Hochschule Konstanz
Technik, Wirtschaft und Gestaltung

Machine Learning – Life Cycle Tools

Benjamin Wagner

Gliederung

- **MLOps**
- **ML Prozessmodelle**
  - Crips DM
  - Modell nach Google
- **Komponenten eines vollständigen MLOps Systems**
- **Technologie-Stack**
  - Single Solutions
  - Integrated Solutions

MLOps

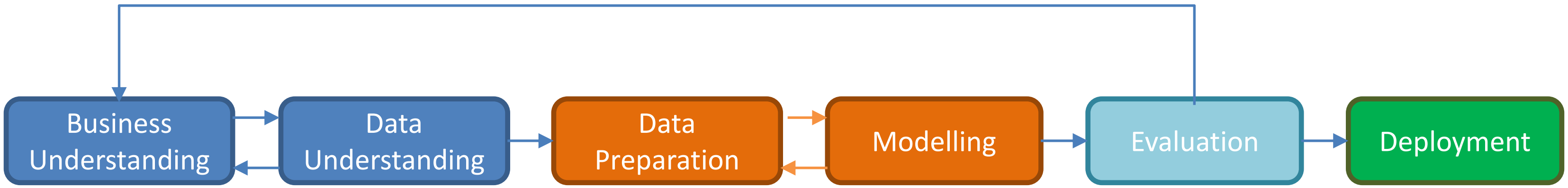
Anwendung der DevOps Prinzipien auf Machine Learning Systeme

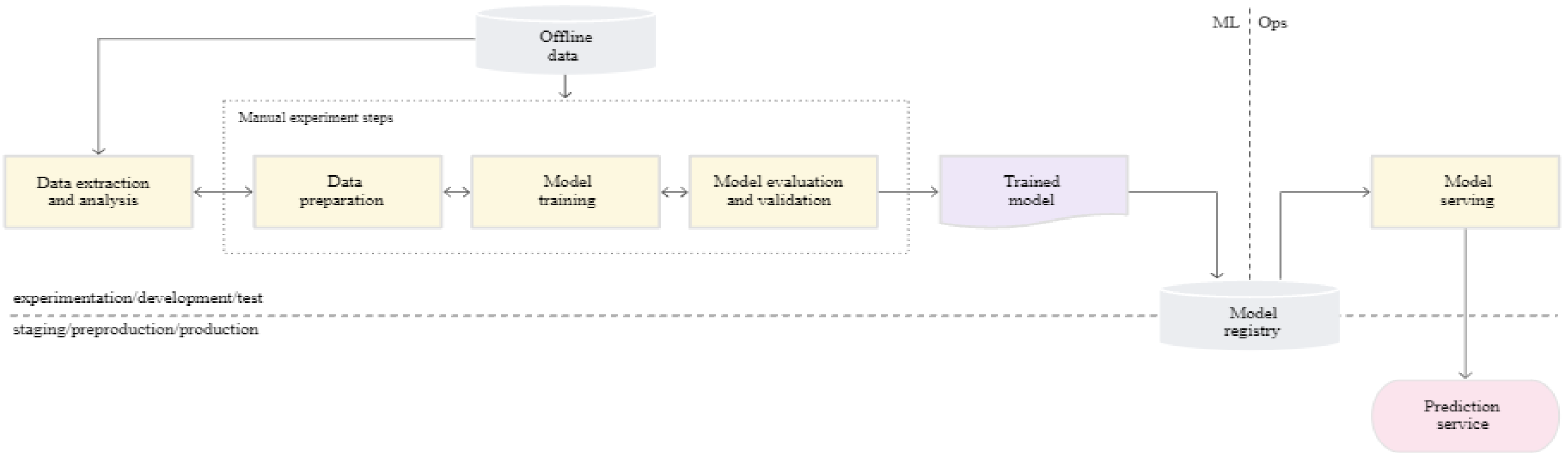
- Reproduzierbarkeit
- Skalierbarkeit
- Automation
- Deployment
- Monitoring und Management
- …

