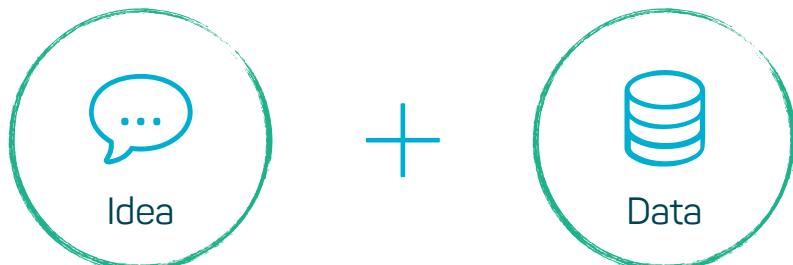

Machine Learning in Produktion

oder: Warum Software-Entwicklung auch hier wichtig ist

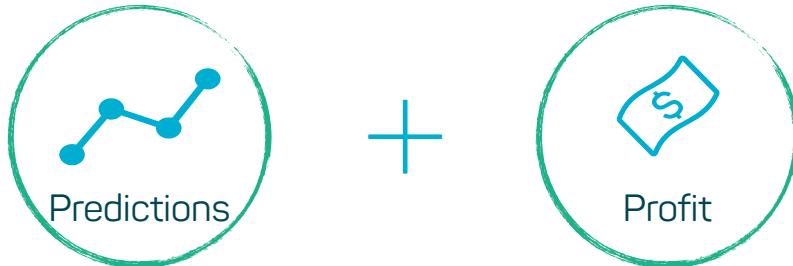


Data Science Projekte – Eine neue Idee

Have:

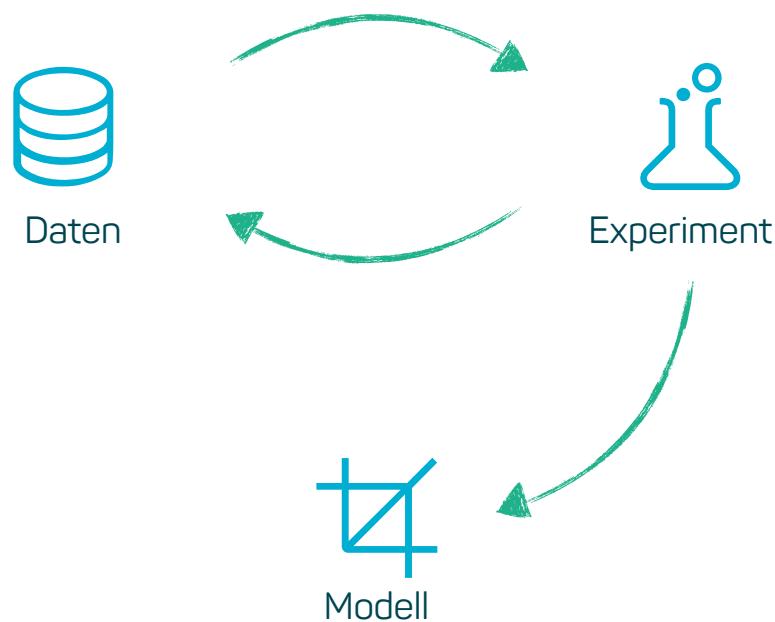


Want:



Data Projekte – Exploration

Ziel: Schnelle Exploration und schnelle Experimente

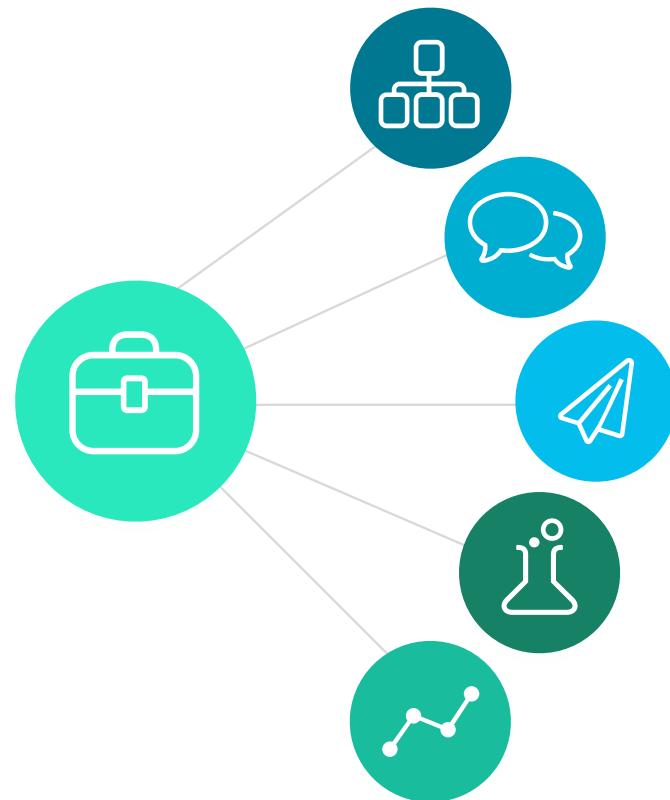


Jupyter notebook

```
data = load_data("input.csv")
data = preprocess(data)
model = train_model(data)
print(model.predictions)
```

Modell Entwicklung

-  Datenbeschaffung & Speichern
-  Cleaning
-  Vorverarbeitung
-  Modell Training
-  Evaluation



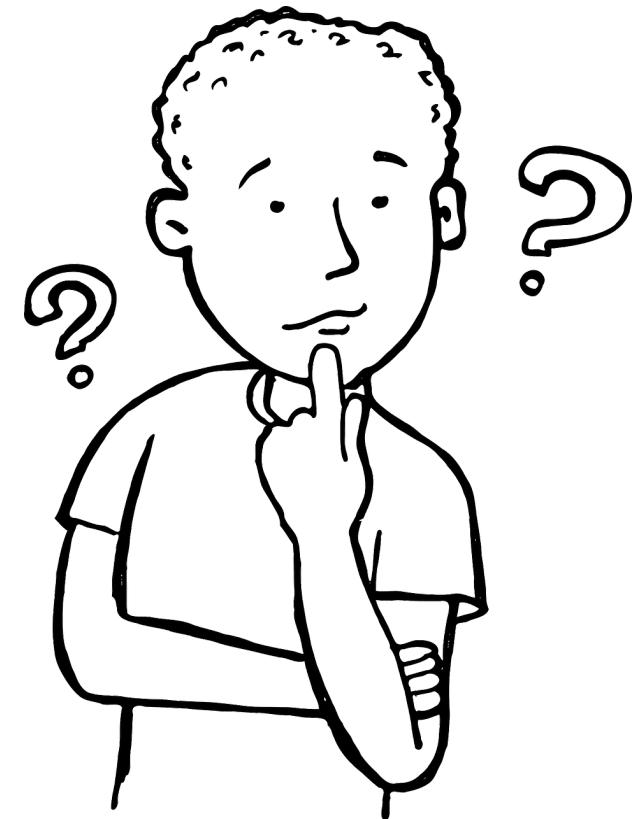


Experiment funktioniert!

Und nun?

Wie geht es weiter?

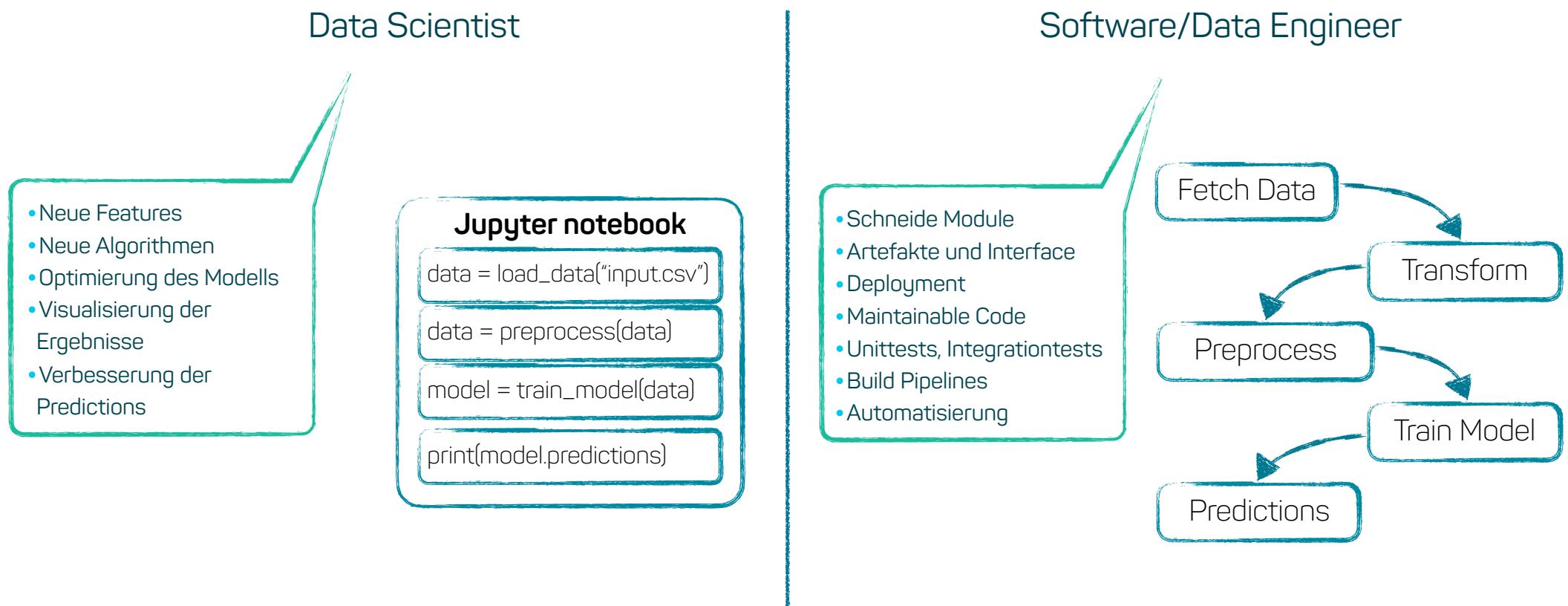
- Wie kommen die Predictions zum User?
- Wie soll das Modell deployed werden?
- Welche Bereiche des Unternehmens sind involviert?
- Woher kommen "live" Daten?
- Wie könnten User frühzeitig einbezogen werden?
- Was muss für einen wirtschaftlichen Nutzen passieren?



