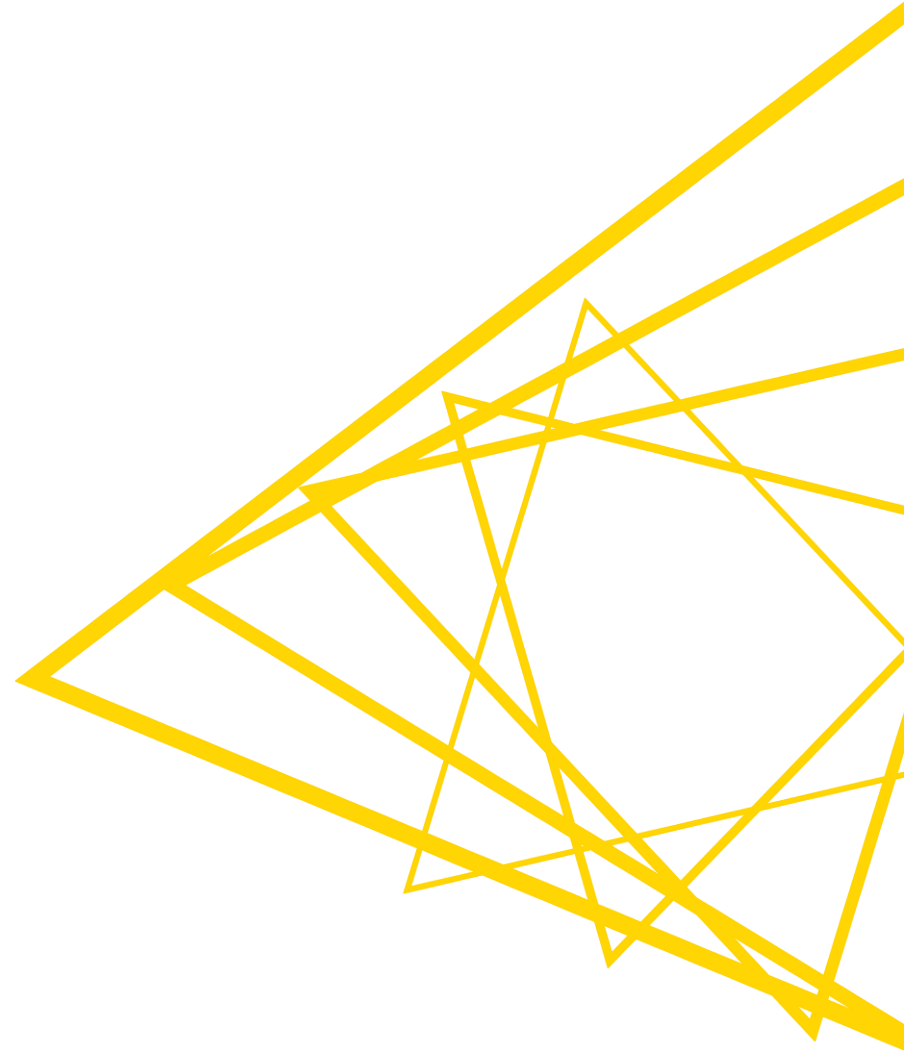


Open for Innovation

KNIME

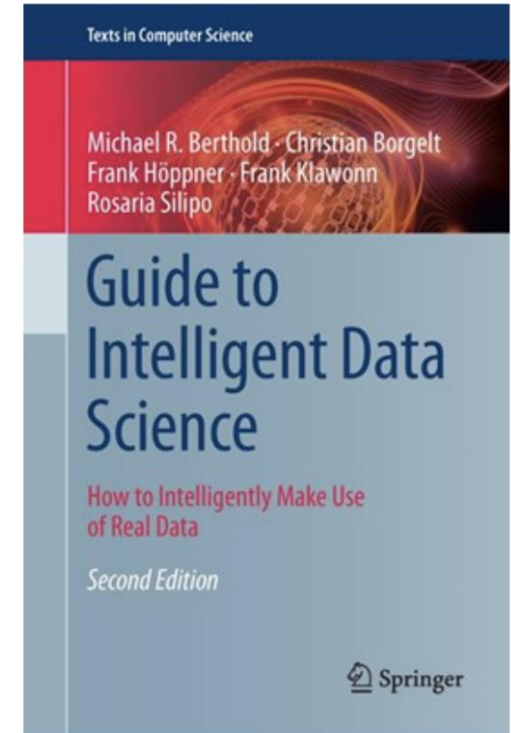
[L4-ML] Introduction to Machine Learning Algorithms

KNIME AG



Material

- Berthold, Borgelt, Höppner, Klawonn, Silipo:
Guide to Intelligent Data Science, 2nd Edition
Springer, 2020.
- 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
Springer Verlag, 2003.



What is Data Science?

[Wikipedia quoting Dhar 13, Leek 13]

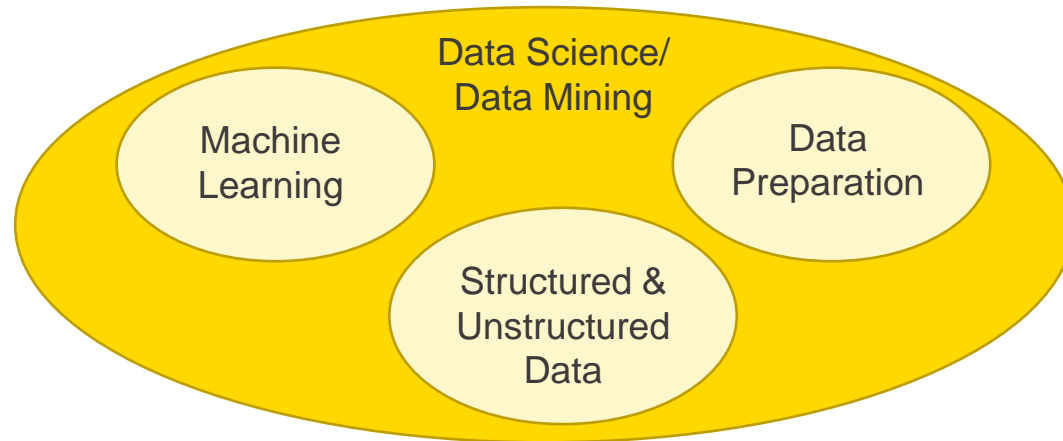
Data science is a multi-disciplinary field that uses scientific methods, processes, algorithms and systems to **extract knowledge and insights** from structured and unstructured data.

[Fayyad, Piatetsky-Shapiro & Smyth 96]

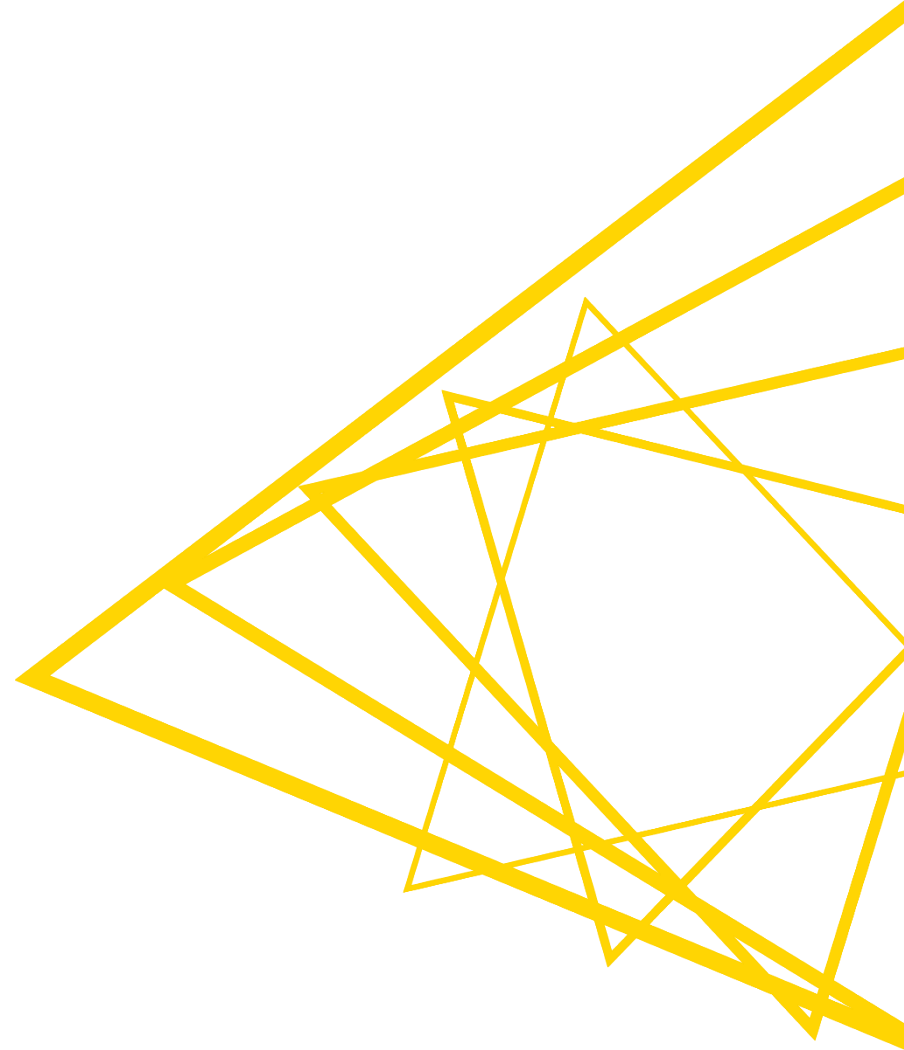
Knowledge discovery in databases (KDD) is the process of (semi-)automatic **extraction of knowledge** from databases which is *valid, previously unknown, and potentially useful*.

Some Clarity about Words

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



Use Case Collection



Exercise: Let's Collect some Use Cases

Churn Prediction



- CRM System
Data about your customer
- Demographics
 - Behavior
 - Revenues



Model



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

Customer Segmentation



- CRM System
Data about your customer
- Demographics
 - Behavior
 - Revenues



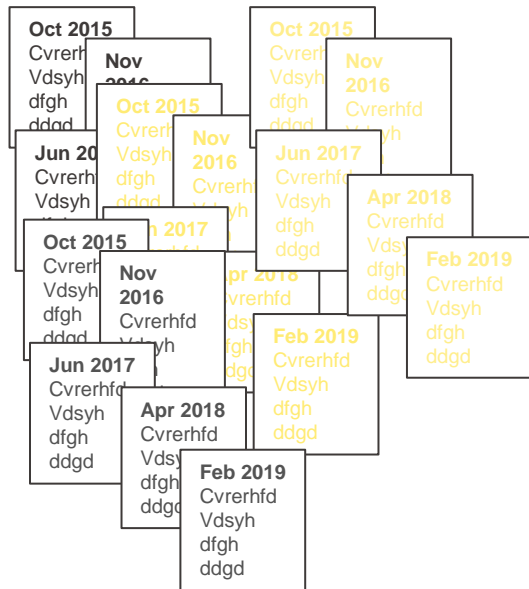
Model



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

Risk Assessment

Customer History




Model



Risk Prognosis

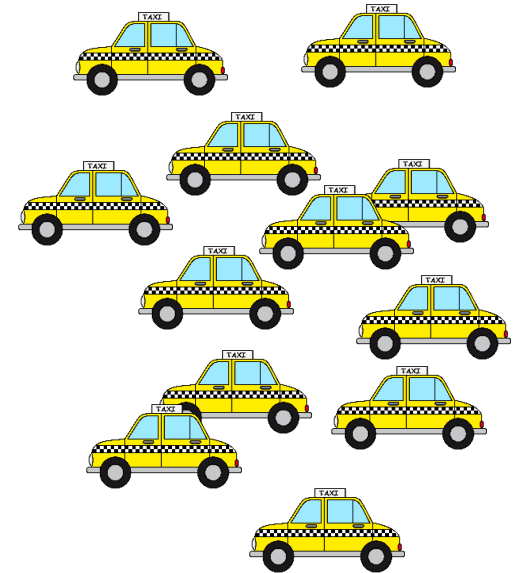
- High Risk
- Low Risk
- High Risk
- Very High Risk
- Very Low Risk
- Medium Risk
- ...

First Name	Last Name
<input type="text" value="First Name"/>	<input type="text" value="Last Name"/>
DOB	Gender
<input type="text" value="dd/MM/yy"/>	<input type="text" value="Gender"/>
Employment	Credit Risk
<input type="text" value="Employment"/>	<input type="text" value="High"/>



Demand Prediction

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



Model

Recommendation Engines / Market Basket Analysis



Model



Recommendation



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 [\[User\]](#) on May 29, 2017

Color: Gold | **Verified Purchase**

This practically new beautiful phone well exceeded my expectations!



★☆☆☆☆ **One Star**

By [\[User\]](#) on August 3, 2016

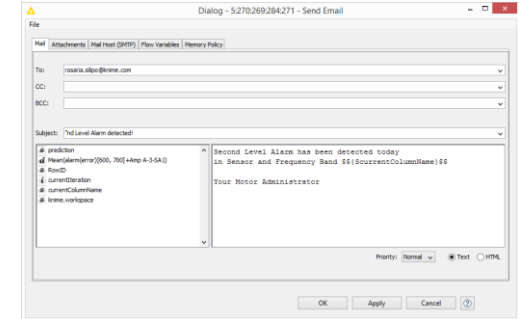
Color: Black Onyx | **Verified Purchase**

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?

- Input features
- Input attributes
- Independent variables

$$\mathbf{X} = (x_1, x_2, \dots, x_n)$$



y

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

Model parameters

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

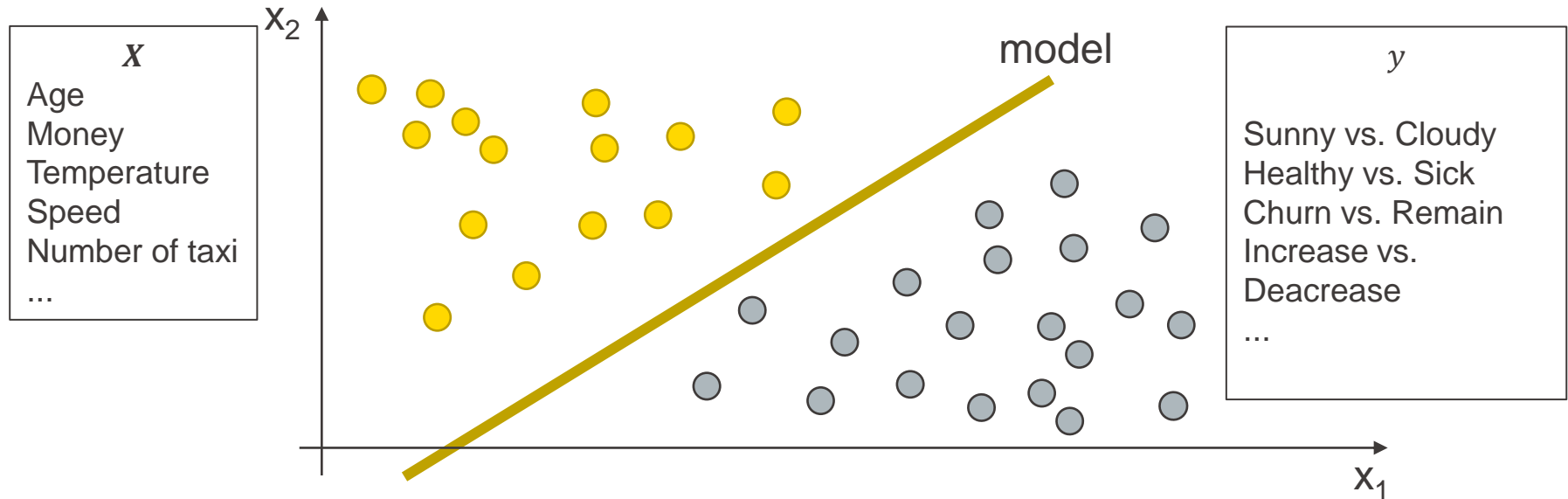
A learning algorithm adjusts (learns) the model parameters $\boldsymbol{\beta}$ throughout a number of iterations to maximize/minimize a likelihood/error function on y .

Algorithm Training / Learning

- The model ***learns / is trained*** during the ***learning / training*** phase to produce the right answer y (a.k.a., label)
- That is why ***machine learning*** 😊
- Many different algorithms for three ways of learning:
 - Supervised
 - Unsupervised
 - Semi-supervised

Supervised Learning

- $X = (x_1, x_2)$ 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 $y = \{\text{label 1}, \dots, \text{label n}\}$ 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

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

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

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

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

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

Numerical Predictions (Regression)

$y = \text{temperature}$

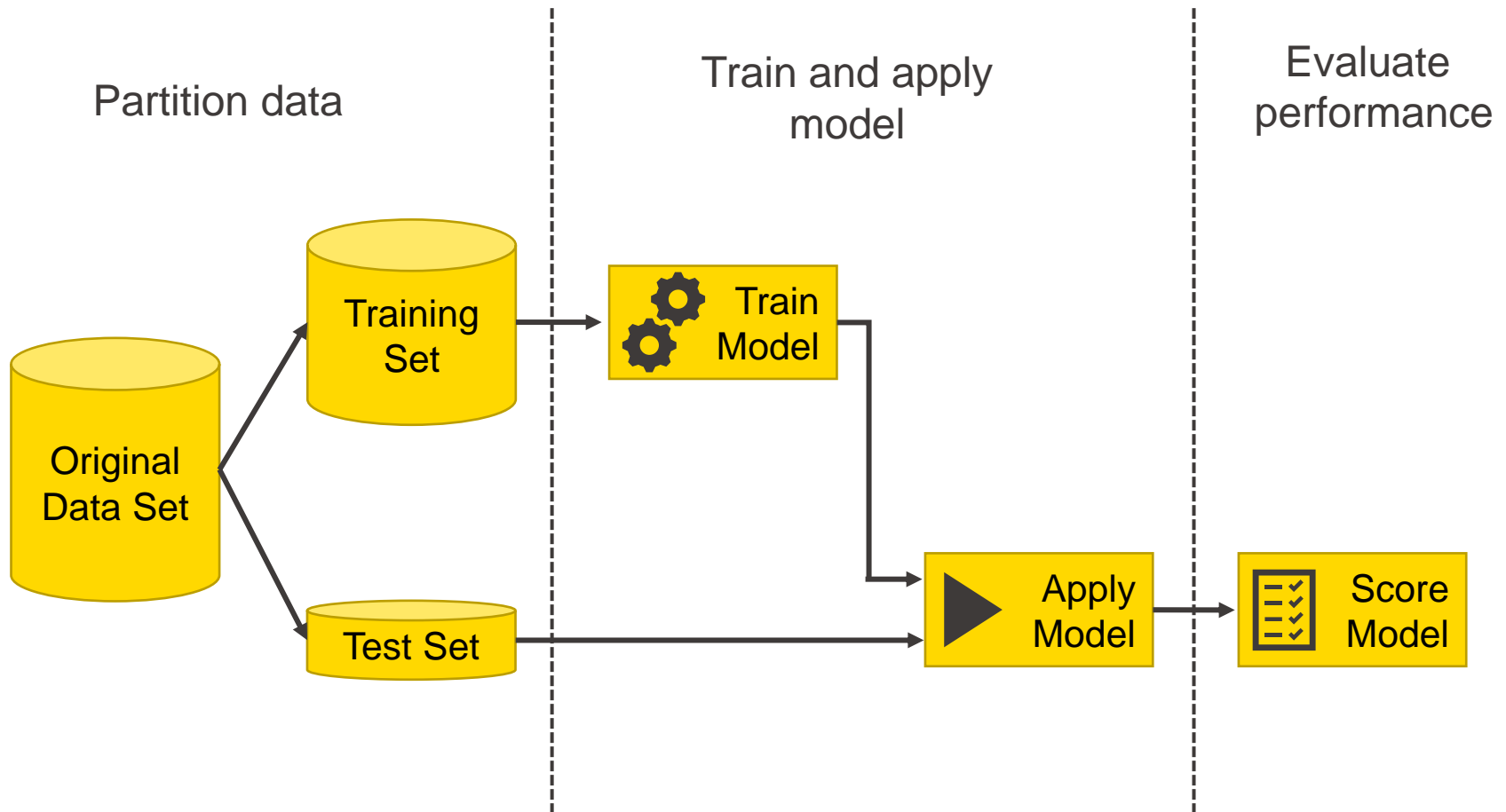
$y = \text{number of visitors}$

$y = \text{number of kW}$

$y = \text{price}$

$y = \text{number of hours}$

Process Overview for Supervised Learning

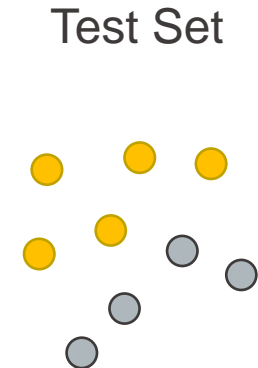


Training vs. Testing: Partitioning

- **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)

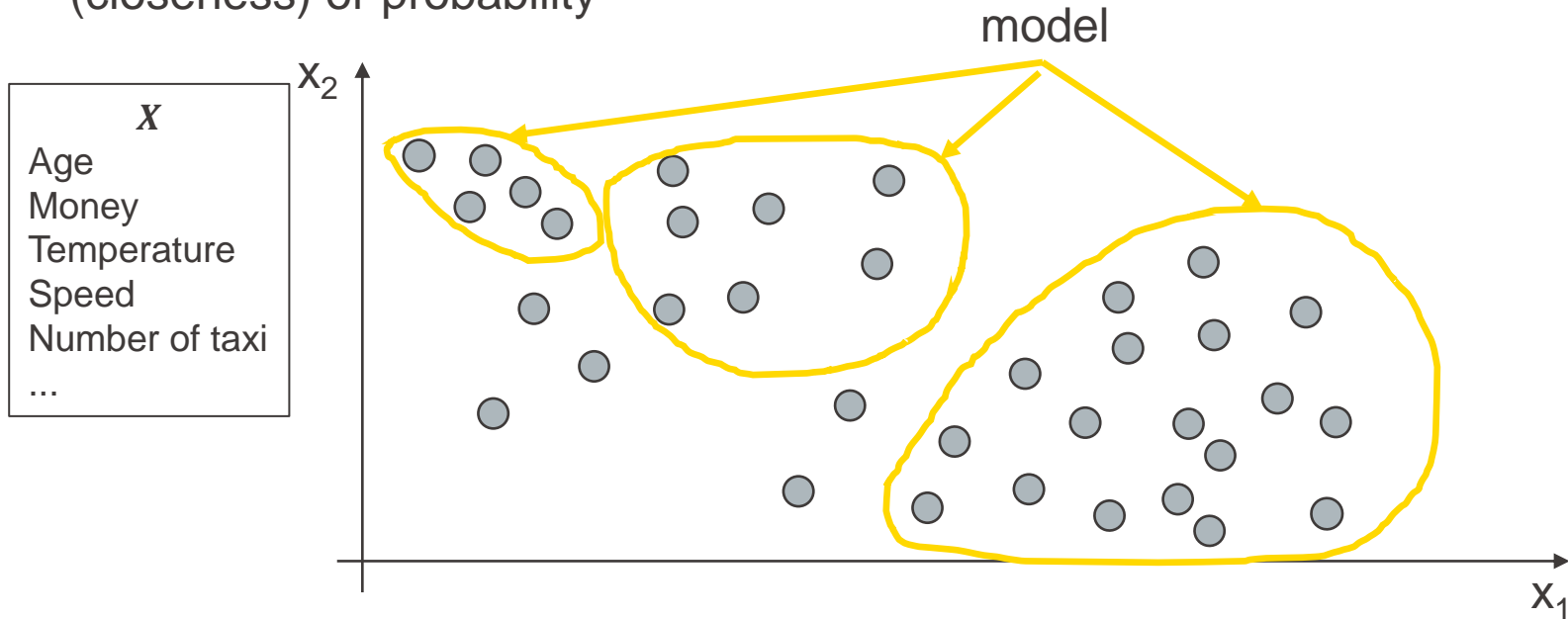


* sometimes



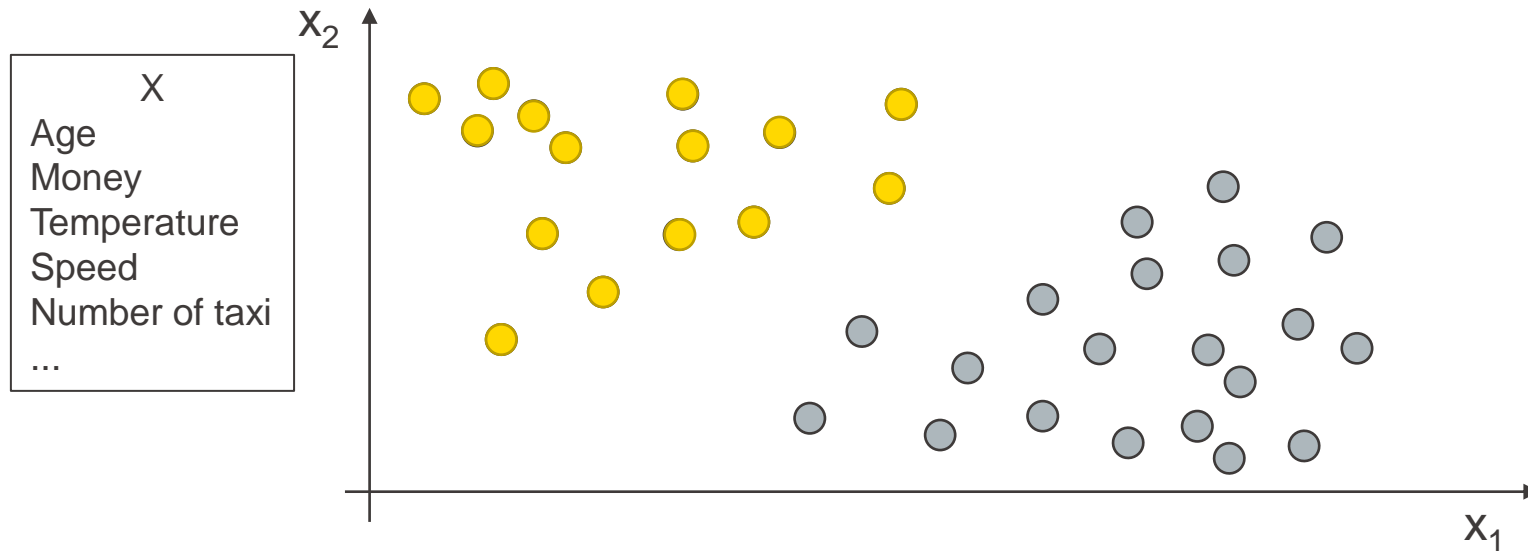
Unsupervised Learning

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



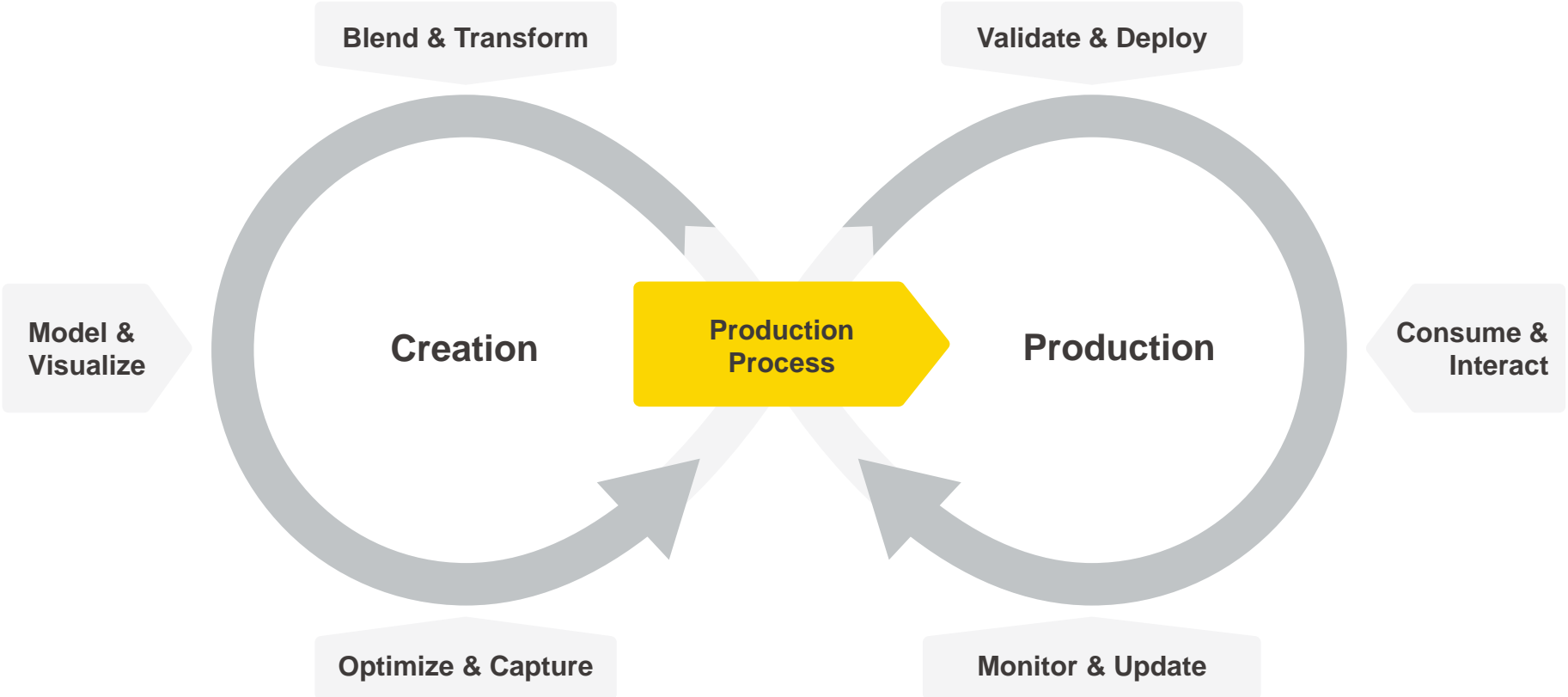
The CRISP-DM Cycle



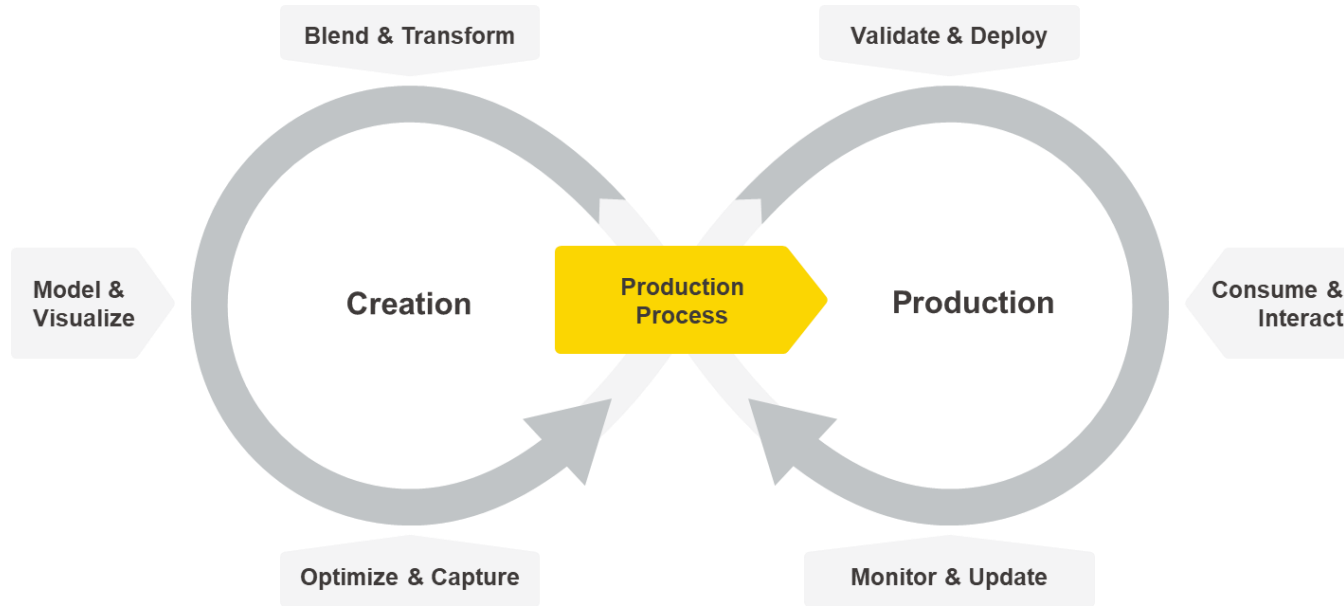
The screenshot shows the Wikipedia page for 'Cross Industry Standard Process for Data Mining'. The URL in the browser is https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining. The article text includes: 'From Wikipedia, the free encyclopedia', 'Cross Industry Standard Process for Data Mining, commonly known by its acronym CRISP-DM,^[1] is a data website (KDNuggets) in 2002, 2004, 2007 and 2014 show that it was the leading methodology used by industry. Many people reported using CRISP-DM. A review and critique of data mining process models in 2009 called the models include Kurgan and Musilek's 2006 review,^[7] and Azevedo and Santos' 2008 comparison of CRISP-DM (SIG) responsible along with the website has long disappeared (see History of CRISP-DM). In 2015, IBM Corporation released a new methodology called *Analytics Solutions Unified Method for Data Mining*.' The 'Contents' section lists: 1 Major phases, 2 History, 3 References, 4 External Links. The 'Major phases' section begins with 'CRISP-DM breaks the process of data mining into six major phases.'^[10]

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

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 Business Hub

Team Collaboration

End User Applications

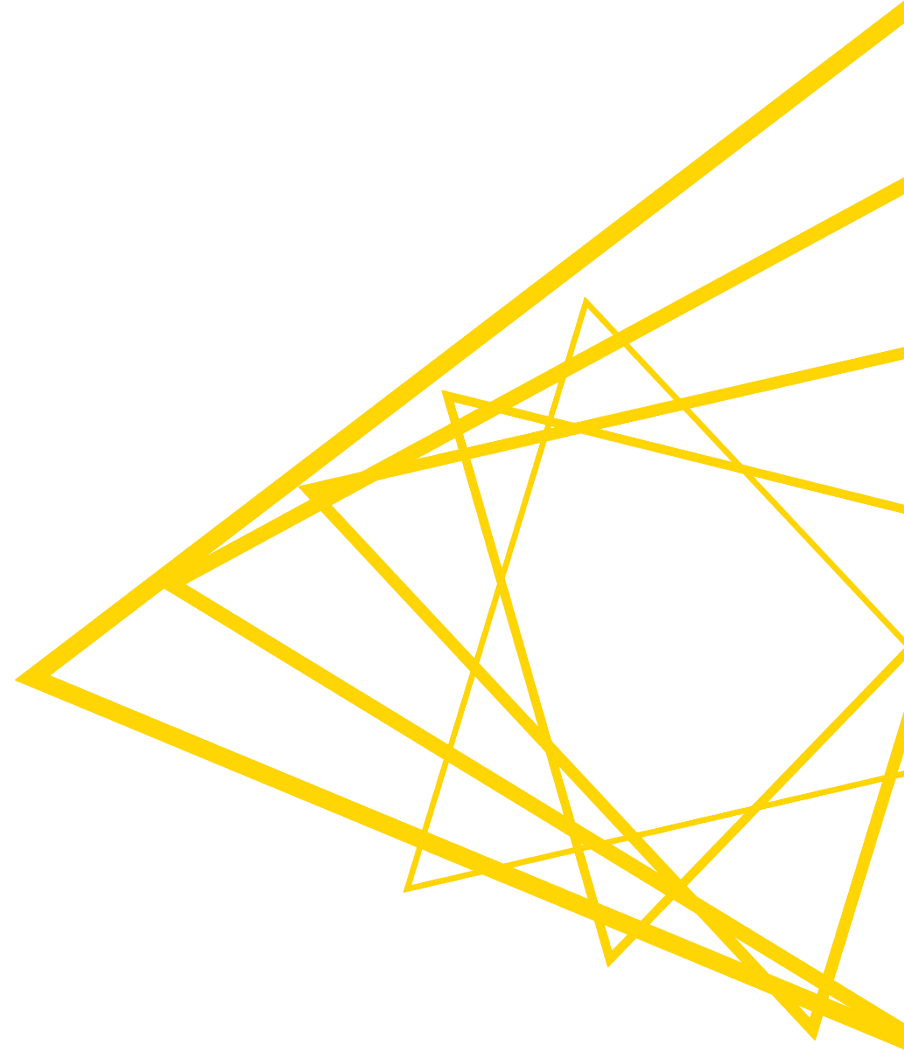
API Services

Managed Execution

Exercise

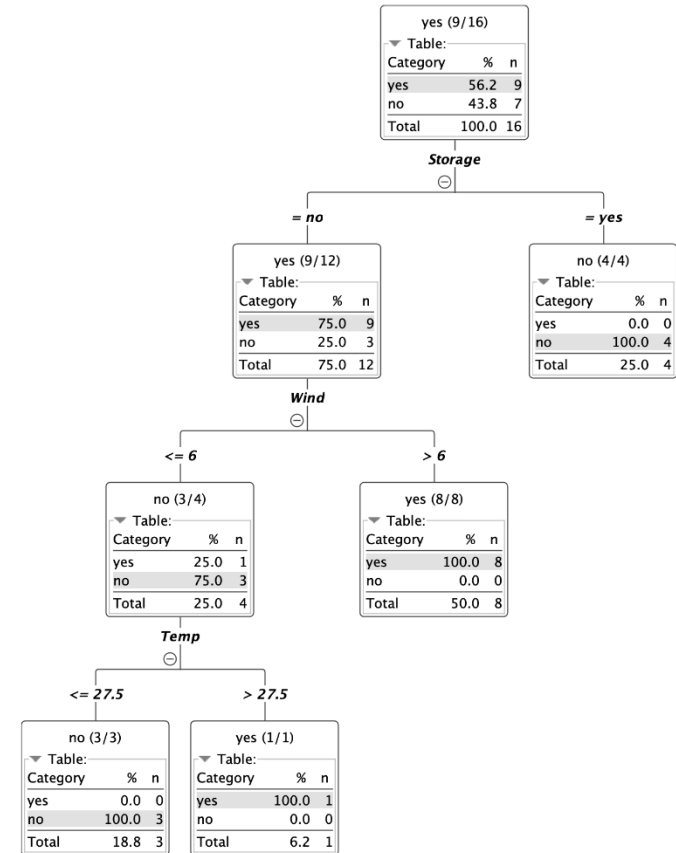
- Let's recap the different types of data science problems from a technical perspective
- Let's match the collected use cases to different data science problems

Decision Tree Algorithm

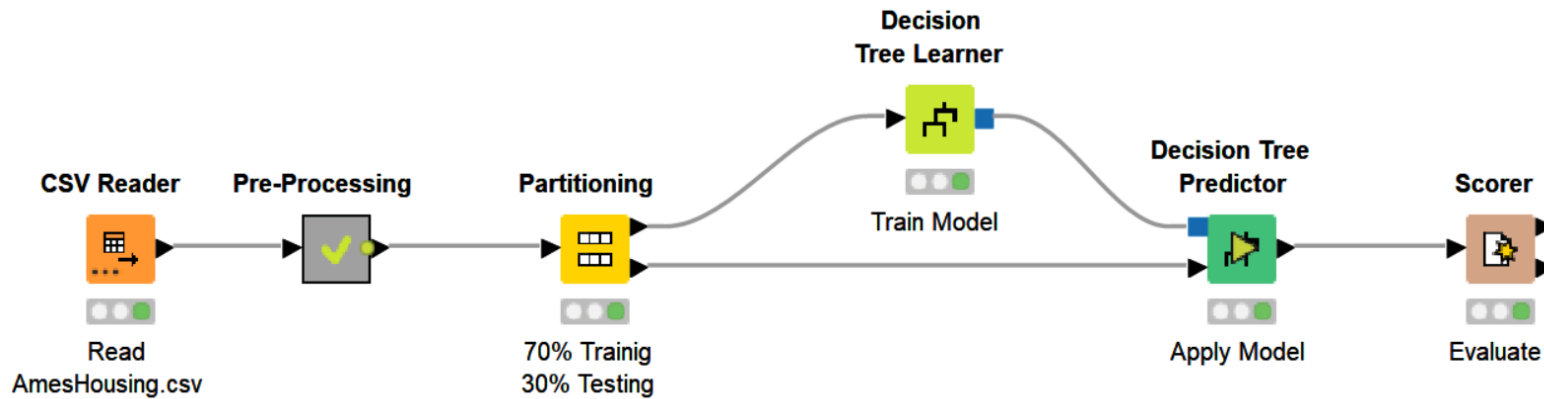


Goal: A Decision Tree

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

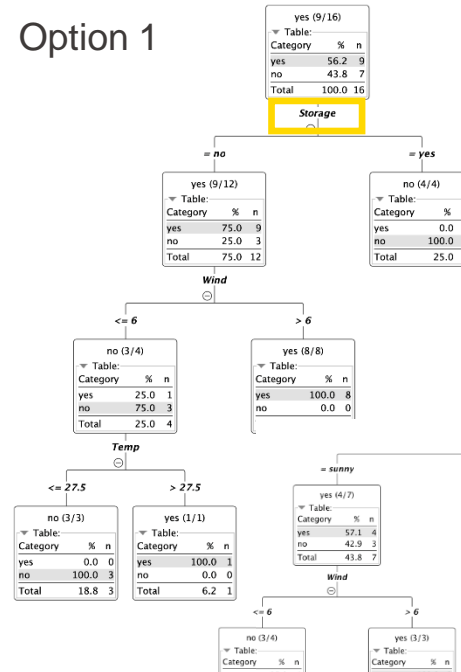


Goal: A Decision Tree

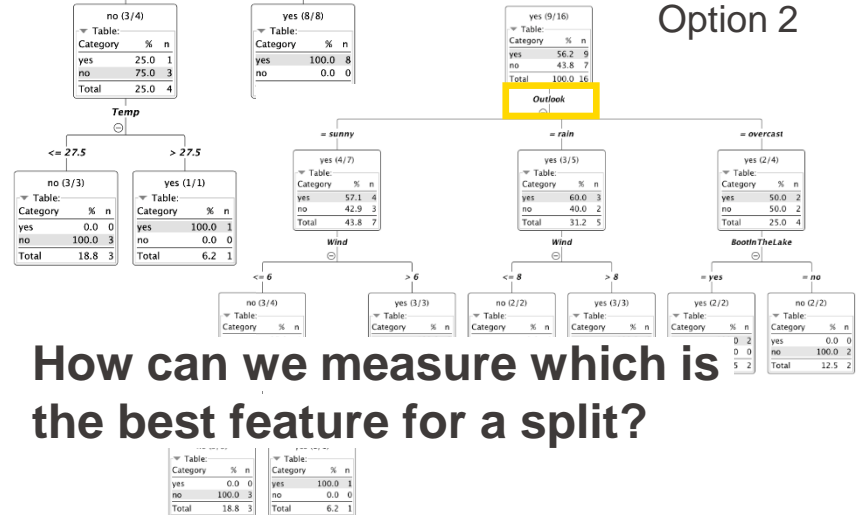
Outlook	Wind	Temp	Storage	Sailing
sunny	3	30	yes	yes
sunny	3	25	yes	no
rain	12	15	yes	yes
overcast	15	2	no	no
rain	16	25	yes	yes
sunny	14	18	yes	yes
rain	3	5	no	no
sunny	9	20	yes	yes
overcast	14	5	no	no
sunny	1	7	no	no
rain	4	25	yes	no
rain	14	24	yes	yes
sunny	11	20	yes	yes
sunny	2	18	yes	no
overcast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



