
Technische Hochschule Georg Agricola

# INDUSTRIEAUTOMATION

## DATA SCIENCE IN DER PRODUKTION

**Bochum**

**Prof. Dr.-Ing. Lars N. Josler**

# AGENDA

1. Einordnung von Data Science für Ingenieure
2. Datenspeicherung in Datenbanken
3. Datenverarbeitung in Python

4. Machine Learning Basics
5. Ablauf einer Machine Learning Pipeline
6. Anwendungsbeispiel im Ingenieurskontext

# AGENDA

1. **Einordnung von Data Science für Ingenieure**
2. Datenspeicherung in Datenbanken
3. Datenverarbeitung in Python

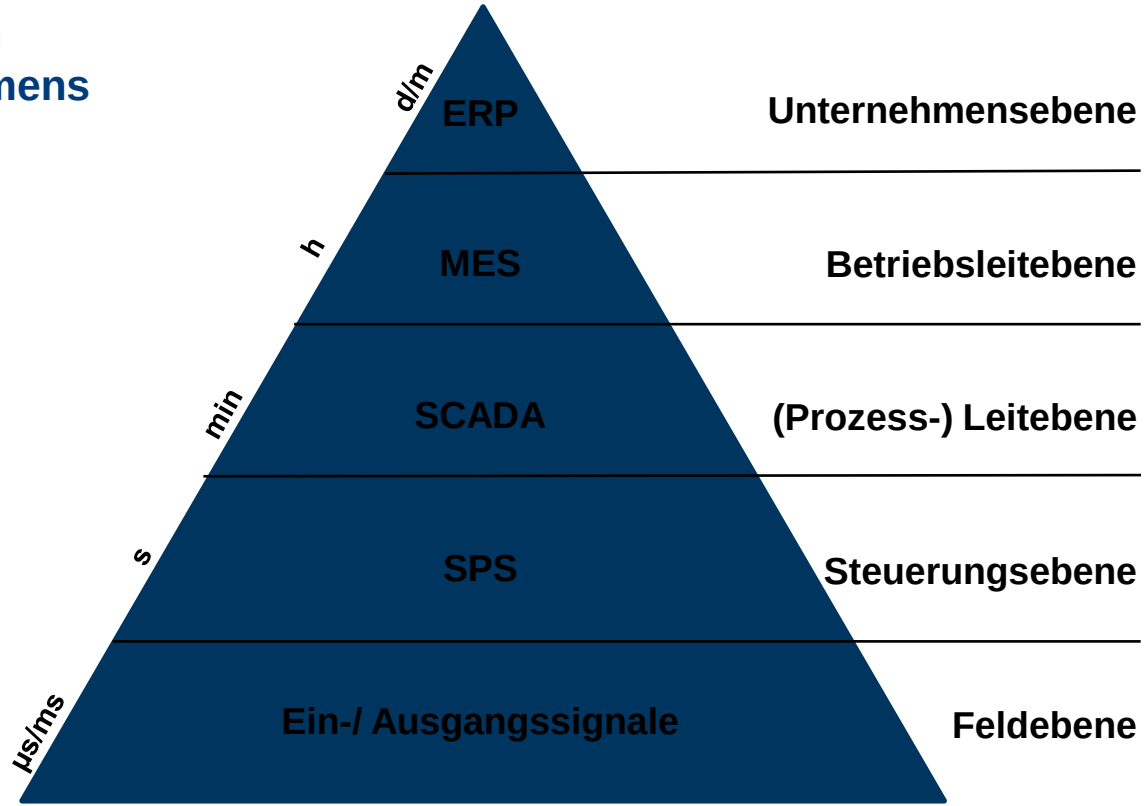
4. Machine Learning Basics
5. Ablauf einer Machine Learning Pipeline
6. Anwendungsbeispiel im Ingenieurskontext

# DATEN IM INGENIEURSKONTEXT

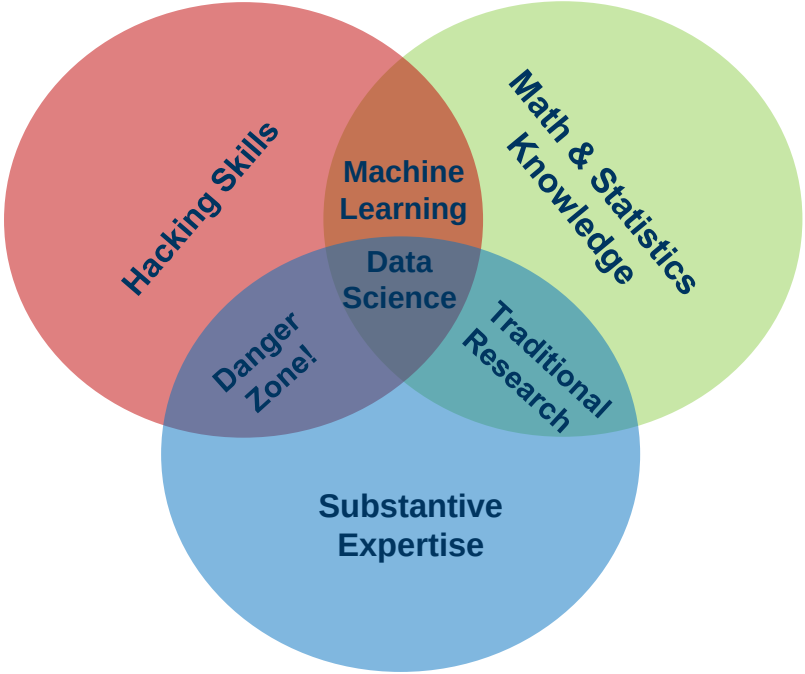
- **Produktionslinien**
  - Maschinendaten
  - Fertigungsdaten
- **Sensordaten**
  - (Autonome) Fahrzeuge
  - Wearables
  - VR/AR
  - IoT
  - …
- **Kundendaten**
  - Adressen
  - Bestellungen
  - …