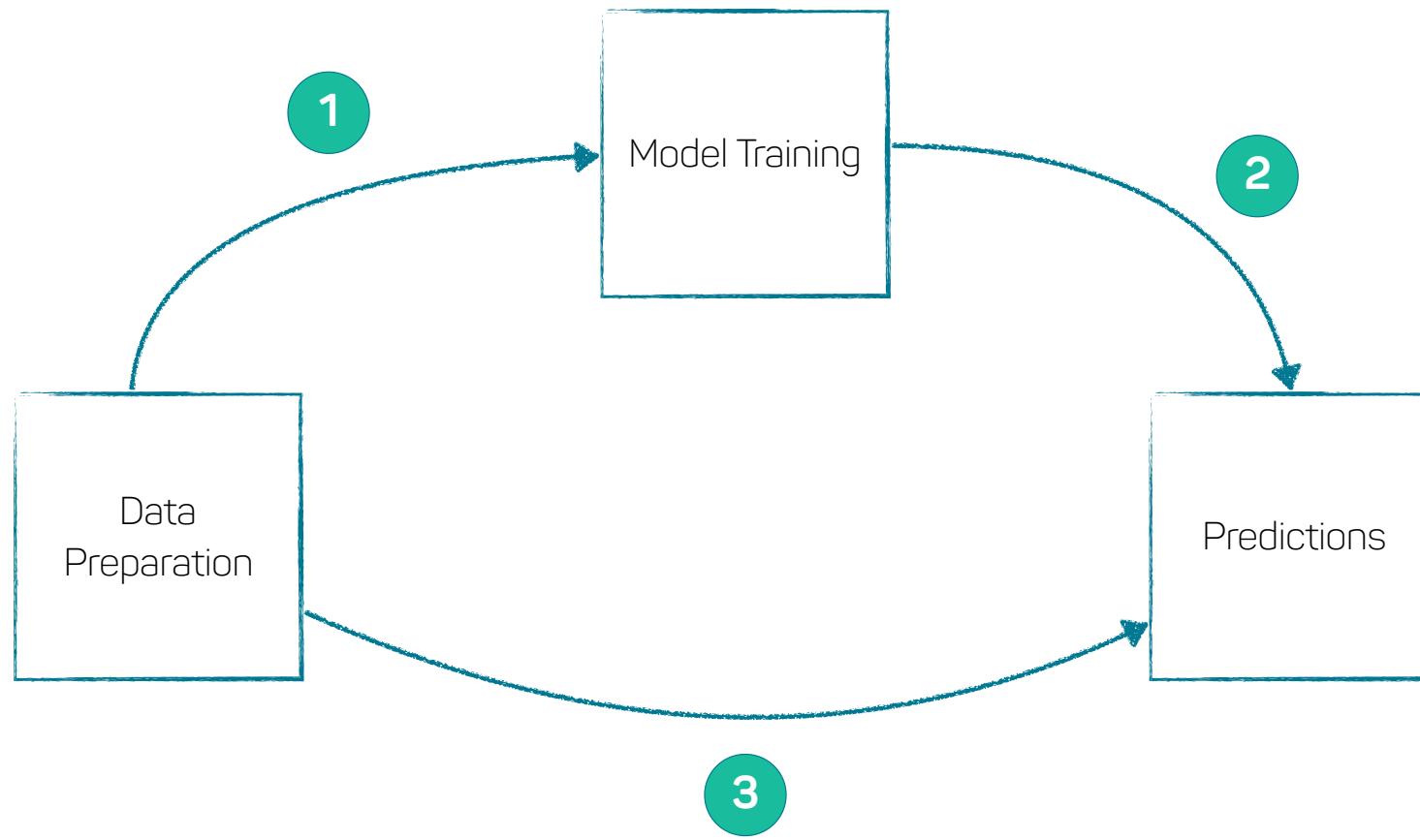


Die zwei Seiten eines Daten-Projekts

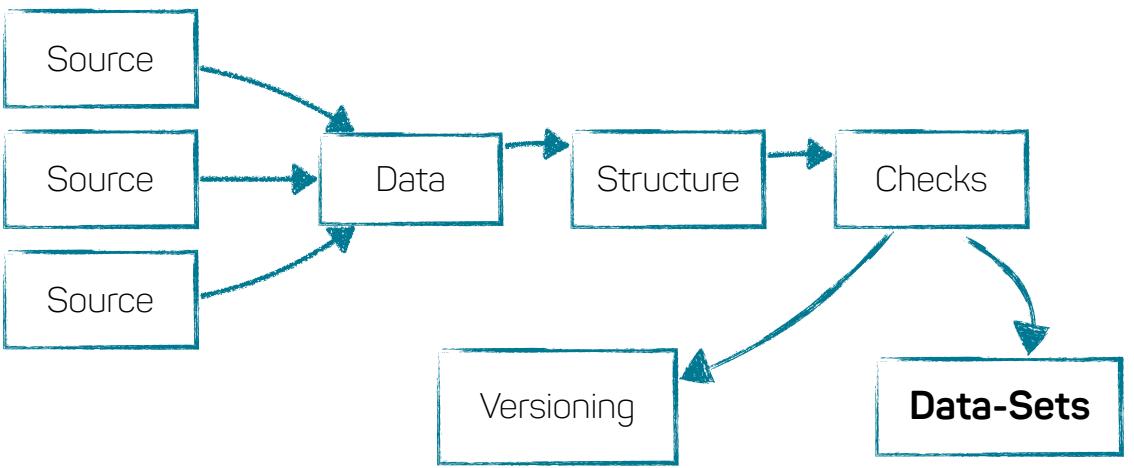
Daten-Projekt – Zwei Perspektiven



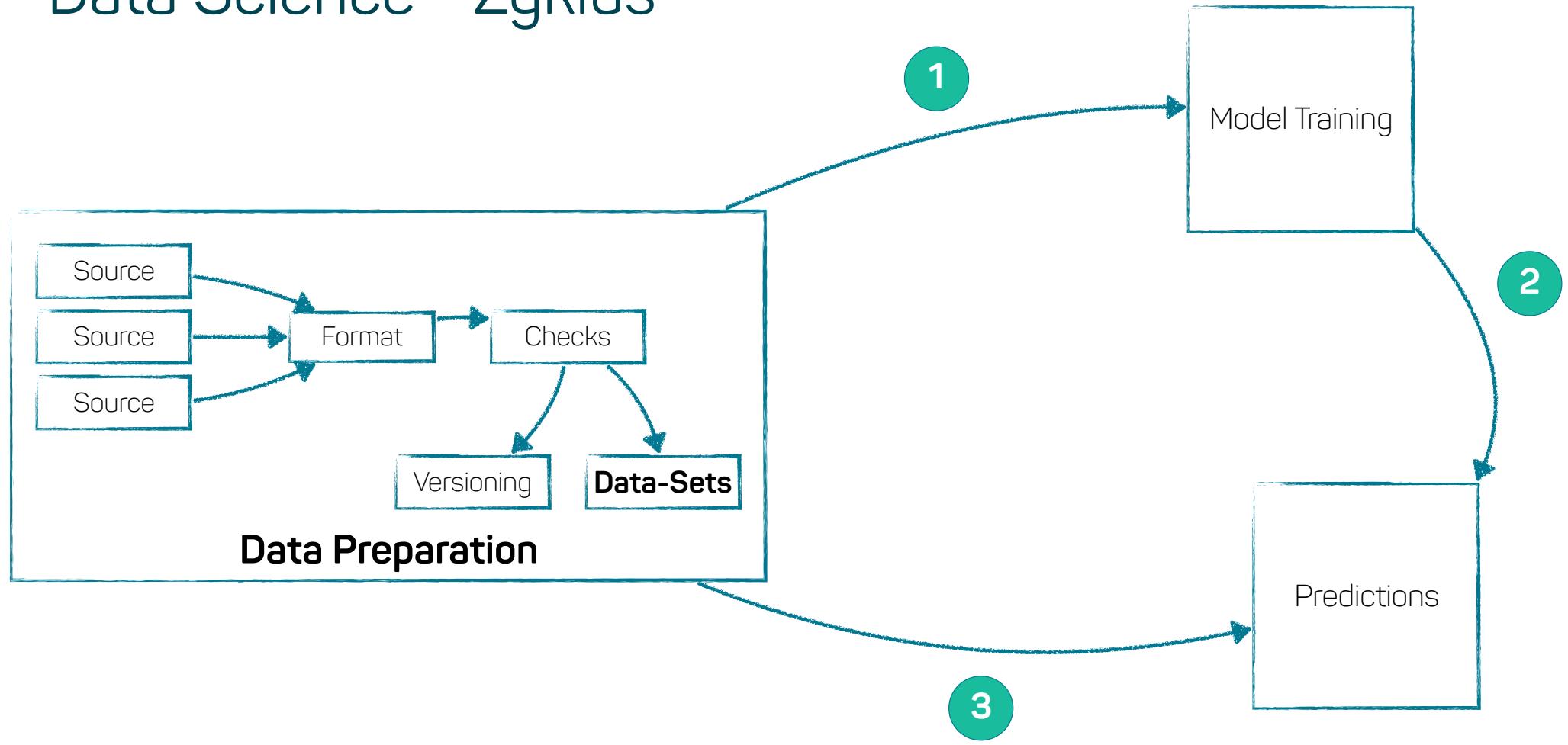
Data Science - Zyklus



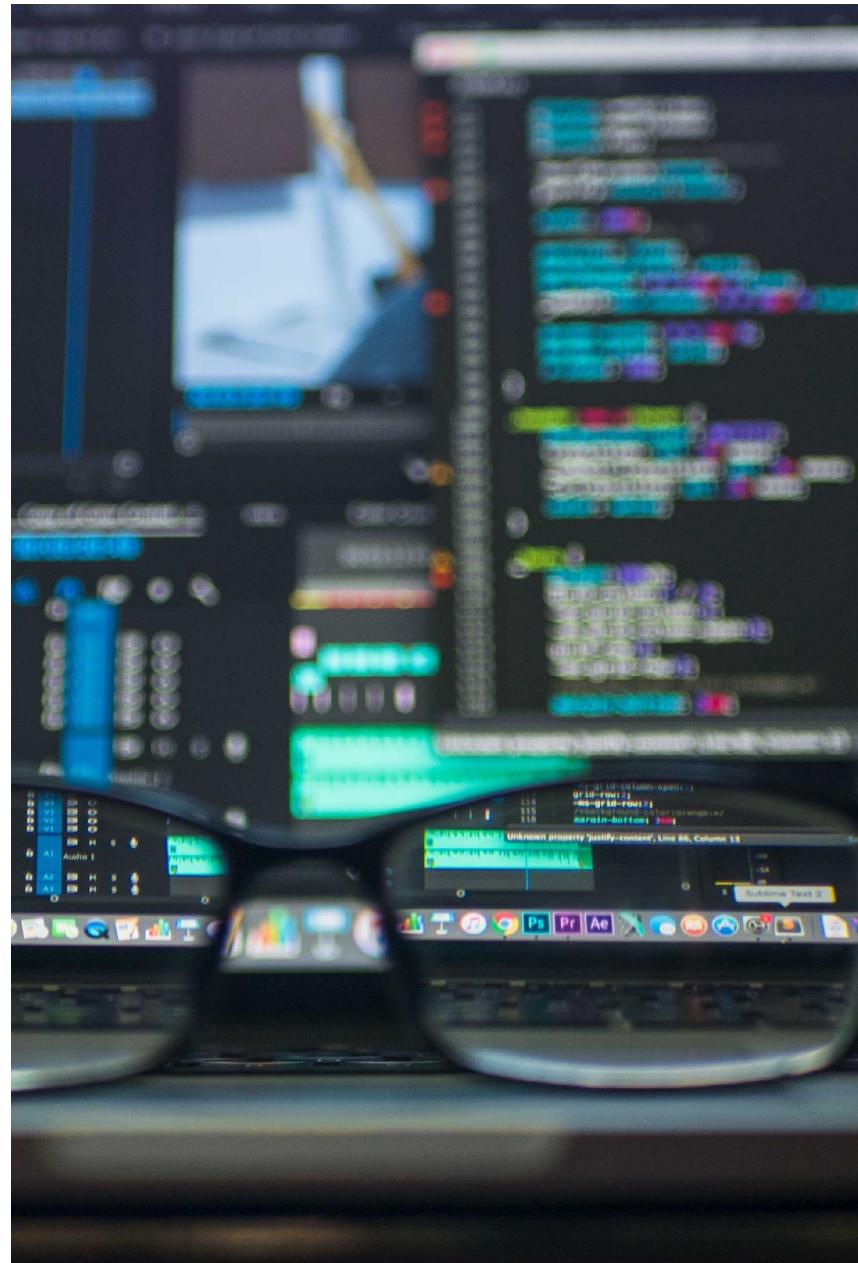
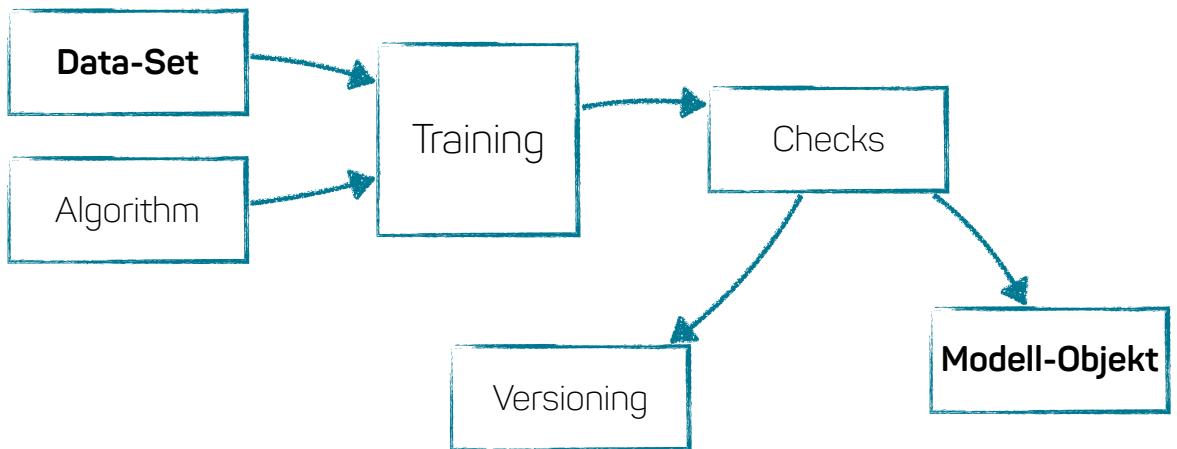
Vorbereiten der Daten