Option 1



Option 2



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

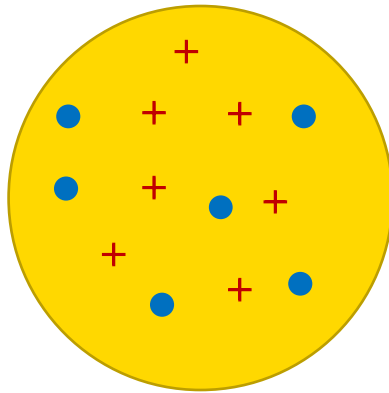
Possible Split Criterion: Gain Ratio

Based on entropy = measure for information / uncertainty

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

$$p_1 = 7/13$$

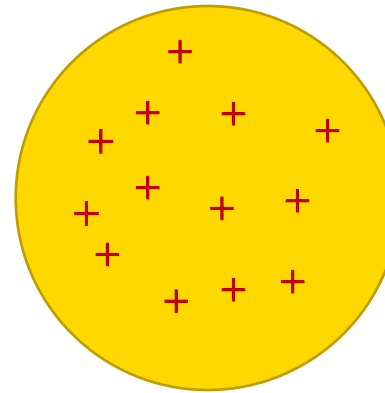
$$p_2 = 6/13$$



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

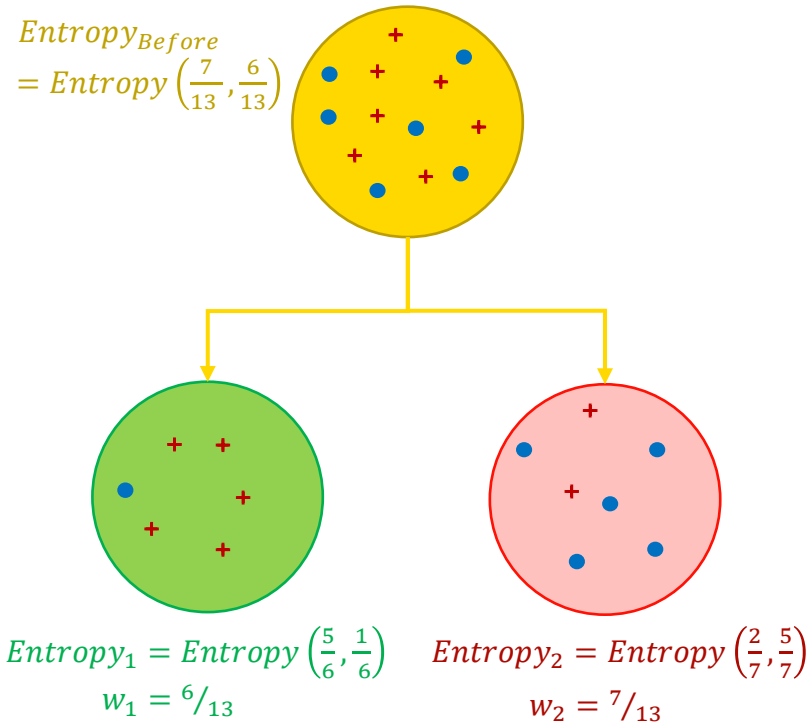
$$p_1 = 13/13 = 1$$

$$p_2 = 0/13 = 0$$



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

Possible Split Criterion: Gain Ratio



Split criterion:

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

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

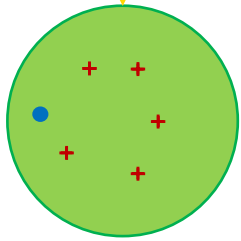
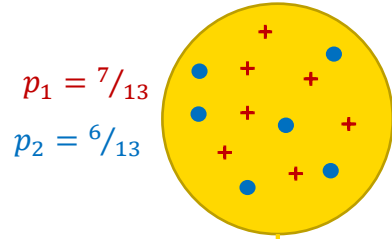
Next splitting feature: Feature with highest *Gain*

Problem: Favors features with many different values

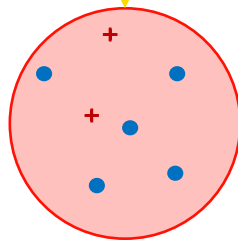
Solution: *Gain Ratio*

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

Possible Split Criterion: Gini Index



$$Gini_1 = Gini(5/6, 1/6)$$
$$w_1 = 6/13$$



$$Gini_2 = Gini(2/7, 5/7)$$
$$w_2 = 7/13$$

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:

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

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

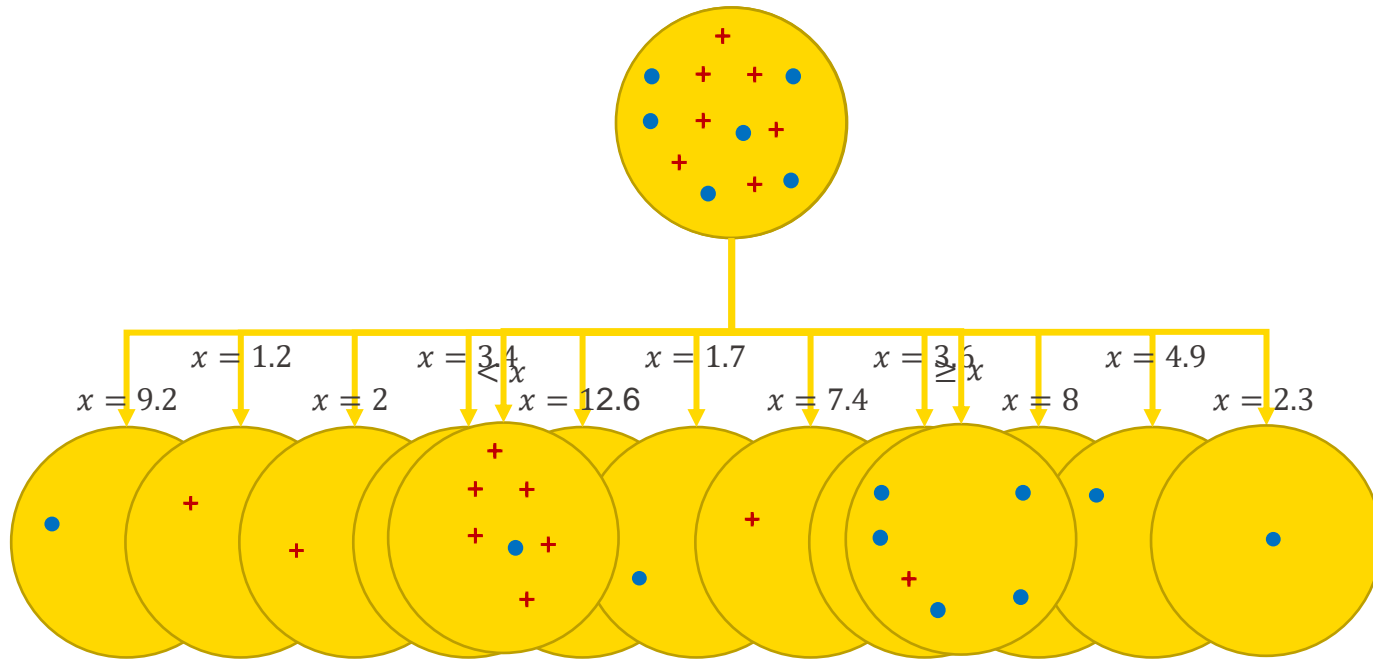
Next splitting feature:

Feature with lowest $Gini_{Index}$

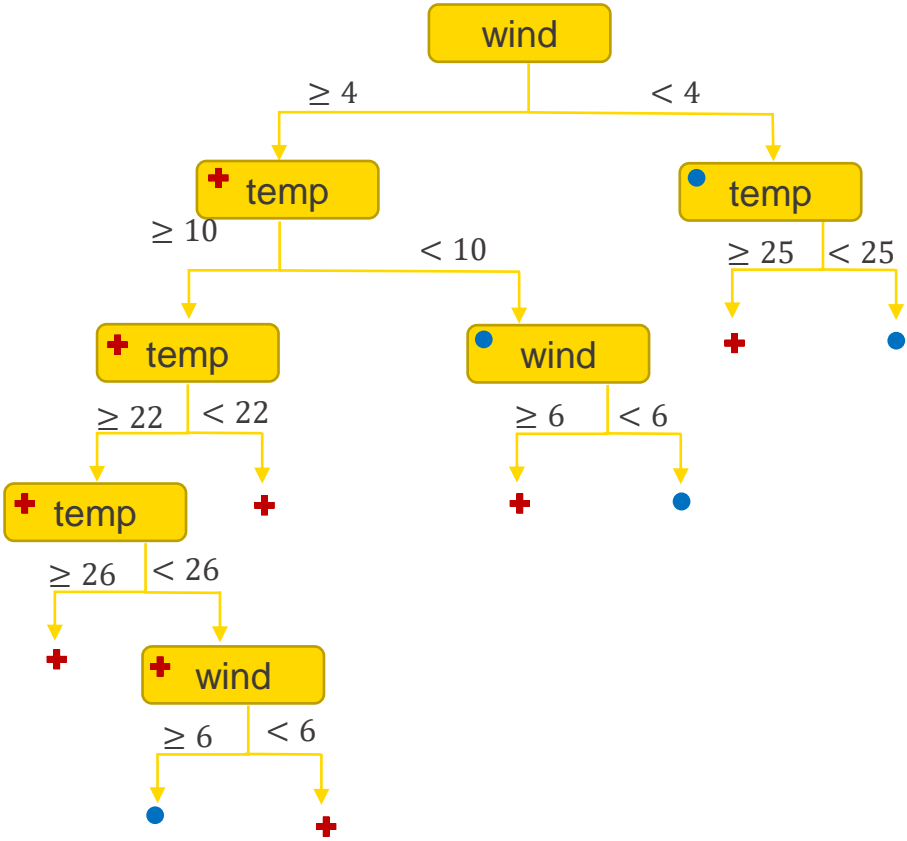
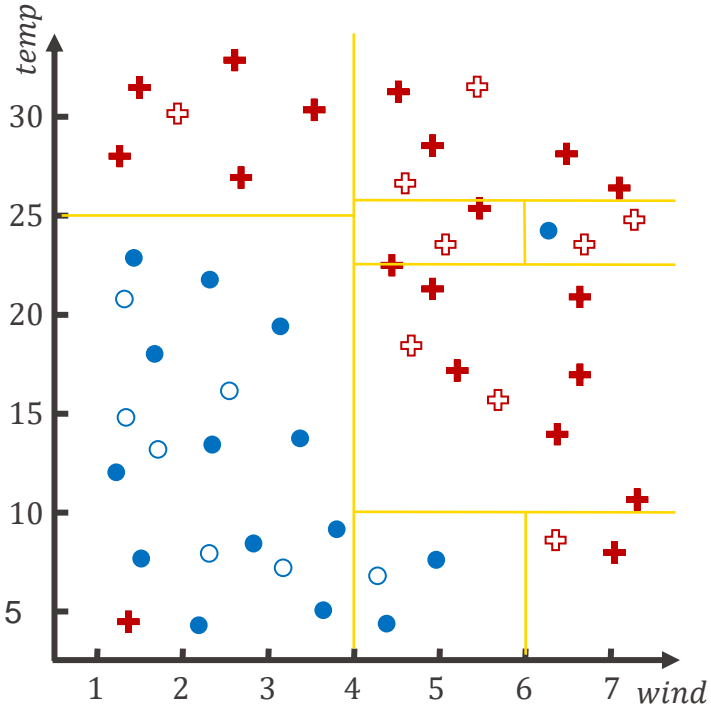
What happens for numerical Input Features?

Subset for each value? – NO

Solution: Binary splits

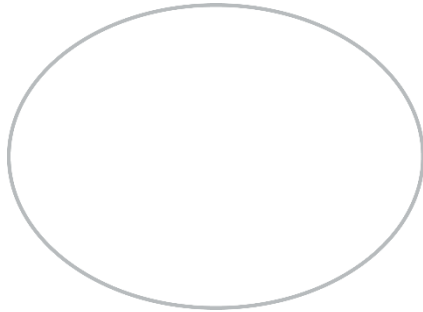


The Deeper the Better?!



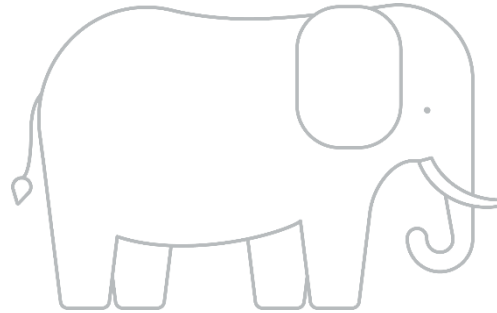
Overfitting vs Underfitting

Underfitted



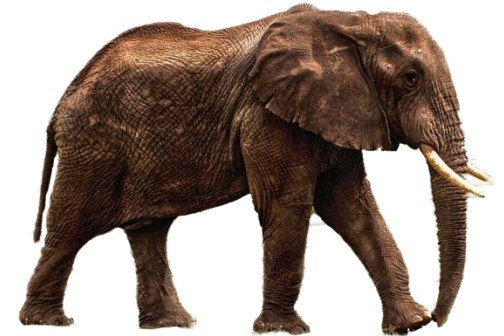
Model overlooks
underlying
patterns in the
training set

Generalized



Model captures
correlations in the
training set

Overfitted



Model memorizes
the training set
rather than finding
underlying patterns

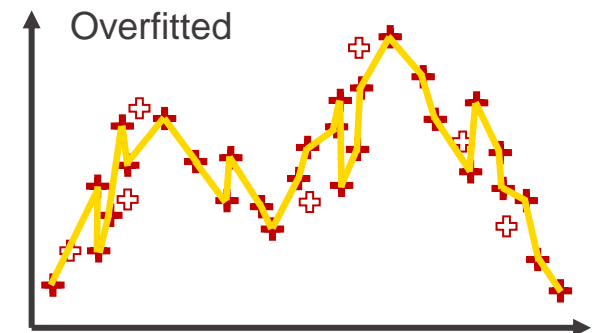
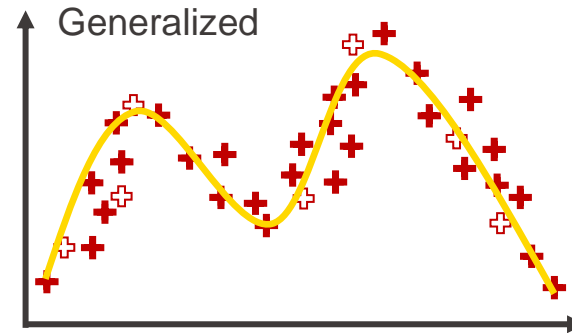
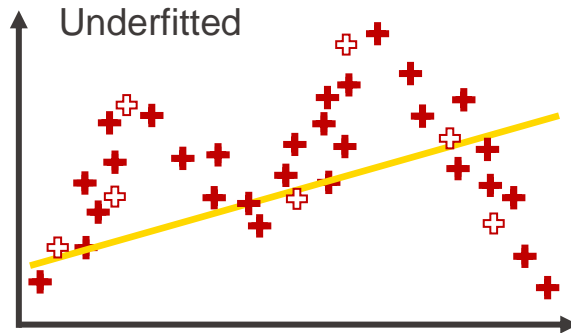
Overfitting vs Underfitting

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



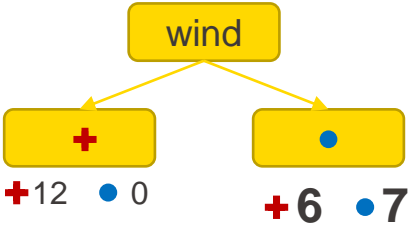
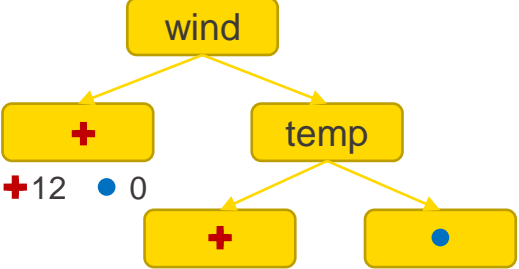
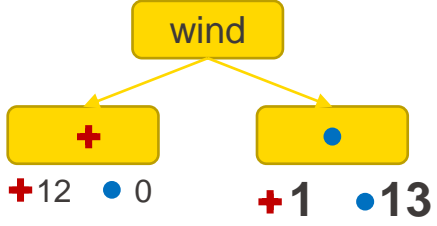
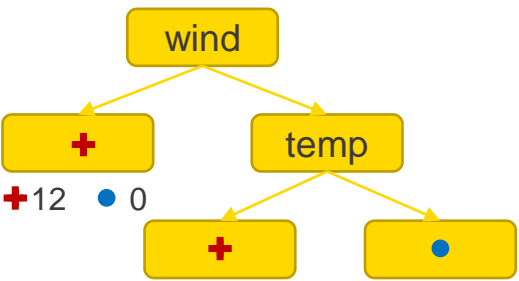
Controlling the Tree Depth

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

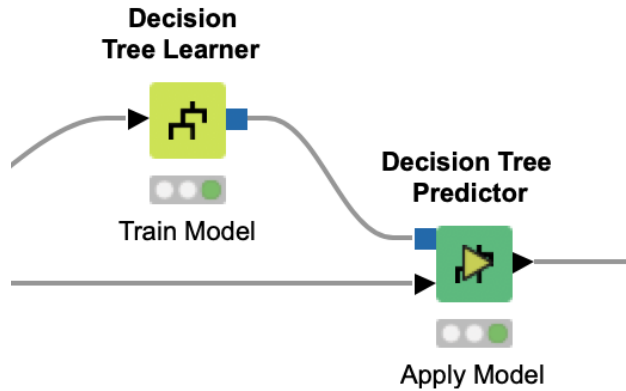
Pruning	Early stopping
Idea: Cut branches that seem as result from overfitting	Idea: Define a minimum size for the tree leaves
Techniques: <ul style="list-style-type: none">• Reduced Error Pruning• Minimum description length	

Pruning - Minimum Description Length Pruning (MDL)

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

	Tree 1	Tree 2	Note
Example 1			Many misclassified samples in tree 1 → $\text{DL}(\text{Tree 1}) > \text{DL}(\text{Tree 2})$ → Select Tree 2
Example 2			Only 1 misclassified sample in tree 1 → $\text{DL}(\text{Tree 1}) < \text{DL}(\text{Tree 2})$ → Select Tree 1

Applying the Model – What are the Outputs?



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

File Hilite Navigation View

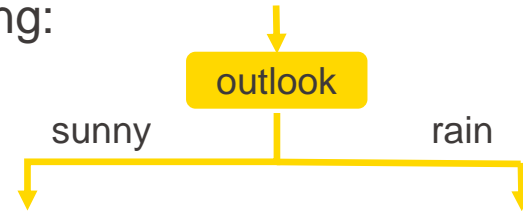
Table "default" - Rows: 879 Spec - Columns: 82 Properties Flow Variables

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

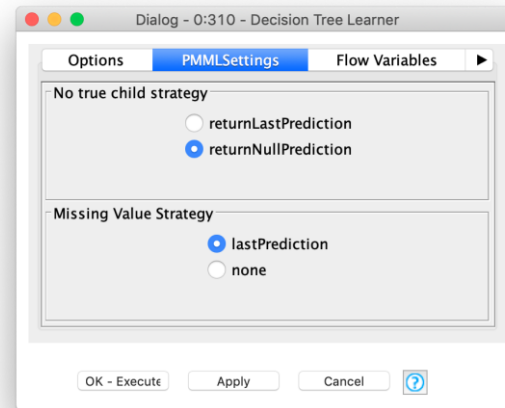
No True Child Strategy

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

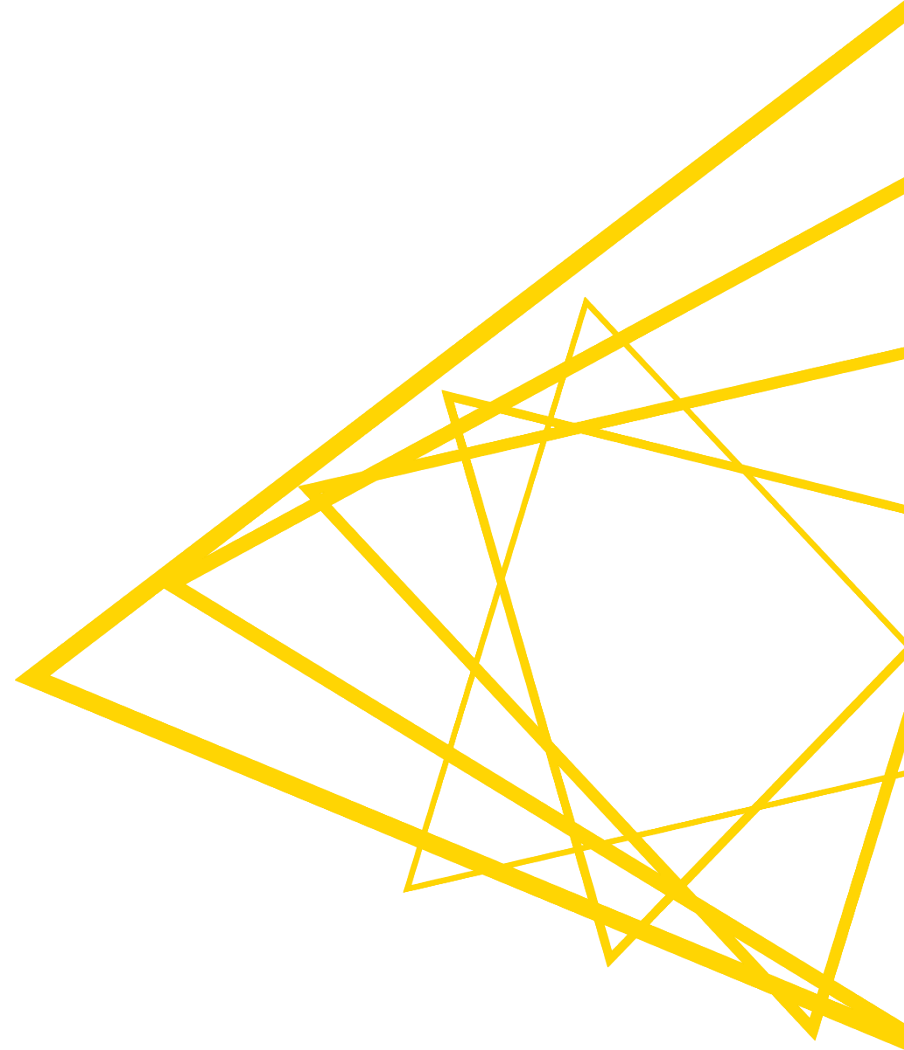
Training:



What happens with outlook = overcast?



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

Scorer (Java Script)



Numeric Scorer



ROC Curve



Overall Accuracy

- Definition:

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

- 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

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

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

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

- 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

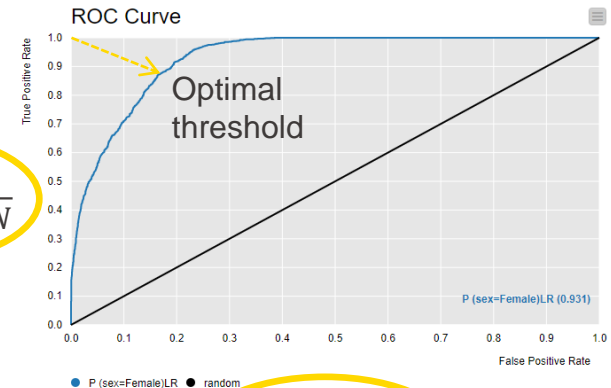
Use these four statistics to calculate other evaluation metrics, such as overall accuracy, true positive rate, and false positive rate

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)

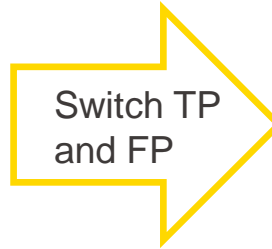
$$TPR = \frac{TP}{TP + FN}$$



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

Cohen's Kappa (κ) 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

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

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

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

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

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

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

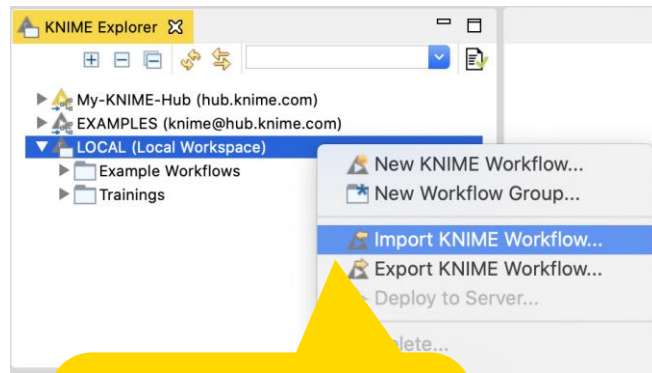
Exercise: Decision_Tree_exercise

- 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.

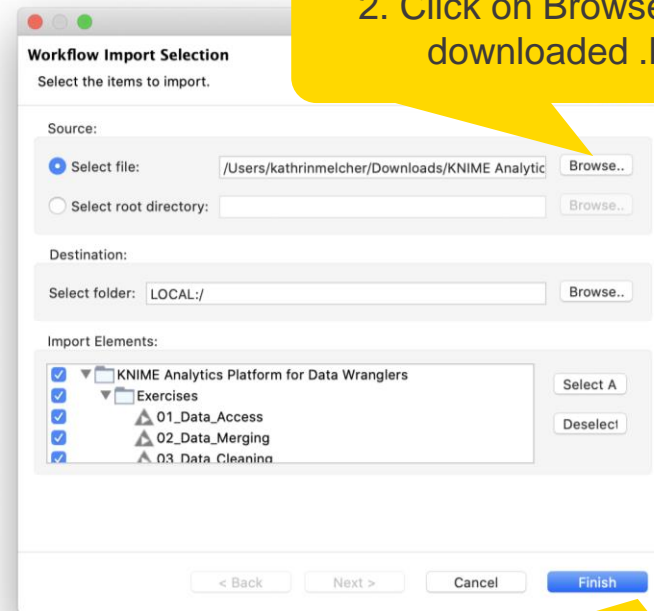
You can download the training workflows from the KNIME Community Hub:
<https://hub.knime.com/knime/spaces/Education/latest/Courses/>

Exercise Session 1

- Import the course material to KNIME Analytics Platform



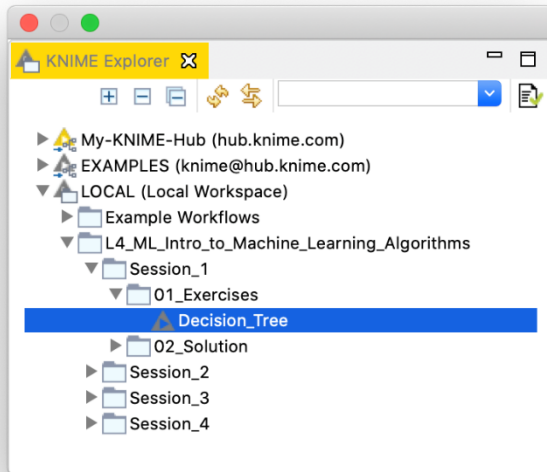
1. Right click on LOCAL and select Import KNIME Workflow....



2. Click on Browse and select downloaded .knar file

3. Click on Finish

Exercise: Decision_Tree_exercise

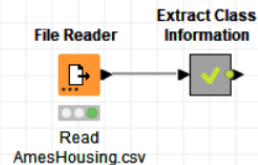


Use Case Description

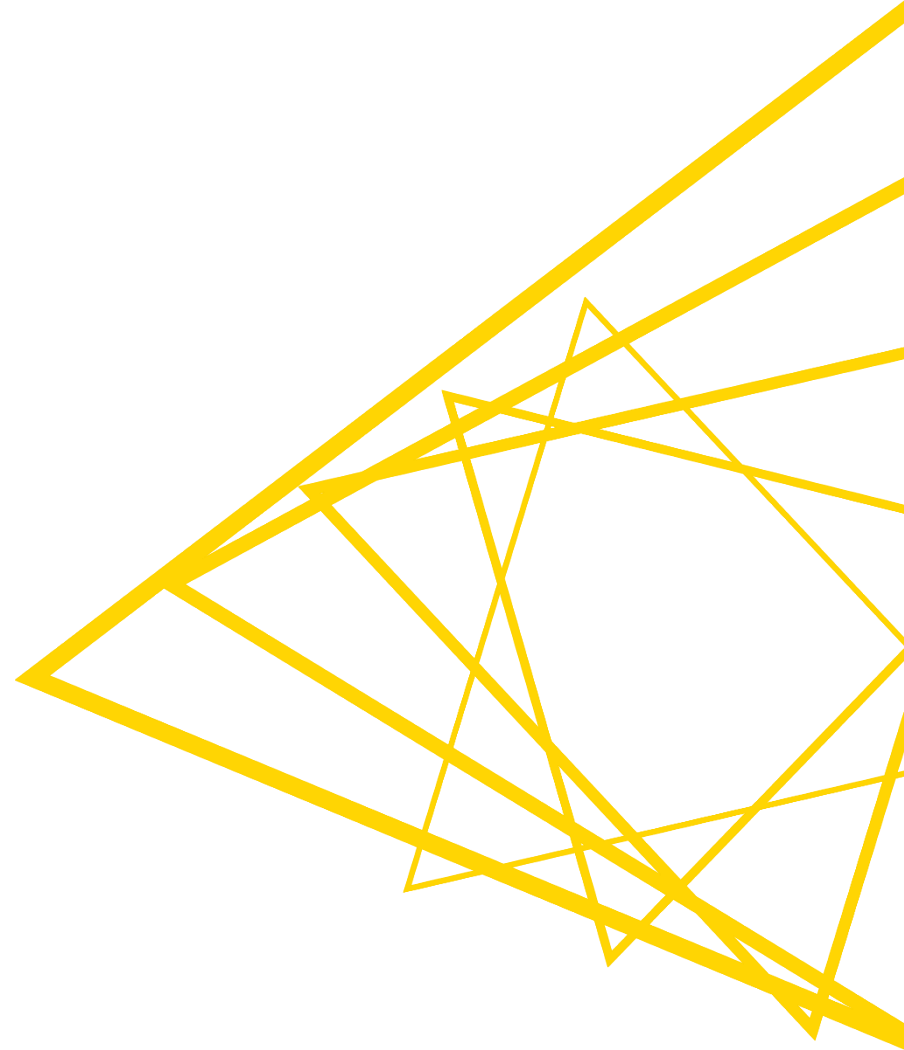
The dataset we use in this exercise describes the sale of individual residential properties in Ames, Iowa from 2006 to 2010. One of the columns is the overall condition ranking, with values between 1 and 10. The goal of this exercise is to train a binary classification model, which can predict whether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

Exercise: Decision Tree

- 1) Use 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?



Regression



Regression Analysis

- Goal: Explain how target attribute depends on descriptive attributes
 - Target attribute → **Response variable, Target**
 - Descriptive attribute(s) → **Regressor variable(s), Features**
- Commonality with models of Classification
 - First construct the model
 - Second, use the model to predict
- Difference from Classification
 - Classification model aims to predict categorical class labels
 - Regression model aims at predicting continuous values

Regression

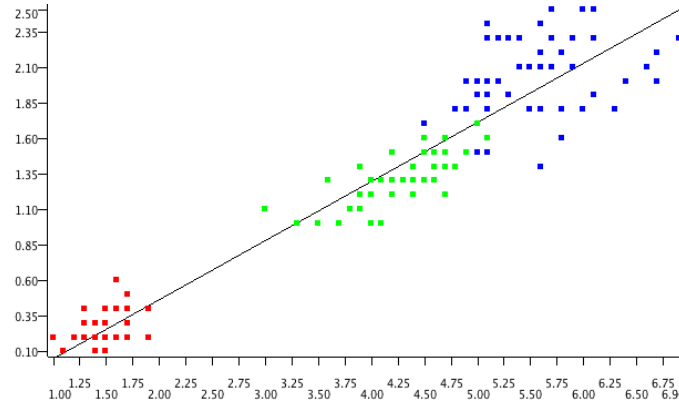
Predict *numeric* outcomes on existing data (supervised)

Applications

- Forecasting
- Quantitative Analysis

Methods

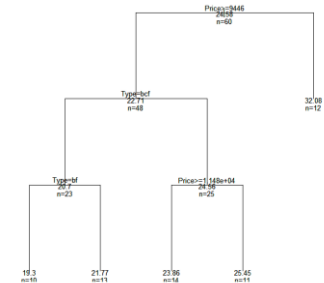
- Linear
- Polynomial
- Regression Trees
- Partial Least Squares



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

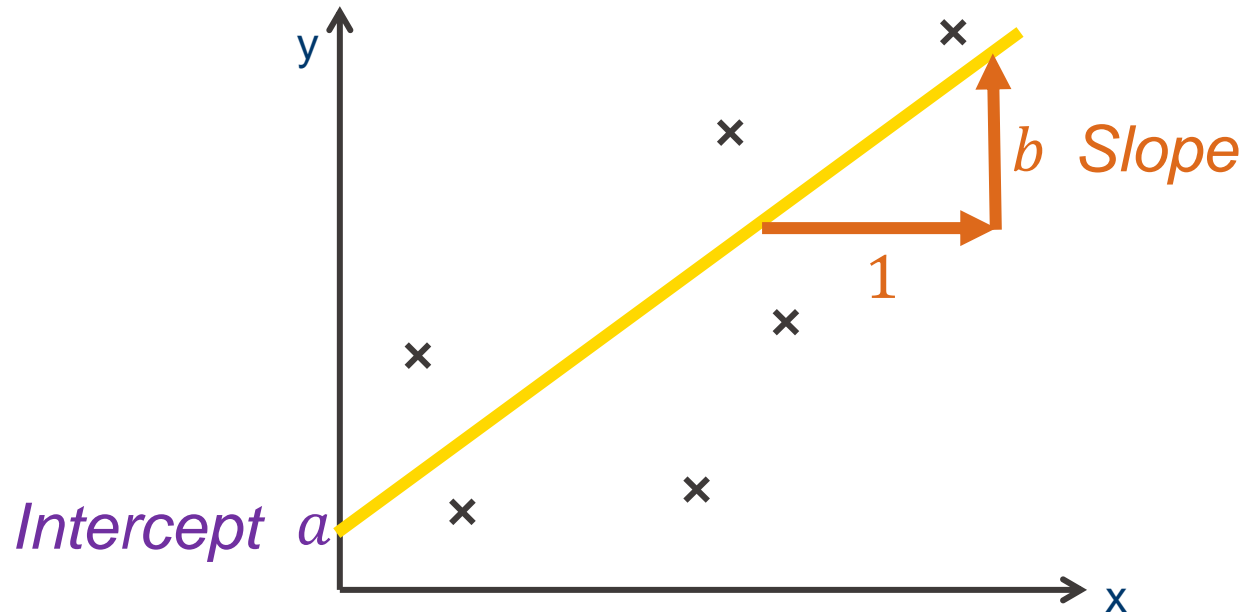


Regression Line

- Given a data set with two continuous attributes, x and y
- There is an approximate linear dependency between x and y

$$y \approx \boxed{a} + \boxed{b}x$$

Intercept *Slope*



Regression Line

- Given a data set with two continuous attributes, x and y
- There is an approximate linear dependency between x and y

$$y \approx \overset{\text{Intercept}}{\boxed{a}} + \overset{\text{Slope}}{\boxed{b}}x$$

- We find a **regression line** (i.e., determine the parameters a and b) such that the line fits the data as well as possible

- Examples:
 - Trend estimation (e.g., oil price over time)
 - Epidemiology (e.g., cigarette smoking vs. lifespan)
 - Finance (e.g., return on investment vs. return on all risky assets)
 - Economics (e.g., spending vs. available income)

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

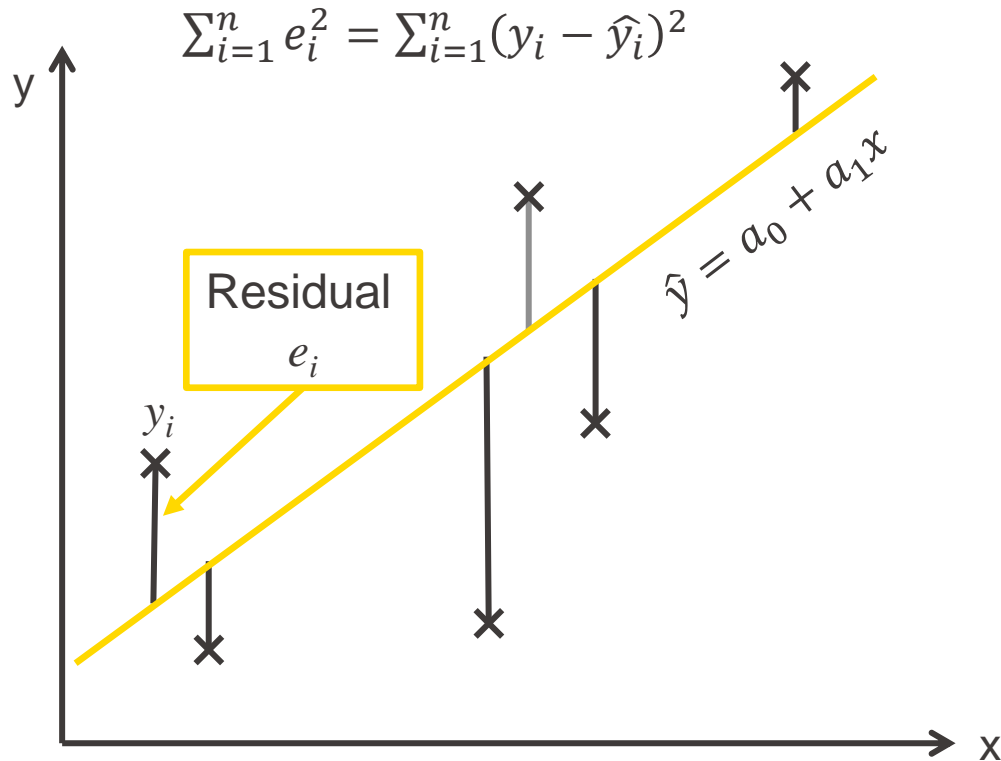
$$\text{Two input features: } \hat{y} = a_0 + a_1x_1 + a_2x_2$$

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

- 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

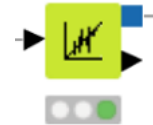
- Optimization goal: minimize the squared residuals

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

- Solution:

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

Linear Regression
Learner



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

Linear Regression: Summary

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

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

Polynomial Regression

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

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

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

Evaluation of Regression Models



Numeric Errors: Formulas

Error Metric	Formula	Notes
R-squared	$1 - \frac{\sum_{i=1}^n (y_i - f(x_i))^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	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

Easy to interpret – mean absolute error

All errors are equally weighted

Generally smaller than RMSE

RMSE

Cannot be directly interpreted as the average error

Larger errors are weighted more

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

Relative measure:

Proportion of variability explained by the model

Range: Usually between 0 and 1.

