Come2Data

TRAINING: Python für die Datenanalyse in den Sozialwissenschaften

Teil 3: Grundlagen Maschinelles Lernen mit Python

Machine Learning Pipelines

GEFÖRDERT VOM
Bundesministerium für Forschung, Technologie und Raumfahrt

SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

SPEAKER: Matthias Täschner

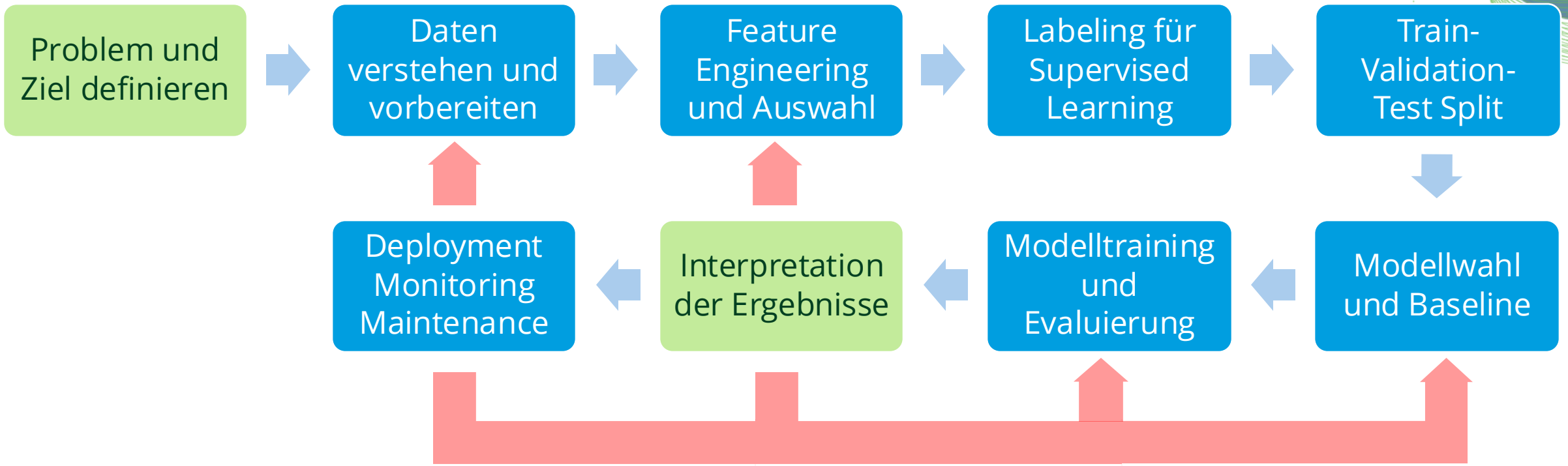
Mit Verwendung von Materialien von Robert Haase (ScaDS.AI / Universität Leipzig)

AGENDA

- Machine Learning Pipelines - Überblick
- Einzelne Schritte
  - Daten verstehen und vorbereiten
  - Feature Engineering und Auswahl
  - Train-Validation-Test Split
  - Modellwahl
  - Modelltraining und Evaluierung
  - Deployment und Monitoring
- Praktische Übung: Aufbau einer ML-Pipeline

Überblick einer ML-Pipeline

„Modell auswählen, Daten rein, Vorhersagen raus …fertig, oder???“



“80% Daten, 20% Modell” – Gute Vorbereitung ist wichtiger als ausgefallene Algorithmen
“Evaluiere nicht damit, womit du trainierst” – Datentrennung vermeidet falsche Erfolgsmeldungen
“Deployment ≠ Fertig!” – Modelle altern, Daten ändern sich

Schritte in einer ML-Pipeline
Daten verstehen und vorbereiten

Datenquellen und Qualität
- Welche Quellen mit welchen Datenformaten gibt es, wie wird darauf zugegriffen
- Welche Qualität haben die Daten (Fehlwerte, Einheitenmix, Dubletten, Zeiträume, …)

Explorative Datenanalyse
- Verteilungen, Korrelationen
- Datentypen, Skalierungen
- Ausreißer

Bereinigung
- Umgang mit Fehlwerten und Ausreißern
- Umwandeln von Datentypen
- Deduplizierung von Datensätzen

Bias-Check
- Sind die Daten für meine Fragestellung repräsentativ?
- Sind die enthaltenen Klassen ausbalanciert?