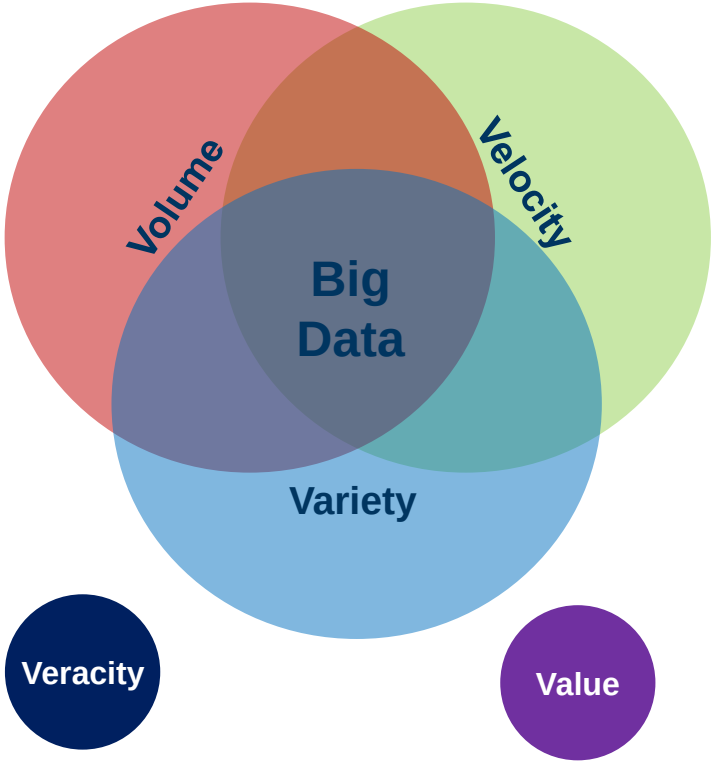
# WO ENTSTEHEN IN EINEM UNTERNEHMEN DATEN?

- **Daten entstehen in vielen Bereichen des Unternehmens**
- **Hohe Variation der Daten innerhalb der Ebenen in:**
  - Aggregationslevel
  - Frequenz
  - Informationsgehalt
  - Etc.
- **5 V‘s von Big Data**
  - Velocity
  - Variety
  - Veracity
  - Value
  - Volume



# EINLEITUNG MASCHINELLES LERNEN
## MACHINELLES LERNEN VS. BIG DATA



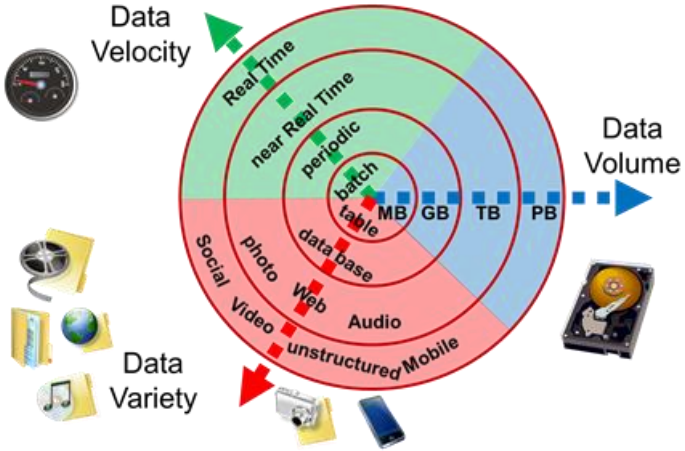

http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram

# HOW BIG IS BIG DATA?

- **Beispiel LFF**
  - $10\ \frac{MB}{Maschine \cdot Tag} \cdot 365\ \frac{Tage}{Jahr} \cdot 10\ Maschinen \approx 40\ GB/Jahr$

- **Beispiel Energieversorger (z.B. Smart-Meter Daten)**
  - $5\ \frac{MB}{Kunde \cdot Tag} \cdot 365\ \frac{Tage}{Jahr} \cdot 40000\ Kunden \approx 73000\ \frac{GB}{Jahr}$

- **Beispiel Social Media Website**
  - $100 \cdot 10^6\ \frac{Fotos}{Tag} \cdot 365\ \frac{Tage}{Jahr} \cdot 1\ \frac{MB}{Foto} \approx 36500000\ \frac{GB}{Jahr} = 36.5\ PB\ (Petabyte)$