0 = no variability explained

1 = all variability explained

RMSE

Absolute measure:

How much deviation at each point

Same scale as the target

Example:

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

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

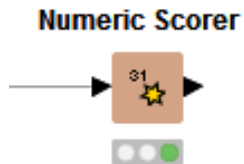
Case 2: Predicted Values = [3, 3, 7, 7]



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

Numeric Scorer

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



Statistics - 0:393 - Numeric Scorer






File Hilite Navigation View

Table "Scores" - Rows: 6 Spec - Column: 1 Properties Flow Variables

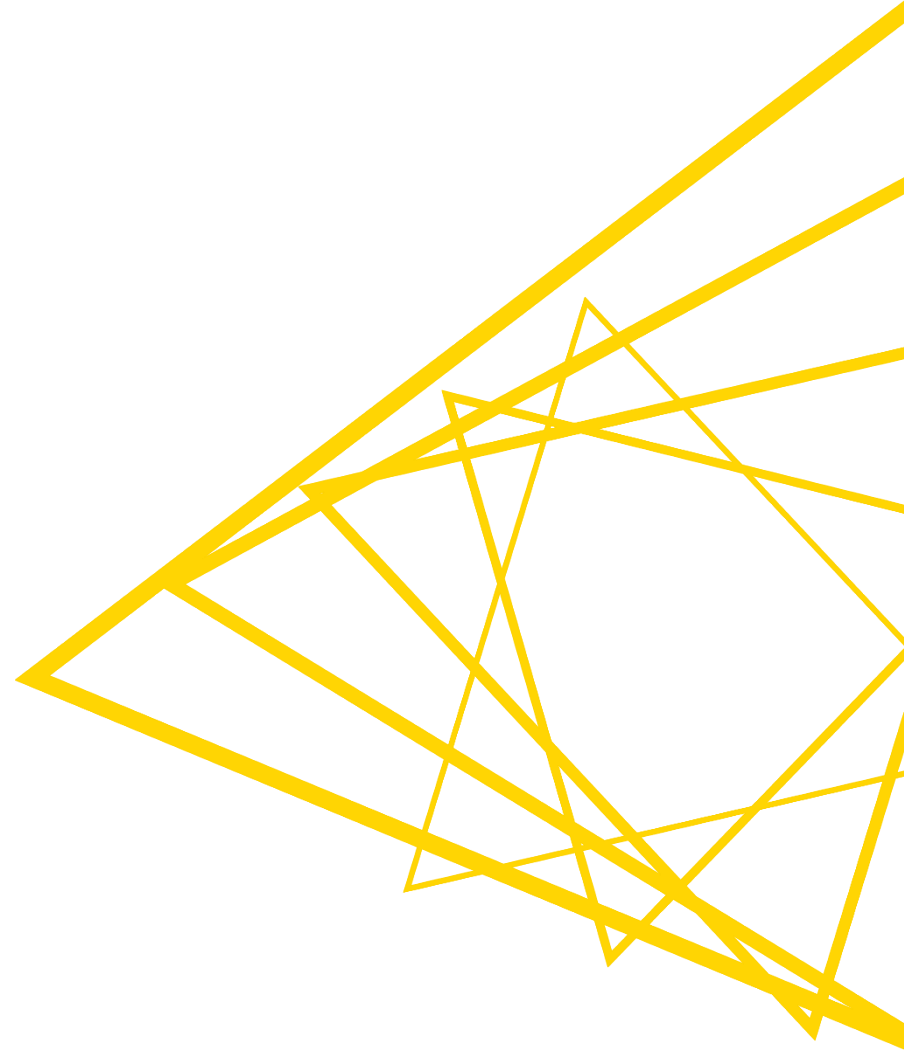
Row ID	D MA(Irregular Component)
R^2	0.343
mean absolute error	0.773
mean squared error	2.413
root mean squared error	1.553
mean signed difference	-0.003
mean absolute percentage error	7.064

Exercises

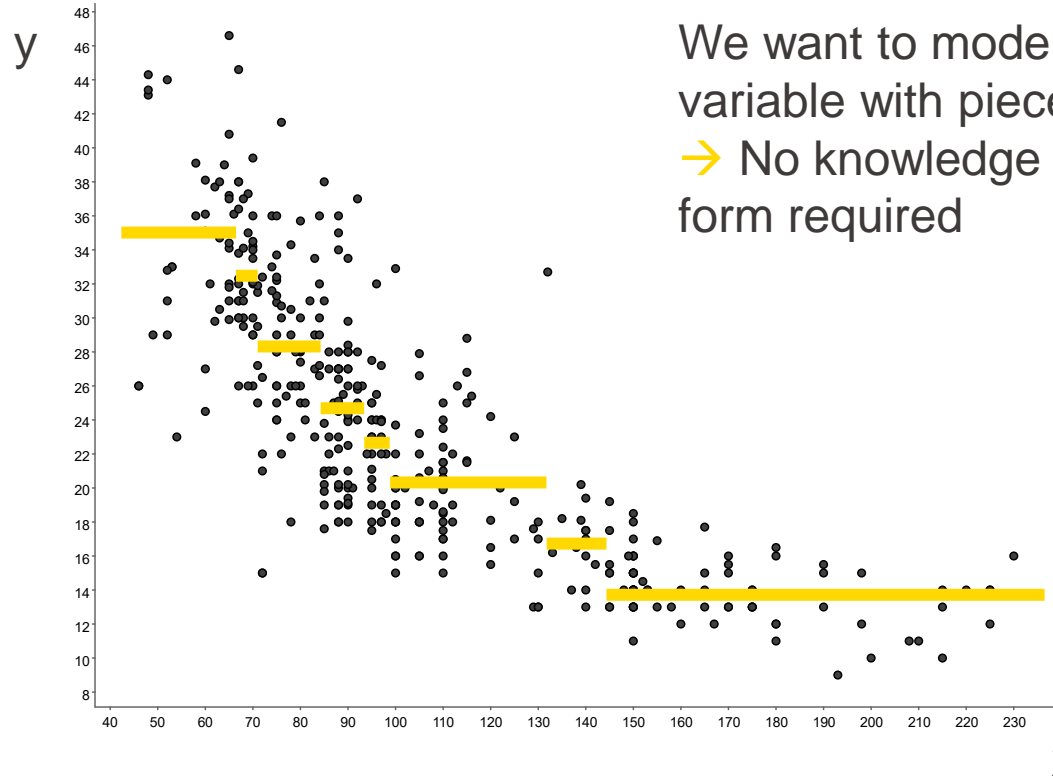
- Regression Exercises:
 - Goal: Predicting the house price
 - 01_Linear_Regression_exercise

- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - ▼  Session_2
 - ▼  01_Exercises
 - ▲ 01_Linear_Regression_exercise
 - ▲ 02_Regression_Tree_exercise
 - ▲ 03_Random_Forest_exercise
 - ▲ 04_Logistic_Regression_exercise
 - ▼  02_Solutions
 - ▲ 01_Linear_Regression_solution
 - ▲ 02_Regression_Tree_solution
 - ▲ 03_Random_Forest_solution
 - ▲ 04_Logistic_Regression_solution

Regression Tree



Regression Tree: Goal



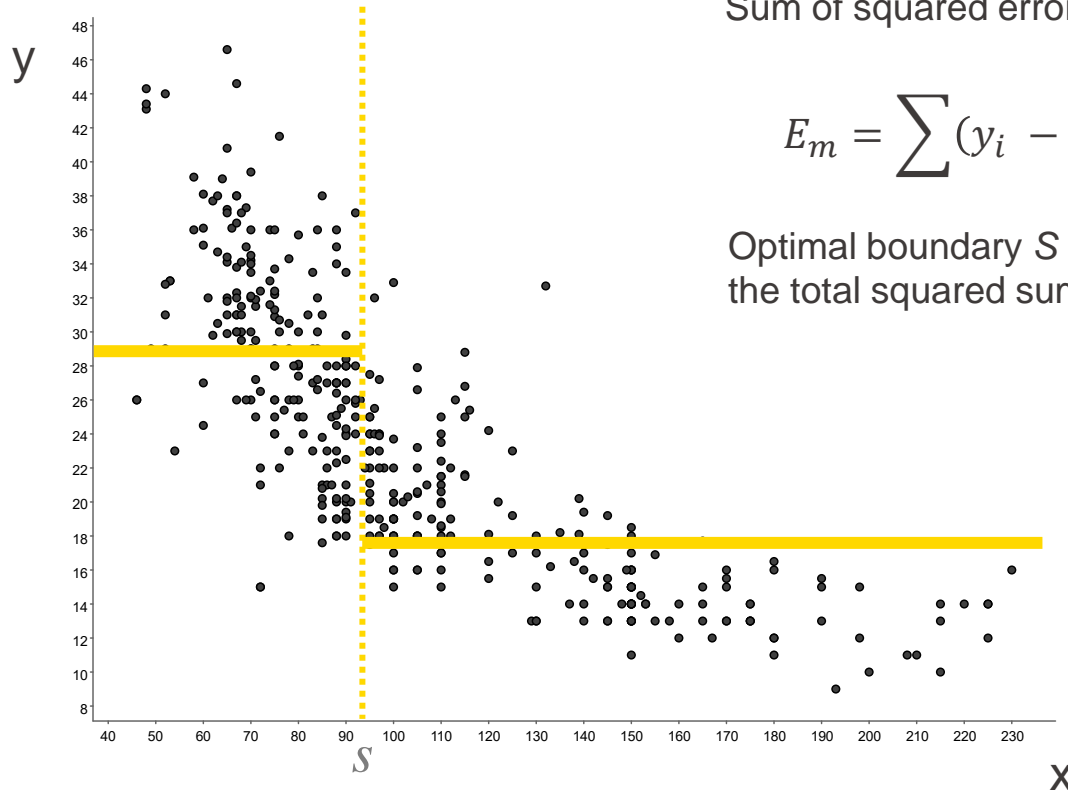
We want to model the target variable with piecewise lines
→ No knowledge of functional form required

Regression Tree: Initial Split

Local mean:

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

For observations in segment m



Sum of squared errors:

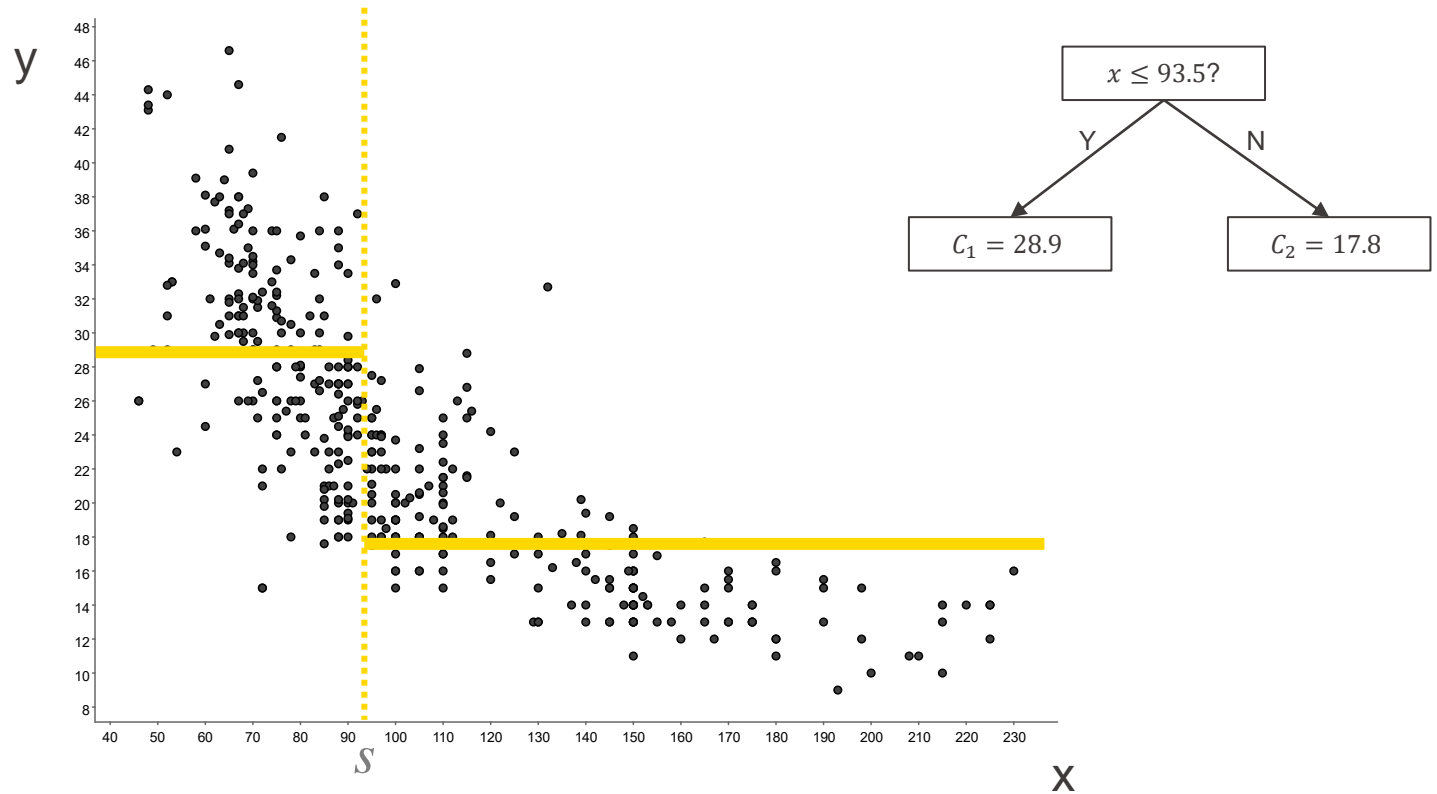
$$E_m = \sum (y_i - c_m)^2$$

Optimal boundary S should minimize the total squared sum:

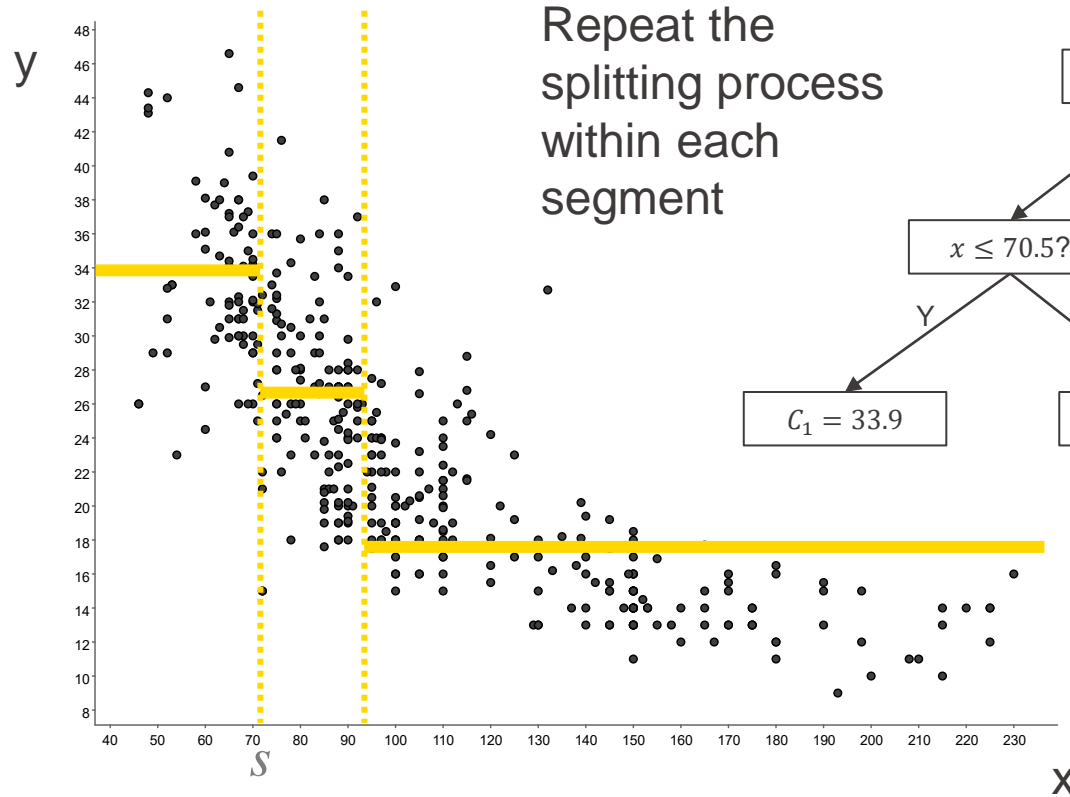
$$\sum E_m$$

For all segments m

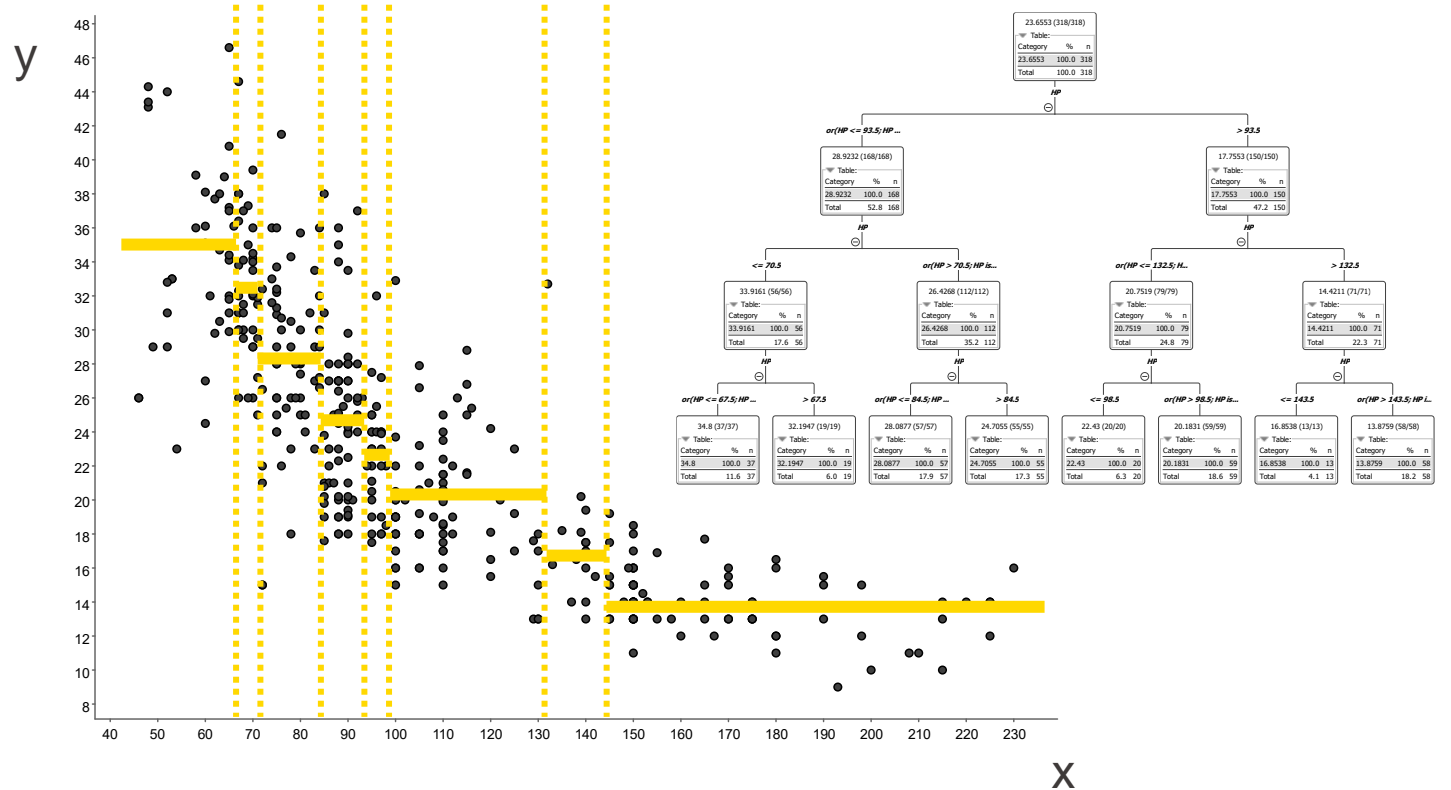
Regression Tree: Initial Split



Regression Tree: Growing the Tree



Regression Tree: Final Model

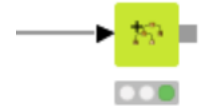


Regression Tree: Algorithm

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 \rightarrow stop
 - A node contains less than the minimum node size \rightarrow 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

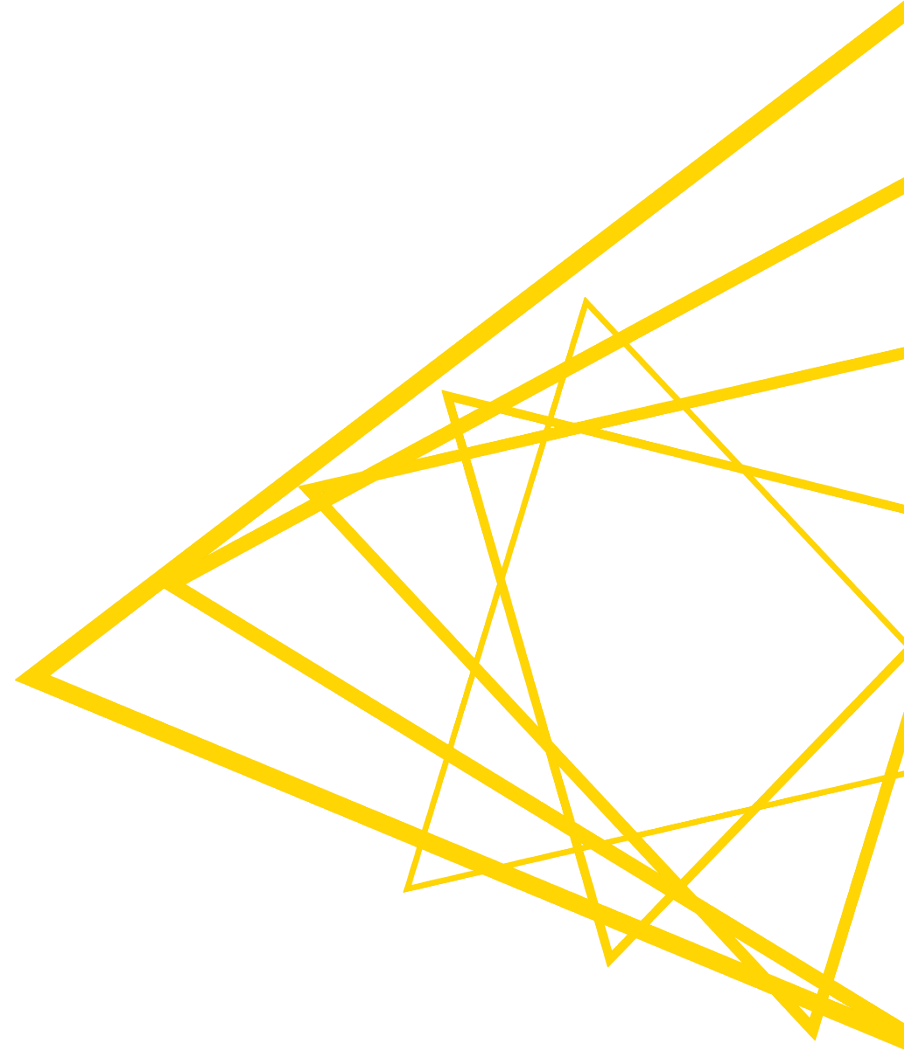
- Finding of (local) regression values (average)
- Problems:
 - No interpolation across borders
 - Heuristic algorithm: unstable and not optimal.
- Extensions:
 - Fuzzy trees (better interpolation)
 - Local models for each leaf (linear, quadratic)

Exercises

- Regression Exercises:
 - Goal: Predicting the house price
 - 02_Regression_Tree_exercise

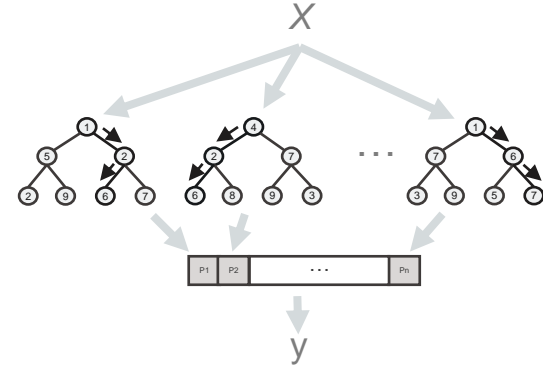
- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - ▼  Session_2
 - ▼  01_Exercises
 - ▲  01_Linear_Regression_exercise
 - ▲  02_Regression_Tree_exercise
 - ▲  03_Random_Forest_exercise
 - ▲  04_Logistic_Regression_exercise
 - ▼  02_Solutions
 - ▲  01_Linear_Regression_solution
 - ▲  02_Regression_Tree_solution
 - ▲  03_Random_Forest_solution
 - ▲  04_Logistic_Regression_solution

Ensemble Models



Tree Ensemble Models

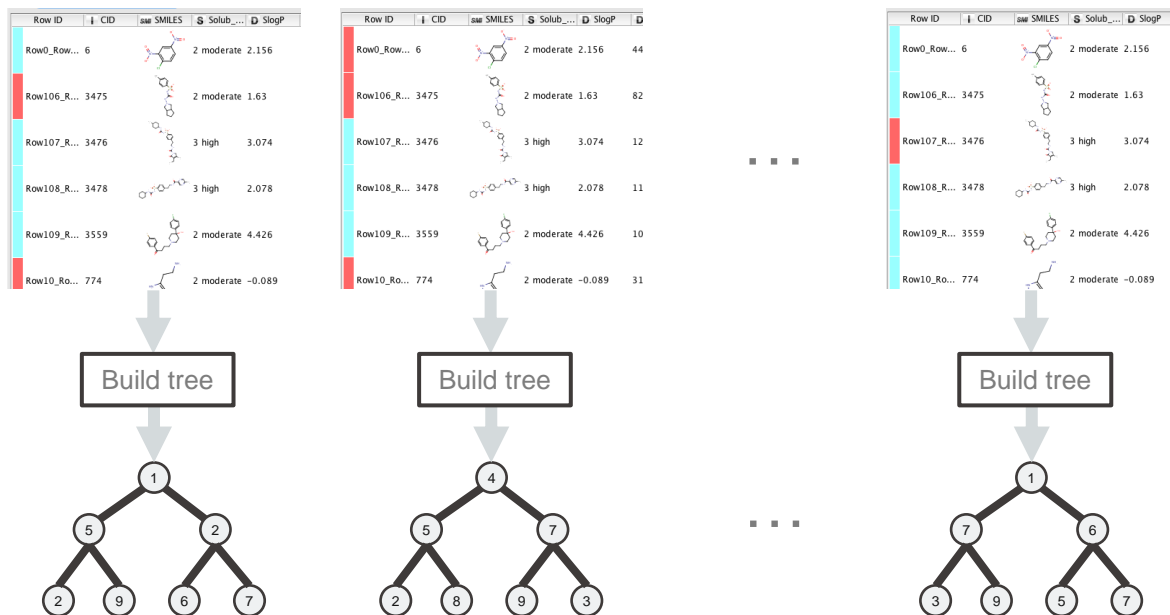
- General idea: take advantage of the “wisdom of the crowd”
- Ensemble models: Combining predictions from many predictors, e.g. decision trees
- Leads to a more accurate and robust model
- Model is difficult to interpret
 - There are multiple trees in the model



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

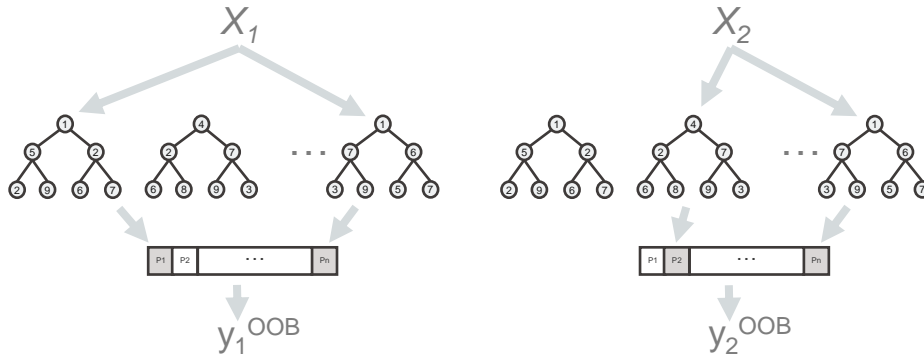
RowID	x_1	x_2	y
Row_1	2	6	Class 1
Row_2	4	1	Class 2
Row_3	9	3	Class 2
Row_4	2	7	Class 1
Row_5	8	1	Class 2
Row_6	2	6	Class 1
Row_7	5	2	Class 2

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









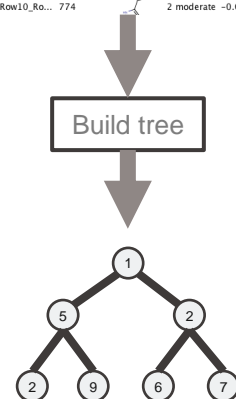
Out-of-bag error estimates - 0:105 - Random Forest Learner

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	JH	1	1	0	0	1			33			
Row3_Row2	IJ	1	1	0	0	1			37			
Row4_Row3	JH	0.528	0.472	0	0.528				36			
Row5_Row4	JK	0.976	0.024	0	0.976				41			
Row6_Row5	L	0.848	0.152	0	0.848				33			
Row7_Row6	IA	0.833	0.167	0	0.833				36			
Row9_Row8	A	0.667	0.333	0	0.667				30			
Row11_Row...	N	0.138	0.862	1	0.862				29			
Row13_Ro...	A	0.974	0.026	0	0.974				39			
Row14_Ro...	IT	0.917	0.083	0	0.917				36			
Row15_Ro...	A	0.387	0.613	1	0.613				31			
Row18_Ro...	T	0.974	0.026	0	0.974				39			
Row19_Ro...	A	1	0	0	1				38			
Row21_Ro...	L	0.971	0.029	0	0.971				34			
Row22_Ro...	O	0.03	0.97	1	0.97				33			
Row23_Ro...	Z	0.854	0.146	0	0.854				41			
Row25_Ro...	A	0.973	0.027	0	0.973				37			
Row26_Ro...	IE	0.886	0.114	0	0.886				35			
Row27_Ro...	Y	0.912	0.088	0	0.912				34			
Row28_Ro...	IT	0.976	0.024	0	0.976				42			
Row29_Ro...	IO	1	0	0	1				42			
Row30_Ro...	II	1	0	0	1				40			
Row32_Ro...	IH	0.914	0.086	0	0.914				35			
Row33_Ro...	A	0.875	0.125	0	0.875				32			

Random Forest

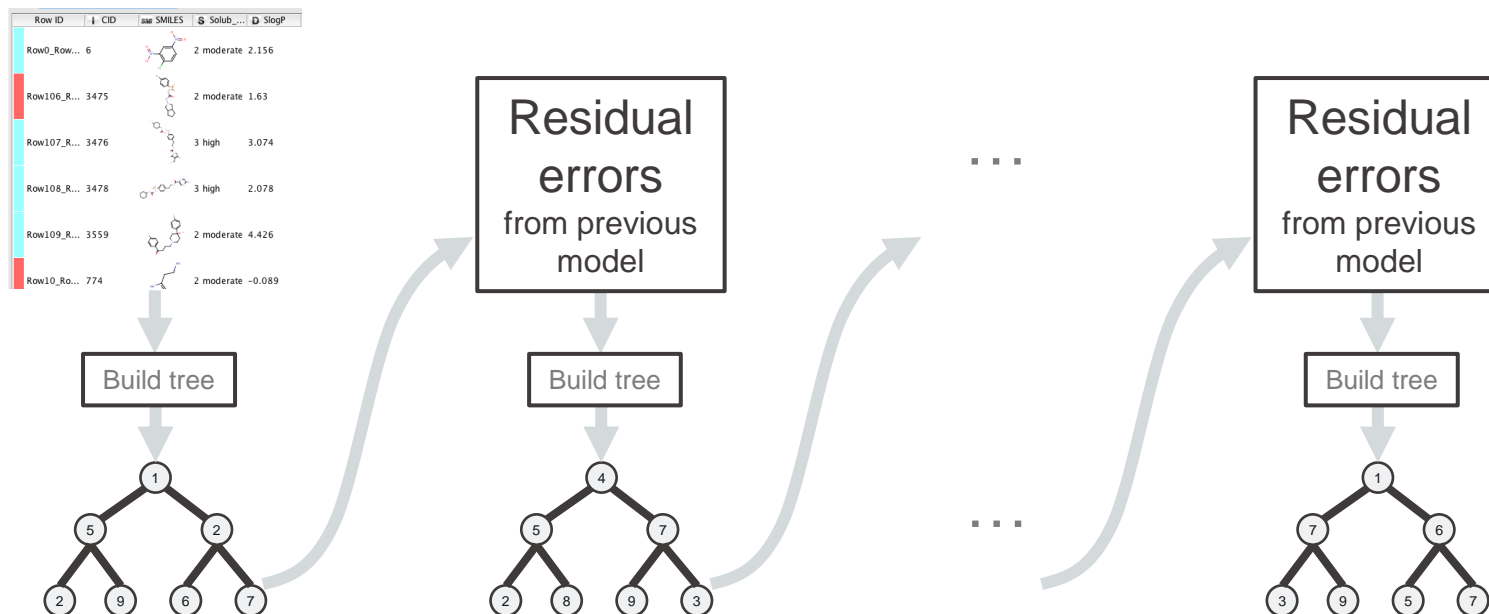
- Bag of decision trees, with an extra element of randomization
- **Each node** in the decision tree only “sees” **a subset of the input features**, typically \sqrt{N} to pick from
- Random forests tend to be very robust w.r.t. overfitting

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



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



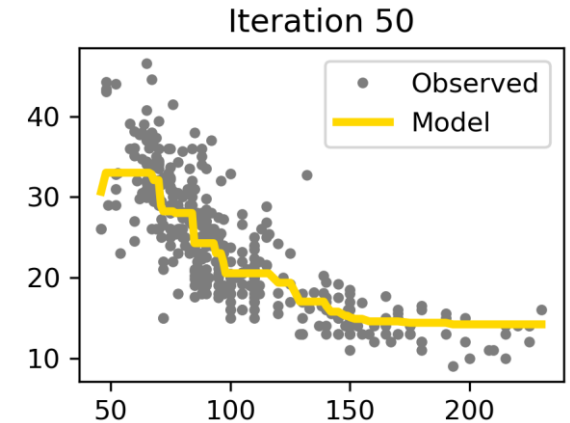
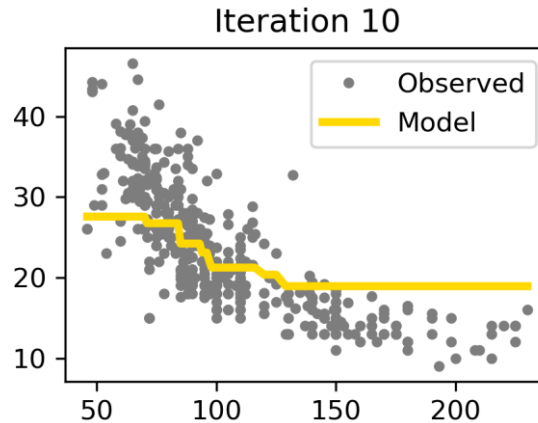
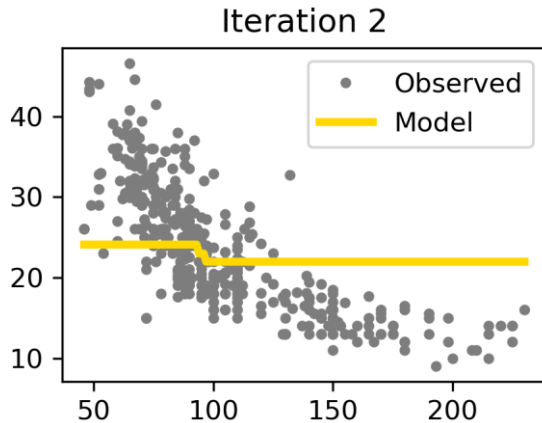
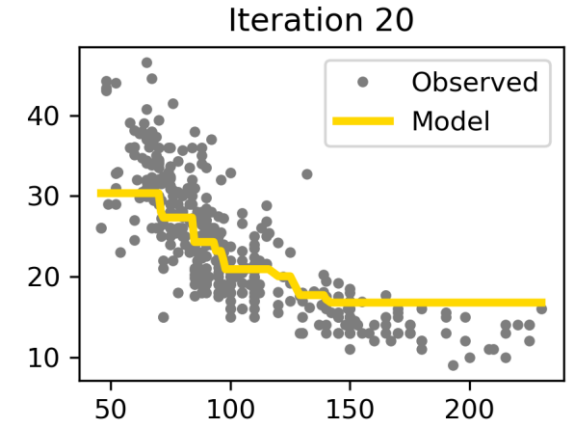
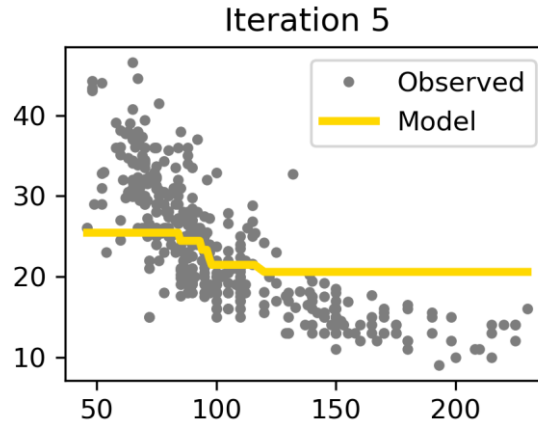
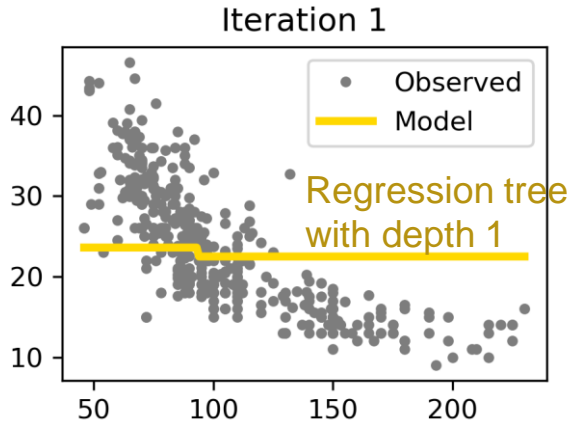
Boosting - Idea

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

Gradient Boosting Example – Regression

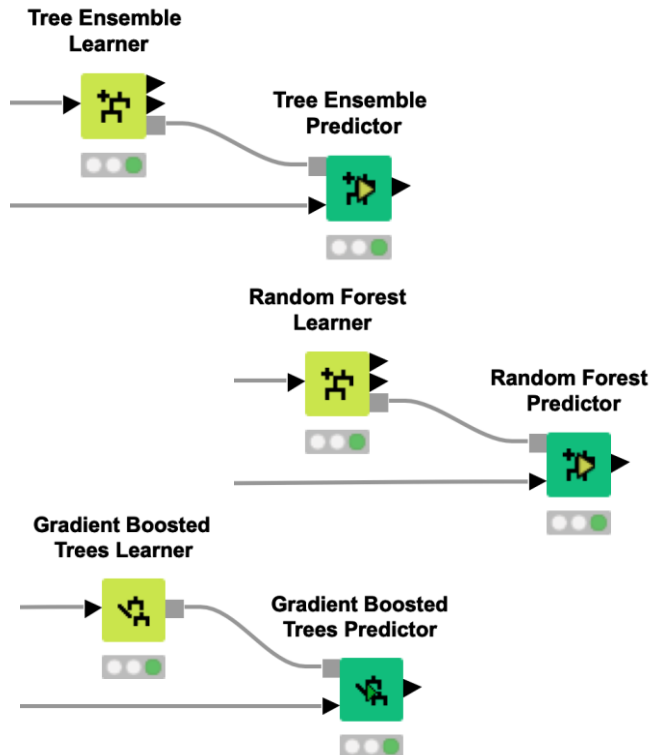


Gradient Boosted Trees

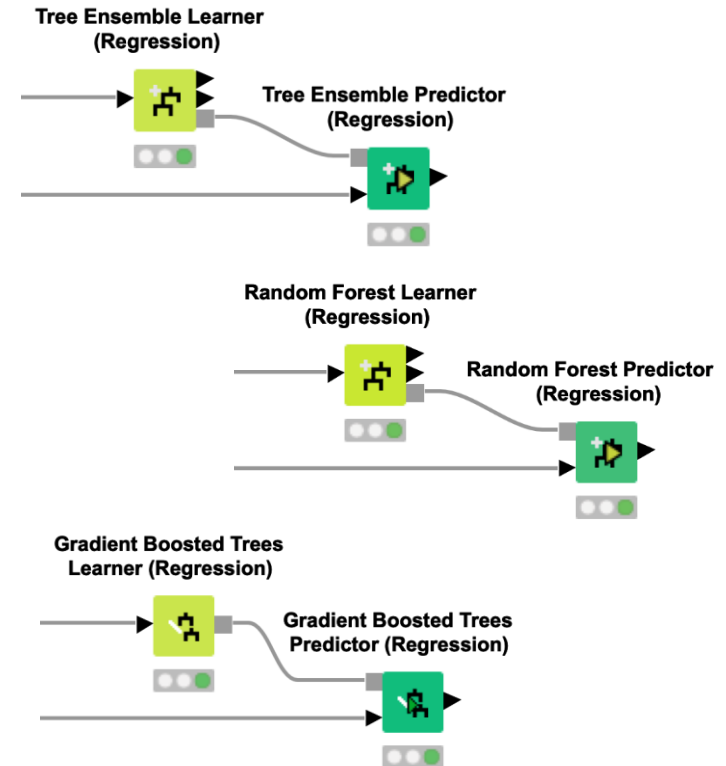
- Can be used for classification and regression
- Large number of iterations – prone to overfitting
 - ~100 iterations are sufficient
- Can introduce randomness in choice of data subsets (“stochastic gradient boosting”) and choice of input features

Ensemble Tree Nodes in KNIME Analytics Platform

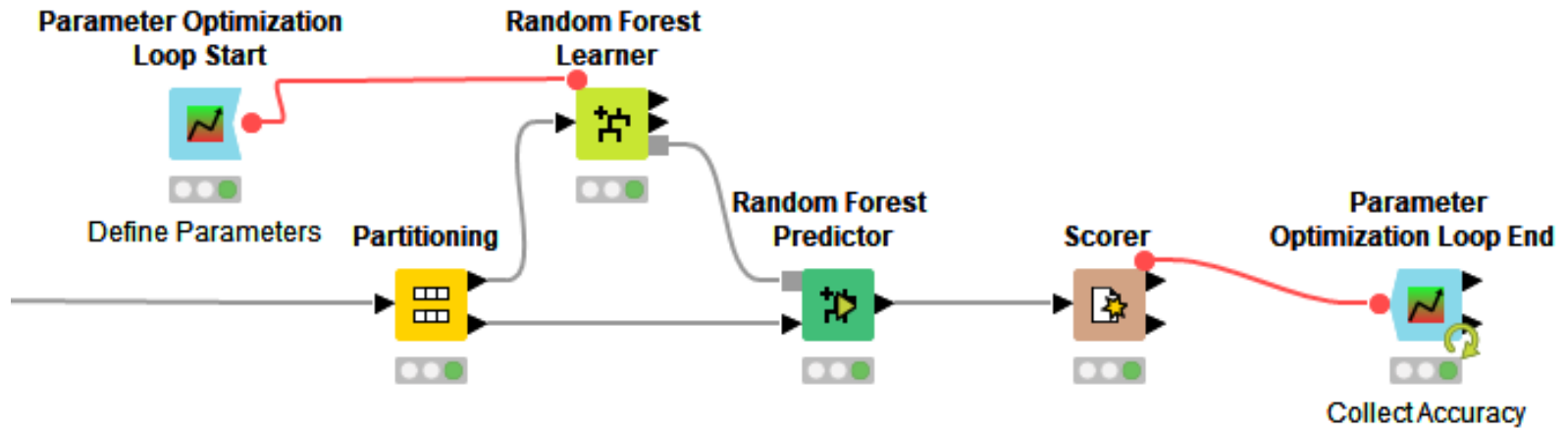
Classification Problems



Regression Problems
















Parameter Optimization



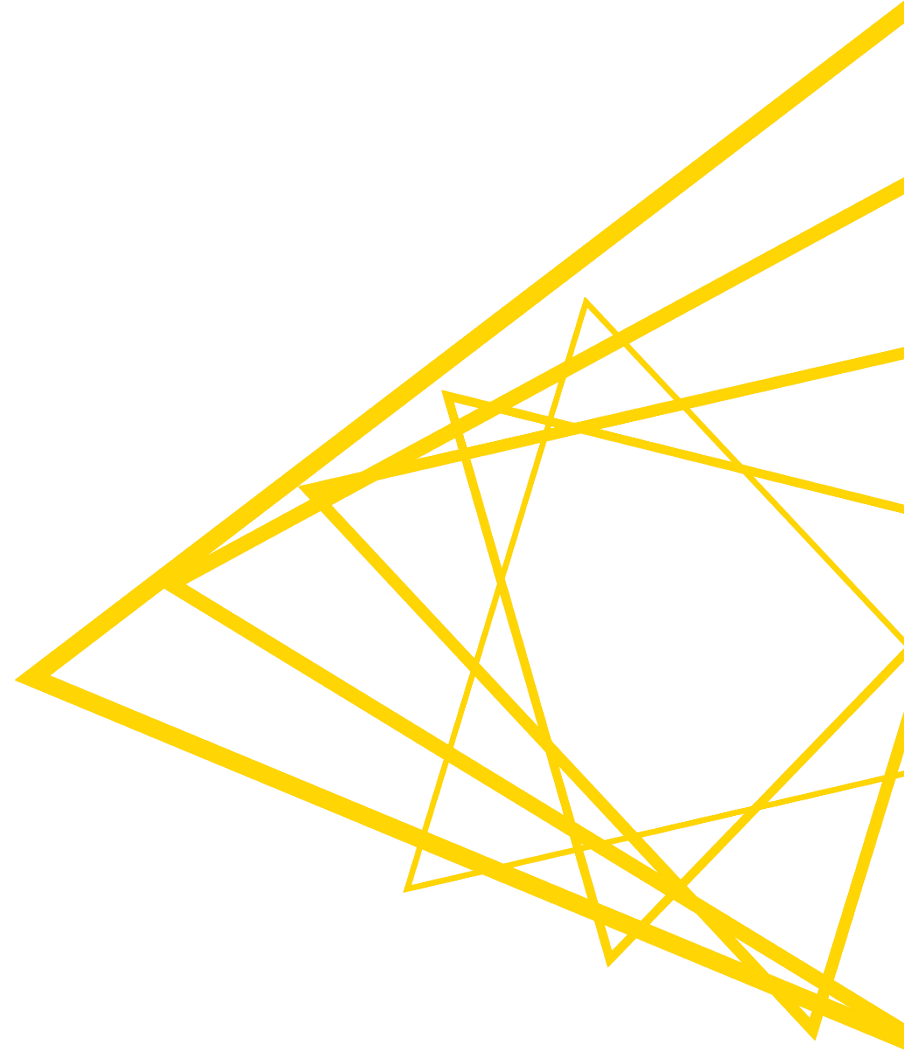
Exercises

- Classification Exercises:

- Goal: Predicting the house condition (high /low)
- 03_Random_Forest_exercise (with optional exercise to build a parameter optimization loop)

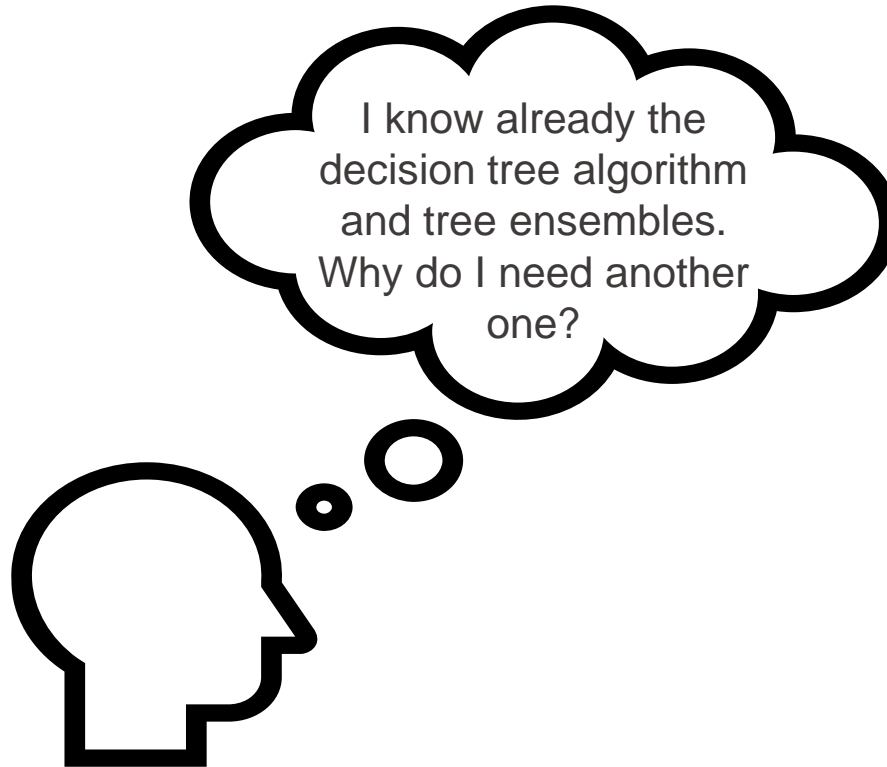
- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - ▼  Session_2
 - ▼  01_Exercises
 - ▲  01_Linear_Regression_exercise
 - ▲  02_Regression_Tree_exercise
 - ▲  03_Random_Forest_exercise
 - ▲  04_Logistic_Regression_exercise
 - ▼  02_Solutions
 - ▲  01_Linear_Regression_solution
 - ▲  02_Regression_Tree_solution
 - ▲  03_Random_Forest_solution
 - ▲  04_Logistic_Regression_solution

Logistic Regression

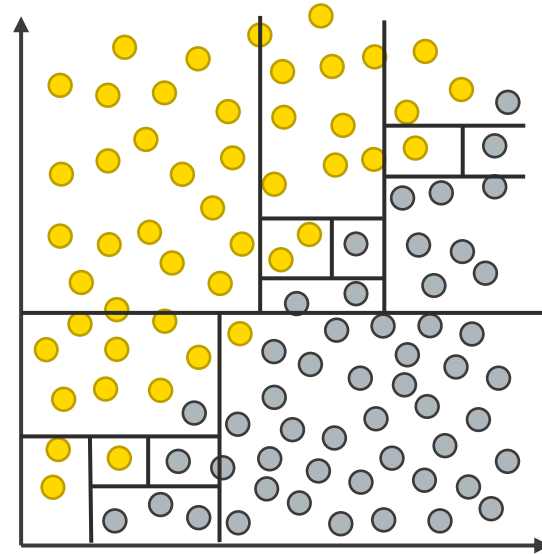
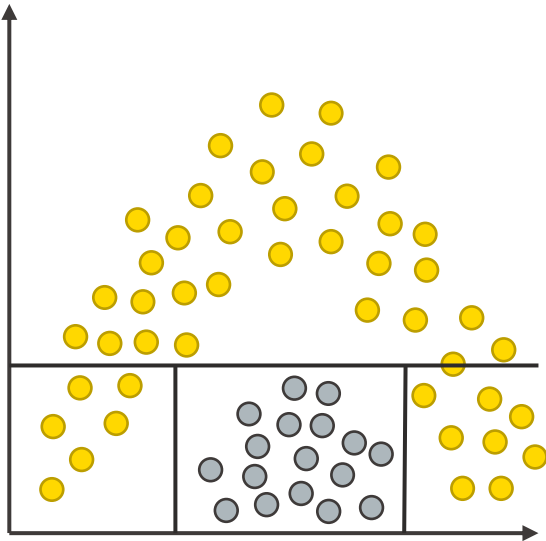


What is a Logistic Regression (algorithm)?