Schritte in einer ML-Pipeline
Feature Engineering und Auswahl, Labeling

Engineering

- *Überführung der Rohdaten in für ML-Modelle erlernbare Features (überwiegend numerisch)*
- Transformation: Skalierung, Normalisierung, Standardisierung, Binning
- Repräsentation: Encoding, Tokenisierung, Embeddings (siehe NLP-Basics aus Teil 2 der Schulung)
- Extraktion neuer Feature aus vorhandenen (z.B. Wochentag aus Zeitstempel, …)

Auswahl

- Auswahl relevanter Features mit hohem Informationsgehalt → Signal ⬆ Rauschen ⬇
- Auswahl über Domänenexpertise
- Automatisierte Verfahren, z.B. Dimensionsreduktion, Recursive Feature Elimination, …

Feature-Store

- Repository vorverarbeiteter und kuratierter Feature mit Dokumentation

Labeling für Supervised Learning

- Erstellen einer Ground Truth / Gold Standard

Schritte in einer ML-Pipeline
Feature Engineering und Auswahl, Labeling

Engineering

- *Überführung der Rohdaten in für ML-Modelle erlernbare Features (überwiegend numerisch)*
- Text muss in eine numerische Darstellung überführt werden
- Bag of Words
  - Repräsentation eines Dokuments bzgl. der Anzahl der im Korpus vorkommenden Wörter

Doc1
The dog chased the cat across the yard.

Doc2
A cat sits in the yard. There is another cat.

	dog	chase	cat	across	yard	sit	in	there	be	another
Doc1	1	1	1	1	1	0	0	0	0	0
Doc2	0	0	2	1	1	1	1	1	1	1

→ [1, 1, 1, 1, 1, 0, 0, 0, 0, 0]

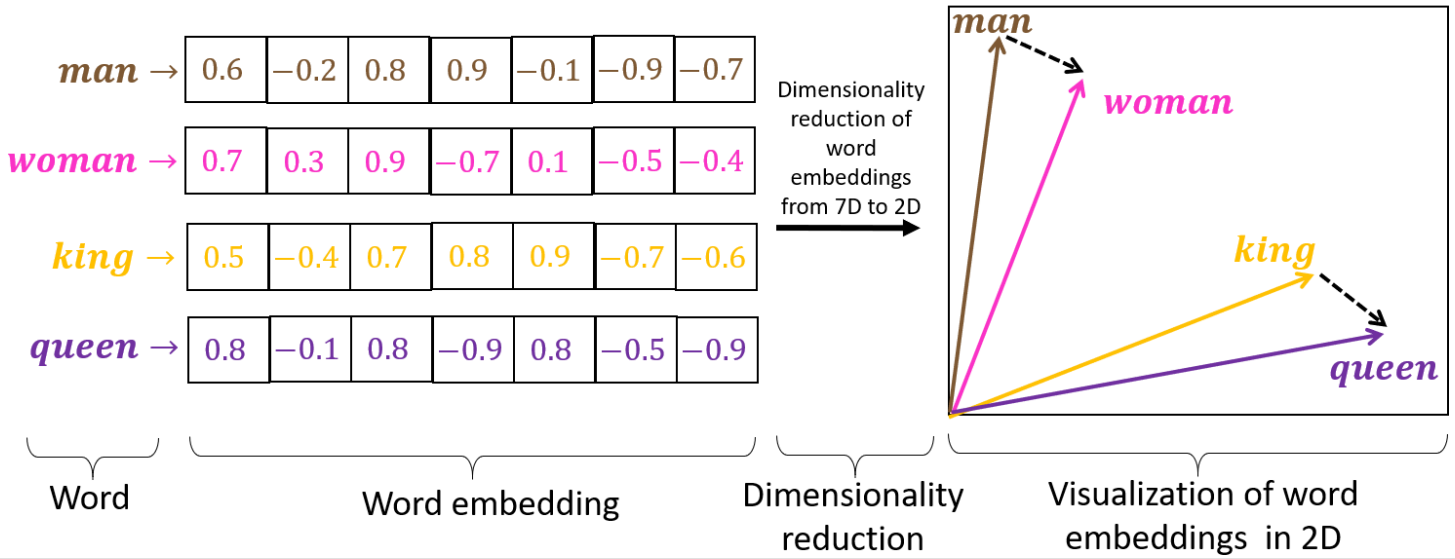
→ [0, 0, 2, 1, 1, 1, 1, 1, 1, 1]