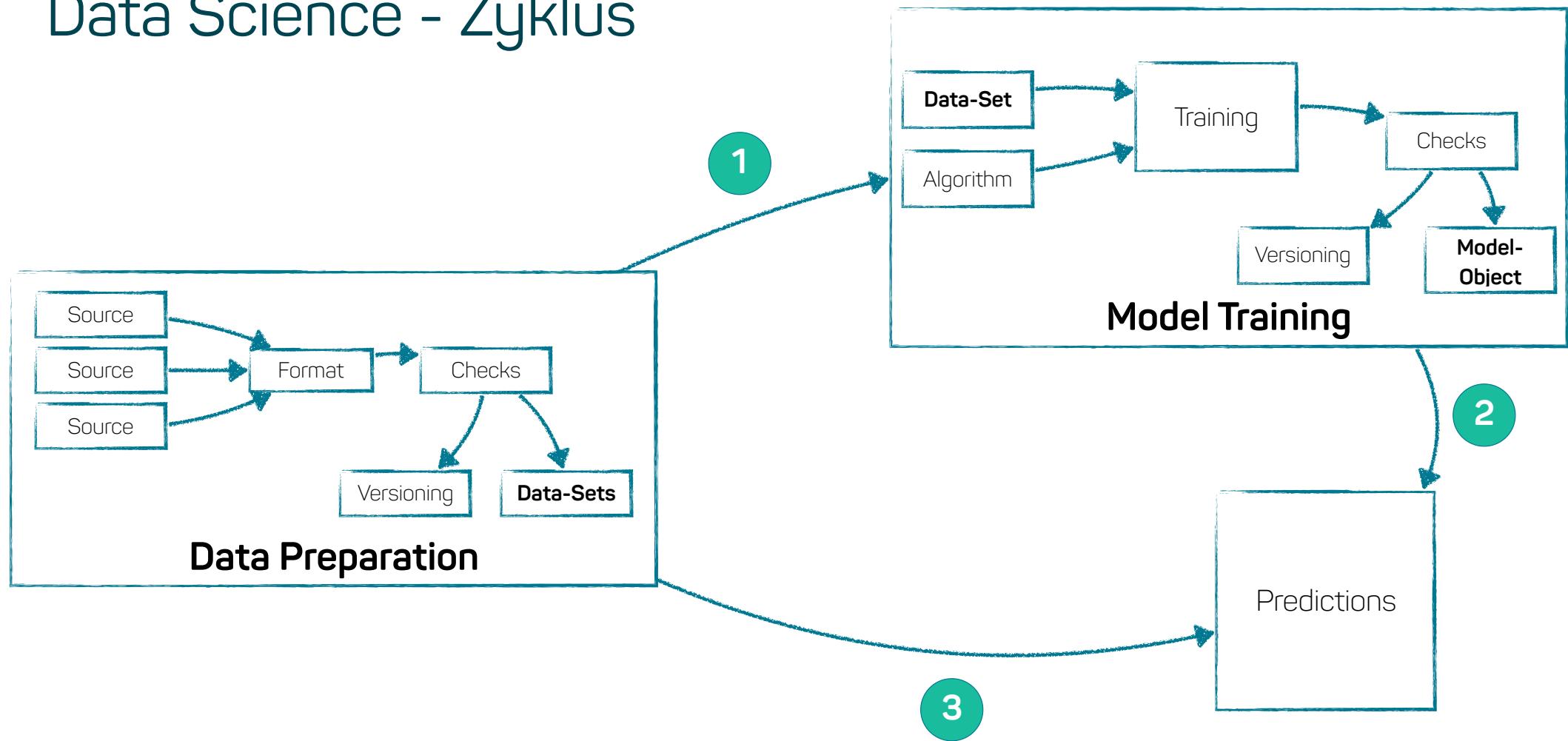
Data Science - Zyklus



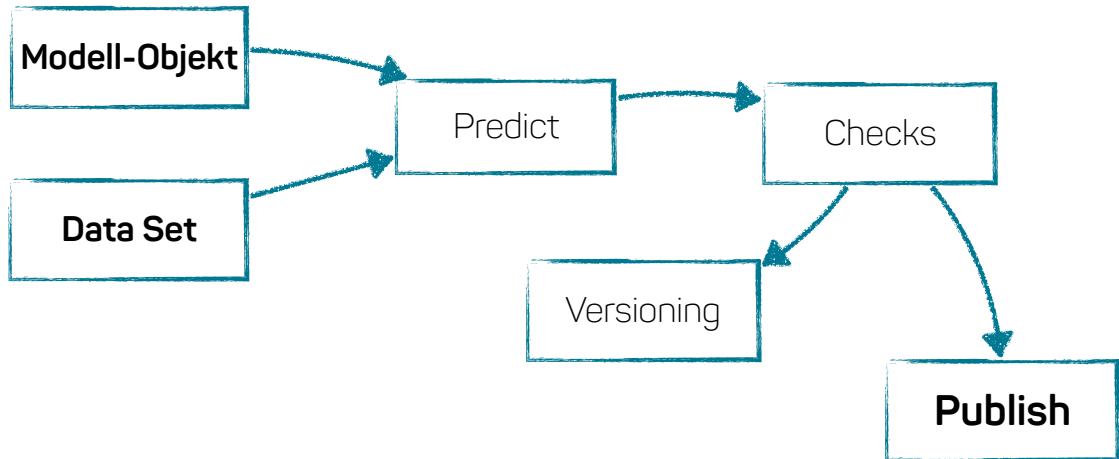
Trainieren des Modells



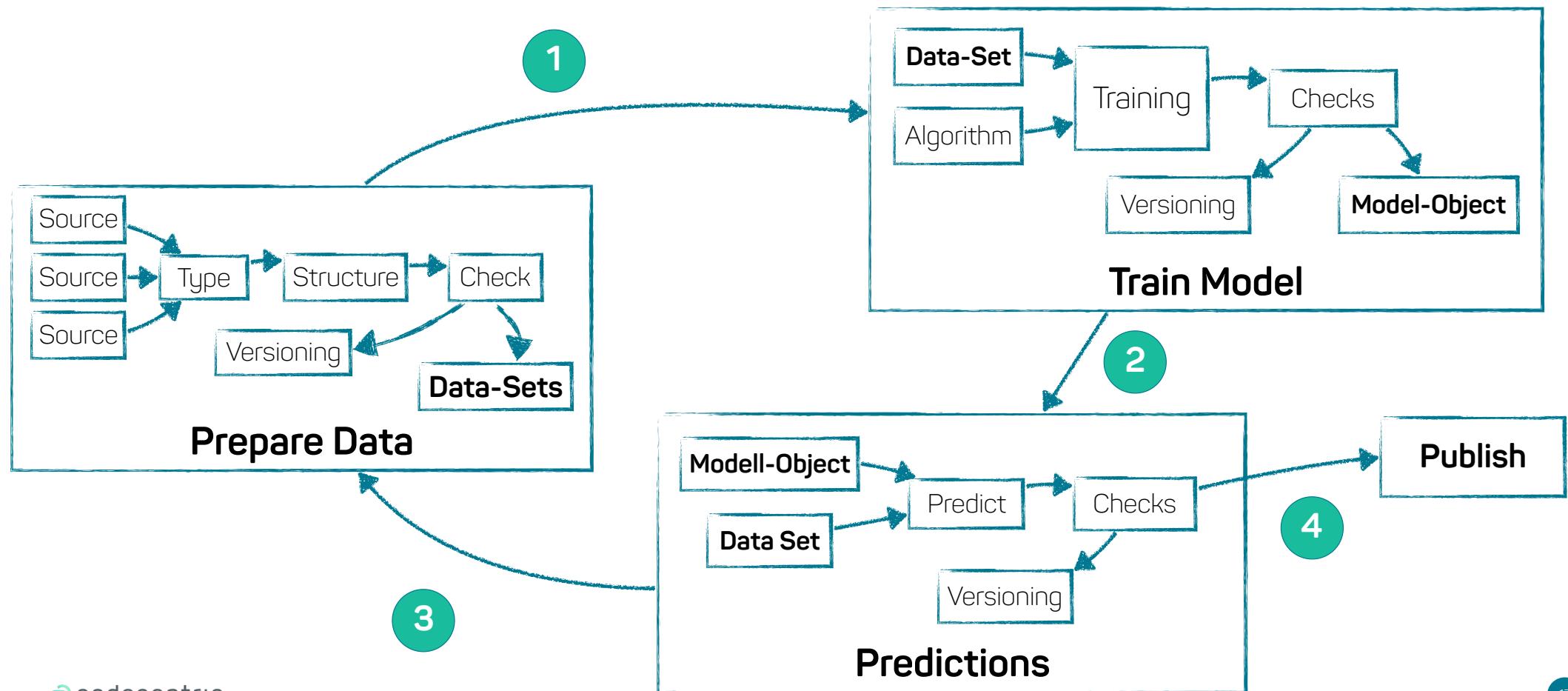
Data Science - Zyklus



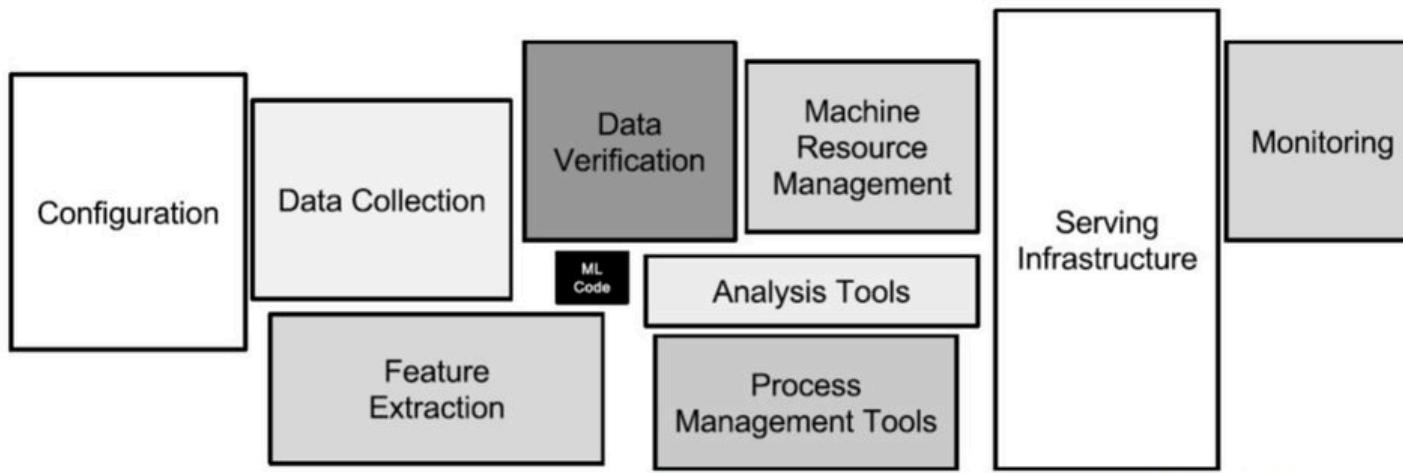
Forecasting



ML-Systeme



ML-Systeme sind komplex

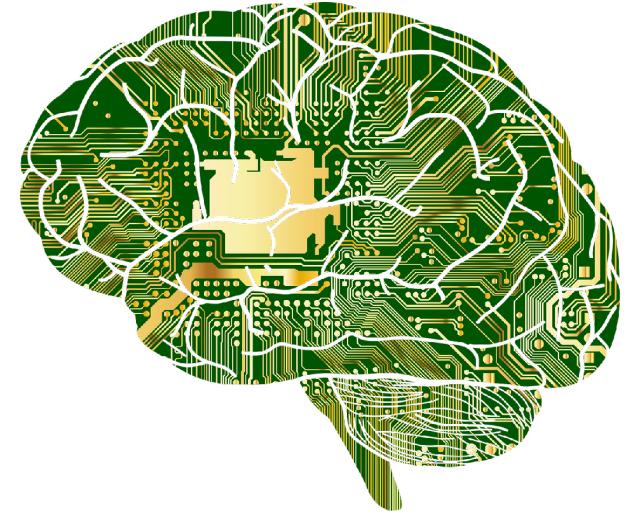


Sculley, David (Google), et al. "Hidden technical debt in machine learning systems." *Advances in neural information processing systems*. 2015.