ML Prozessmodelle

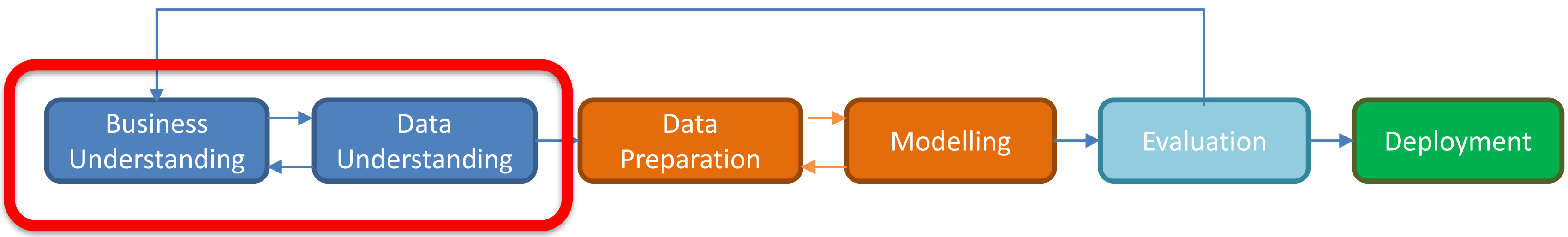
Crisp DM / Google

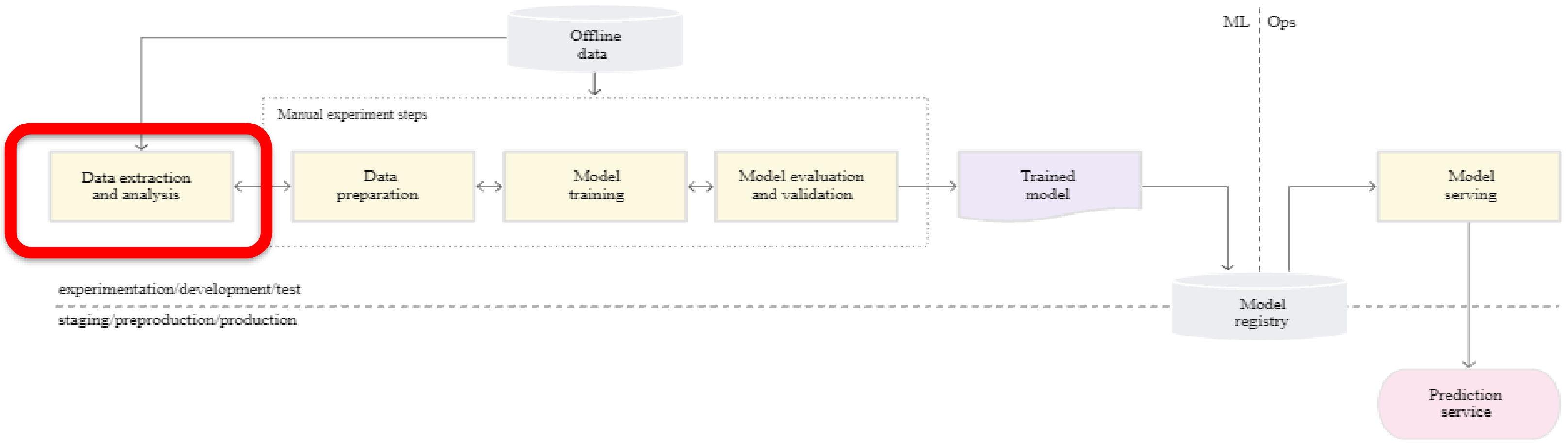
Hochschule Konstanz
Technik, Wirtschaft und Gestaltung