- TF-IDF (Term Frequency – Inverse Document Frequency)
  - Gewichtete Bedeutung eines Worts bzgl. eines Dokuments und der Wörter aus Korpus
  - Seltene Wörter („dog“) erhalten höheres Gewicht als häufigere Wörter („cat“)
  - TF-IDF Vektor für Doc1 → [0.14, 0.14, 0, 0.14, 0, 0, 0, 0, 0, 0]
  - TF-IDF Vektor für Doc2 → [0, 0, 0, 0, 0, 0.09, 0.09, 0.09, 0.09, 0.09]

Schritte in einer ML-Pipeline
Feature Engineering und Auswahl, Labeling

Engineering

- *Überführung der Rohdaten in für ML-Modelle erlernbare Features (überwiegend numerisch)*
- Text muss in eine numerische Darstellung überführt werden
- Embeddings
  - Text wird in eine n-dimensionale Vektor-Darstellung überführt
  - Vektoren mit ähnlicher Richtung im n-dimensionalen Raum haben auch ähnliche Semantik

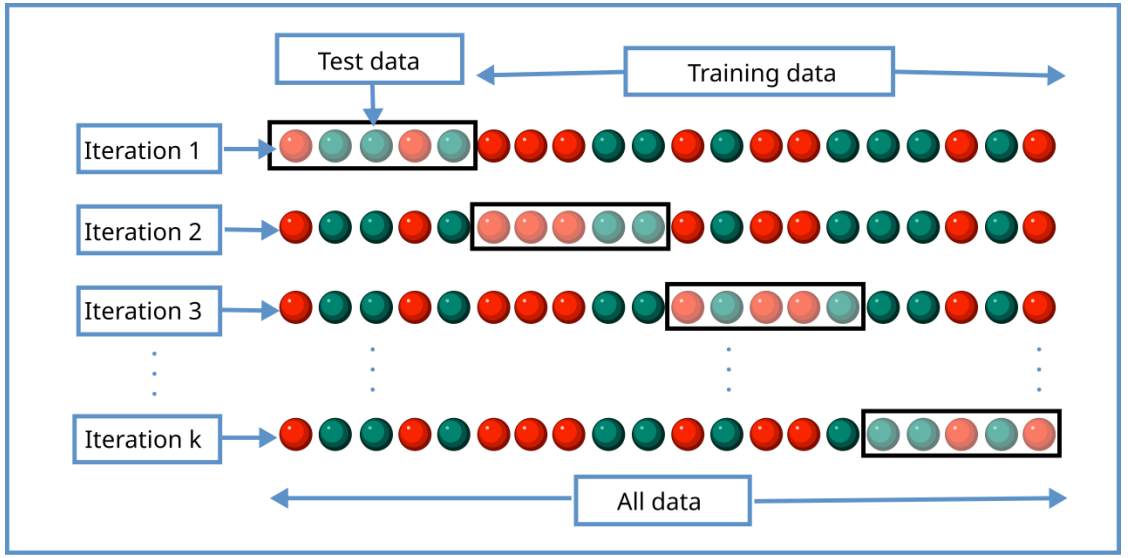


Rozado, David (2020). Word embeddings map words in a corpus of text to vector space. PLOS ONE. Figure. https://doi.org/10.1371/journal.pone.0231189.g008 (CC BY 4.0)

