Part II-3: Sentiment Analysis
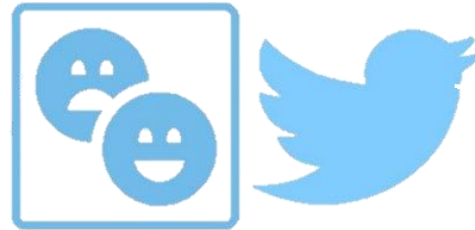
Outline

i.   Word Embeddings

ii.   Feature Collection

iii. Sentiment Analysis

    i.   Machine Learning Background

    ii.   Training & Prediction

    iii. Performance Evaluation

    iv.   Visualization

iv.   Excursus: AutoML Pipeline

Part II-3: Sentiment Analysis

Word Embeddings

Word Embeddings Setting

- **Recall our goal:** numeric representation of texts by variables that co-occur across documents

- **Approaches**
  - Vocabulary-based → *done*
  - Neural network representations → *later: BERT*
  - Word embeddings → *let's see*

Word Embeddings **Vocabulary-Based**

- **Revisited: BOW**
  - Vocabulary with all occurring words in documents
  - Assumption: each word independent from others present in document
  - No accounting for word order
  - Each document represented by **term frequency vector** (occurrence of all distinct words present in document)
- **Idea**: weighting
- **Term Frequency - Inverse Document Frequency (TF-IDF)**
  - Not implying that all terms are considered equally important
  - **Idea**: penalize words that are too frequent

Word Embeddings **Example BOW**

Documents

"Die Ausgrenzung von MigrantInnen ist inakzeptabel und rassistisch."

"Die Maskenpflicht ist sinnvoll."

"Die Diskriminierung von Frauen ist inakzeptabel."

Vector-space representations

	die	ausgrenzung	von	migrantinnen	ist	inakzeptabel	und	rassistisch	maskenpflicht	sinnvoll	diskriminierung	frauen
Doc1	1	1	1	1	1	1	1	1	0	0	0	0
Doc2	1	0	0	0	1	0	0	0	1	1	0	0
Doc3	1	0	1	0	1	1	0	0	0	0	1	1

Word Embeddings Example TF-IDF

Documents

"Die Ausgrenzung von MigrantInnen ist inakzeptabel und rassistisch."

"Die Maskenpflicht ist sinnvoll."

"Die Diskriminierung von Frauen ist inakzeptabel."

Vector-space representations

	die	ausgrenzung	von	migrantinnen	ist	inakzeptabel	und	rassistisch	maskenpflicht	sinnvoll	diskriminierung	frauen
Doc1	0	.48	.18	.48	0	.18	.48	.48	0	0	0	0
Doc2	0	0	0	0	0	0	0	0	.48	.48	0	0
Doc3	0	0	.18	0	0	.18	0	0	0	0	.48	.48

Word Embeddings **Idea**

- Word embeddings *aka* word vectors *aka* word representations
- **Idea:** model semantic importance of words in numeric form

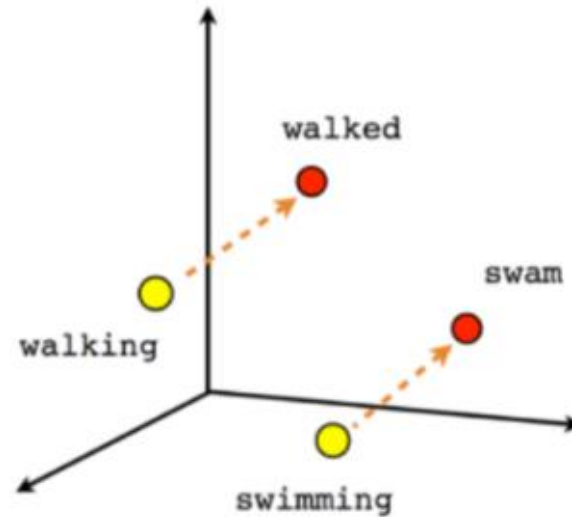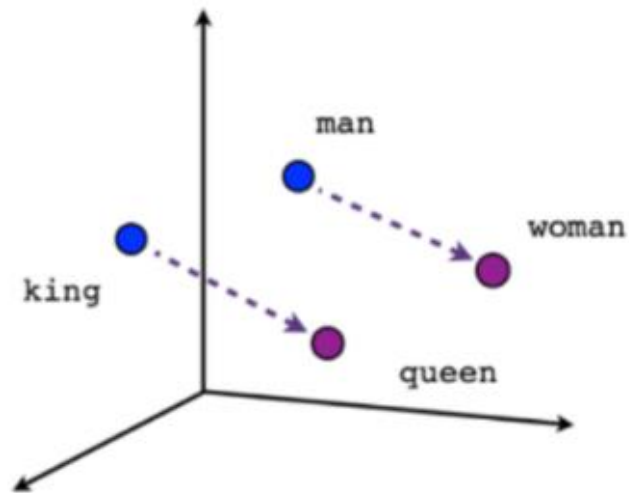 *we wish to embed words into the continuous space of real numbers*