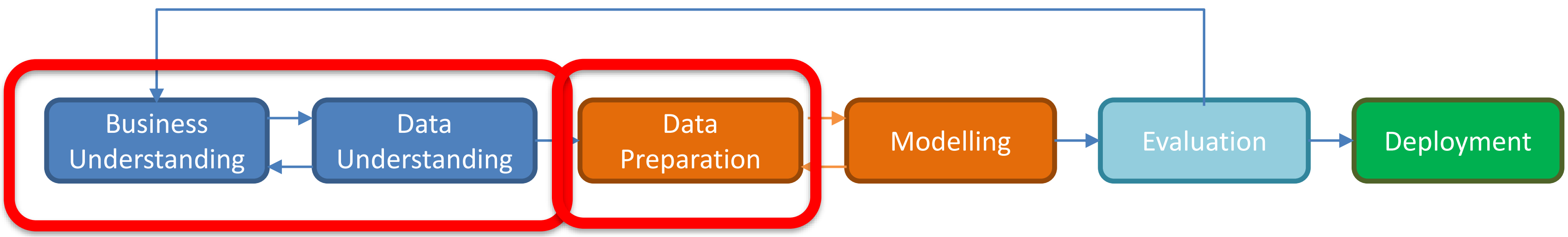


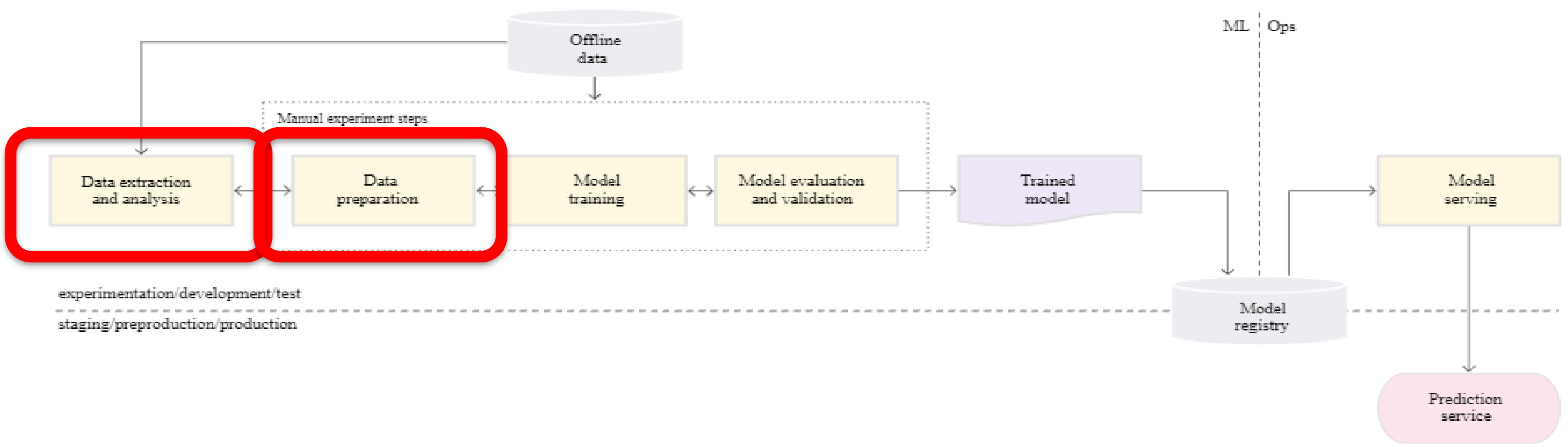
Crisp DM / Google



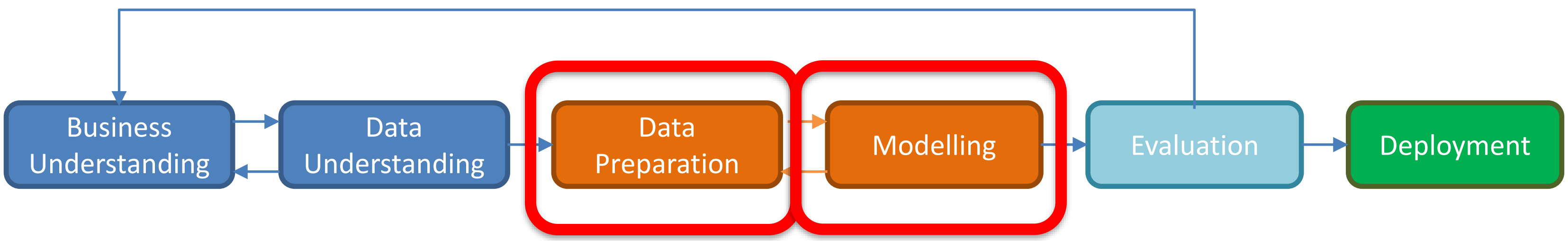


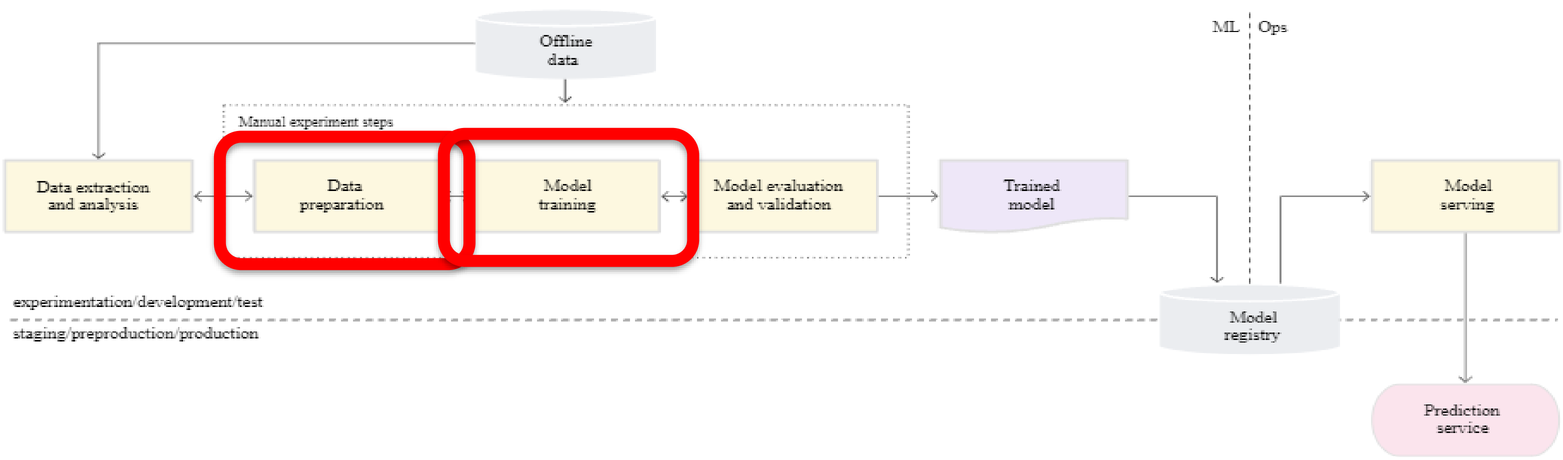
Crisp DM / Google





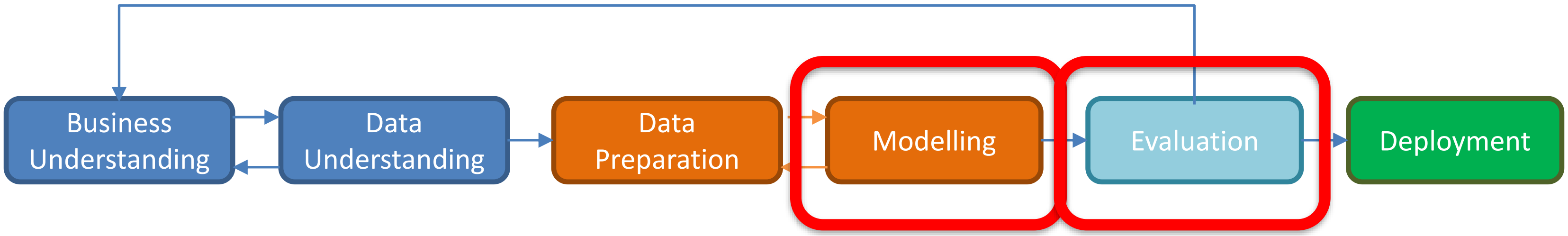