Schritte in einer ML-Pipeline
Train-Validation-Test Split

Vermeiden von „Data Leakage“ / „Auswendig Lernen“ des Modells

- Für Generalisierung auf neuen Daten ist Aufteilen der Daten nötig - separate Teile für
  - eigentliches Training des Modells
  - weitere Validierung mit Parameter Tuning, Auswahl anderer Feature, …
  - finale Evaluierung
- Verhältnis von *train:validate:test* z.B. 70:20:10 oder 80:10:10
- Immer auch abhängig vom Umfang der Daten, Verteilung von Klassen in den Daten (balanced), …
- Bei wenig Daten ist Cross-Validierung möglich



Quelle: Gufosowa, https://commons.wikimedia.org/w/index.php?curid=82298768, CC BY-SA 4.0

Schritte in einer ML-Pipeline
Modellwahl und Baseline

Modell nach Anwendungsfall auswählen

- Dummy-Modelle für spätere Vergleiche (naive Ansätze für Klassifikation / Regression)
- Einfach beginnen, z.b. Lineare Regression statt Neuronales Netz
- Kriterien:
  - Problemstellung und Analyse-Ziel
  - Beschaffenheit und Umfang der Daten
  - Art und Anzahl der Feature
  - Interpretierbarkeit
  - …

- Vergleich mehrerer Modell-Familien (z.B. Linear vs. Decision Tree vs. Neuronales Netz)
  - Kosten vs. Nutzen
  - Fortschritte messbar machen (Metriken, Laufzeit)
  - Versionsverwaltung sinnvoll