- Unsupervised learning task
- Also achieved by BOW/TF-IDF, but: **high dimensionality**
- **Goal:** dense representation

Word Embeddings **Idea**

- **Dimensionality reduction**

- **Embeddings** / factor loadings
  - Characterize words by their surrounding context
  - Latent dimensions by which words can be represented
  - Similar meaning = similar representation in the vector space

	💡 masculinity?	💡 royalty?
word	embedding 1	embedding 2
king	0.87	0.73
queen	-0.12	0.75
woman	0.03	-0.08

Word Embeddings Example

https://towardsdatascience.com/the-magic-behind-embedding-models-c3af62f71fb

Enabling mathematical operations on the vocabulary:
- $\phi: W \rightarrow R^n$
- $\phi("king") - \phi("man") + \phi("woman") = \phi("queen")$

Word Embeddings Approaches

- **Approaches**: various possibilities, often adopted from general dimensionality reduction
  - Unifying idea: data observed in (extremely) high-dimensional space but truly much lower-dimensional $\rightarrow$ retrieve principal dimensions
  - **GloVe**
  - Word2vec
  - fastText
  - t-distributed stochastic neighbor embedding (t-SNE)
  - …

Word Embeddings GloVe

- **GloVe**: **Glo**bal **Ve**ctors

- Developed by Stanford University (2014)

- Based on word co-occurrence matrix
  - Studying neighborhood relations between words
  - Defined via window size (symmetric/asymmetric)
  - Underlying assumption: close-lying words are more strongly linked
  - Entry in i-th row & j-th column: how likely is word i to appear in the context of word j?

The quick brown fox jumps over the lazy dog.

Word Embeddings GloVe

- **Computation**
  - R: package `text2vec`
  - Most important hyperparameters: number of embedding dimensions & skip-gram window size

- Alternatively: pre-trained embeddings

- Here: **topic-specific** embeddings
  - Subset corpus by topic labels
  - Compute embeddings for subsets
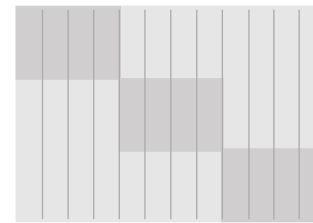
words have different meanings in different contexts

Word Embeddings **Embeddings vs BOW**

- Both result in vector representations for each word in a corpus.
  - **BOW**
    - ⊕ Easy to understand and implement
    - ⊕ Feasible for any corpus
    - ⊖ No accounting for order, semantics
    - ⊖ High-dimensional representations
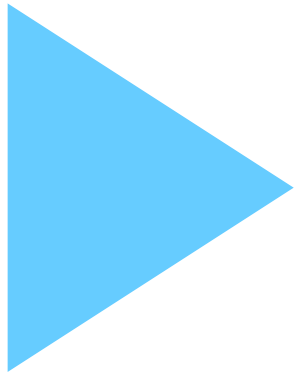 - **Embeddings**
    - ⊕ Capturing semantics and heeding word order
    - ⊕ Low-dimensional representations
    - ⊖ Large and „high-quality" corpus required for meaningful embeddings
    - ⊖ Pre-trained models often computationally demanding and not applicable to tasks with different domain (zero vector for unknown words)

