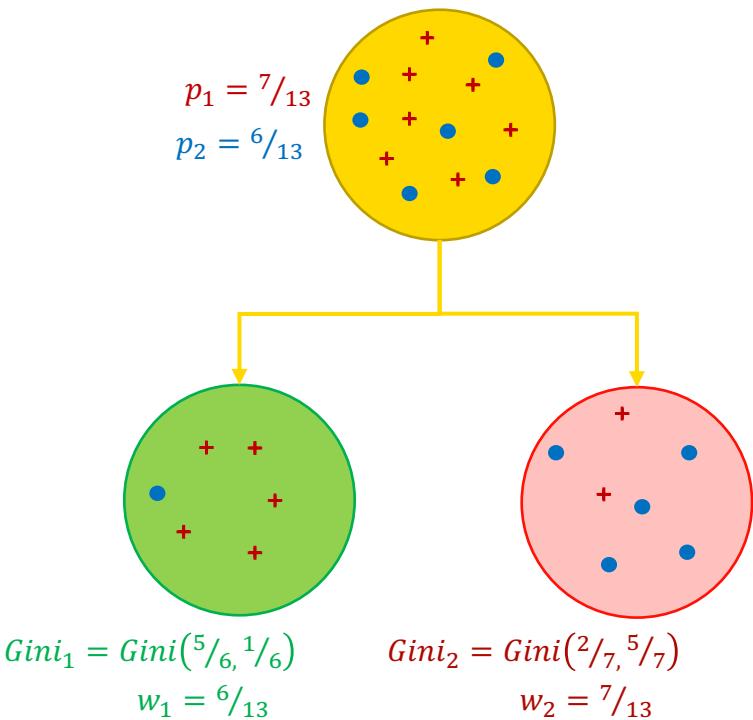
**Next splitting feature:** Feature with highest *Gain*

**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



**Gini index is based on Gini impurity:**

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

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

**Split criterion:**

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

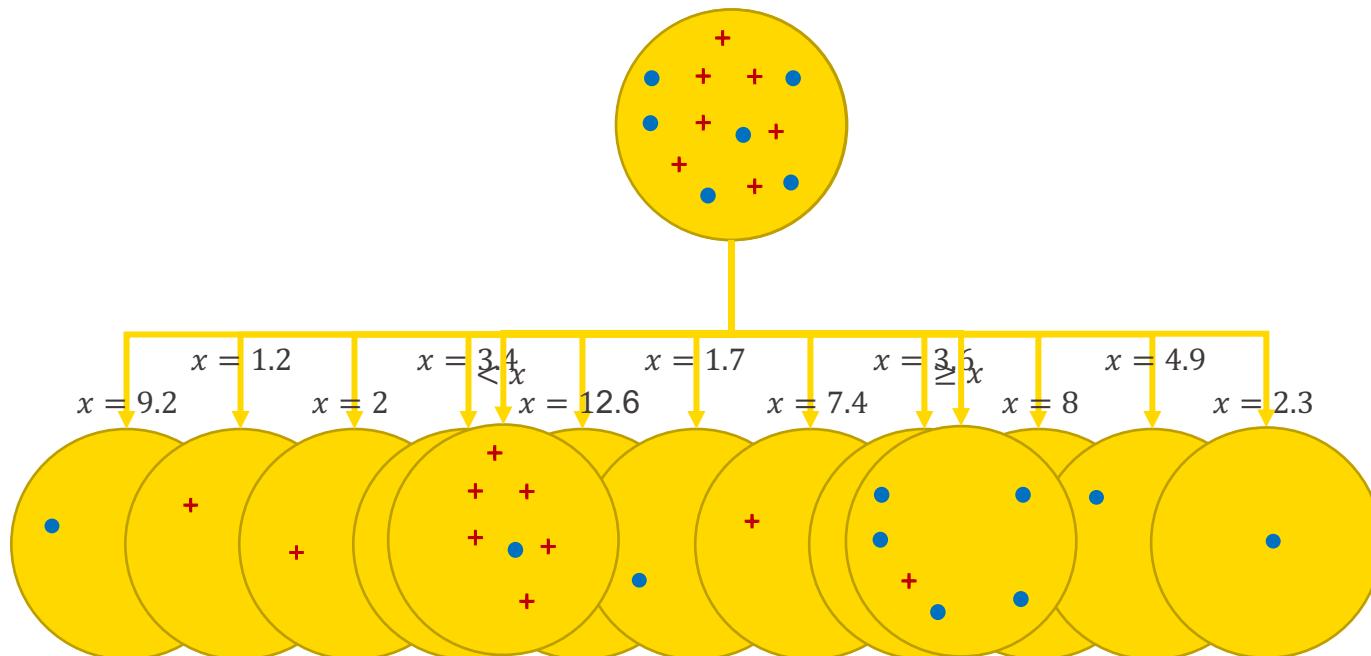
$$Gini_{Index} = \frac{6}{13} Gini_1 + \frac{7}{13} Gini_2$$

**Next splitting feature:**  
Feature with lowest  $Gini_{Index}$

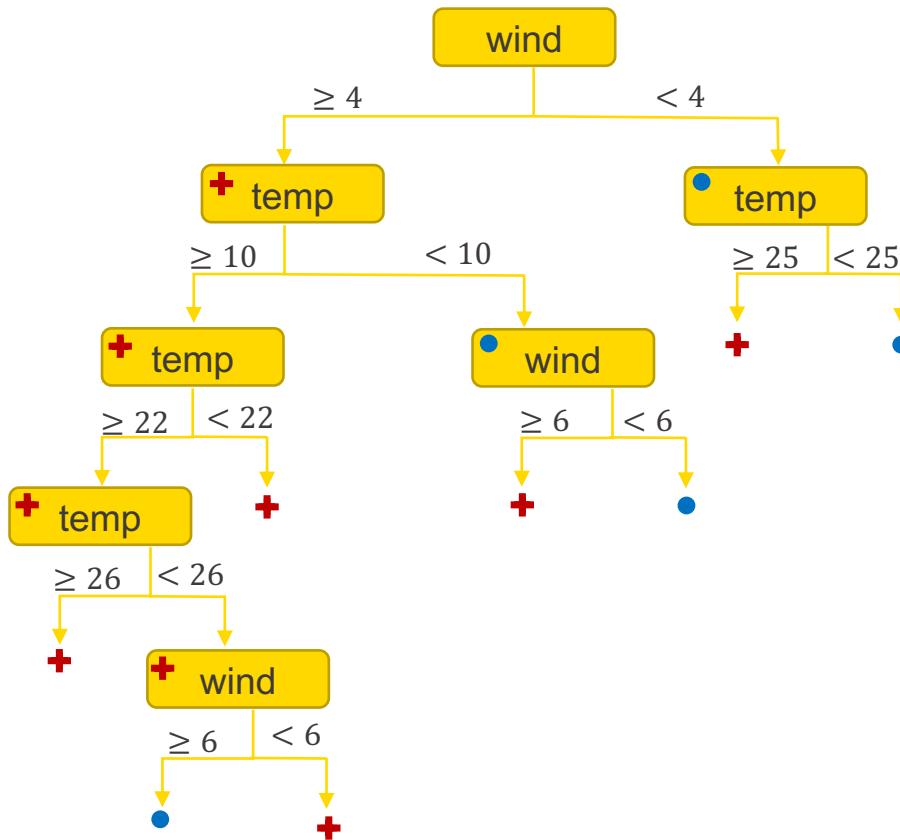
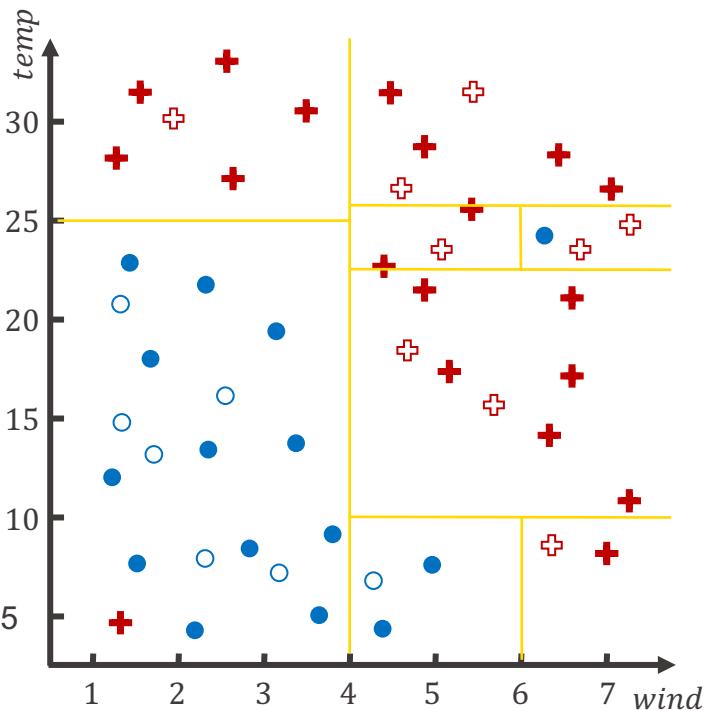
# What happens for numerical Input Features?

Subset for each value? – NO

**Solution:** Binary splits



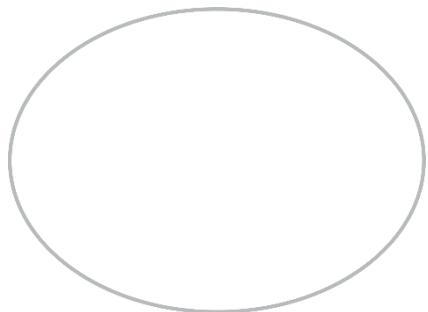
# The Deeper the Better?!



# Overfitting vs Underfitting

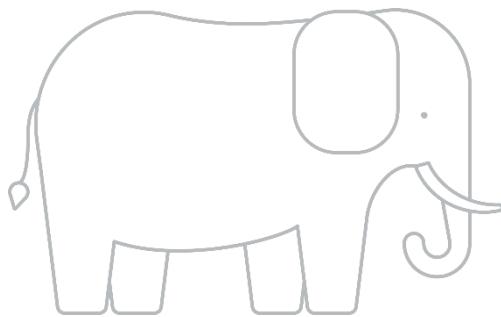
---

Underfitted



Model overlooks  
underlying  
patterns in the  
training set

Generalized



Model captures  
correlations in the  
training set

Overfitted

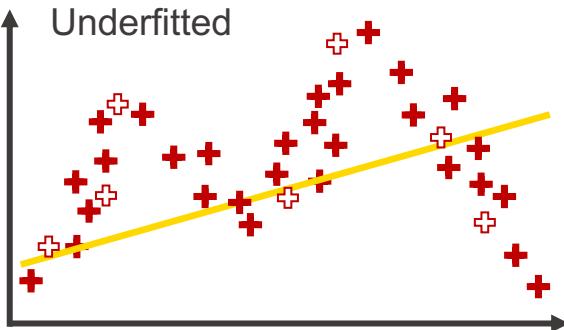


Model memorizes  
the training set  
rather then finding  
underlying patterns

# Overfitting vs Underfitting

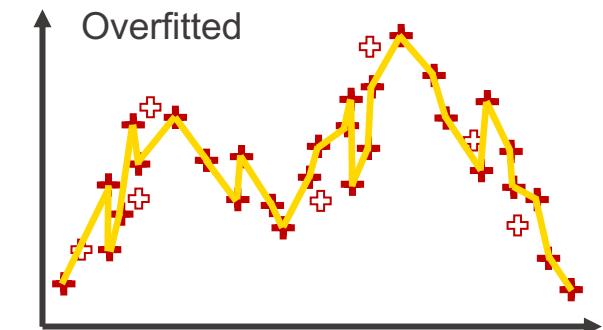
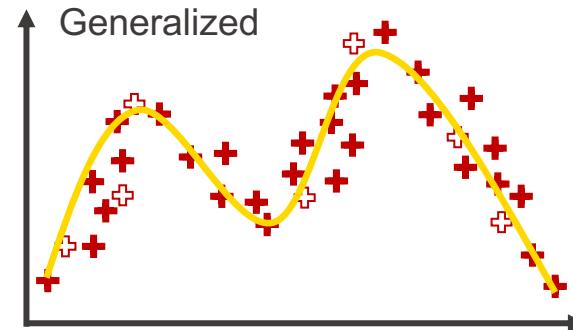
## Underfitting

- A model that can neither model the training data nor generalize to new data



## Overfitting

- Model that fits the training data too well, including details and noise
- Negative impact on the model's ability to generalize



# Controlling the Tree Depth

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

## Pruning

**Idea:** Cut branches that seem as result from overfitting

## Techniques:

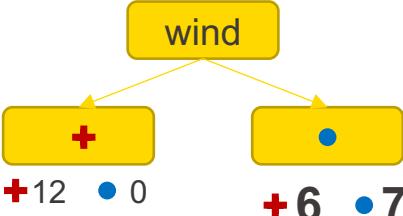
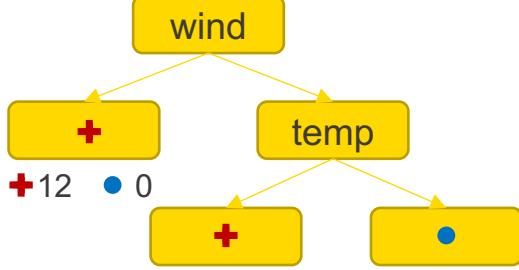
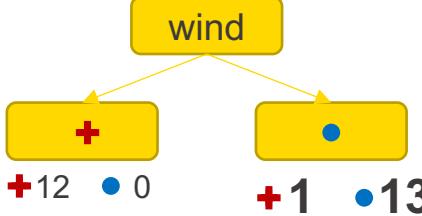
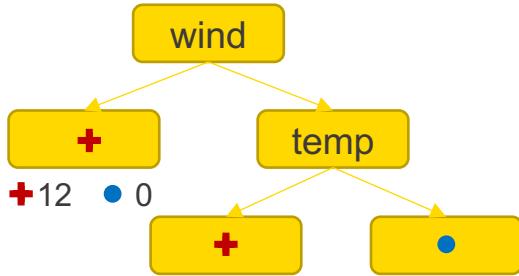
- Reduced Error Pruning
- Minimum description length

## Early stopping

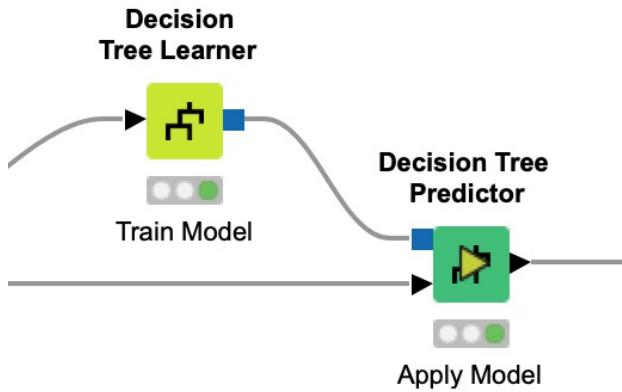
**Idea:** Define a minimum size for the tree leaves

# Pruning - Minimum Description Length Pruning (MDL)

Definition: Description length =  $\# \text{bits(tree)} + \# \text{bits(misclassified samples)}$

	Tree 1	Tree 2	Note
Example 1	 <p>Tree 1</p> <p>wind</p> <p>+12    0</p> <p>+ 6    7</p>	 <p>Tree 2</p> <p>wind</p> <p>+12    0</p> <p>+    +</p> <p>+    .</p>	<p>Many misclassified samples in tree 1</p> <p>=&gt; <math>\text{DL(Tree 1)} &gt; \text{DL(Tree 2)}</math></p> <p>=&gt; Select Tree 2</p>
Example 2	 <p>Tree 1</p> <p>wind</p> <p>+12    0</p> <p>+ 1    13</p>	 <p>Tree 2</p> <p>wind</p> <p>+12    0</p> <p>+    +</p> <p>.    .</p>	<p>Only 1 misclassified sample in tree 1</p> <p>=&gt; <math>\text{DL(Tree 1)} &lt; \text{DL(Tree 2)}</math></p> <p>=&gt; Select Tree 1</p>

# Applying the Model – What are the Outputs?



Classified Data - 0:65 - Decision Tree Predictor (Apply Model)

Row ID	..	SalePr...	rank	D P (rank=Low)	D 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

Training

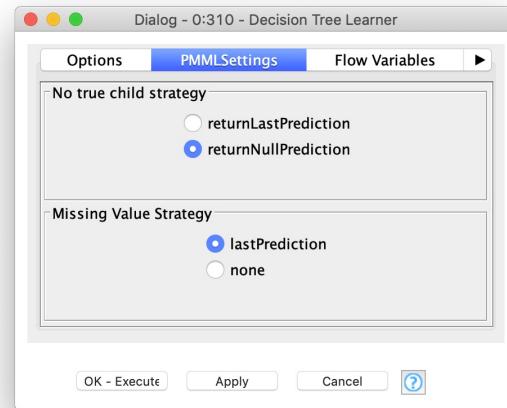
Outlook	Wind	Temp	Storage	Sailing
sunny	3	30	yes	yes
sunny	3	25	yes	no
rain	12	15	yes	yes
rain	16	25	yes	yes
sunny	14	18	yes	yes
rain	3	5	no	no
sunny	9	20	yes	yes
sunny	1	7	no	no
rain	4	25	yes	no
rain	14	24	yes	yes
sunny	11	20	yes	yes
sunny	2	18	yes	no
overcast	8	22	yes	yes
overcast	13	24	yes	yes

Testing

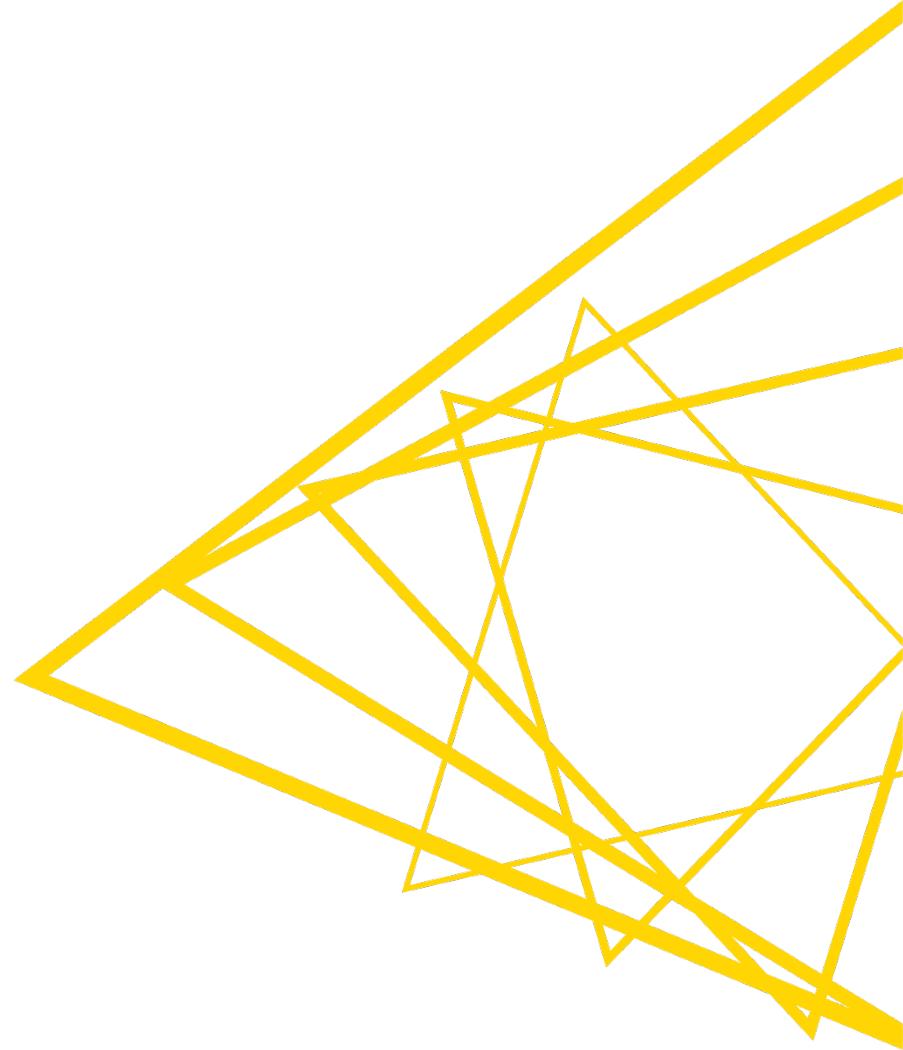
Training:



What happens with outlook = overcast?

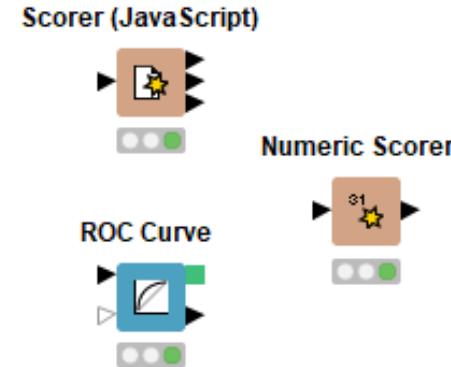


# Evaluation of Classification Models



# 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:

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

- 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

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

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

- 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

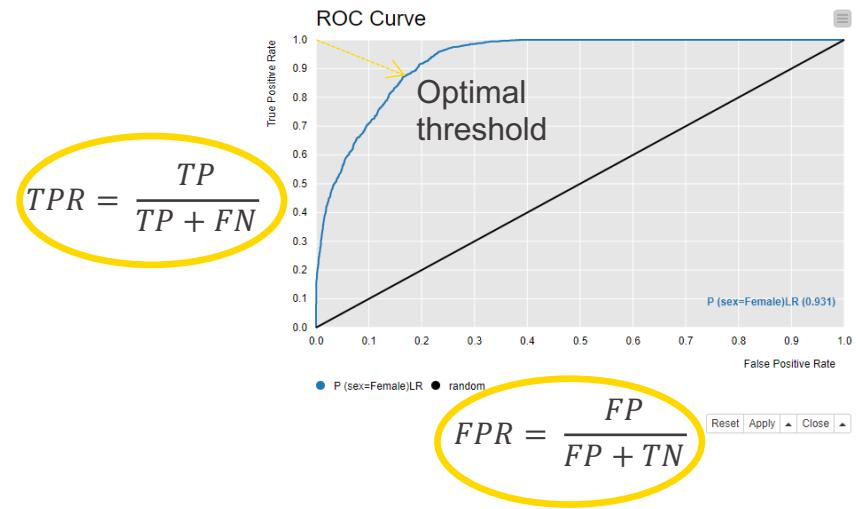
**FALSE POSITIVE (FP)**: Actual class is negative and predicted positive

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

# ROC Curve

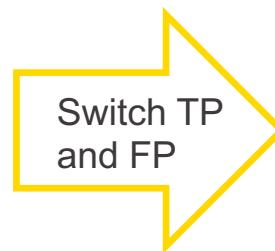
- The ROC Curve shows the false positive rate and true positive rate for different threshold values
  - False positive rate (FPR)
    - negative events incorrectly classified as positive
  - True positive rate (TPR)
    - positive events correctly classified as positive

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



# Cohen's Kappa ( $\kappa$ ) vs. Overall accuracy

	Positive	Negative
Positive	14	6
Negative	5	75



	Positive	Negative
Positive	6	14
Negative	5	75

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

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

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

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

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

Overall accuracy

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

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

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

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

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

$\kappa = 1$ : perfect model performance  
 $\kappa = 0$ : the model performance is equal to a random classifier

# Exercise: 01\_Training\_a\_Decision\_Tree\_Model

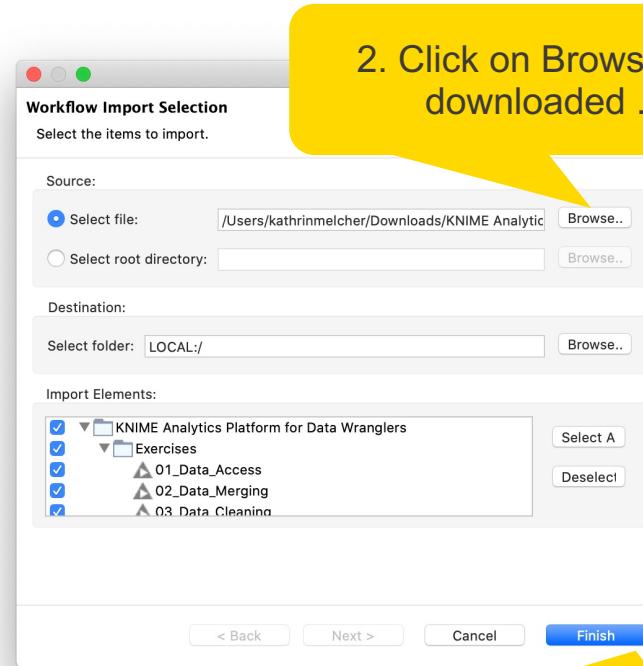
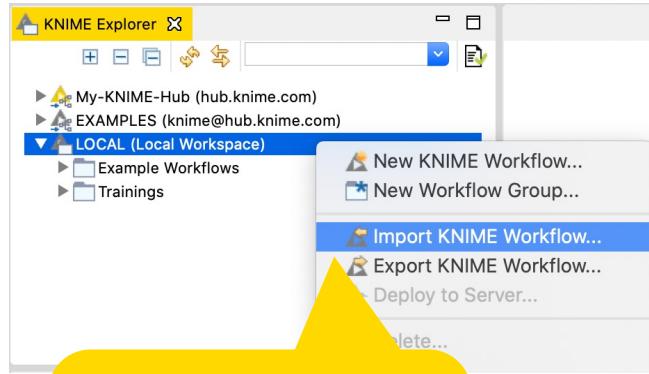
---

- 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

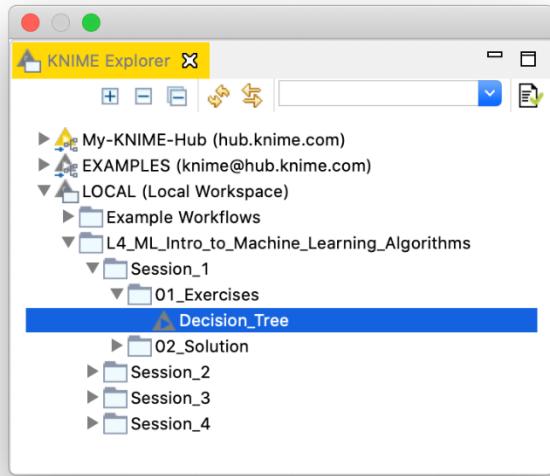
- Import the course material to KNIME Analytics Platform



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.

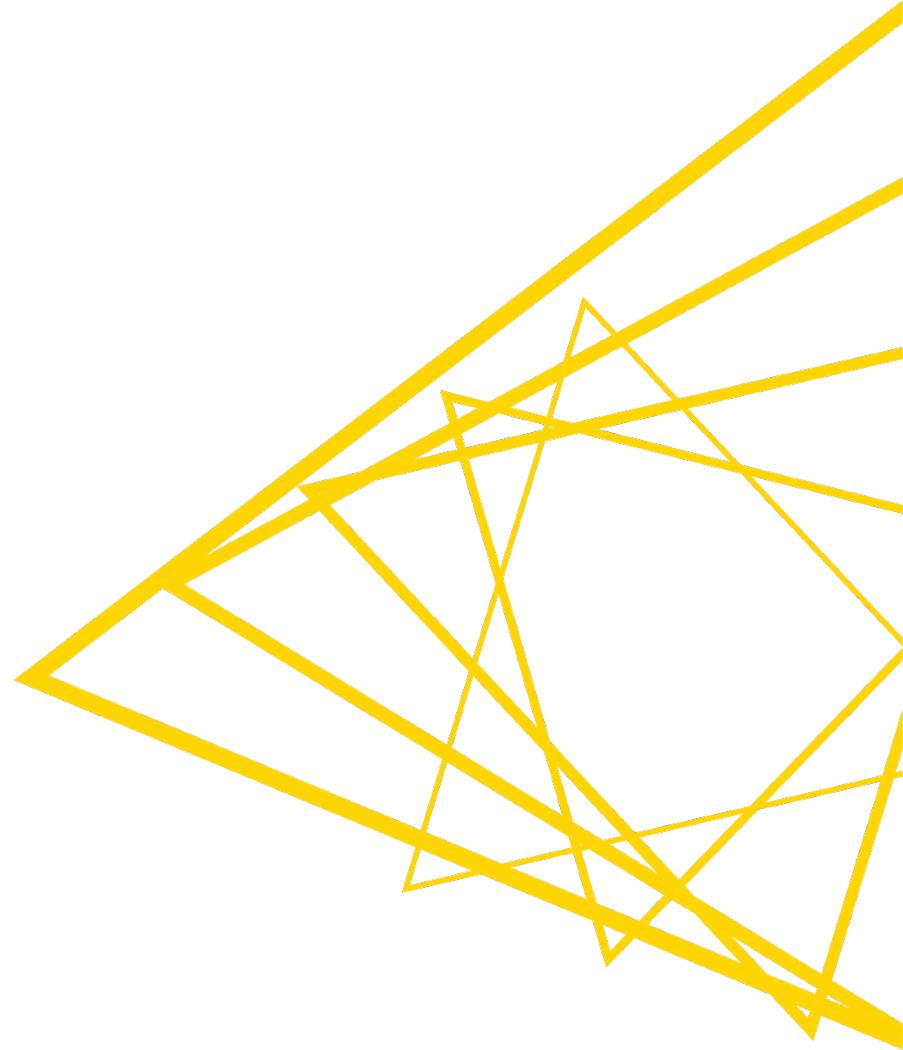
The goal of this exercise is to train a binary classification model, which can predict whether the overall condition is high or low. To do so, the workflow below reads the data set and creates the class column based on overall condition ranking, which is called rank and has the values low if the overall condition is smaller or equal to 5, otherwise high. It is now up to you to continue this workflow!

## Exercise: Decision Tree

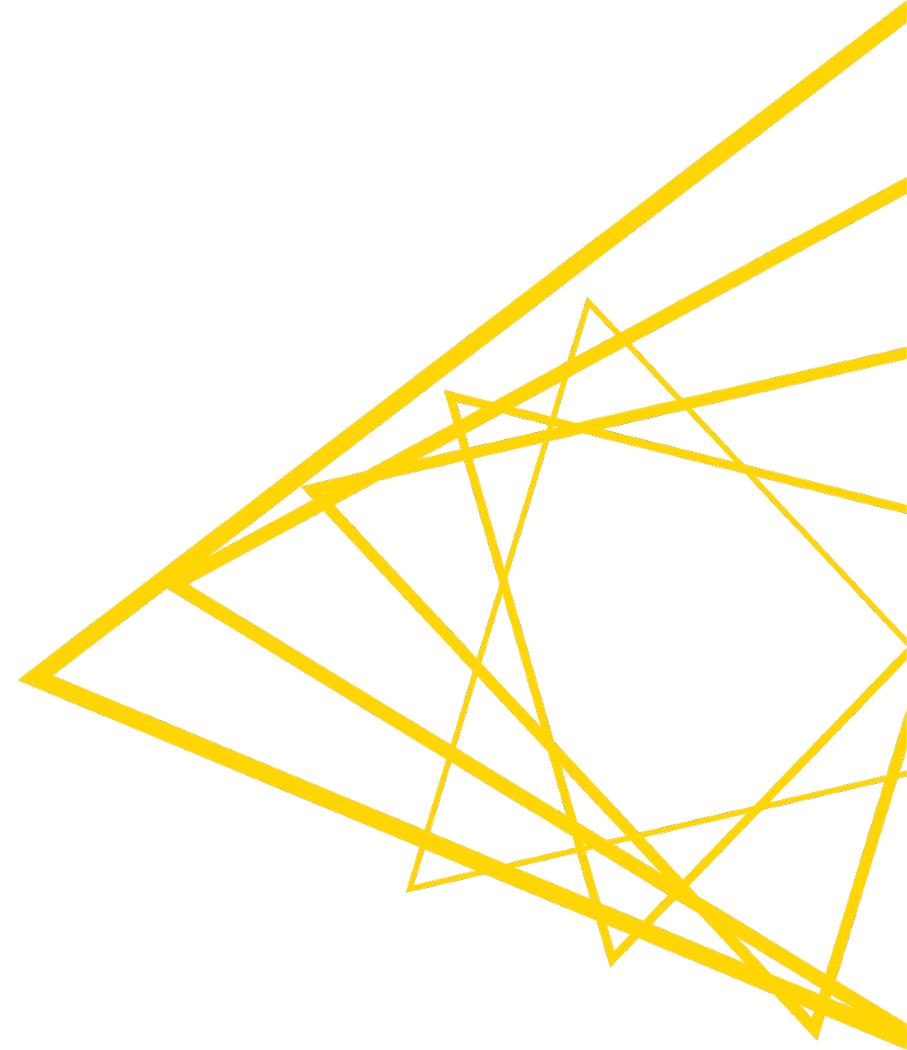
- 1) Use the Partitioning node to split data into training (70%) and test set (30)
  - use stratified sampling based on the column rank, to retain the distribution of the class values in both output tables.
- 2) Train a Decision Tree model to predict the overall condition of a house (high/low) (Decision Tree Learner node)
  - Select the "rank" column as the class column
- 3) Use the trained model to predict the rank of the houses in the test set (Decision Tree Predictor node)
- 4) 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?
- 5) Visualize the ROC curve (ROC Curve node)
  - Make sure that the 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
- 6) **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

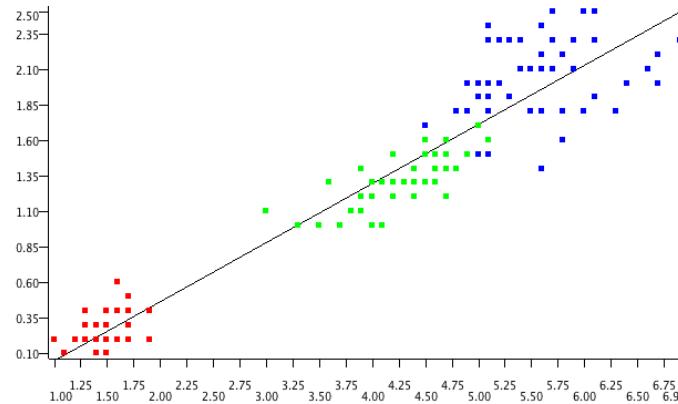
Predict **numeric outcomes on existing data (supervised)**

## Applications

- Forecasting
- Quantitative Analysis

## Methods

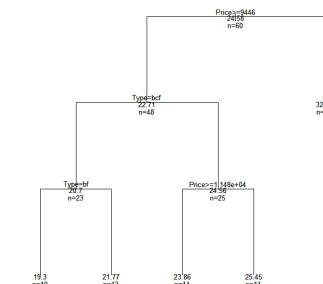
- Linear
- Polynomial
- Regression Trees
- Partial Least Squares



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)  $x_j$

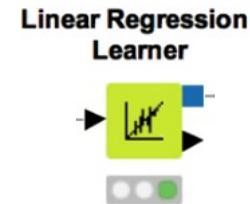
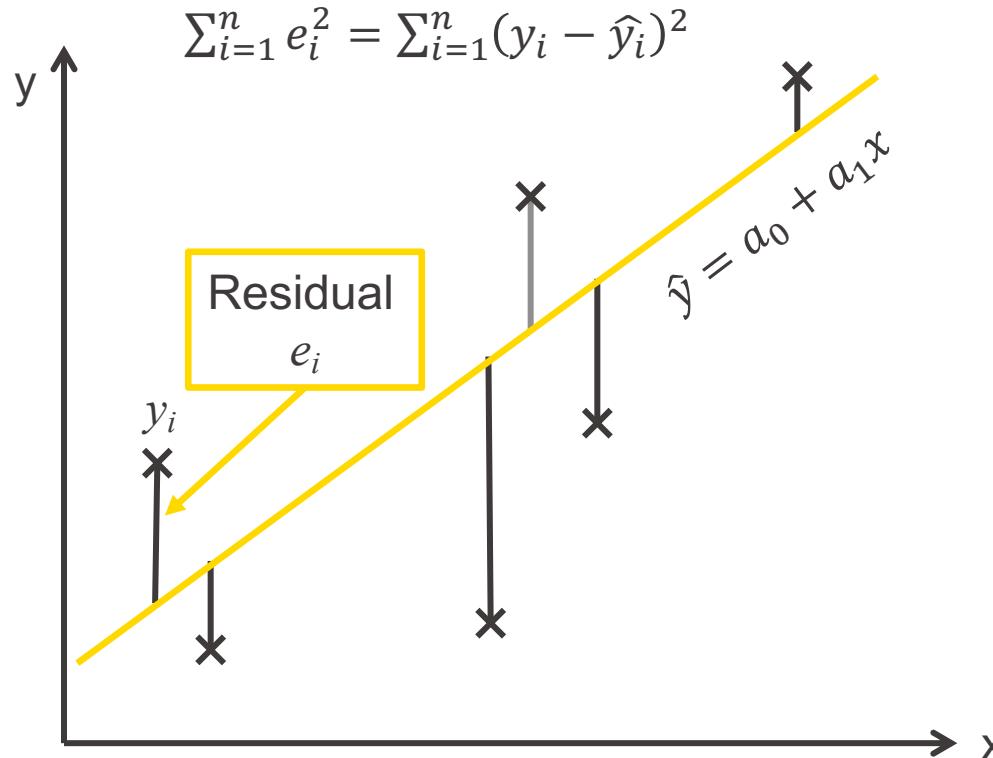
Two input features:  $\hat{y} = a_0 + a_1x_1 + a_2x_2$

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

- 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



# 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  $b$

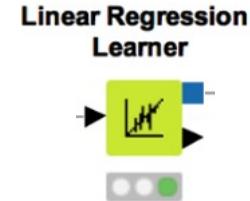
# Linear Regression

- Optimization goal: minimize the squared residuals

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

- Solution:

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



- 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)  $x_j$

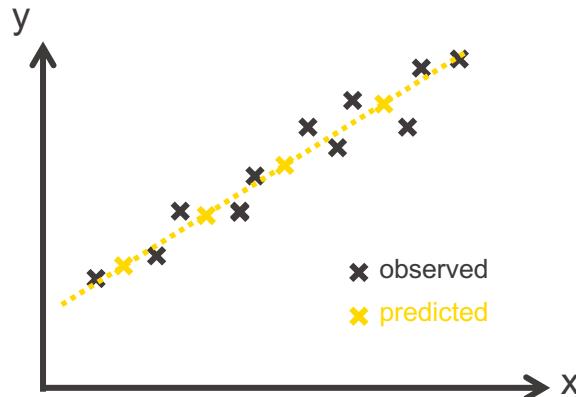
$$\tilde{y} = a_0 + \sum_{j=1}^p a_{j,1}x_j + \sum_{j=1}^p a_{j,2}x_j^2 + \cdots + \sum_{j=1}^p a_{j,d}x_j^d$$

- 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

# Regression vs. Time Series Analysis

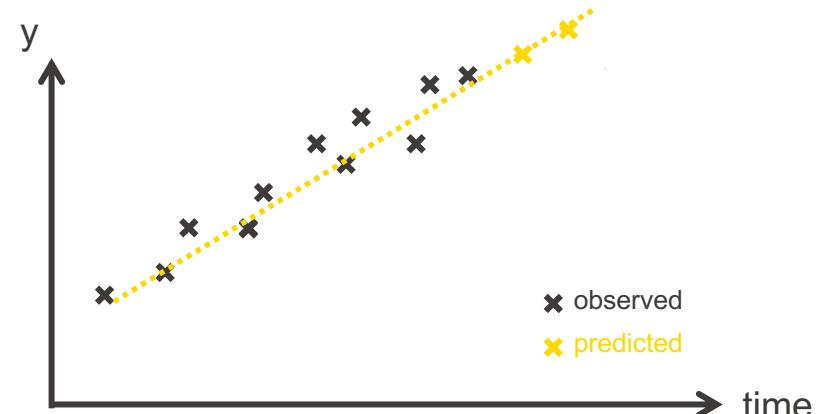
## Regression

- Targets & set of input features
- Describing the relationship between the target and input features
- Model → interpolation



## Time series analysis

- Sequence of observations
- Predicting future obs from
  - Existing time series
  - Accompanying time series
- Model → extrapolation



# Evaluation of Regression Models



# Numeric Errors: Formulas

Error Metric	Formula	Notes
R-squared	$1 - \frac{\sum_{i=1}^n (y_i - f(x_i))^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	Universal range: the closer to 1 the better
Mean absolute error (MAE)	$\frac{1}{n} \sum_{i=1}^n  y_i - f(x_i) $	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)	$\frac{1}{n} \sum_{i=1}^n \frac{ y_i - f(x_i) }{ y_i }$	Requires non-zero target column values

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

MAE	RMSE
Easy to interpret – mean 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
<b>Relative measure:</b> Proportion of variability explained by the model	<b>Absolute measure:</b> How much deviation at each point
Range: Usually between 0 and 1. 0 = no variability explained 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

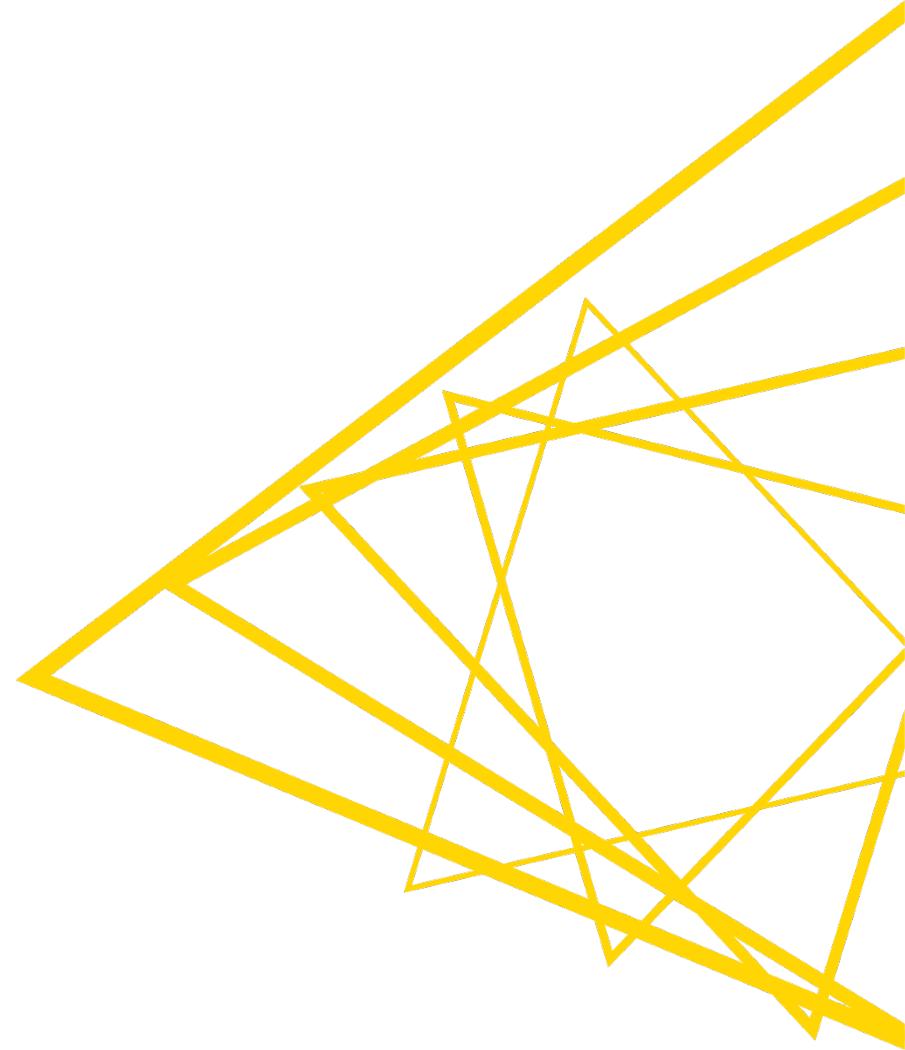
# Numeric Scorer

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

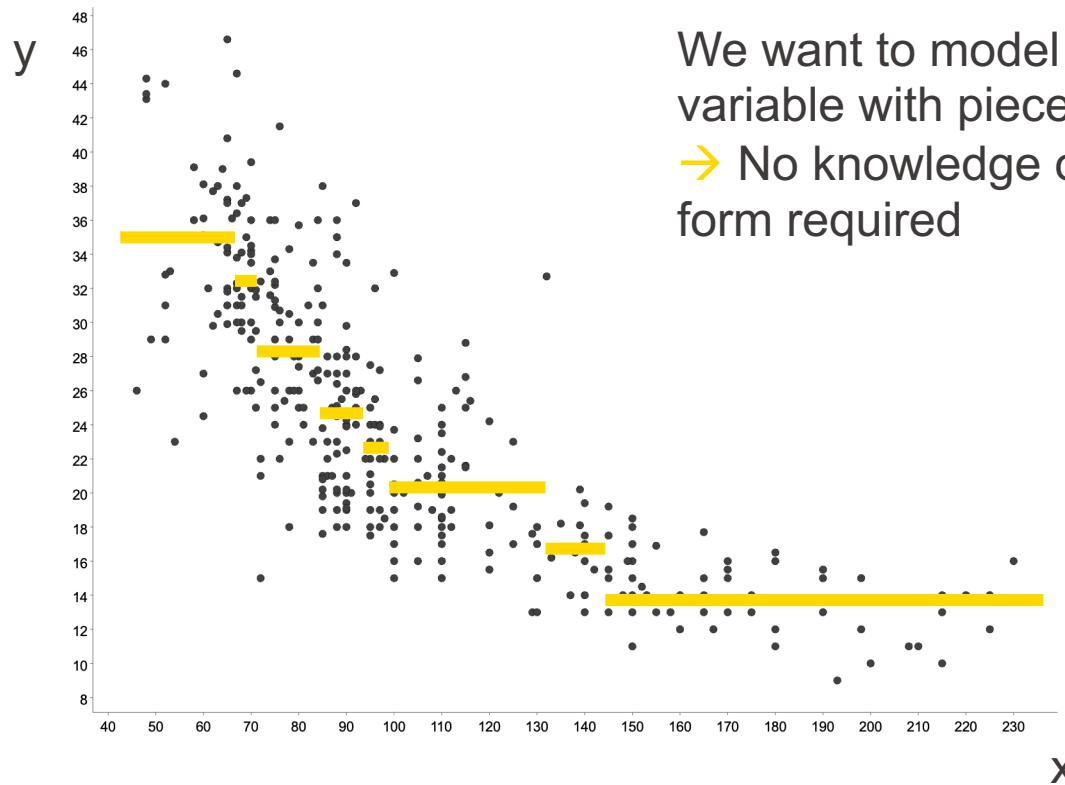
The image shows the KNIME interface. On the left, there is a node titled "Numeric Scorer" with an input arrow pointing to it and an output arrow pointing away. Below the node are three small colored circles (red, green, blue). To the right is a window titled "Statistics - 0:393 - Numeric Scorer". The window has tabs: File, Hilite, Navigation, View, Table "Scores" - Rows: 6, Spec - Column: 1, Properties, and Flow Variables. The "Table" tab is selected, displaying the following data:

Row ID	MA(Irregular Component)
R <sup>2</sup>	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



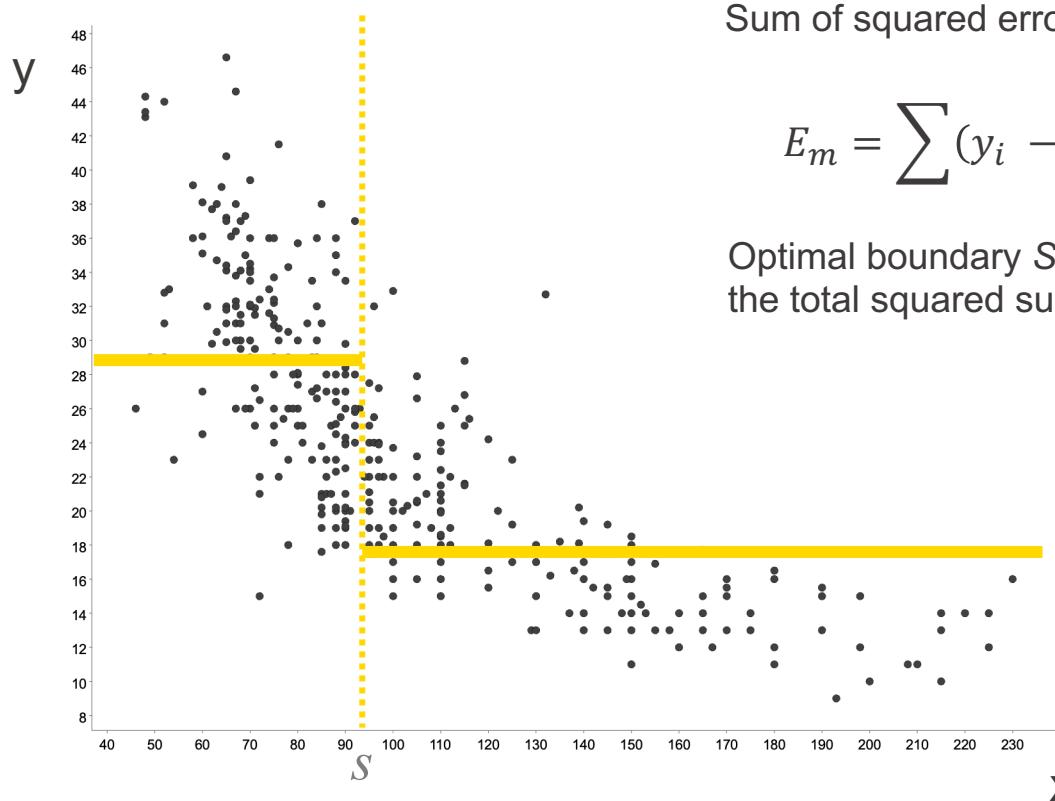
We want to model the target variable with piecewise lines  
→ No knowledge of functional form required

# Regression Tree: Initial Split

Local mean:

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

For observations in  
segment  $m$



Sum of squared errors:

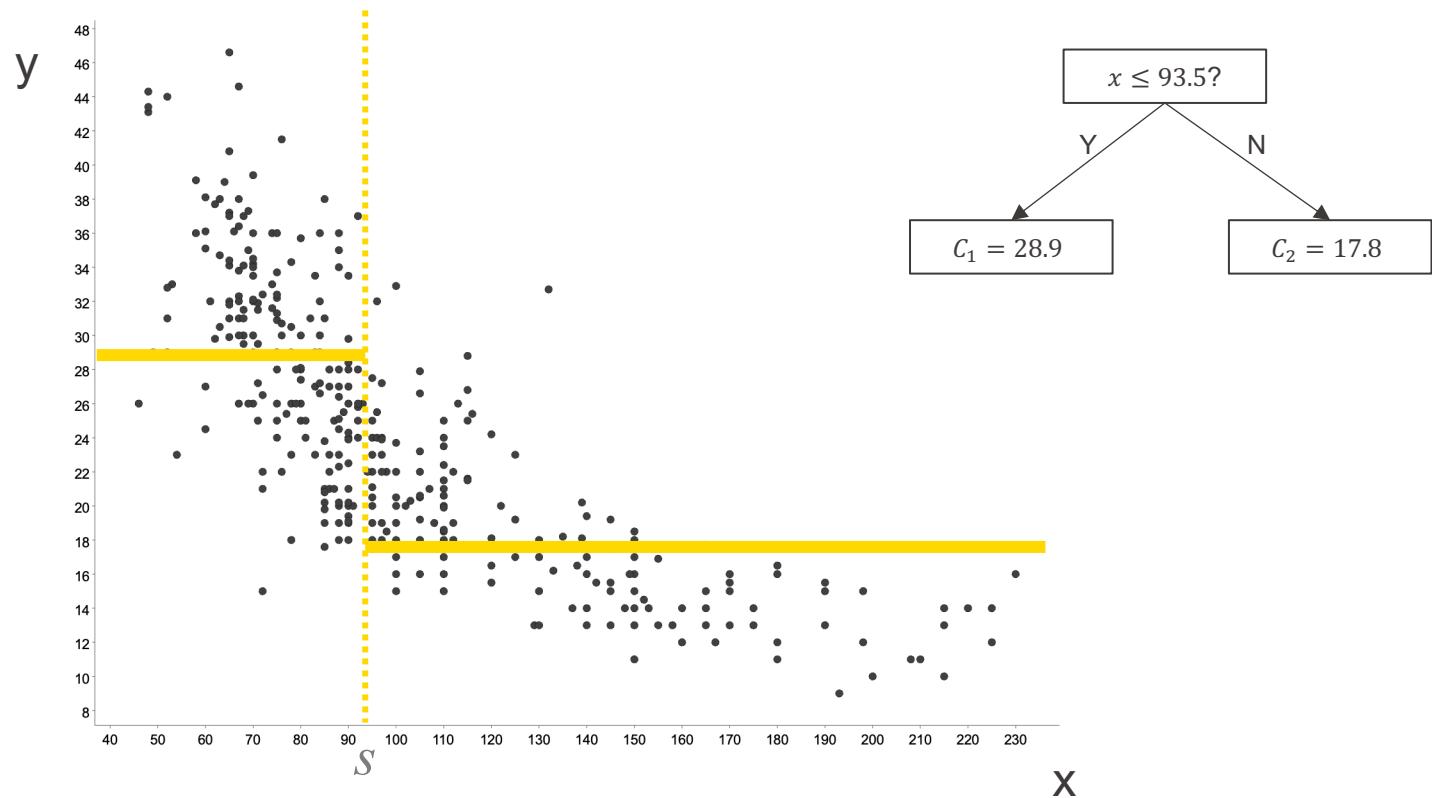
$$E_m = \sum (y_i - c_m)^2$$

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

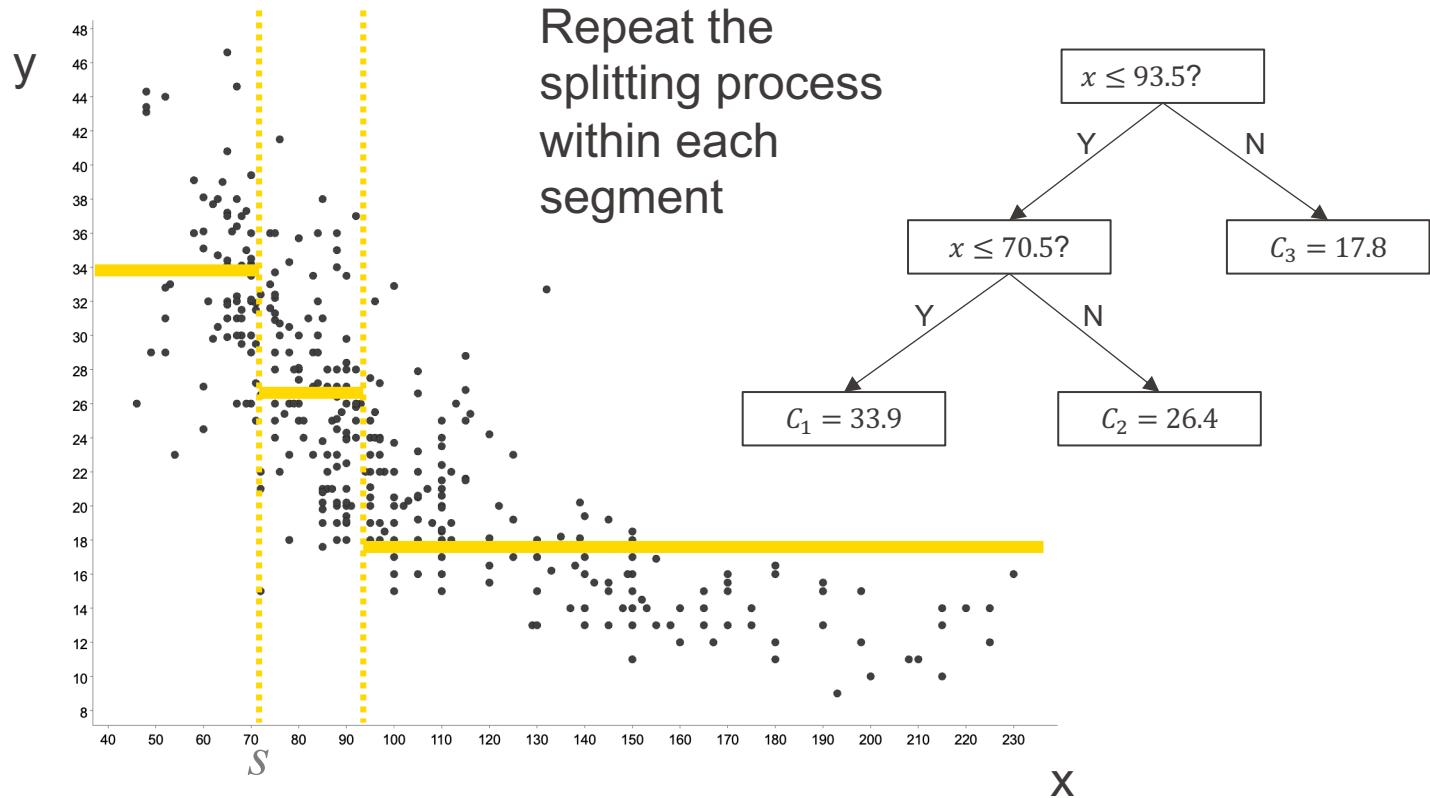
$$\sum E_m$$

For all segments  $m$

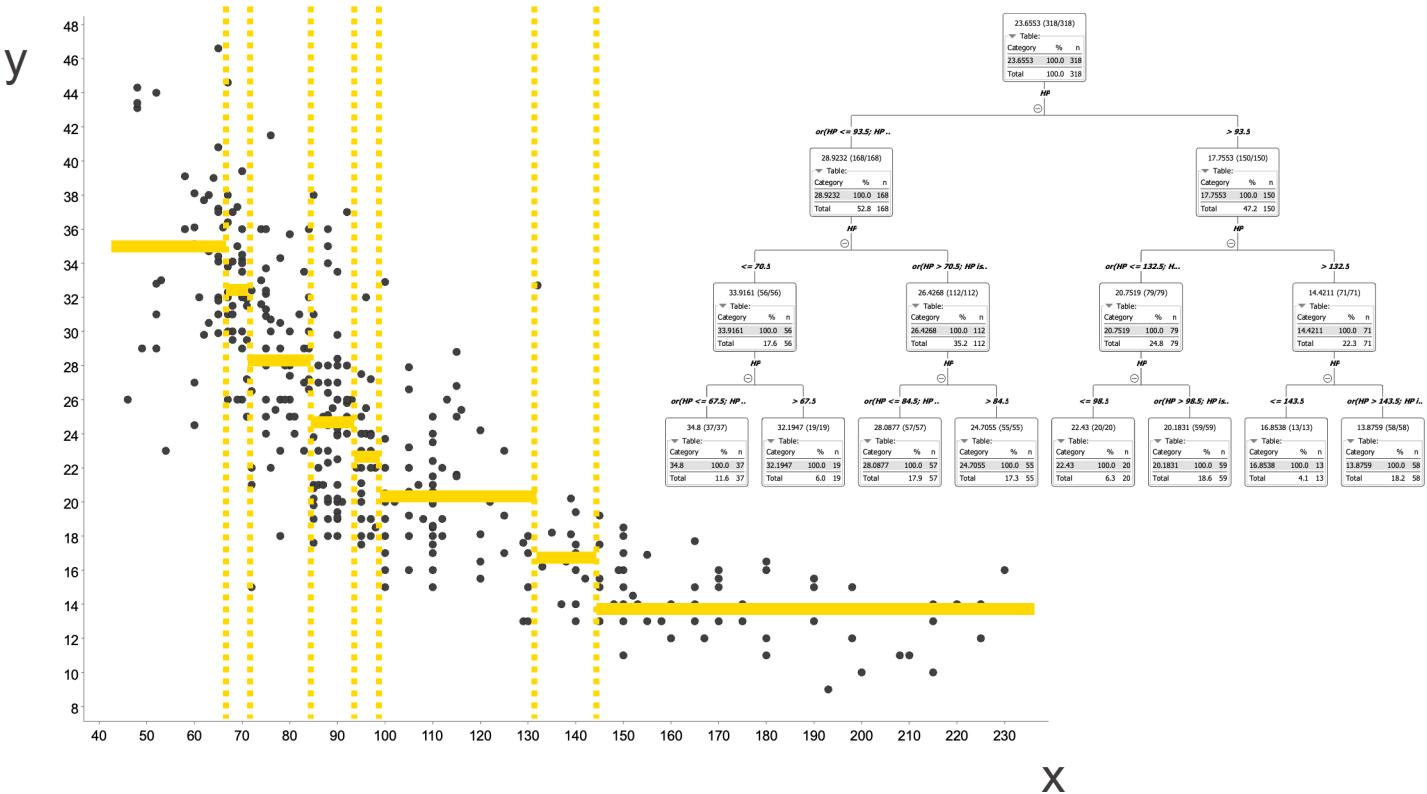
# Regression Tree: Initial Split



# Regression Tree: Growing the Tree



# Regression Tree: Final Model

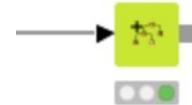


# Regression Tree: Algorithm

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.

Simple Regression  
Tree Learner



# Regression Trees: Summary

---

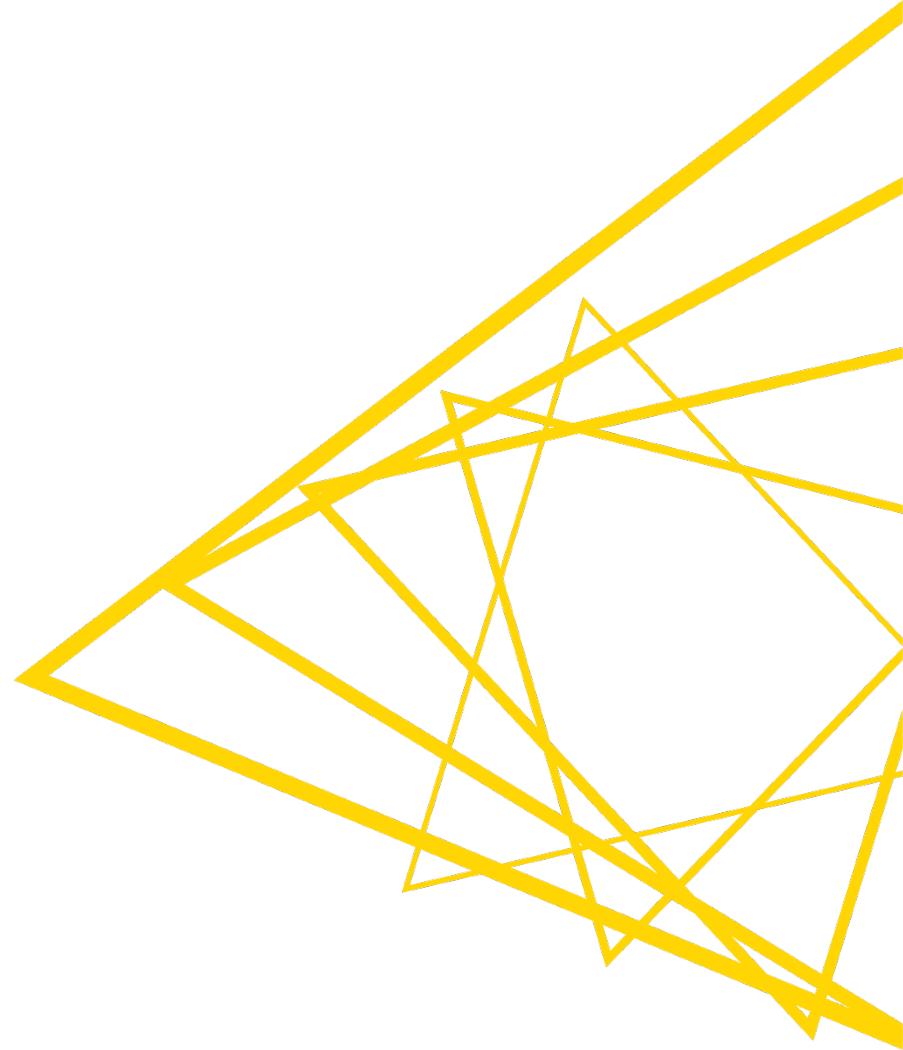
- Differences to decision trees:
  - Splitting criterion: minimizing intra-subset variation (error)
  - Pruning criterion: based on numeric error measure
  - Leaf node predicts average target values of training instances reaching that node
- Can approximate piecewise constant functions
- Easy to interpret

# Regression Trees: Pros & Cons

---

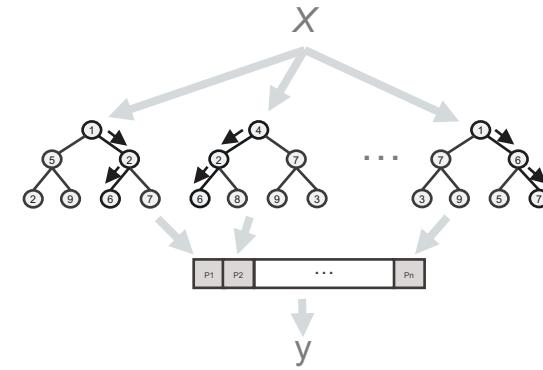
- Finding of (local) regression values (average)
- Problems:
  - No interpolation across borders
  - Heuristic algorithm: unstable and not optimal.
- Extensions:
  - Fuzzy trees (better interpolation)
  - Local models for each leaf (linear, quadratic)

# Ensemble Models



# Tree Ensemble Models

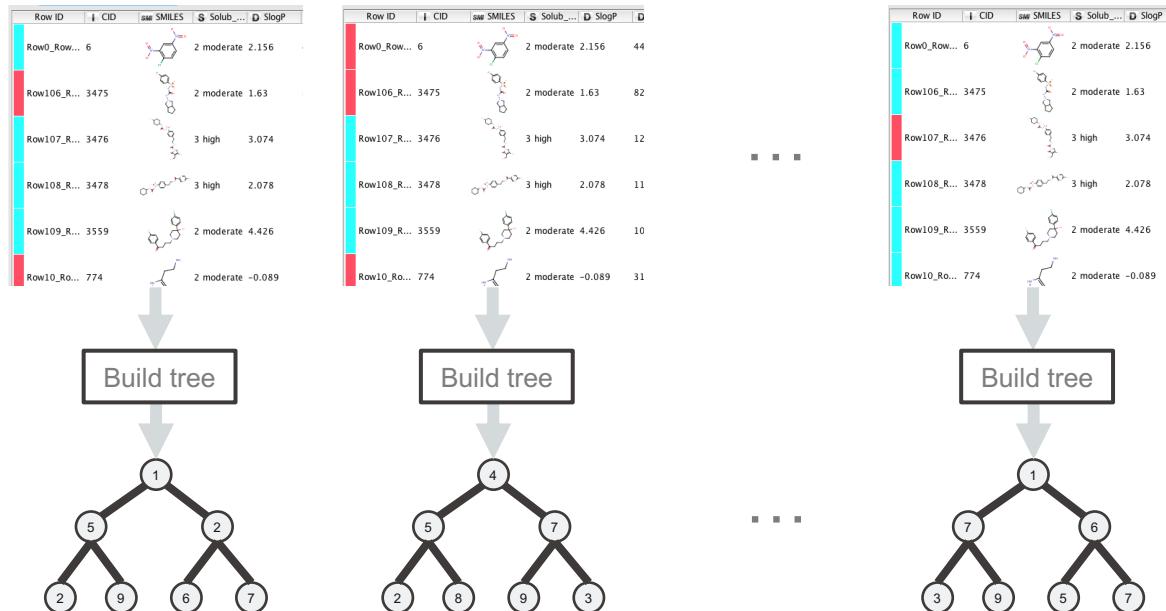
- General idea: take advantage of the “wisdom of the crowd”
- Ensemble models: Combining predictions from many predictors, e.g. decision trees
- Leads to a more accurate and robust model
- Model is difficult to interpret
  - There are multiple trees in the model



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



# Example for Bagging

Full training set

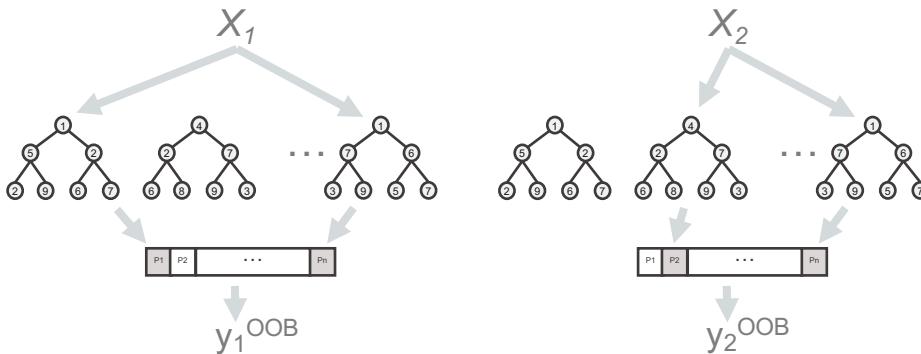
RowID	$x_1$	$x_2$	$y$
Row_1	2	6	Class 1
Row_2	4	1	Class 2
Row_3	9	3	Class 2
Row_4	2	7	Class 1
Row_5	8	1	Class 2
Row_6	2	6	Class 1
Row_7	5	2	Class 2

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

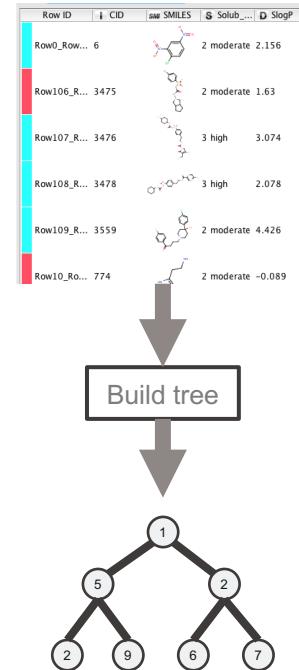
File Hilite Navigation View

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	IH	1	0	0	1	33
Row3_Row2	U	1	0	0	1	37
Row4_Row4	IH	0.528	0.472	0	0.528	36
Row5_Row5	JK	0.976	0.024	0	0.976	41
Row6_Row6	IL	0.848	0.152	0	0.848	33
Row7_Row7	IA	0.833	0.167	0	0.833	36
Row9_Row8	A	0.667	0.333	0	0.667	30
Row11_Row...	V	0.138	0.862	1	0.862	29
Row13_Row...	A	0.974	0.026	0	0.974	39
Row14_Row...	IT	0.917	0.083	0	0.917	36
Row15_Row...	A	0.387	0.613	1	0.613	31
Row18_Row...	T	0.974	0.026	0	0.974	39
Row19_Row...	A	1	0	0	1	38
Row21_Row...	L	0.971	0.029	0	0.971	34
Row22_Row...	O	0.03	0.97	1	0.97	33
Row23_Row...	Z	0.854	0.146	0	0.854	41
Row25_Row...	A	0.973	0.027	0	0.973	37
Row26_Row...	IE	0.886	0.114	0	0.886	35
Row27_Row...	Y	0.912	0.088	0	0.912	34
Row28_Row...	IT	0.976	0.024	0	0.976	42
Row29_Row...	IO	1	0	0	1	42
Row30_Row...	II	1	0	0	1	40
Row32_Row...	IH	0.914	0.086	0	0.914	35
Row33_Row...	A	0.875	0.125	0	0.875	32

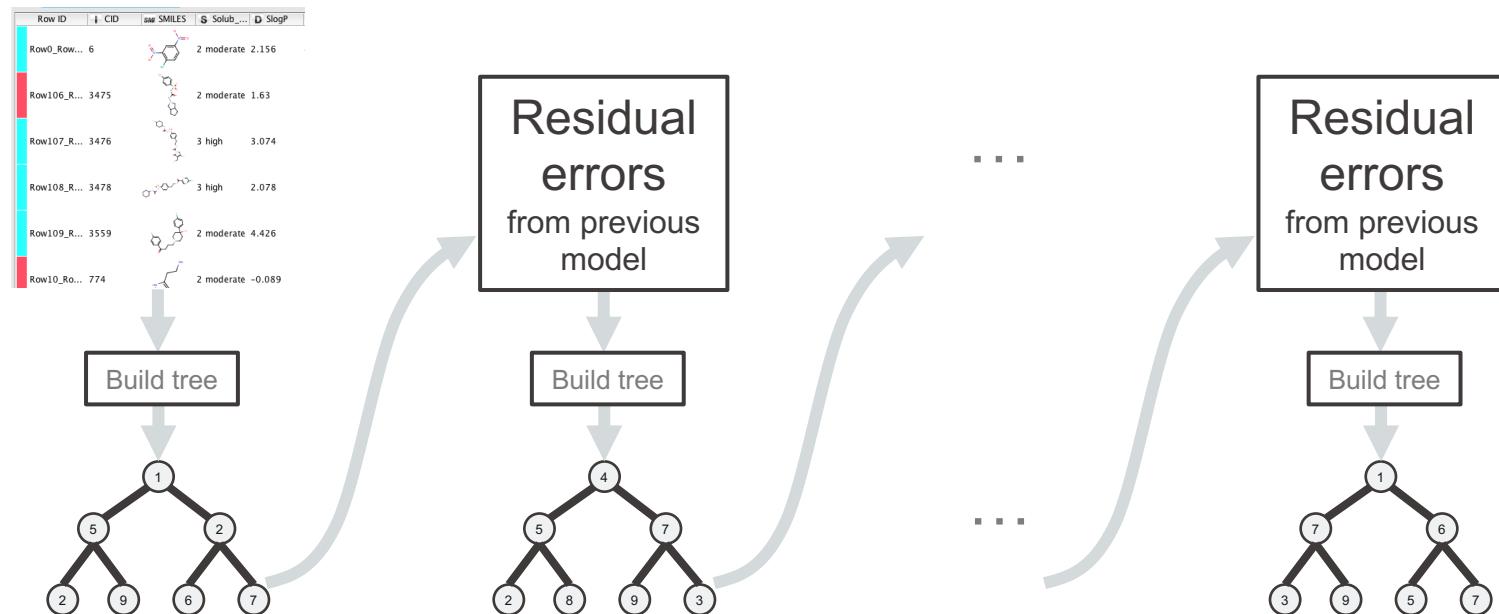
# Random Forest

- 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



# 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



# Boosting - Idea

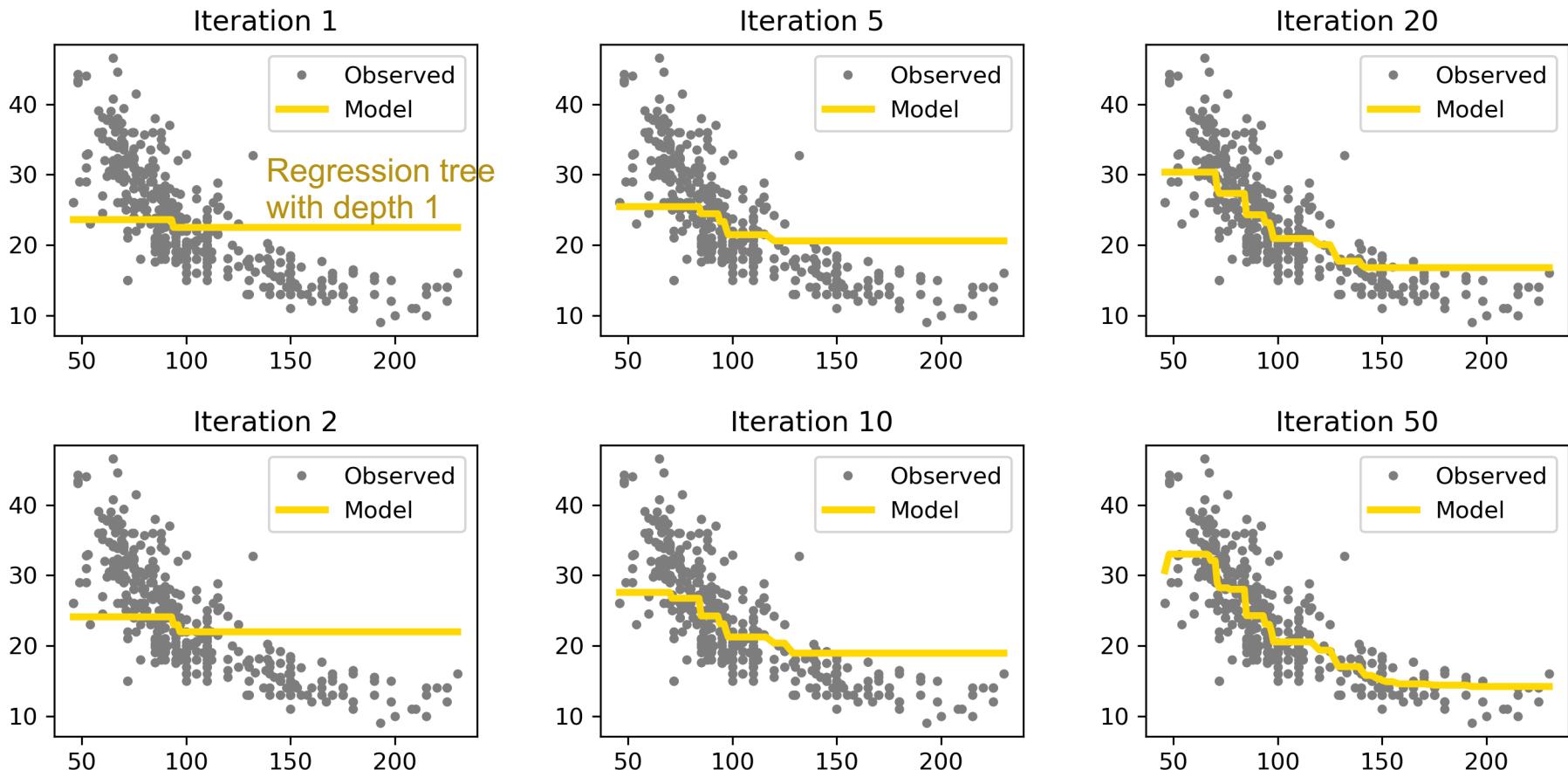
---

- **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  $\gamma_m$  is chosen to minimize the loss function
  - Loss function: quantifies the difference between model predictions and data