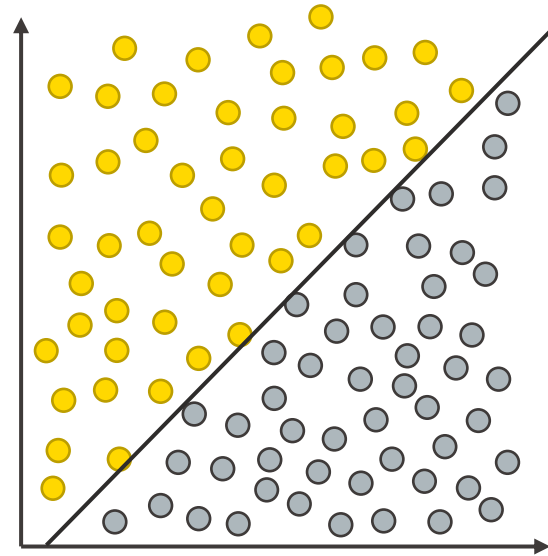
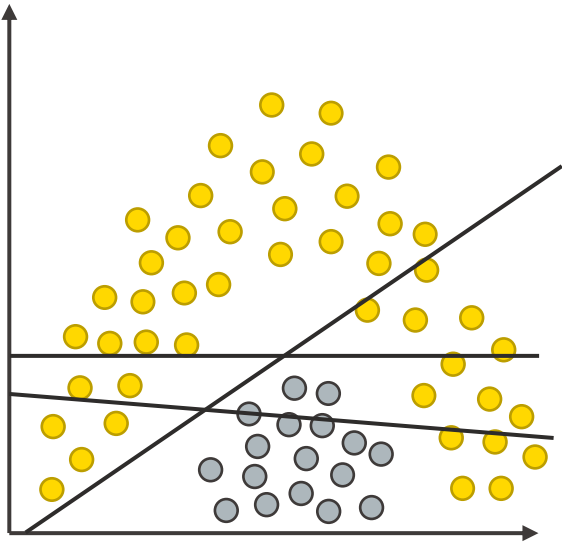
- Another algorithm to train a classification model



Why Shouldn't we Always use the Decision Tree?



Decision Boundary of a Logistic Regression



Linear Regression vs. Logistic Regression

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

Goal: Find the regression coefficients β_0, \dots, β_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

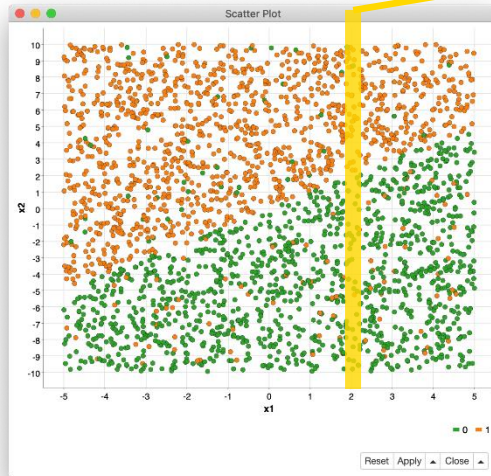
$$P(y = 1) = p_1 \Rightarrow P(y = 0) = 1 - p_1$$

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

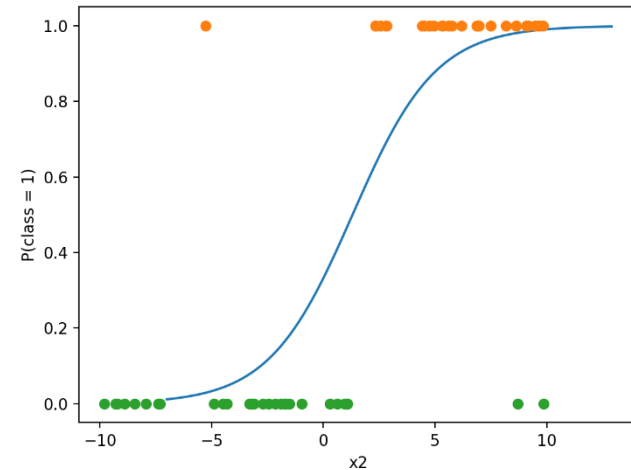
Let's Find Out How Binary Logistic Regression Works!

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

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 β ?

Maximize the product of the probabilities → Likelihood function

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

Why does it make sense to maximize this function?

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

Remember:

$$\pi_i = P(y = 1)$$

$$u^0 = 1 \text{ for } u \in \mathbb{R}$$

$$u^1 = u \text{ for } u \in \mathbb{R}$$

Max Likelihood and Log Likelihood Functions

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

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

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

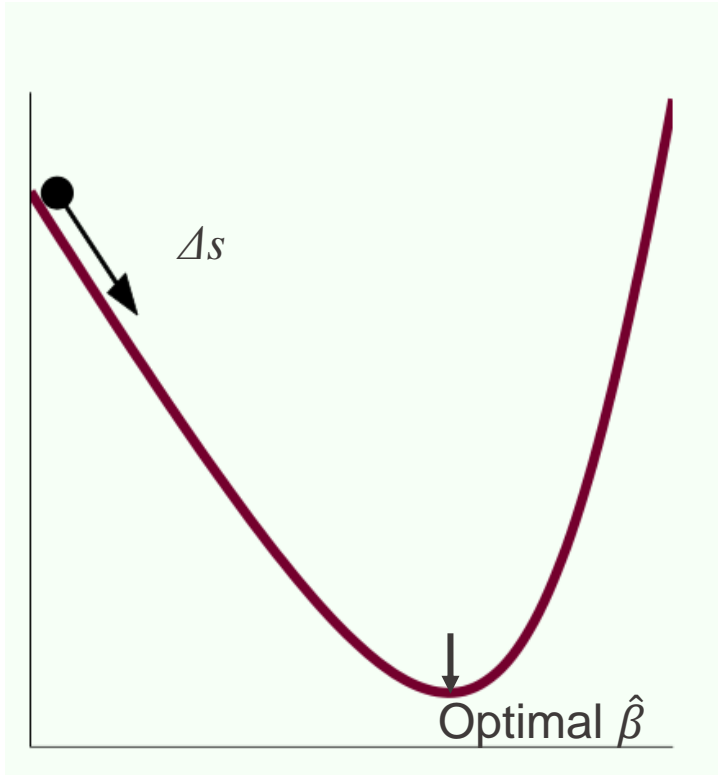
$$\max_{\boldsymbol{\beta}} LL(\boldsymbol{\beta}; \mathbf{y}, \mathbf{X}) = \max_{\boldsymbol{\beta}} \sum_{i=1}^n y_i \ln(\pi_i) + (1 - y_i) \ln(1 - \pi_i)$$

How can we find this Coefficients?

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

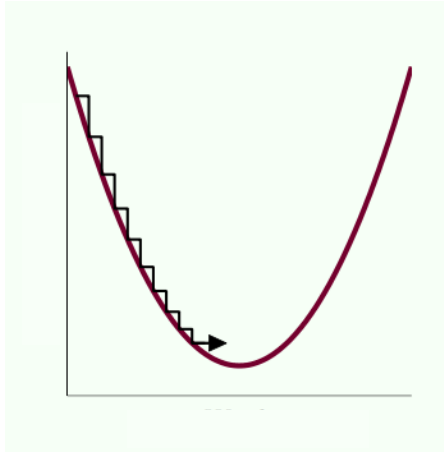
$$\max LL(\beta; X, y) \Leftrightarrow \min -LL(\beta; X, y)$$



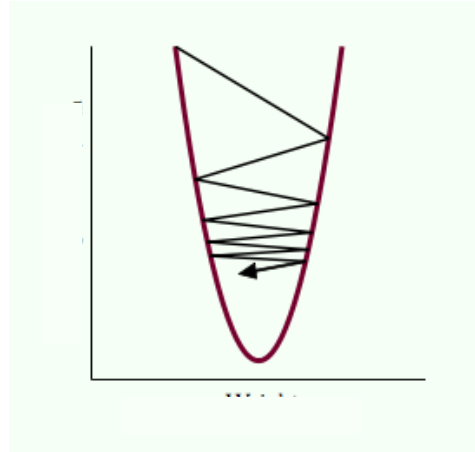
- Example: $\min -LL(\beta) := f(\beta)$
- Start from an arbitrary point
- Move towards the minimum
- With step size Δs
- If $f(\beta)$ is strictly convex
 - ➔ Only one global minimum exists
- Z normalization of the input data lead to better convergence

Learning Rate / Step Length Δs

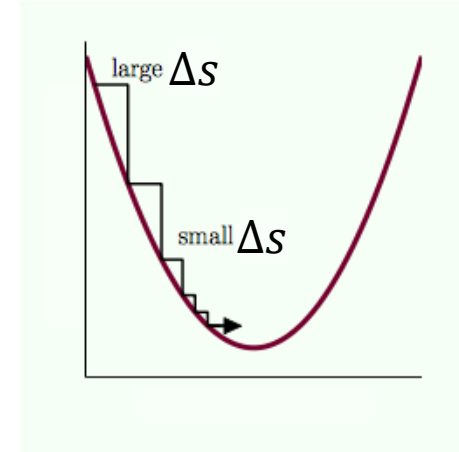
Δs too small



Δs too large



Just right



Learning Rate Δs

- Fixed:

$$\Delta s_k = \Delta s_0$$

- Annealing:

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

with iteration number k and decay rate α

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

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

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

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

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

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

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

Impact of Regularization



Interpretation of the Coefficients

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

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

- $p\text{-value} < \alpha$: 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.
- Regularization can be used to avoid overfitting.
- The p-value shows us whether an independent variable is significant

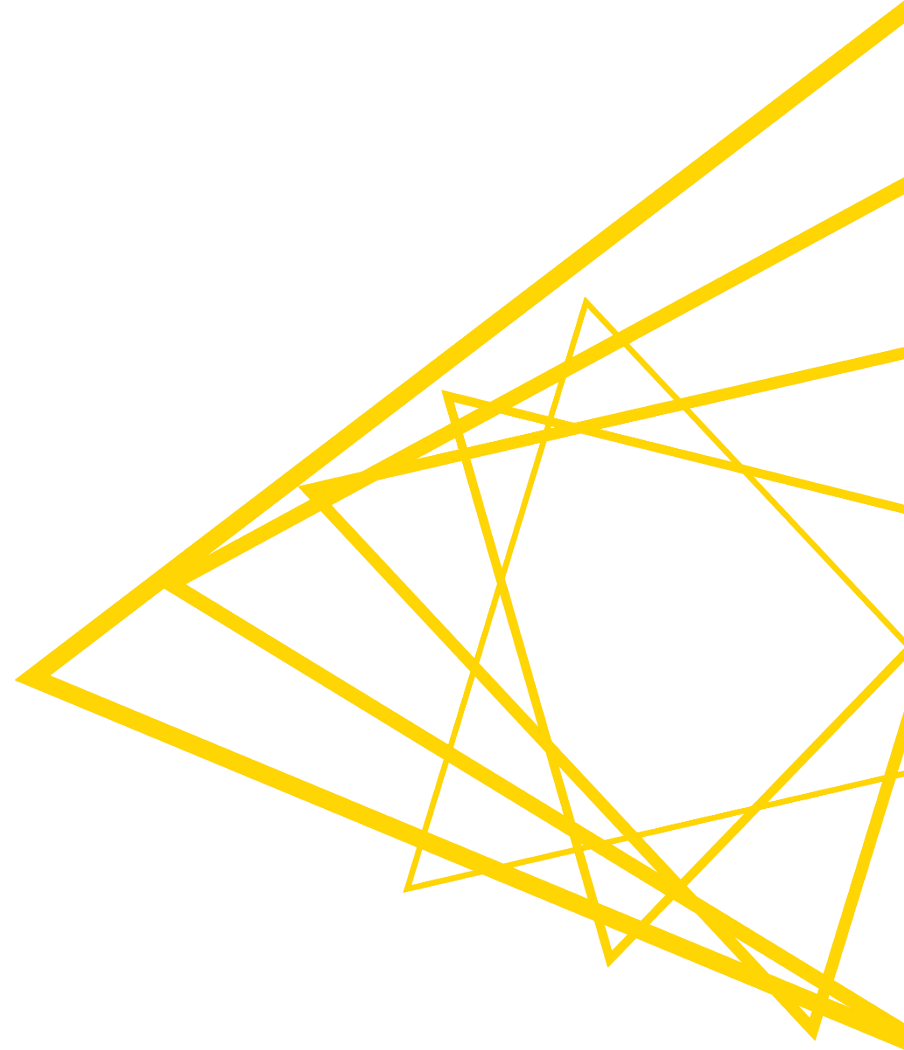
Exercises

- Classification Exercises:

- Goal: Predicting the house condition (high /low)
- 04_Logistic_Regression_exercise

- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - ▼  Session_2
 - ▼  01_Exercises
 - ▲  01_Linear_Regression_exercise
 - ▲  02_Regression_Tree_exercise
 - ▲  03_Random_Forest_exercise
 - ▲  04_Logistic_Regression_exercise
 - ▼  02_Solutions
 - ▲  01_Linear_Regression_solution
 - ▲  02_Regression_Tree_solution
 - ▲  03_Random_Forest_solution
 - ▲  04_Logistic_Regression_solution

Recommendation Engines



Recommendation Engines and Market Basket Analysis

From the analysis of many shopping baskets ...

A-priori algorithm

Recommendation



IF



+



THEN



A-priori Algorithm: the Association Rule

IF



+

THEN



Antecedent

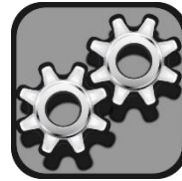
Consequent

Building the Association Rule

N shopping baskets



Search for
frequent itemsets



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

From “Frequent Itemsets” to “Rules”

{A, B, F, H}



{A, B, F} → H

{A, B, H} → F

{A, F, H} → B

{B, F, H} → A

Which rules shall I choose?

Support, Confidence, and Lift

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

- Item set support $s = \frac{freq(A,B,F,H)}{N}$



How often these items are found together

- Rule confidence $c = \frac{freq(A,B,F,H)}{freq(A,B,F)}$



How often the antecedent is together with the consequent

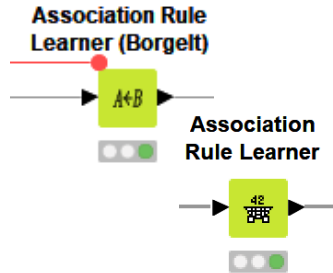
- Rule lift = $\frac{support(\{A,B,F\} \Rightarrow H)}{support(A,B,F) \times support(H)}$



How often antecedent and consequent happen together compared with random chance

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

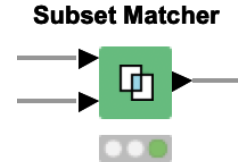
Association Rule Mining (ARM): Two Phases



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

$$X \Rightarrow Y \quad \rightarrow \quad \text{“if X then Y”}$$

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



Two phases:

1. find all frequent itemsets (FI)

← Most of the complexity

- Select itemsets with a minimum support

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

2. build strong association rules

- Select rules with a minimum confidence:

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

User parameters

A-Priori Algorithm: Example

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

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

support

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

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

confidence

A-Priori Algorithm: 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

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

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

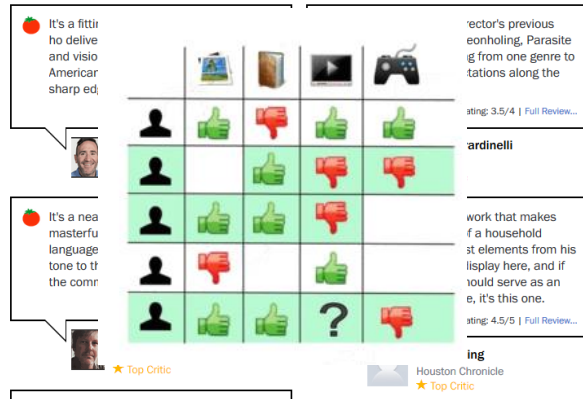
$$\begin{aligned} \text{Rule lift} &= \frac{s(milk, diaper, beer)}{s(milk, diaper) \times s(beer)} \\ &= \frac{0.4}{0.6 \times 0.6} = 1.11 \end{aligned}$$

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

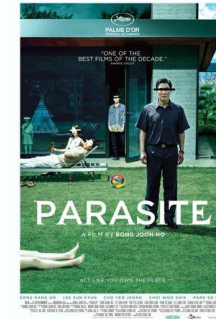
From the analysis of the reactions of many people to the same item ...



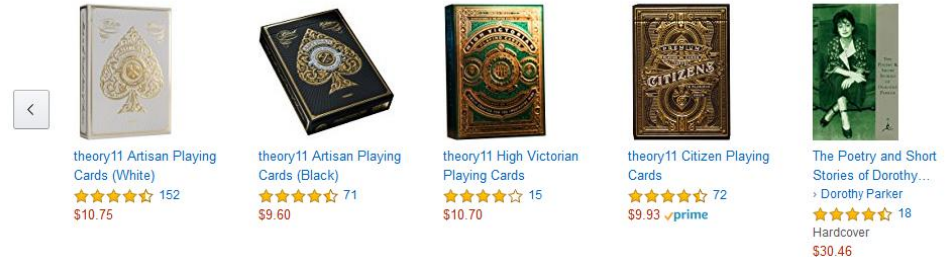
Collaborative Filtering



Recommendation



Inspired by your purchases



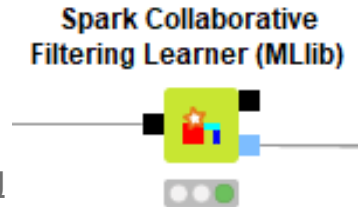
IF A has the same opinion as B on an item,
THEN A is more likely to have B's opinion on another item than that of a randomly chosen person

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>

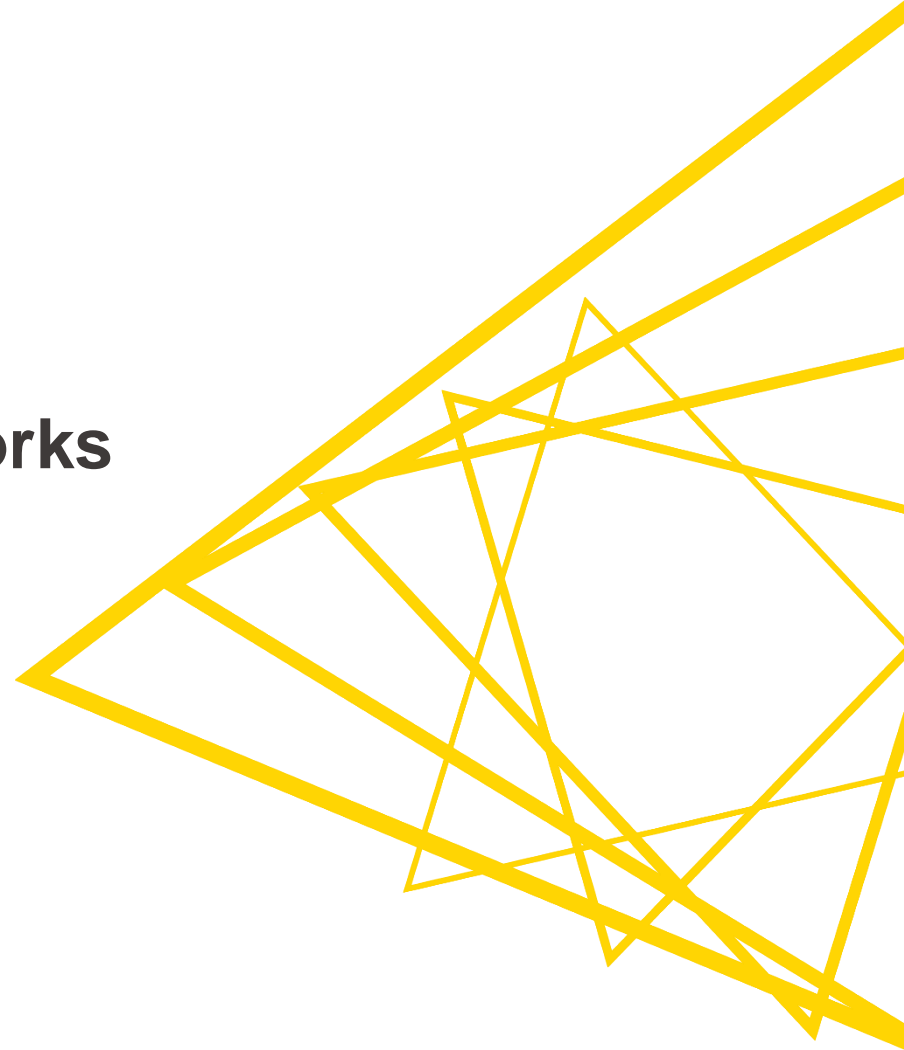


Exercises:

- Market Basket Analysis
 - 02_Build_Association_Rules_for_MarketBasketAnalysis_exercise
 - 03_Apply_Association_Rules_for_MarketBasketAnalysis_exercise

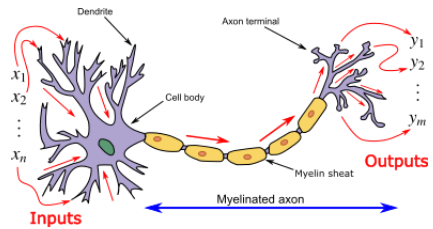
- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - >  Session_2
 - ▼  Session_3
 - ▼  01_Exercises
 - ▲ 01_Simple_Neural_Network_exercise
 - ▲ 02_Build_Association_Rules_for_MarketBasketAnalysis_exercise
 - ▲ 03_Apply_Association_Rules_for_MarketBasketAnalysis_exercise
 - ▼  02_Solutions
 - ▲ 01_Simple_Neural_Network_solution
 - ▲ 02_Build_Association_Rules_for_MarketBasketAnalysis_solution
 - ▲ 03_Apply_Association_Rules_for_MarketBasketAnalysis_solution

Artificial Neurons and Networks

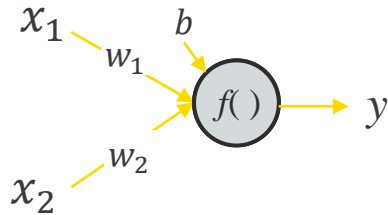


Biological vs. Artificial

Biological Neuron



Artificial Neuron (Perceptron)

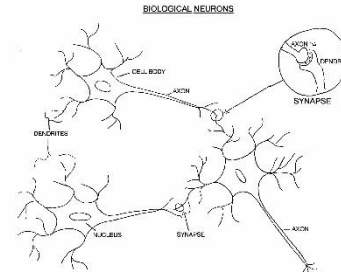


$$y = f(x_1w_1 + x_2w_2 + b)$$

$$b = w_0$$

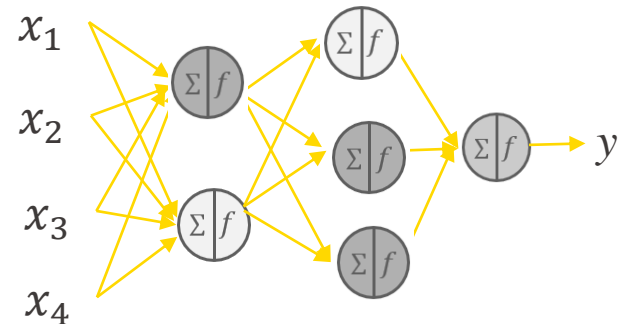
$$y = f\left(\sum_{i=0}^n x_iw_i\right)$$

Biological Neural Networks

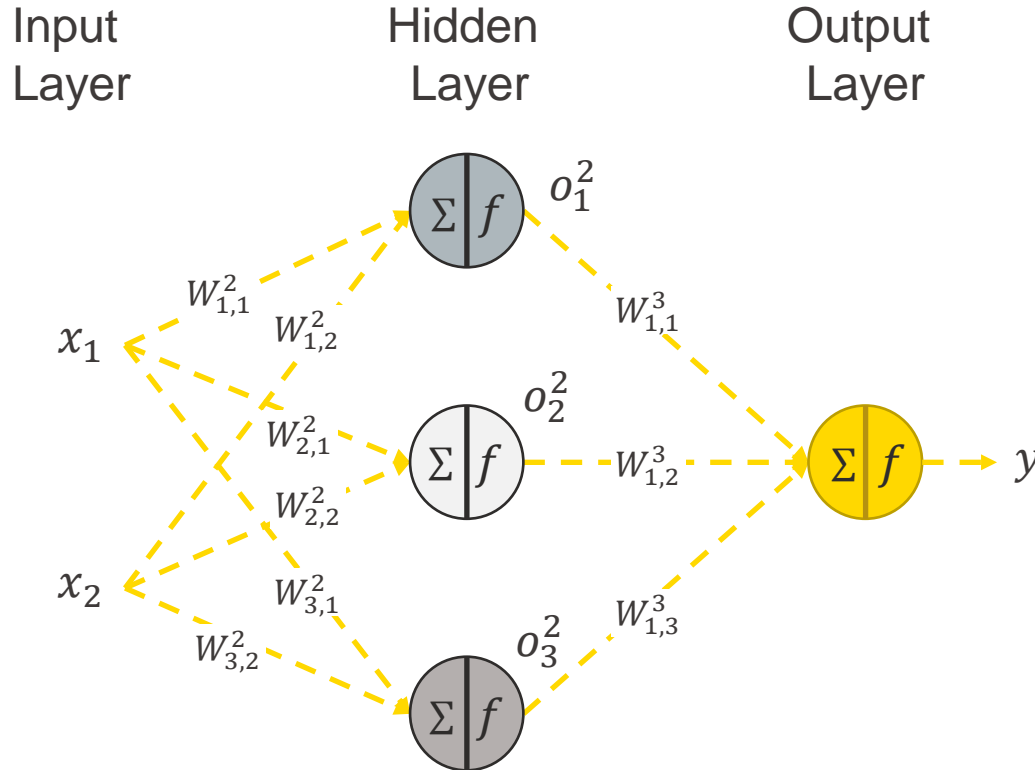


Artificial Neural Networks

(Multilayer Perceptron, MLP)



Architecture / Topology

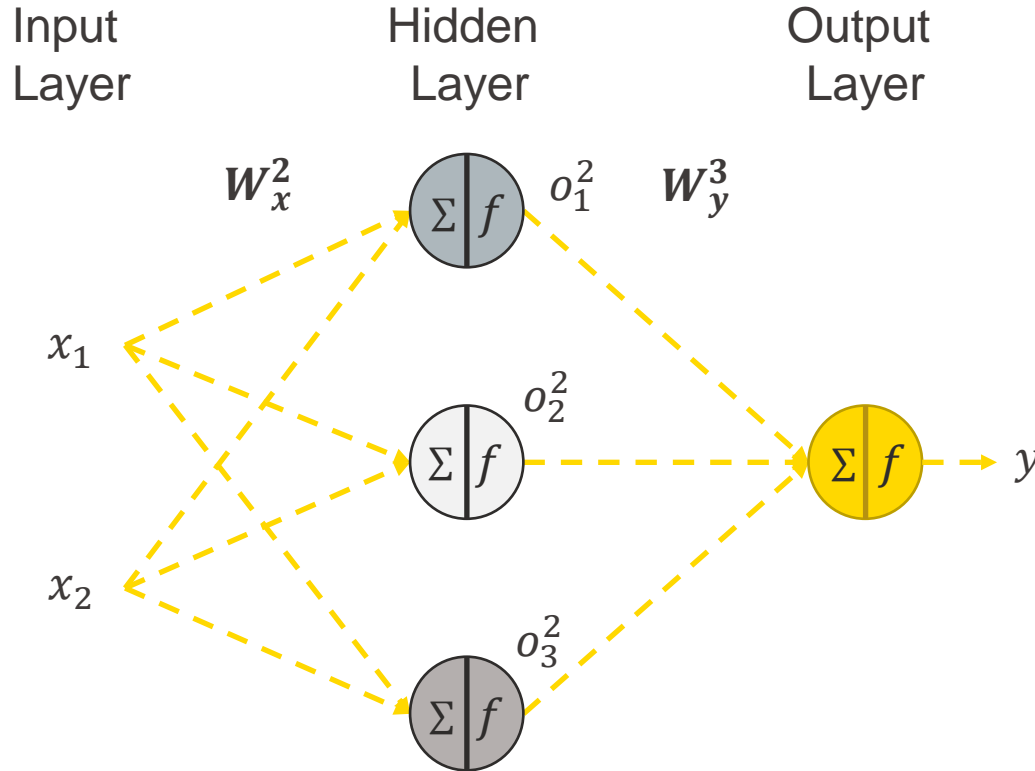


Forward pass:

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

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

Same with Matrix Notations



Forward pass:

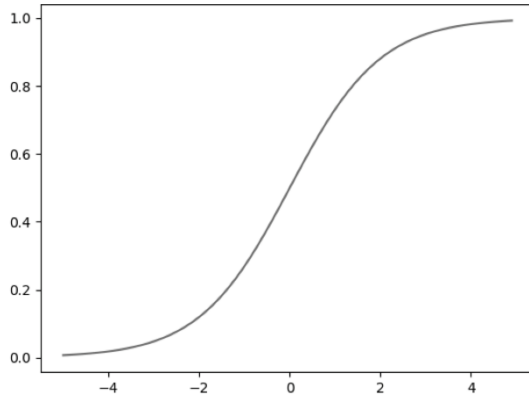
$$\mathbf{o} = f(W_x^2 \mathbf{x})$$

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

$f(\) =$ activation function

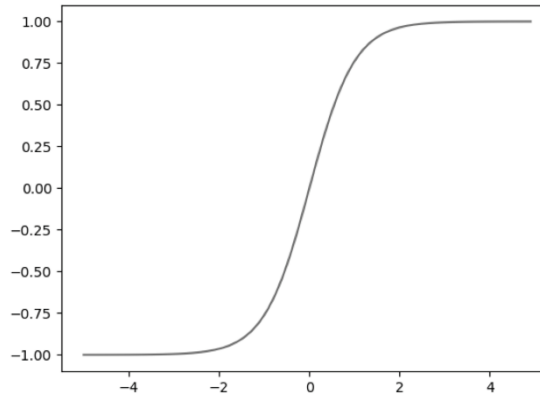
Frequently used activation functions

Sigmoid



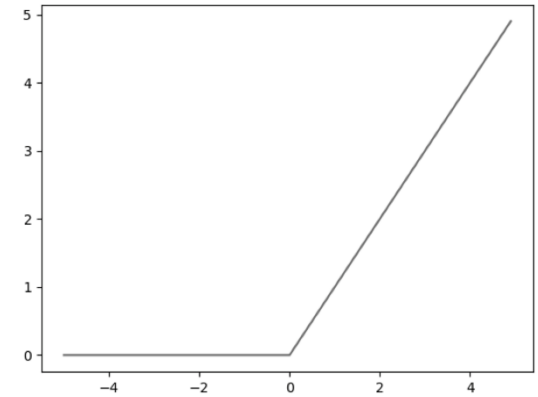
$$f(a) = \frac{1}{1 + e^{-ha}}$$

Tanh



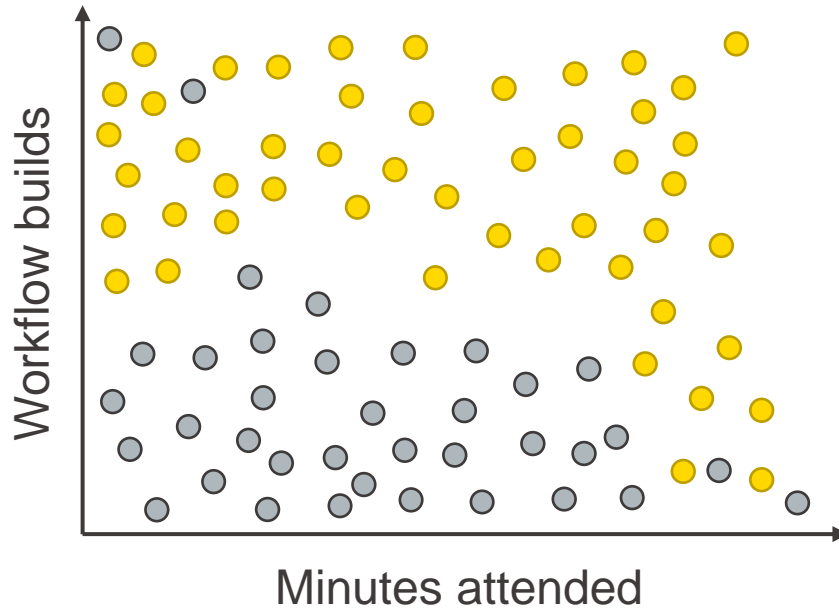
$$f(a) = \frac{e^{2ha} - 1}{e^{2ha} + 1}$$

Rectified Linear Unit (ReLU)



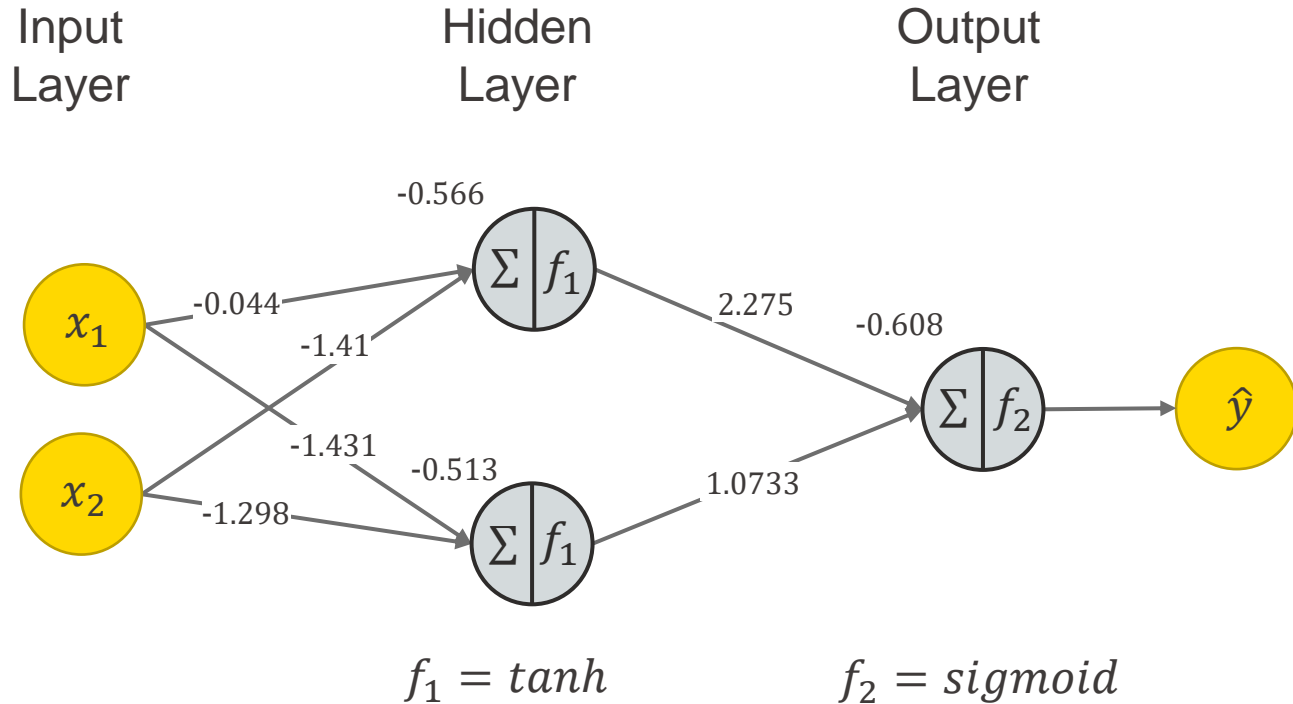
$$f(a) = \max\{0, ha\}$$

Example: Passing the KNIME L1-Certification



- Passed certification
- Didn't pass certification

Example: Passing the KNIME L1-Certification



Input features:

x_1 = minutes attended

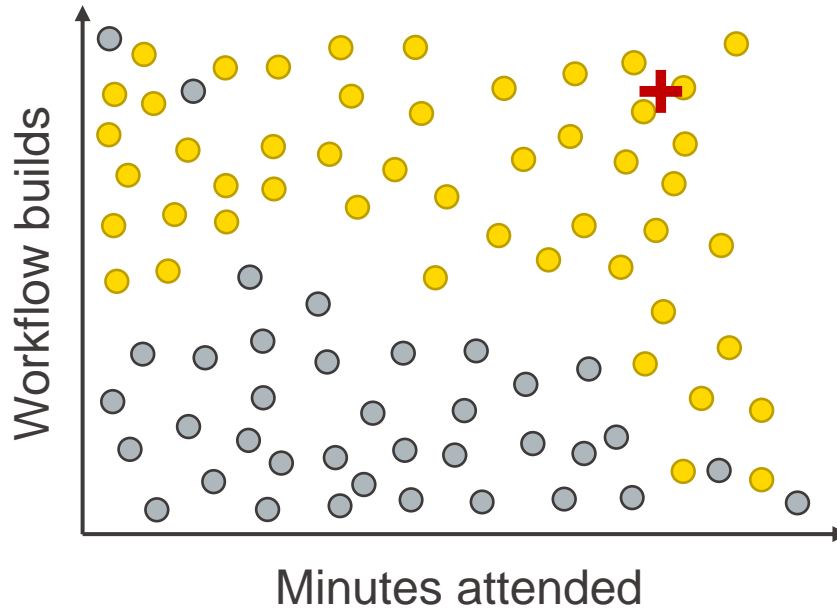
x_2 = workflows build

Output:

\hat{y} = Probability that a person passed

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

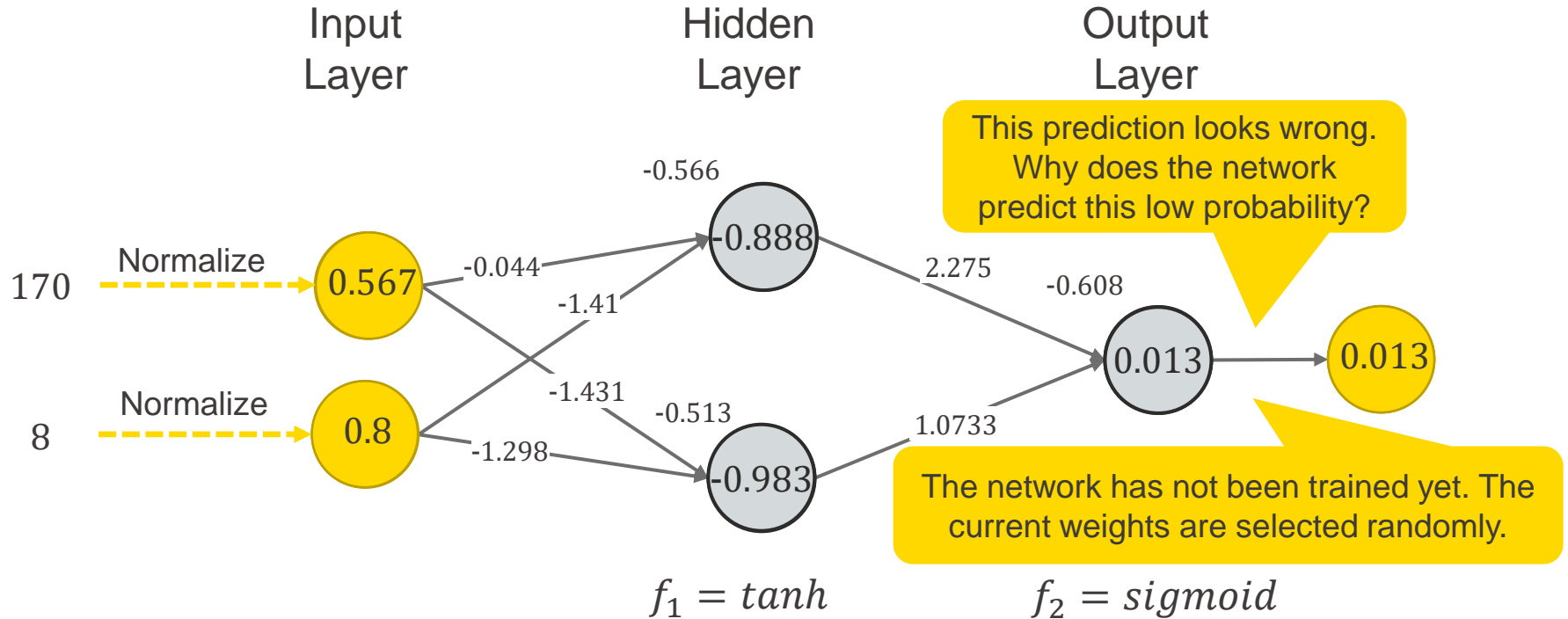
Example: Passing the KNIME L1-Certification



- 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

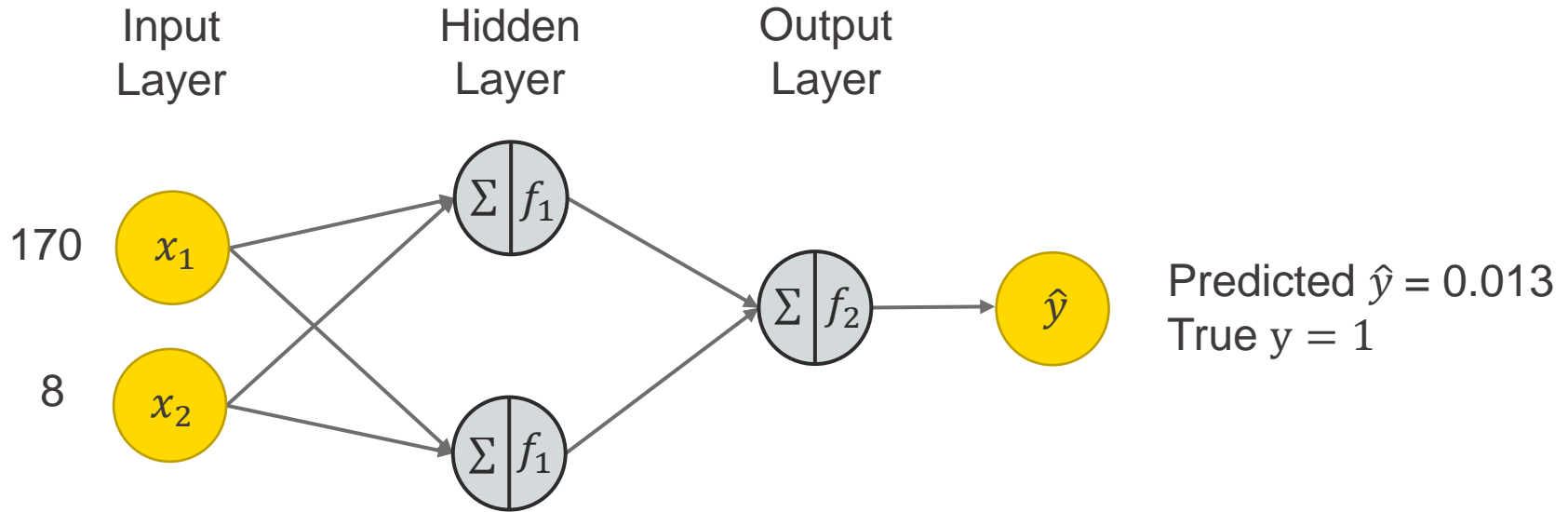
x_2 = workflows build

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

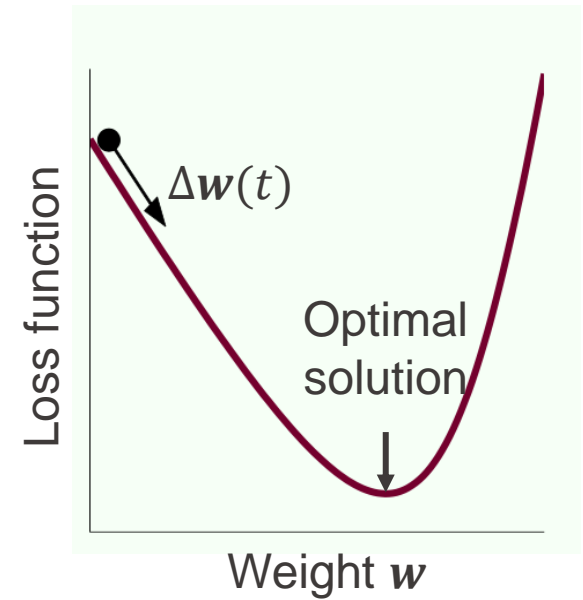
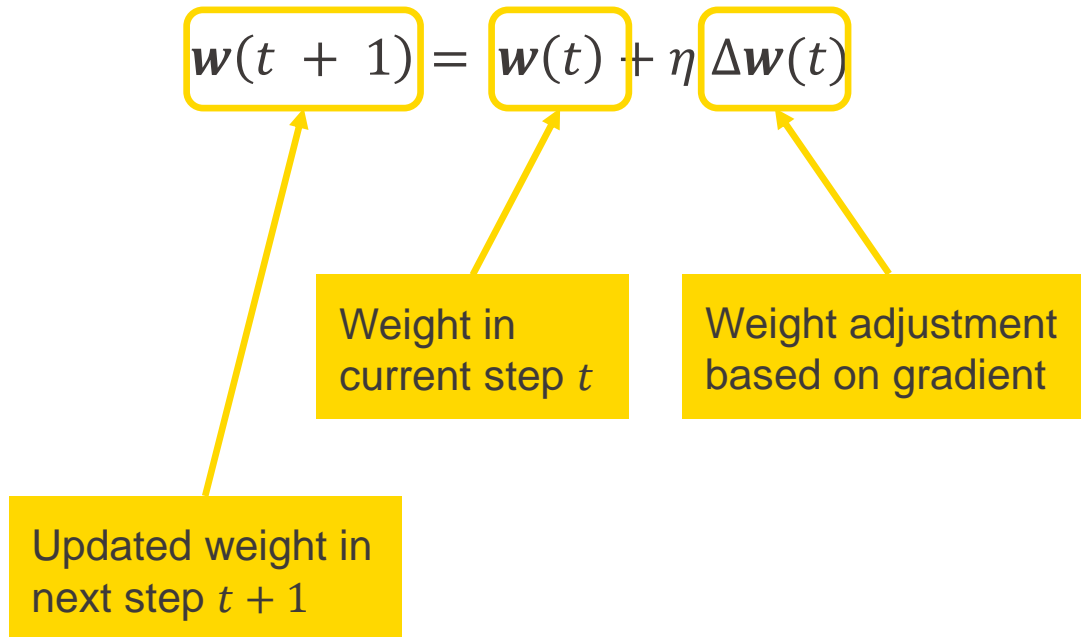


$$J(W) = \sum \mathcal{L}(\hat{y}(x_1, x_2, W), y)$$

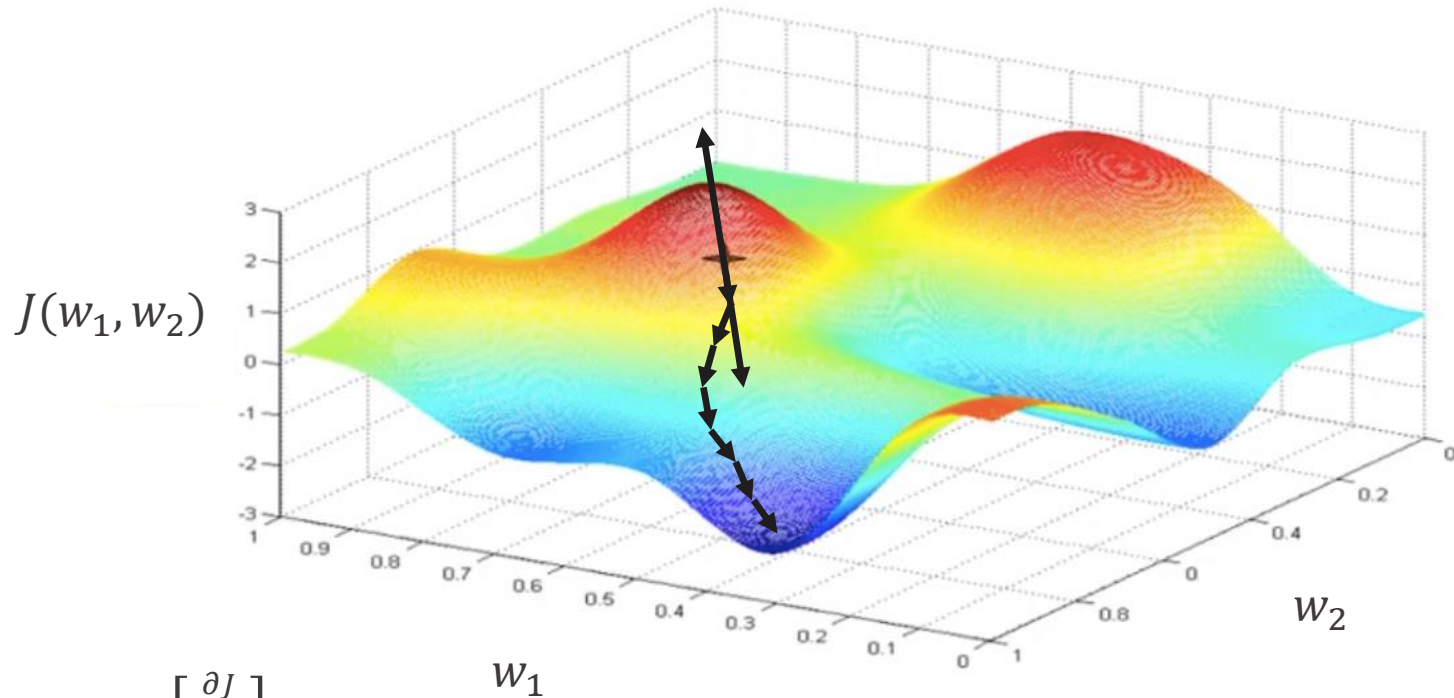
Binary cross entropy
 $\mathcal{L} = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$

Learning Rule from Gradient Descent

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



Idea Behind Gradient Descent

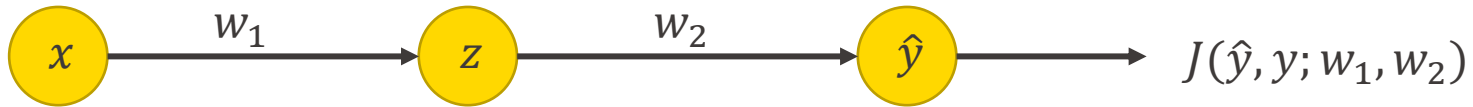


$$\nabla_W J(x, W) = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \end{bmatrix}$$

Backpropagation

- Efficient way to calculate the gradient during optimization

Forward pass



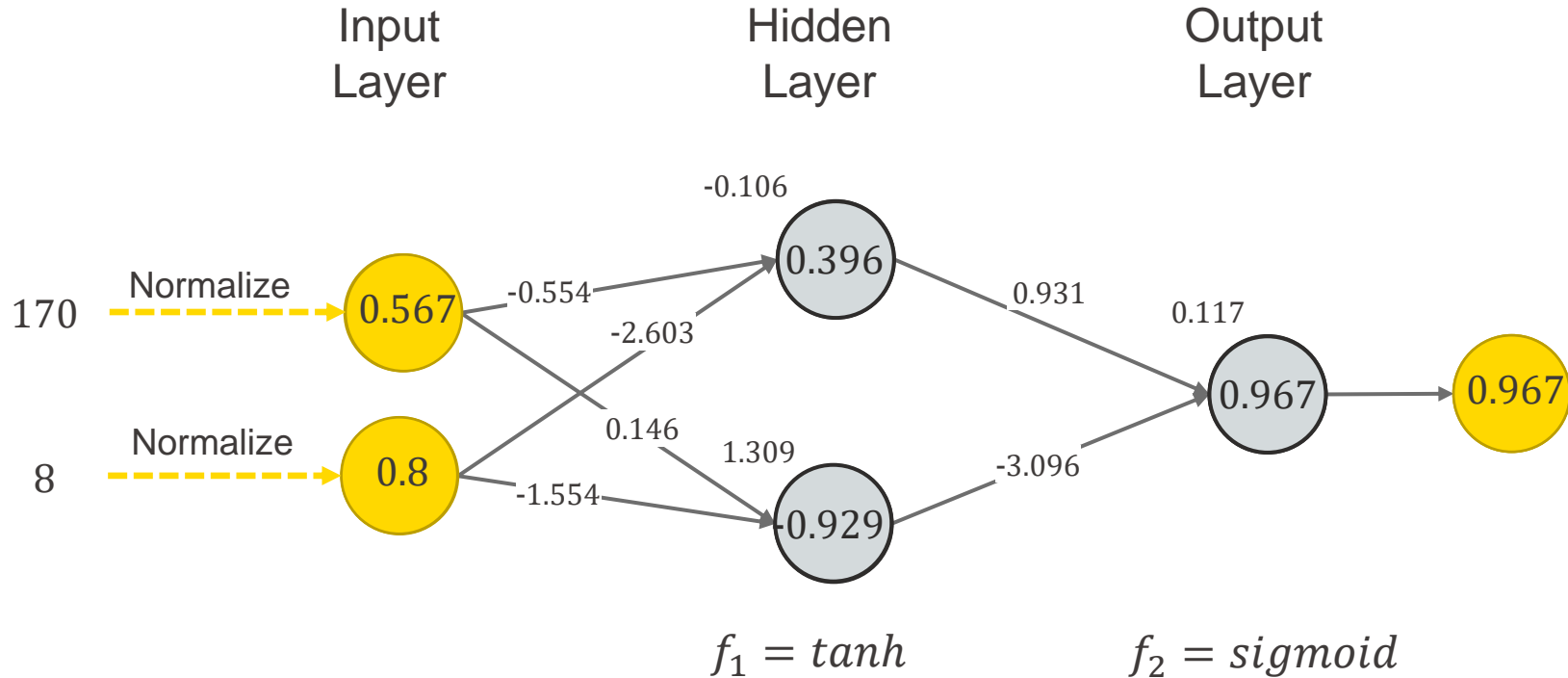
Backward pass



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

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

Example: Passing the KNIME L1 Certification



Input features:

x_1 = minutes attended

x_2 = workflows build

Output:

y = Probability that a person passed

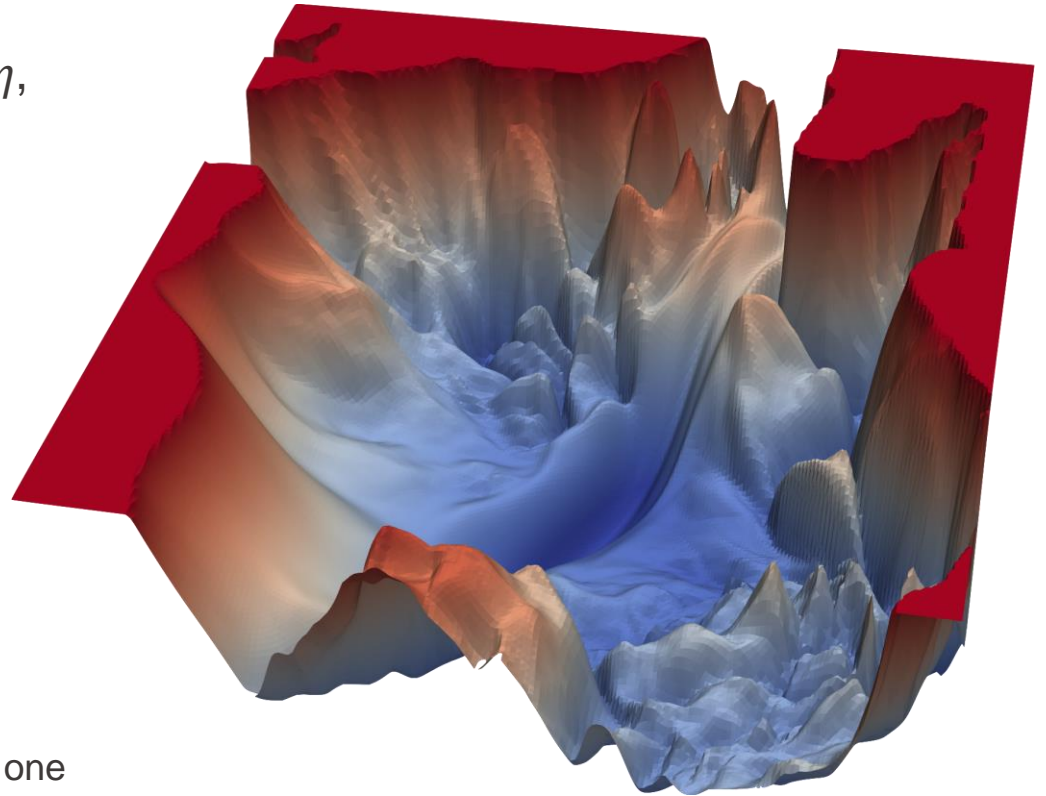
$y \geq 0.5 \Rightarrow \text{Passed}$ and $y < 0.5 \Rightarrow \text{Failed}$

Loss Landscape of a Real Neural Network

- A good choice for the step size η , aka learning rate, is important

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

- Many different optimizers with adaptive learning rates are available
 - Adam, Adadelata, 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 (optional)

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?

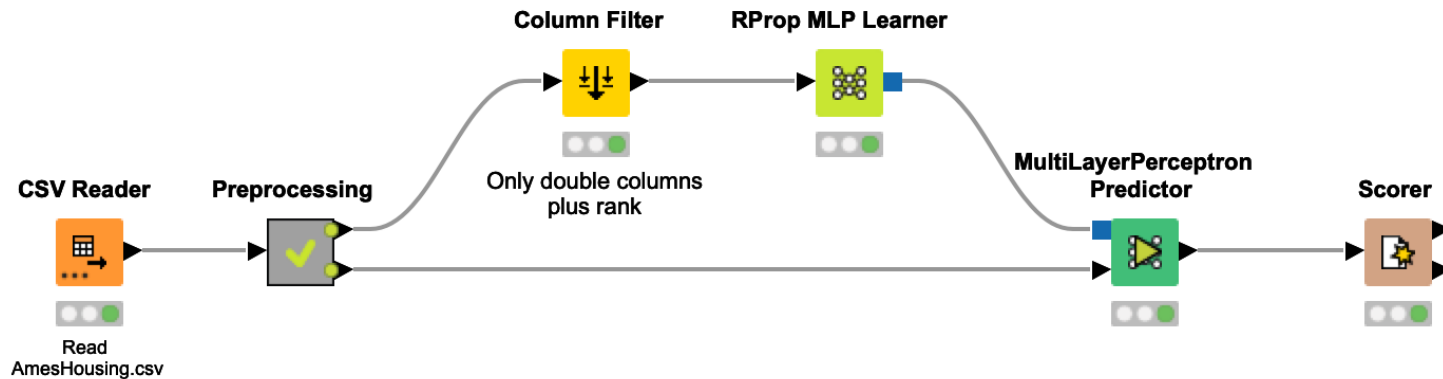
- Depends on the problem you are working on

✓ Recommended
△ Can be used

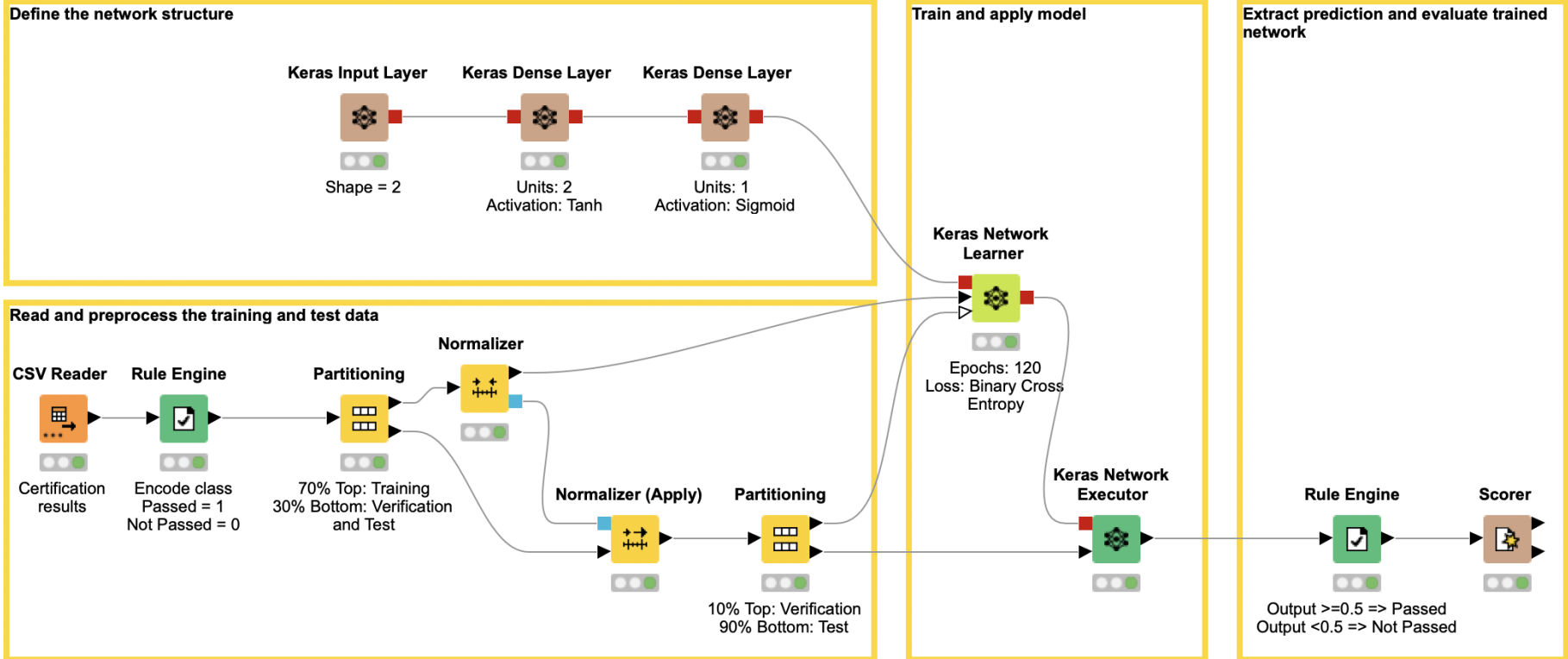
Problems		Activation Functions								Loss Functions					
		Hidden Layers			Output Layer					Binary CE	Hinge	Categorical CE	MSE	MSLE	MAE
		Sigmoid	Tanh	ReLU	Sigmoid	Tanh	Linear	ReLU	Softmax						
Classification	Binary classification (0 vs 1)	✓	✓	✓	✓					✓					
	Binary classification (-1 vs 1)	✓	✓	✓		✓					✓				
	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

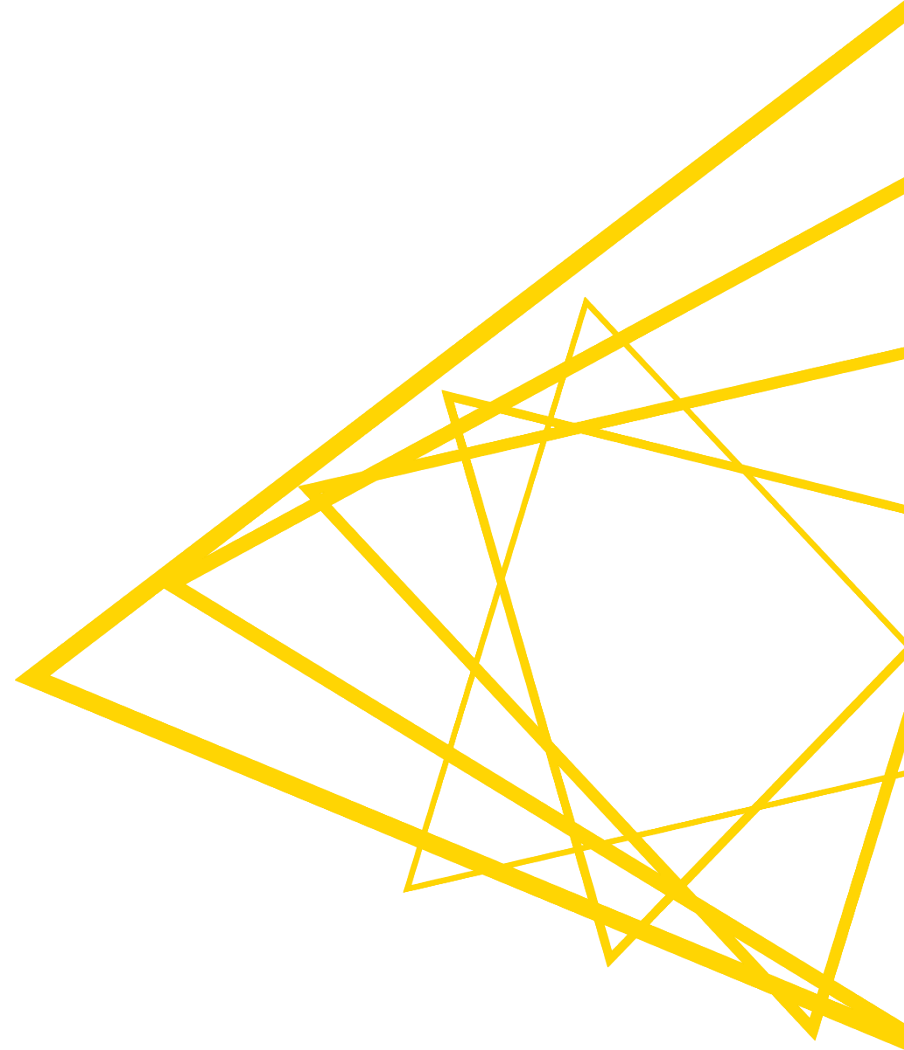


Exercise

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

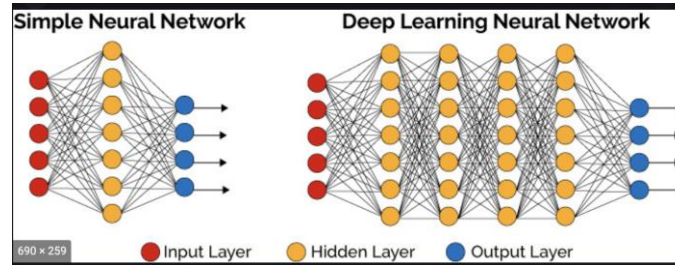
- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - >  Session_2
 - ▼  Session_3
 - ▼  01_Exercises
 - ▲ 01_Simple_Neural_Network_exercise
 - ▲ 02_Build_Association_Rules_for_MarketBasketAnalysis_exercise
 - ▲ 03_Apply_Association_Rules_for_MarketBasketAnalysis_exercise
 - ▼  02_Solutions
 - ▲ 01_Simple_Neural_Network_solution
 - ▲ 02_Build_Association_Rules_for_MarketBasketAnalysis_solution
 - ▲ 03_Apply_Association_Rules_for_MarketBasketAnalysis_solution

Deep Learning



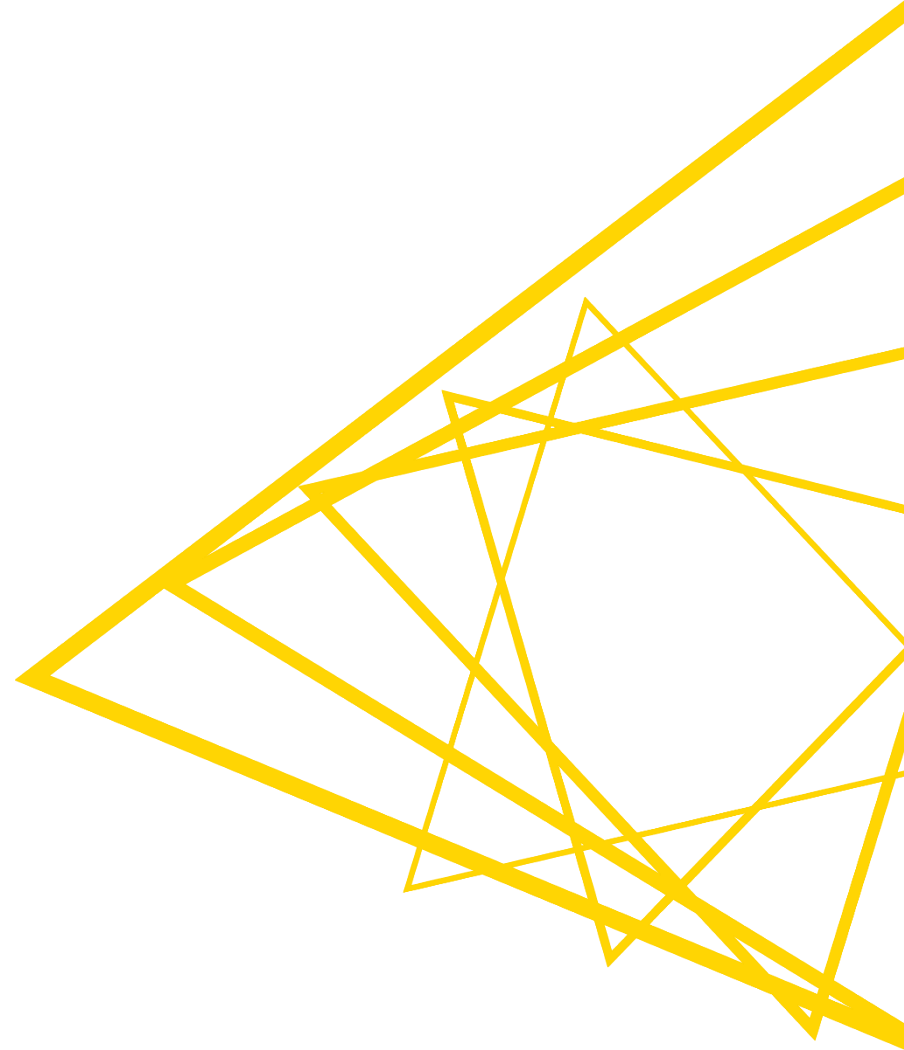
From Neural Network To Deep Learning

- Deep feed forward network



- Many additional layer types
 - Convolutional Layers for Images
 - Image classification, Image segmentation
 - Recurrent Layers for sequential data (join our next webinar)
 - Time series prediction, language models, neural machine translation
- New architectures
 - GANs
 - Transformer networks

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

Neural Network: Code-free

Define Network

Keras Input Layer



Input Shape
?, Dictionary Size

Keras LSTM Layer



Output: Sequence of
Hidden States

Keras Dropout Layer



Regularization

Keras Dense Layer



Activation:
Linear

Keras Dense Layer



Activation:
Softmax

Read and Pre-Process Input Data

Pre-Processing



Train Network

Keras Network
Learner



Edit and Save Networks

DL Python
Network Editor



Add Temperature
and Remove Dropout

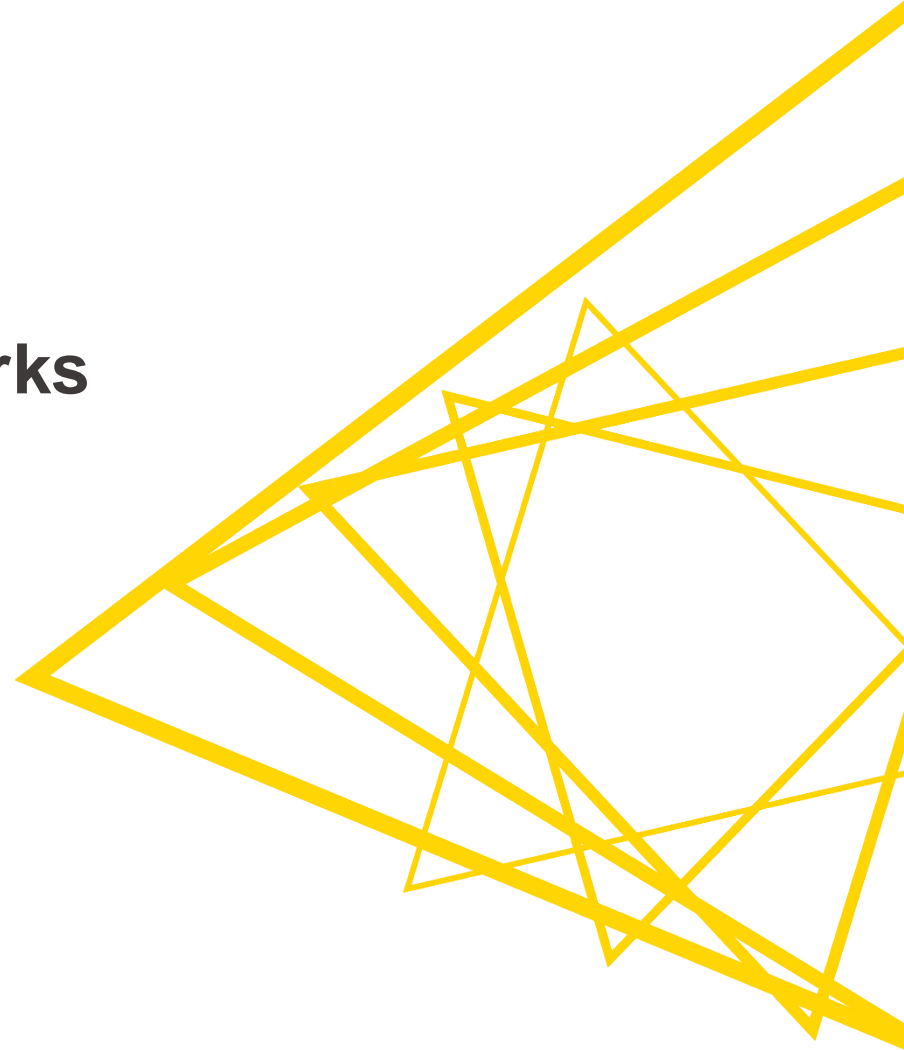
Keras to TensorFlow
Network Converter



TensorFlow
Network Writer



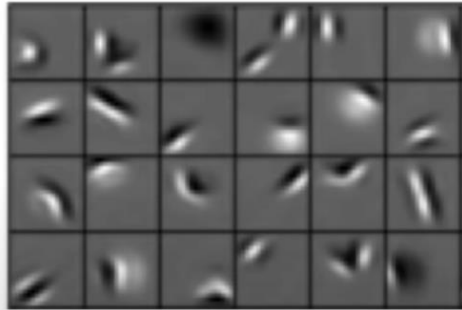
Convolutional Neural Networks (CNN)



Convolutional Neural Network (CNN)

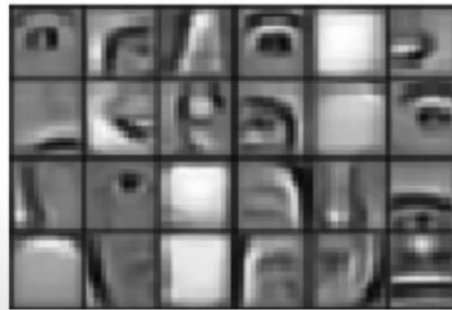
- A CNN is a neural network with at least one convolutional layer.
- CNNs are commonly used when data has spatial relationships, e.g. images
- CNN learns a hierarchy of features using multiple convolution layers that detect different features.

Low level features



Edges, dark spots

Mid level features



Eyes, ears, nose

High level features



Facial structure

Images from: 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

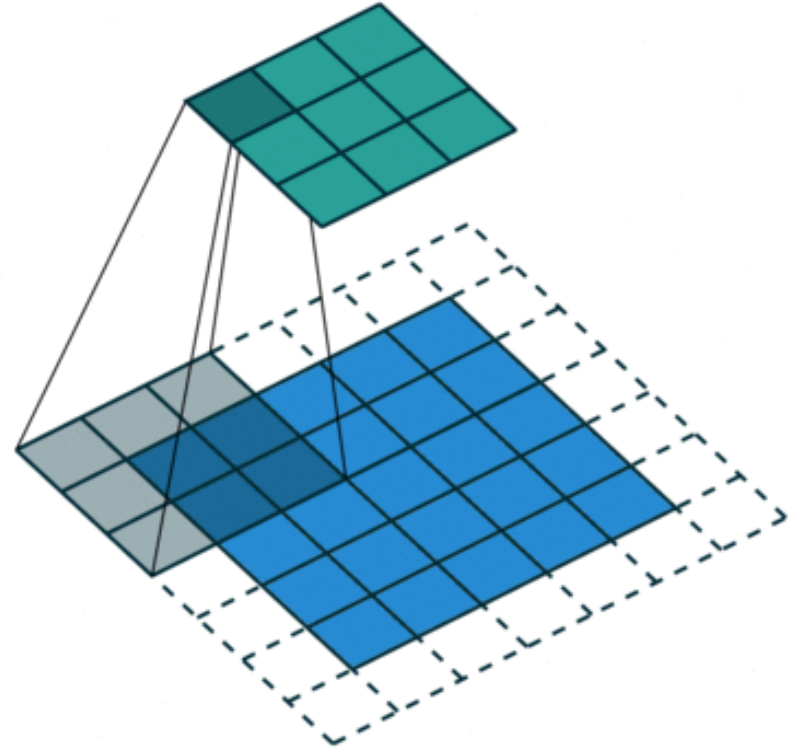
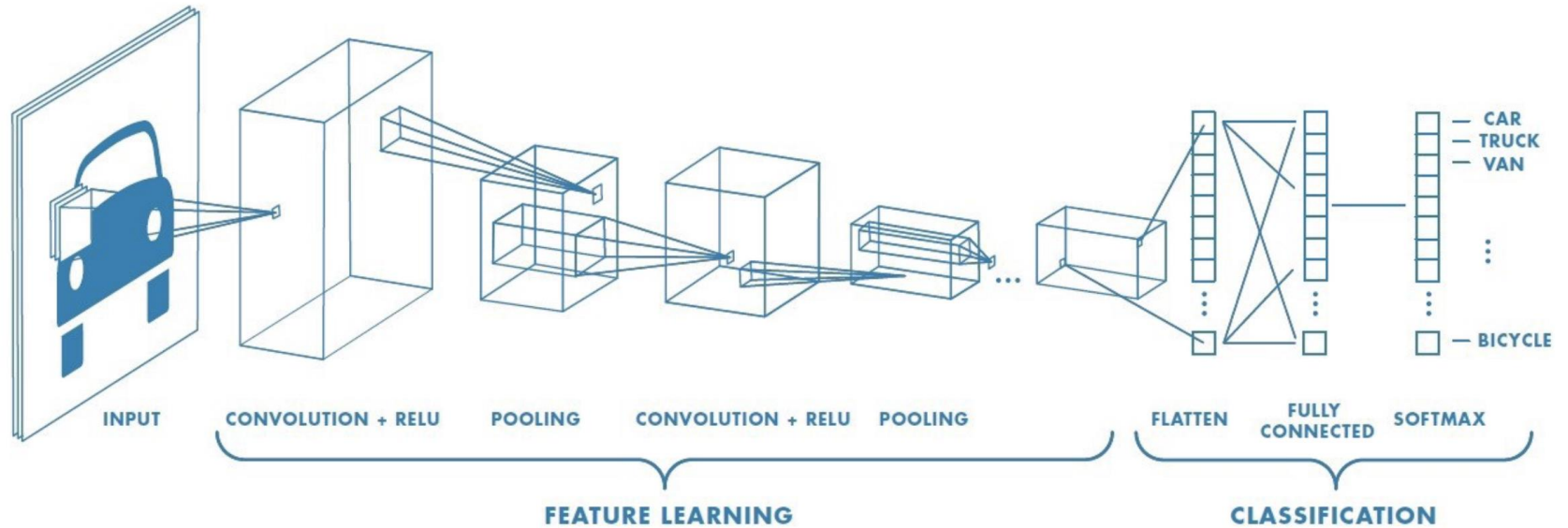
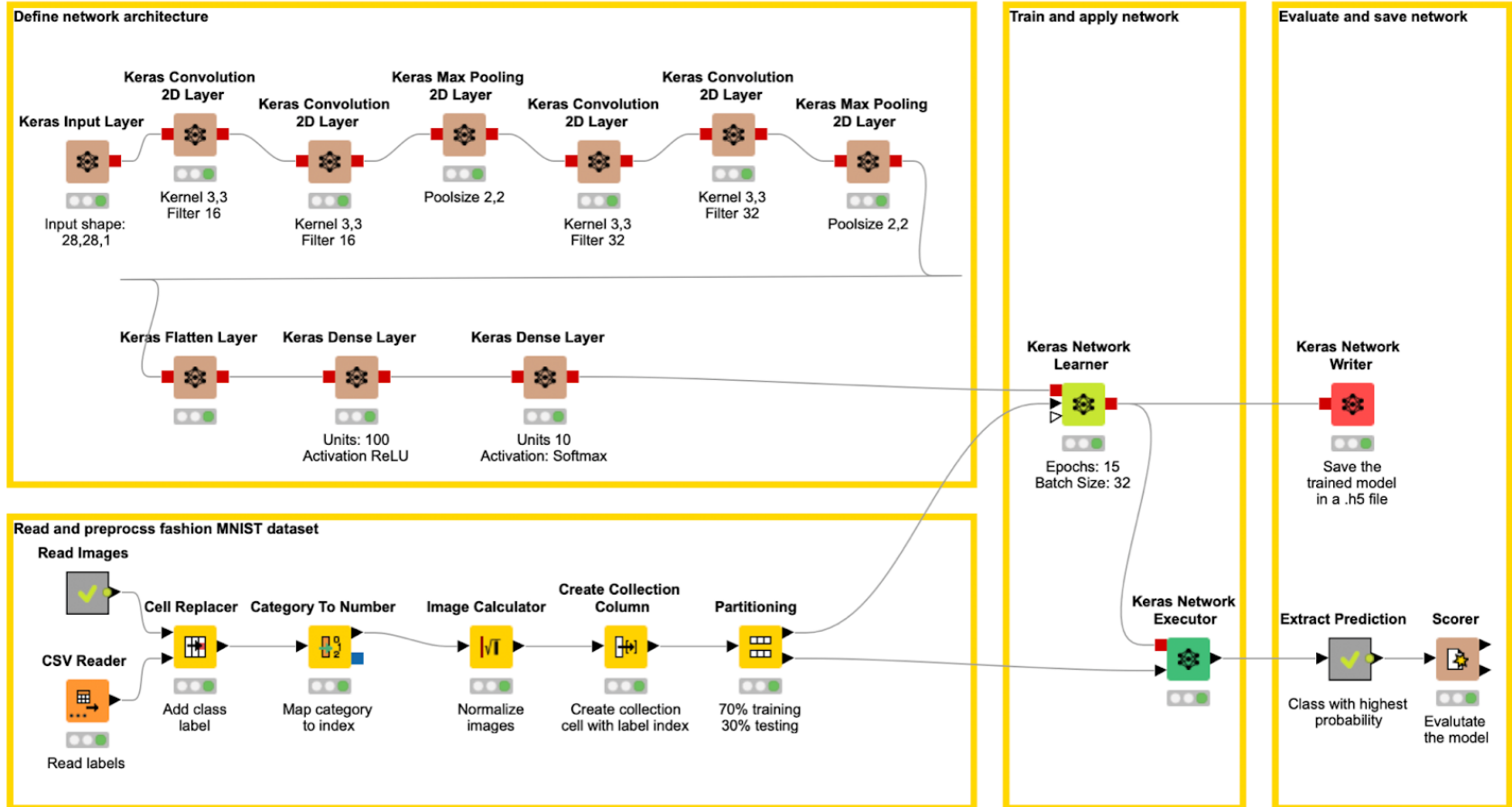