both suffer from the cold-start problem

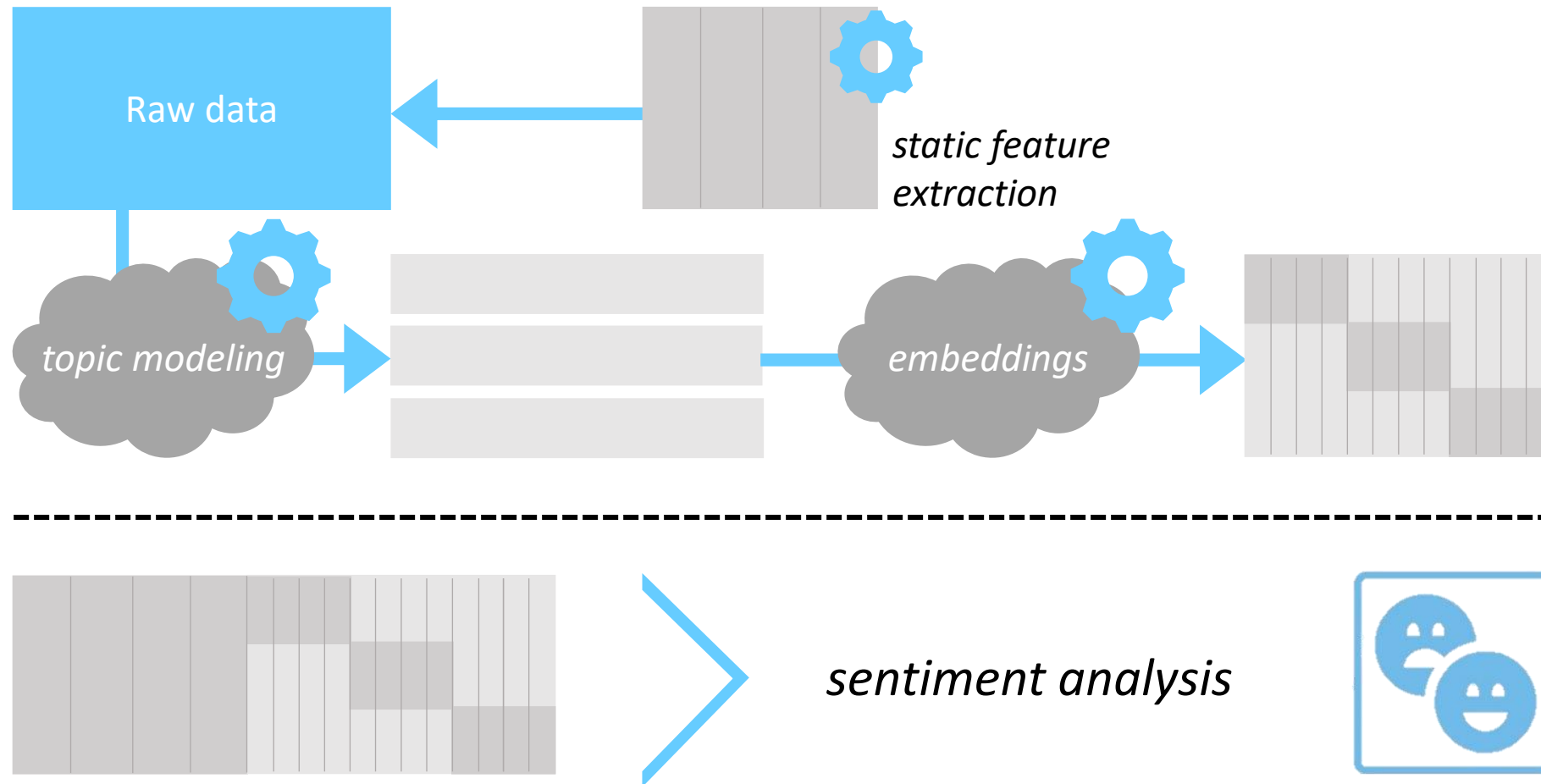# Word Embeddings Example

Demo 8: Word Embeddings

Part II-3: Sentiment Analysis

Feature Collection

Feature Collection **Recall: Task Structure**

Raw data

static feature extraction

topic modeling
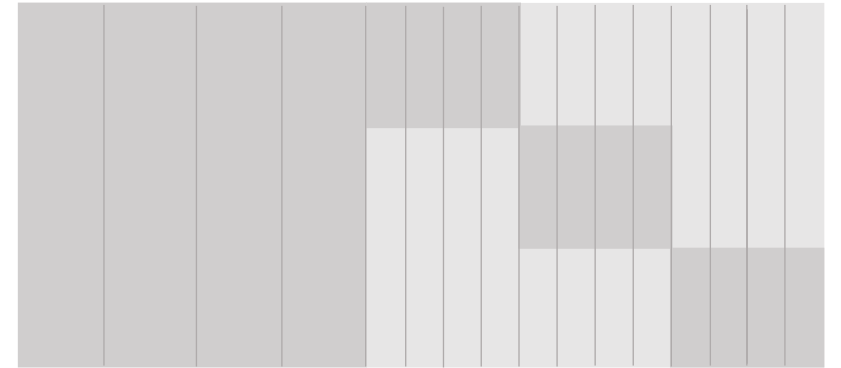
embeddings

sentiment analysis

Feature Collection Recall: Task Structure

- **Static features**
  - Polarity clues
  - Negations, intensifications, punctuations, repetitions
  - Word/character n-grams
  - Part-of-speech (POS) tags
  - Twitter-specific features

- **Dynamic features**
  - Word embeddings per topic

Part II-3: Sentiment Analysis

Sentiment Analysis

ML Background Overview

- **Machine Learning (ML)**
  - **Supervised learning**
  - Unsupervised learning
  - Reinforcement learning
  - Deep learning methods for all three