ML-Systeme

Ziel: Automatisiertes System, um aus Daten Business Value zu generieren

- Besteht aus Data-Science und Data-Engineering Anteilen
- Beinhaltet eine Machine Learning Komponente
- Ist ein Software System mit "Intelligenz"
- Verfügen über einen Continuous Improvement Cycle
- Sind Software Systeme





Machine Learning Deployment

ML Deployment - Das Modell

Iterativer Prozess



Das Modell



ONNX - Open Neural Network Exchange

Ziel: Alle Modelle in einheitlicher Laufzeitumgebung

- Open Source Standard für ML Modelle
- Bietet eine einheitliche Representation für Modelle
- Wird von großen Firmen unterstützt



NVIDIA.

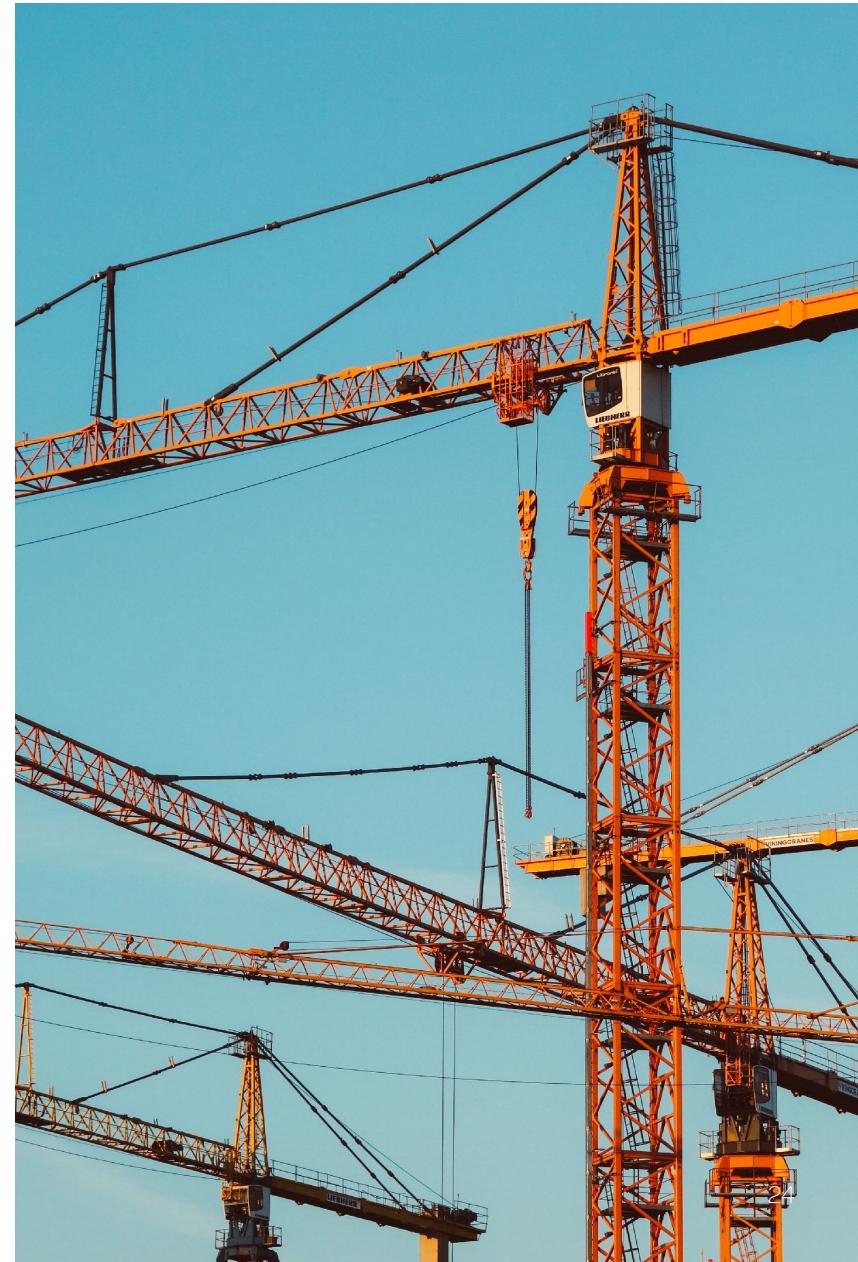


Microsoft



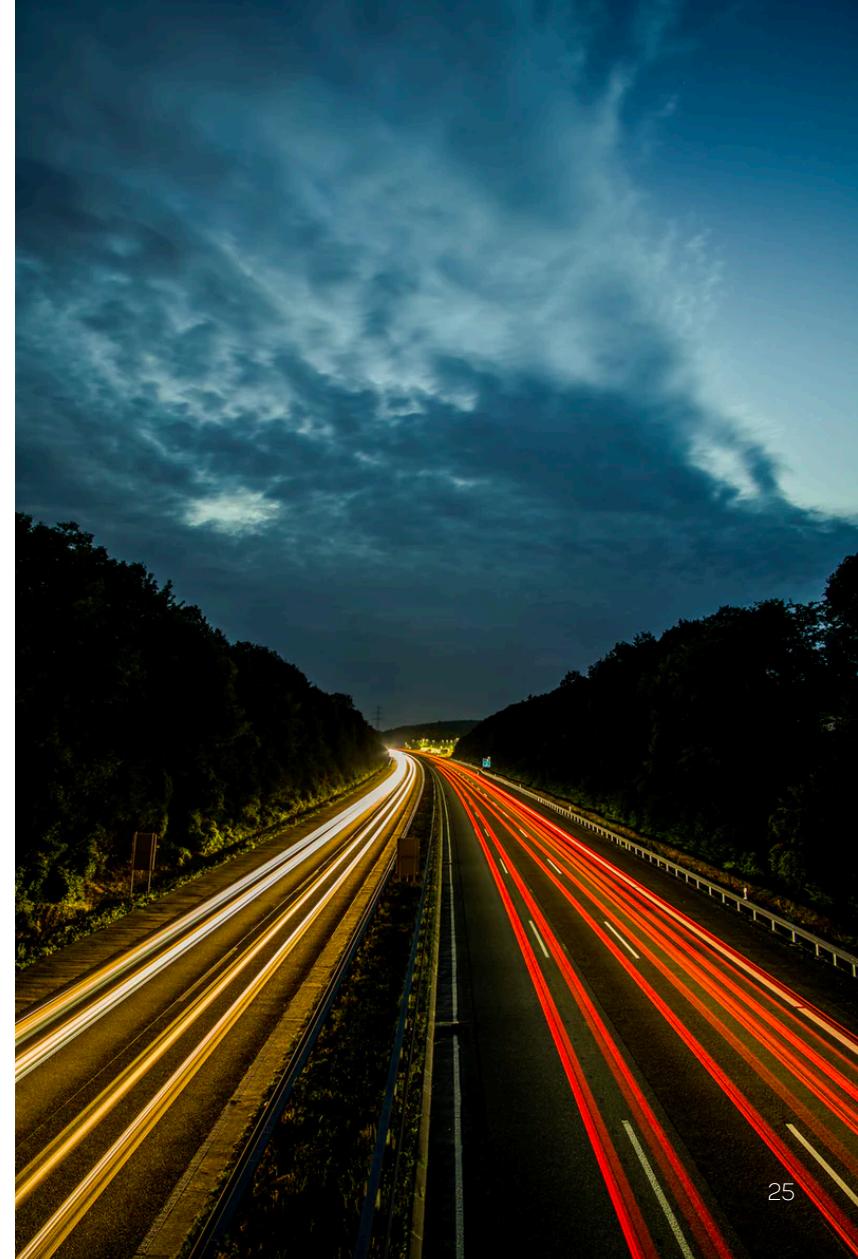
Deployment - Runtime

- Deployment von Code -> Docker
 - Deployment von Containern (K8s)
 - TensorFlow Serving
 - AWS SageMaker
 - Google Cloud ML Engine
 - Azure ML Service
 - FaaS
-
- Inference != Training



Deployment - Batch vs Online

- Batch: Speichern & Exportieren der Resultate zur weiteren Verarbeitung
 - Langsameres Feedback
 - Einfacher zu implementieren
- Online: API, PubSub, Datenbank
 - Schnelle Reaktionszeiten
 - Häufig einhergehend: kürzere Feedbackzyklen
 - Aber auch: komplexere Evaluierung
- Außerdem zu beachten:
 - Skalierung
 - Infrastruktur / Monitoring
 - Kosten
 - Komplexität des Modells



Deployment - Scheduling & Orchestrierung

- Scheduling -> Airflow/Luigi
 - Tool zum kontinuierlichen Ausführen des Cycles
 - Löst Dependencies zwischen Tasks
- Orchestrierung bei
 - Feature Engineering
 - Hyperparametertuning