Image from: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

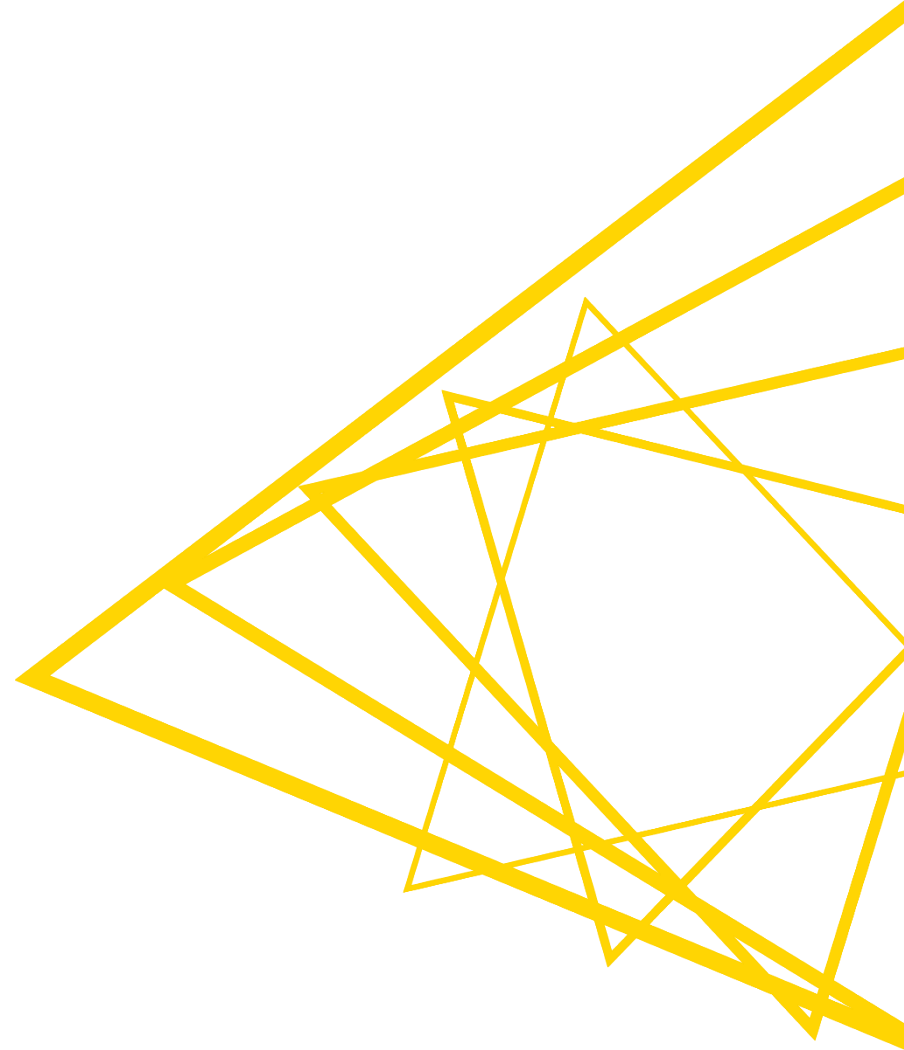
Convolutional Neural Networks



Building CNNs with KNIME



Unsupervised Learning: Clustering

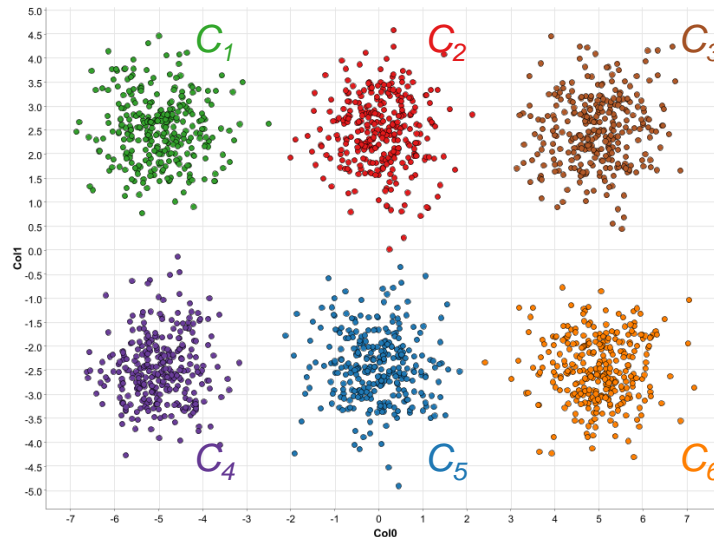


Goal of Clustering Analysis

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



Clustering Applications

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

Methods

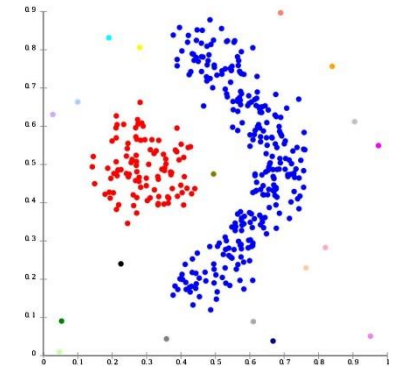
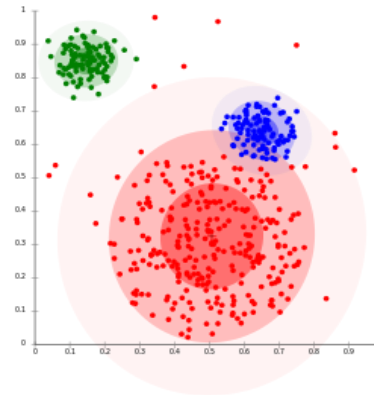
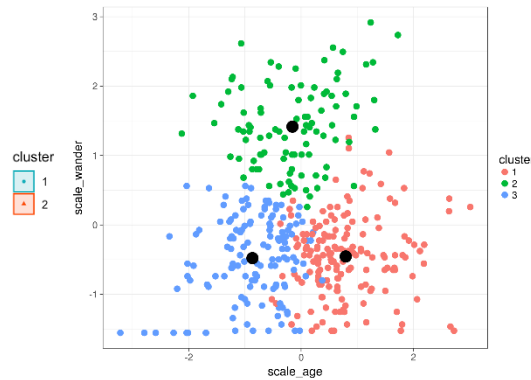
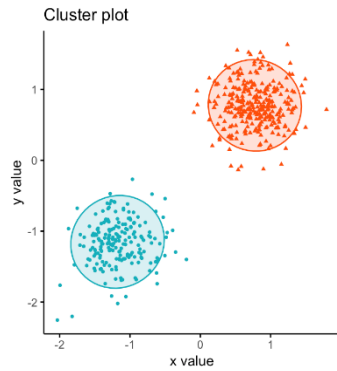
- K-means
- Hierarchical
- DBScan

Examples

- Customer segmentation
- Molecule search
- Anomaly detection

Cluster Properties

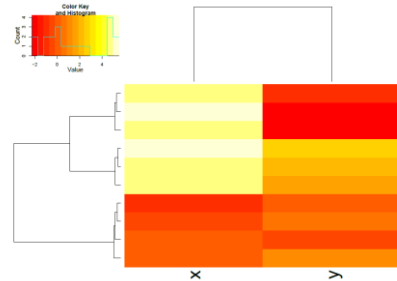
- Clusters may have different sizes, shapes, densities
- Clusters may form a hierarchy
- Clusters may be overlapping or disjoint



Types of Clustering Approaches

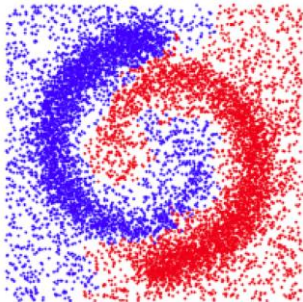
Linkage Based

e.g. Hierarchical Clustering



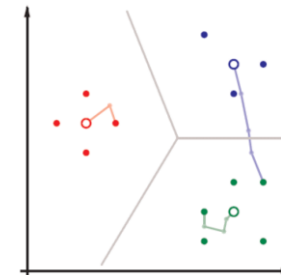
Density based Clustering

e.g. DBSCAN



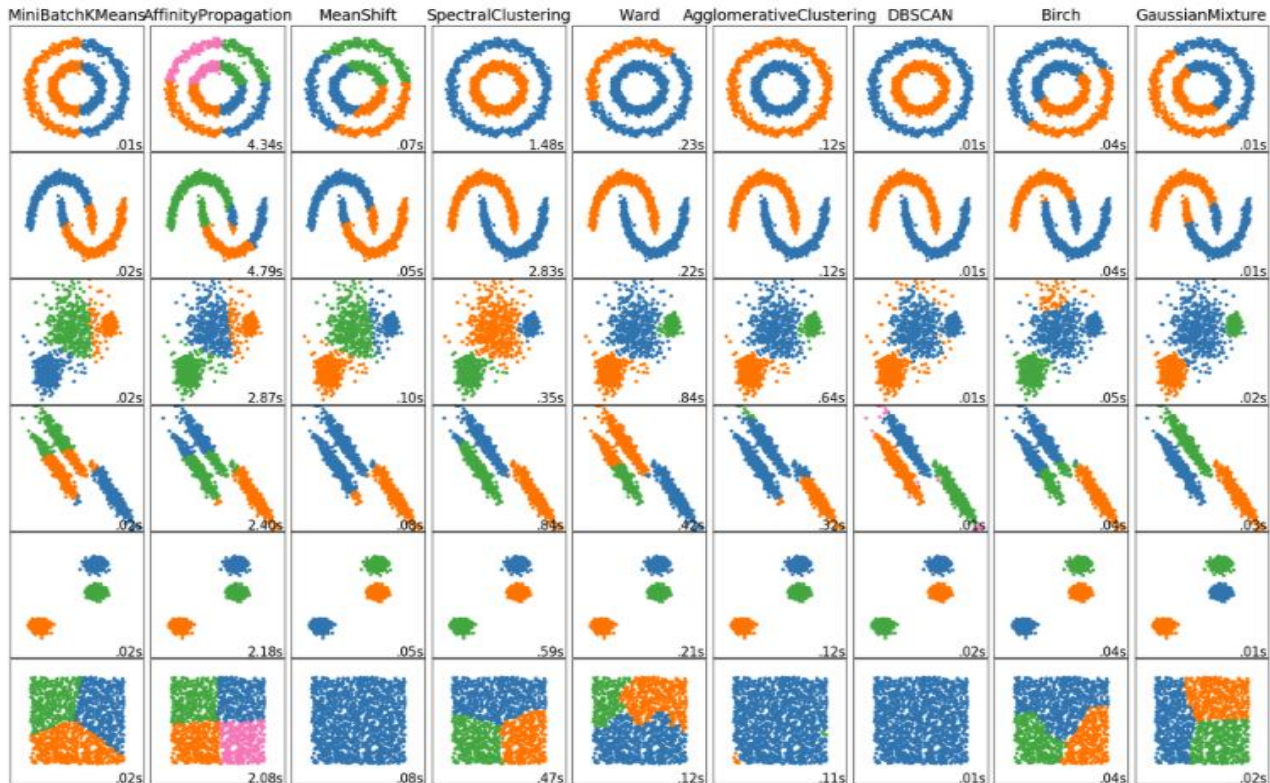
Clustering by Partitioning

e.g. k-Means

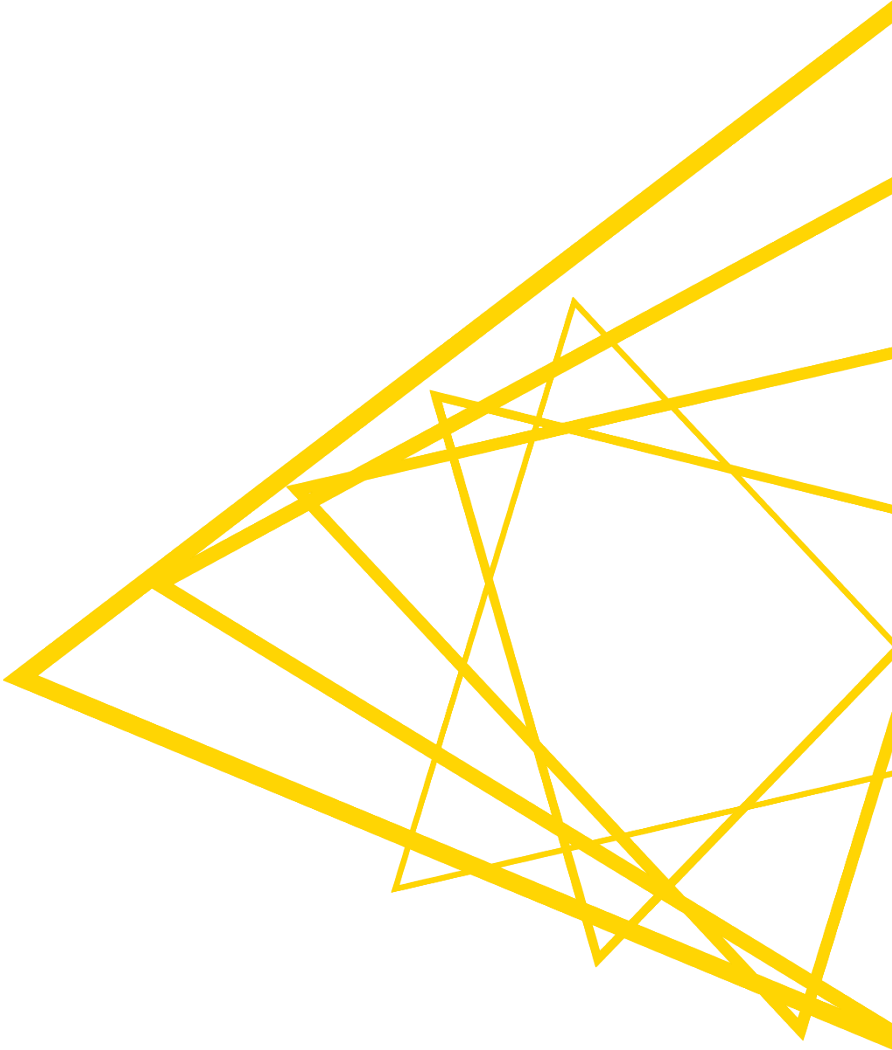


Types of Clustering Approaches

- No clustering method works universally well



Clustering: Partitioning k-Means

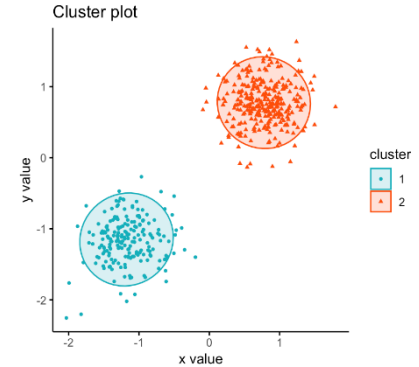


Partitioning

Goal:

A (disjoint) partitioning into k clusters with minimal costs

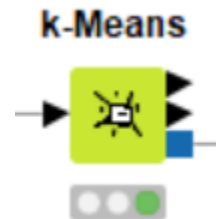
- Local optimization method:
 - choose k initial cluster representatives
 - optimize these representatives iteratively
 - assign each object to its **most similar cluster** representative
- Types of cluster representatives:
 - Mean of a cluster (*construction of central points*)
 - Median of a cluster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



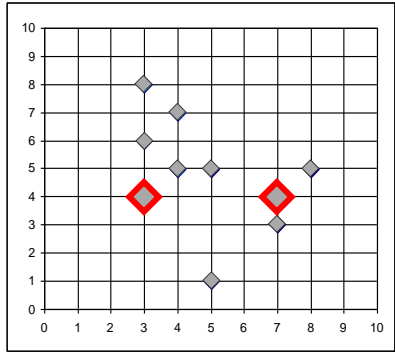
k-Means-Algorithm

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

1. Randomly choose k objects as the initial centroids
2. Assign each object to the cluster with the **nearest** centroid
3. Re-compute the centroids as the centers of the newly formed clusters
4. Go back to Step 2, repeat until the updated centroids stop moving significantly

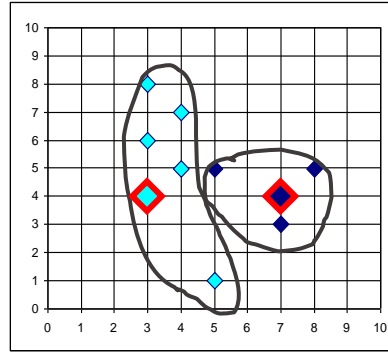


k-Means Algorithm

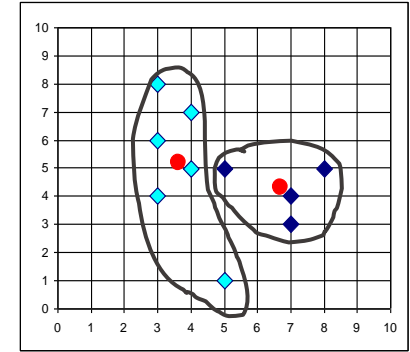


Centroids randomly chosen

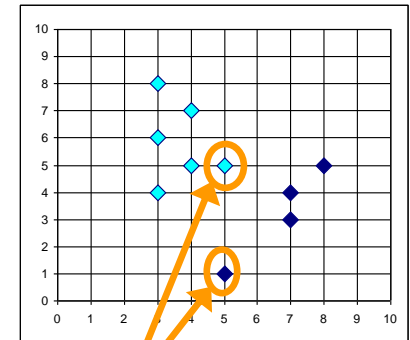
Cluster assignment



Calculation of new centroids

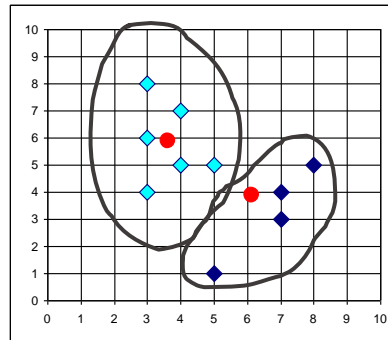


Cluster assignment ↓



Changes in cluster assignment

Calculation of new centroids



Repeat until cluster centers stop moving

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

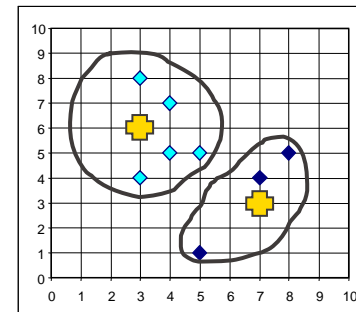
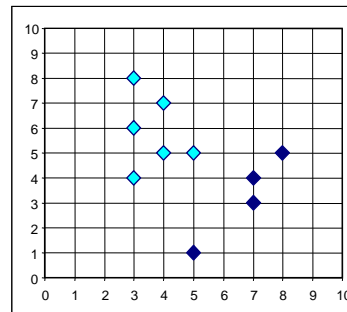
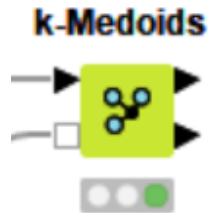
Outliers: k-Means vs k-Medoids

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



Influence of Distance Function / Similarity

- Clustering vehicles:

- red Ferrari
- green Porsche
- red Bobby car



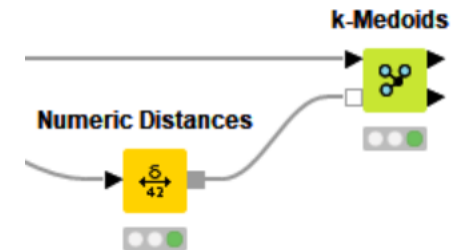
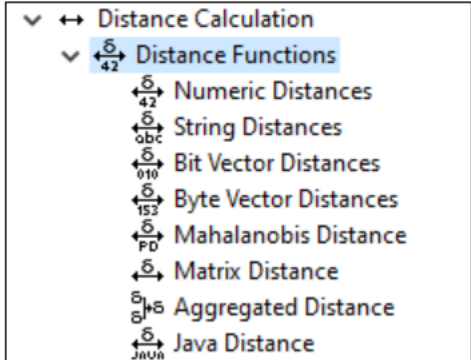
- Distance Function based on maximum speed (numeric distance function):

- Cluster 1: Ferrari & Porsche
- Cluster 2: Bobby car

- Distance Function based on color (nominal attributes):

- Cluster 1: Ferrari and Bobby car
- Cluster 2: Porsche

The distance function affects the shape of the clusters



Distance 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)*

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

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

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

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

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

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

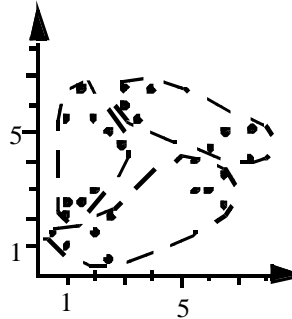
$$\text{dist}(x, y) = \max_{1 \leq i \leq d} \{|x_i - y_i|\}$$

Clustering: Quality Measures Silhouette

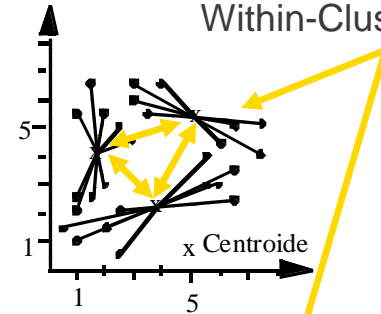


Optimal Clustering: Example

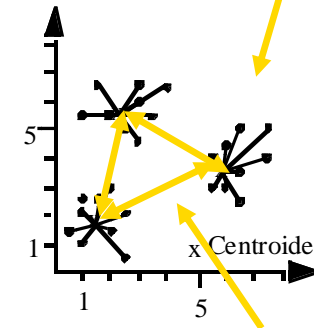
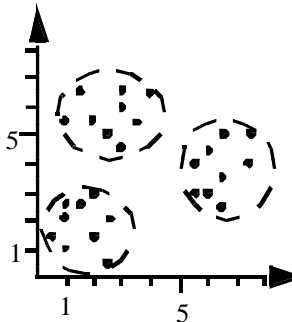
Bad Clustering



Within-Cluster Variation



Good Clustering



Between-Cluster Variation

Cluster Quality Measures

Centroid μ_C : mean vector of all objects in clustering C

- Within-Cluster Variation:

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

- 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

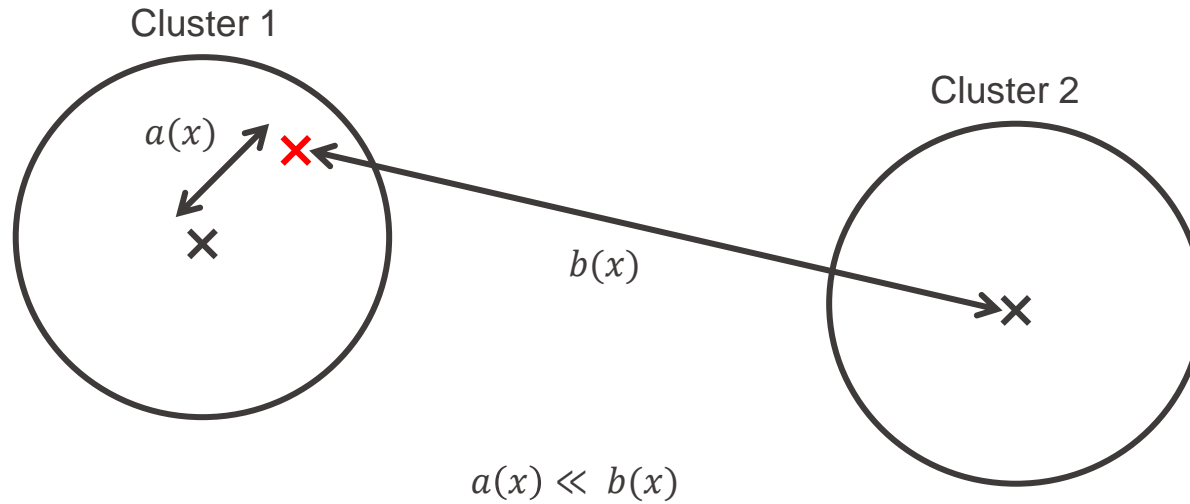
Silhouette-Coefficient [Kaufman & Rousseeuw 1990] measures the quality of clustering

- $a(x)$: 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

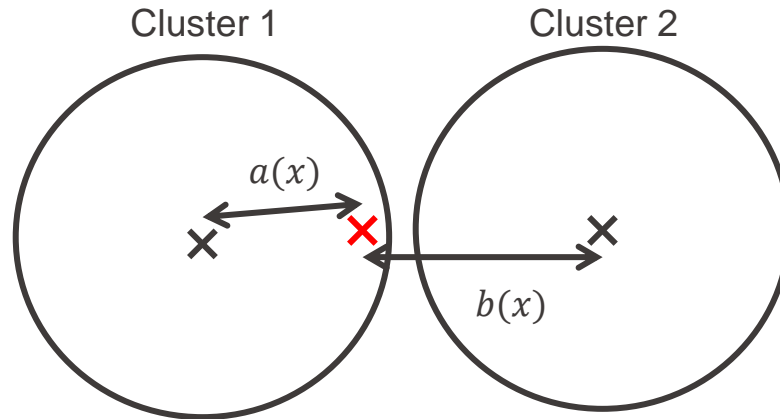
Good clustering...



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

Silhouette-Coefficient

...not so good...

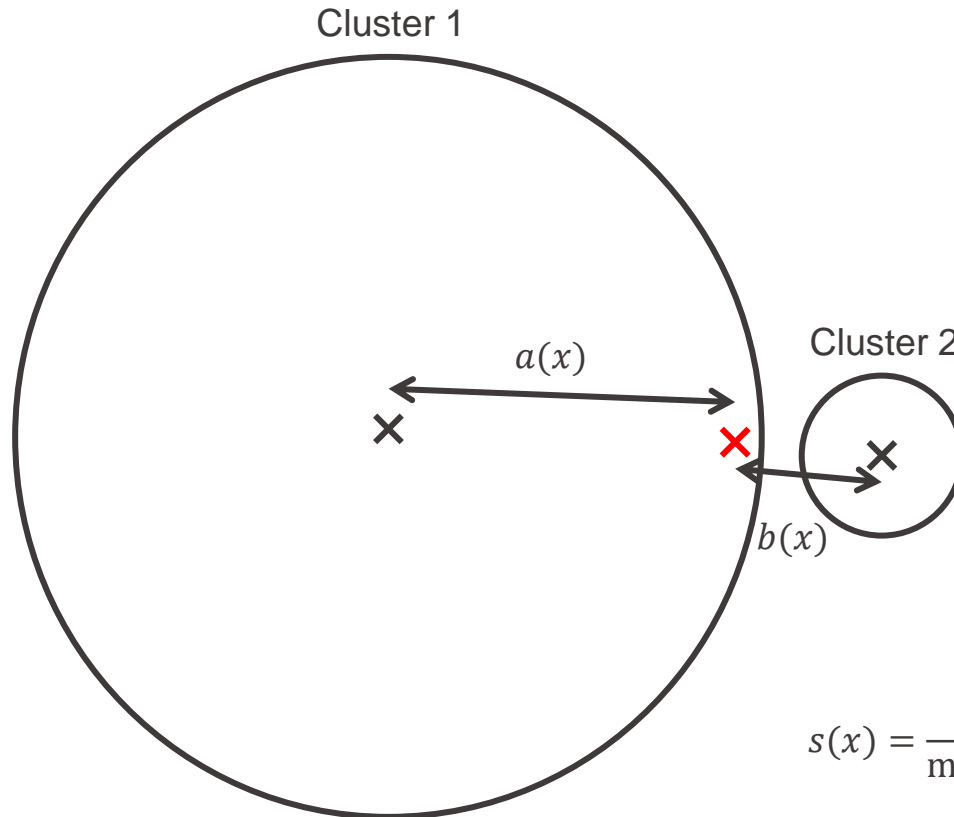


$$a(x) \approx b(x)$$

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

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

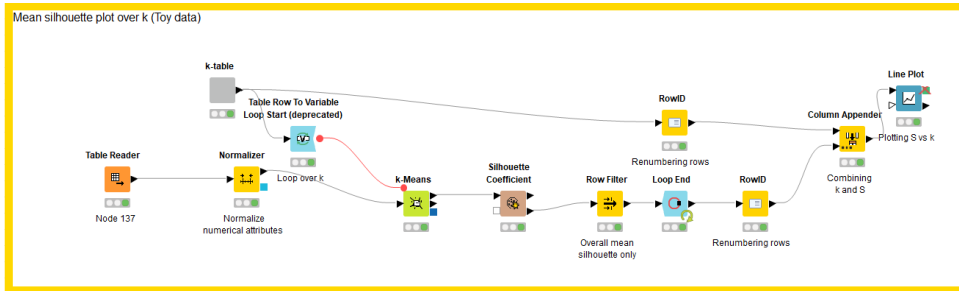
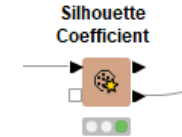
- Silhouette coefficient s_c for clustering C is the average silhouette over all objects $x \in C$

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

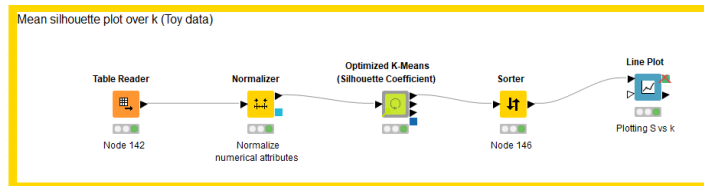
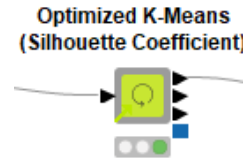
- Interpretation of silhouette coefficient:
 - $s_c > 0.7$: strong cluster structure,
 - $s_c > 0.5$: reasonable cluster structure,
 - ...

Silhouette Coefficient over a Range of k

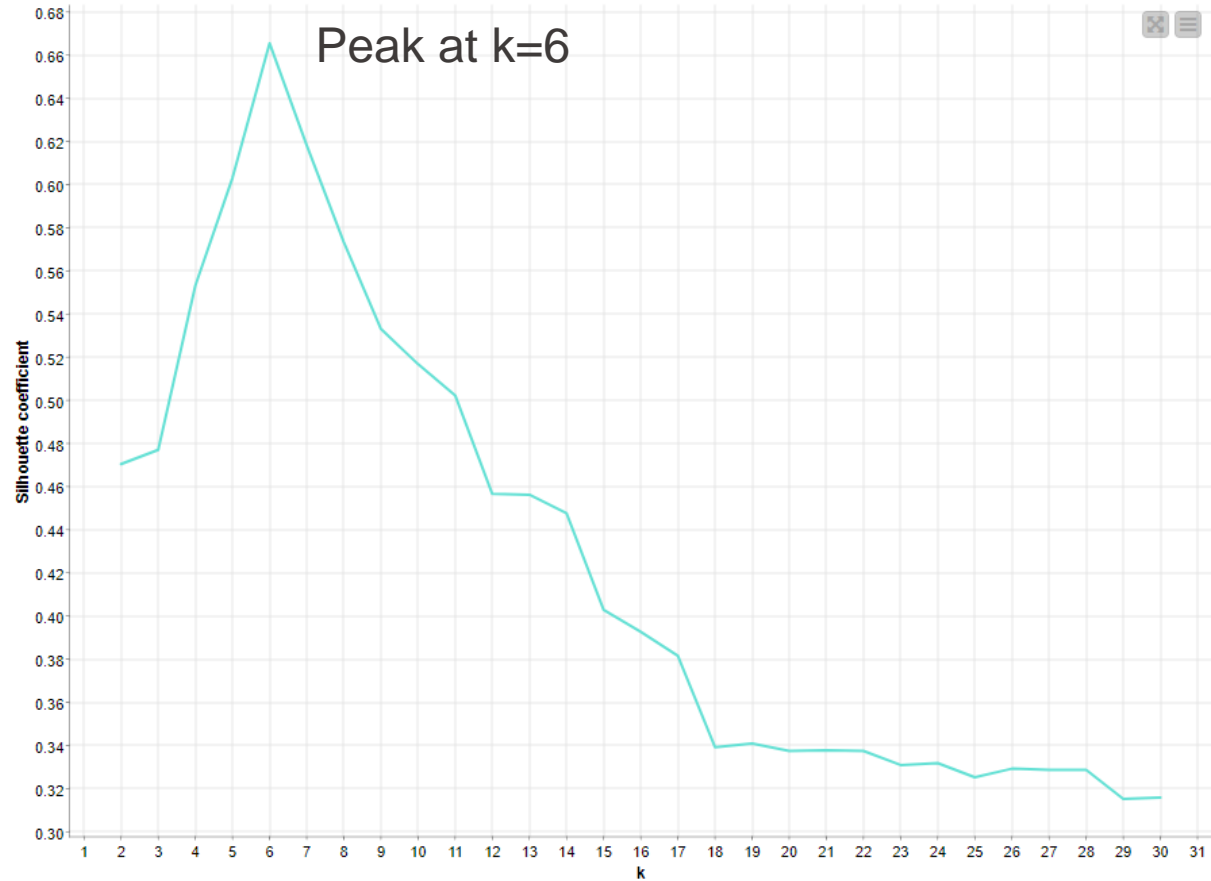
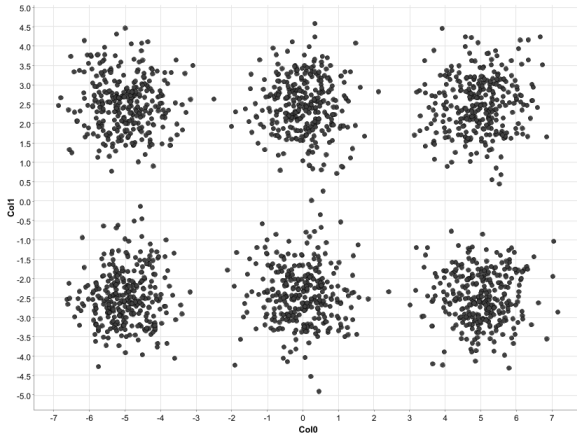
- Silhouette Coefficient Node in KNIME Analytics Platform
- Loop over various values of k



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



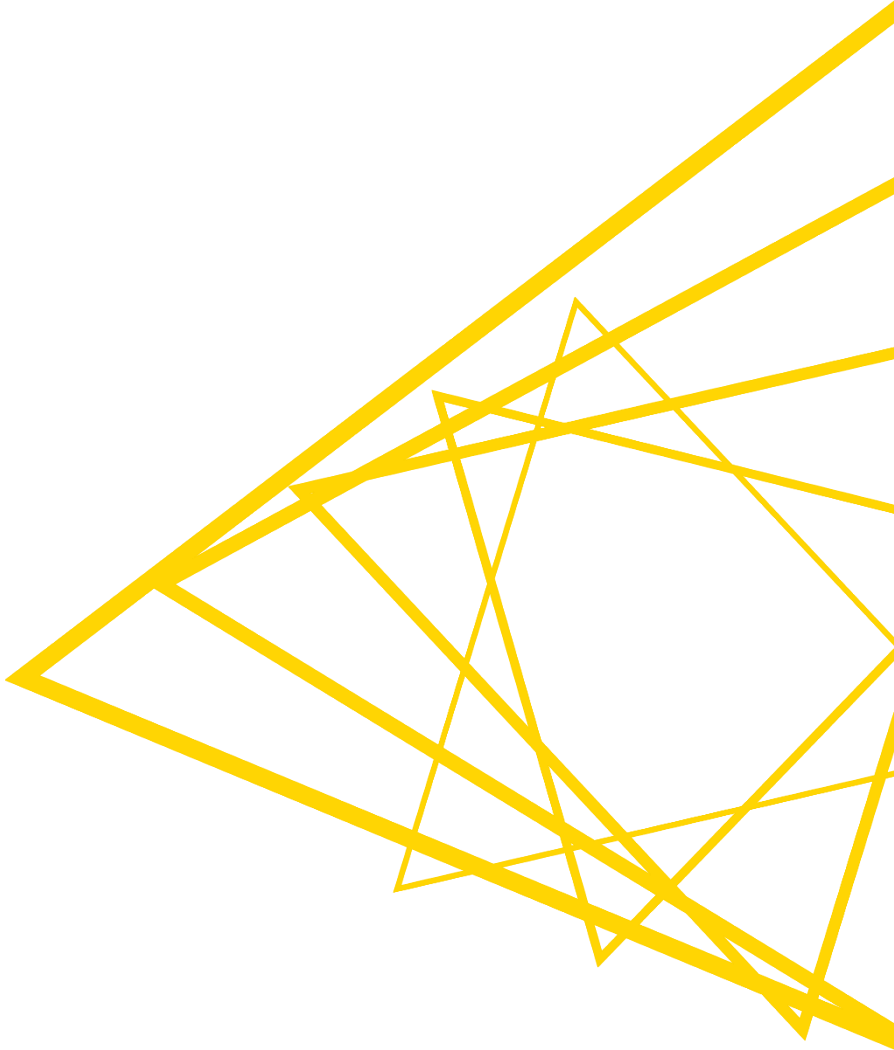
Silhouette Coefficient over k



Summary: Clustering by Partitioning

- Scheme always similar:
 - Find (random) starting clusters
 - Iteratively update centroid 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



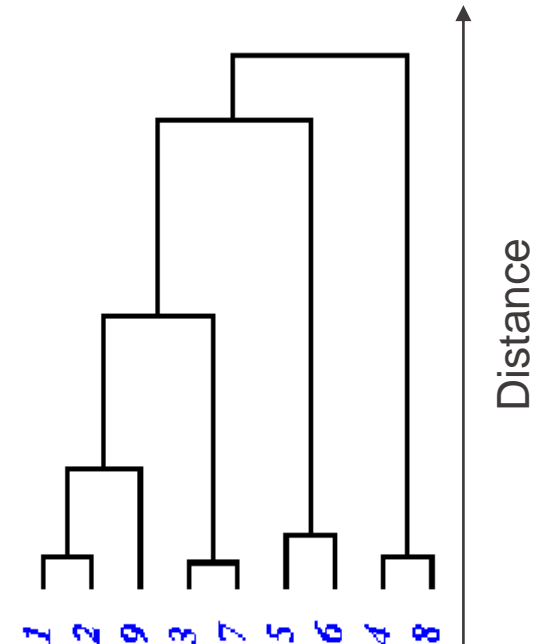
Linkage Hierarchies: Basics

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



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.