- Supervised learning
  - Learn feature-target relationship from labeled data
  - **Classification**: predict class label from data features
 - **Regression**: predict continuous response from data features
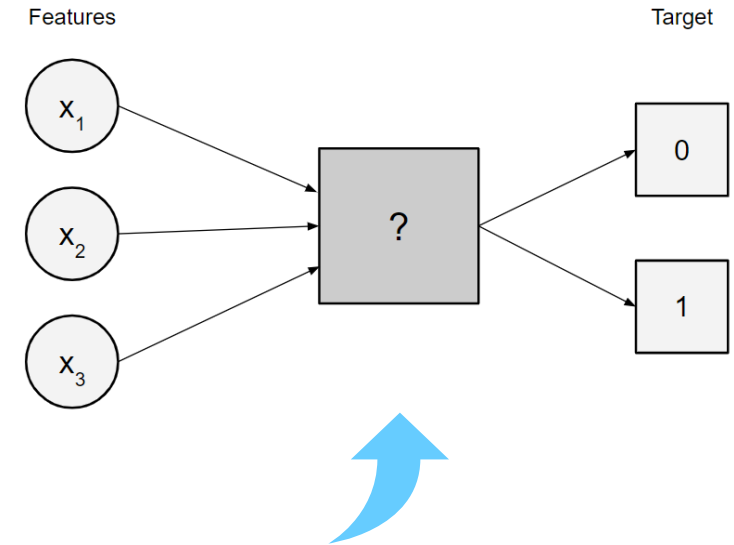
ML Background mlr3 Package

- **mlr3**
  - Very extensive, all-purpose ML package developed and maintained by LMU's Statistical Learning & Data Science chair
  - **Unifying framework** for many ML functionalities
  - End-to-end programming from feature generation to prediction

- Useful sources
  - Introduction to Machine Learning lecture
    https://introduction-to-machine-learning.netlify.app/
  - mlr3 book *https://mlr3book.mlr-org.com/*

ML Background Components

- **ML components**
  - Task
    - Train set
    - Test set
  - Learner
    - Hypothesis space
    - Risk
    - Optimization
  - Performance measure