Crisp DM / Google

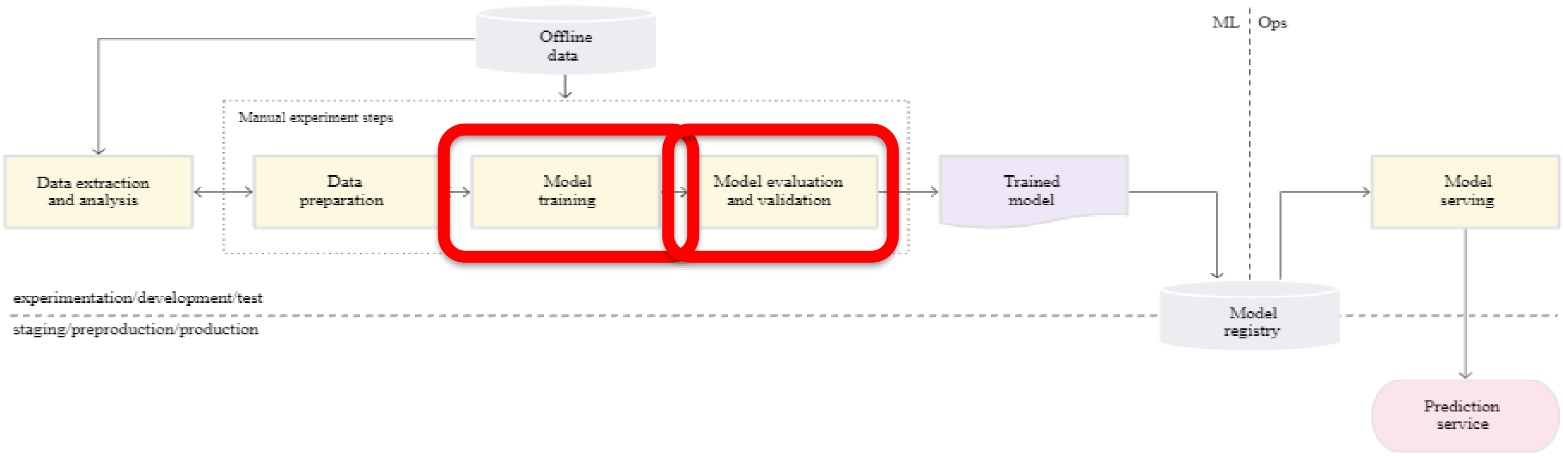




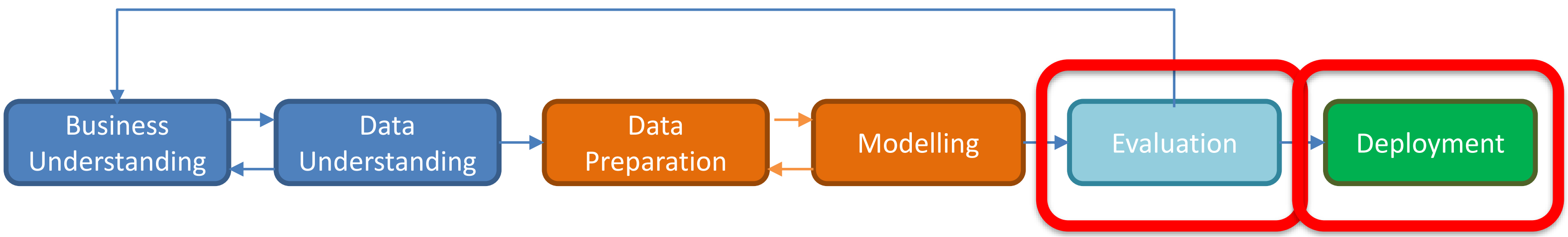
Crisp DM / Google

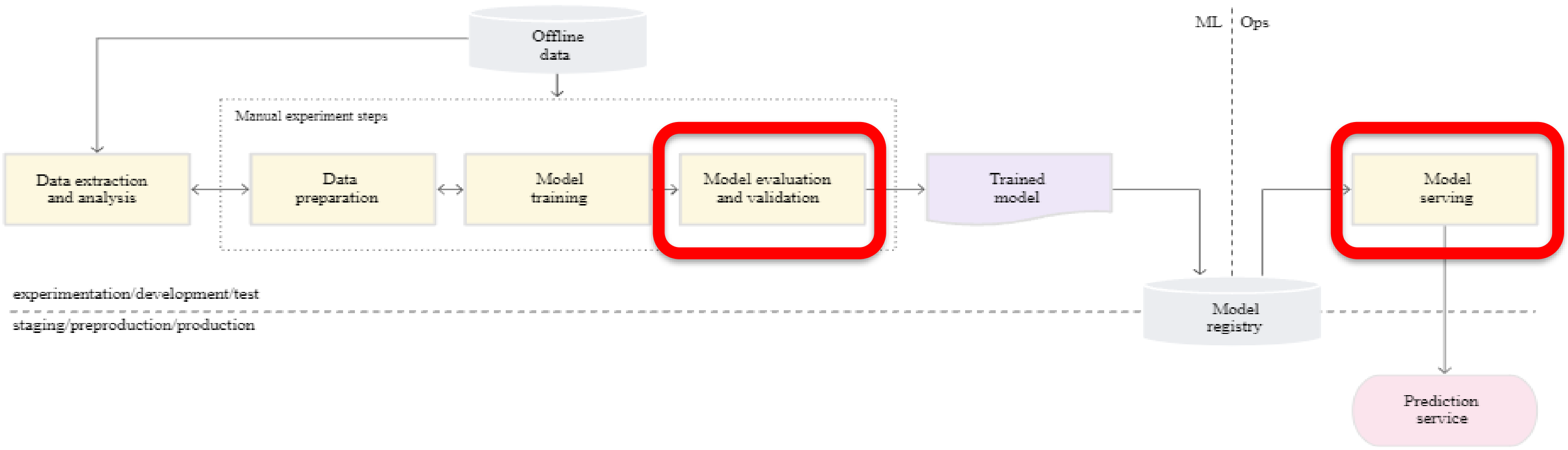
Hochschule Konstanz
Technik, Wirtschaft und Gestaltung