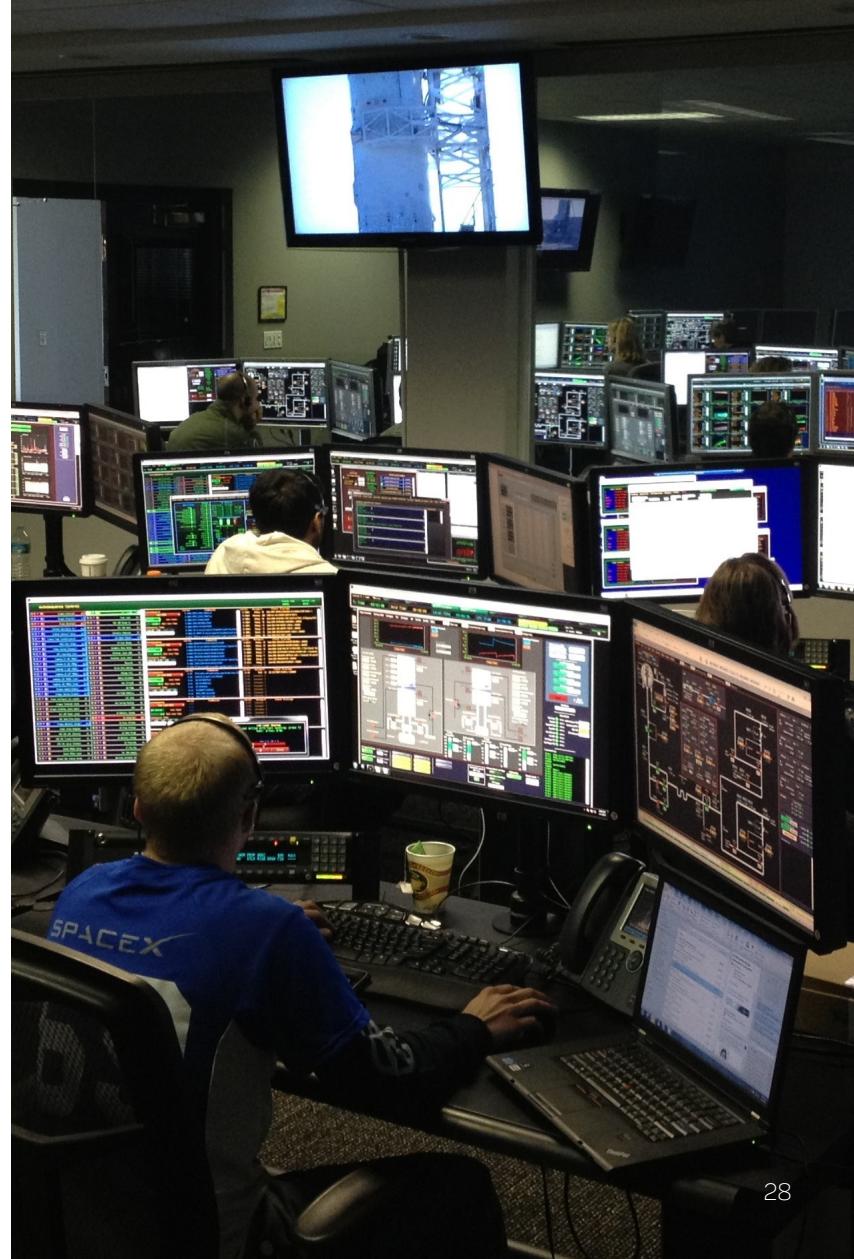
Deployment - Feedback

- Einfach: Scoring Evaluierung
 - Samples mitloggen / extrahieren
 - Metriken bereitstellen
- Challenge: Business Feedback für Backfeeding bekommen
 - Feedback häufig erst lange nach der Prediction möglich
 - Feedbackkanal häufig nicht vorgesehen

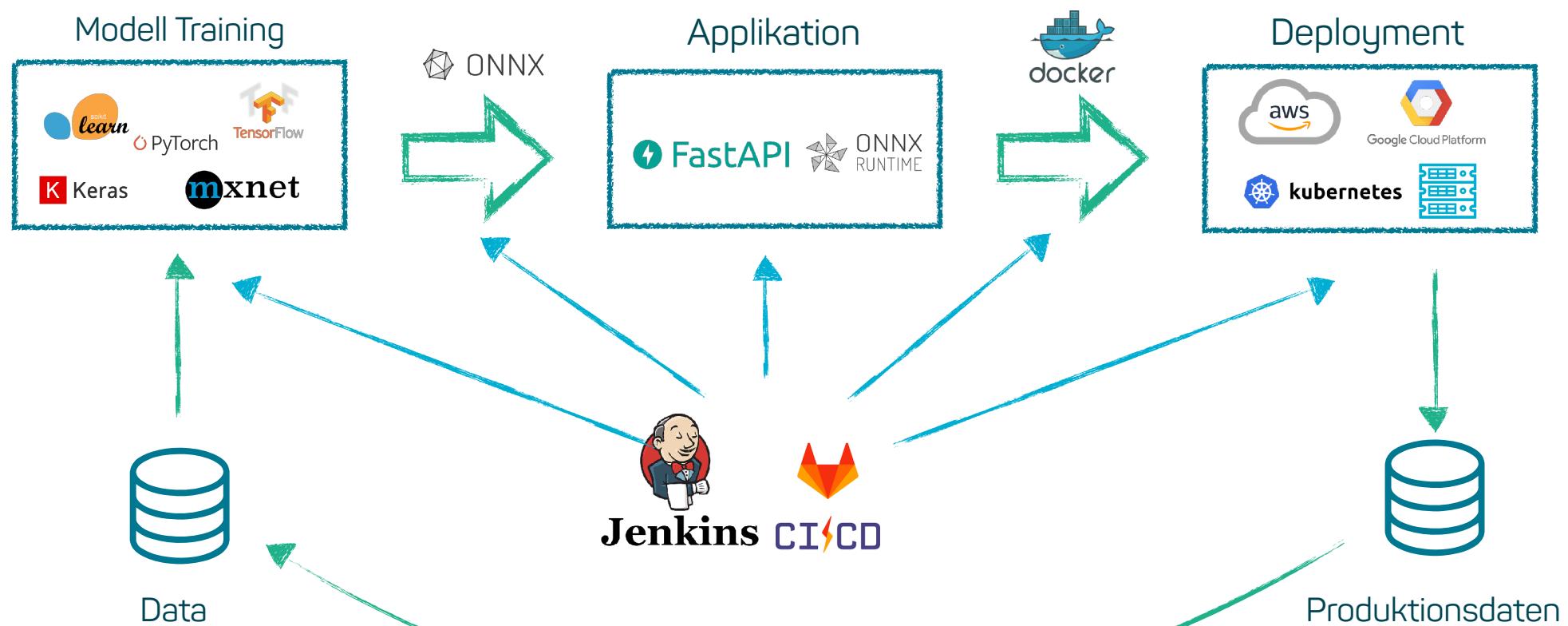


Monitoring

- Technisches Monitoring:
 - Laufen Predictions durch? Wie ist die Cycle-Time?
 - Wie ist die Auslastung der Ressourcen: CPU/RAM/IO
- Fachliches Monitoring:
 - Verteilung der Klassen innerhalb der Predictions
 - Grundverteilung der Input Daten
- Nachträgliche Evaluierung:
 - A/B Testing: Aufteilung in unterschiedliche Gruppen
 - Post-Eval: Zurückschauen in die Vergangenheit auf alte Prognose und mittlerweile bekannte Ground-Truth



ML Deployment - Applikationssicht



Weitere hilfreiche Tools

Data Version Control / Version Control



ML Deployments



Beispiel aus der Praxis - Ausgangslage



Churn Prediction, Recommender



Erste Modelle mittels R implementiert



Komplett manueller Prozess



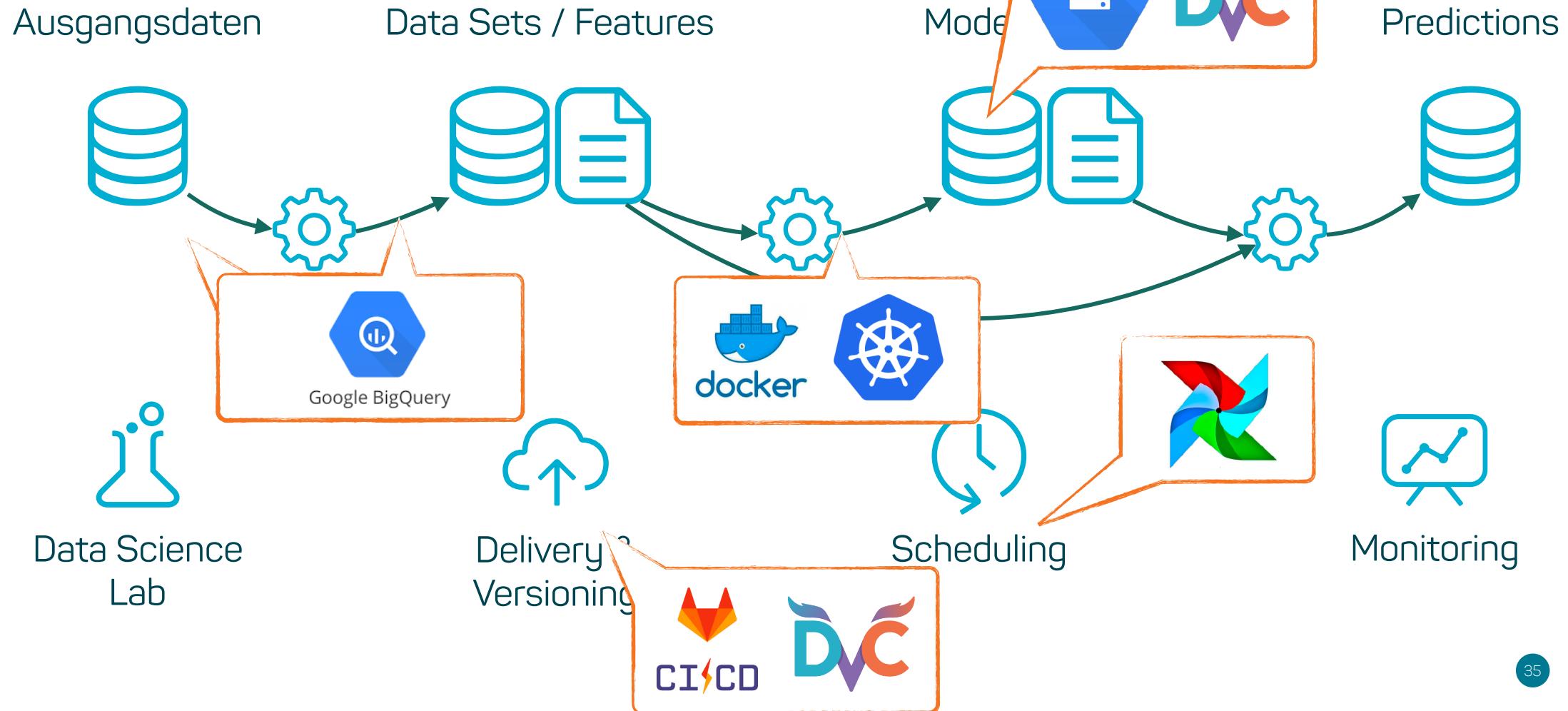
Langwierig und deshalb wenig genutzt



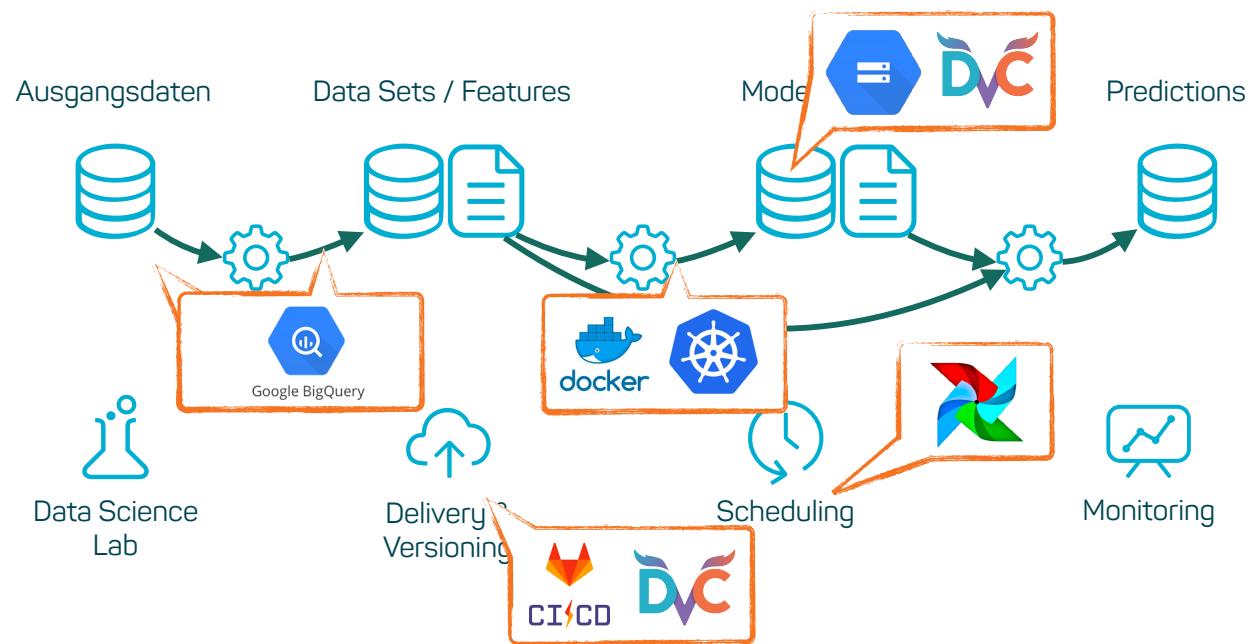
Skalierung nicht möglich

- Technisch (lokaler Rechner, on-premises)
- Business (Ausrollen in mehrere, unabhängige Regionen)