https://mlr3book.mlr-org.com/
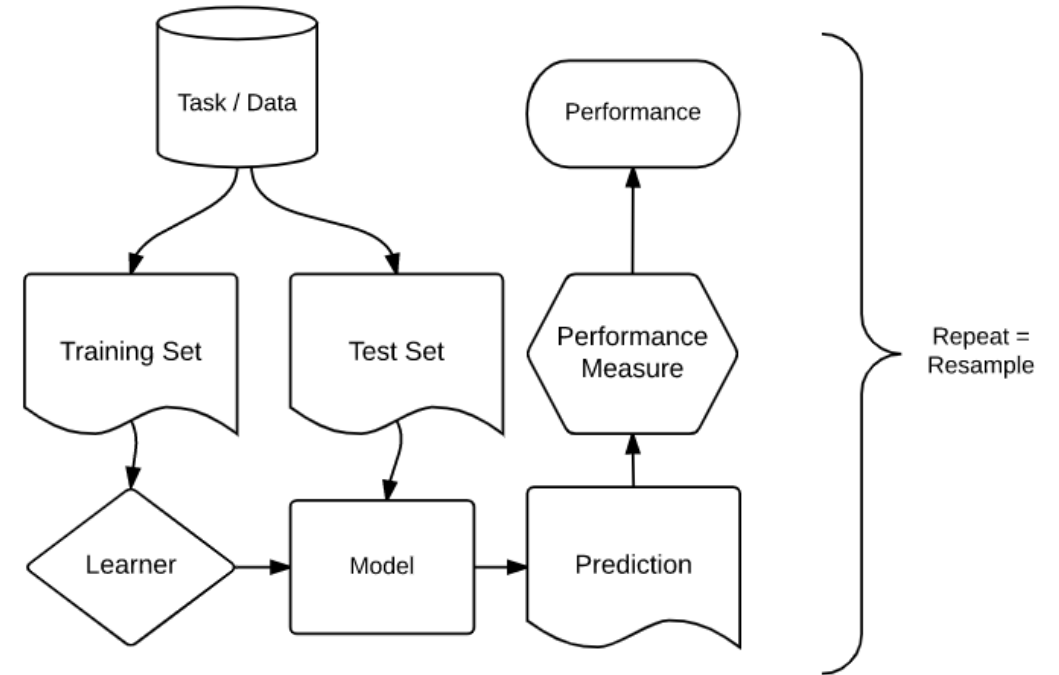
Training & Prediction Classification Tasks

- Components of a (classification) task
  - **Features X**: all (numeric) variables describing our observations
  - **Target *y***: class label, here $\in$ {positive, negative}

- Train-test split
  - Fundamental ML principle: **dichotomy** between training and test sphere $\rightarrow$ Avoid bias in performance estimation
  - Train on training data, evaluate on test data
  - Possibly create repeated splits (**resampling**)

Training & Prediction **Learners**

- Components of a learner
  - **Hypothesis space**: defines what kind of model can be learned, e.g.,
    - Logistic regression model
    - Decision tree
    - Random forest
  - **Risk**: quantifies by how much our predictions deviate from the true target
    $\rightarrow$ To be minimized
  - **Optimization**: defines how to search for the best model
- **Empirical risk minimization (ERM)**
- **Result**: model with trained parameters

Performance Evaluation **Idea**

- How well does our model **perform** on unseen data?
  $\rightarrow$ **Generalization** ability

  typically, test error > training error

- Measured on test set(s)

- *Aka* **outer loss** $\leftrightarrow$ inner loss used for training the model via ERM
  - We might or might not use the same metric for both.
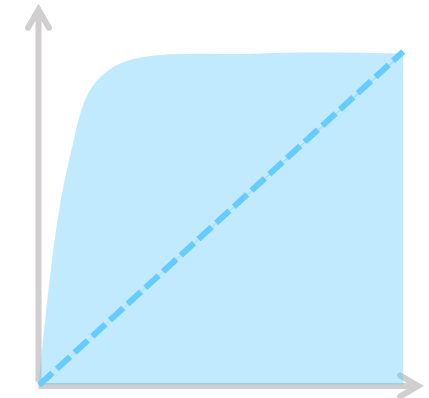 - Various evaluation metrics exist.