# Gradient Boosting Example – Regression



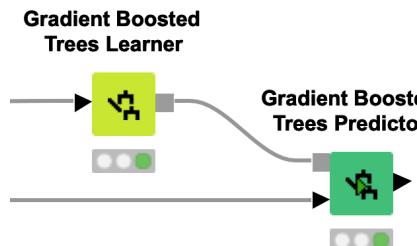
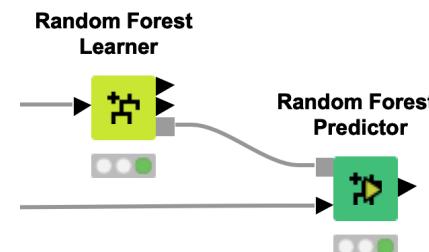
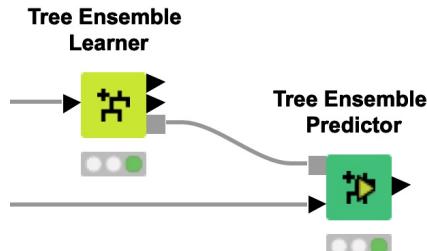
# Gradient Boosted Trees

---

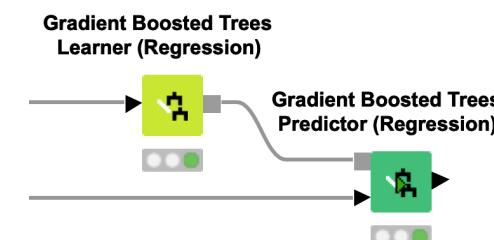
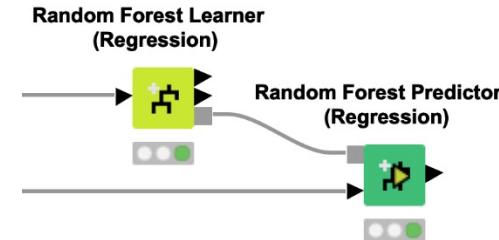
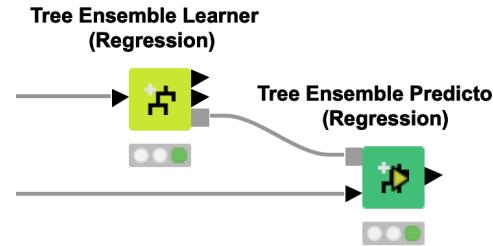
- Can be used for classification and regression
- Large number of iterations – prone to overfitting
  - ~100 iterations are sufficient
- Can introduce randomness in choice of data subsets (“stochastic gradient boosting”) and choice of input features

# Ensemble Tree Nodes in KNIME Analytics Platform

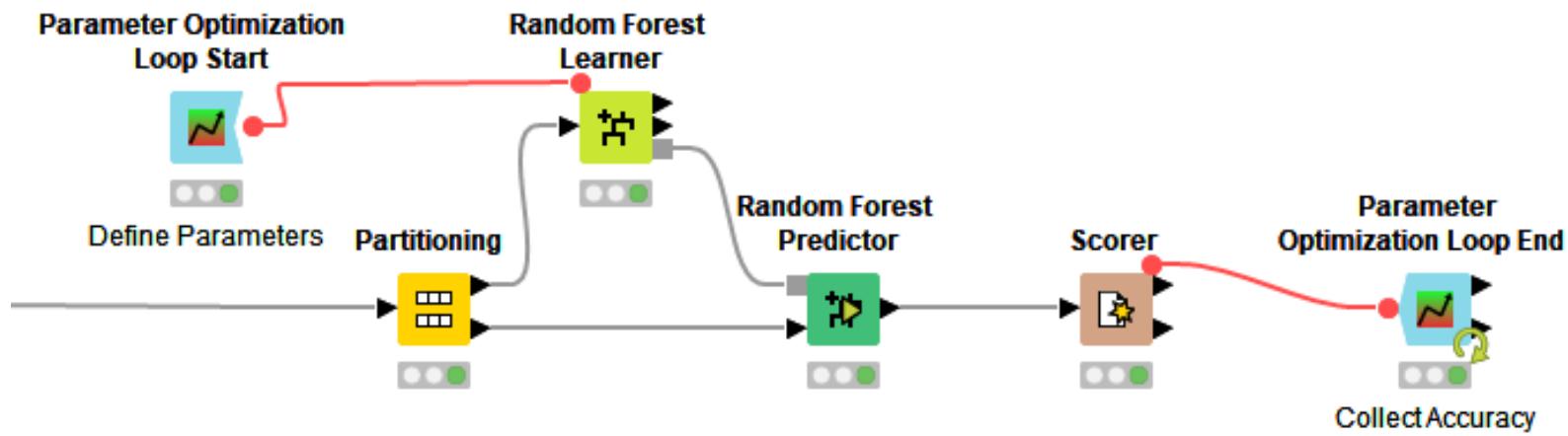
## Classification Problems



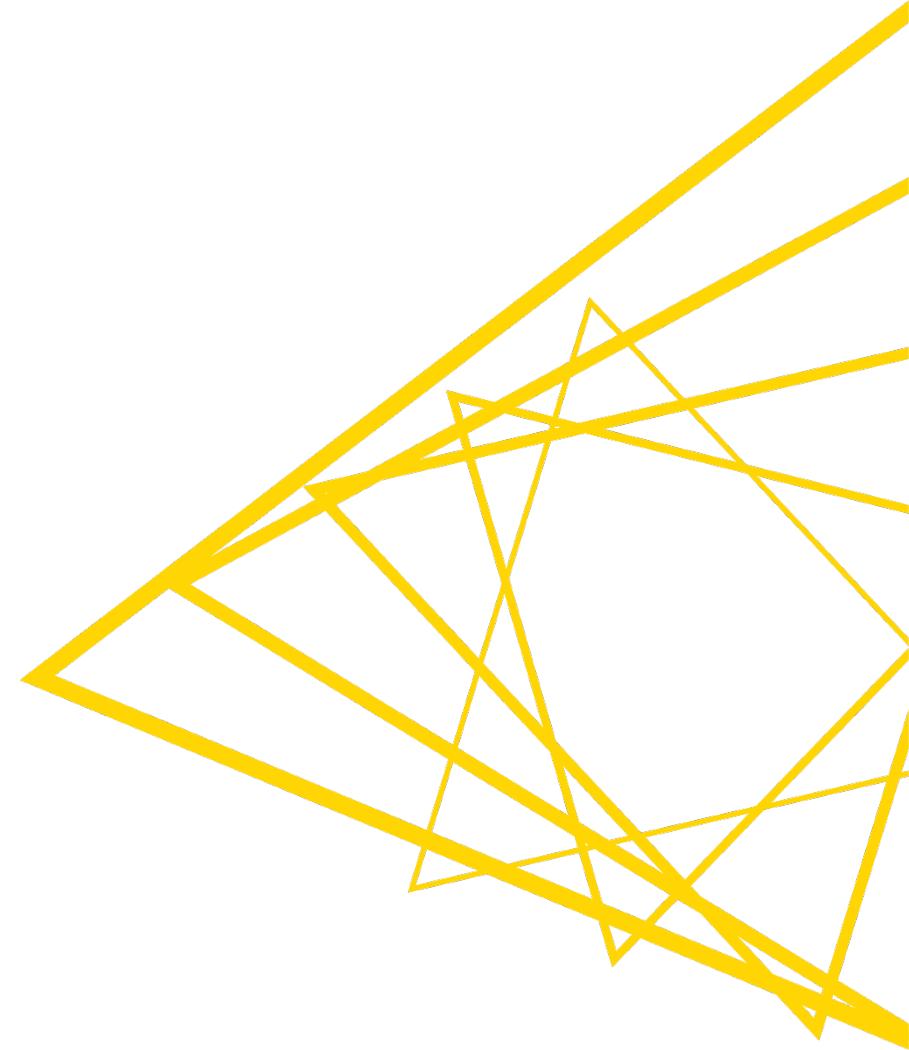
## Regression Problems



# Parameter Optimization



# Logistic Regression



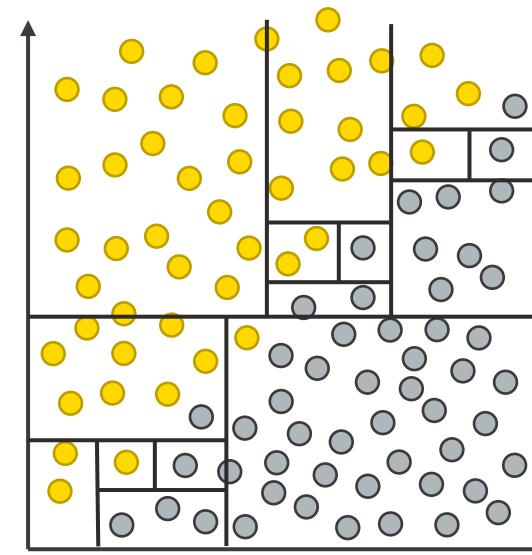
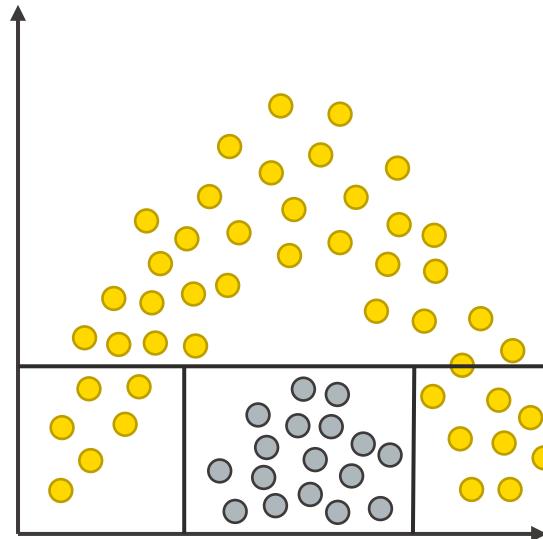
# What is a Logistic Regression (algorithm)?

- Another algorithm to train a classification model

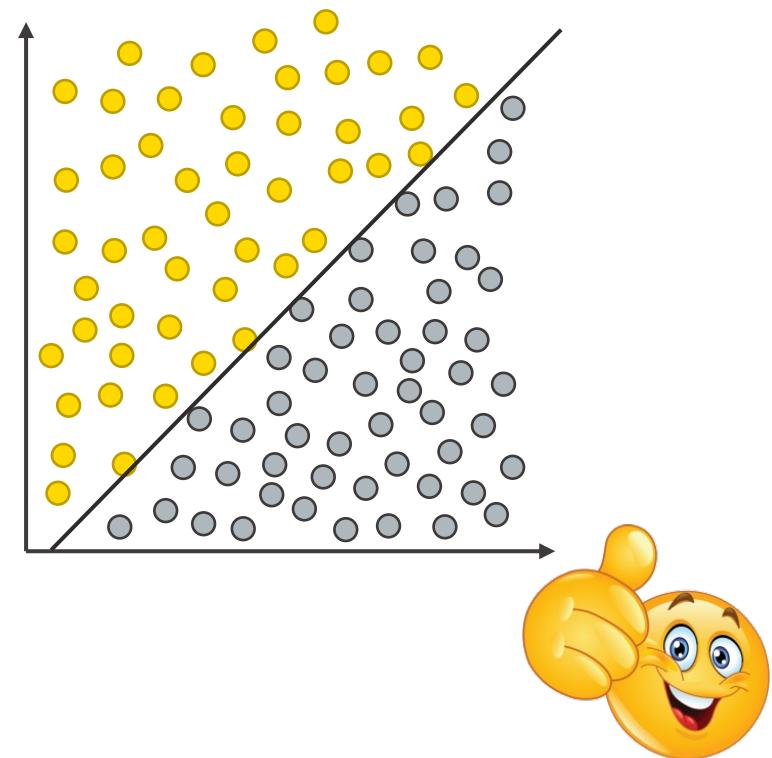
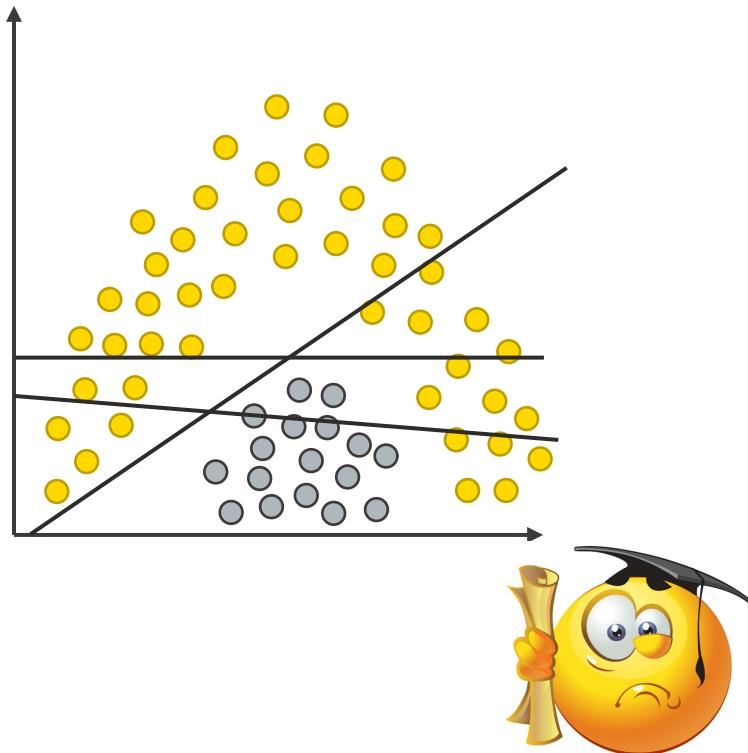


I know already the decision tree algorithm and tree ensembles. Why do I need another one?

# 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 $y \in (-\infty, \infty) / [a, b]$	<b>Nominal</b> $y \in \{0, 1, 2, 3\} / \{\text{red}, \text{white}\}$
Functional relationship between features and...	... target value $y$ $y = f(x_1, \dots, x_n, \beta_0, \dots, \beta_n)$ $y = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$	... <b>class probability <math>P(y = \text{class } i)</math></b> $P(y = c_i) = f(x_1, \dots, x_n, \beta_0, \dots, \beta_n)$

**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

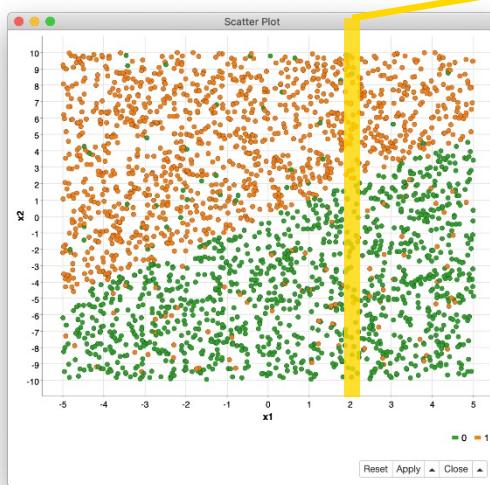
$$P(y = 1) = p_1 \Rightarrow P(y = 0) = 1 - p_1$$

→ It's sufficient to model the probability for one class

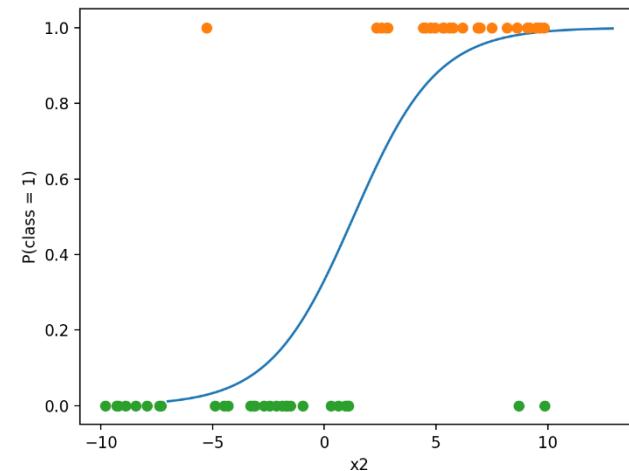
# Let's Find Out How Binary Logistic Regression Works!

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

Feature space



Probability function given  $x_1 = 2$



# More General: Binary Logistic Regression

---

- Model:

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

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

- 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,  $\mathbf{X} = (1, x_1, \dots, x_n)$
  - The training data set has  $m$  observations  $(y_i, x_1^i, \dots, x_n^i)$

# How can we Find the Best Coefficients $\beta$ ?

Maximize the product of the probabilities  $\rightarrow$  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?

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

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

# Max Likelihood and Log Likelihood Functions

---

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

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

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

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

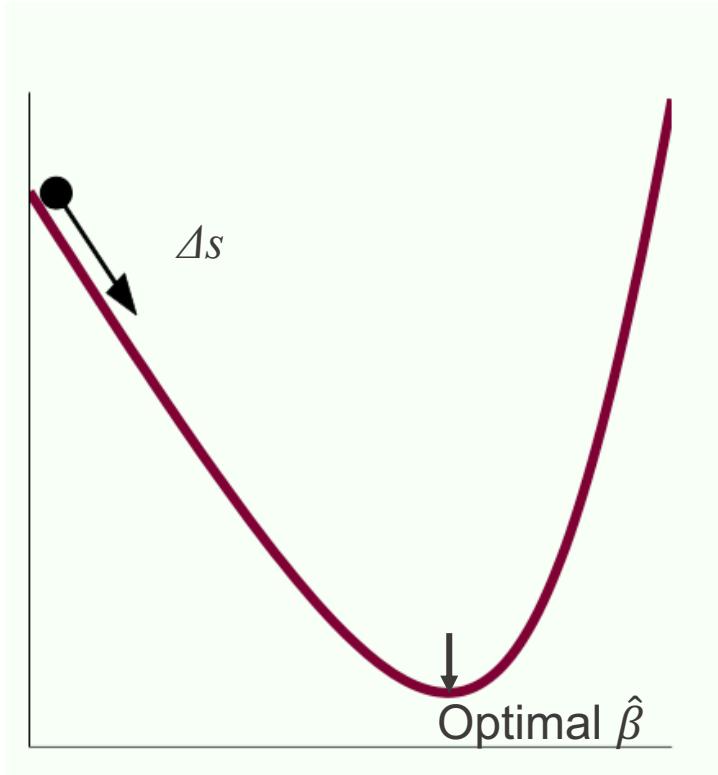
# How can we find this Coefficients?

---

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

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

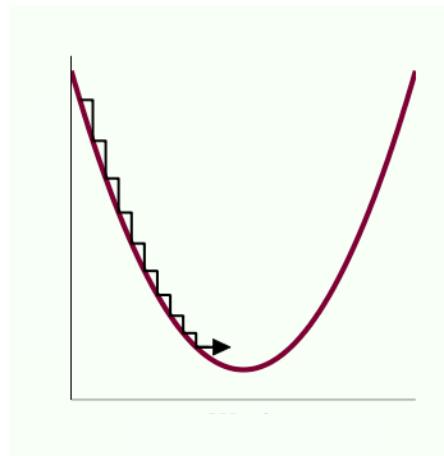


- Example:  $\min -LL(\beta) := f(\beta)$
- Start from an arbitrary point
- Move towards the minimum
- With step size  $\Delta s$
- If  $f(\beta)$  is strictly convex
  - ➔ Only one global minimum exists
- Z normalization of the input data lead to better convergence

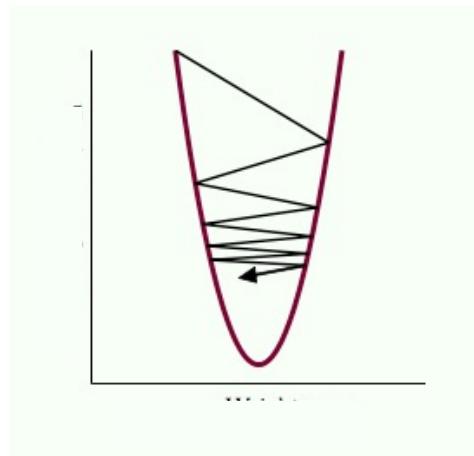
# Learning Rate / Step Length $\Delta s$

---

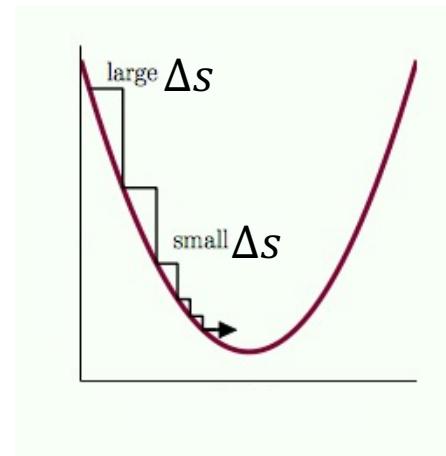
$\Delta s$  too small



$\Delta s$  too large



Just right



# Learning Rate $\Delta s$

---

- 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  $\alpha$

- 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  $\sigma = \frac{\sqrt{2}}{\lambda}$

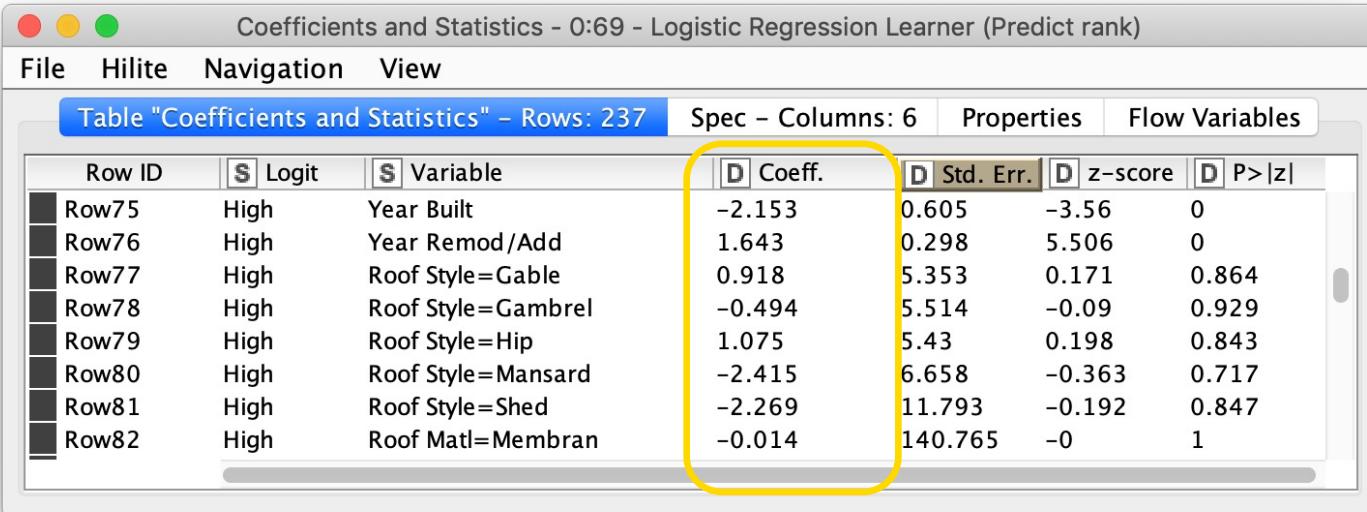
$$l(\hat{\beta}; y, X) := -LL(\hat{\beta}; y, X) + \lambda \|\hat{\beta}\|_1$$

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

# Impact of Regularization



# Interpretation of the Coefficients



The screenshot shows a KNIME node titled "Coefficients and Statistics - 0:69 - Logistic Regression Learner (Predict rank)". The table has the following columns:

Row ID	Logit	Variable	Coeff.	Std. Err.	z-score	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)						
File	Hilite	Navigation	View	Spec - Columns: 6	Properties	Flow Variables
Table "Coefficients and Statistics" – Rows: 237			Coeff.	Std. Err.	z-score	P> z
Row ID	Logit	Variable	D	D	D	D
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 <  $\alpha$ : input feature has a significant impact on the dependent variable.

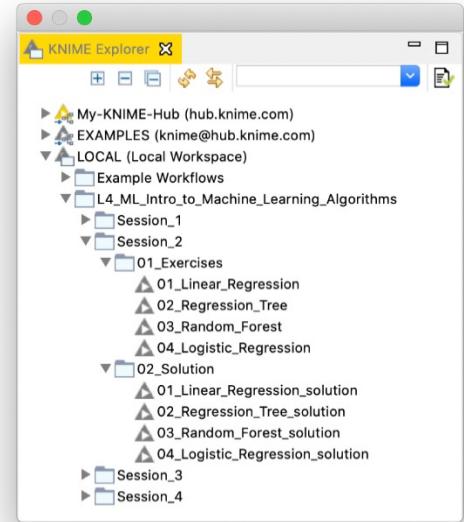
# Summary Logistic Regression

---

- 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.
- The p-value shows us whether an independent variable is significant

# Exercises

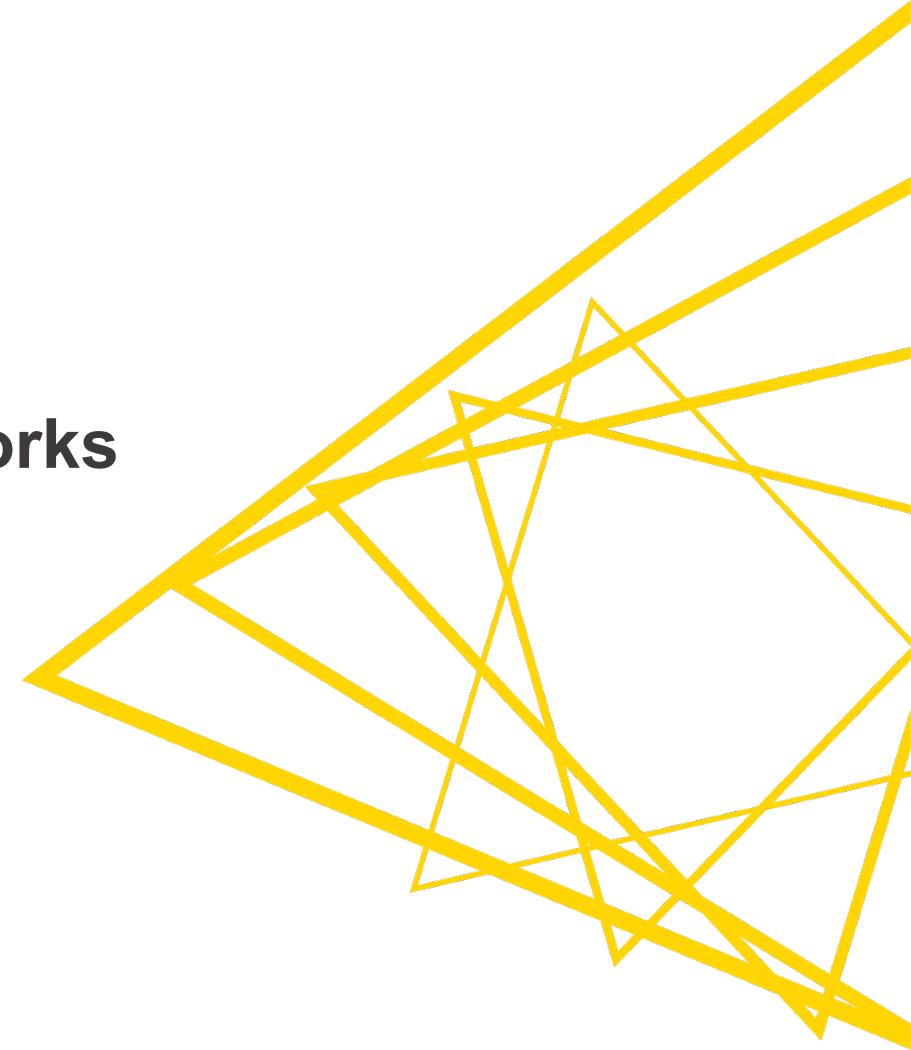
- Regression Exercises:
  - Goal: Predicting the house price
  - 01\_Linear\_Regression
  - 02\_Regression\_Tree
- Classification Exercises:
  - Goal: Predicting the house condition (high /low)
  - 03\_Radom\_Forest (with optional exercise to build a parameter optimization loop)
  - 04\_Logistic\_Regression



# Session 3: Neural Networks and Recommendation Engines

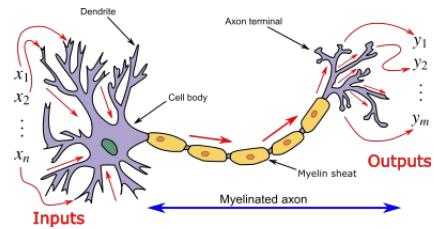


# Artificial Neurons and Networks

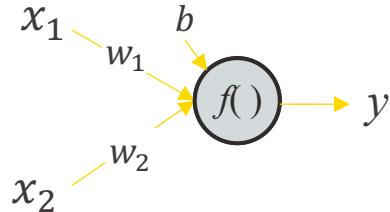


# Biological vs. Artificial

## Biological Neuron



## Artificial Neuron (Perceptron)

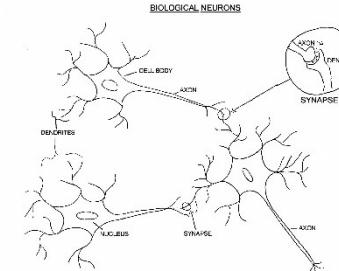


$$y = f(x_1w_1 + x_2w_2 + b)$$

$$b = w_0$$

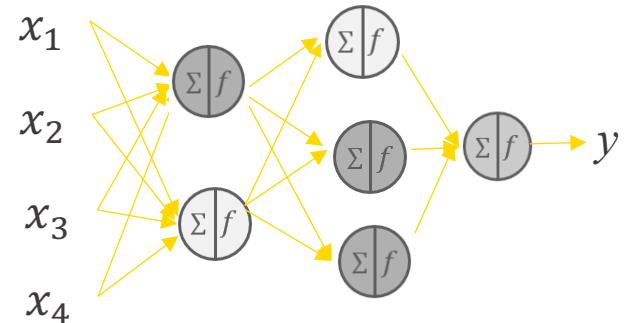
$$y = f\left(\sum_{i=0}^n x_iw_i\right)$$

## Biological Neural Networks

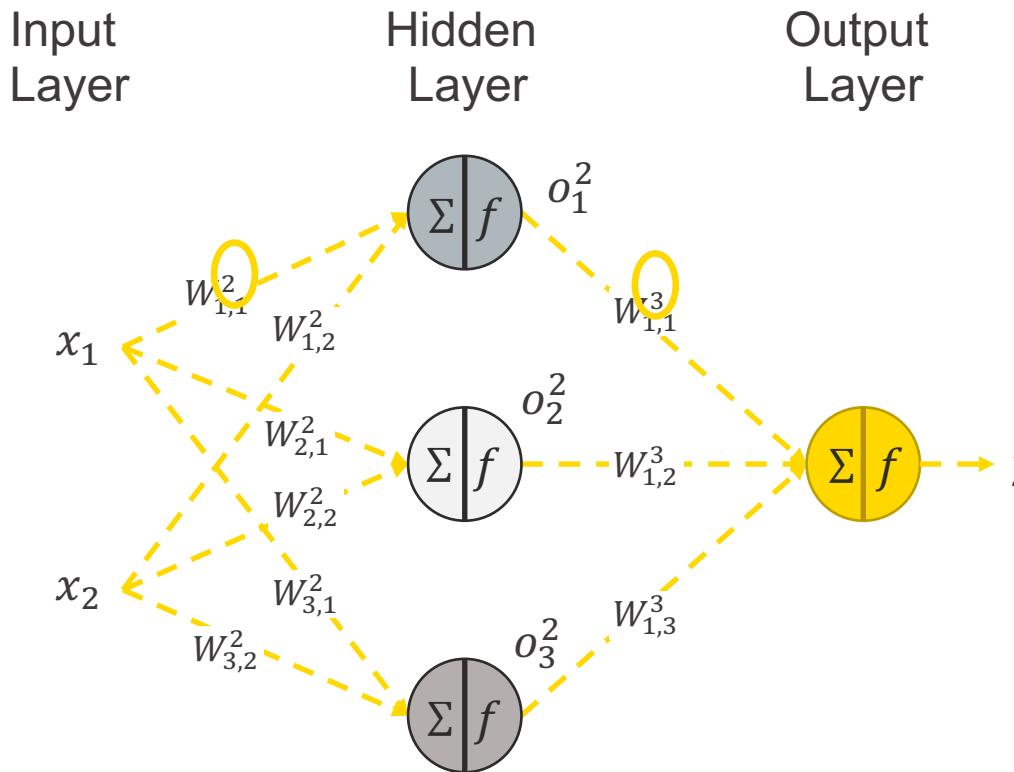


## Artificial Neural Networks

(Multilayer Perceptron, MLP)



# Architecture / Topology



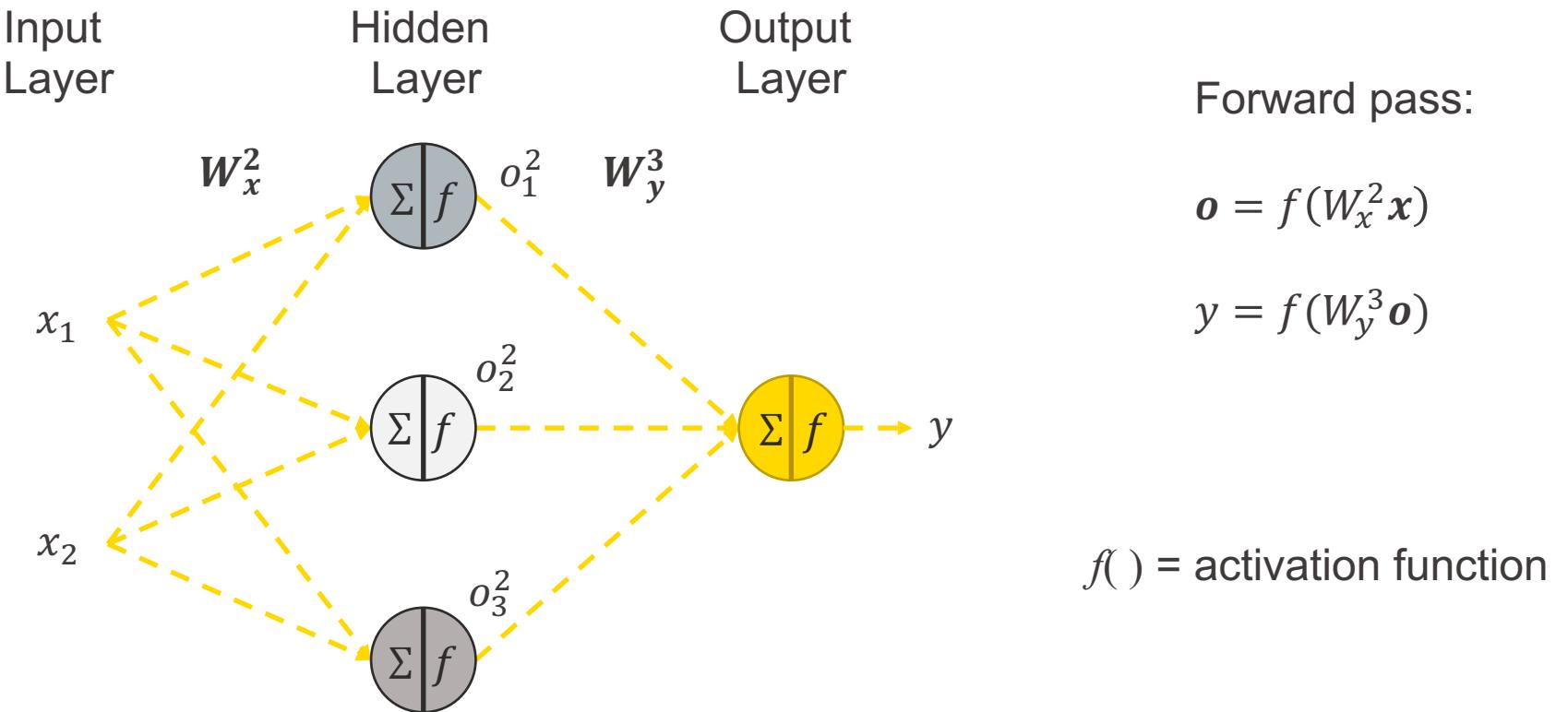
Forward pass:

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

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

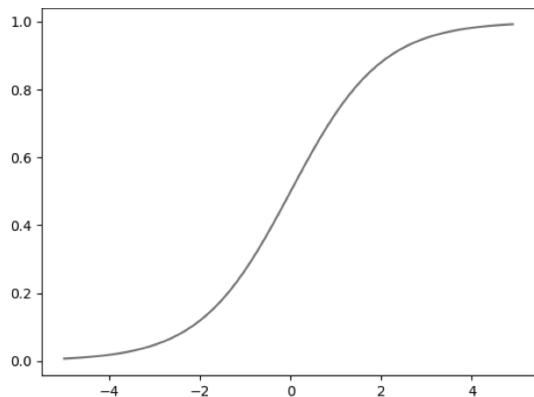
fully connected  
feed forward

# Architecture / Topology

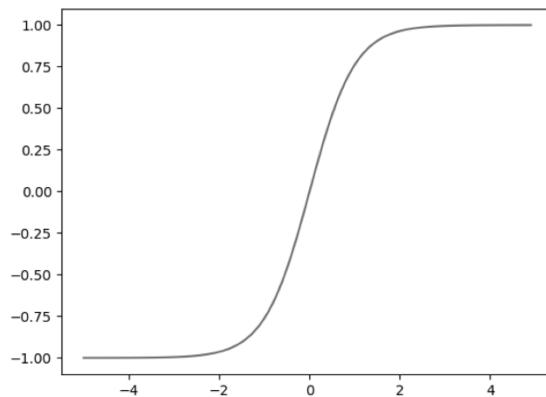


# Frequently used activation functions

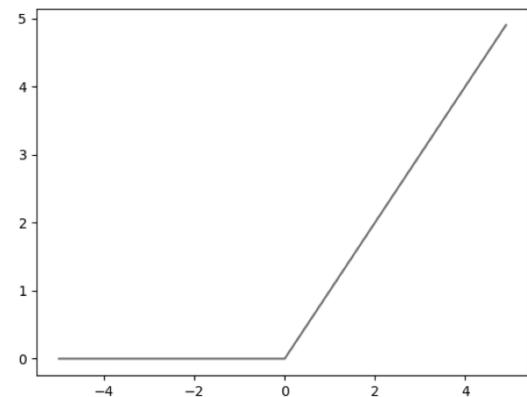
Sigmoid



Tanh



Rectified Linear Unit (ReLU)

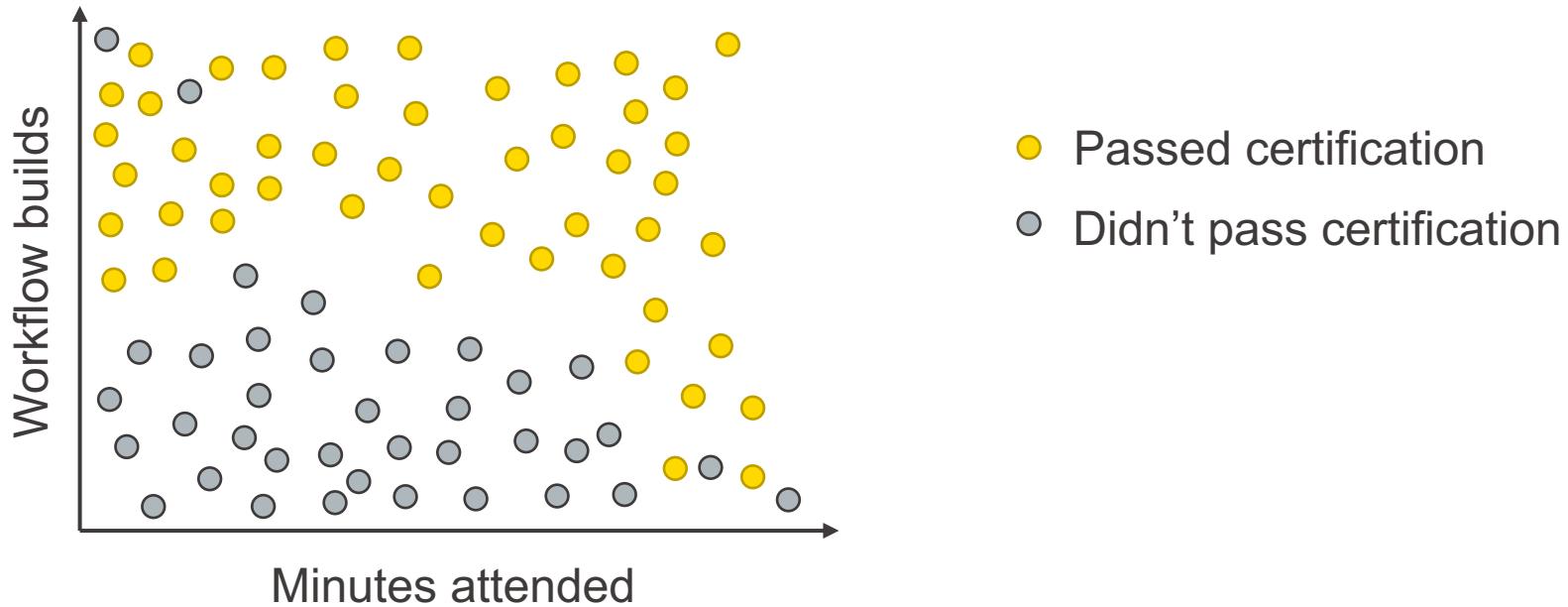


$$f(a) = \frac{1}{1 + e^{-ha}}$$

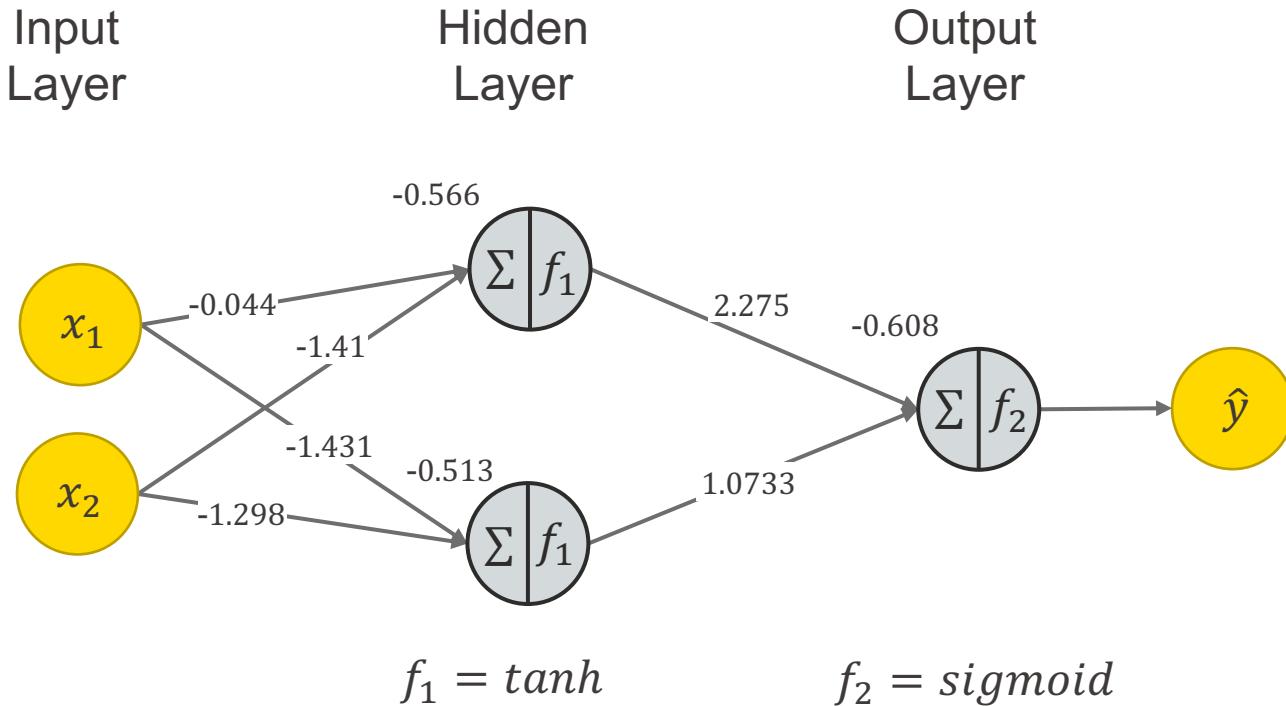
$$f(a) = \frac{e^{2ha} - 1}{e^{2ha} + 1}$$

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

# Example: Passing the KNIME L1-Certification



# Example: Passing the KNIME L1-Certification



Input features:

$x_1$  = minutes attended

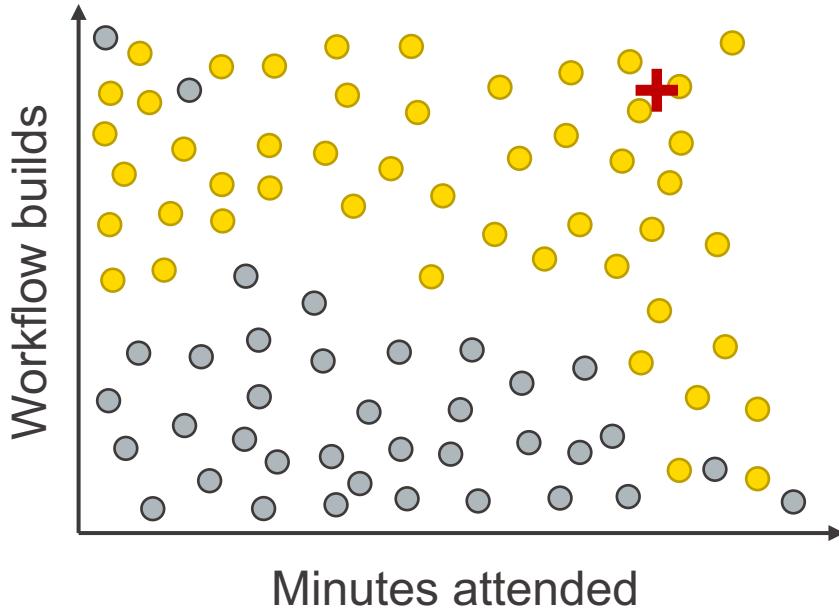
$x_2$  = workflows build

Output:

$\hat{y}$  = Probability that a person passed

$\hat{y} \geq 0.5 \Rightarrow \text{Passed}$  and  $\hat{y} < 0.5 \Rightarrow \text{Failed}$

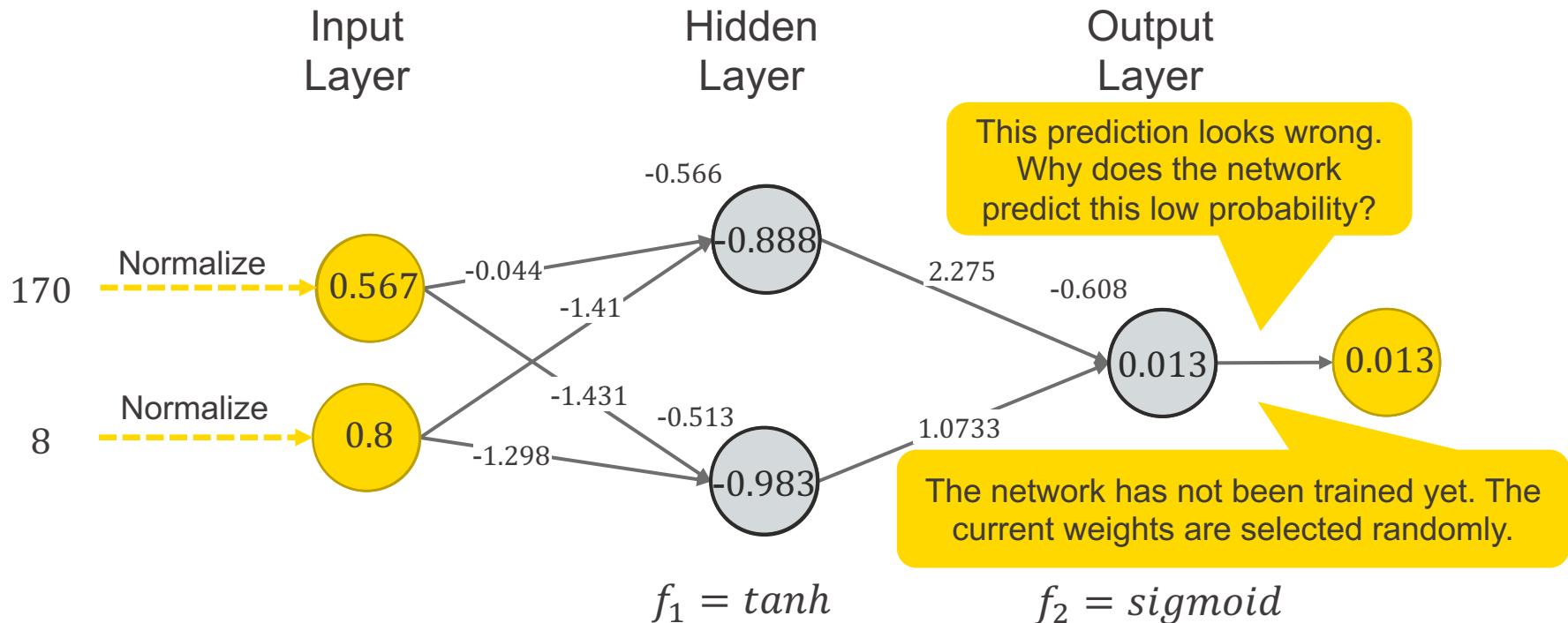
# Example: Passing the KNIME L1-Certification



- Passed certification
- Didn't pass certification
- ✚ New sample

$x_1$  = minutes attended = 170  
 $x_2$  = workflows build = 8

# Example: Passing the KNIME L1-Certification



Input features:

$x_1$  = minutes attended

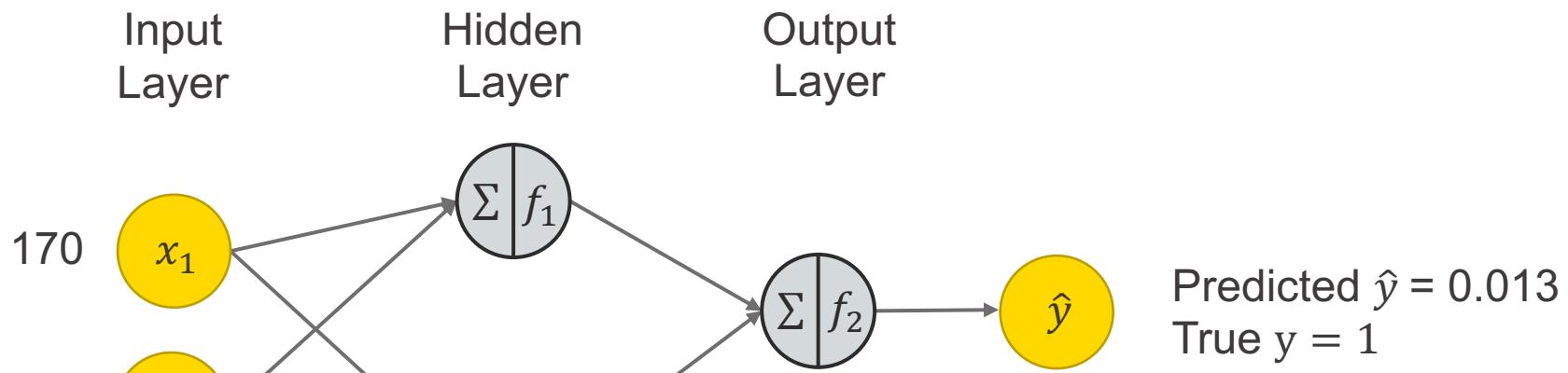
$x_2$  = workflows build

Output:

$\hat{y}$  = Probability that a person passed

$\hat{y} \geq 0.5 \Rightarrow \text{Passed}$  and  $\hat{y} < 0.5 \Rightarrow \text{Failed}$

# Training a Neural Network = Finding Good Weights



$$J(W) = \sum \mathcal{L}(\hat{y}(x_1, x_2, W), y)$$

Binary cross entropy  
$$\mathcal{L} = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

# Learning Rule from Gradient Descent

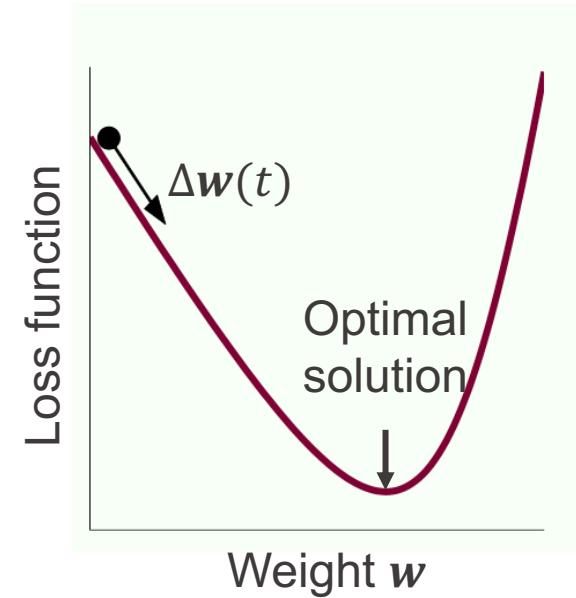
- Adjust the weight for the next step by the adjustment term  $\Delta w(t)$

$$w(t + 1) = w(t) + \eta \Delta w(t)$$

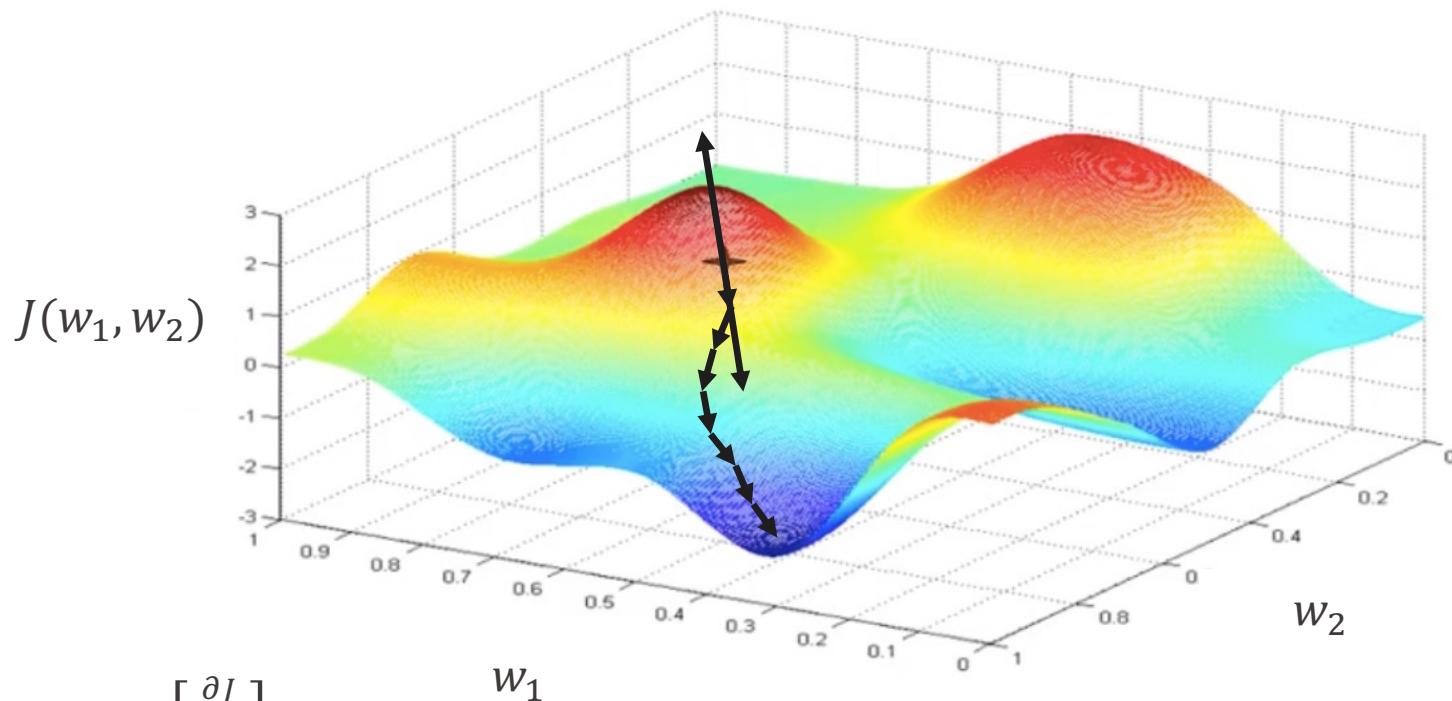
Updated weight in next step  $t + 1$

Weight in current step  $t$

Weight adjustment based on gradient



# Idea Behind Gradient Descent

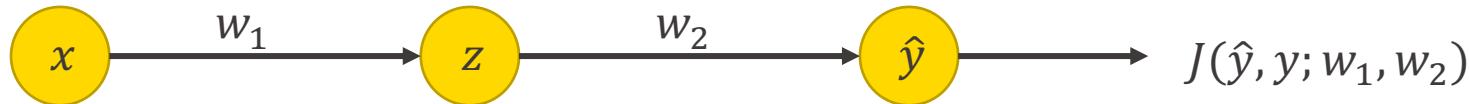


$$\nabla_W J(x, W) = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \end{bmatrix}$$

# Backpropagation

- Efficient way to calculate the gradient during optimization

Forward pass



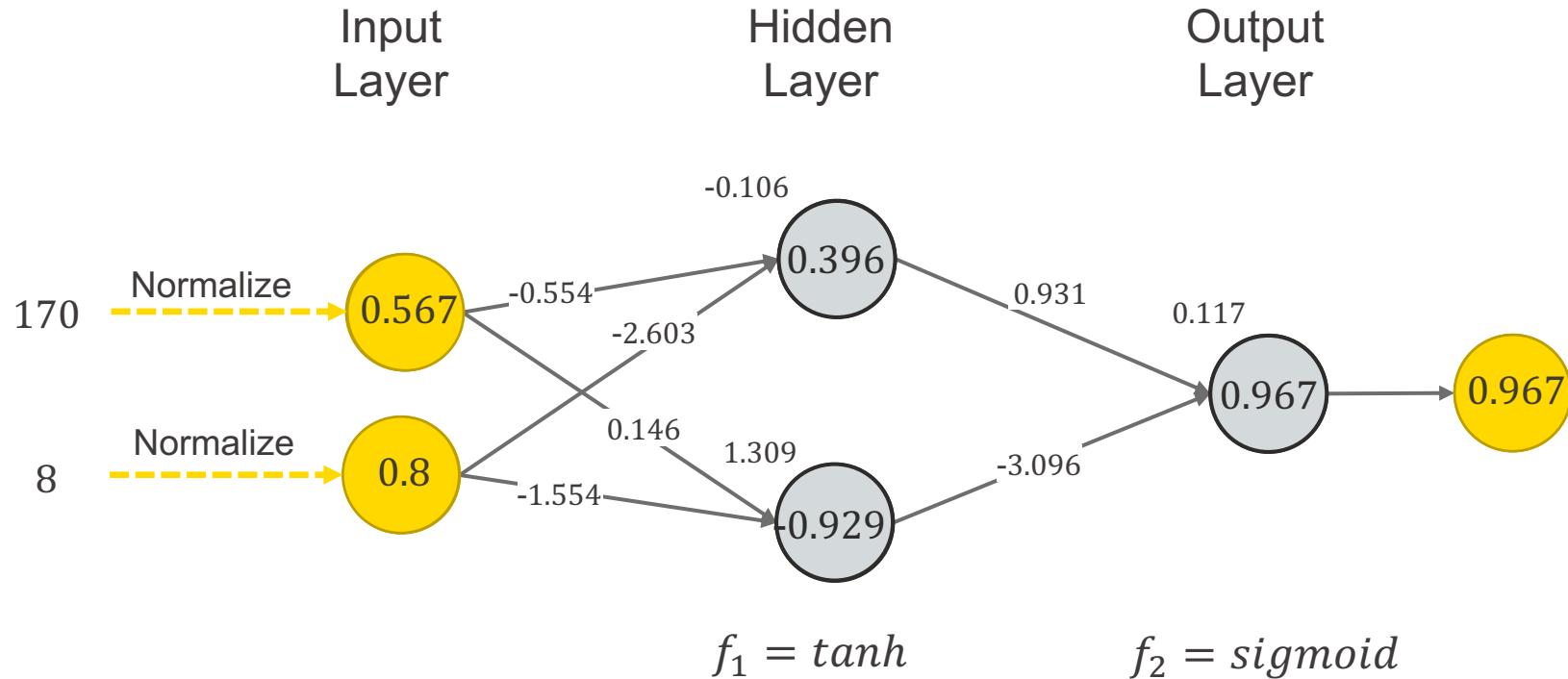
Backward pass



$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2}$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial w_1}$$

# Example: Passing the KNIME L1 Certification



Input features:

$x_1$  = minutes attended

$x_2$  = workflows build

Output:

$y$  = Probability that a person passed

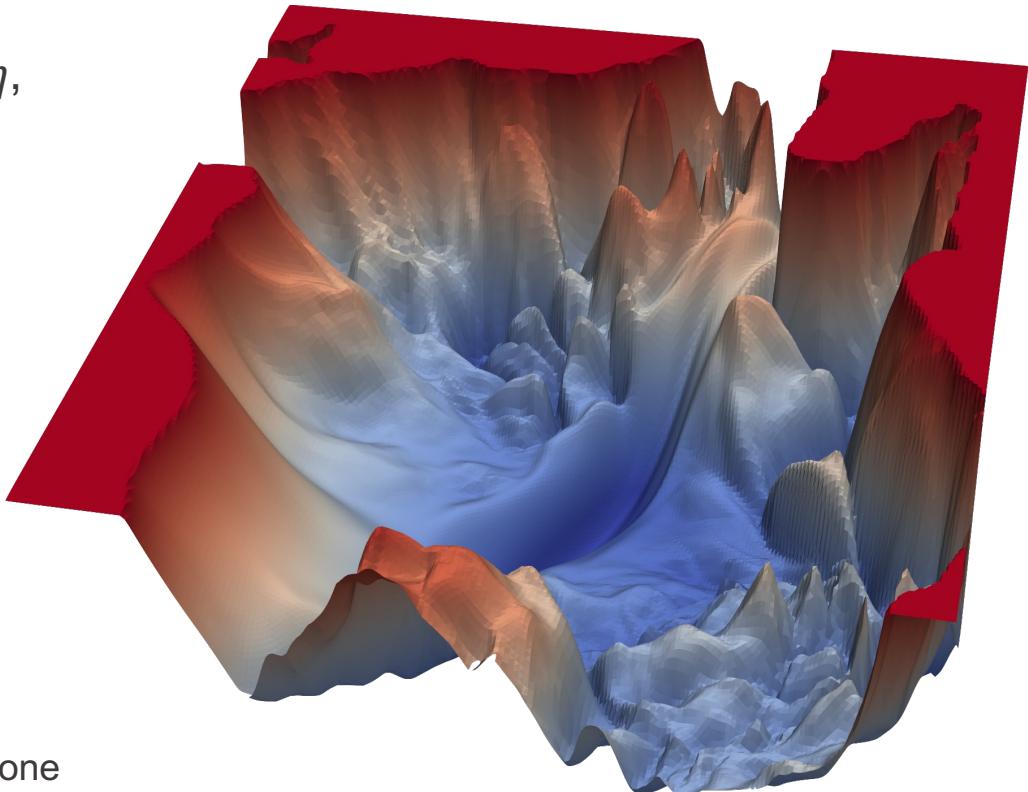
$y \geq 0.5 \Rightarrow \text{Passed}$  and  $y < 0.5 \Rightarrow \text{Failed}$

# Loss Landscape of a Real Neural Network

- A good choice for the step size  $\eta$ , aka learning rate, is important

$$W \leftarrow W - \eta \nabla_W J(x, W)$$

- Many different optimizers with adaptive learning rates are available
  - Adam, Adadelta, Adagrad, ect
- Other important settings
  - Batch size, aka number of samples for one update
  - Number of epochs, aka how often each sample is used



Source: <https://www.cs.umd.edu/~tomg/projects/landscapes/>

# Optimizers in Keras

Optimizer	How it works	Strengths	Weaknesses	When to use
SGD with momentum	Use the previous gradient to accelerate convergence	-Reduces oscillation near maxima	-Const learning rate	
NAG (Nesterov accelerated gradient)	Use the current gradient to predict gradient	-Increased responsiveness	-Additional hyperparameter	RNN
Adagrad	Updating by cumulating sum of sq gradients from past	-Different learning parameters for different features	-Computationally expensive -Shrinking learning rate	Sparse data (e.g. text)
Adadelta	Modified Adagrad with decaying average of sq gradients from past	-Learning rate not dramatically shrinking like Adagrad	-Computationally expensive	Sparse data (e.g. text)
RMSProp	Modified Adagrad with sq gradients added very slowly	-Learning rate not dramatically shrinking like Adagrad		
Adam (Adaptive Moment Estimation)	RMSProp plus decaying average of gradients from past	-Fast convergence	-Computationally expensive	

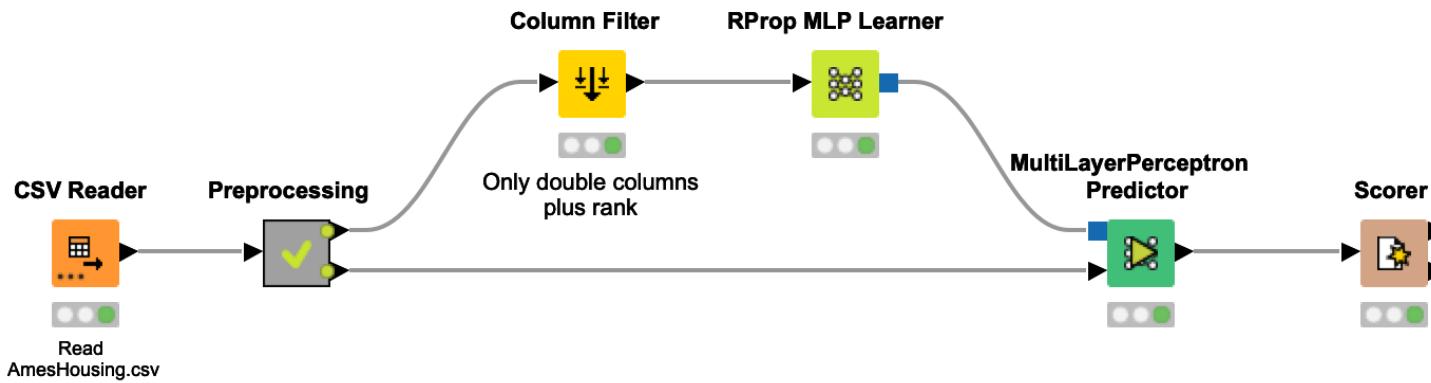
# Which Activation Functions? Which Loss Functions?

- Depends on the problem you are working on

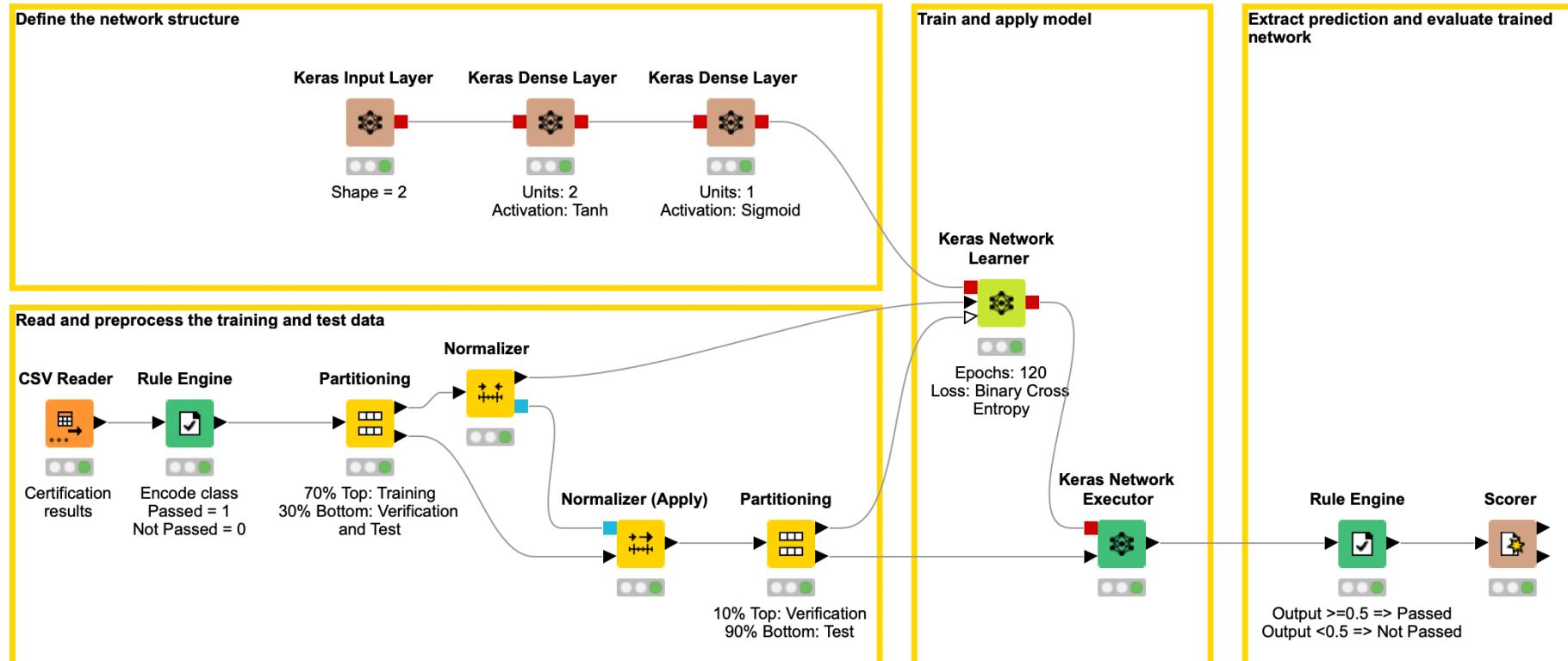
Problems		Activation Functions					Loss Functions					
		Hidden Layers			Output Layer		Binary CE	Hinge	Categorical CE	MSE	MSLE	MAE
		Sigmoid	Tanh	ReLU	Sigmoid	Tanh						
Classification	Binary classification (0 vs 1)	✓	✓	✓	✓		✓					
	Binary classification (-1 vs 1)	✓	✓	✓		✓			✓			
	Multi-class classification	✓	✓	✓				✓		✓		
Regression	Regression	✓	✓	✓	△	△	✓	△		✓		
	Regression (wide range)	✓	✓	✓			✓				✓	
	Regression (possible outliers)	✓	✓	✓			✓				✓	

# Codeless Deep Learning with KNIME Analytics Platform

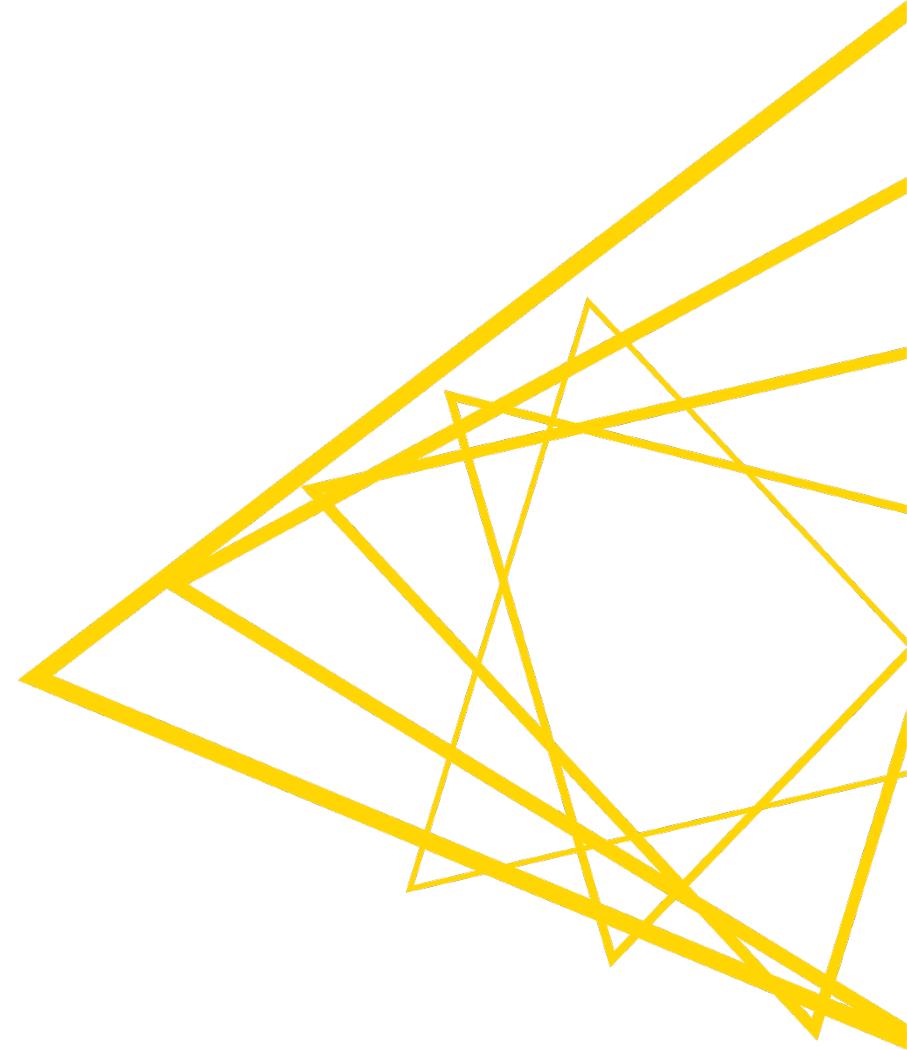
- Simple option for feed forward neural networks with activation function sigmoid



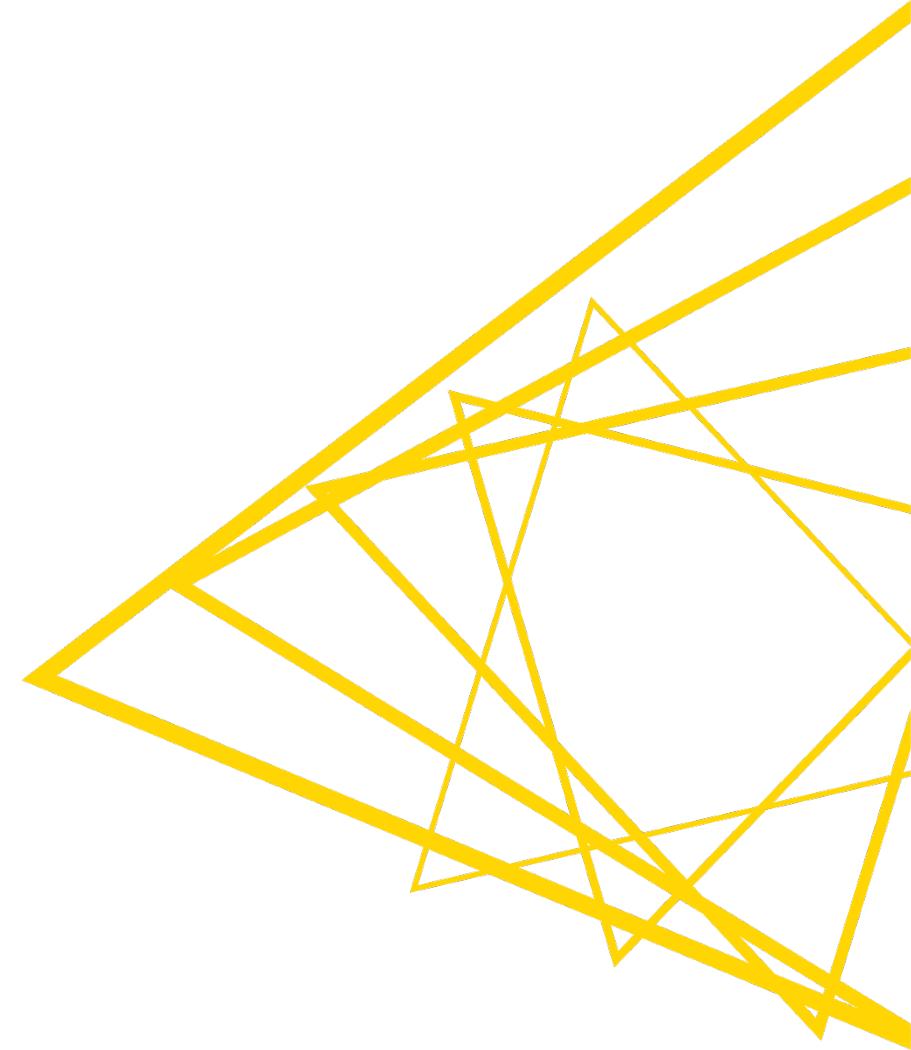
# Codeless Deep Learning with KNIME Analytics Platform



# Deep Learning



# Recurrent Neural Networks



# What are Recurrent Neural Networks?

---

- Recurrent Neural Network (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

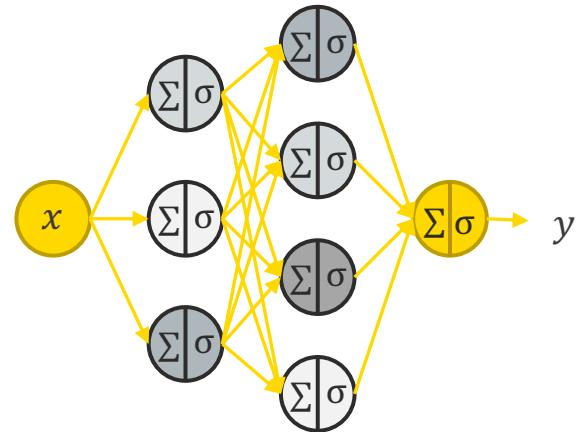
# Why do we need RNNs for Sequential Data?