Beispiel aus der Praxis



Beispiel aus der Praxis



Eine mögliche technische Lösung unter Berücksichtigung von

- Vorhandener Infrastruktur
- Vorgaben bzgl. Cloud Nutzung
- Skills
- Flexibilität

- Viele spezialisierte Lösungen am Markt
- Erste All-in Lösungen on-Prem
- ML-Plattformen der großen Cloud Anbieter

Technische Herausforderungen



Experiment Tracking

Reproduzierbarkeit

Skalierung &
(lokale) Infrastruktur

Monitoring

Hohe Anzahl verschiedener
Komponenten

(noch) wenige Standards

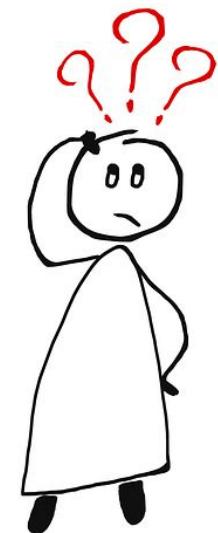
Integration in bestehende Umgebung

Datenschutz erhöht die Komplexität

PoC – Mindset

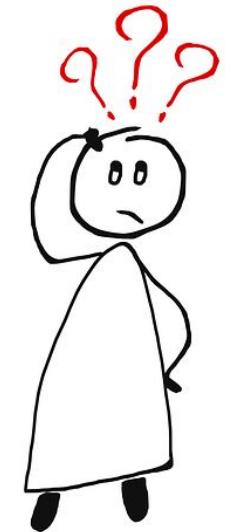
Aufgabe: Evaluierung der Durchführbarkeit

- Entwickle ein einfaches Basis Modell
- Treffe pragmatische Entscheidungen und Annahmen
- Schärfe den Use-Case / Business Case
- Helfe dem Management zu Verstehen (Business Metrics)
- Nur Experimente liefern belastbare Entscheidungsgrundlagen
- Arbeitet data-driven nicht discussion-driven



PoC – Mindset

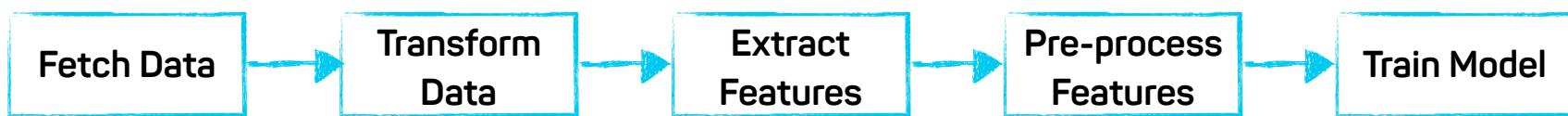
- Jupyter Notebooks sind am Anfang okay
- Reproduzierbarkeit und Vergleichbarkeit sind wichtig
- Beginne mit Automatisierung, Bash-Scripte sind okay
- Entwickle ein "ausreichend gutes" nicht "das beste" Modell
- Denke an das Big Picture
- Sobald ein Mehrwert geschaffen ist Integration



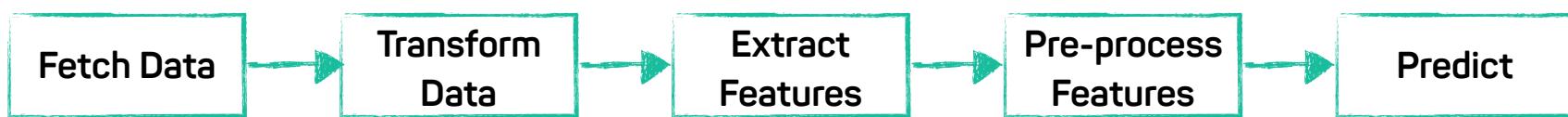
Minimal Valuable Product – Pipelines

Herausforderung: Deployment der gesamten Pipeline

Trainings-Phase



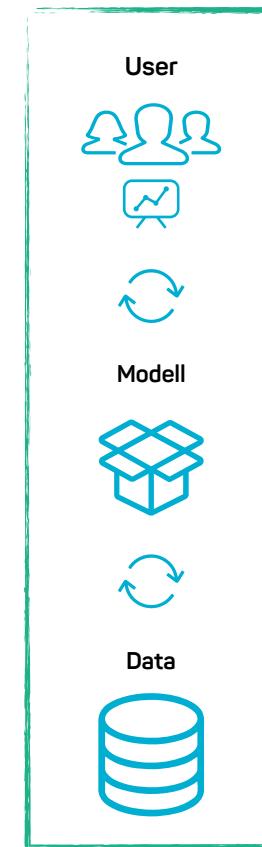
Prediction-Phase



Minimal Valuable Product

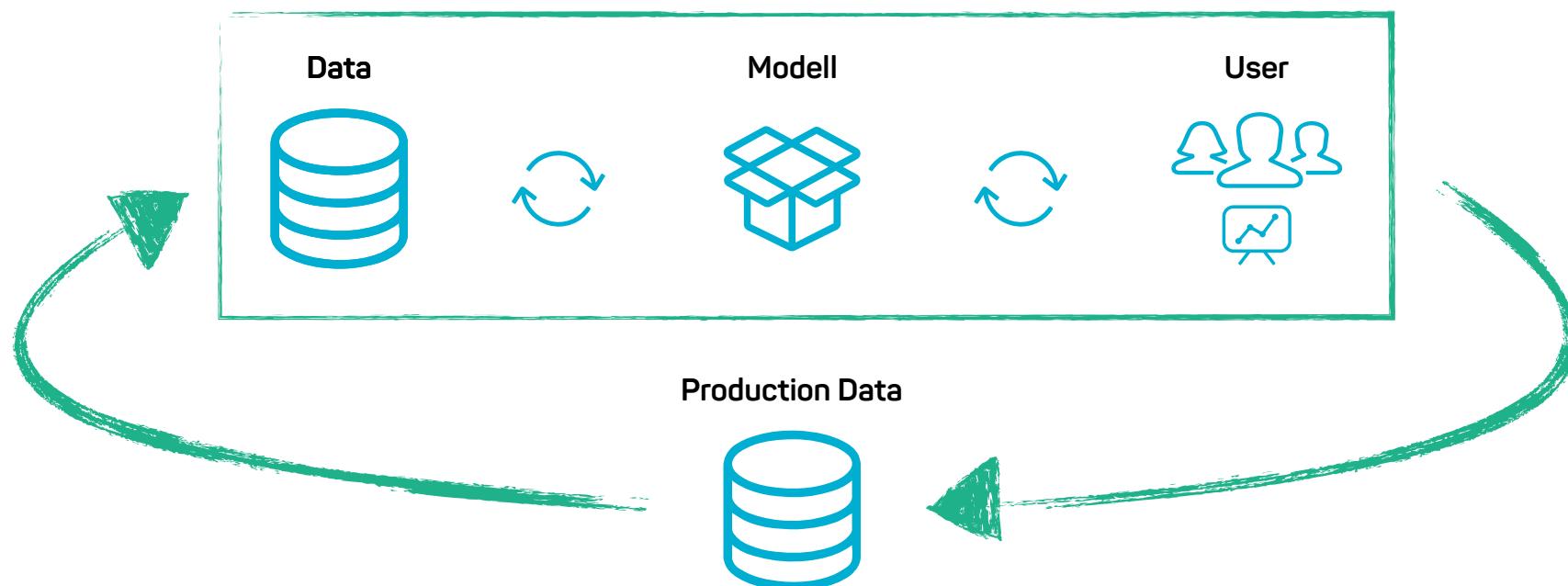
Aufgabe: Integration des Modells in die vorhandene Infrastruktur

- Baue "richtiges" Python aus den Jupyter Notebooks
- Schreibe Unitests und Integrationstests
- Entwickle eine initiale Deployment Infrastruktur
- Automatisiertes Training und Evaluation
- Es ist okay die Anwendung am User zu entwickeln



Professionalisierung des Systems

Ziel: Continuous Improvement Cycle und Monitoring



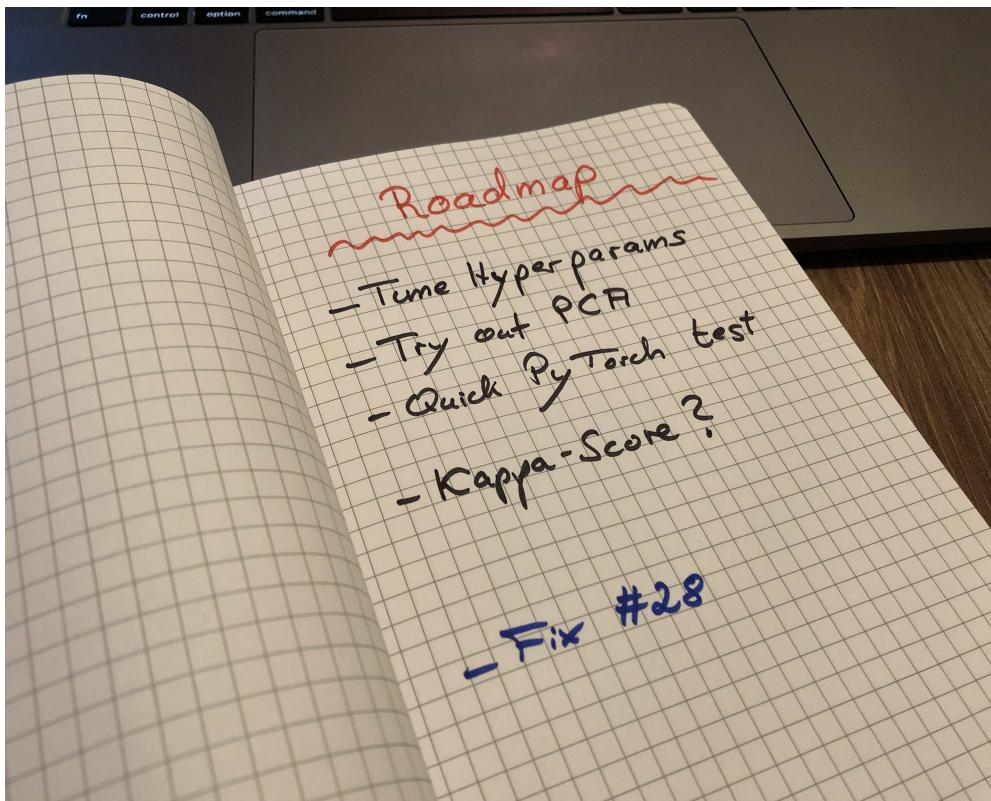
Take aways

- ML in Production ist schwer
 - Engineering & Science Perspektive
 - Komplexe ML-Systeme
 - Mehr Software & Engineering als vermutet
- „Klassische“ Lösungen aus der SE
- Vermehrt spezialisierte Lösungen bis hin zu Komplettansätzen
- Cloud kann die Fertigungstiefe verringern
- Vorgehen:
 - PoC schrittweise ausbauen
 - Nicht mit Plattform starten



Versioning and Reproducable Training

- Data Science is experiment-driven ...