Performance Evaluation **Idea**
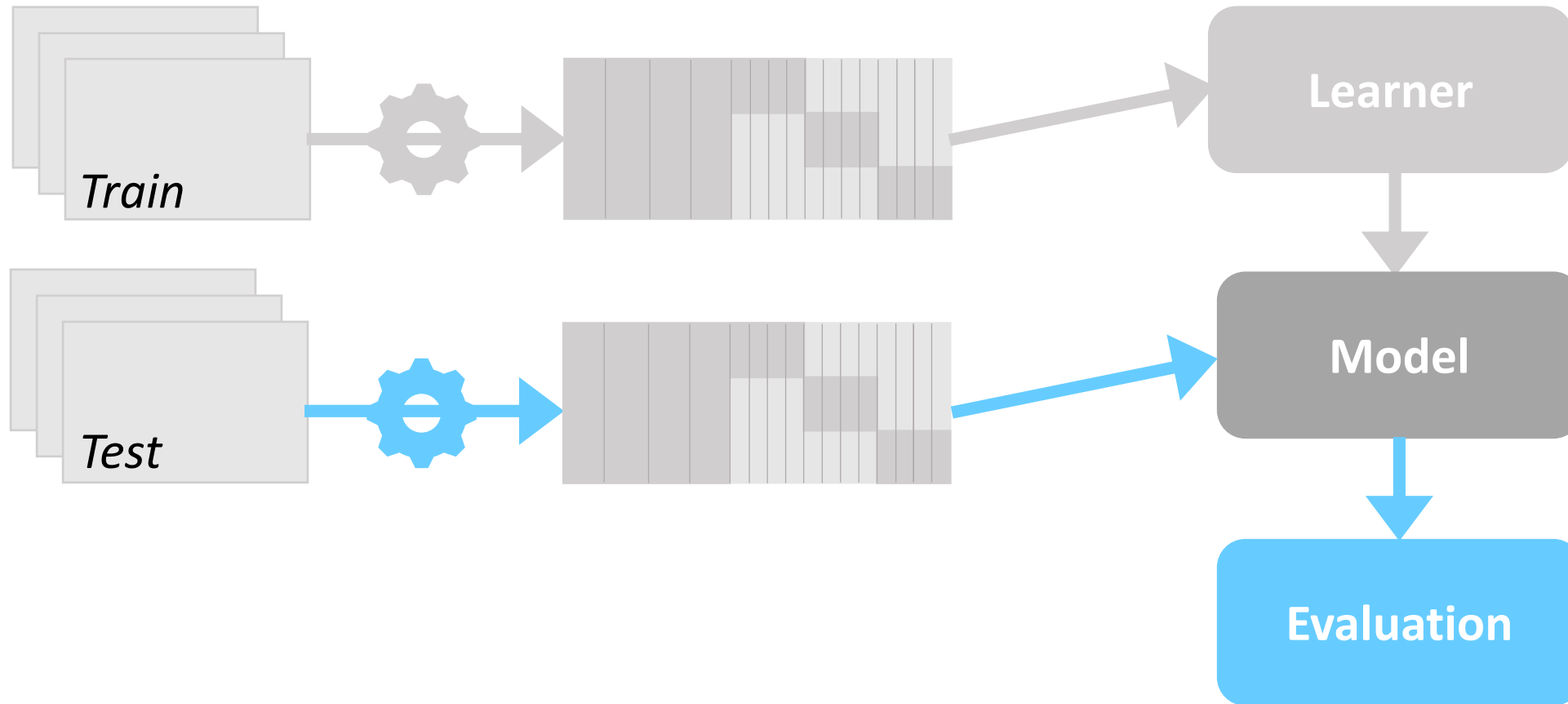
Performance Evaluation **Metrics**

- Many different ones, reflecting what kind of error we wish to keep small
- Special metrics for binary classification: **ROC**-based

	Actual: YES	**Actual: NO**
Predicted: YES	True Positive (TP)	False Positive (FP)
Predicted: NO	False Negative (FN)	True Negative (TN)

Performance Evaluation **Metrics**

|	Total population	True condition			
		Condition positive	**Condition negative**	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive,** Power	**False positive,** Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative,** Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$	$F_1 \text{ score} = \frac{1}{\frac{1}{Recall} + \frac{1}{Precision}}{2}$

https://en.wikipedia.org/wiki/F-score#Diagnostic_testing
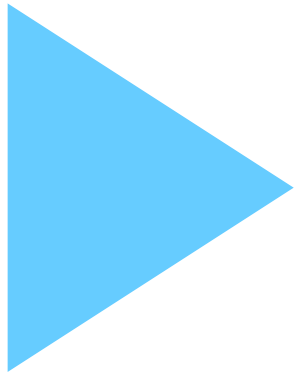
Training & Prediction **Example**
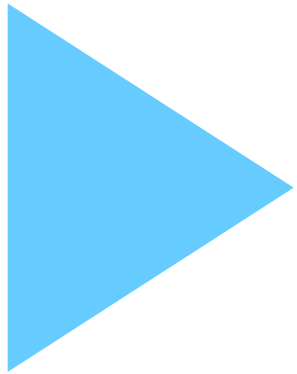
Demo 9: Sentiment Analysis