- Goal: Translation network from German to English

*“Ich mag Schokolade”*  
=> *“I like chocolate”*

- One option: Use feed forward network to translate word by word
- But what happens with this question?

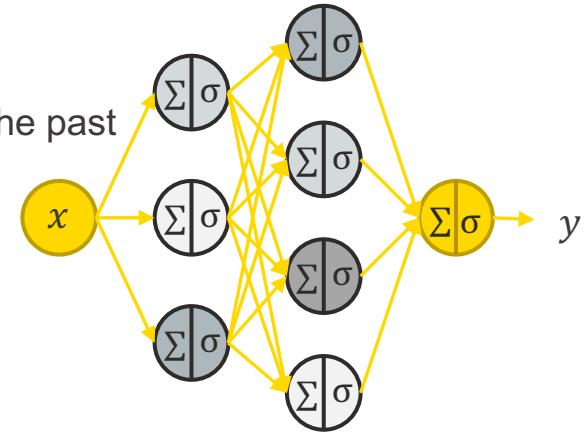
*“Mag ich Schokolade?”*  
=> *“Do I like chocolate?”*



Input x	Output y
Ich	I
mag	like
Schokolade	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?

---

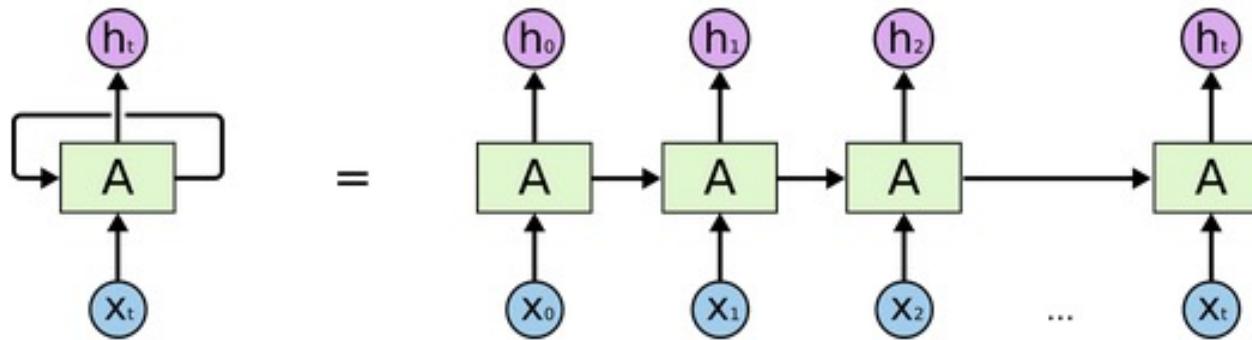
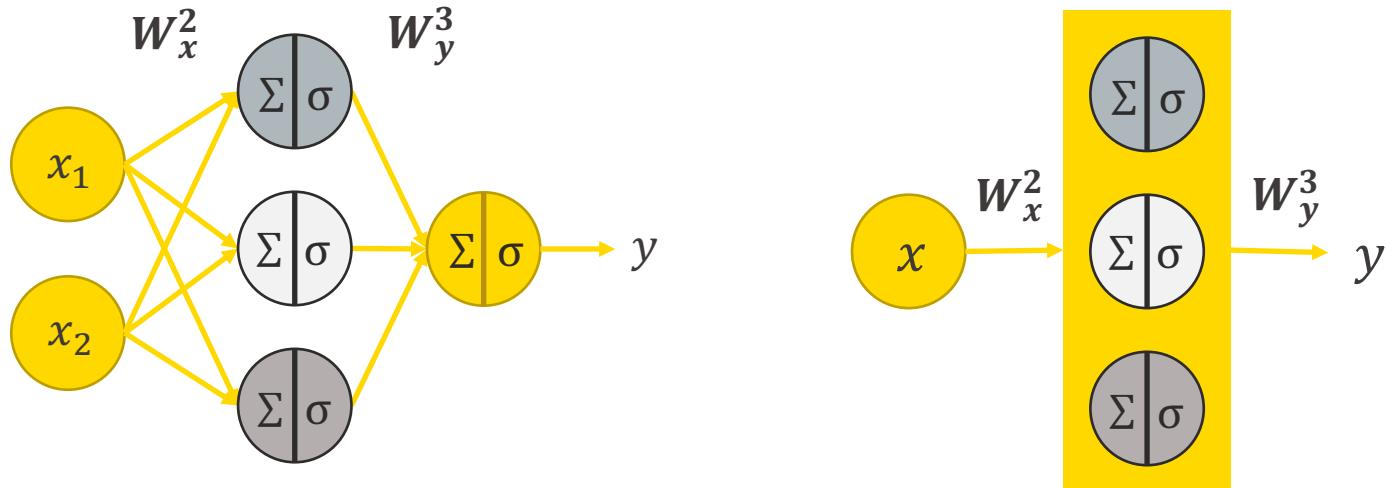


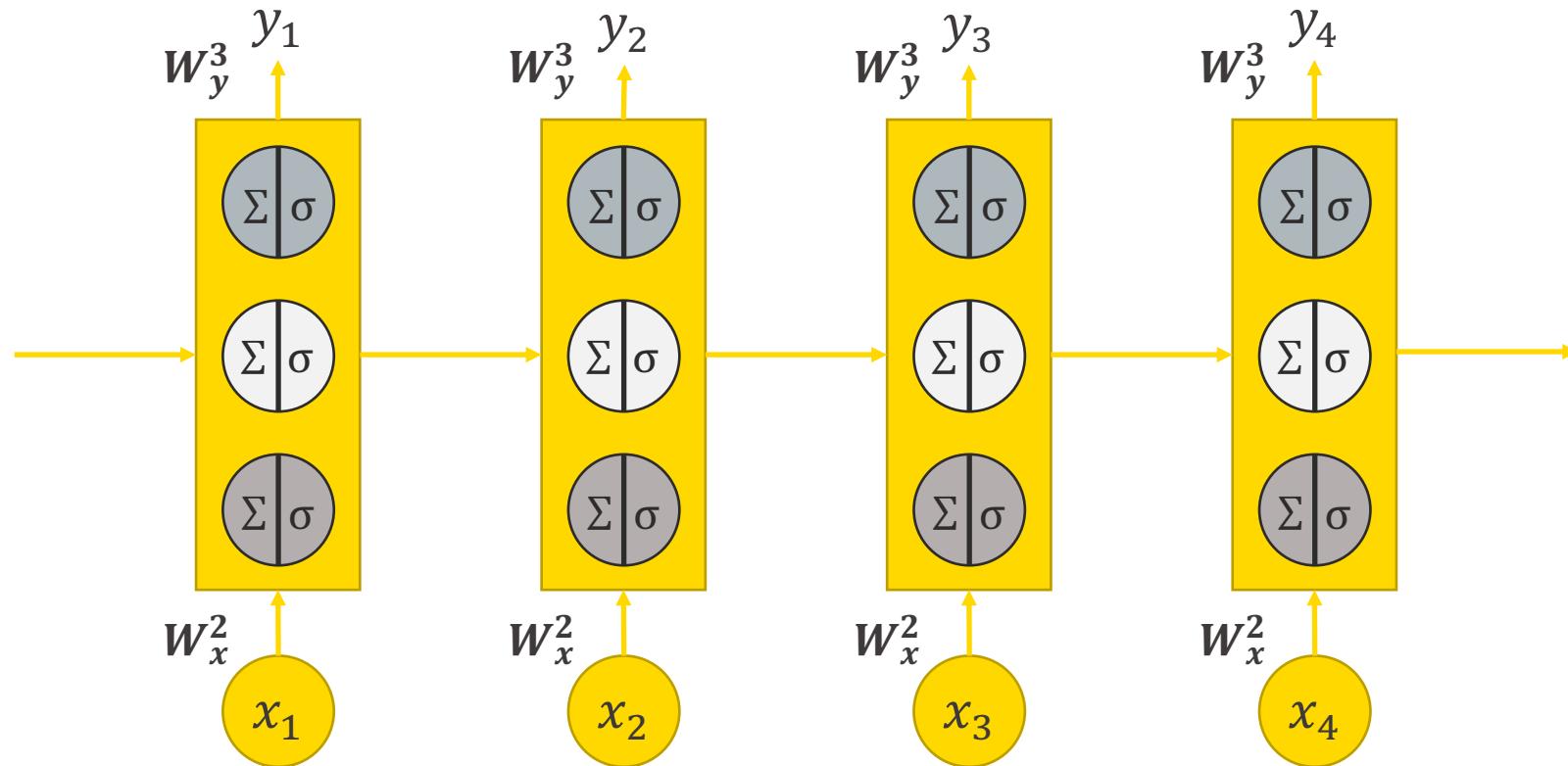
Image Source: Christopher Olah, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# From Feed Forward to Recurrent Neural Networks

---



# From Feed Forward to Recurrent Neural Networks



# Simple RNN unit

---

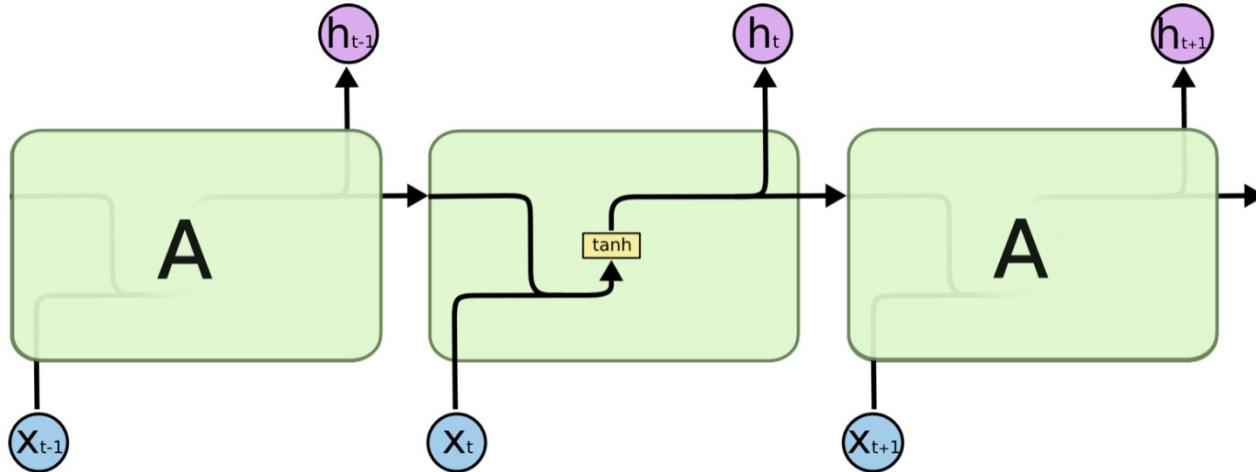


Image Source: Christopher Olah, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# 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

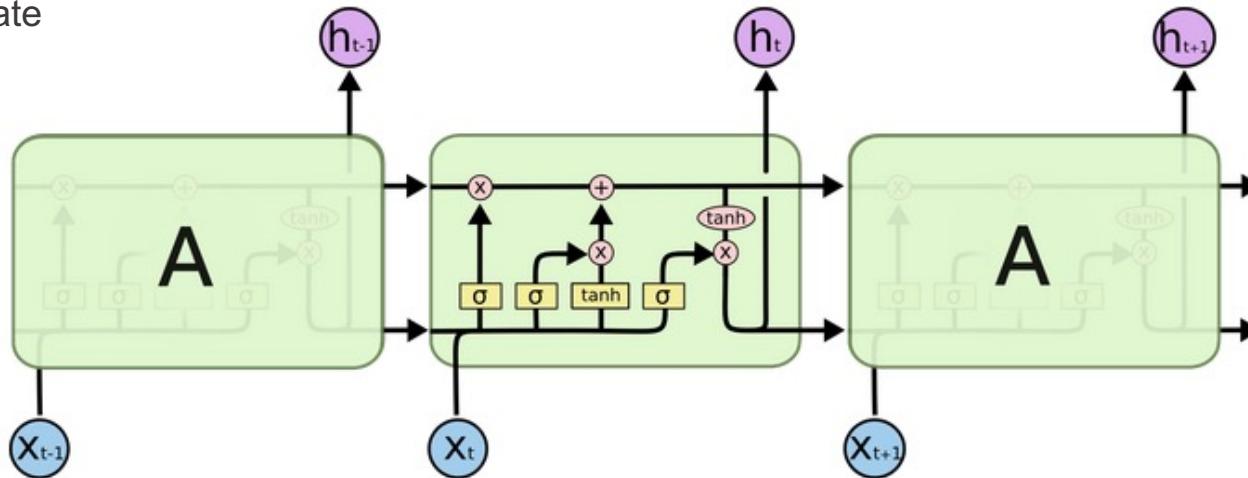
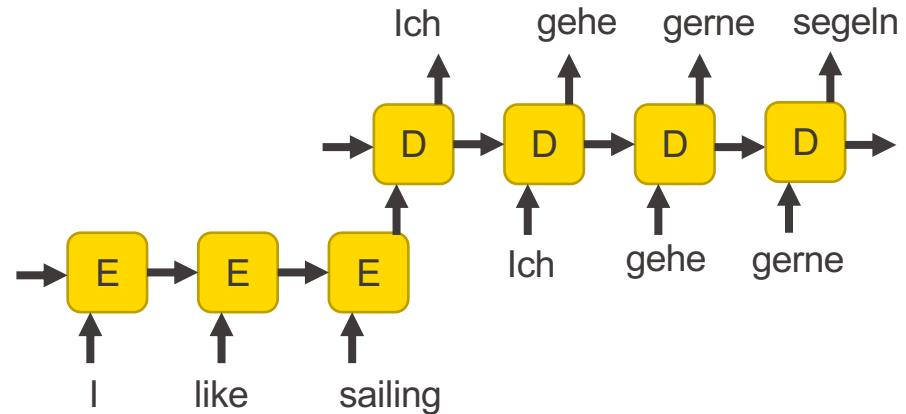
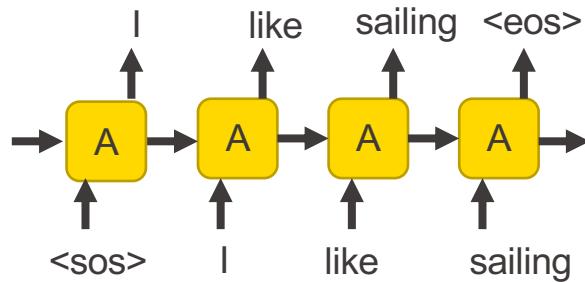


Image Source: Christopher Olah, <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Different Network-Structures and Applications

Many to Many

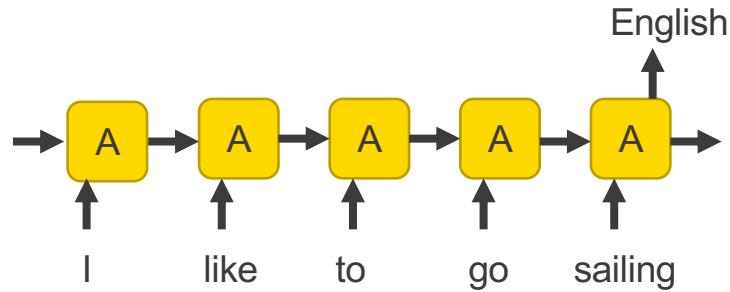


Language model

Neural machine translation

# Different Network-Structures and Applications

Many to one



Language classification  
Text classification

One to many

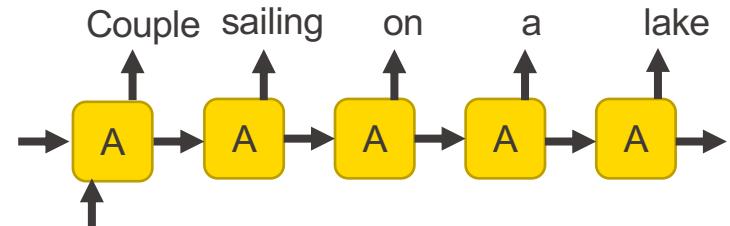
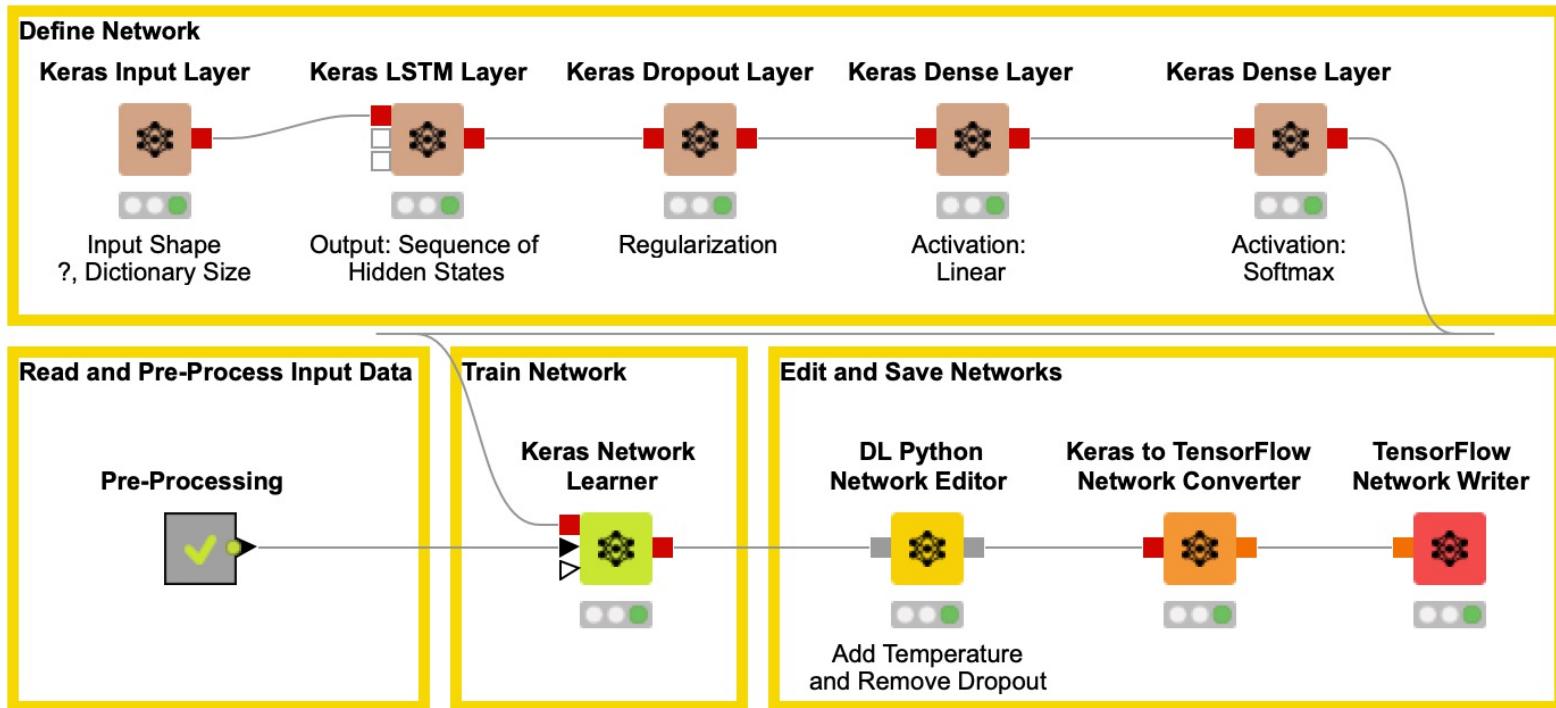


Image captioning

# Neural Network: Code-free

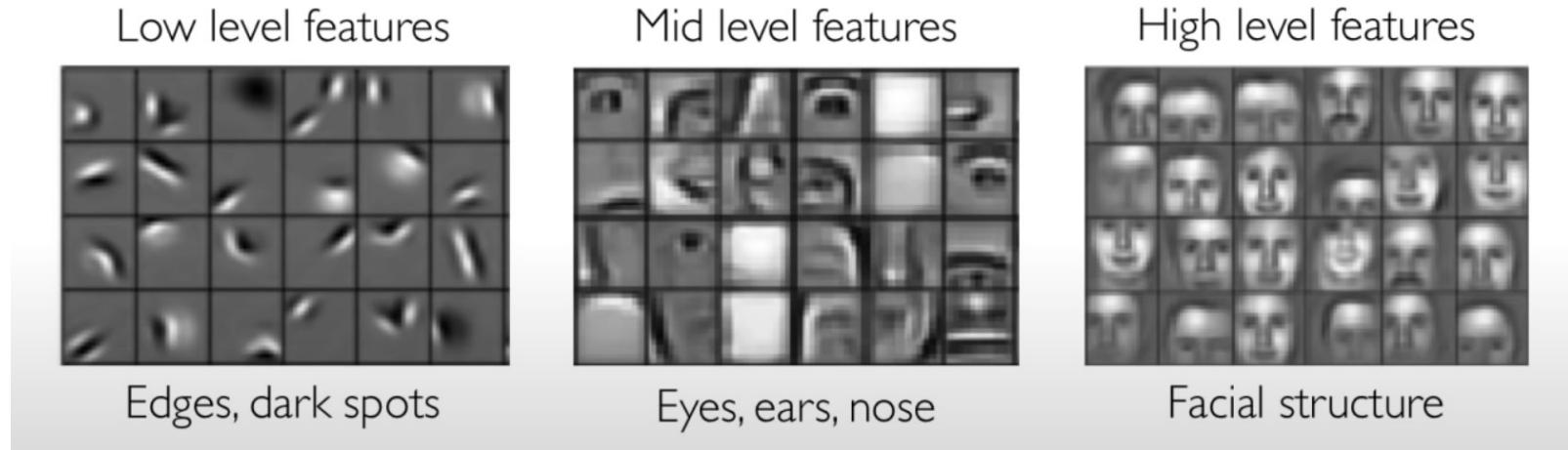


# Convolutional Neural Networks (CNN)



# Convolutional Neural Network (CNN)

- A CNN is a neural network with at least one convolutional layer.
- CNNs are commonly used when data has spatial relationships, e.g. images
- CNN learns a hierarchy of features using multiple convolution layers that detect different features.



Images from: [http://introtodeeplearning.com/slides/6S191\\_MIT\\_DeepLearning\\_L3.pdf](http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L3.pdf)

# Convolutional Neural Networks (CNN)

- Instead of connecting every neuron to the new layer a sliding window is used, which applies a filter on different parts of the image
- Some convolutions may detect edges or corners, while others may detect cats, dogs, or street signs inside an image

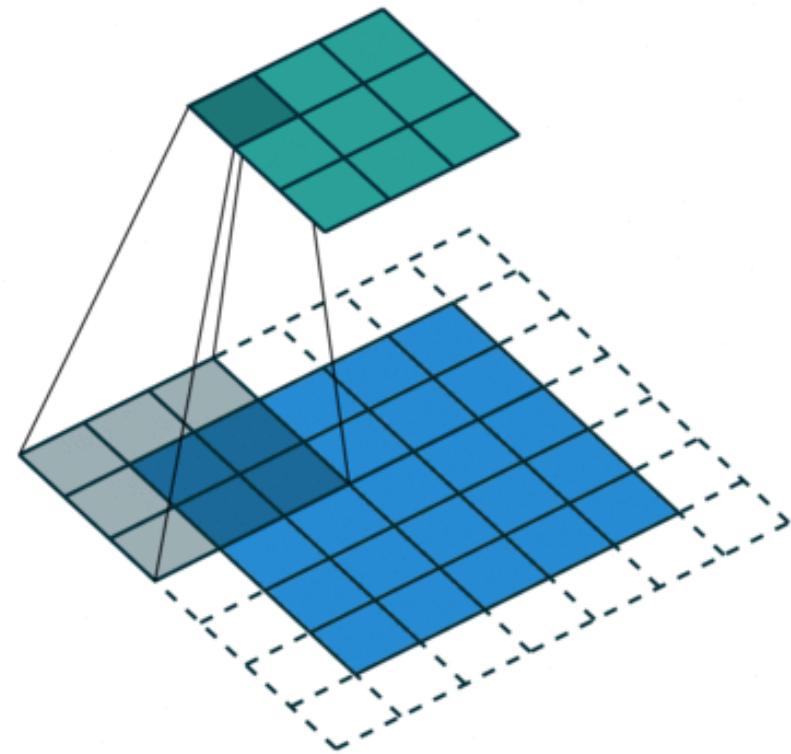
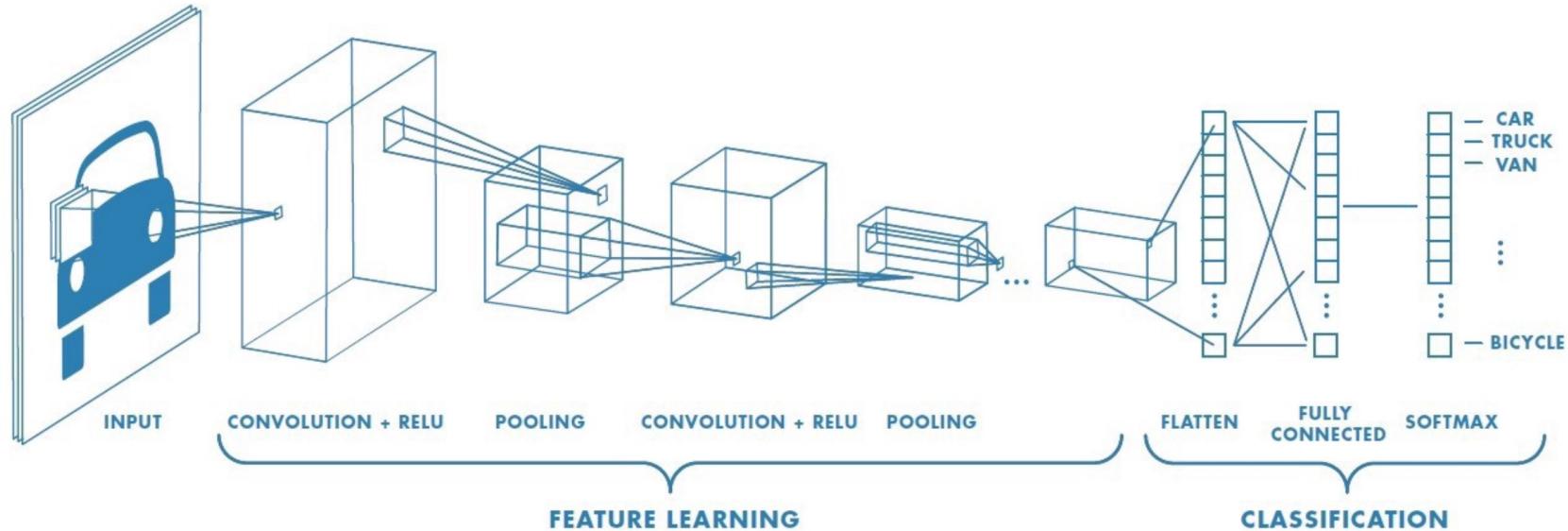
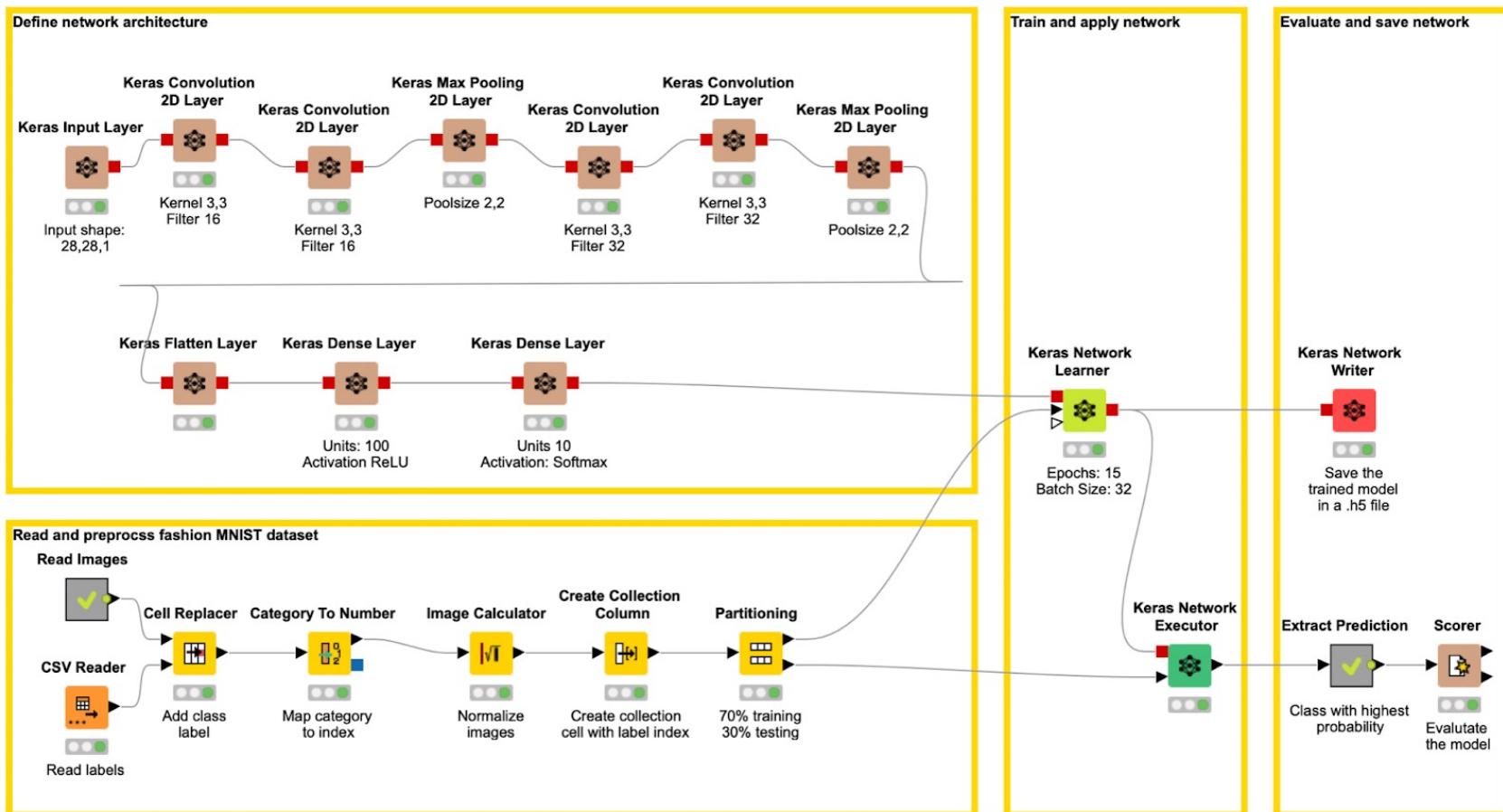


Image from: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

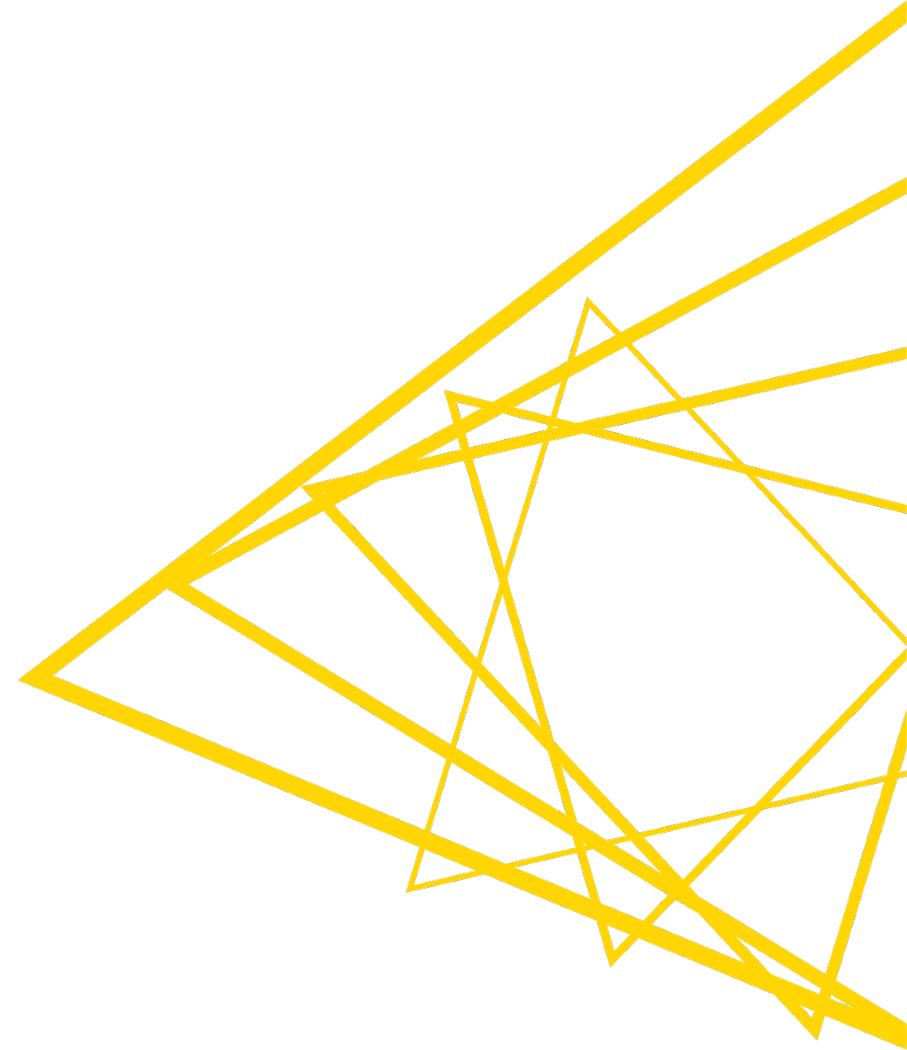
# Convolutional Neural Networks



# Building CNNs with KNIME



# Recommendation Engines



# Recommendation Engines and Market Basket Analysis

From the analysis of many shopping baskets ...