Schritte in einer ML-Pipeline
Modelltraining und Evaluierung

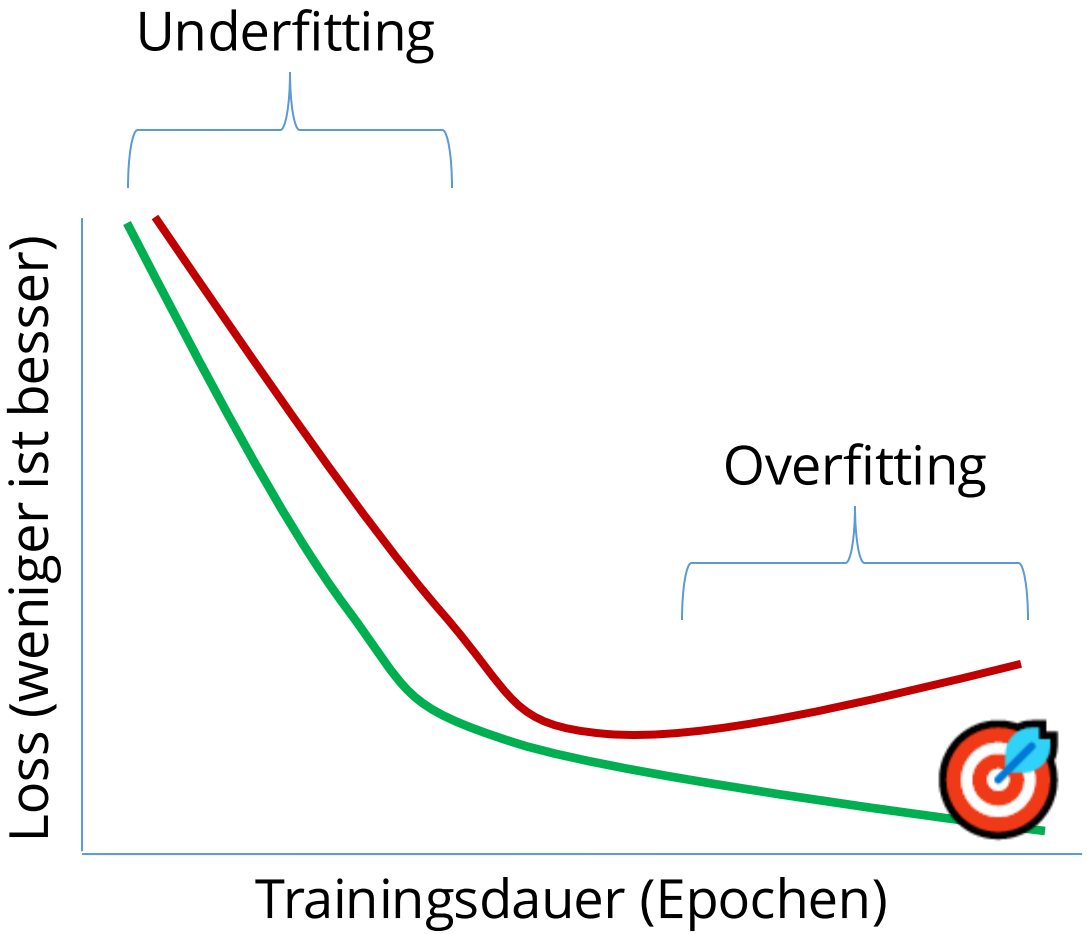
Training und Hyperparameter-Tuning

- Trainingsziel ist (meistens) die Minimierung eines definierten Fehlers (Loss-Function)
- Sowohl Modellgüte als auch Trainingsprozess lassen sich über Parameter optimieren
  - Modell-Parameter: z.B. Depth & Splits bei Decision Trees, Anzahl Layer & Neuronen bei NN, …
  - Trainings-Parameter: z.B. Anzahl Samples & Epochen, Learning Rate bei NN, …
- Viele Bibliotheken bieten dafür automatisierte Prozesse (Parameter-Grid-Search, …)

Evaluierung und Metriken

- Modellgüte (Qualität der Vorhersage) lässt sich mit unterschiedlichen Metriken prüfen
  - Klassifikation: Accuracy, Precision, Recall, F1, ROC-AUC, …
  - Regression: Variance, Mean Squared Error, r2, …
  - Clustering: Completeness Score, Silhouette Score, Elbow Method, …
- Konfusionsmatrix für Analyse der Vorhersagefehler bei Klassifikation
- Überblick gibt z.B. scikit-learn: https://scikit-learn.org/stable/modules/model_evaluation.html

Schritte in einer ML-Pipeline
Modelltraining und Evaluierung



Learning Curve / Loss Curve

- Hilft u.a. folgende Fragen zu beantworten
  - Lernt mein Modell überhaupt etwas?
  - Konvergiert mein Modell?
  - Trainiere ich lange genug / zu kurz?
  - Generalisiert mein Modell auf neuen Daten?
- Unterstützt beim Finetuning der Modell- und Trainingsparameter

Schritte in einer ML-Pipeline
Deployment und Monitoring

Bereitstellung des Modells

- Z.B. im Rahmen einer (Online) Anwendung
- Tests, Deployment, Versionierung, CI/CD
- IT-Sicherheit, Rechtlicher Rahmen

Monitoring des Modells / Anwendung

- Performance, Zugriffsraten
- Fehlerbehandlung
- Model Drift (Qualität der Ergebnisse ändert sich über die Zeit)

Monitoring der Daten

- Data Drift
  - Eigenschaften der Daten ändern sich über die Zeit
- Concept Drift
  - Zusammenhänge zwischen Eingabedaten und Vorhersageziel ändern sich über die Zeit

ML-Pipelines für Klassifikation und Clustering von Textdaten