ML Pipeline Exercise
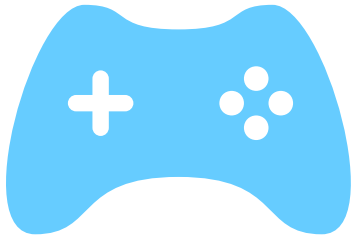
Exercise 5: Sentiment Analysis

# Visualization Plotting Results

Demo 10: Visualizing Results

Part II-3: Sentiment Analysis

Excursus: AutoML Pipeline

AutoML Pipeline Motivation

- **Recall**: static vs dynamic features

- Frequently, we are in situations where
  - we tune/evaluate multiple learners, and/or
  - feature generation affects surrounding observations.

- These steps typically call for repeated train-test splits (**resampling**, nested resampling).

- For predictions to remain unbiased this requires training & evaluation of the entire (AutoML) **pipeline**.
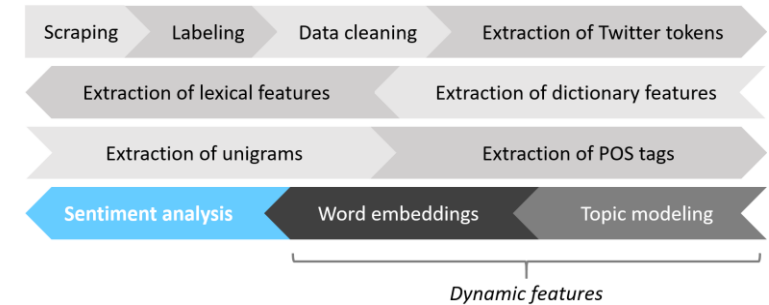
AutoML Pipeline Graph Learners

- Building pipelines via **graph learners**
  - Encompassing all steps from pre-processing to evaluation
  - Modular building approach
  - All methods (training, prediction, tuning, …) applicable as usual
- Automatization of entire procedure possible to large extent
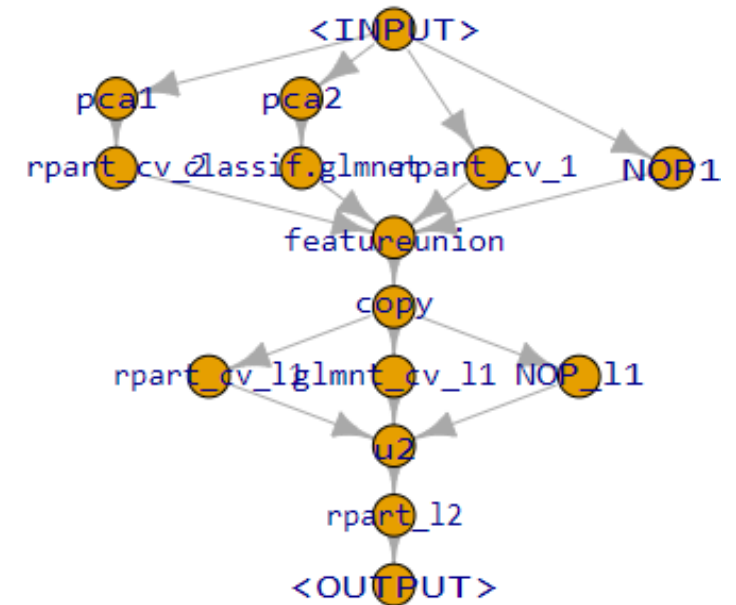
tutorial on creating AutoML systems on *https://mlr3gallery.mlr-org.com/*

Part II-3: Sentiment Analysis

Literature and References

Bishop, C. (2006): Pattern Recognition and Machine Learning, Springer.

Hastie, T., Tibshirani, R, and Friedman, J. (2017): The Elements of Statistical Learning. Data Mining, Inference, and Prediction, 2nd ed., Springer.

Japkowicz, N., and Shah, M. (2011): Evaluationg Learning Algorithms. A Classification Perspective, Cambridge University Press.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013): Efficient Estimation of Word Representations in Vector Space, arXiv:1301.3781.

Pennington, J., Socher, R., and Manning, C. (2014): GloVe: Global Vectors for Word Representation

https://introduction-to-machine-learning.netlify.app/

https://mlr3book.mlr-org.com/