**Quellen:**
PC
Server
Amazon
Snowmobile

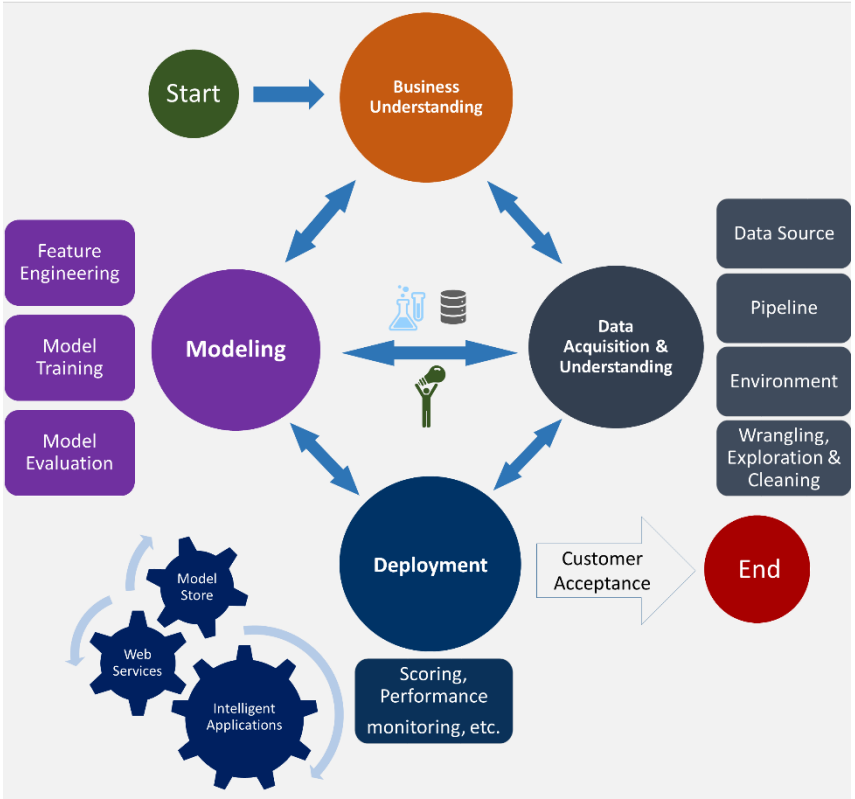
# LEAN / 5S BEI DATEN

| | | |
|---|---|---|
| ▪ **Seiri** ([Aus-]Sortieren) | ←→ | **Nur relevante Daten betrachten** |
| ▪ **Seiton** (Systematisieren) | ←→ | **Datenkonsistenz (Bsp: Eindeutige ID)** |
| ▪ **Seiso** (Säubern) | ←→ | **Entferne fehlerhafte Werte** |
| ▪ **Seiketsu** (Standardisieren) | ←→ | **übergreifenden Standards schaffen** |
| ▪ **Shitsuke** (Selbstdisziplin) | ←→ | **manuelle (Betriebs-)Daten korrekt eingeben** (Störungen, Fehler etc.) |

# DATA SCIENCE LEBENSZYKLUS

- **Team Data Science-Prozess von Microsoft (TDSP)**
  - Beschreibt verschiedene Phasen der Datenanalyse
  - Kein statischer Verlauf; Iterationszyklen
  - Alle Modelle beinhalten 4 zentrale Phasen:
    1. Business Understanding
    2. Datenerzeugung und Datenverständnis
    3. Modellbildung
    4. Nutzung

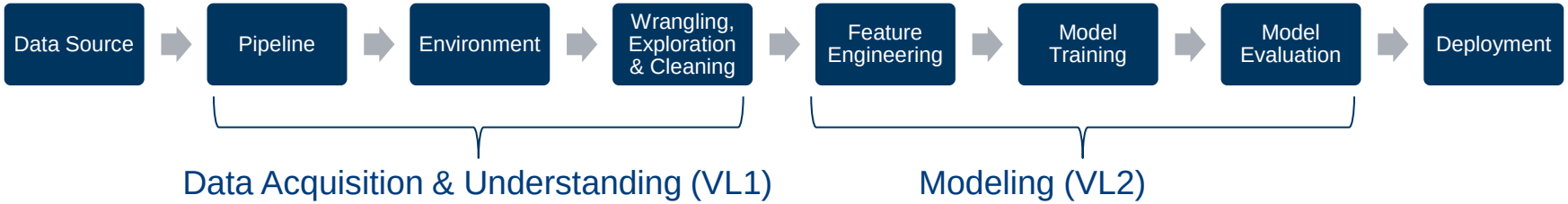
- **Weitere Modelle:**
  - CRISP-DM (Cross Industry Standard Process for Data Mining)
  - KDD (Knowledge Discovery in Databases)




https://docs.microsoft.com/de-de/azure/machine-learning/team-data-science-process/lifecycle

# VERSCHIEDENE PHASEN VON DATA SCIENCE

- **Fokus der Vorlesungsreihe auf den Bereichen des Data-Engineering und der Data-Analysis**




- **Data Acquisition & Understanding:**
  - Was ist eine sinnvolle Art die Daten zu speichern?
  - Wie werden die Daten bereitgestellt?
  - Wie können die Daten verarbeitet werden?
  - Wie müssen die Daten aufbereitet werden?

- **Modeling:**
  - Welche Frage möchte ich beantworten? / Welches Problem möchte ich lösen?
  - Welche (historischen) Daten stehen zur Verfügung?
  - Wie sehen meine Daten aus?
  - Welche Modellierungsart eignet sich?