A-priori algorithm

Recommendation

IF



+

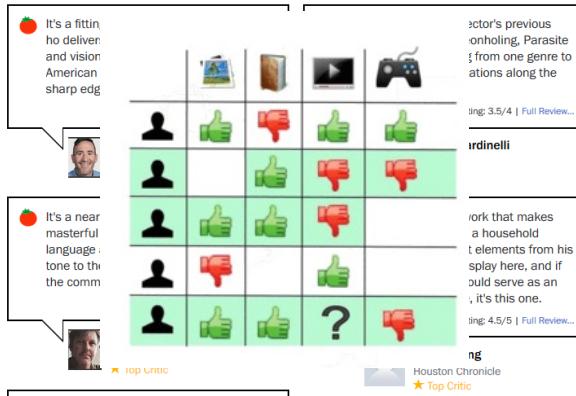


THEN



# Recommendation Engines or Market Basket Analysis

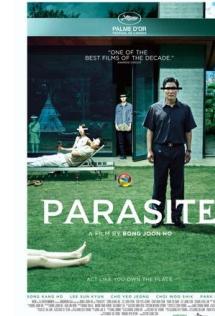
From the analysis of the reactions of many people to the same item ...



## Collaborative Filtering



## Recommendation



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 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

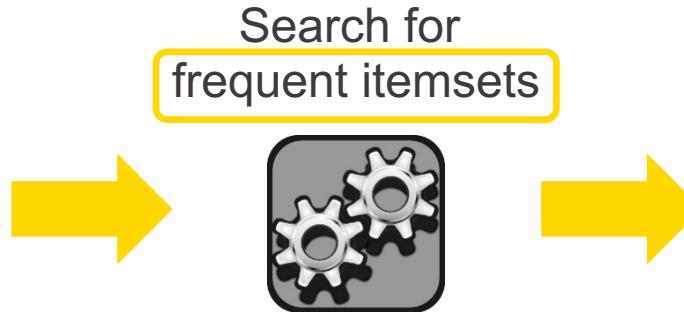
# A-priori Algorithm: the Association Rule

---



# 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}  $\rightarrow$  H

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

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

{B, F, H}  $\rightarrow$  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

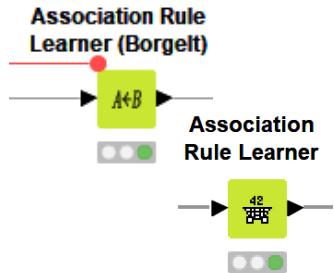
- Rule lift =  $\frac{support(\{A,B,F\} \Rightarrow H)}{support(A,B,F) \times support(H)}$



How often antecedent and consequent happen together compared with random chance

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

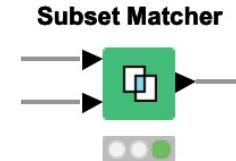
# Association Rule Mining (ARM): Two Phases



Discover all **frequent** and **strong** association rules

$$X \Rightarrow Y \rightarrow \text{"if } X \text{ then } Y"$$

with sufficient **support** and **confidence**



Two phases:

## 1. find all frequent itemsets (FI)

- Select itemsets with a minimum support

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

← Most of the complexity

## 2. build strong association rules

- Select rules with a minimum confidence:

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

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

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

confidence

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

# A-priori algorithm: an 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?

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

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

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

$$\begin{aligned} Rule\ lift &= \frac{s(milk, diaper, beer)}{s(milk, diaper) \times s(beer)} \\ &= \frac{0.4}{0.6 \times 0.6} = 1.11 \end{aligned}$$

# 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

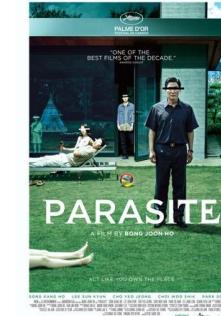
From the analysis of the reactions of many people to the same item ...



## Collaborative Filtering



## Recommendation



Inspired by your purchases



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
- $U = \text{set of top } N \text{ users most similar to user } u$
- Rating of user  $u$  on item  $i$  calculated as average of ratings of all similar users in  $U$ :

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

Pearson correlation

$$\text{simil}(u, u') = \frac{\sum_{i \in I_{xy}} (r_{u,i} - \bar{r}_u) (r_{u',i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in I_{xy}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{xy}} (r_{u',i} - \bar{r}_{u'})^2}}$$

Set of items rated by both user x and y

# Exercises:

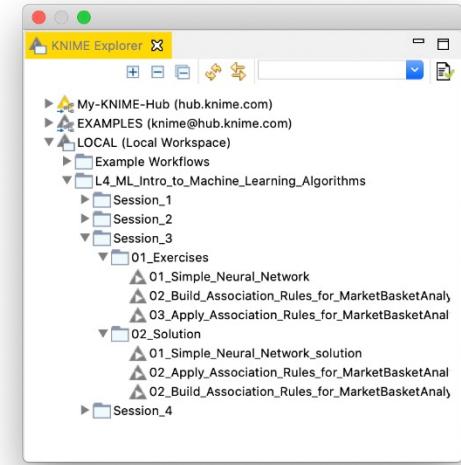
---

## ■ Neural Network

- Goal: Train an MLP to solve our classification problem (rank: high/low)
- 01\_Simple\_Neural\_Network

## ■ Market Basket Analysis

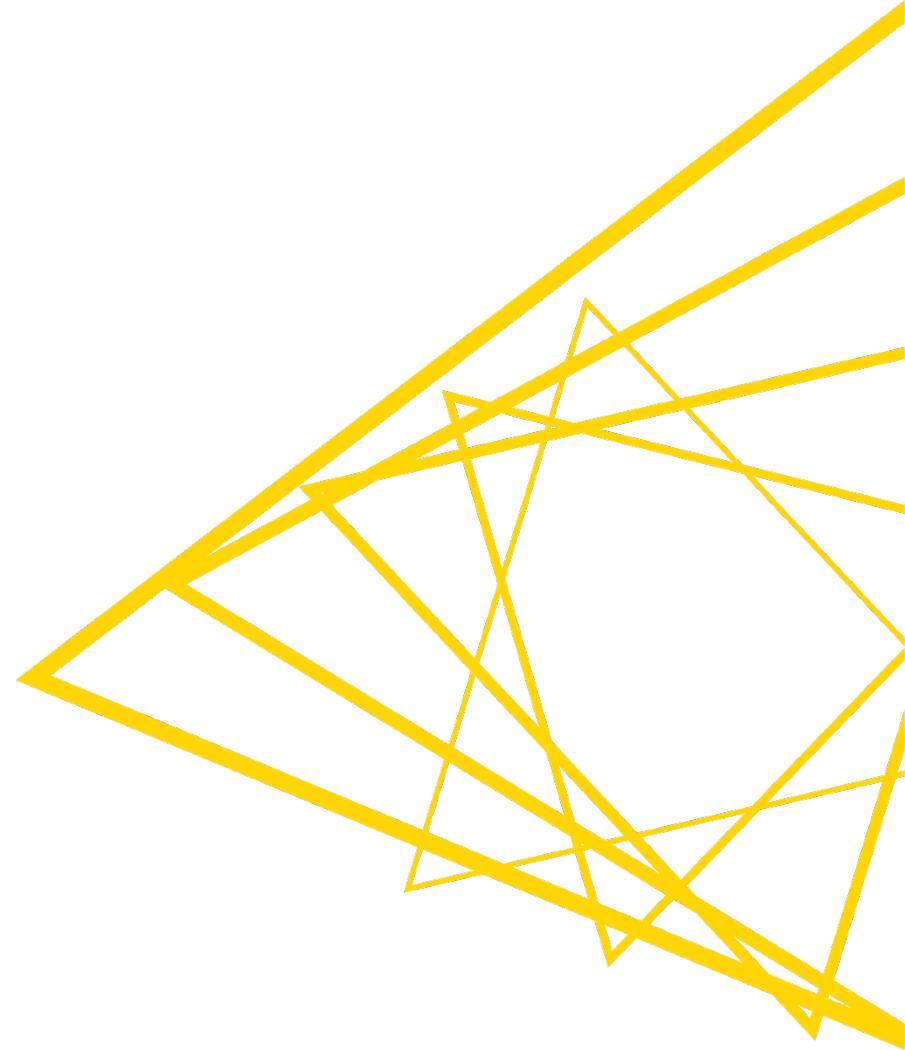
- 02\_Build\_Association\_Rules\_for\_MarketBasketAnalysis
- 03\_Apply\_Association\_Rules\_for\_MarketBasketAnalysis



# Session 4: Clustering & Data Preparation



# Unsupervised Learning: Clustering

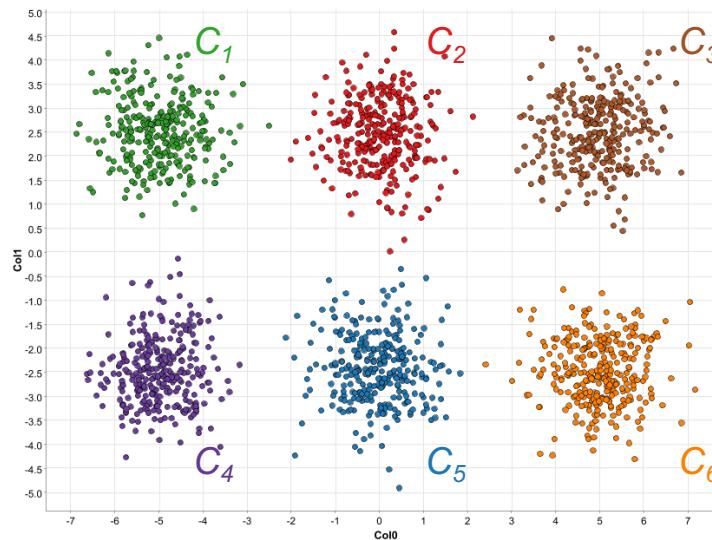


# Goal of Clustering Analysis

Discover hidden structures in **unlabeled** data (unsupervised)

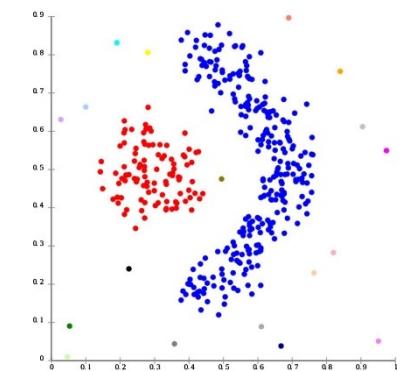
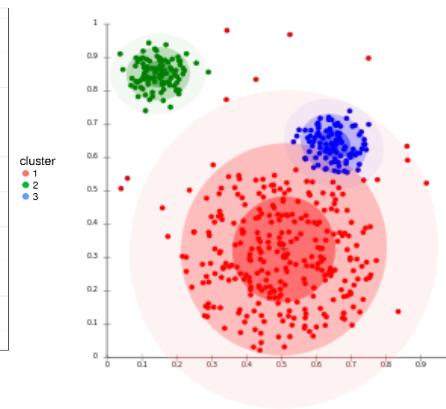
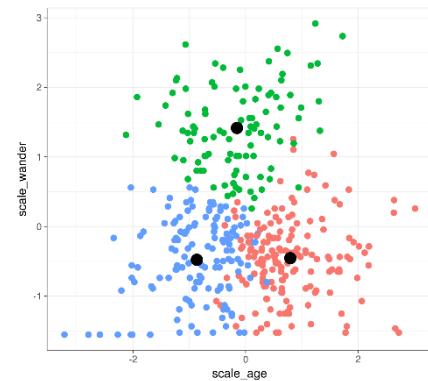
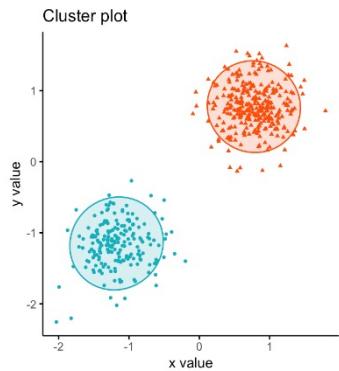
**Clustering** identifies a finite set of groups (*clusters*)  $C_1, C_2 \dots, C_k$  in the dataset such that:

- 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
  - 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

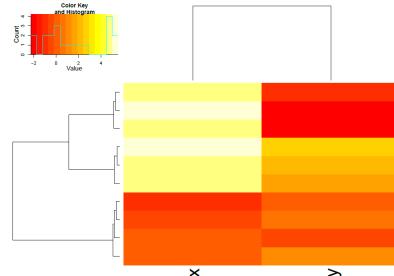
## Examples

- Customer segmentation
- Molecule search
- Anomaly detection

# Types of Clustering Approaches

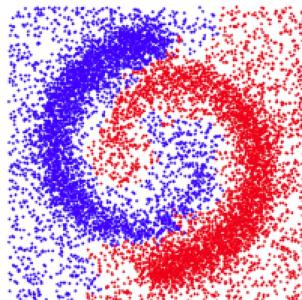
## Linkage Based

e.g. Hierarchical Clustering



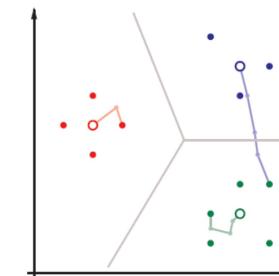
## Density based Clustering

e.g. DBSCAN



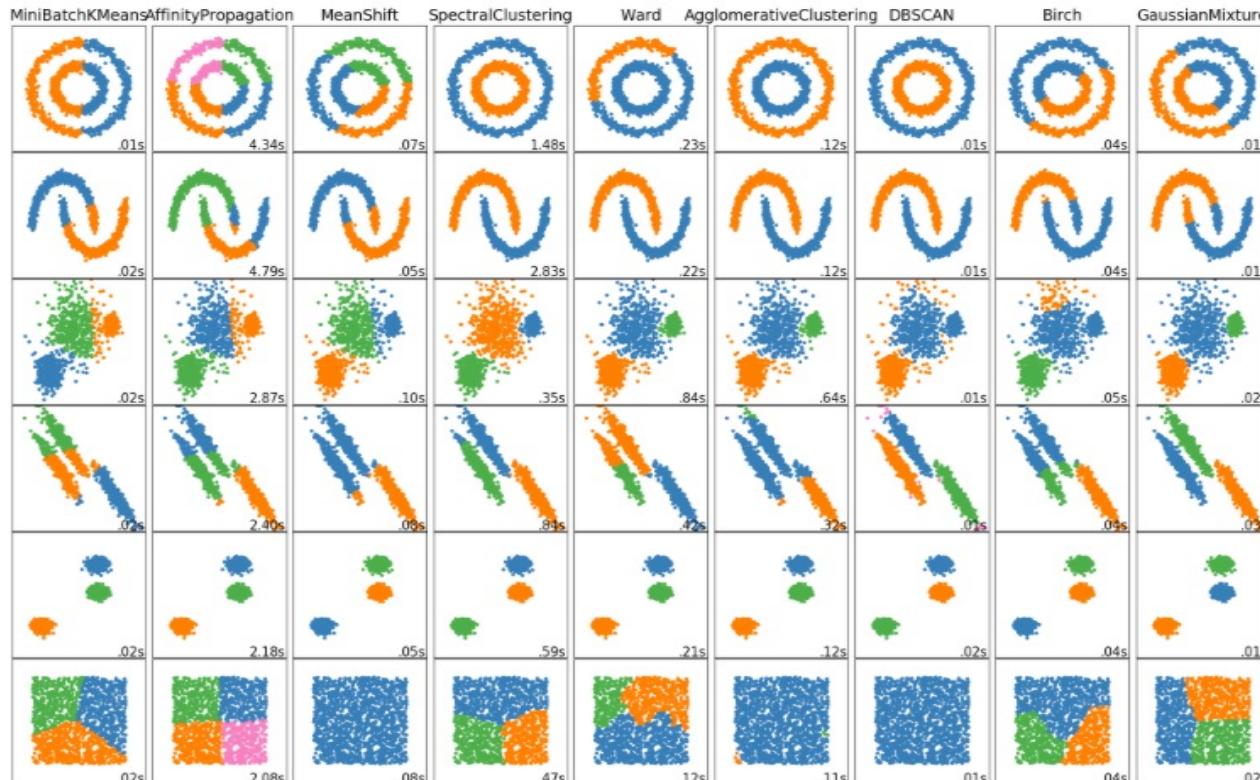
## Clustering by Partitioning

e.g. k-Means

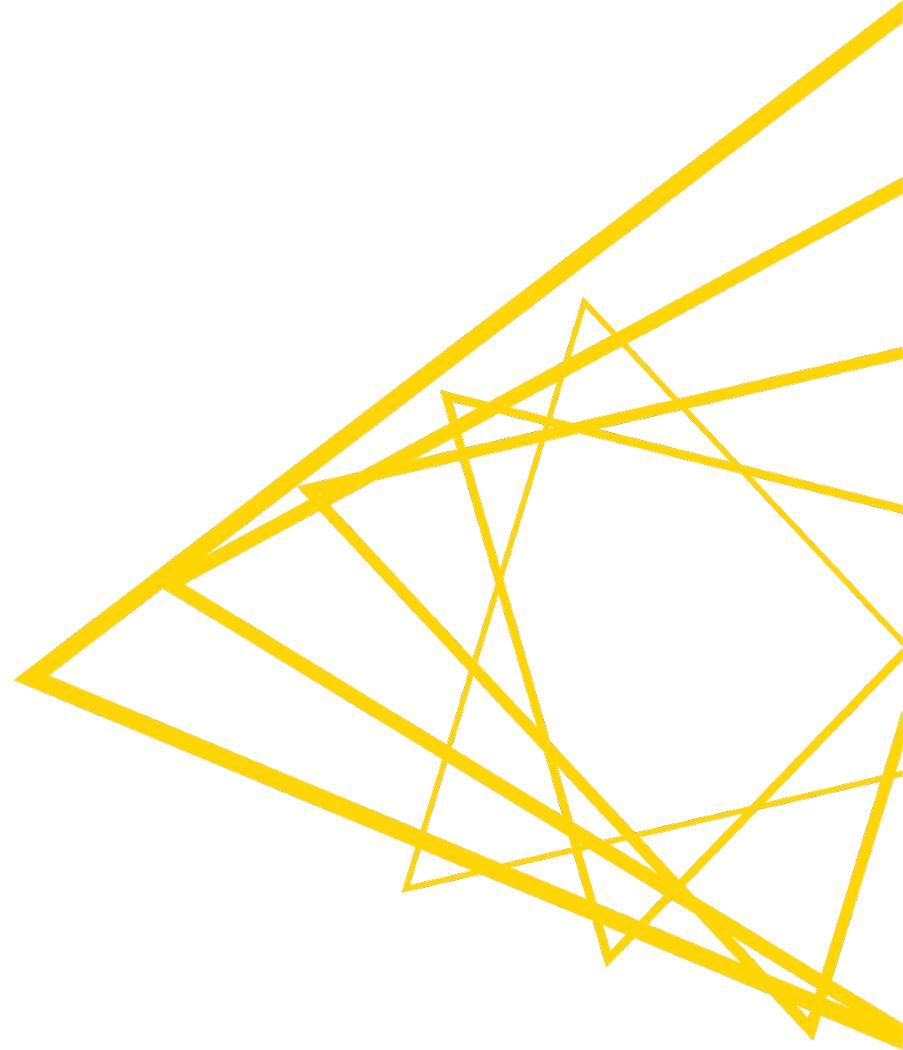


# Types of Clustering Approaches

- No clustering method works universally well



# Clustering: Partitioning k-Means

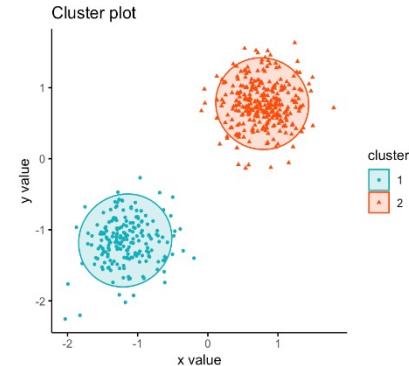


# Partitioning

## Goal:

A (disjoint) partitioning into  $k$  clusters with minimal costs

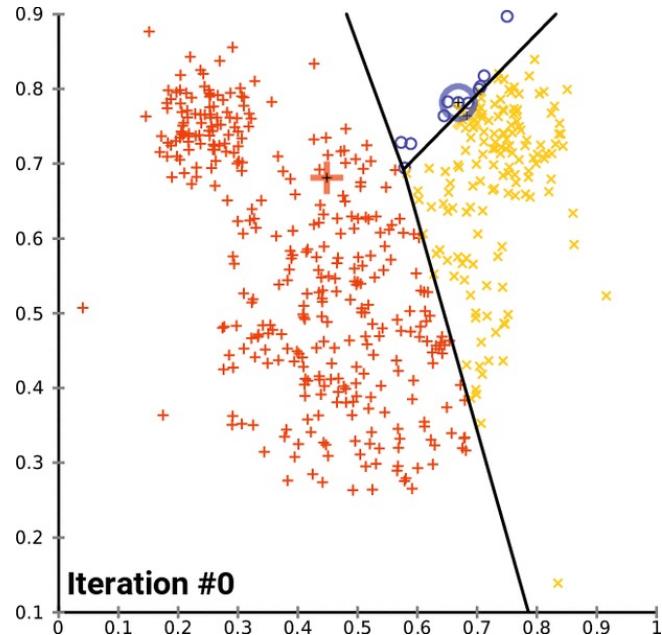
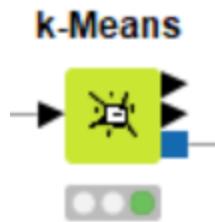
- Local optimization method:
  - choose  $k$  initial cluster representatives
  - optimize these representatives iteratively
  - assign each object to its **most similar cluster representative**
- Types of cluster representatives:
  - Mean of a cluster (*construction of central points*)
  - Median of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

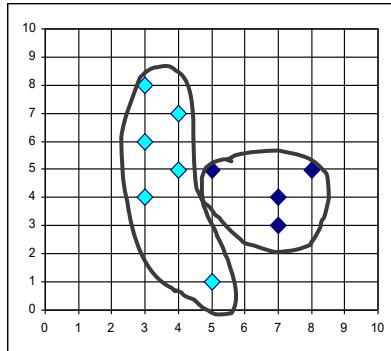
1. Use  $k$  objects / rows as the initial centroids
2. Assign each object to the cluster with the **nearest** centroid (Euclidean distance)
3. Compute the centroids from the current partition
4. Go back to Step 2, repeat until the updated centroids stop moving significantly



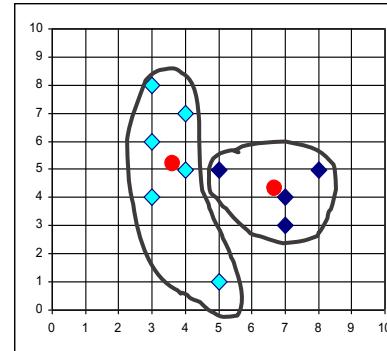
[https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

# k-Means Algorithm

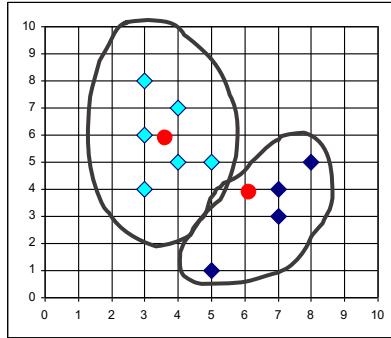
K = 2



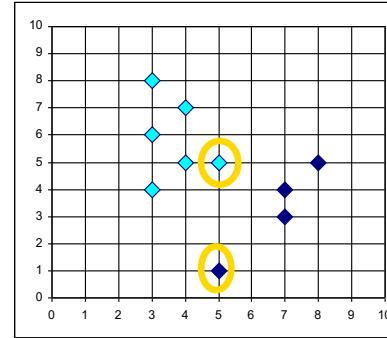
Calculation of new centroids



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

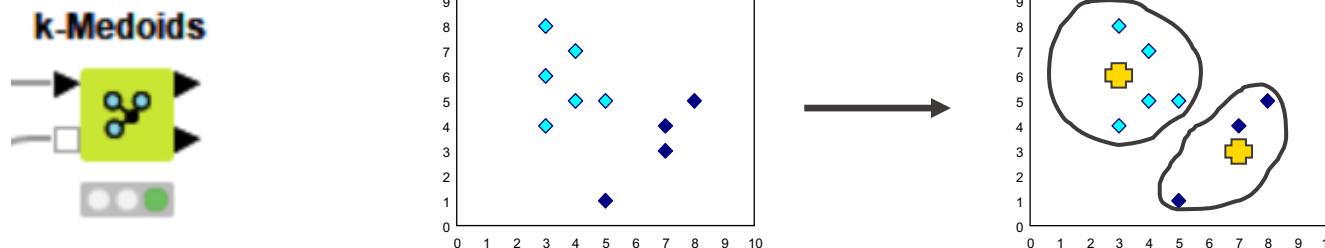
# Outliers: k-Means vs k-Medoids

## 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: Distance Functions



# Distance Functions for Numeric Attributes

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

- $L_p$ -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* ( $p = \infty$ )

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

# Influence of Distance Function / Similarity

- Clustering vehicles:

- red Ferrari
- green Porsche
- red Bobby car



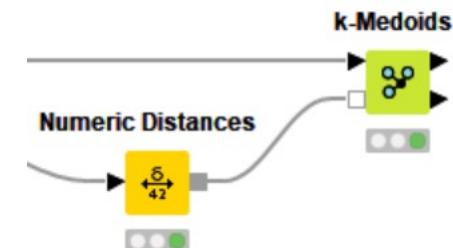
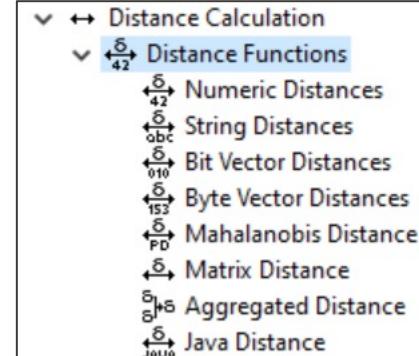
- Distance Function based on maximum speed (numeric distance function):

- 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



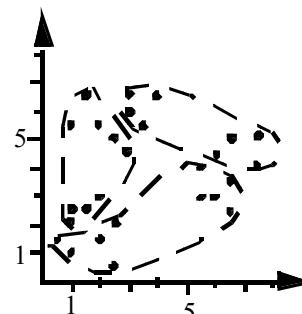
# Clustering: Quality Measures

## Silhouette

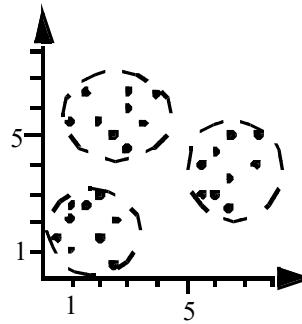


# Optimal Clustering: Example

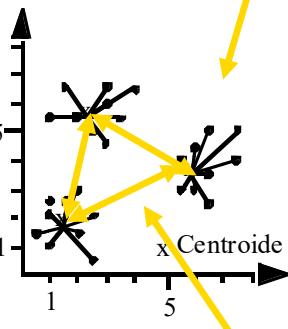
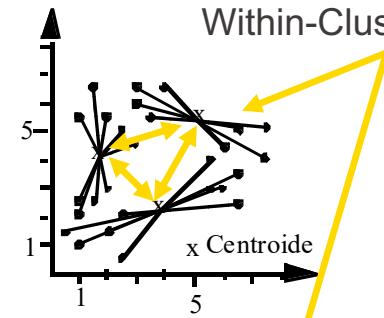
Bad  
Clustering



Good  
Clustering



Within-Cluster Variation



Between-Cluster Variation

# Cluster Quality Measures

---

Centroid  $\mu_C$ : mean vector of all objects in clustering  $C$

- Within-Cluster Variation:

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

- 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

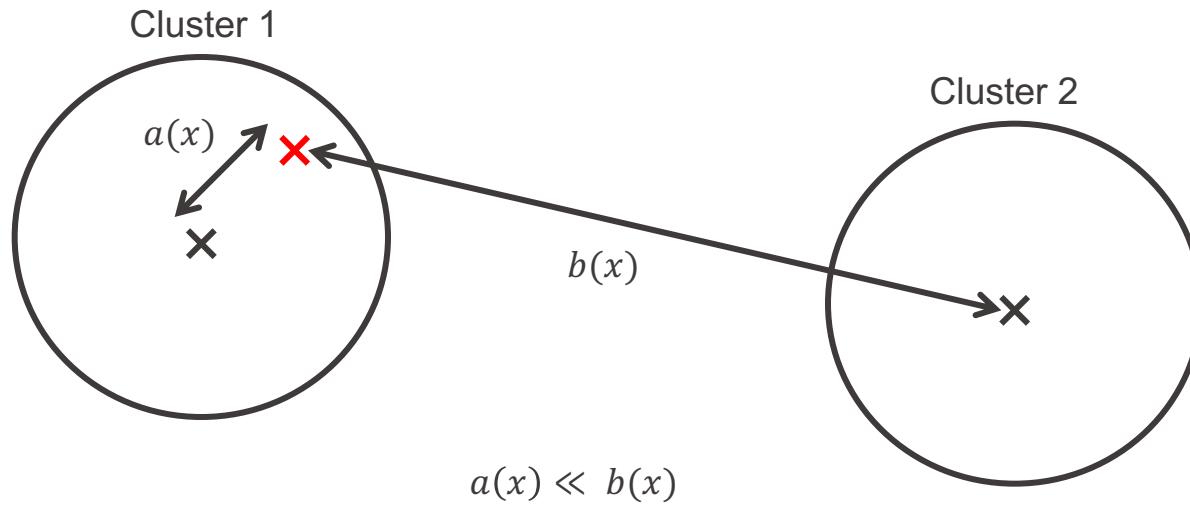
- $a(x)$ : distance of object  $x$  to its cluster representative
- $b(x)$ : distance of object  $x$  to the representative of the „second-best“ cluster
- **Silhouette**  $s(x)$  of  $x$

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

# Silhouette-Coefficient

---

Good clustering...

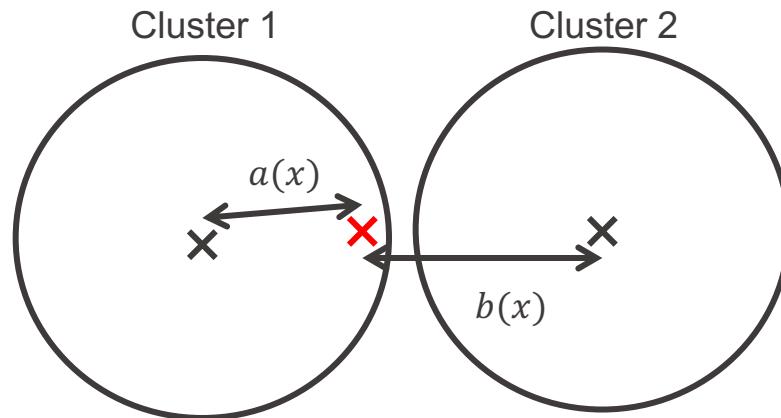


$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \approx \frac{b(x)}{b(x)} = 1$$

# Silhouette-Coefficient

---

...not so good...

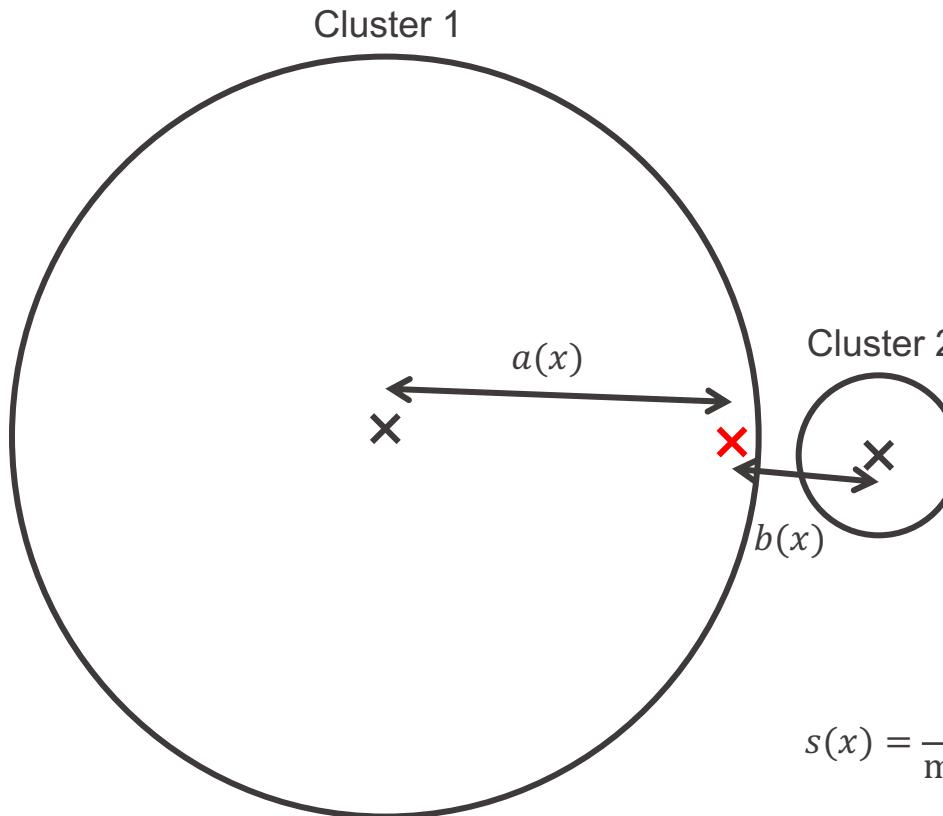


$$a(x) \approx b(x)$$

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

# Silhouette-Coefficient

...bad clustering.



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

# Silhouette-Coefficient for Clustering C

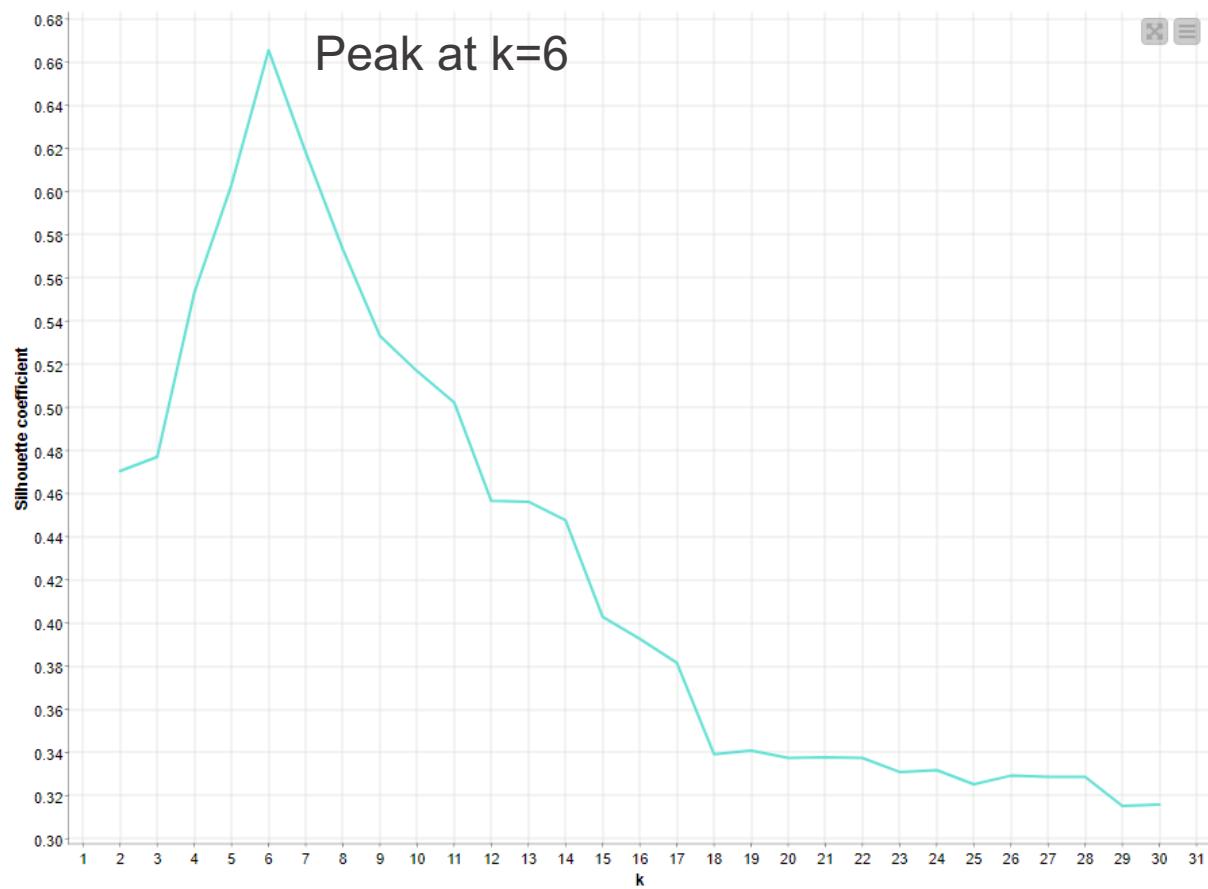
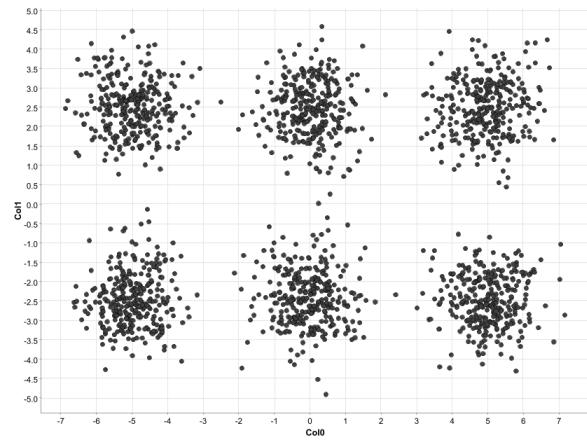
---

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

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

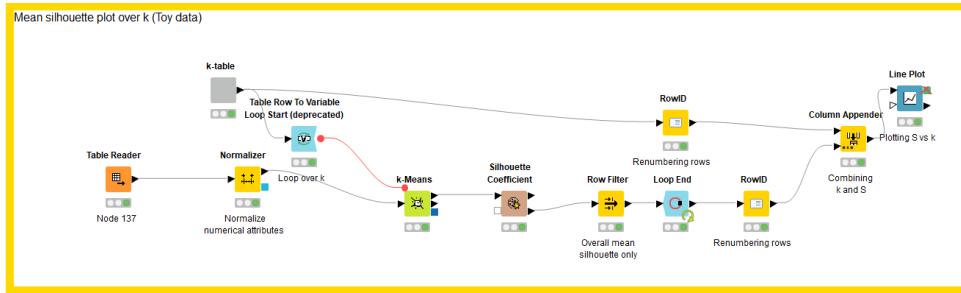
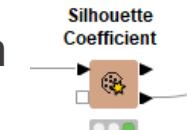
- Interpretation of silhouette coefficient:
  - $s_c > 0.7$  : strong cluster structure,
  - $s_c > 0.5$  : reasonable cluster structure,
  - ...