A STORY TOLD IN FILE NAMES:				
Filename	Date Modified	Size	Type	
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file	
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file	
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file	
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file	
data_2010.05.28_huh???.dat	7:20 PM 5/28/2010	30 KB	DAT file	
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file	
data_2010.05.29_aaarrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file	
data_2010.05.29_#\$@*!!?.dat	2:40 AM 5/29/2010	0 KB	DAT file	
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file	
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file	
data_2010.05.29_woohoo!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file	
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file	
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file	
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file	
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file	
JUNK...	2:45 PM 5/29/2010		Folder	
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file	

Reproducible Training

- ... and collaborative!

```
Tims-MacBook-Pro:code tsabsch$ git log --graph --decorate --oneline
*   fc88d36 (HEAD -> master) Merge branch 'tim_fix_zerodivisionerror'
|\ \
| * 2cc23f0 (tim_fix_zerodivisionerror) Fix ZeroDivisionError
* |   e486d76 Merge branch 'mark_batchnorm'
|\ \
| / \
|/
| * b988f6e (mark_batchnorm) Decrease BN threshold
| * c83736d Add batch normalisation
* | f80b108 Add preprocessing
|/
* a87aa63 Add simple tensorflow model
* 7c0d21e Initial commit
Tims-MacBook-Pro:code tsabsch$
```



Modelmanagement - by hand - hyper parameters

A	B	C
Date	Alpha	L1_ratio
2019-04-03 1:00 PM	1	1
2019-04-03 4:00 PM	1	0.5
2019-04-04 9:00 AM	1	0.2
2019-04-04 11:00 AM	1	0

Modelmanagement - by hand - metrics

A	B	C	D	E	F
Date	Alpha	L1_ratio	mea	r2	mse
2019-04-03 1:00 PM	1	1	0.649	0.04	0.862
2019-04-03 4:00 PM	1	0.5	0.648	0.046	0.859
2019-04-04 9:00 AM	1	0.2	0.628	0.125	0.823
2019-04-04 11:00 AM	1	0	0.619	0.176	0.799

Modelmanagement - by hand - model

	A	B	C	D	E	F	G
1	Date	Model	Alpha	L1_ratio	mea	r2	mse
2	2019-04-03 1:00 PM	model_v7.p	1	1	0.649	0.04	0.862
3	2019-04-03 4:00 PM	model_v7_l1-05.p	1	0.5	0.648	0.046	0.859
4	2019-04-04 9:00 AM	model_v7_l1-02_d3.p	1	0.2	0.628	0.125	0.823
5	2019-04-04 11:00 AM	model_v7_l1-zero_d3.p	1	0	0.619	0.176	0.799

Modelmanagement - by hand - data

	A	B	C	D	E	F	G	H
1	Date	Dataset	Model	Alpha	L1_ratio	mea	r2	mse
2	2019-04-03 1:00 PM	data_v2	model_v7.p	1	1	0.649	0.04	0.862
3	2019-04-03 4:00 PM	data_v2	model_v7_l1-05.p	1	0.5	0.648	0.046	0.859
4	2019-04-04 9:00 AM	data_v2_upd_May	model_v7_l1-02_d3.p	1	0.2	0.628	0.125	0.823
5	2019-04-04 11:00 AM	data_v2_upd_May	model_v7_l1-zero_d3.p	1	0	0.619	0.176	0.799

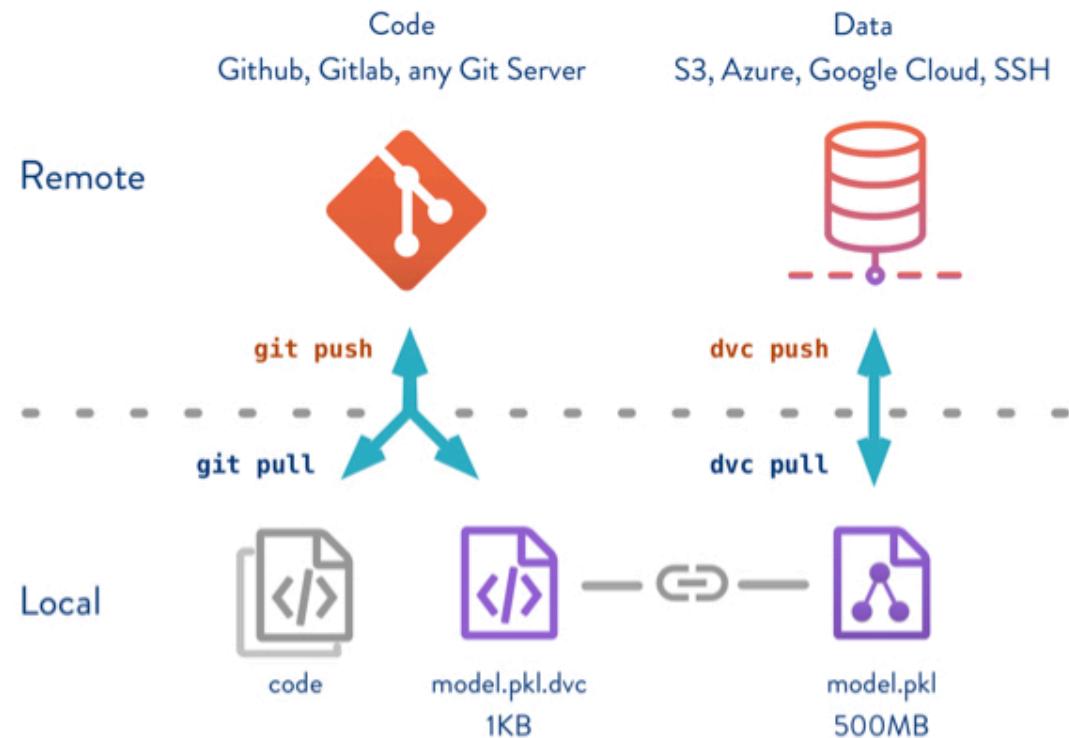
Modelmanagement - by hand - code

A	B	C	D	E	F	G
Date	Dataset	Model	Commit	Alpha	L1_ratio	mea
2019-04-03 1:00 PM	data_v2	model_v7.p	9a5a01b	1	1	0.6
2019-04-03 4:00 PM	data_v2	model_v7_l1-05.p	9a5a01b	1	0.5	0.6
2019-04-04 9:00 AM	data_v2_upd_May	model_v7_l1-02_d3.p	9a5a01b	1	0.2	0.6
2019-04-04 11:00 AM	data_v2_upd_May	model_v7_l1-zero_d3.p	c8b2c10	1	0	0.6



DVC to the Rescue!

- **Large File Storage**, similar to git-lfs
- Metadata in git (name, hashsum etc.)
- Contents in cache + remote storage



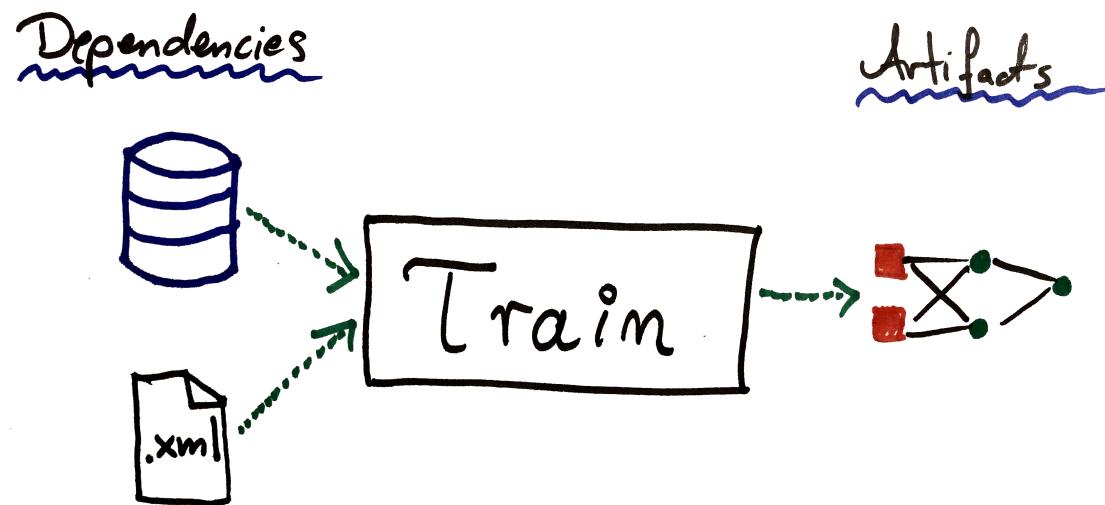
<https://dvc.org>



DVC to the Rescue!

- **Pipelines**

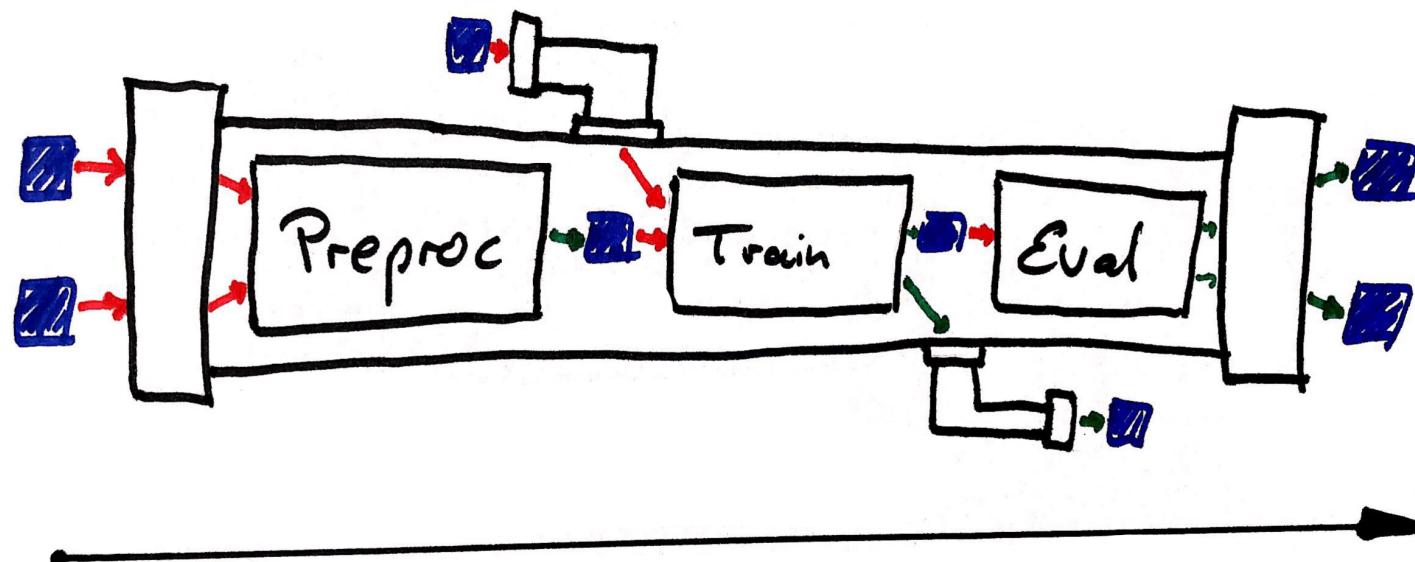
- Optimize reproduction by splitting your process into stages
- Stages store dependencies, processing and output artefacts



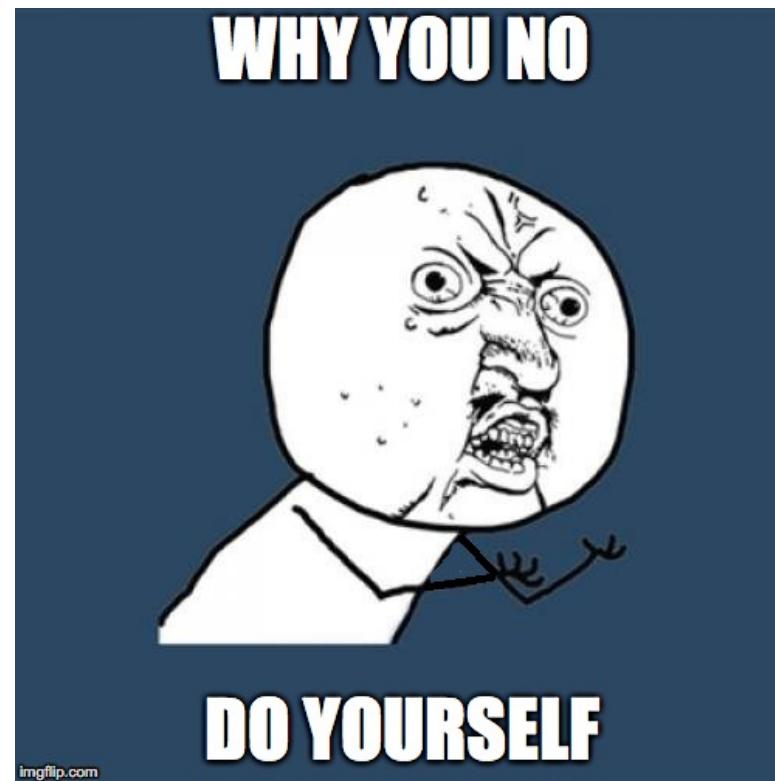


DVC to the Rescue!