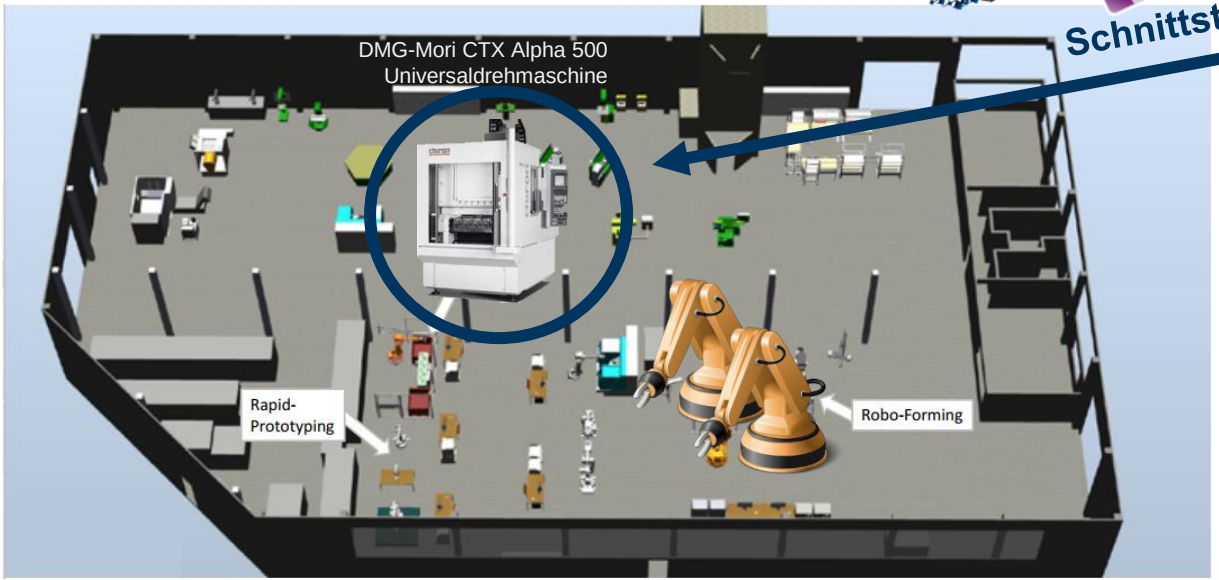
# AGENDA

1. Einordnung von Data Science für Ingenieure
2. **Datenspeicherung in Datenbanken**
3. Datenverarbeitung in Python

4. Machine Learning Basics
5. Ablauf einer Machine Learning Pipeline
6. Anwendungsbeispiel im Ingenieurskontext

# STATUS QUO

# DIGITALE FABRIK




OPC UA
MQTT.ORG
Schnittstellen

Open for Innovation KNIME
tableau
rapidminer
python

Data-Analysis
Software

https://de.wikipedia.org/wiki/Datei:Python_logo_and_wordmark.svg
https://rapidminer.com
https://www.tableau.com/de-de
https://www.knime.com/knime-analytics-platform/
http://mqtt.org/
https://new.siemens.com/global/de/produkte/automatisierung/industrielle-kommunikation/opc-ua.html

# PRO & CON VON STATUS QUO

## Status Quo:

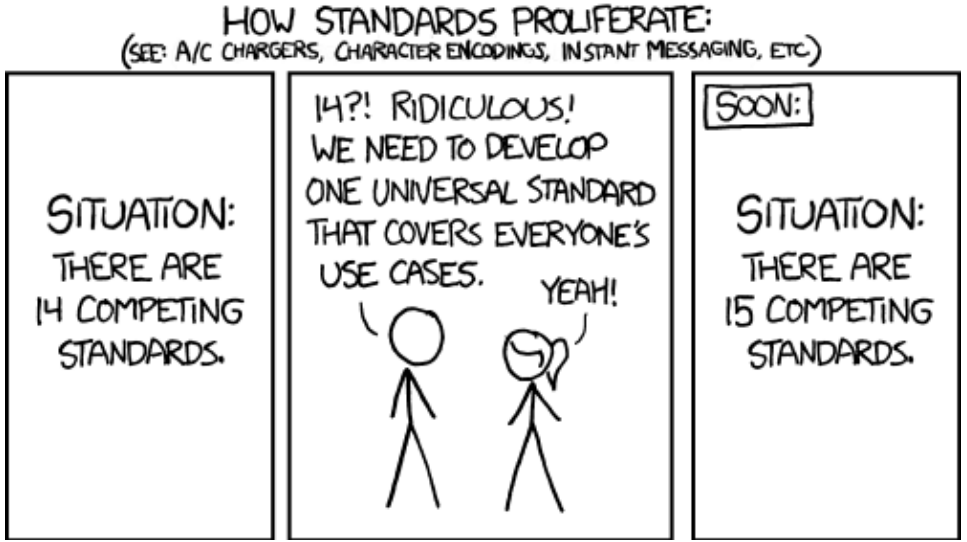
- **Pros**
  - Infrastruktur schon vorhanden
  - Know-How/ Kompetenzen schon vorhanden
  - „Das haben wir schon immer so gemacht“
- **Cons**
  - Keine zentrale Datenspeicherung
  - Eingeschränkte Data Governance
  - Eingeschränkte Vernetzung der Produktionssysteme/ in der gesamten Organisation
  - Kein Backups von Daten
  - Prinzipien für der Datenhaltung werden häufig nicht eingehalten

## Digitale Fabrik:

- **Pros**
  - Regelmäßige Backups und dezentraler Datenzugriff möglich
  - Steuerung + Regelung der Produktionsprozesse
  - Übersicht über bestehende Datenbasis der Produktionslinie/ des Unternehmens
  - Auseinandersetzung mit bestehenden Daten fördert Prozesswissen
  - Grundlage für jeden Data Science Use-Case
- **Cons**
  - Zusätzliche Hardware / Know-How/ Wartung notwendig
  - Zusätzliche Wartung der IT notwendig
  - Komplexe Einbindung in bestehende Systeme und Prozesse

# FEHLER / INKONSISTENZEN IN DATEN

- **Realweltdaten sind „hässlich“**
  - Oftmals keine Standardisierung vorhanden
  - Dezimaltrennzeichen 23,4 vs. 23.4 (z.B. in einem System verbaute EU- und US-Sensoren)
  - NaN – Not a Number
  - Unterschiedliche wissenschaftliche Notation: 1.5832e^-5 vs. 1.5832E^-5
  - Encodings und Decoding (utf-8, Ascii etc.)
  - Zeitzonen (UTC, CET) / Zeitstempel (UNIX)
  - Hohe Aggregation z.B. Messwert: 23.1 cm Abweichung jedoch in µm wichtig.
  - Inkonsistenzen (Adresse ohne Geokoordinate; Keine Eindeutigkeit in IDs)



# WARUM KEIN EXCEL ZUR SPEICHERUNG?

## Status Quo: Viel in Excel

### Pros

- GUI-Interface
- In jeder Firma auf jedem Computer verfügbar
- Für jeden zugänglich und weit verbreitet
- Automatisierung mit VBA

### Cons

- Nicht für „große“ Datenmengen ausgelegt (>1.000.000 Zeilen)
- Speicherung und Berechnung der Daten sind nicht getrennt
- Verwaltung von Zugriffsrechten aufwändig
- Geringe Skalierbarkeit und Verlustsicherheit

## Datenbank als Alternative

### Pros

- Skalierbarkeit
- Trennung von Speicherung und Berechnung
- Verwaltung von Zugriffsrechten und Backup der Daten
- Verteilte Datenhaltung möglich