Crisp DM / Google





Komponenten eines Vollständigen MLOps System

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Technologie-Stack

Single Solutions

Metadata management
- ML Flow
- Neptune
- Comet

Hyperparameter Tuning
- Optuna
- SigOPT

Monitoring
- Fiddler
- Amazon Sage Maker

ML code

Versioning
- Pachyderm
- Apache Airflow
- DVC
- GIT

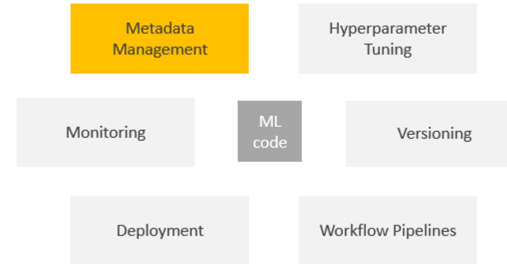
Deployment
- BentoML
- Cortex

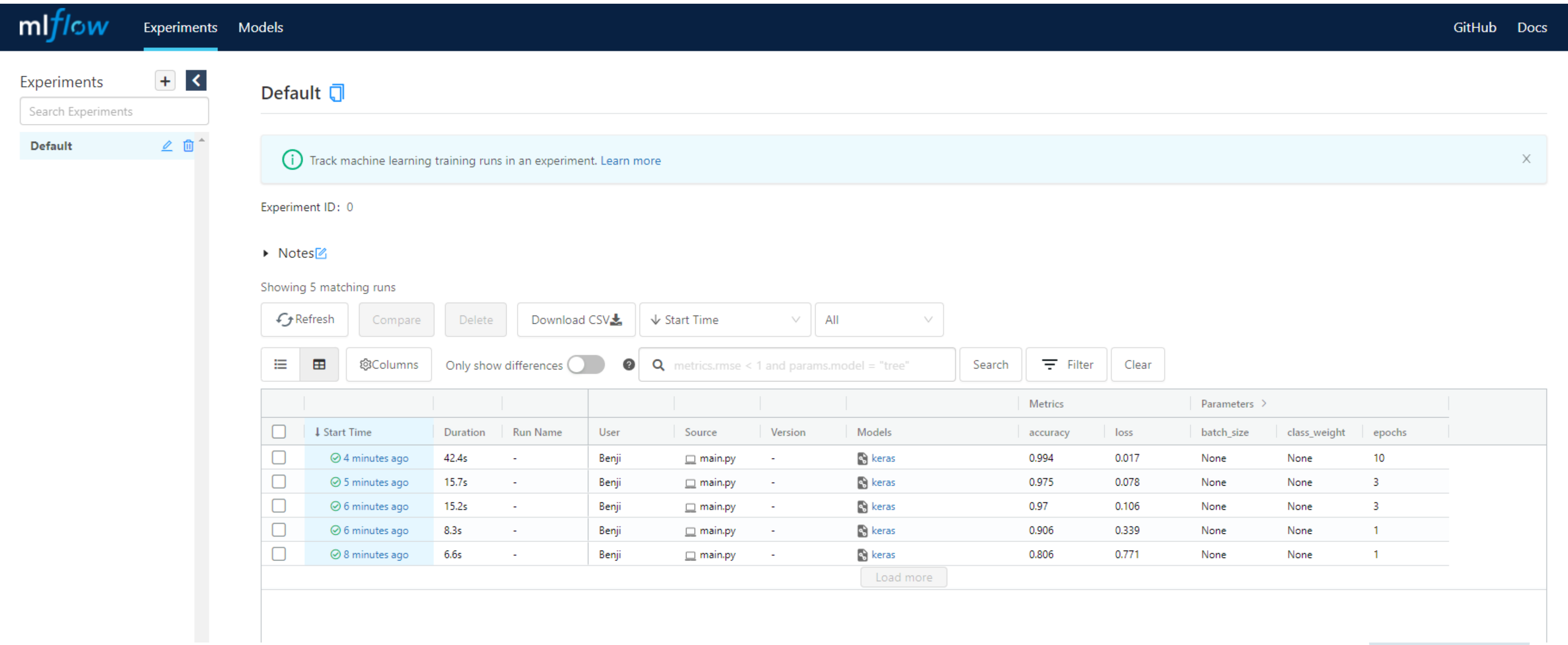
Workflow Pipelines
- Kubeflow
- Polyaxon