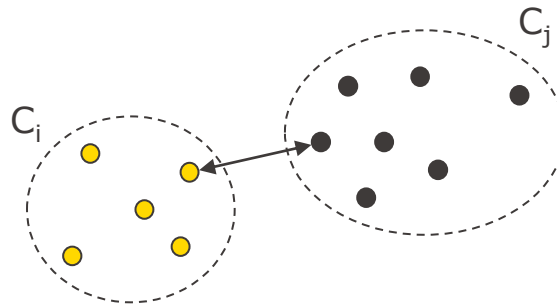
Single Linkage

- 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



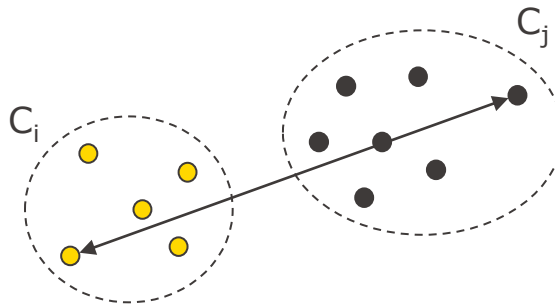
Complete Linkage

- 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

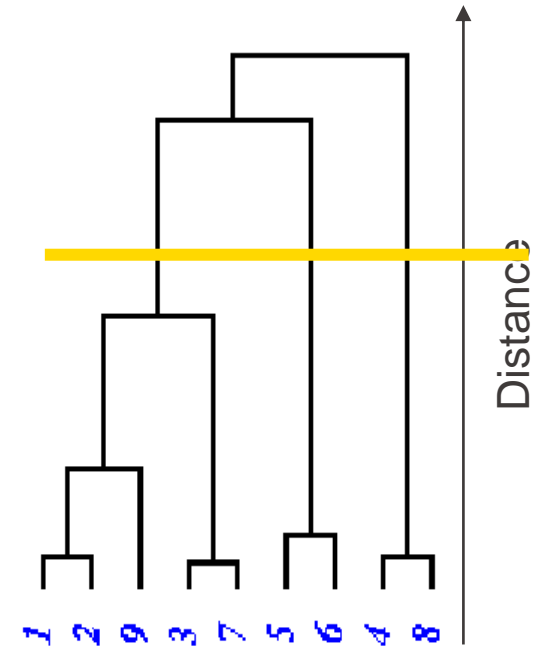
$$Dist_{mean}(C_1, C_2) = dist(mean(C_1), mean(C_2))$$

Distance between two centroids

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

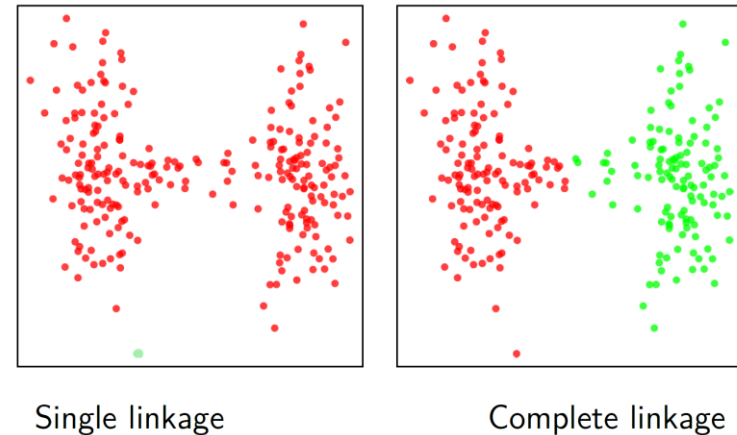
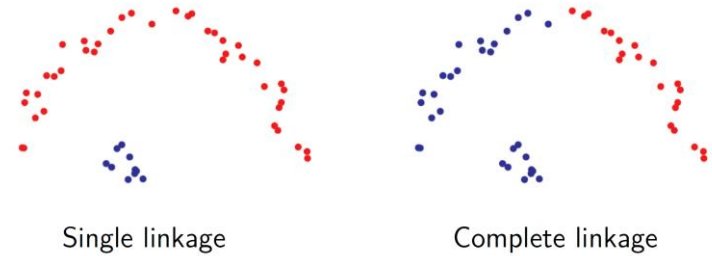
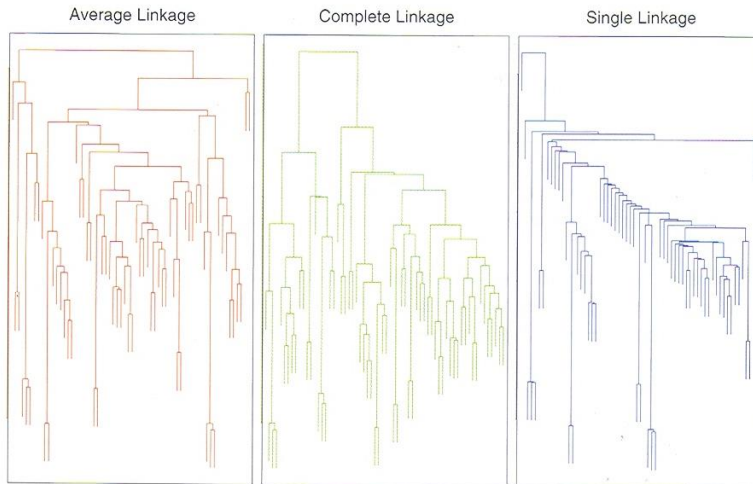
Comments on Single Linkage and Variants

- + 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

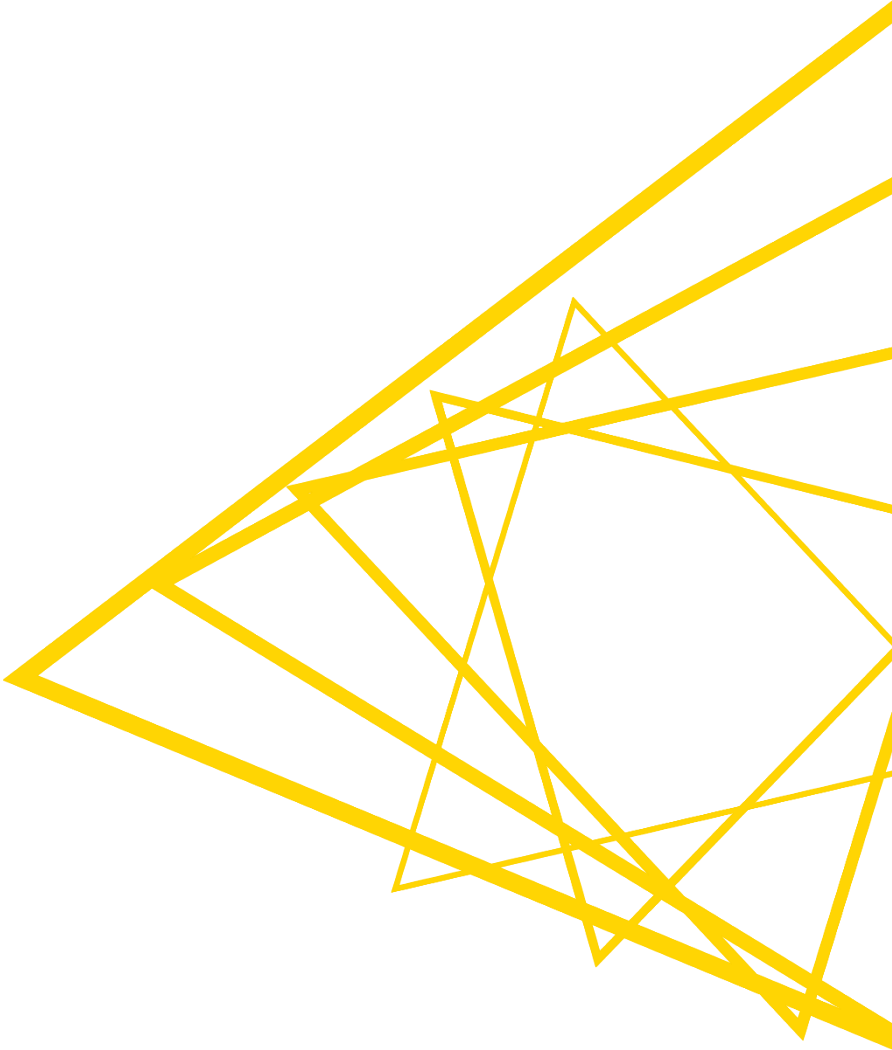


Linkage Based Clustering

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



**Clustering: Density
DBSCAN**



Clustering: DBSCAN

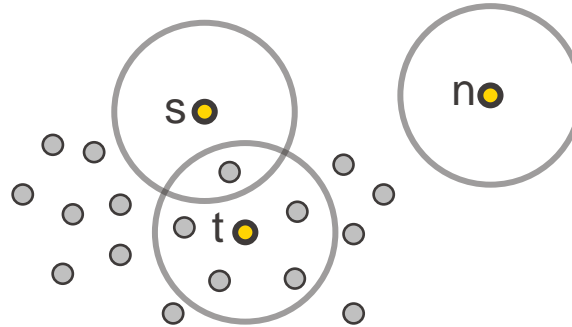
DBSCAN - a density-based clustering algorithm - defines five types of points in a dataset.

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

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

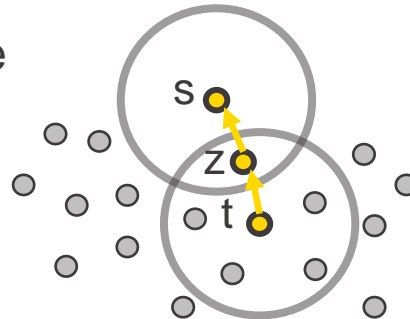
Example with MinPts = 3

Core Point
vs. Border Point
vs. Noise



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

Directly Density Reachable
vs. Density Reachable

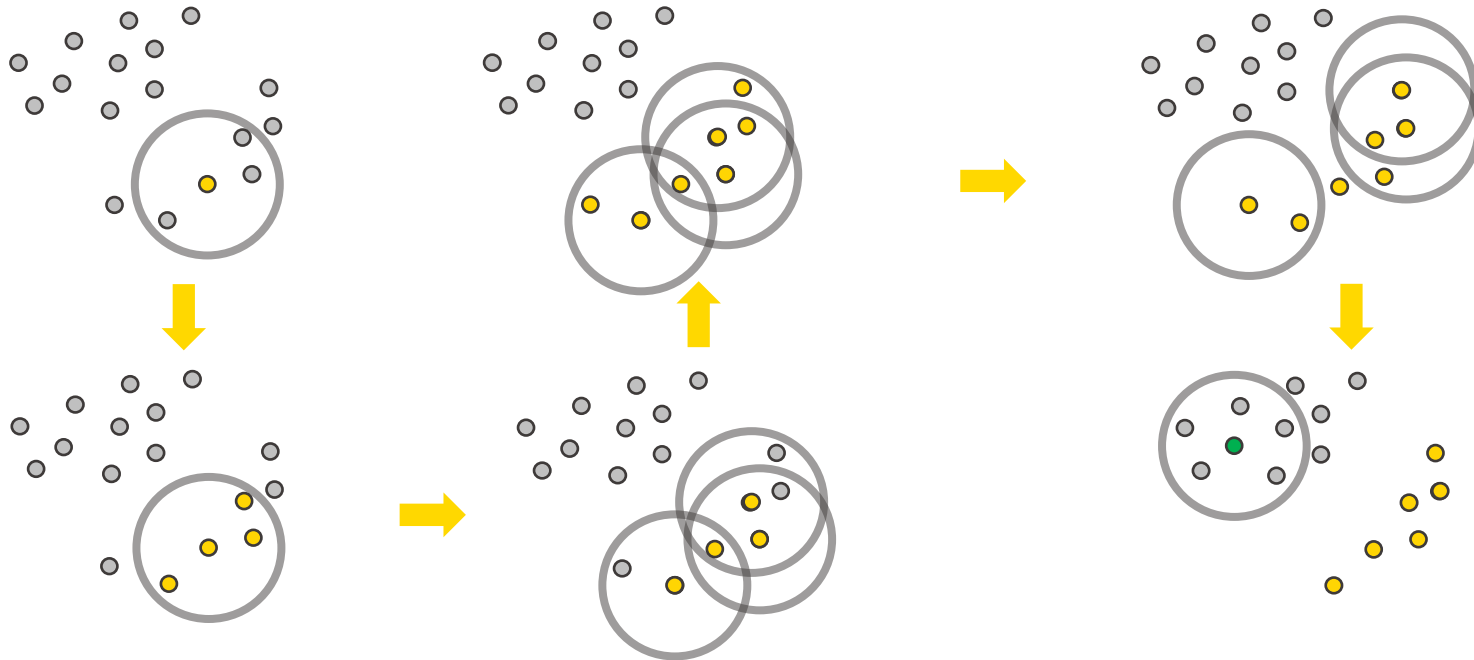


- z is directly density reachable from t
- s is not directly density reachable from t, but density reachable via z

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

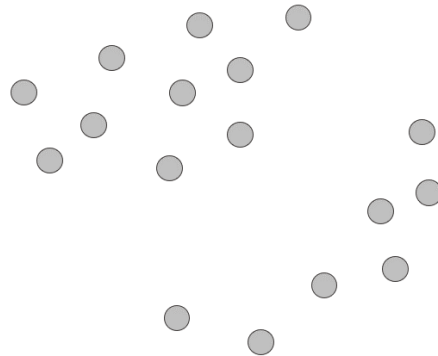
DBSCAN [Density Based Spatial Clustering of Applications with Noise]

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



DBSCAN [Density Based Spatial Clustering of Applications with Noise]

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



Summary: DBSCAN

Clustering:

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

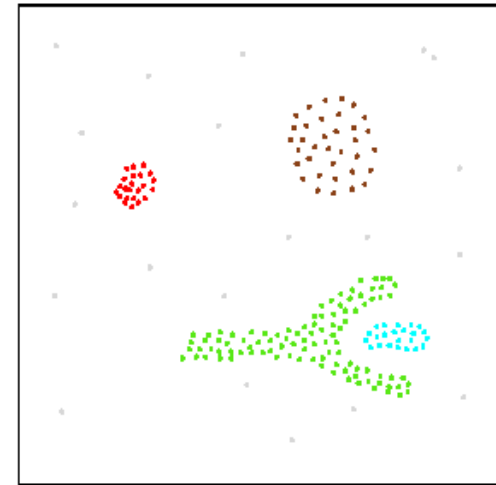
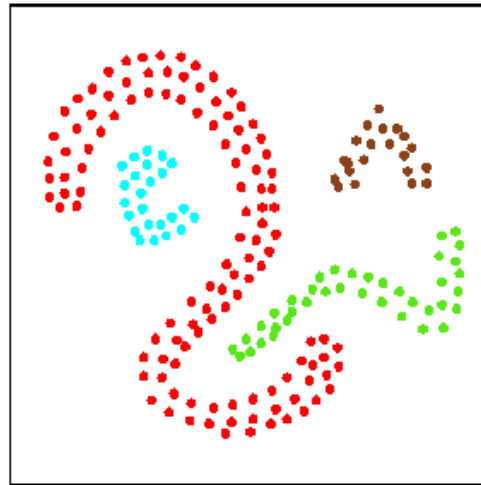
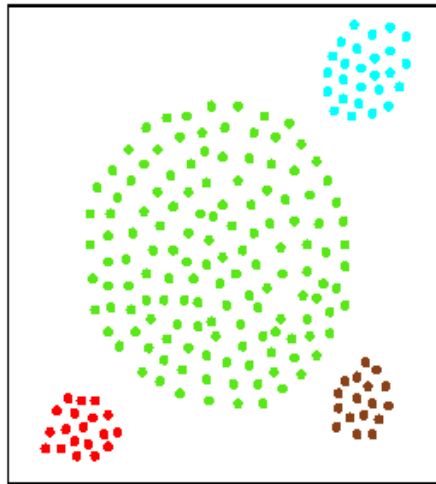
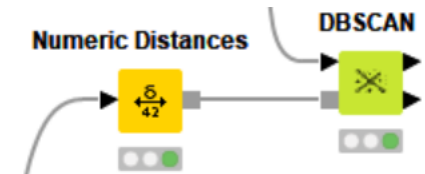
Property:

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

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


















DBSCAN [Density Based Spatial Clustering of Applications with Noise]

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

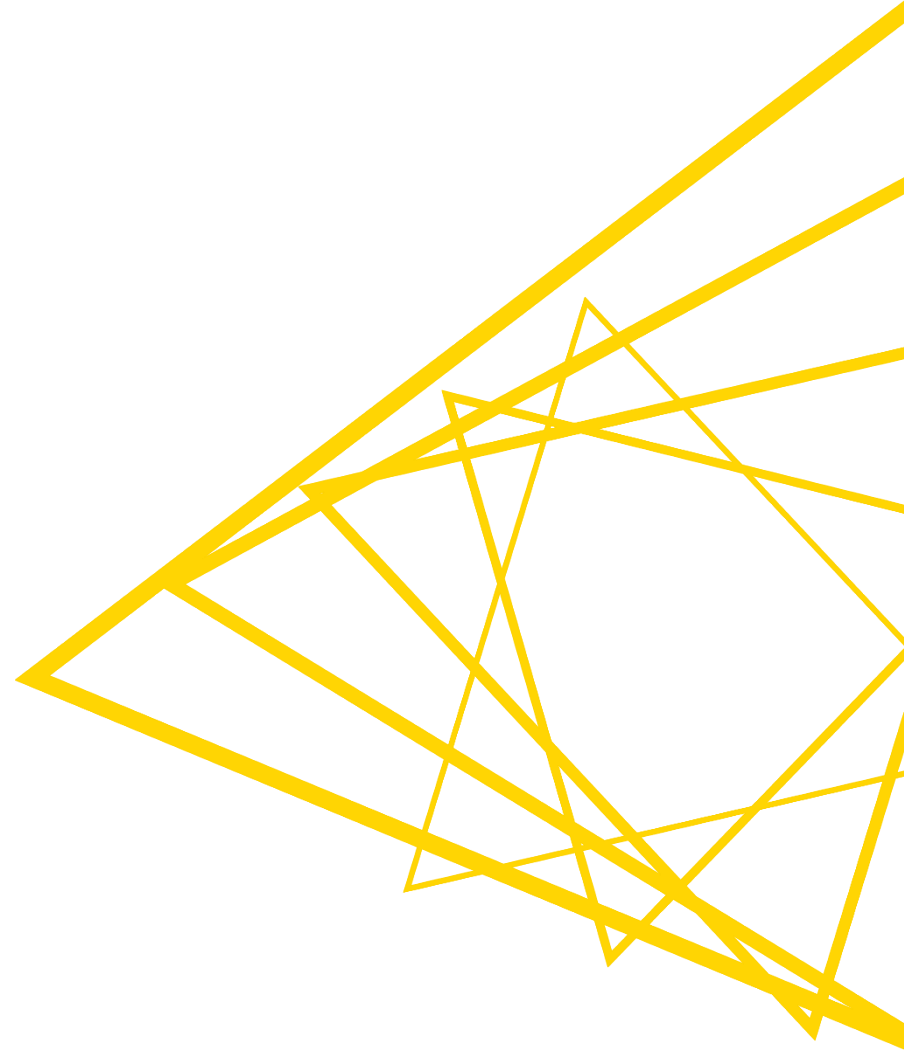


Exercises

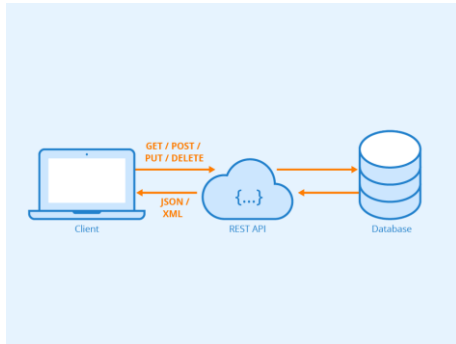
- Clustering
 - Goal: Cluster location data from California
 - 01_Clustering_exercise

- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - >  Session_2
 - >  Session_3
 - ▼  Session_4
 - ▼  01_Exercises
 - ▲  01_Clustering_exercise
 - ▲  02_Missing_Value_Handling_exercise
 - ▲  03_Outlier_Detection_exercise
 - ▲  04_Dimensionality_Reduction_exercise
 - ▲  05_Feature_Selection_exercise
 - ▼  02_Solutions
 - ▲  01_Clustering_solution
 - ▲  02_Missing_Value_Handling_solution
 - ▲  03_Outlier_Detection_solution
 - ▲  04_Dimensionality_Reduction_solution
 - ▲  05_Feature_Selection_solution

Deployment Options



Deploying the ML model



Inside its own application

REST service

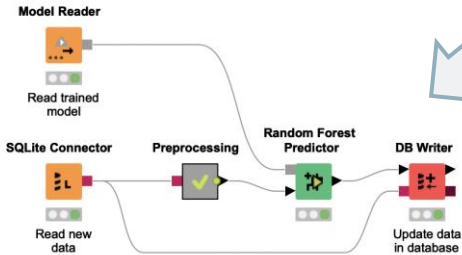
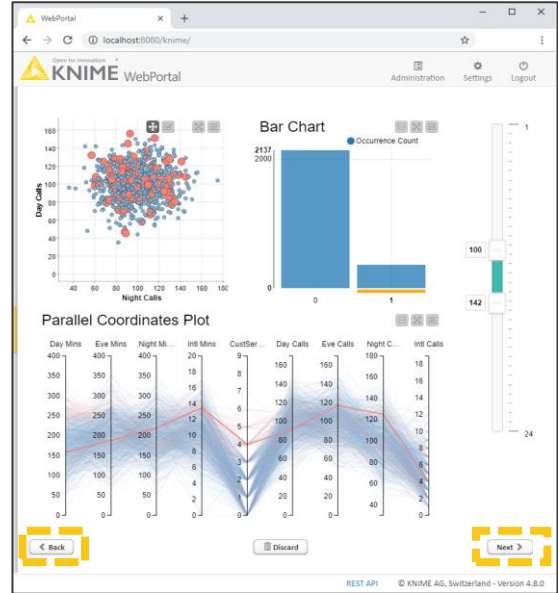
Data App



ML model

Scheduled Workflow

Component

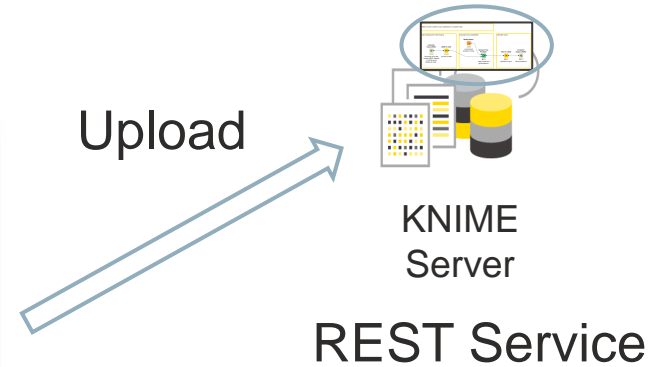
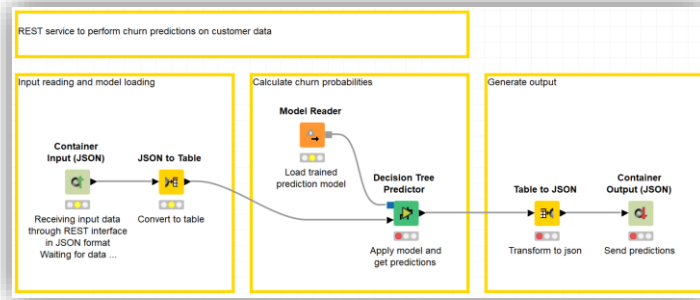


Component
ARIMA Predictor



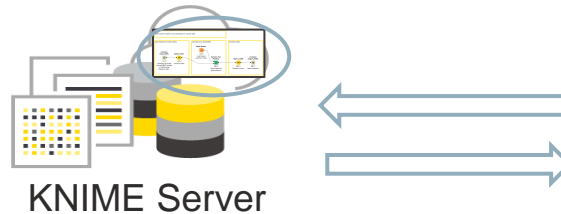
Building a web service

1



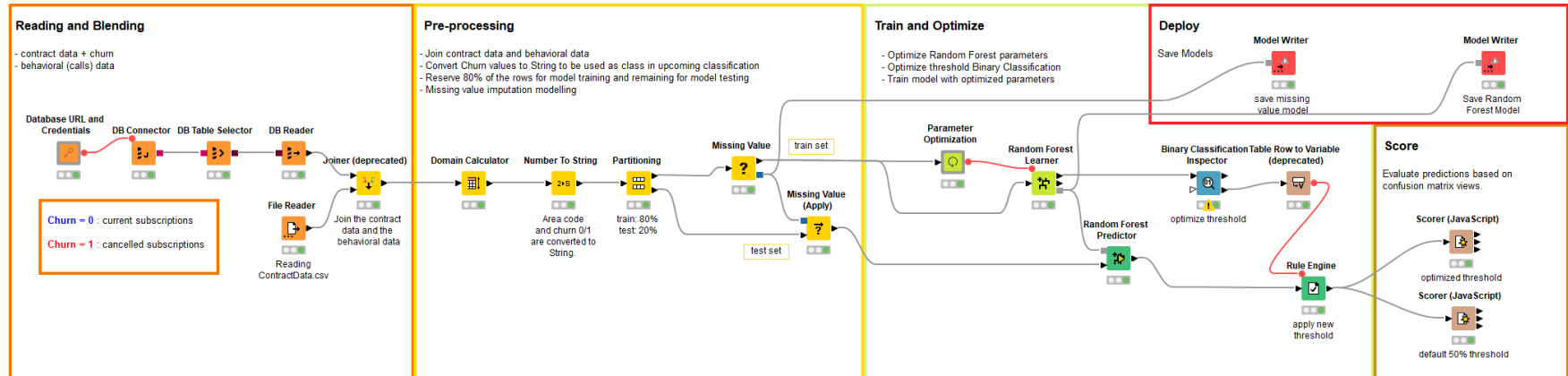
2

REST Service



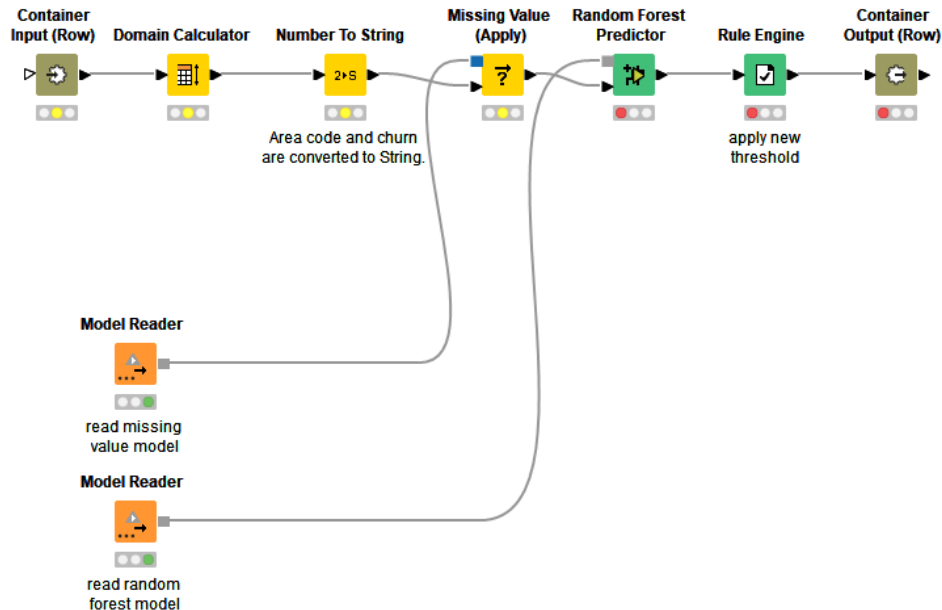
Classification using KNIME Analytics Platform

- The machine learning Classification task (pipeline) is broken into 4 pieces
 - Reading and Blending
 - Pre-processing
 - Train, Optimize and score
 - Deploy



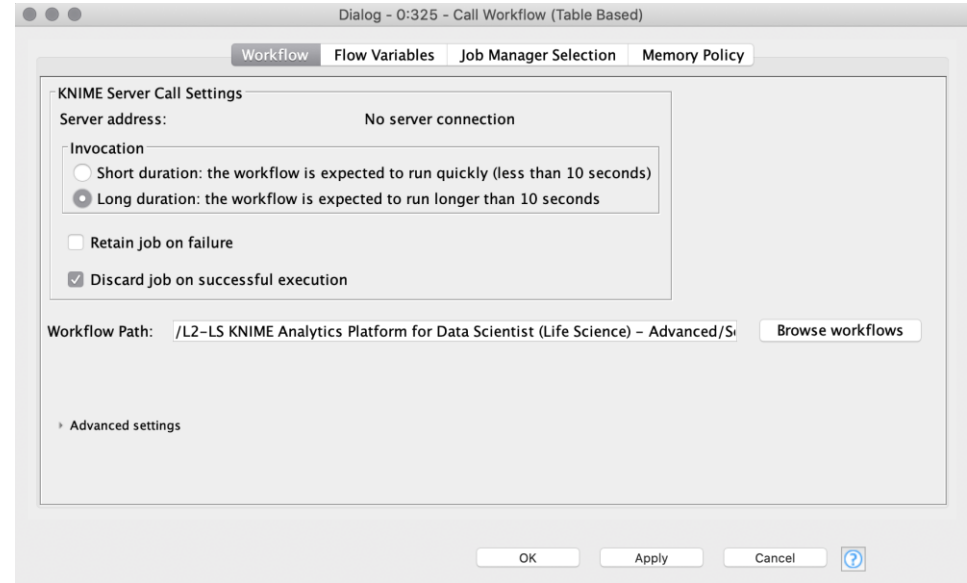
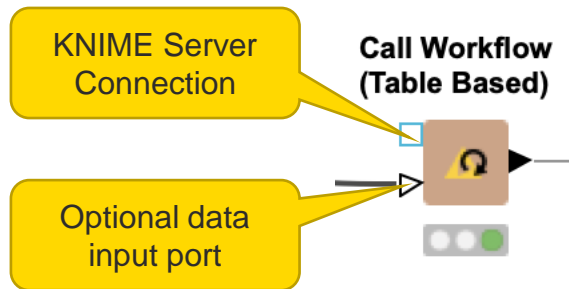
Deployment (Manual)

- The Deployment workflow must be able to take input data from external sources and generate predictions using the trained model



Call Workflow (Table Based)

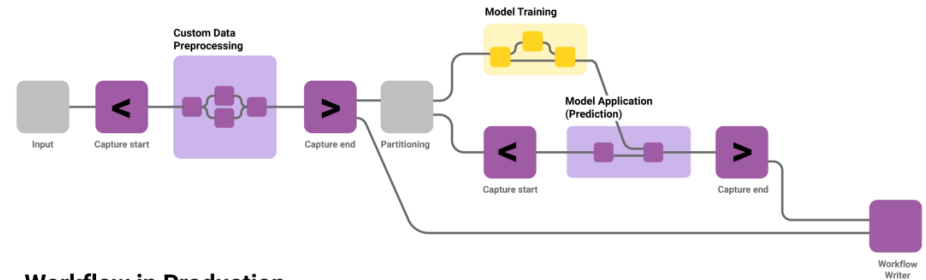
- Calls and executes another workflow
- Use the KNIME Server Connection node if the other workflow is located on KNIME Server



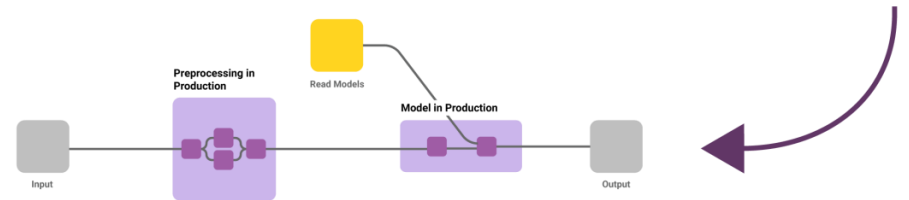
Concept of integrated deployment

- Use your design workflow to automate creation of your production workflow
- Lets you capture the parts of your workflow that are needed in a production environment, like:
 - Custom data preprocessing
 - Model application for prediction
 - Anything else you might need, so that you can...
- Automatic deployment to KNIME Business Hub

Creating Prediction Model



Workflow in Production



- ▼ KNIME Labs
 - ▼ Integrated Deployment
 - Capture Workflow Start
 - Capture Workflow End
 - Workflow Combiner
 - Workflow Writer
 - Deploy Workflow to Server
 - Workflow Executor

Integrated Deployment

- Closing the gap between Data Science Creation and Production

Benefits

Time savings



Save time and free up both data science and model operations resources.

Fewer errors



Reduce the error risk associated with moving from model creation to deploying production processes.

Increased compliance



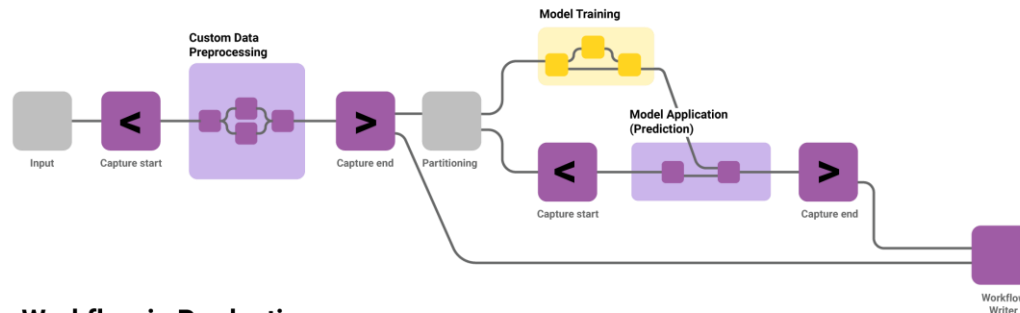
Capture and store both creation and production processes = full support of governance and compliance reporting.

Optimized processes

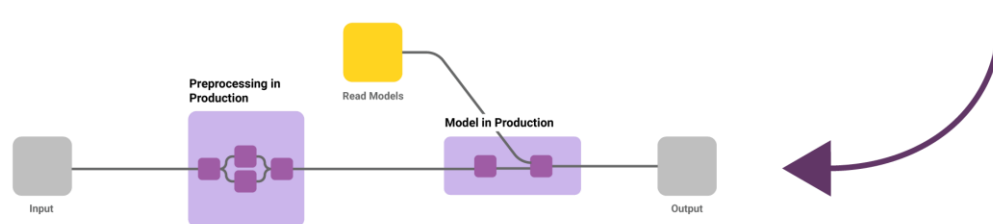


Optimize the updating of production processes automatically.

Creating Prediction Model

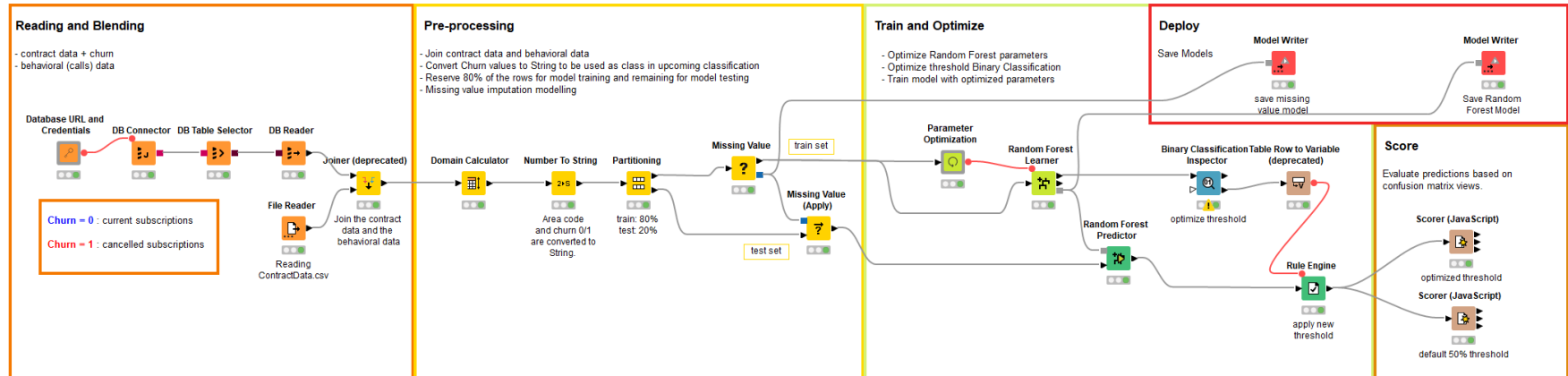


Workflow in Production



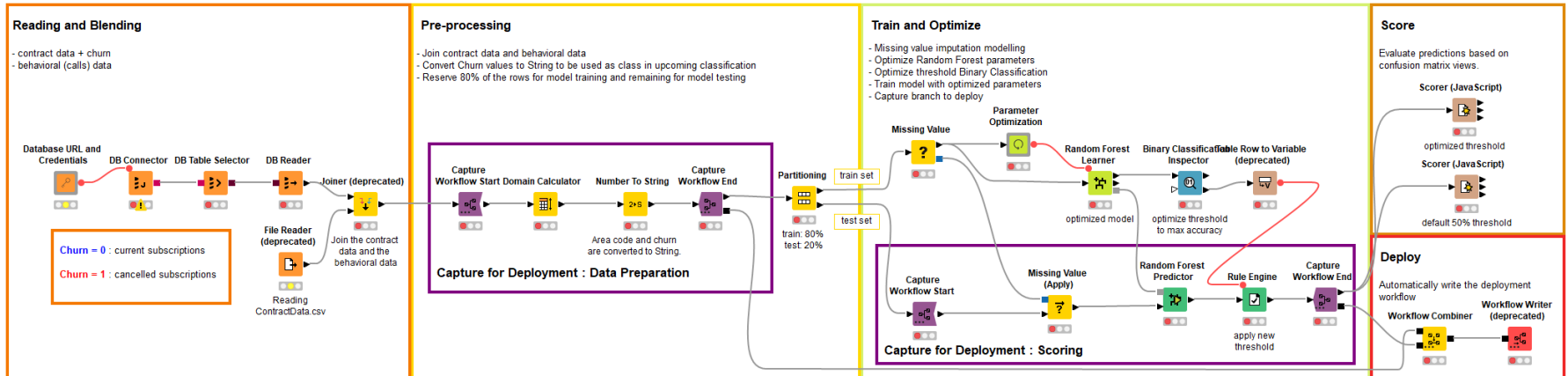
Classification using KNIME Analytics Platform

- The machine learning Classification task (pipeline) is broken into 4 pieces
 - Reading and Blending
 - Pre-processing
 - Train, Optimize and score
 - Deploy

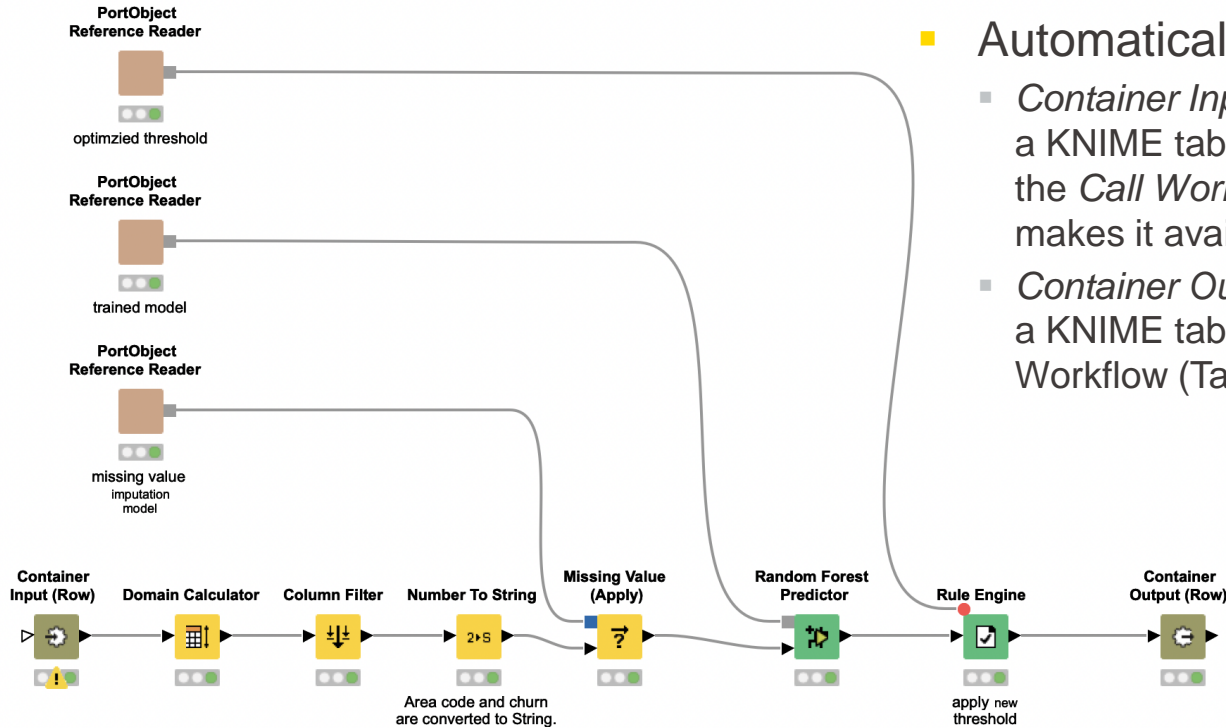


Integrated Deployment in KNIME

- The training workflow automatically writes the deployment workflow, so any changes done to the training workflow will be simultaneously pushed to deployment workflow



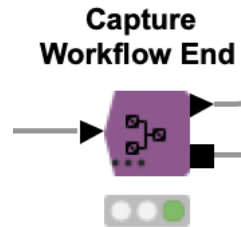
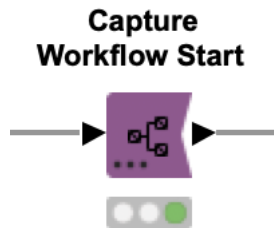
Example: Workflow in Production



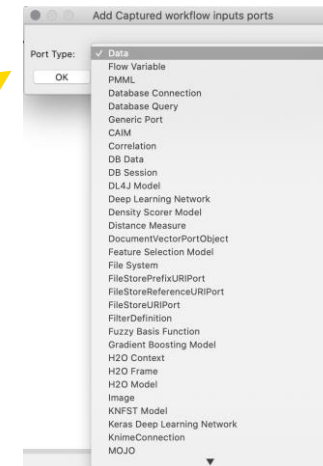
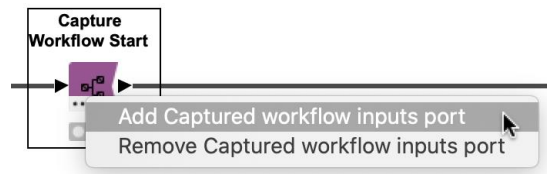
- Captured workflow parts are combined
- Automatically added nodes:
 - *Container Input (Table)* node: This node receives a KNIME table from an external caller (i.e. the *Call Workflow (Table Based)* node) and makes it available at the output port.
 - *Container Output (Table)* node: This node sends a KNIME table to an external caller (i.e. the *Call Workflow (Table Based)* node)

Capture Workflow Start/End

- Define the start and end point of the workflow you want to capture

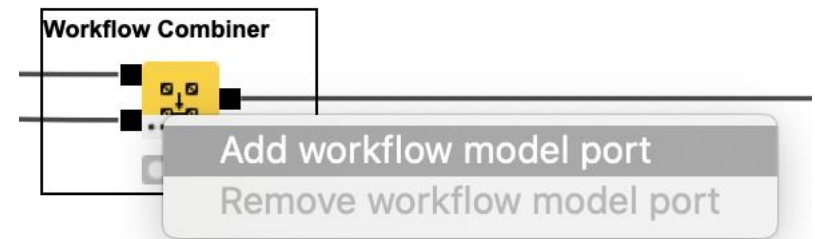
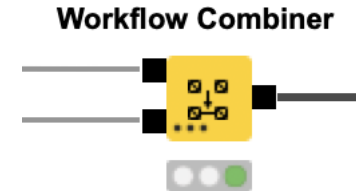


- Select the input and output ports



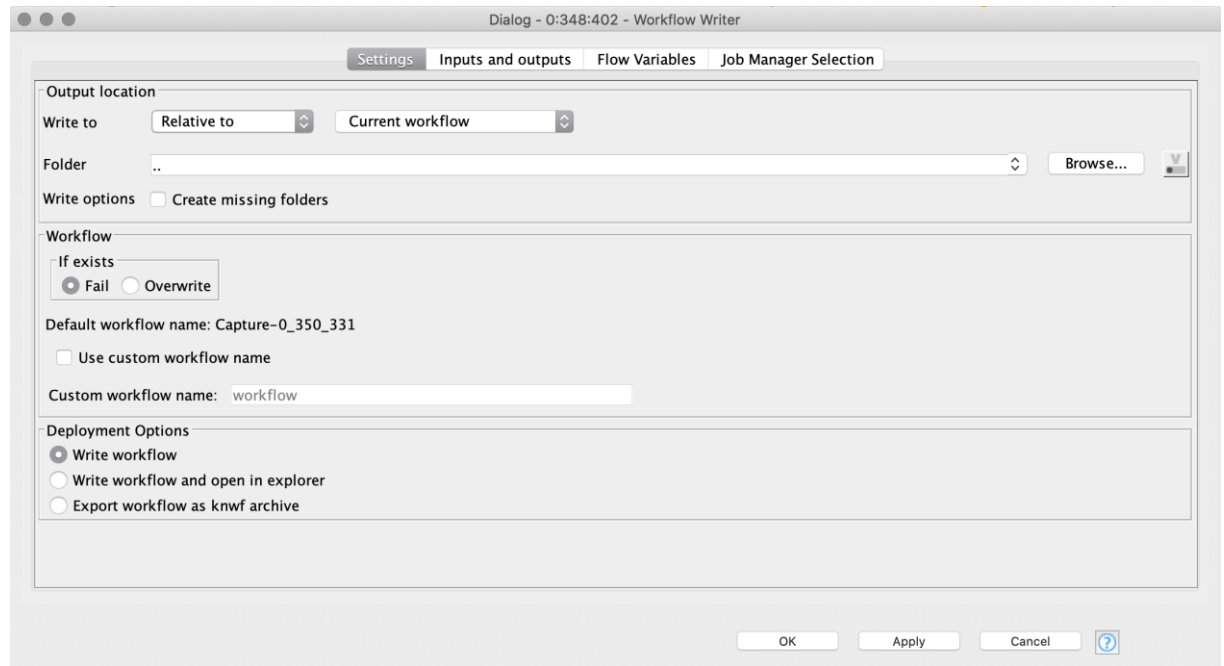
Workflow Combiner

- If you are capturing more than one part of your workflow
- Connect first workflow part to upper input port
- Connect second workflow part to lower input port
- Add further input ports to combine more workflow parts

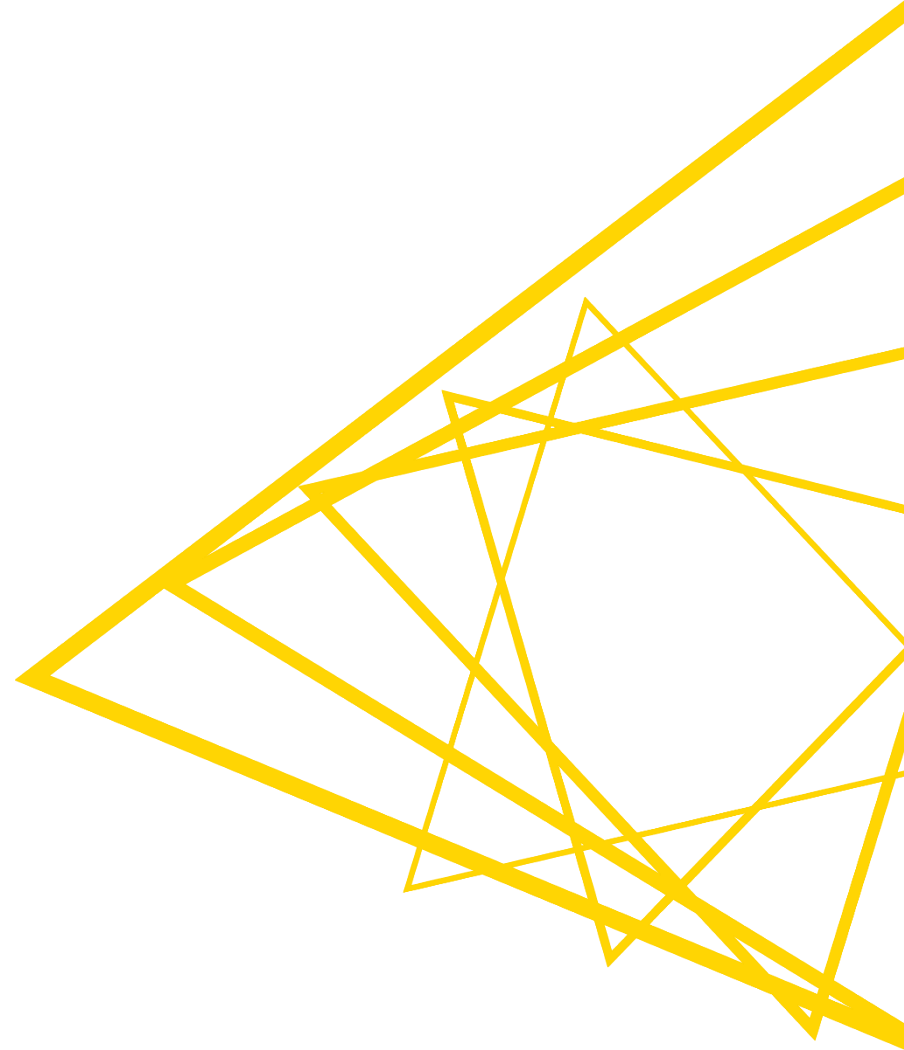


Workflow Writer

- Write captured workflow to selected output location



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 mg dominate over the length values for the similarity of records!