### Cons

- Schwer zugänglich, hohe Lernkurve
- In der Regel kein GUI-Interface
- Nicht auf jedem System verfügbar
- SQL als Sprache weniger einfach zugänglich als Excel Formeln

# DATENTYPEN

## Standard Datentypen

| Name | Beispiel Beschreibung | Beispiel Eintrag |
|---|---|---|
| **Bool** (Binär) | Maschine an/aus | True oder False |
| **Integer** (Ganzzahl) | Anzahl an Schrauben | 42 |
| **Float** (Gleitkommazahl) | Stromstärke | 15.732 mA |
| **Char/String** (Zeichenkette) | Name Bearbeiter | Hans Müller |
| **Datetime** (Datum) | Zeitstempel Bearbeitung | 2018-17-11 17:37:25 |

SQL-Datenbanken → Strukturierte Daten

## Komplexe Datentypen

- Audiodateien
- Bilder/Video
- Geometrien
- Objekte
- CAD-Dateien
- …

No-SQL Datenbanken → Nicht strukturierbare Daten

# KONKRETES BEISPIEL TABELLE

| Zeitstempel | Maschinen-name | Maschinen ID | Stromstärke Motor [mA] | Maschine an | Werkzeug ID | Produkt ID |
|---|---|---|---|---|---|---|
| 2018-17-11 17:25:11:25 | Fz12 | 2385 | 17.35 | 1 | 2357 | 67800 |
| 2018-17-11 17:25:11:50 | Fz12 | 2385 | 17.78 | 1 | 2357 | 67800 |
| … | | | | | | |
| 2018-17-11 17:37:00:00 | Fz12 | 2385 | 0 | 0 | 2357 | 67800 |
| … | | | | | | |
| 2018-17-11 17:40:00:00 | Fz12 | 2385 | 18.50 | 1 | 9047 | 67800 |
| Date | String | Integer | Float | Bool | Integer | Integer |

# DATENBANKEN BEGRIFFE

- **Tabelle/ Relation**

**Attribut/Spalte/Feld**

**Primärschlüssel**

**Tupel/Zeile**

| Spaltenname 1 | Spaltenname 2 | Spaltenname 3 |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

**View**

- **Datenbank**

- **Datenbanksystem**

# DATENBANKEN BEGRIFFE

- Relation: Tabelle von strukturierbaren Daten
- Attribut/ Spalte/ Feld: Spalte in einer Tabelle
- Tupel: Zeile in einer Tabelle
- Primärschlüssel: Wert zur eindeutigen Identifizierung einer Zeile → muss innerhalb der Tabelle eindeutig sein
- View: Subset von ausgewählten Spalten und Zeilen einer Relation
- Relationale Datenbank: Sammlung von einzelnen Relationen
- Datenbanksystem: Sammlung von relationalen Datenbanken

# PRIMÄRSCHLÜSSEL – JA ODER NEIN

- **Primärschlüssel: Wert zur eindeutigen Identifizierung einer Zeile → muss innerhalb der Tabelle eindeutig sein**

| Tabelle | Wert | JA | NEIN |
|---|---|---|---|
| Studenten | Name + Vorname + Geburtsdatum | | |
| Studenten | Matrikelnummer | | |
| Arbeitnehmer | Sozialversicherungsnummer | | |
| Kraftfahrzeuge | Kennzeichen | | |
| Kraftfahrzeuge | VIN-Nummer | | |

| Kennzeichen |
|---|
| BOS738 |
| B-OS-738 |
| BO-S-738 |

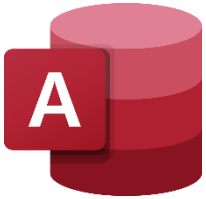
# PRIMÄRSCHLÜSSEL – JA ODER NEIN

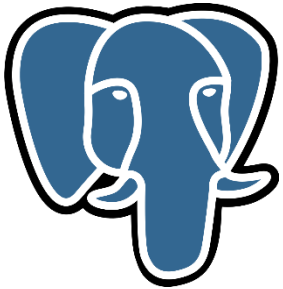
- **Primärschlüssel: Wert zur eindeutigen Identifizierung einer Zeile → muss innerhalb der Tabelle eindeutig sein**

| Tabelle | Wert | JA | NEIN | ES KOMMT DRAUF AN |
|---|---|---|---|---|
| Studenten | Name + Vorname + Geburtsdatum | | x | |
| Studenten | Matrikelnummer | x | | |
| Arbeitnehmer | Sozialversicherungsnummer | x | | |
| Kraftfahrzeuge | Kennzeichen | | | x |
| Kraftfahrzeuge | VIN-Nummer | x | | |

| Kennzeichen |
|---|
| BOS738 |
| B-OS-738 |
| BO-S-738 |

# BEISPIELE FÜR DATENBANKSYSTEME







- **Kriterien für die Auswahl eines Datenbanksystems:**
  - Art der Daten (SQL vs. No-SQL)
  - Anzahl der Daten
  - Änderungsfrequenz der Daten (statisch vs. Dynamisch vs. Stream)
  - Budget (Open-Source vs. Proprietär)
  - Verwaltungsaufwand
  - Speicherort (lokal, Server, verteilt)
  - Anzahl an Nutzern

# BEISPIEL TABELLE STUDIERENDE

- **Informationen über Studierende:**
  - Allgemeine Informationen (Matrikelnummer, E-Mail Adresse, Name, Vorname, Geburtsjahr, Studienbeginn, Studiengang)
  - Adresse (Straße, Hausnr., PLZ, Stadt)
  - Ausweisinformationen (Ausweisnummer, Ausweisguthaben)
  - Fächer (belegte Fächer, Anzahl an Versuchen, Note)
  - Sonstiges (Semesterbeitrag bezahlt)

Welche Möglichkeiten gibt es, diese Informationen in einer Datenbank zu speichern?