Metadata Management

ML Flow

ML Flow
Metadata Management





ML Flow
Metadata Management

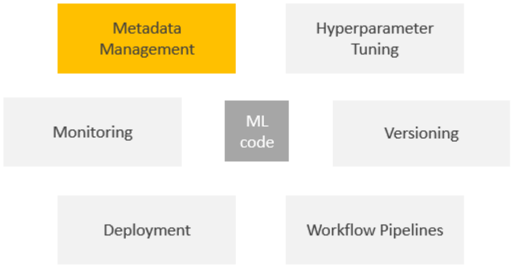
Metadata Management | Hyperparameter Tuning | Monitoring | ML code | Versioning | Deployment | Workflow Pipelines

Metrics

Name	Value
accuracy	0.994
loss	0.017

ML Flow

Metadata Management



Environment Dependencies

```
channels:
- conda-forge
dependencies:
- python=3.9.5
- pip
- pip:
  - mlflow
  - keras==2.6.0
  - pillow==8.4.0
  - scipy==1.7.1
  - tensorflow==2.6.0
name: mlflow-env
```

ML Flow
Metadata Management

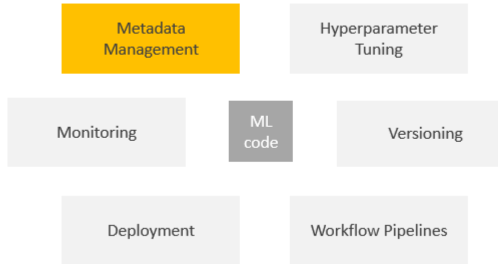
Metadata Management
Hyperparameter Tuning
Monitoring
ML code
Versioning
Deployment
Workflow Pipelines

Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

ML Flow

Metadata Management

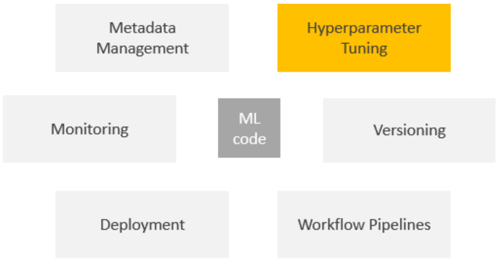


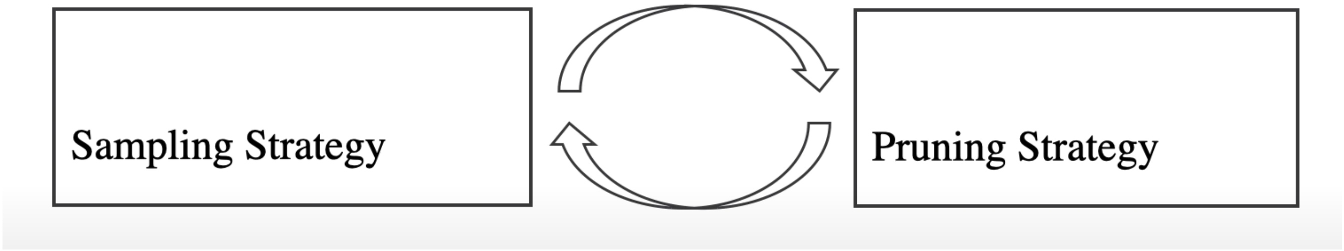
- **Läuft mit Python und R**
- **Sehr einfach einzubinden in bestehende Projekte**
  - Z.b via >> mlflow.tensorflow.autolog()
- **Unterstützt diverse Bibliotheken**
  - Tensorflow, Scikit, Pytorch, Keras, Fastai ...
- **Übersichtliches UI**
  - Starten >> mlflow ui

Hyperparameter Tuning

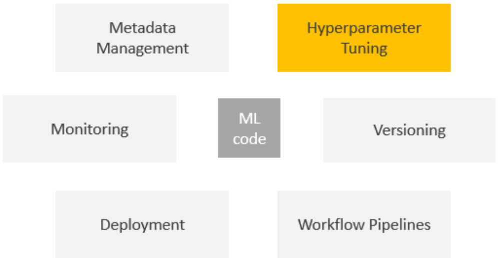
Optuna

Optuna
Hyperparameter Tuning





1. **Aufstellen des Suchraumes**
2. **Abbrechen von wenig erfolgsversprechenden Strategien**



Optuna

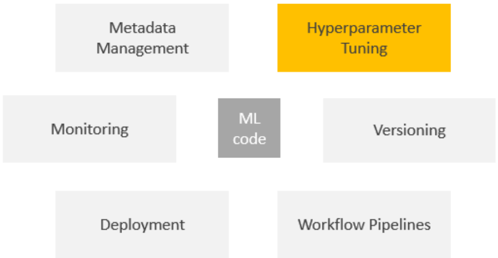
Hyperparameter Tuning

Code Integration

```
import optuna

def objective(trial):
        Your code here!
    return evaluation_score

study = optuna.create_study()
study.optimize(objective, n_trials= Number of trials )
```