# Silhouette Coefficient over k

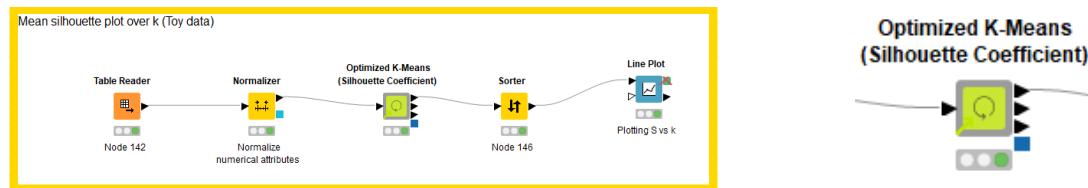


# Silhouette Coefficient over a Range of k

- Silhouette Coefficient Node in KNIME Analytics Platform
- Loop over various values of k



- Optimized k-means component
- Loop over various values of k



Optimized K-Means  
(Silhouette Coefficient)

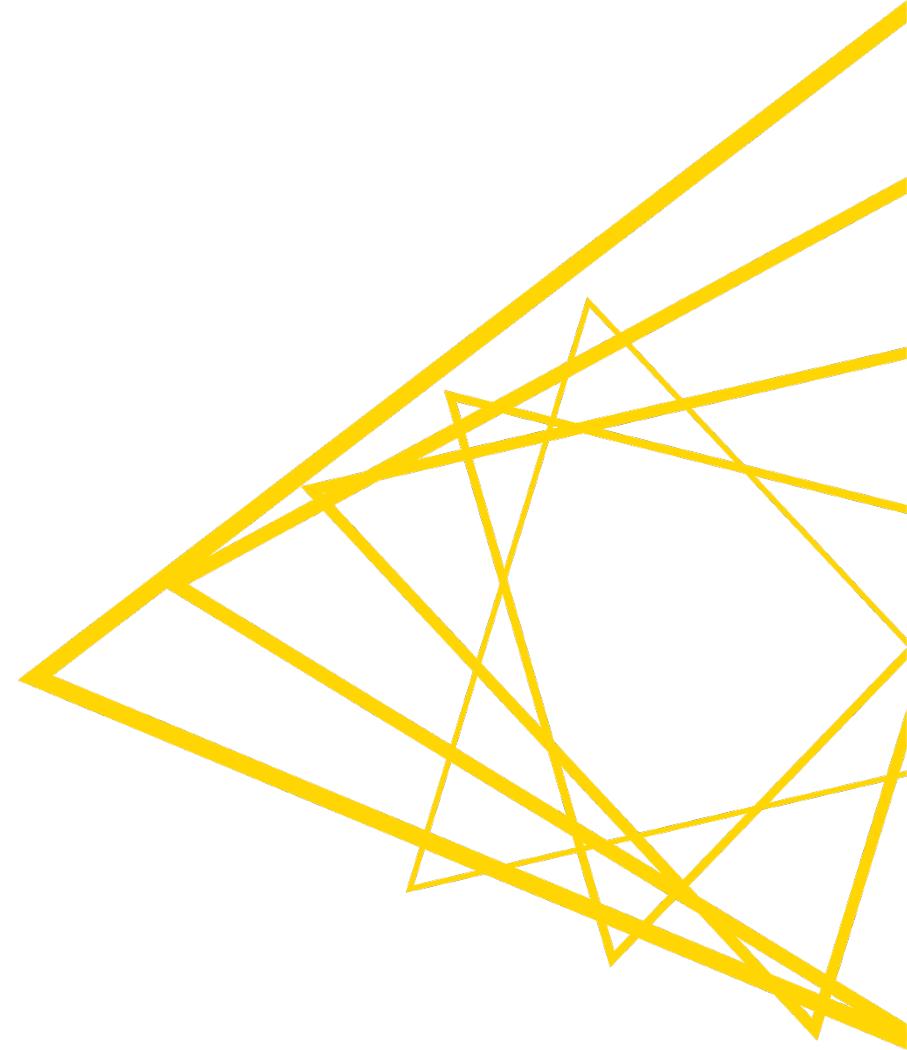


# Summary: Clustering by Partitioning

---

- 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: Linkage Hierarchical Clustering



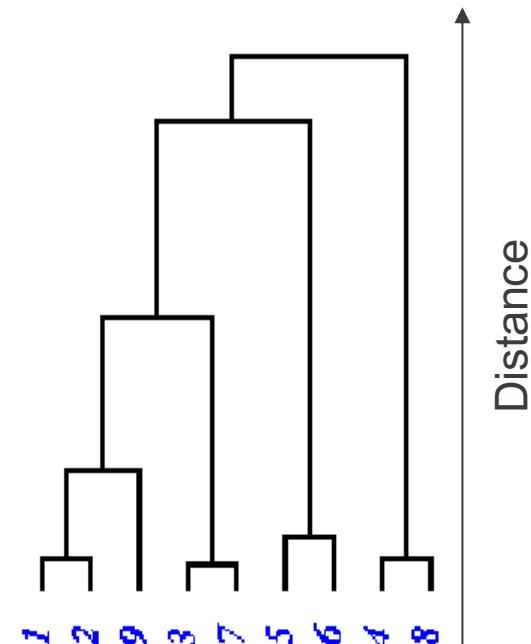
# Linkage Hierarchies: Basics

## Goal

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

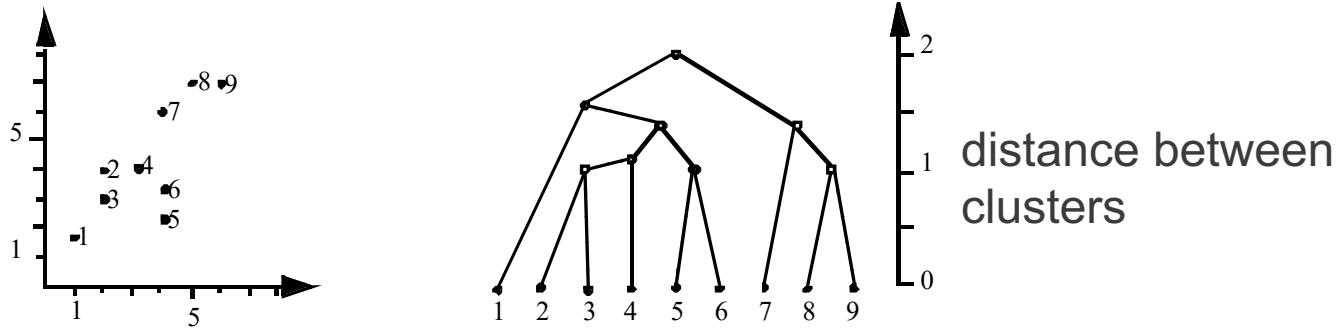
## Dendrogram:

- 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



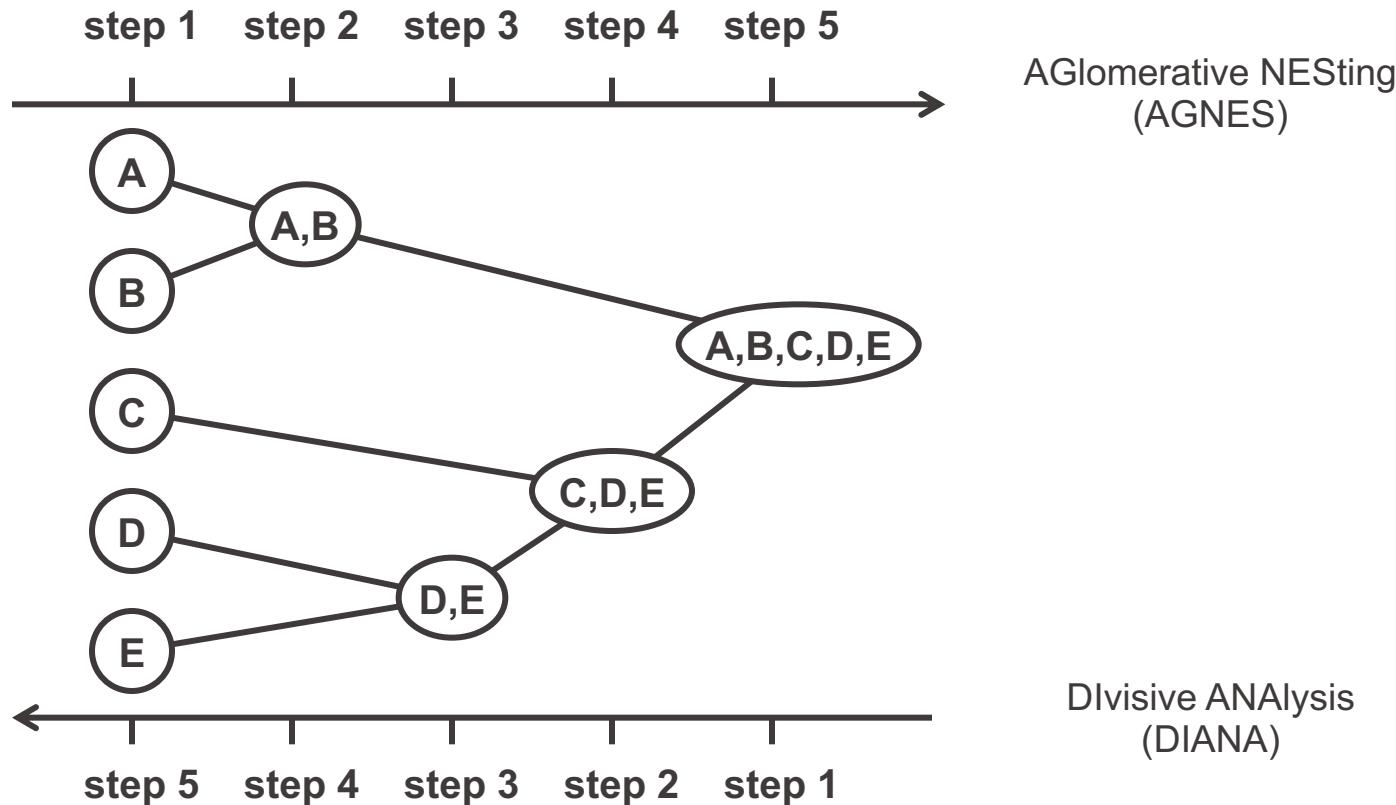
# Linkage Hierarchies: Basics

- Example dendrogram



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

# Agglomerative vs. Divisive Hierarchical Clustering



# 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.

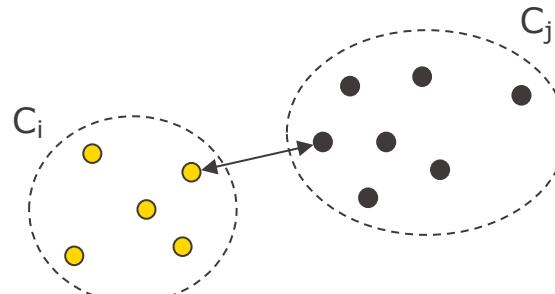
# Single Linkage

- 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



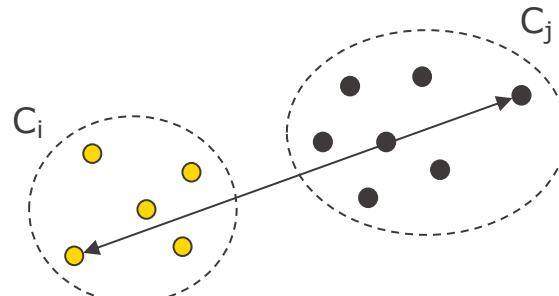
# Complete Linkage

- 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):

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

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

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

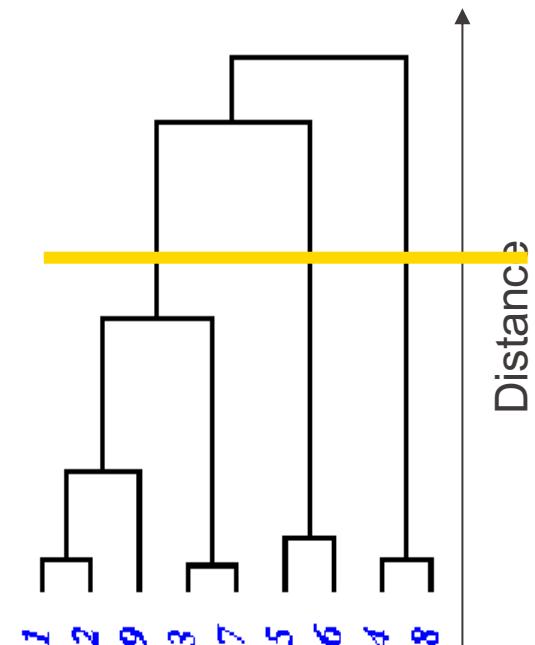
Distance between two centroids

- Merge Step:

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

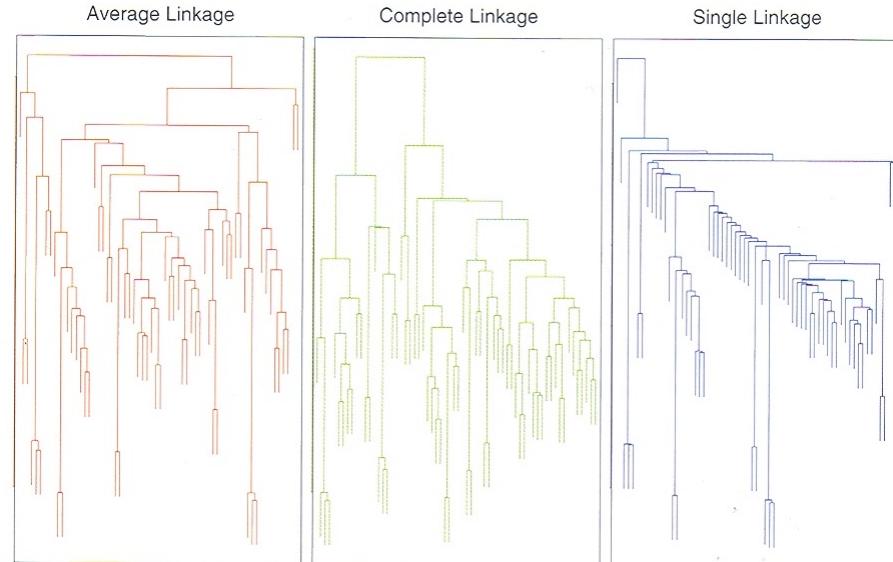
# Comments on Single Linkage and Variants

- + 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

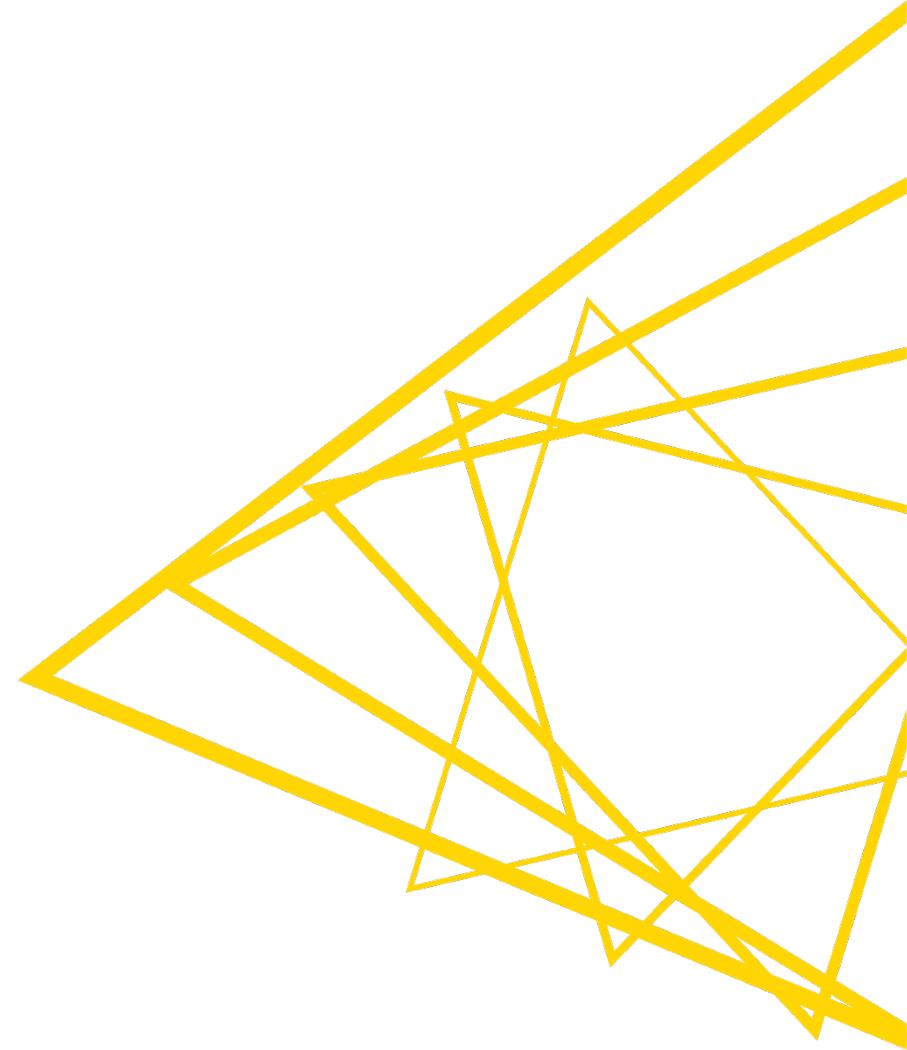


# Linkage Based Clustering

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



# Clustering: Density DBSCAN



# Clustering: DBSCAN

---

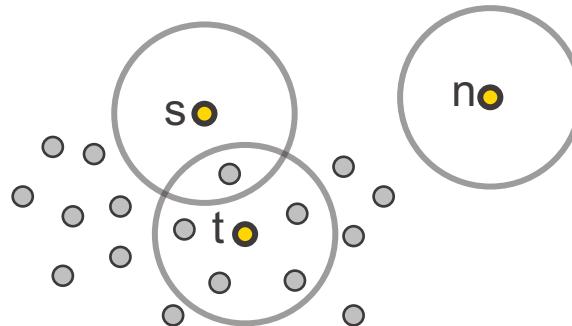
DBSCAN - a density-based clustering algorithm - defines five types of points in a dataset.

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

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

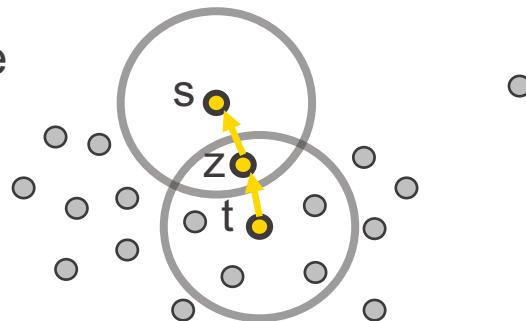
## Example with MinPts = 3

Core Point  
vs. Border Point  
vs. Noise



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

Directly Density Reachable  
vs. Density Reachable

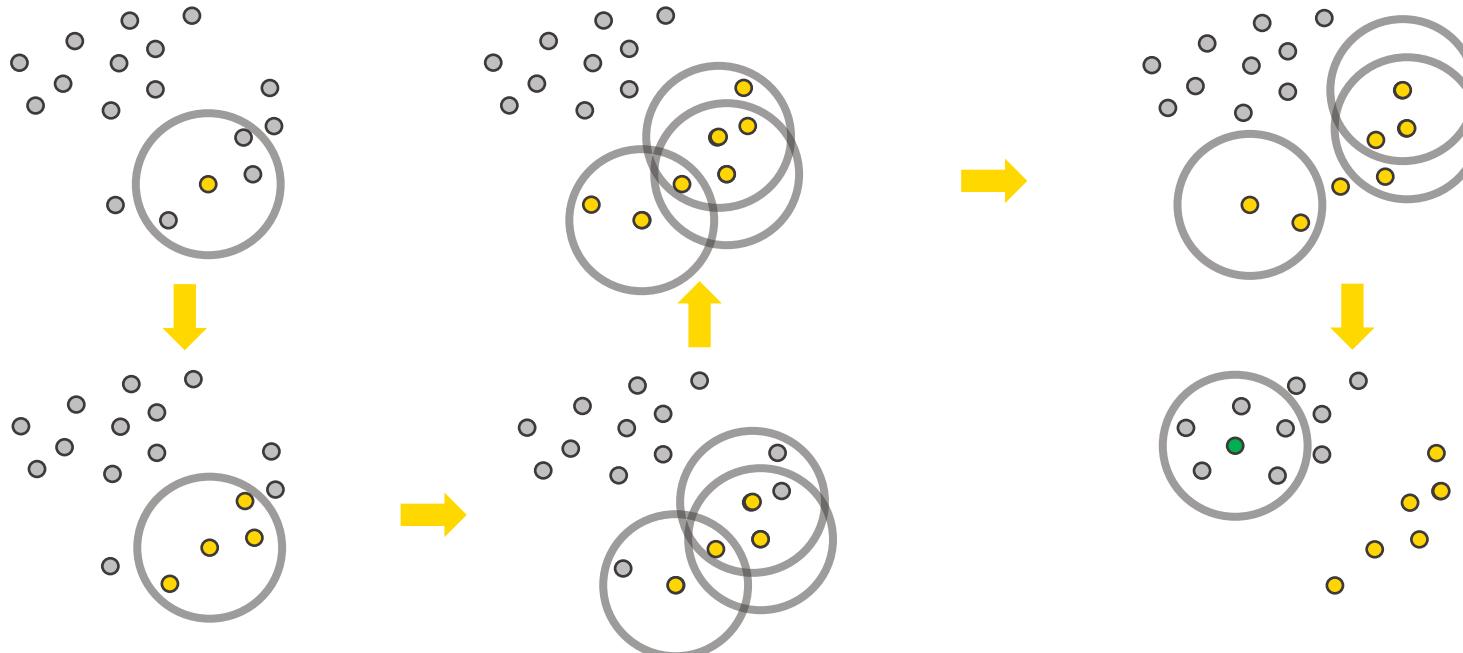


- z is directly density reachable from t
- s is not directly density reachable from t, but density reachable via z

*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  $\epsilon$ -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.  $\varepsilon$  and MinPts is the set of all density-based clusters  $C_i$  w.r.t.  $\varepsilon$  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.

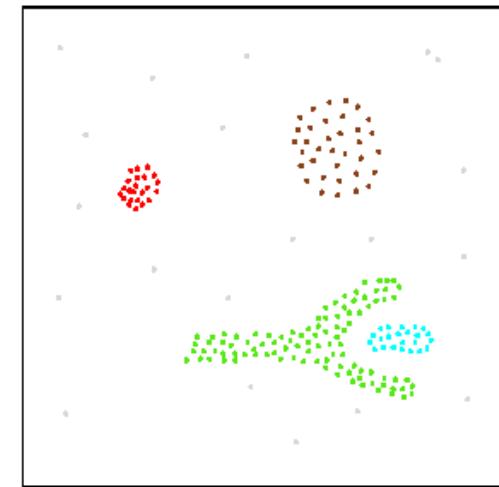
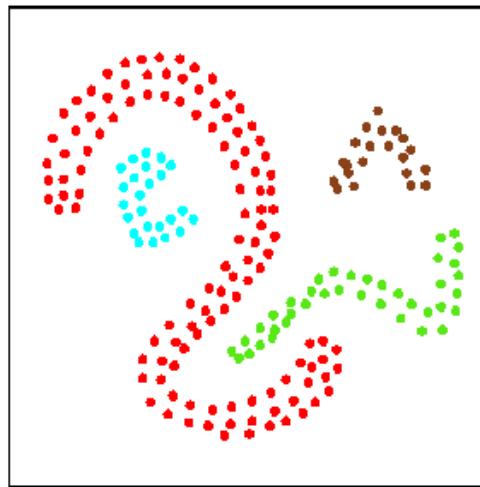
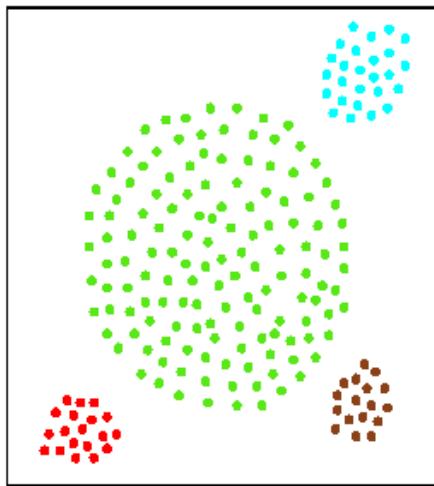
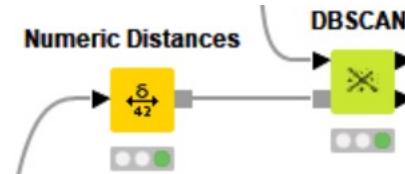
Property:

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

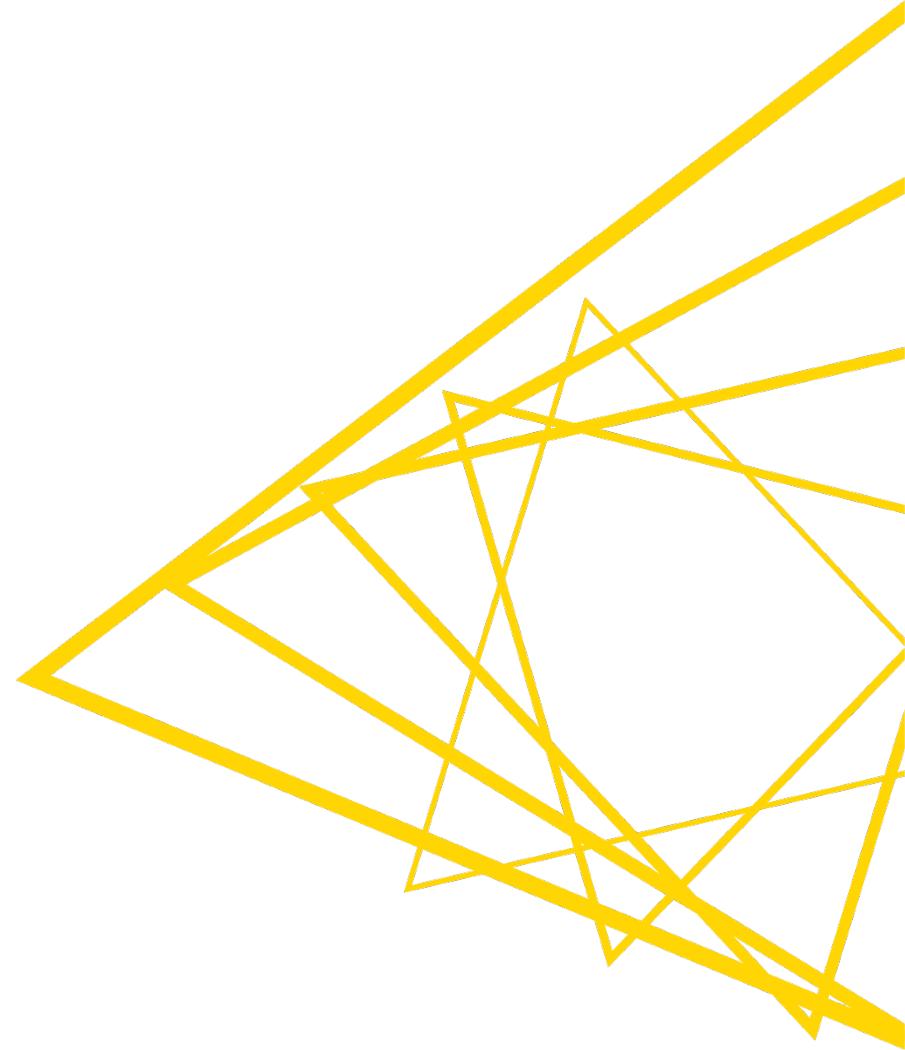
$$C_i = \{o \in D \mid o \text{ density-reachable from } p \text{ w.r.t. } \varepsilon \text{ and } \text{MinPts}\}.$$

# DBSCAN [Density Based Spatial Clustering of Applications with Noise]

- DBSCAN uses (spatial) index structures for determining the  $\varepsilon$ -environment:  
→ computational complexity  $O(n \log n)$  instead of  $O(n^2)$
- Arbitrary shape clusters found by DBSCAN
- Parameters:  $\varepsilon$  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 analysis 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 gram dominate over the length values for the similarity of records!

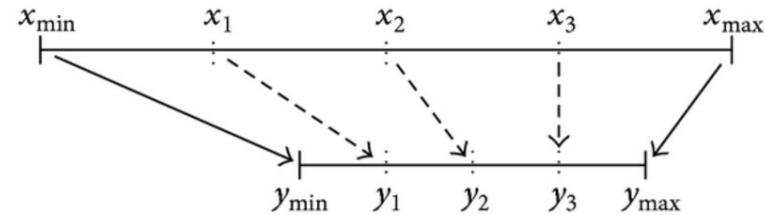
Goal of normalization:

- Transformation of attributes to make record ranges comparable

# Normalization: Techniques

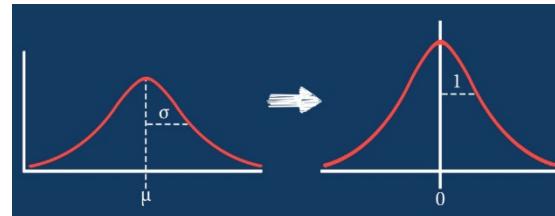
- min-max normalization

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



- z-score normalization

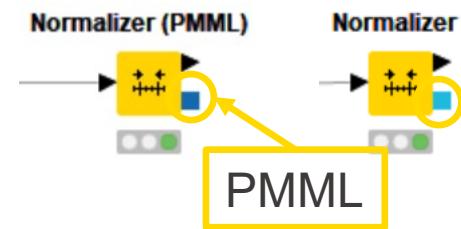
$$y = \frac{x - \text{mean}(x)}{\text{stddev}(x)}$$



- normalization by decimal scaling

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

Here  $[y_{min}, y_{max}]$  is  $[-1, 1]$



# 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

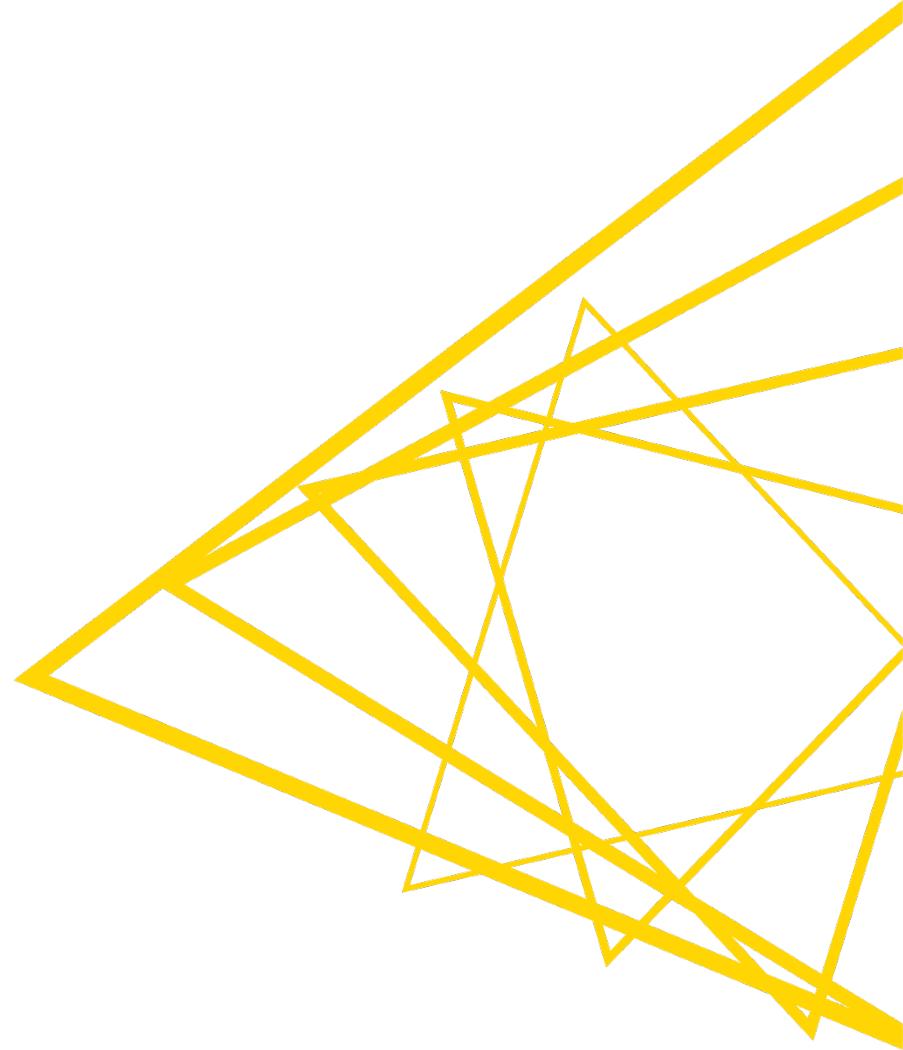
---

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, ...)