# NORMALFORMEN

## Möglichkeit I: Alle Informationen in einer Tabelle speichern

| Matr. Nr. | Name | Vorname | Adresse | Ausweis Nr. | Ausweisguthaben | Fach | Note |
|---|---|---|---|---|---|---|---|
| 1001 | Simpson | Homer | 742 Evergreen Terrace | 7777 | 0.70 € | Reaktorsicherheit | 1.0 |
| 1001 | Simpson | Homer | 742 Evergreen Terrace | 7777 | 0.35 € | Reaktorsicherheit | 1.0 |
| 0001 | Burns | Mr. | Am Reaktor 3 | 8567 | 10000000 € | Mechanik A | 2.7 |
| 0001 | Burns | Mr. | Am Reaktor 3 | 8567 | 10000000 € | Reaktorsicherheit | 4.0 |
| 0001 | Burns | Mr. | Am Reaktor 2 | 8567 | 10000000 € | Bachelor Arbeit | 1.7 |
| 1003 | Szyslak | Moe | 15 Moe‘s Tavern | - | - | - | - |

**Probleme:**

- **Redundanzen**
- **Unübersichtlich**
- **Probleme beim Löschen/ Updaten/ Einfügen von Informationen**

# NORMALFORMEN

**Möglichkeit II: Mehrere Tabellen anlegen, die in Normalform sind**

**Normalform: Tabelle befindet sich in einer Form, die keine vermeidbaren Redundanzen enthält**

## Studierende

| Matr. Nr. | Name | Vorname | Adresse |
|---|---|---|---|
| 1001 | Simpson | Homer | 742 Evergreen Terrace |
| 0001 | Burns | Mr. | Am Reaktor 3 |
| 1003 | Szyslak | Moe | 15 Moe‘s Tavern |

## Ausweis

| Ausweis Nr. | Matr. Nr. | Ausweisguthaben |
|---|---|---|
| 7777 | 1001 | 0.70 € |
| 8567 | 0001 | 10000000 € |

## 0001

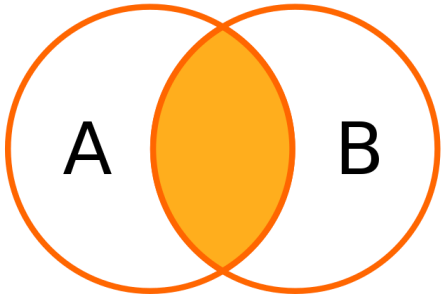
| Fach | Note |
|---|---|
| Mechanik A | 2.7 |
| Reaktorsicherheit | 4.0 |
| Bachelor Arbeit | 1.7 |

## Reaktorsicherheit

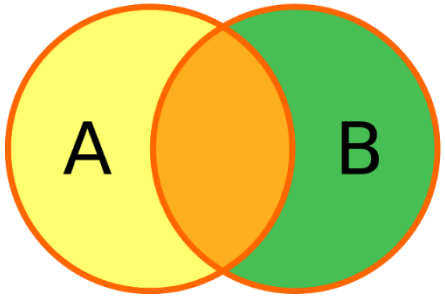
| Matr. Nr. | Note |
|---|---|
| 1001 | 1.0 |
| 0001 | 4.0 |

# JOINS

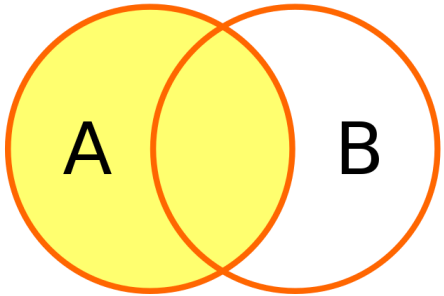
- **Kombinieren von mehreren Tabellen für einfache Berechnungen**



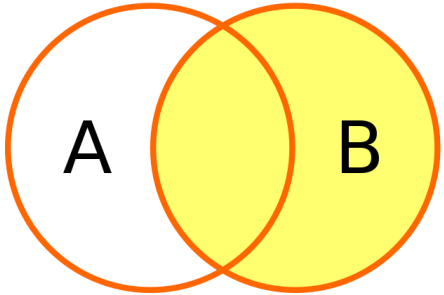

Inner-Join




Outer-Join




Left-Join




Right-Join

## LIVE DEMO JOIN

# SQL

- **Structured Query Language**
  - Sprache zur Handhabung von relationalen Datenbanken → Erstellung, Veränderung und **Durchsuchen** und Rechteverwaltung von Datenbanken
  - Wird von jeder (SQL-) Datenbank unterstützt und ist weitestgehend einheitlich definiert
  - Deklarative Programmiersprache → Lesbar und an die englische Sprache angelehnt
  - Von Nutzern von Datenbanken am häufigsten verwendete Funktionalitäten
    - Auswahl/ Filterung
    - Sortieren von Ergebnissen
    - Basic Rechenoperationen (Summen, Durchschnitt, Median, Standardabweichung, Zählen, Min, Max)
    - Join Operation (Kombinieren von Informationen aus mehreren Tabellen)
    - Group-By Operation ( Anwendung von Aggregation auf Gruppen von Elementen)

# AGENDA

1. Einordnung von Data Science für Ingenieure
2. Datenspeicherung in Datenbanken
3. **Datenverarbeitung in Python**

4. Machine Learning Basics
5. Ablauf einer Machine Learning Pipeline
6. Anwendungsbeispiel im Ingenieurskontext

# WARUM KEIN EXCEL ZUR BERECHNUNG?

## Status Quo: Alles in Excel

- **Pros**
  - GUI-Interface
  - In jeder Firma auf jedem Computer verfügbar
  - Für jeden zugänglich und weit verbreitet
  - Erweitertes Programmieren mit VBA
- **Cons**
  - Geringe Skalierbarkeit
  - Zahlen sind sichtbar, aber Formeln versteckt
  - Eingeschränkte Toolboxen
  - Code kann nicht in Repositories verwaltet werden