- **Pipelines**
- Connect stages to a pipelines via dependencies



Let's get Practical: DVC



Deployment - Scheduling & Orchestrierung

- Scheduling -> Airflow/Luigi
 - Tool for continuous execution of the cycle
 - Solves dependencies between tasks
- Orchestration with
 - Feature engineering
 - Hyperparameter tuning

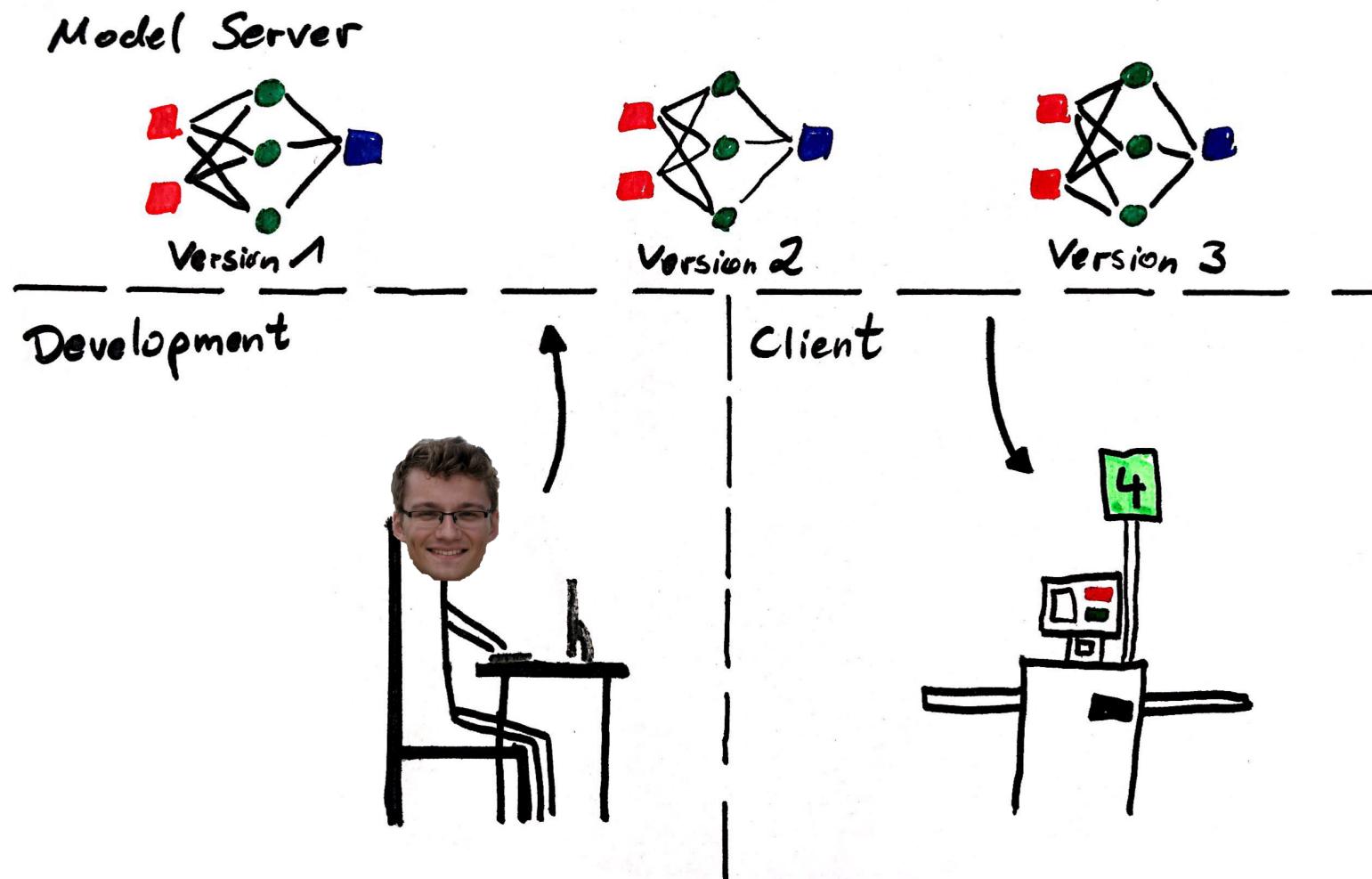


Model Serving with Tensorflow Serving

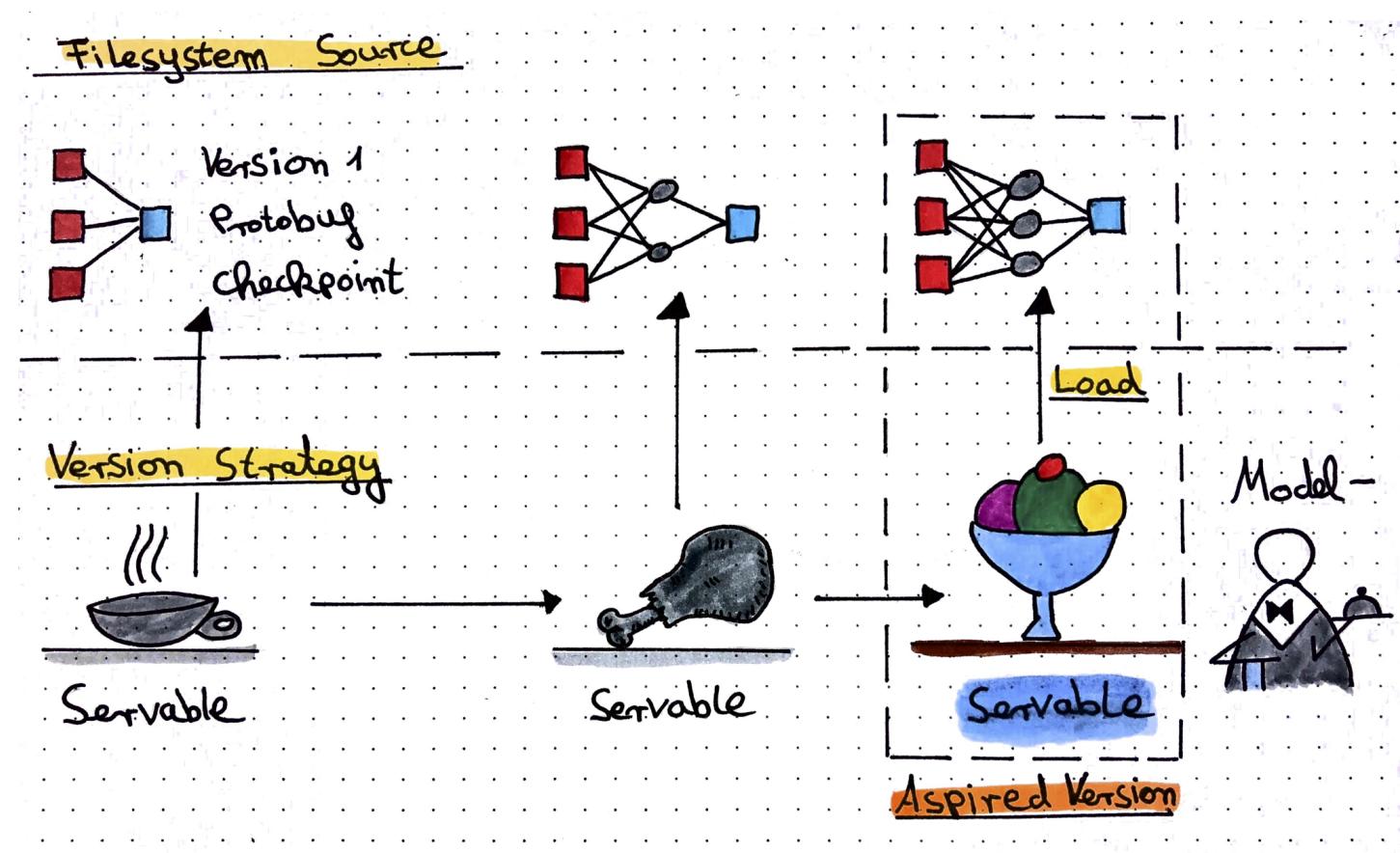
- Lightweight model server for Tensorflow graphs
- Useful for
 - Distributed model applications
 - Fast and continuous delivery
 - Model versioning



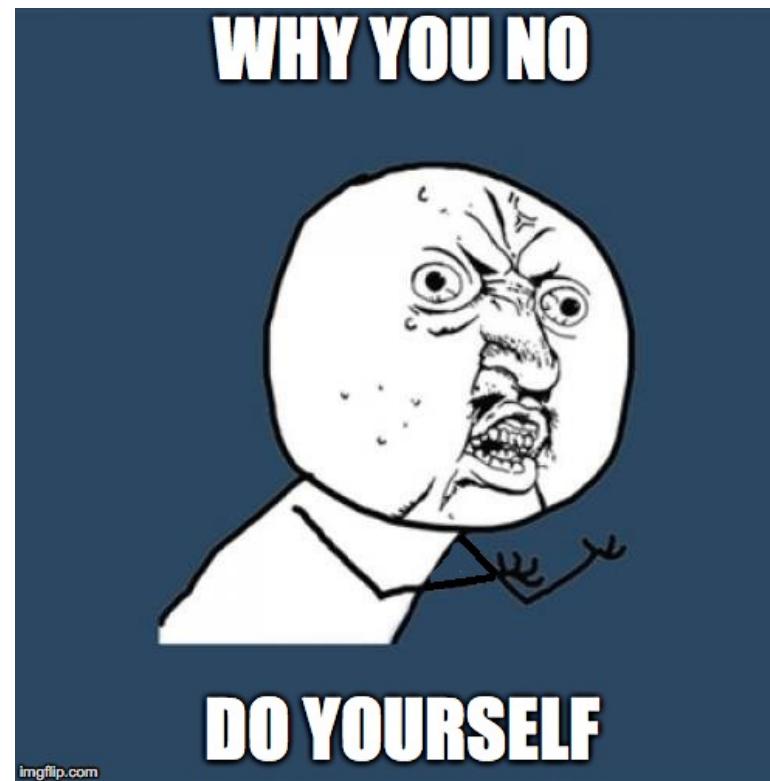
Tensorflow Serving



TensorFlow Serving in Detail



Let's get Practical: Tensorflow Serving



Preprocessing and the Deployment

- Preprocessing and model are tightly coupled
- How can the client "be sure" to do the right thing?

Possible solutions

- The client just knows
- Wrapper service around model and expose API
- Integrate preprocessing into the model

Preprocessing and the Deployment

```
import tensorflow as tf
from keras.layers import Lambda

def preprocess(x):
    resized = tf.image.resize_images(x, size=[100, 100])
    max_color = tf.constant(255, dtype=tf.float32)
    rescaled = tf.divide(resized, max_color)
    return rescaled

additional = Lambda(preprocess, output_shape=(100, 100, 3), name="preprocessing")
```

Preprocessing and the Deployment

```
from keras.models import Sequential
from keras.layers import InputLayer

prod_model = Sequential()
prod_model.add(InputLayer(input_shape=(None, None, 3)))
prod_model.add(additional)

for layer in model.layers:
    prod_model.add(layer)
```