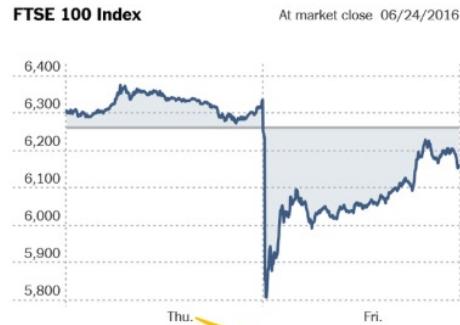


# Data Preparation: Outlier Detection

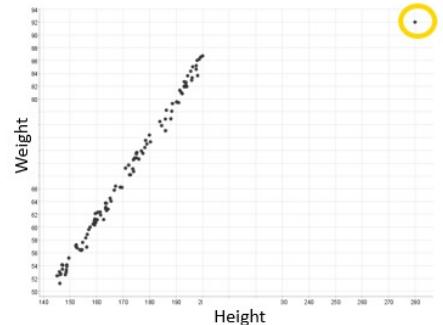


# Outlier Detection

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



Brexit  
referendum



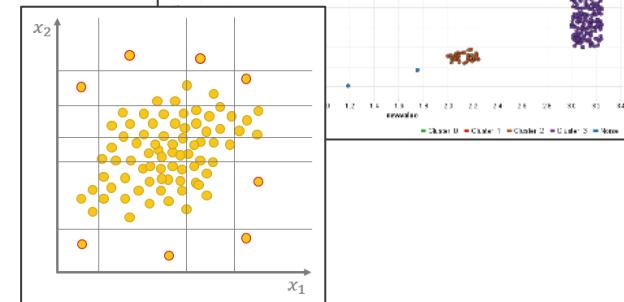
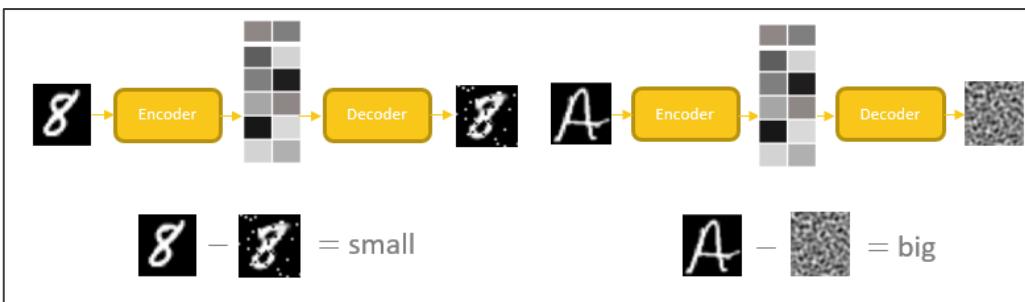
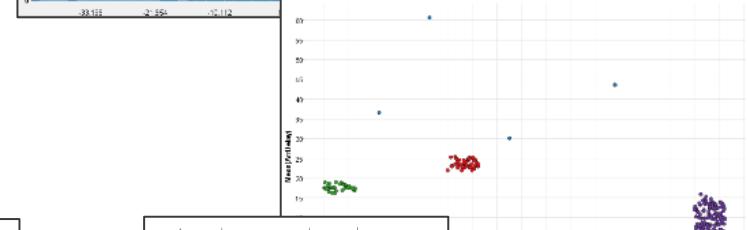
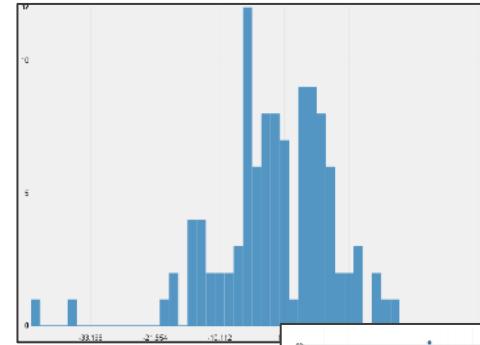
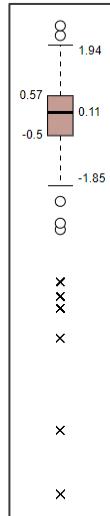
# Outlier Detection: Motivation

---

- 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



Open for Innovation ®

Hub Blog Forum Events Careers Contact [Download](#)

SOFTWARE / SOLUTIONS / LEARNING / PARTNERS / COMMUNITY / ABOUT

Home > About > Blog  
/ News  
**/ Blog**  
/ Team  
/ Careers  
/ Contact Us  
/ Travel Information  
/ KNIME Open Source Story  
/ Open for Innovation

## Four Techniques for Outlier Detection

Mon, 10/01/2018 - 10:00 – admin  
*Authors: Maarit Widmann and Moritz Heine*

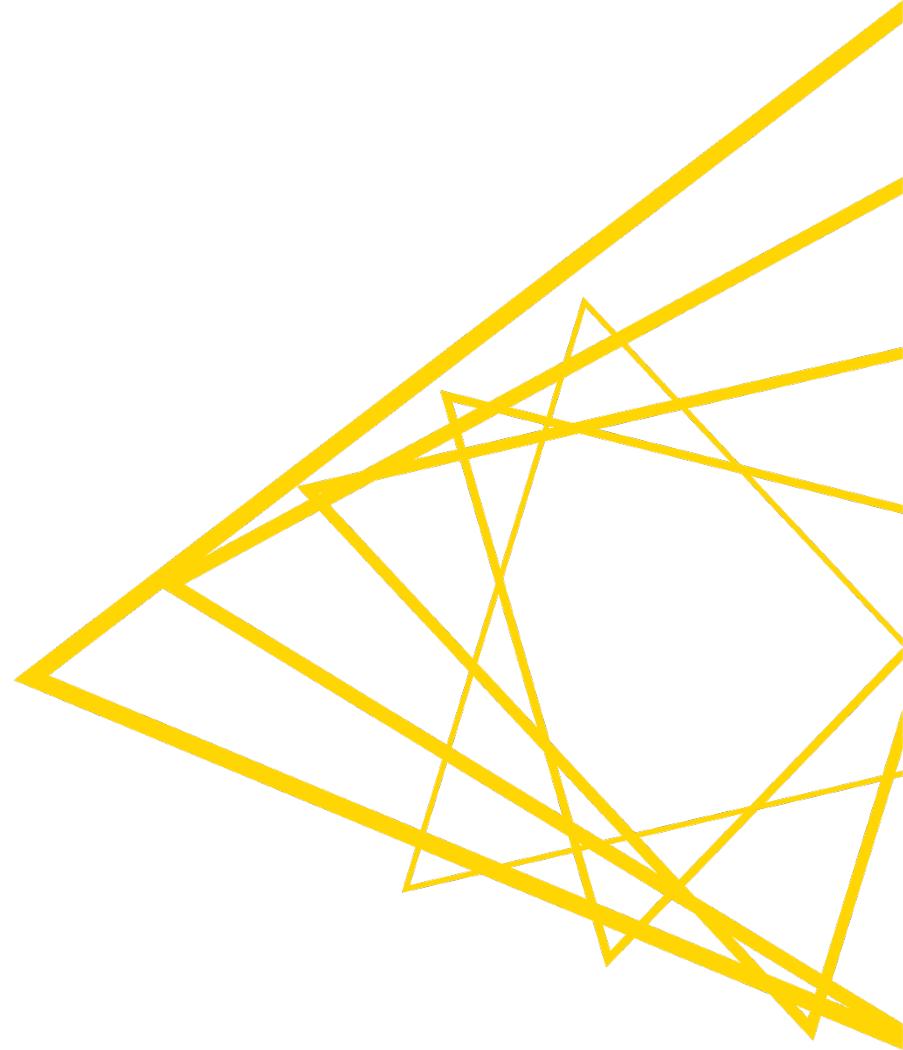
Ever been skewed by the presence of outliers in your set of data? Anomalies, or outliers, can be a serious issue when training machine learning algorithms or applying statistical techniques. They are often the result of errors in measurements or exceptional system conditions and therefore do not describe 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

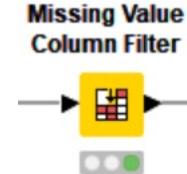
# Missing Values Ratio

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

File Hilit Navigation View

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

Row ID	D	Var 16	I	Var 17	I	Var 18	I	Var 19	S	Var 20	I	Var 21	I	Var 22	I	Var 23	I	Var 24	I	Var 25	I	Var 26	I	Var 27	D	Var 28
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	?	?	?	?	?	?	?	?	?	?	?	Missing Value	
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	



IF (% missing value > threshold) THEN remove column

# Low Variance

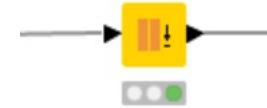
⚠ Output table - 0:347:0:337 - Missing Value (Numeric: 0)

File Hilite Navigation View

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	1	336	420	0	8	72	0	0	133.12
Row52	1	120	150	0	0	16	0	0	286.96
Row54	1	124	155	0	0	0	0	0	234.72
Row55	1	184	230	0	4	64	0	0	642.64
Row56	1	268	335	0	4	88	0	0	133.12
Row57	1	128	160	0	0	96	0	0	198.88
Row59	1	132	165	0	0	112	0	0	253.52
Row60	1	44	55	0	0	24	0	0	186.64
Row61	1	104	130	0	4	72	0	0	166.56
Row62	1	212	265	0	6	136	0	0	379.6
Row63	1	20	25	0	0	0	0	0	166.56
Row65	1	492	615	0	18	256	0	0	133.12
Row66	1	148	185	0	2	8	0	0	186.64
Row68	1	140	175	0	2	40	0	0	176.56
Row69	1	0	0	0	0	0	0	0	166.56
Row71	1	0	0	0	0	0	0	0	392.08
Row72	1	124	155	0	6	88	0	0	153.2
Row73	1	152	190	0	0	32	0	0	253.52
Row74	1	324	405	0	8	104	0	0	186.64
Row75	1	0	0	0	0	0	0	0	0
Row76	1	60	75	0	6	0	0	0	200
Row77	1	180	225	0	4	88	0	0	166.56
Row78	1	232	290	0	4	144	0	0	200
Row79	1	16	20	0	0	16	0	0	313.68
Row81	1	152	190	0	0	48	0	0	220.08
Row82	1	108	135	0	4	88	0	0	166.56

Low Variance Filter

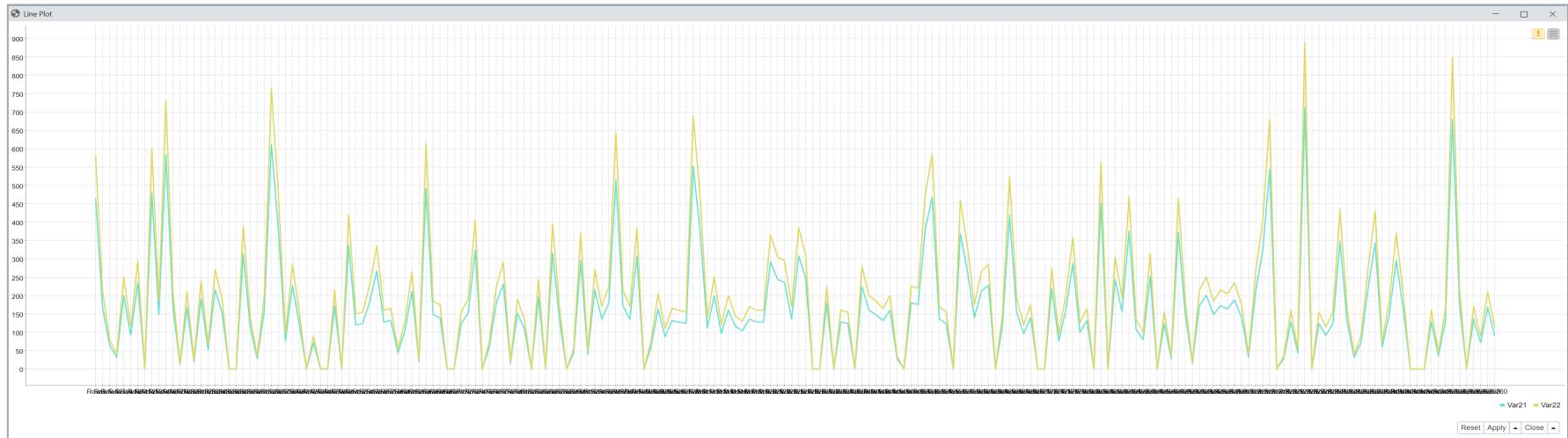
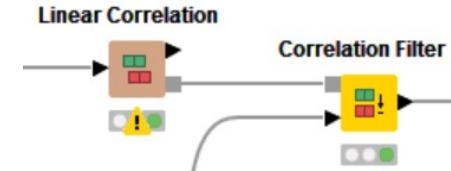


Note: requires min-max-normalization, and only works for numeric columns

- 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 (  $\text{corr}(\text{var1}, \text{var2}) > \text{threshold}$  ) => remove var1



# 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.

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

- 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  $(PC_1, PC_2, \dots PC_{k-1})$ .

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>

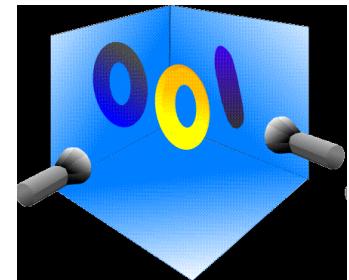
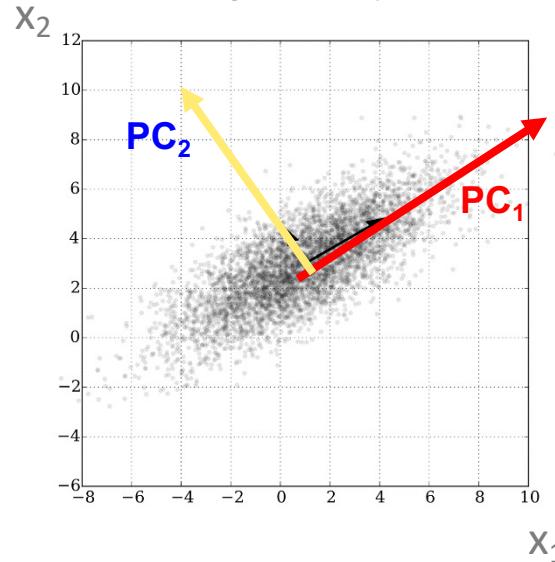


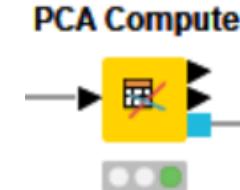
Image from Wikipedia



# 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.,  $PC_1, PC_2, \dots, PC_m$ ) components with little - if any - loss of information
- Caveats:
  - Results of PCA are quite difficult to interpret
  - Normalization required
  - Only effective on numeric columns

Dimensionality Reduction

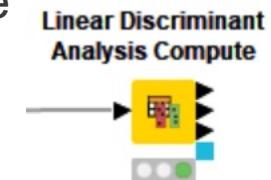


# Linear Discriminant Analysis (LDA)

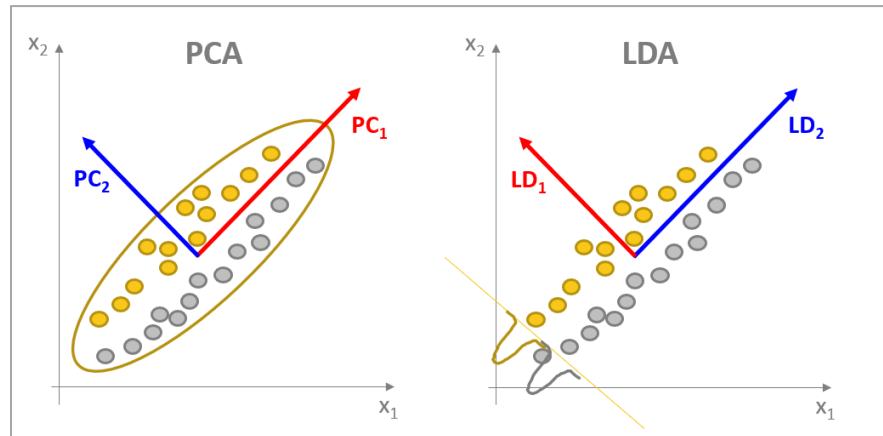
- LDA is a statistical procedure that **orthogonally** transforms the original  $n$  coordinates of a data set into a new set of  $k-1$  coordinates, called linear discriminants, where  $k$  is the number of classes in the class variable

$$(LD_1, LD_2, \dots LD_{k-1}) = LDA(X_1, X_2, \dots X_n)$$

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



- PCA : unsupervised
- LDA : supervised



# Linear Discriminant Analysis (LDA)

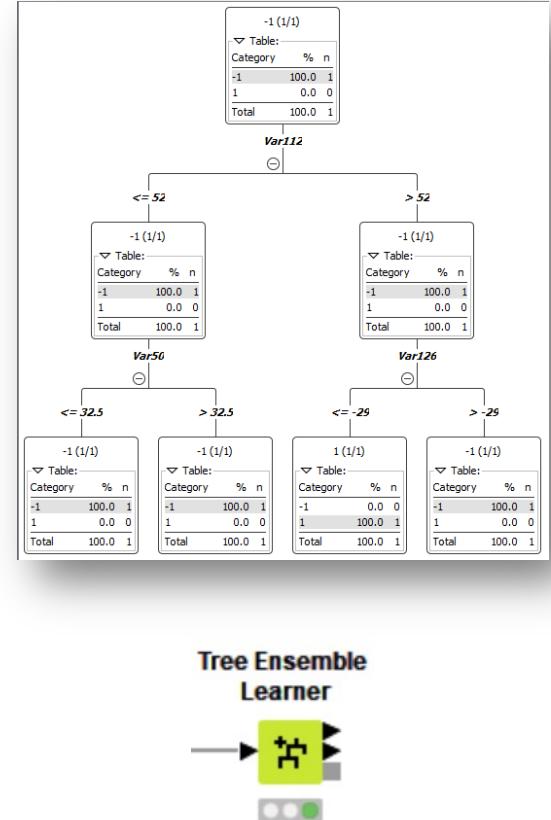
---

- $LD_1$  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  $m < k - 1$  (i.e.,  $LD_1, LD_2, \dots, LD_m$ ) discriminants with little - if any - loss of information
- Caveats:
  - Results of LDA are quite difficult to interpret
  - Normalization required
  - Only effective on numeric columns

Dimensionality Reduction

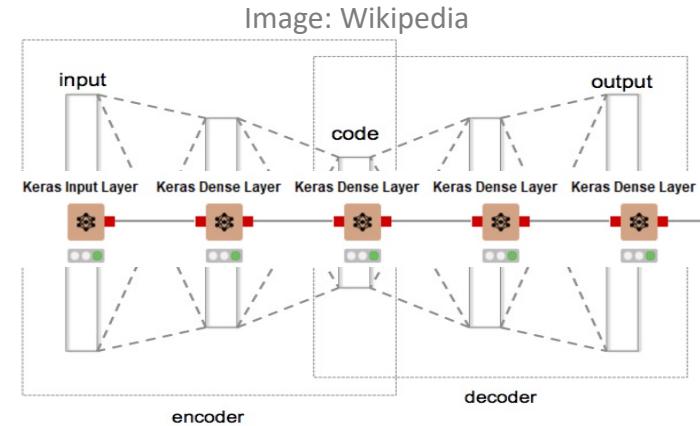
# Ensembles of Shallow Decision Trees

- 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.



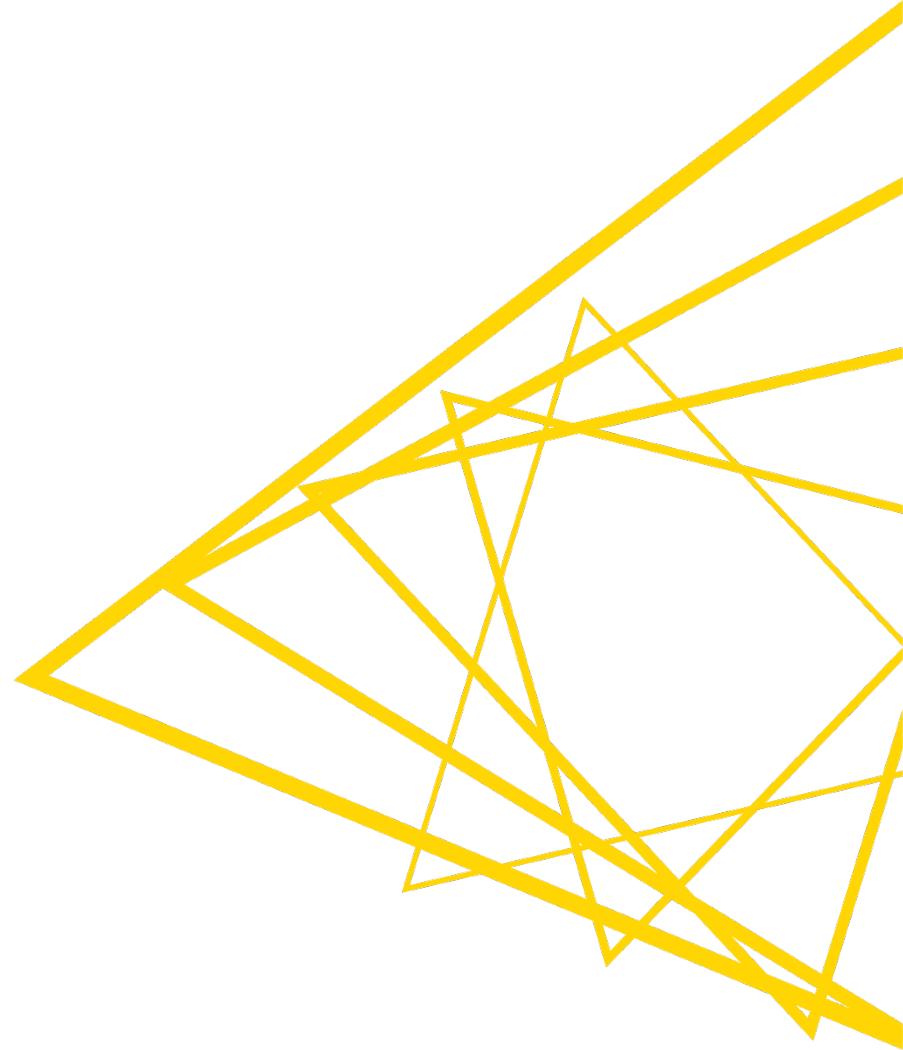
# Autoencoder

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



- 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.

# 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
0.965	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

OK Apply Cancel ?

The diagram illustrates a feature selection process using a loop. It starts with a 'Feature Selection Loop Start' node, followed by a 'Maximize accuracy' node, which then connects to a 'Feature Selection Filter' node. A yellow callout highlights this 'Feature Selection Filter' node with the text 'Do the final filtering here'. A large yellow arrow points from the 'Feature Selection Filter' node back to the 'Feature Selection Loop Start' node, indicating a feedback loop. The loop ends at a 'Feature Selection Loop End' node. A yellow box labeled 'node by state' is also present in the diagram.

# 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

**THE NEW STACK** books Podcasts Events Newsletter

Architecture Development Operations C

MACHINE LEARNING / CONTRIBUTED

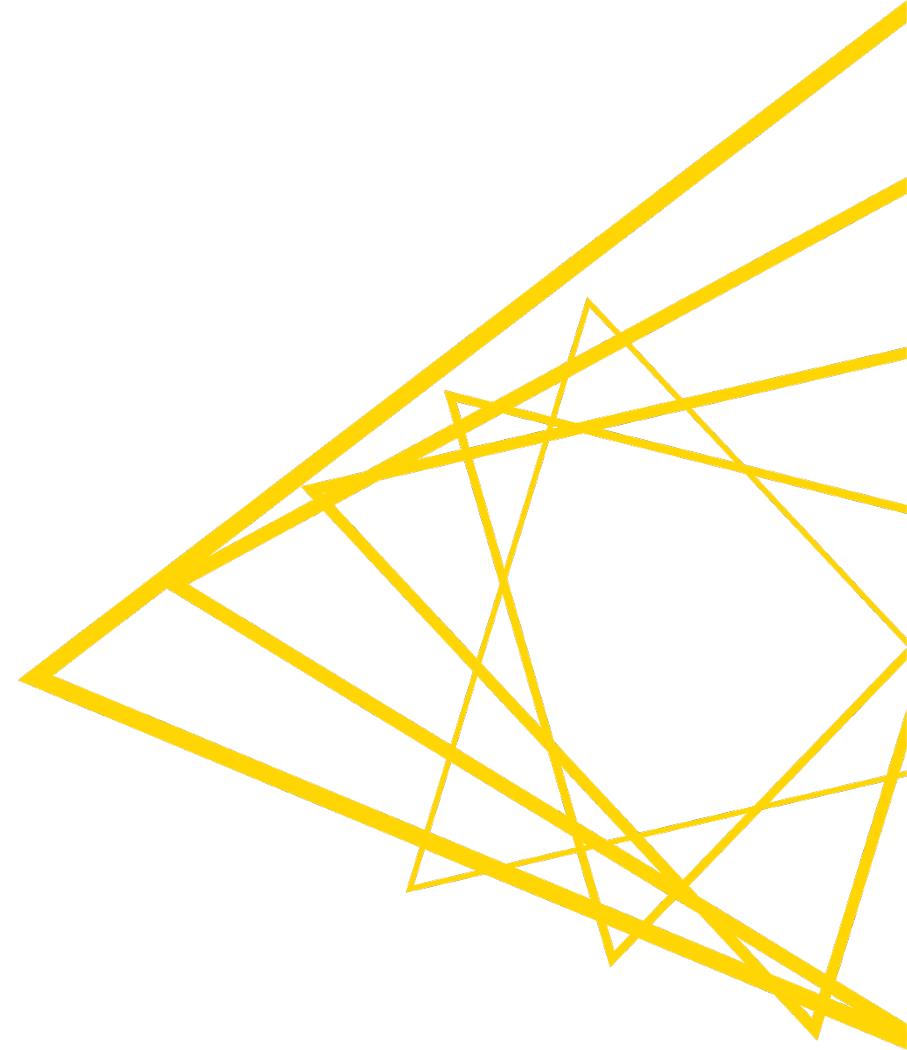
## 3 New Techniques for Data-Dimensionality Reduction in Machine Learning

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



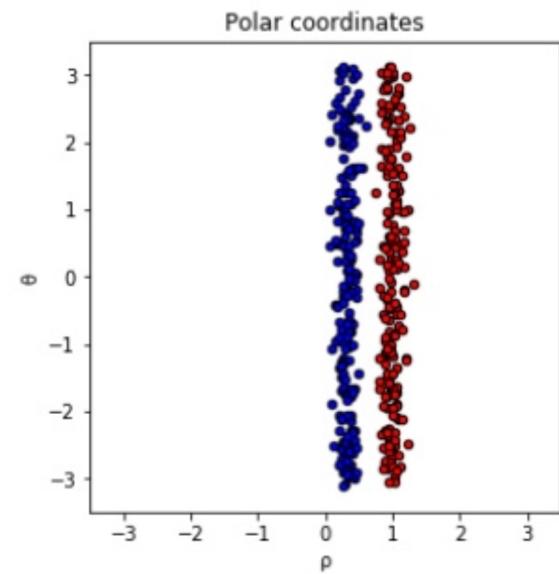
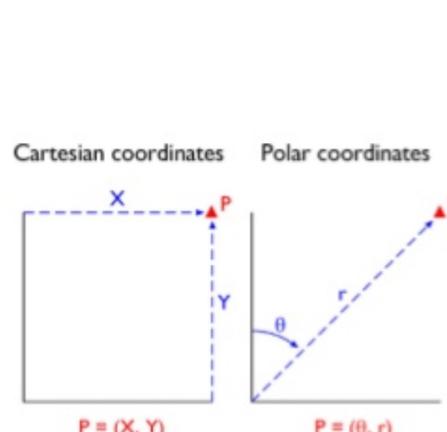
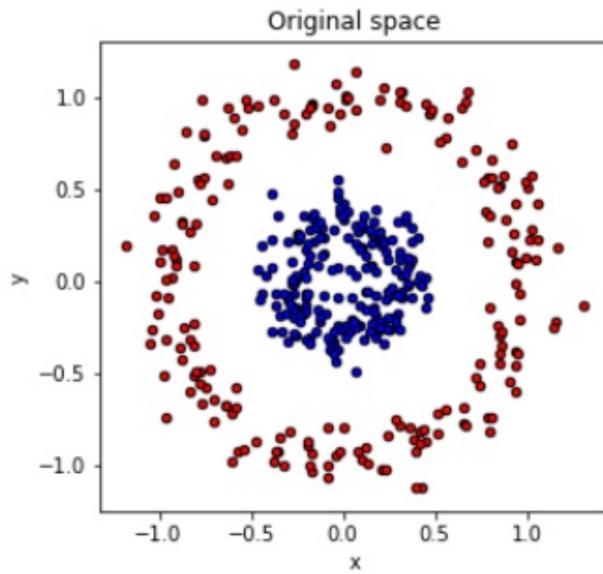
<https://thenewstack.io/3-new-techniques-for-data-dimensionality-reduction-in-machine-learning/>

# 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

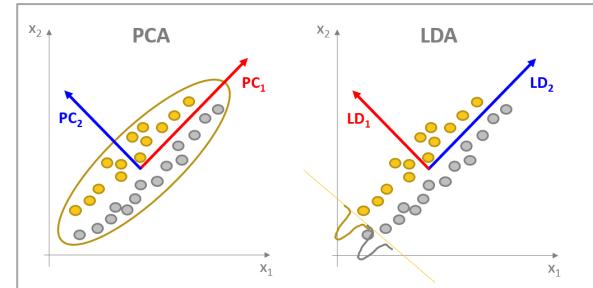
- Simple math transformations on single columns

$(e^x, x^2, x^3, \tanh(x), \log(x) , \dots)$

- Combining together multiple columns in math functions

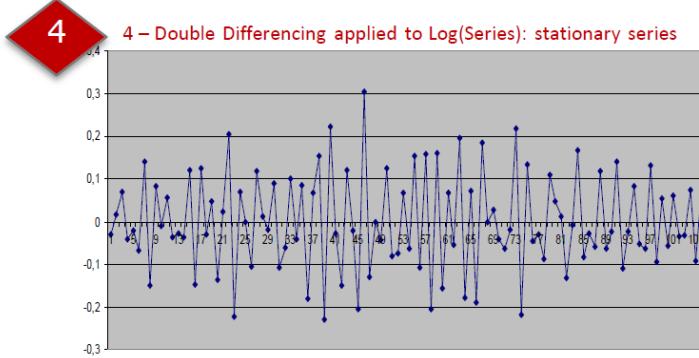
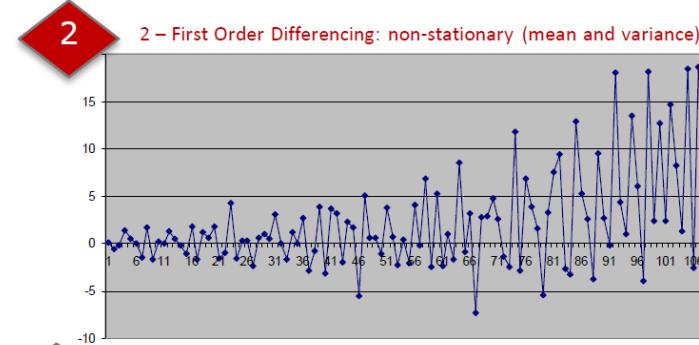
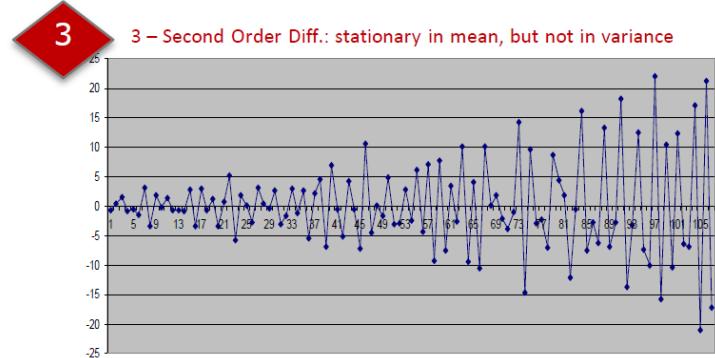
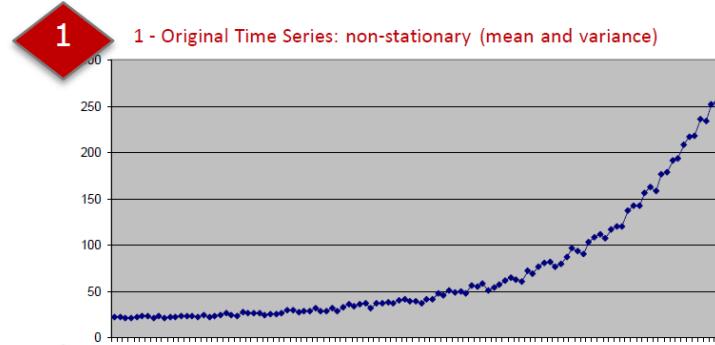
$(f(x_1, x_2, \dots xn), x_2 - x_1, \dots)$

- 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:  $\log(y'(t))$



# Confirmation of Attendance and Survey

- The [link](#) 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"

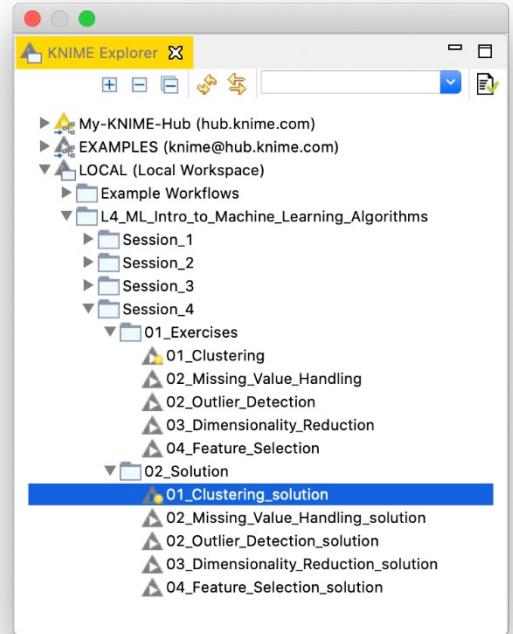
Your answer \_\_\_\_\_

Back      Submit

Never submit passwords through Google Forms.

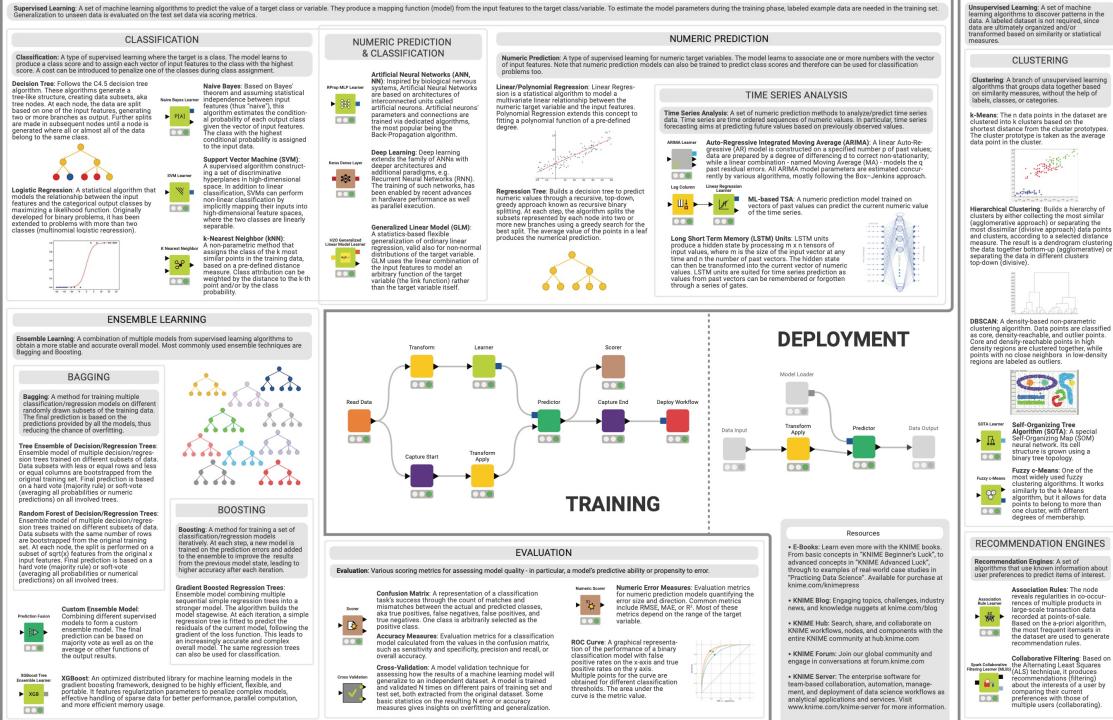
# Exercises

- Clustering
  - Goal: Cluster location data from California
  - 01\_Clustering
- Data Preparation
  - 02\_Missing\_Value\_Handling
  - 03\_Outlier\_Detection
  - 04\_Dimensionality\_Reduction
  - 05\_Feature\_Selection



# Machine Learning Cheat Sheet

## Cheat Sheet: Machine Learning with KNIME Analytics Platform



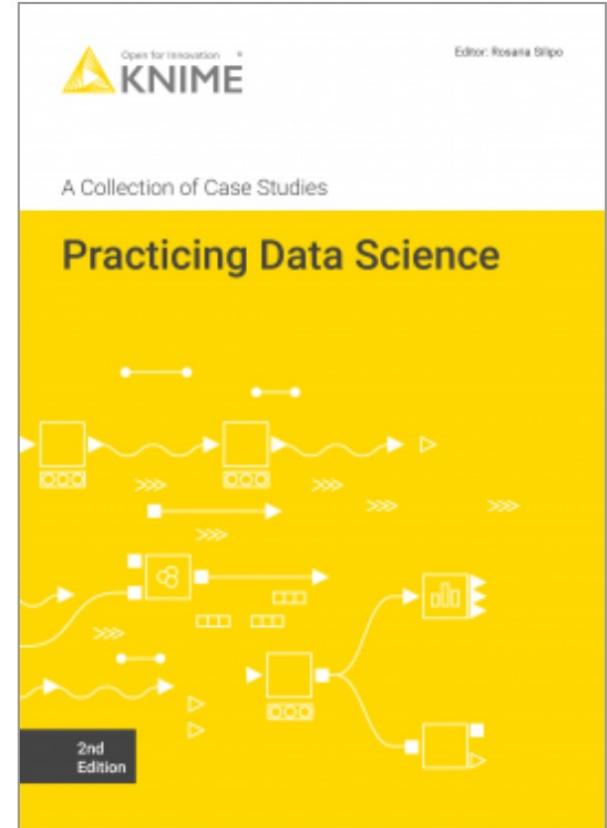
[https://www.knime.com/sites/default/files/2021-07/CheatSheet\\_ML\\_A3.pdf](https://www.knime.com/sites/default/files/2021-07/CheatSheet_ML_A3.pdf)

# KNIME Books

Course books downloadable from KNIME Press

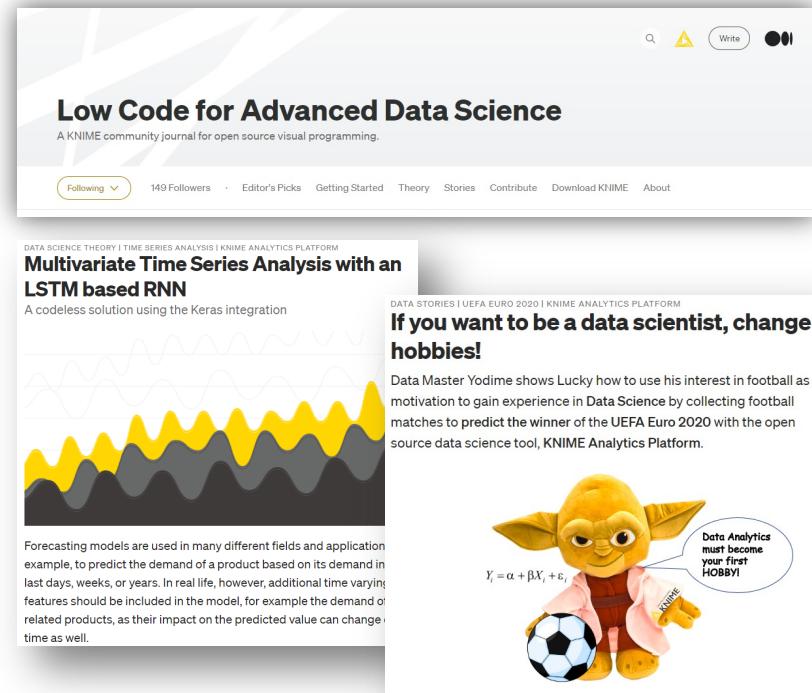
<https://www.knime.com/knimepress>

with code: L4-ML-0721



# Stay Up-To-Date and Contribute

- Follow the KNIME Community Journal on Medium  
[Low Code for Advanced Data Science](#)
- Daily content on data stories, data science theory, getting started with KNIME and more for the community by the community
- Would you like to share your data story with the KNIME community?



[Contributions](#) are always welcome!



Open for Innovation

**KNIME**

**Thank You!**