## Wissenschaftliches Rechnen als Alternative

- **Pros**
  - Skalierbarkeit auf große/ sehr große Daten
  - Trennung von Speicherung und Berechnung
  - Open-Source Gedanke weit verbreitet
  - i.d.R. ein größeres „Ökosystem“ an verfügbaren Toolboxen
  - Umsetzung komplexer Softwareprojekte möglich
- **Cons**
  - Erfordert grundsätzliches Wissen über Programmierung
  - Nicht standardmäßig auf jedem System verfügbar
  - Standardmäßig kein GUI-Interface → weniger zugänglich

# WISSENSCHAFTLICHES RECHNEN

## Kompiliert

- Kompiliert: Quellcode → Compiler → ausführbare Datei
- C, C++, C#, Java, Fortran
- Effiziente und schnelle Ausführung
- Sehr low-level, schwierig zu debuggen

```
/* Hello World in C++ */
#include <iostream>
using namespace std;
int main() {
            cout << "Hello, World!";
            return 0;
            }
```

## Interpretiert

- Interpretiert: Quellcode → Interpreter → Ausführen
- Python, Matlab, R, JavaScript
- Einfache Handhabung/ Debugging
- Speicherineffizient und „langsam“

```
# Hello World in Python
print("Hello, World!")
```

## Wissenschaftliches Rechnen

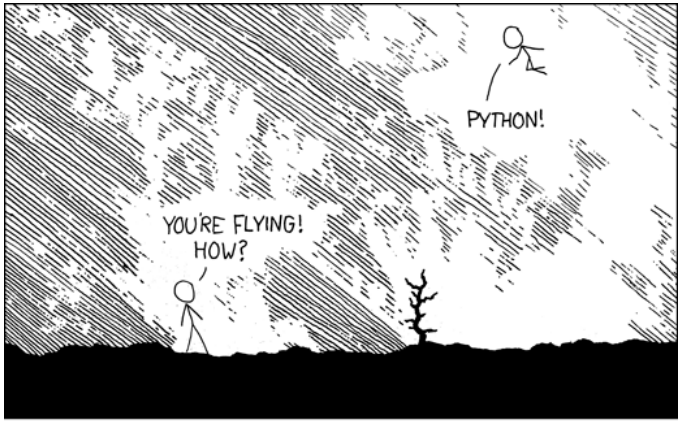
- Programmierschnittstelle zwischen interpretierter und kompilierter Sprache für Nicht-Programmierer
- Interface in interpretierter Sprache und „Heavy-Lifting“ in kompilierter Sprache

# PYTHON

## Python

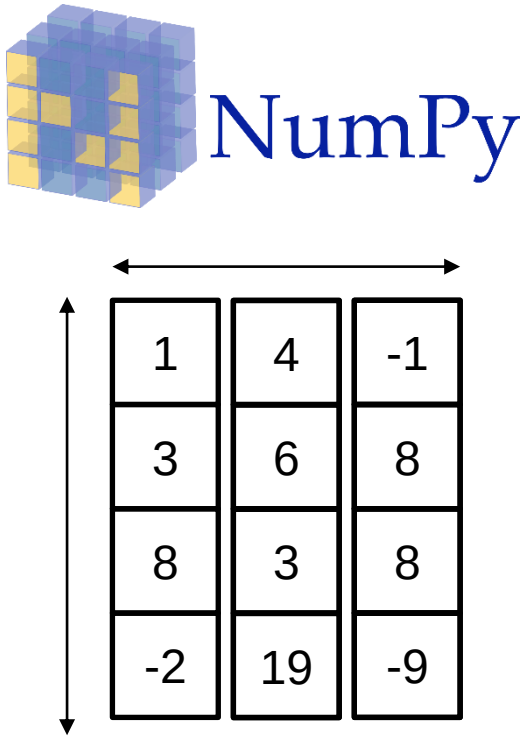
- Interpretierte, high-level, open-source Programmiersprache
- Fokus auf Lesbarkeit und klarer Syntax
- In den letzten Jahren zu einer der populärsten Programmiersprachen geworden
- Enorm große Community und viele Erweiterungen

- **Python für Wissenschaftler und Ingenieure:**
  - **Numpy/ Scipy** für wissenschaftliches Rechnen
  - **Matplotlib/ Bokeh/ Altair** für Plotting
  - **Pandas** für Tabellen/Datenbank Kalkulation
  - **Scikit-Learn/ Tensorflow/ PyTorch** für Machine- und Deep Learning
  - **Anaconda** als Paket Manager
  - **Jupyter** für wiederverwendbare Dokumente mit Text und Code