Optuna

Hyperparameter Tuning

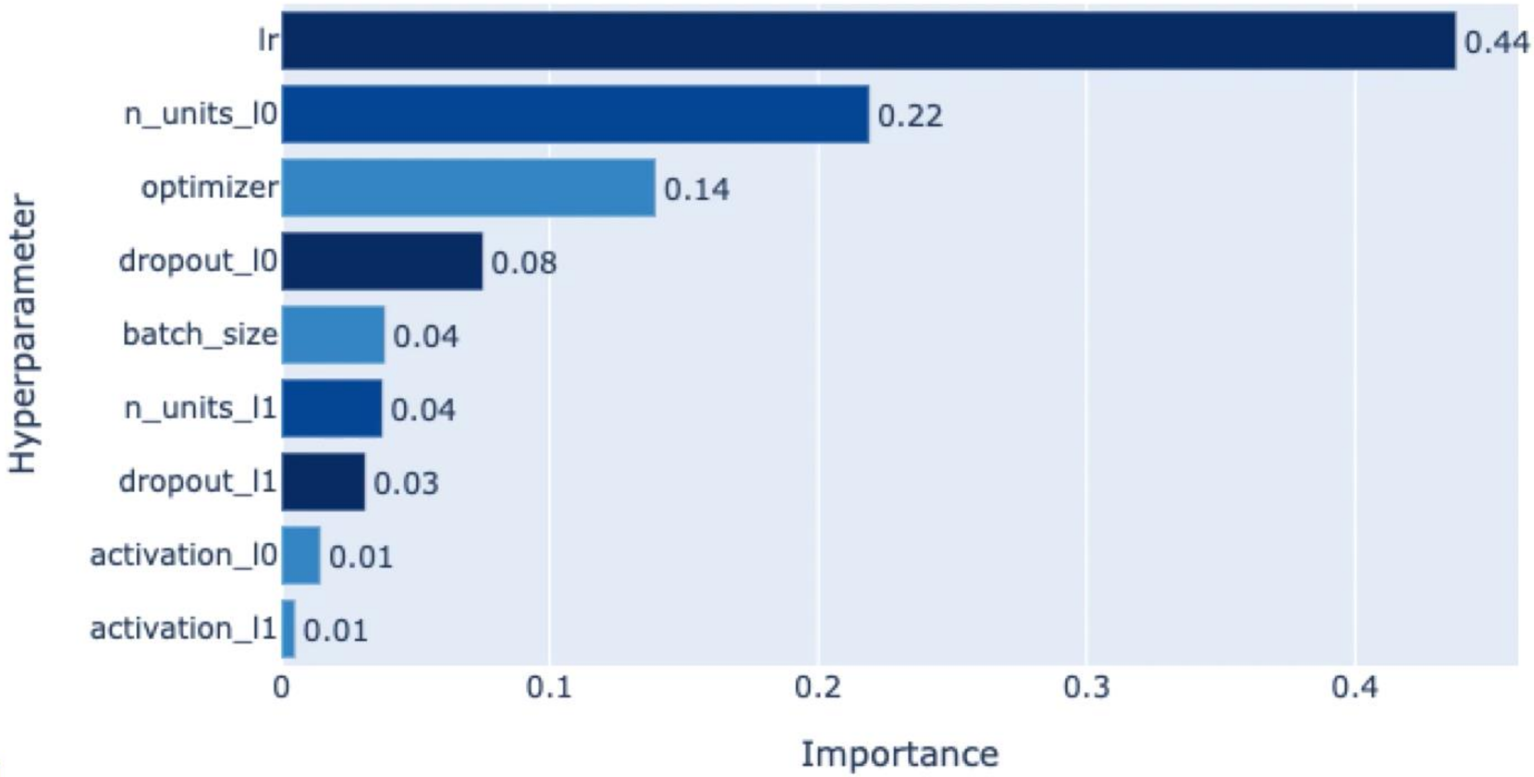


Code Integration

```
657/657 [==============================] - 1s 895us/step - loss: 0.0939 - accuracy: 0.9719
0.9719047546386719
0.09385089576244354
[I 2021-11-03 15:55:32,337] Trial 99 finished with value: 0.09385089576244354 and parameters: {'n_layers': 1, 'n_units_l0': 237, 'optimizer': 'Adam', 'n_epochs': 7}. Best is trial 31 with value: 0.08841179311275482.

Process finished with exit code 0
```

Optuna
Hyperparameter Tuning

Parameter Einfluss



Metadata Management | Hyperparameter Tuning | Monitoring | ML code | Versioning | Deployment | Workflow Pipelines