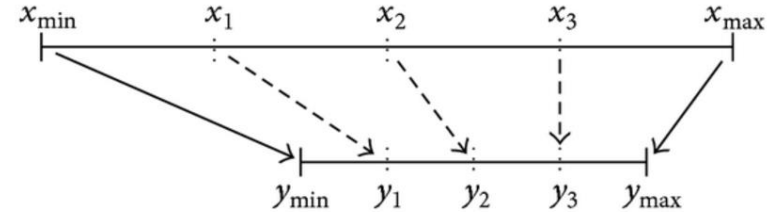
Goal of normalization:

- Transformation of attributes to make record ranges comparable

Normalization: Techniques

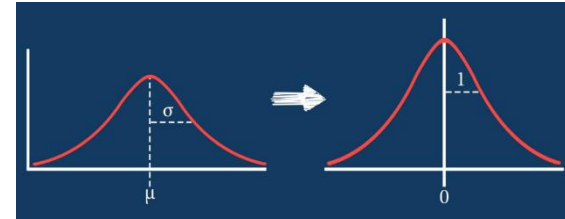
- min-max normalization

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



- z-score normalization

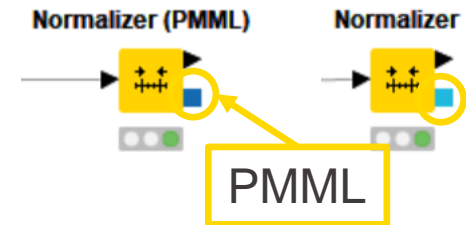
$$y = \frac{x - \text{mean}(x)}{\text{stddev}(x)}$$



- normalization by decimal scaling

$$y = \frac{x}{10^j}$$

where j is the smallest integer for $\max(y) < 1$
Here $[y_{min}, y_{max}]$ is $[-1, 1]$



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 (optional)

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.

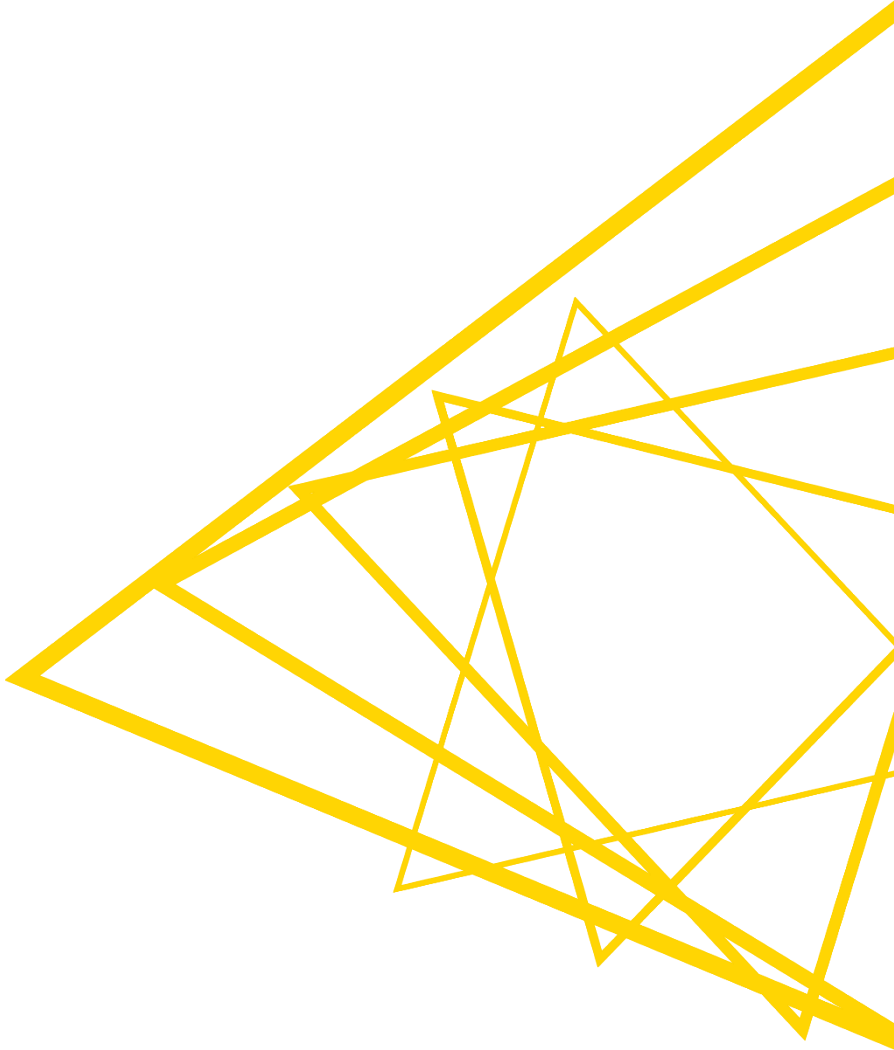
Missing Values Imputation

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
(Optional)**



Outlier Detection

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

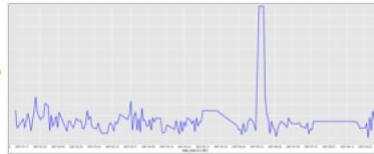
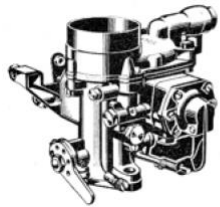
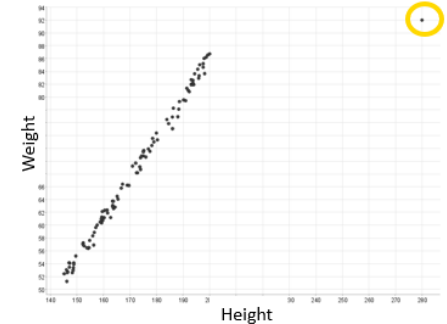


Image:
<https://www.nytimes.com/2016/06/25/business/international/brexit-financial-economic-impact-leave.html>

Brexit
referendum

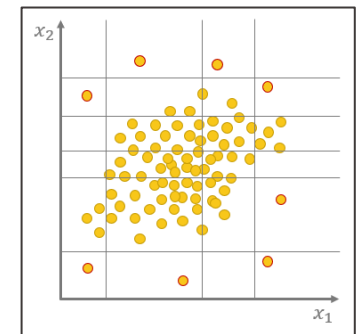
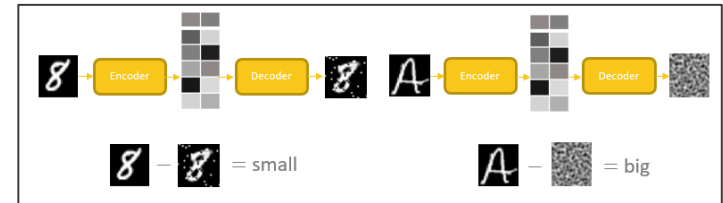
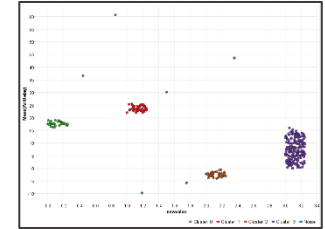
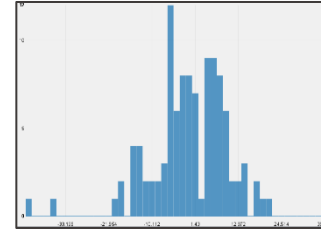
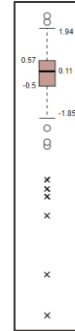


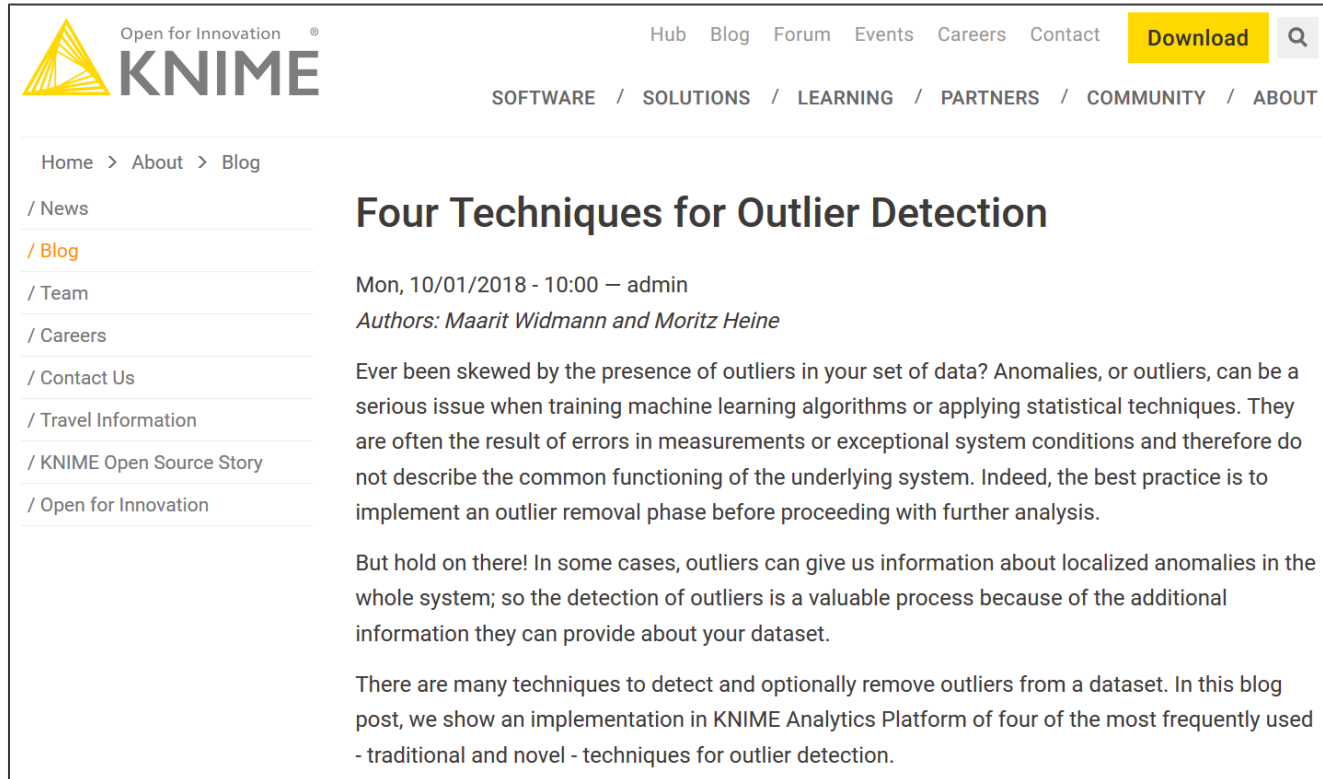
Outlier Detection: Motivation

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

Outlier Detection Techniques

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

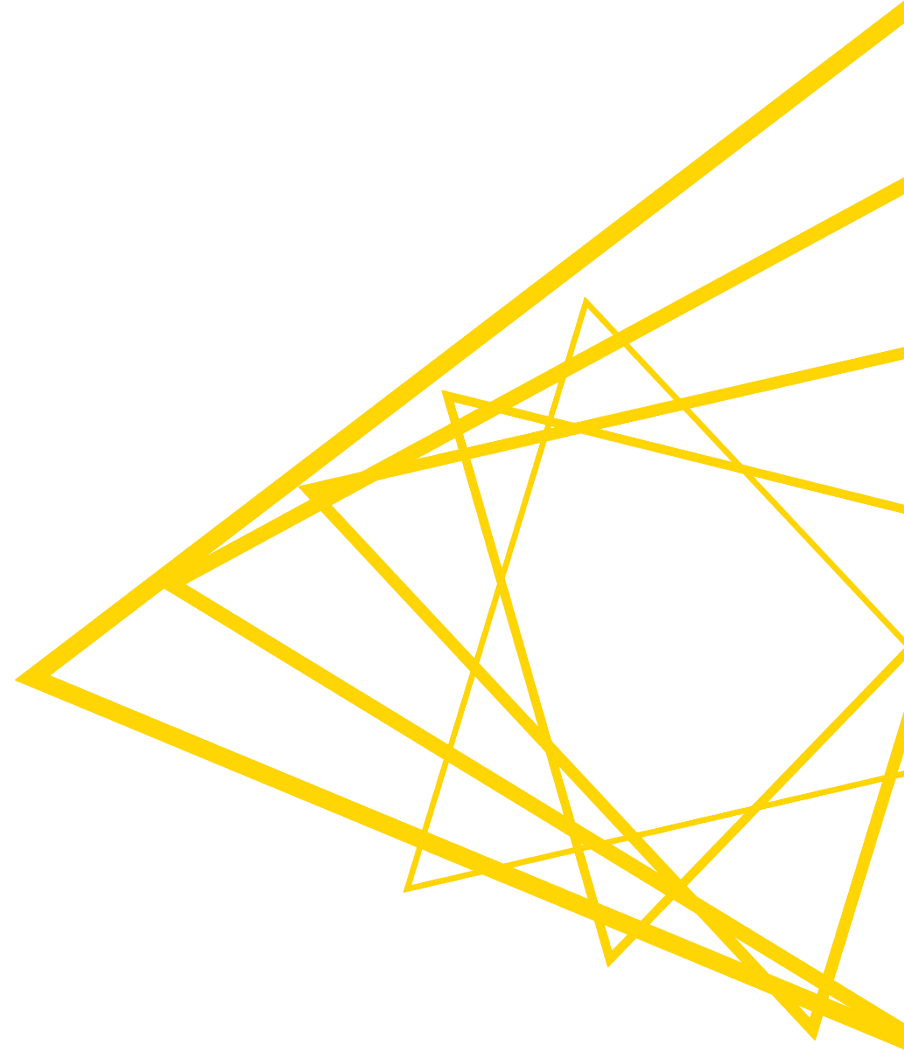




The screenshot shows the KNIME website's blog page. At the top left is the KNIME logo with the tagline 'Open for Innovation'. To the right are navigation links: Hub, Blog, Forum, Events, Careers, Contact, a yellow 'Download' button, and a search icon. Below this is a secondary navigation bar with links for SOFTWARE, SOLUTIONS, LEARNING, PARTNERS, COMMUNITY, and ABOUT. The main content area has a breadcrumb trail: Home > About > Blog. On the left is a sidebar menu with links for / News, / Blog (highlighted in orange), / Team, / Careers, / Contact Us, / Travel Information, / KNIME Open Source Story, and / Open for Innovation. The main article title is 'Four Techniques for Outlier Detection'. Below the title is the date 'Mon, 10/01/2018 - 10:00' and the author 'admin'. The authors are listed as 'Maarit Widmann and Moritz Heine'. The article text begins with '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.' The next paragraph states '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.' The final paragraph says '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 Hilite Navigation View

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

Row ID	Var16	Var17	Var18	Var19	Var20	Var21	Var22	Var23	Var24	Var25	Var26	Var27	Var28
Row0	?	?	?	?	?	464	580	?	14	128	?	?	166.56
Row1	?	?	?	?	?	168	210	?	2	24	?	?	353.52
Row2	?	?	?	?	?	1212	1515	?	26	816	?	?	220.08
Row4	?	?	?	?	?	64	80	?	4	64	?	?	200
Row7	?	?	?	?	?	32	40	?	2	16	?	?	230.56
Row8	?	?	?	?	?	200	250	?	2	64	?	?	300.32
Row10	?	?	?	?	?	92	115	?	6	112	?	?	133.12
Row11	?	?	?	?	?	236	295	?	8	40	?	?	133.12
Row12	?	?	?	?	?	0	0	?	?	0	?	?	240.56
Row13	?	?	?	?	?	480	600	?	10	216	?	?	176.56
Row14	?	?	?	?	?	148	185	?	0	8	?	?	236.08
Row16	?	?	?	?	?	584	730	?	6	320	?	?	220.08
Row17	?	?	?	?	?	168	210	?	2	32	?	?	166.56
Row18	?	?	?	?	?	12	15	?	2	0	?	?	253.52
Row20	?	?	?	?	?	168	210	?	2	56	?	?	272.08
Row21	?	?	?	?	?	20	25	?	2	0	?	?	86.96
Row22	?	?	?	?	?	192	240	?	2	80	?	?	166.56
Row23	?	?	?	?	?	52	65	?	0	56	?	?	198.88
Row24	?	?	?	?	?	216	270	?	8	128	?	?	200
Row25	?	?	?	?	?	152	190	?	4	16	?	?	20.08
Row26	0	0	0	?	?	?	?	?	?	?	?	?	?
Row28	?	?	?	?	?	0	0	?	?	0	?	?	257.28
Row29	?	?	?	?	?	312	390	?	0	120	?	?	200
Row30	?	?	?	?	?	112	140	?	4	56	?	?	166.56
Row31	?	?	?	?	?	28	35	?	0	16	?	?	288.2
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	Var21	Var22	Var23	Var24	Var25	Var26	Var27	Var28
Row51	336	420	0	8	72	0	0	133.12	
Row52	120	150	0	0	16	0	0	286.96	
Row54	124	155	0	0	0	0	0	234.72	
Row55	184	230	0	4	64	0	0	642.64	
Row56	268	335	0	4	88	0	0	133.12	
Row57	128	160	0	0	96	0	0	198.88	
Row59	132	165	0	0	112	0	0	253.52	
Row60	44	55	0	0	24	0	0	186.64	
Row61	104	130	0	4	72	0	0	166.56	
Row62	212	265	0	6	136	0	0	379.6	
Row63	20	25	0	0	0	0	0	166.56	
Row65	492	615	0	18	256	0	0	133.12	
Row66	148	185	0	2	8	0	0	186.64	
Row68	140	175	0	2	40	0	0	176.56	
Row69	0	0	0	0	0	0	0	166.56	
Row71	0	0	0	0	0	0	0	392.08	
Row72	124	155	0	6	88	0	0	153.2	
Row73	152	190	0	0	32	0	0	253.52	
Row74	324	405	0	8	104	0	0	186.64	
Row75	0	0	0	0	0	0	0	0	
Row76	60	75	0	6	0	0	0	200	
Row77	180	225	0	4	88	0	0	166.56	
Row78	232	290	0	4	144	0	0	200	
Row79	16	20	0	0	16	0	0	313.68	
Row81	152	190	0	0	48	0	0	220.08	
Row82	108	135	0	4	88	0	0	166.56	

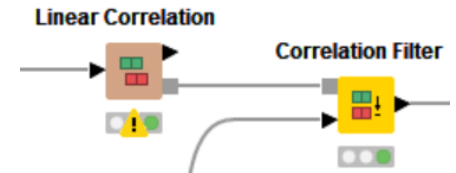


Note: requires min-max-normalization, and only works for numeric columns

- 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) (optional)

- 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>

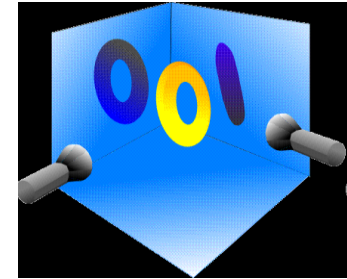
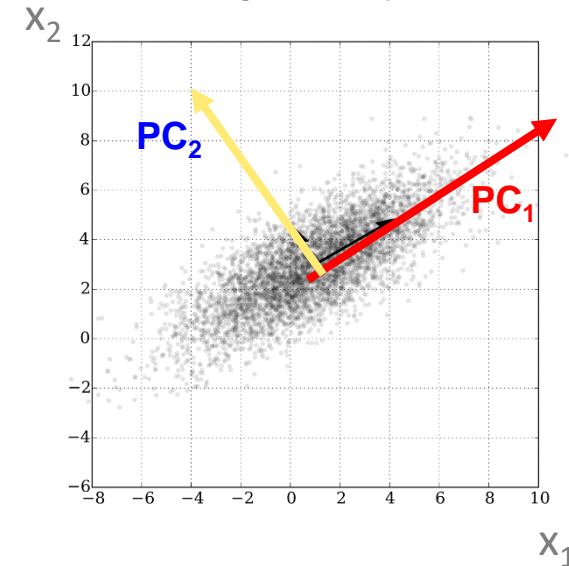


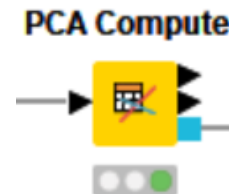
Image from Wikipedia



Principal Component Analysis (PCA)

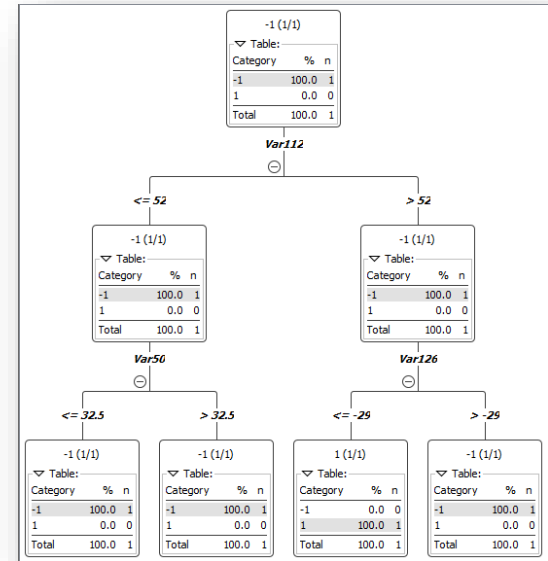
- PC_1 describes most of the variability in the data, PC_2 adds the next big contribution, and so on. In the end, the last PCs do not bring much more information to describe the data.
- Thus, to describe the data we could use only the top $m < n$ (i.e., PC_1, PC_2, \dots, PC_m) components with little - if any - loss of information
- Caveats:
 - Results of PCA are quite difficult to interpret
 - Normalization required
 - Only effective on numeric columns

Dimensionality Reduction

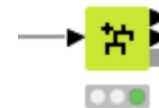


Ensembles of Shallow Decision Trees

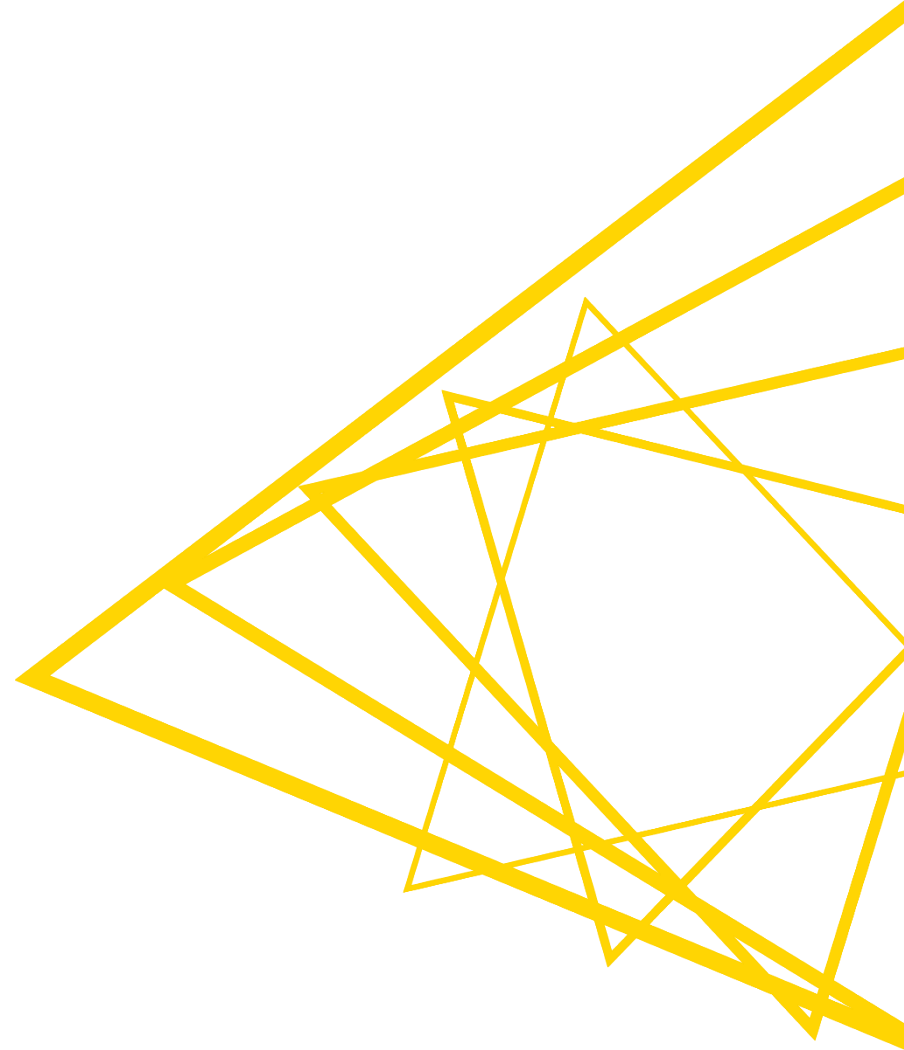
- Often used for classification, but can be used for feature selection too
- Generate a large number (we used 2000) of trees that are very shallow (2 levels, 3 sampled features)
- Calculate the statistics of candidates and selected features. The more often a feature is selected in such trees, the more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



Tree Ensemble
Learner



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	D Var6
0.97	16	D Var7
0.97	61	D Var13
0.968	17	D Var21
0.968	8	D Var22
0.965	10	D Var24
0.965	12	D Var25
0.965	10	D Var28
0.965	10	D Var35
0.965	9	D Var38
0.965	5	D Var44
0.965	59	D Var57
0.963	35	D Var65
0.963	24	D Var73
0.96	44	D Var74
0.96	40	D Var76
0.96	37	D Var78
0.96	32	D Var78
0.96	32	D Var81
0.96	23	D Var83
0.96	14	D Var85
0.96	3	D Var109
0.96	50	D Var112
0.958	62	D Var113
0.958	53	D Var119
0.958	34	D Var123
0.958	30	D Var125
0.958	28	D Var126
0.958	26	D Var132
0.958	22	D Var133
0.958	21	D Var133

OK Apply Cancel ?

node by site

Feature Selection Loop End

Maximize accuracy

Feature Selection Filter

Do the final filtering here

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

MACHINE LEARNING / CONTRIBUTED

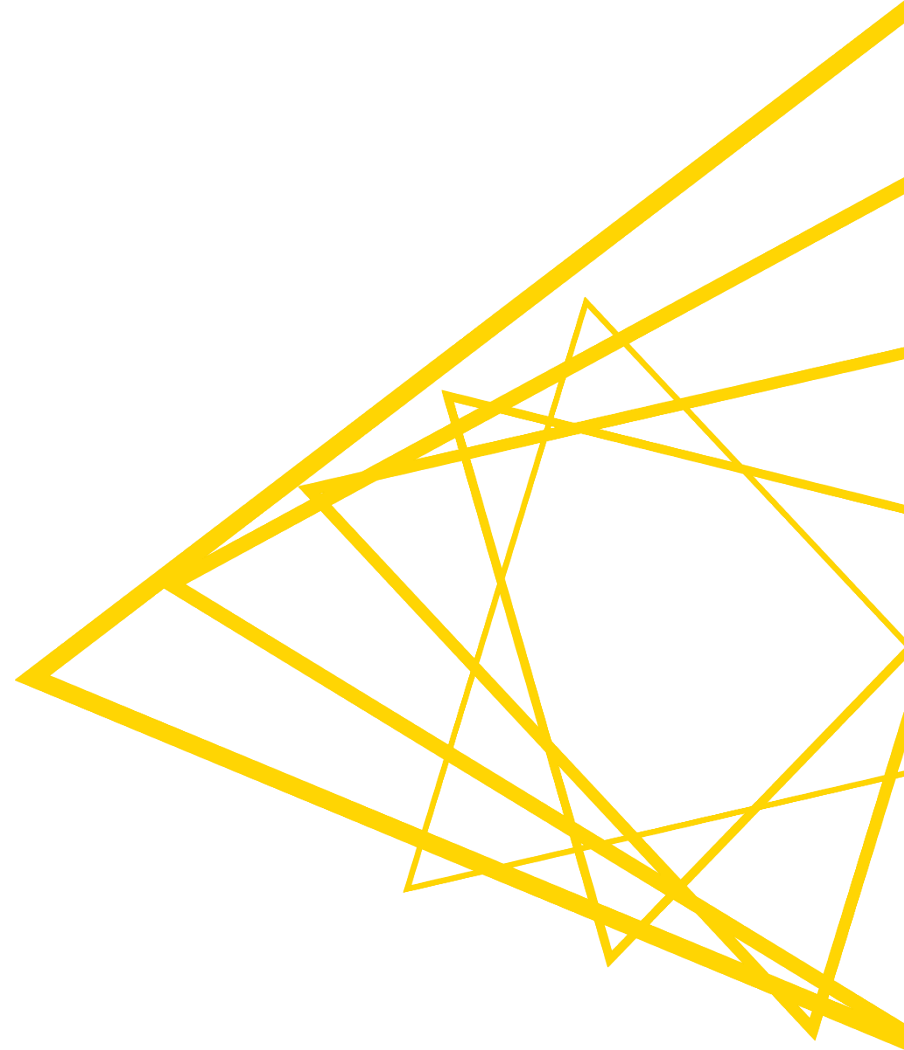
3 New Techniques for Data-Dimensionality Reduction in Machine Learning

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



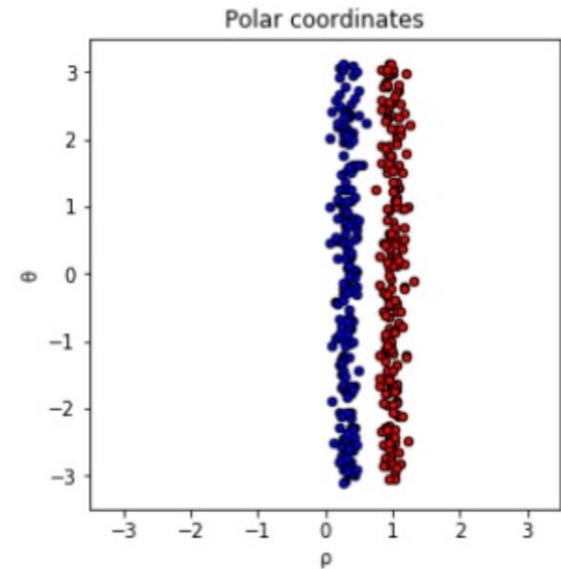
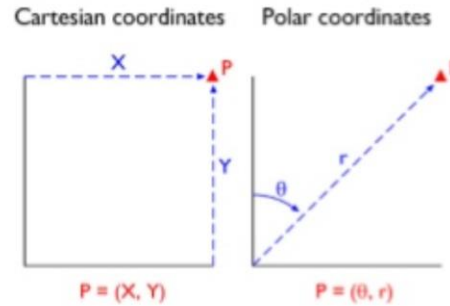
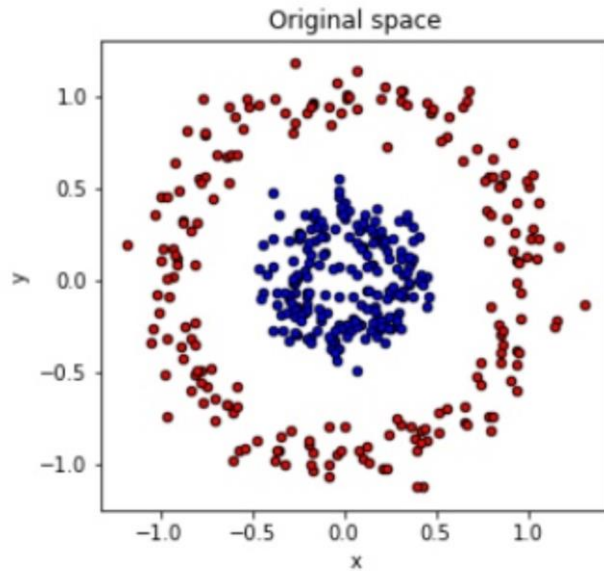
<https://thenewstack.io/3-new-techniques-for-data-dimensionality-reduction-in-machine-learning/>

Data Preparation: Feature Engineering



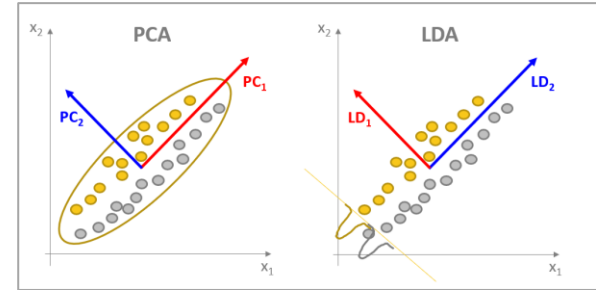
Feature Engineering: Motivation

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 , $\tanh(x)$, $\log(x)$, ...)
- Combining together multiple columns in math functions
($f(x_1, x_2, \dots, x_n)$, $x_2 - x_1$, ...)
- The whole process is domain dependent



Exercises (optional)

- Clustering
 - Goal: Cluster location data from California
 - 01_Clustering_exercise
- Data Preparation
 - 02_Missing_Value_Handling_exercise
 - 03_Outlier_Detection_exercise
 - 04_Dimensionality_Reduction_exercise
 - 05_Feature_Selection_exercise

- ▼  L4-ML Introduction to Machine Learning Algorithms
 - >  Session_1
 - >  Session_2
 - >  Session_3
 - ▼  Session_4
 - ▼  01_Exercises
 - ▲  01_Clustering_exercise
 - ▲  02_Missing_Value_Handling_exercise
 - ▲  03_Outlier_Detection_exercise
 - ▲  04_Dimensionality_Reduction_exercise
 - ▲  05_Feature_Selection_exercise
 - ▼  02_Solutions
 - ▲  01_Clustering_solution
 - ▲  02_Missing_Value_Handling_solution
 - ▲  03_Outlier_Detection_solution
 - ▲  04_Dimensionality_Reduction_solution
 - ▲  05_Feature_Selection_solution

Machine Learning Cheat Sheet

Cheat Sheet: Machine Learning with KNIME Analytics Platform



SUPERVISED LEARNING

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

CLASSIFICATION

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

Decision Tree: Follows the C4.5 decision tree algorithm. These algorithms generate a tree-like structure, ordering data subsets, aka tree nodes. At each node, the data are split based on the value of the input feature, ordering two or more branches as "yes" or "no" or some other value. The leaf nodes of the tree represent the class with the highest conditional probability assigned to the input data.

Naïve Bayes: Based on Bayes' theorem and assuming statistical independence between input features (this "naïve" is why it's called that). This algorithm estimates the conditional probability of each output class, given the vector of input features.

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

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

NUMERIC PREDICTION & CLASSIFICATION

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

Deep Learning: Deep learning extends the family of ANNs with deeper architectures and recurrent neural networks (RNN). The training of such networks has been enabled by recent advances in hardware performance as well as parallel execution.

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

NUMERIC PREDICTION

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

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

Auto-Regressive Integrated Moving Average (ARIMA): A linear Auto-Regressive (AR) model is constructed from a identified number of past vectors, while a linear combination - named Moving Average (MA) - models the *q* past residual errors. An ARIMA model parameters are estimated concurrently by various algorithms, mostly following the Box-Jenkins approach.

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

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

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

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

UNSUPERVISED LEARNING

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

CLUSTERING

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

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

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

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

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

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

RECOMMENDATION ENGINES

Association Rules: The model needs implications to occurrences of multiple products in large-scale transaction data. Based on the apriori algorithm, recommendation engines identify items that are frequently purchased together. The association rules are used to generate recommendation lists.

Collaborative Filtering: Based on the Alternating Least Squares (ALS) algorithm, it generates product recommendations by learning about the interests of a user by comparing them with those of multiple users (collaborator).

ENSEMBLE LEARNING

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

BAGGING

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

The Ensemble of Decision/Regression Trees: Ensemble model of multiple decision/regression trees trained on different subsets of data. Data subsets with less or equal trees and less or equal splits are considered as weak models. The original training set. Final prediction is based on a hard vote (majority class) or soft vote (averaging all probabilities or numeric predictions on all involved trees).

Random Forest of Decision/Regression Trees: Ensemble model of multiple decision/regression trees trained on different subsets of data. Data subsets with the same number of trees are bootstrapped from the original training set. At each node, the split is performed as if the data were randomly split into two halves with (approximately) equal size. The average of probabilities or numeric predictions on all involved trees.

BOOSTING

Boosting: A method for training a set of models sequentially through the use of machine learning. At each step, a new model is trained on the previous model error, and added to the ensemble to improve the results from the previous model state, leading to a higher accuracy after each iteration.

Gradient Boosted Regression Trees: Ensemble model consisting multiple sequential simple regression trees into a stronger model. The algorithm builds the model iteratively. A weak classifier, a simple regression tree, is fitted to predict the residuals of the current model, following the gradient of the loss function. This leads to an increased accuracy and complexity of the overall model. The same regression trees can also be used for classification.

CUSTOM ENSEMBLE MODEL

Combines different supervised models to form a custom prediction can be based on majority vote as well as on the average or other functions of the output results.

MiniBatch: An optimized distributed library for machine learning models in the gradient boosting framework, designed to be highly efficient, flexible, and scalable. It handles decision boundaries, effective handling of sparse data for better performance, parallel computation, and more efficient memory usage.

DEPLOYMENT

Model Loader: A component that loads a trained model from a file or database into the KNIME environment for use in a workflow.

Data Input: A component that provides data to a model for prediction or scoring.

Apply: A component that applies a model to new data to generate predictions or scores.

Predictor: A component that uses a model to generate predictions or scores based on new input data.

Data Output: A component that saves the results of a model's predictions or scores to a file or database.

Resources:

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

EVALUATION

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

Confusion Matrix: A representation of a classification model's performance. The actual and predicted classes, as well as true positives, true negatives, false positives, and false negatives. One class is arbitrarily selected as the positive class.

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

Cross-Validation: A model validation technique for assessing how the results of a machine learning model will generalize to an independent dataset. A model is trained and validated in cross on different pairs of data set and test sets, drawn from the original data set. Some test and basic statistics on the resulting error or accuracy measure gives insights on overfitting and generalization.

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

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

https://www.knime.com/sites/default/files/2021-07/CheatSheet_ML_A3.pdf

Confirmation of Attendance and Survey

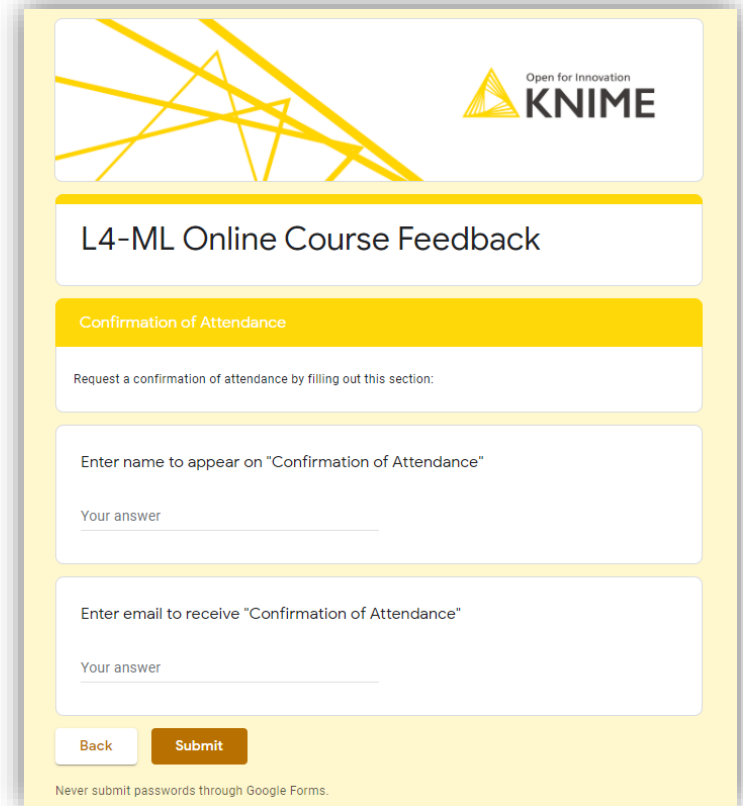
- If you would like to get a “Confirmation of Attendance” please click on the link below*

[Confirmation of Attendance and Survey](#)

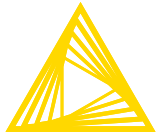
- The link also takes you to our course feedback survey. Filling it in is optional but highly appreciated!

Thank you!

*Please send your request within the next 3 days



The screenshot shows a Google Forms interface for a survey titled "L4-ML Online Course Feedback". At the top right, the KNIME logo is displayed with the tagline "Open for Innovation". The form has a yellow header bar with the title "L4-ML Online Course Feedback". Below this, a yellow bar contains the section title "Confirmation of Attendance". The main content area is white and contains the instruction "Request a confirmation of attendance by filling out this section:". There are two text input fields: the first is labeled "Enter name to appear on 'Confirmation of Attendance'" and the second is labeled "Enter email to receive 'Confirmation of Attendance'". Both fields have "Your answer" placeholder text. At the bottom of the form, there are two buttons: "Back" and "Submit". A small note at the very bottom states "Never submit passwords through Google Forms."



Open for Innovation

KNIME

Thank You!