# NUMPY ARRAY

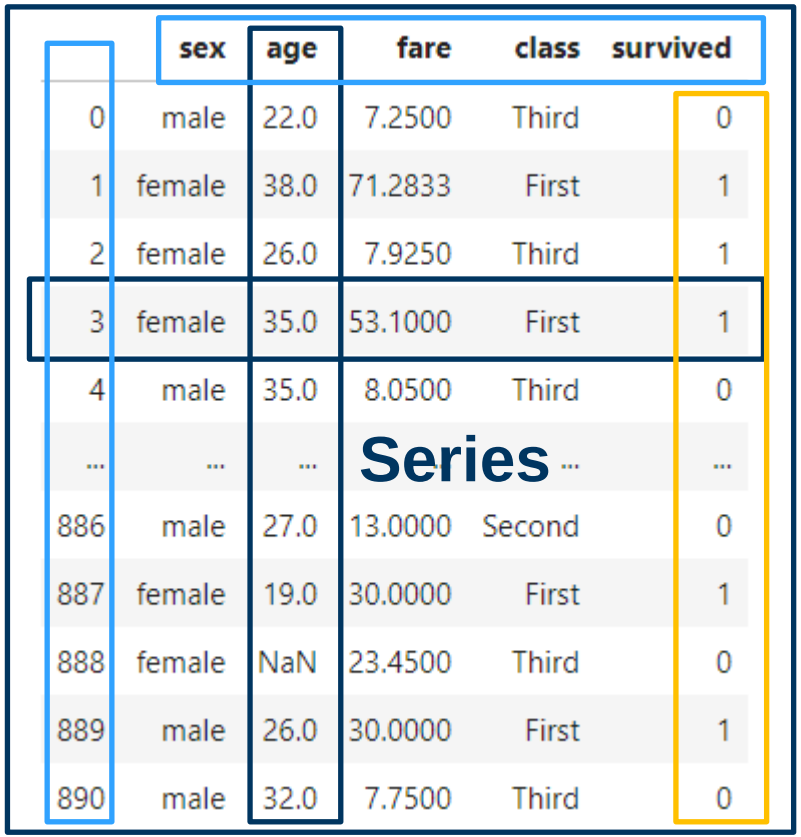
- **Num**erical **Py**thon
  - BLAS (**B**asic **L**inear **A**lgebra **S**ystem) für Python
  - Vektor/Matrix als Basisobjekt → Numpy n.d array
  - **Deutlich** speichereffizienter und performanter als „normales“ Python
  - Verknüpfung von mathematischen Operationen und Matrizenrechnung
  - Alle Einträge eines Vektors/ einer Matrix müssen den gleichen **Datentyp** haben
  - Schnelles und effizientes **Indexing**
  - Basis für viele weitere Software-Pakete innerhalb Pythons Ökosystem



# PANDAS

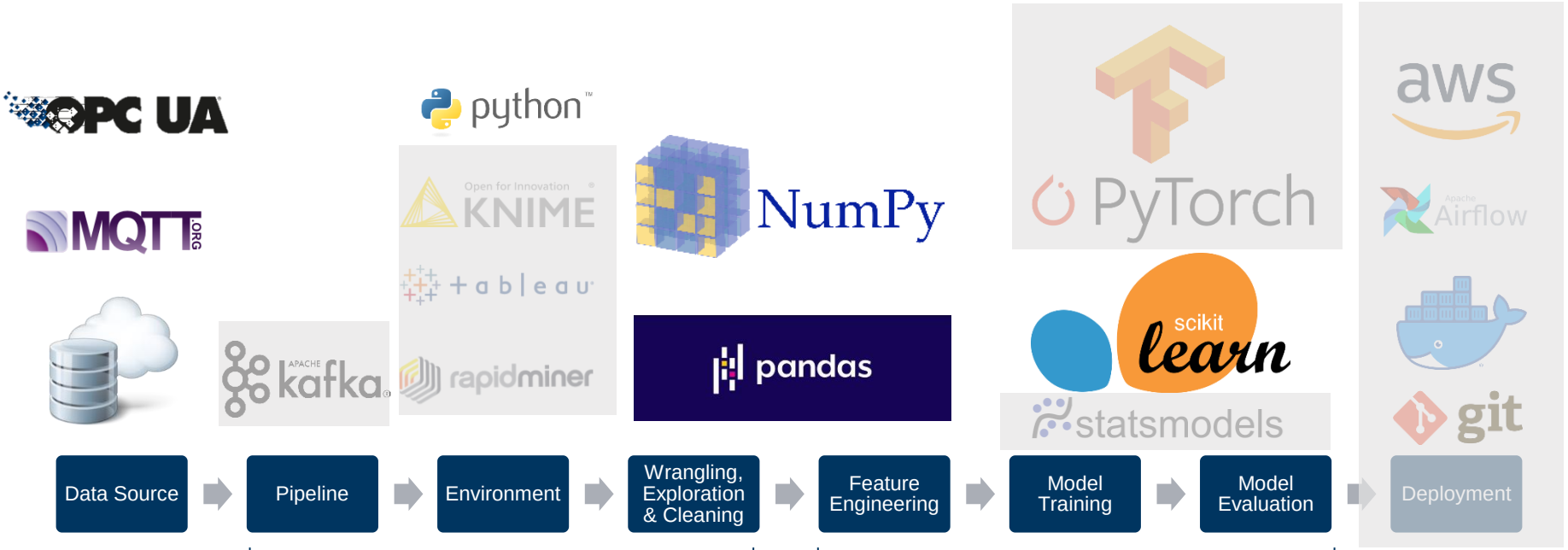
- **Pandas**
  - Sehr effiziente **Datenanalyse** und **Datenmanipulation** in Python
  - Lesen und Schreiben in verschiedene Datenformate (CSV/Excel-files/ Datenbanken)
  - Unterstützt **TimeSeries** und **Indexing** Features
  - Ausführen von **Datenbankoperationen** (Joins/Group-Bys)
  - Verarbeitung von fehlenden Datenpunkten
  - De-facto Standard Datenverarbeitungspaket im Data Science Bereich für das Python Ökosystem
  - In der Industrie und der akademischen Welt weit verbreitet

# PANDAS DATENTYPEN



| | sex | age | fare | class | survived |
|---|---|---|---|---|---|
| 0 | male | 22.0 | 7.2500 | Third | 0 |
| 1 | female | 38.0 | 71.2833 | First | 1 |
| 2 | female | 26.0 | 7.9250 | Third | 1 |
| 3 | female | 35.0 | 53.1000 | First | 1 |
| 4 | male | 35.0 | 8.0500 | Third | 0 |
| ... | ... | ... | ... | ... | ... |
| 886 | male | 27.0 | 13.0000 | Second | 0 |
| 887 | female | 19.0 | 30.0000 | First | 1 |
| 888 | female | NaN | 23.4500 | Third | 0 |
| 889 | male | 26.0 | 30.0000 | First | 1 |
| 890 | male | 32.0 | 7.7500 | Third | 0 |

# DEVELOPMENT STACK




Wikipedia

# WEITERES MATERIAL

- Code Academy SQL
- Database Fundamentals
- Data School Pandas Basics
- Introduction to Analytics with Python
- Data Science Handbook