Optuna

Hyperparameter Tuning

Parallel Computing

```
$ python example.py
[I 2019-05-21 11:16:43,493] Using an existing study with name 'example-study' in
stead of creating a new one.
[I 2019-05-21 11:16:53,916] Finished trial#9 resulted in value: 0.744140625. Cur
rent best value is 0.744140625 with parameters: {'momentum_sgd_lr': 7.9440223034
05195e-05, 'n_layers': 1, 'n_units_l0': 28.753028463759527, 'optimizer': 'Moment
umSGD', 'weight_decay': 2.1924271434430527e-06}.
     total [##############################.....................] 58.88%
this epoch [#############################################.....] 88.80%
       138 iter, 5 epoch / 10 epochs
    17.446 iters/sec. Estimated time to finish: 0:00:05.524267.
```

```
instead of creating a new one.
[I 2019-05-21 11:16:51,250] Finished trial#7 resulted in value: 0.80581430
737. Current best value is 0.805814303457737 with parameters: {'adam_alpha
.5501424552926803e-05, 'n_layers': 1, 'n_units_l0': 4.237994876174356, 'op
zer': 'Adam', 'weight_decay': 4.2909422937320695e-09}.
[I 2019-05-21 11:17:02,585] Finished trial#11 resulted in value: 0.1422776
452606. Current best value is 0.14227764308452606 with parameters: {'momen
sgd_lr': 0.02227271795421007, 'n_layers': 3, 'n_units_l0': 6.0048260623270
'n_units_l1': 27.182953480840045, 'n_units_l2': 24.852836817988184, 'optim
': 'MomentumSGD', 'weight_decay': 1.378459092583759e-10}.
     total [####.............................................]  9.81%
this epoch [################################################.] 98.13%
        23 iter, 0 epoch / 10 epochs
    26.735 iters/sec. Estimated time to finish: 0:00:07.906416.
```

```
$ python example.py
[I 2019-05-21 11:16:43,621] Using an existing study with name 'example-study' in
stead of creating a new one.
[I 2019-05-21 11:16:55,890] Finished trial#10 resulted in value: 0.8976862980052
829. Current best value is 0.16624098271131516 with parameters: {'momentum_sgd_l
r': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356666, 'n_un
its_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimizer': 'Momen
tumSGD', 'weight_decay': 0.0007099850265678694}.
     total [########################################..........] 81.49%
this epoch [#######...........................................] 14.93%
       191 iter, 8 epoch / 10 epochs
    25.217 iters/sec. Estimated time to finish: 0:00:01.720039.
```

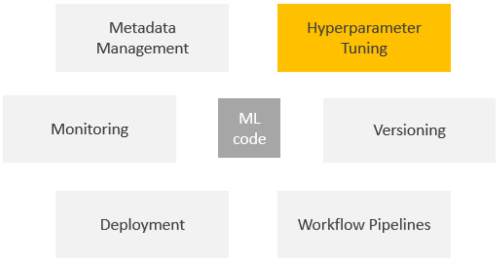
```
$ python example.py
[I 2019-05-21 11:16:42,230] Using an existing study with name 'example-stu
instead of creating a new one.
[I 2019-05-21 11:16:56,449] Finished trial#6 resulted in value: 0.88273737
1315. Current best value is 0.16624098271131516 with parameters: {'momentu
d_lr': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356
'n_units_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimiz
 'MomentumSGD', 'weight_decay': 0.0007099850265678694}.
     total [#######################################...........] 78.93%
this epoch [############################################.....] 89.33%
       185 iter, 7 epoch / 10 epochs
    24.806 iters/sec. Estimated time to finish: 0:00:01.990426.
```

```
$ python example.py
[I 2019-05-21 11:16:42,729] Using an existing study with name 'example-study' in
stead of creating a new one.
[I 2019-05-21 11:16:54,781] Finished trial#8 resulted in value: 0.91624098550528
29. Current best value is 0.16624098271131516 with parameters: {'momentum_sgd_lr
': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356666, 'n_uni
ts_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimizer': 'Moment
umSGD', 'weight_decay': 0.0007099850265678694}.
```

```
$ python example.py
[I 2019-05-21 11:16:42,022] Using an existing study with name 'example-stu
instead of creating a new one.
[I 2019-05-21 11:16:54,576] Finished trial#5 resulted in value: 0.16624098
31516. Current best value is 0.16624098271131516 with parameters: {'moment
gd_lr': 0.012378325131946236, 'n_layers': 3, 'n_units_l0': 10.118405976356
 'n_units_l1': 99.22366485593328, 'n_units_l2': 6.970853785986278, 'optimi
: 'MomentumSGD', 'weight_decay': 0.0007099850265678694}.
     total [######################################............] 76.37%
this epoch [################################..................] 63.73%
       179 iter, 7 epoch / 10 epochs
    22.063 iters/sec. Estimated time to finish: 0:00:02.509897.
```

Optuna

Hyperparameter Tuning



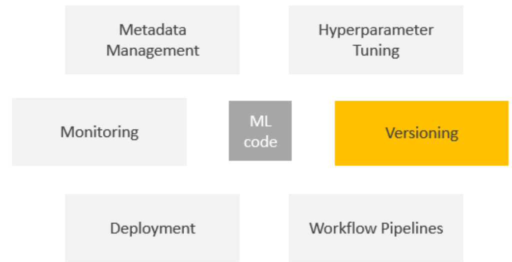
- **Integrierte Bibliotheken**
  - Pytorch lightning
  - Pytorch ignite
  - FastAi
- **Parallel Computing**
- **Visualisierung des Suchraumes**
- **Gewichtung des Parametereinflusses**

Versioning

DVC/MLFlow/Git

Data Version Control

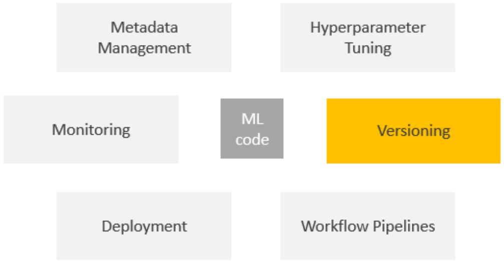
Versioning



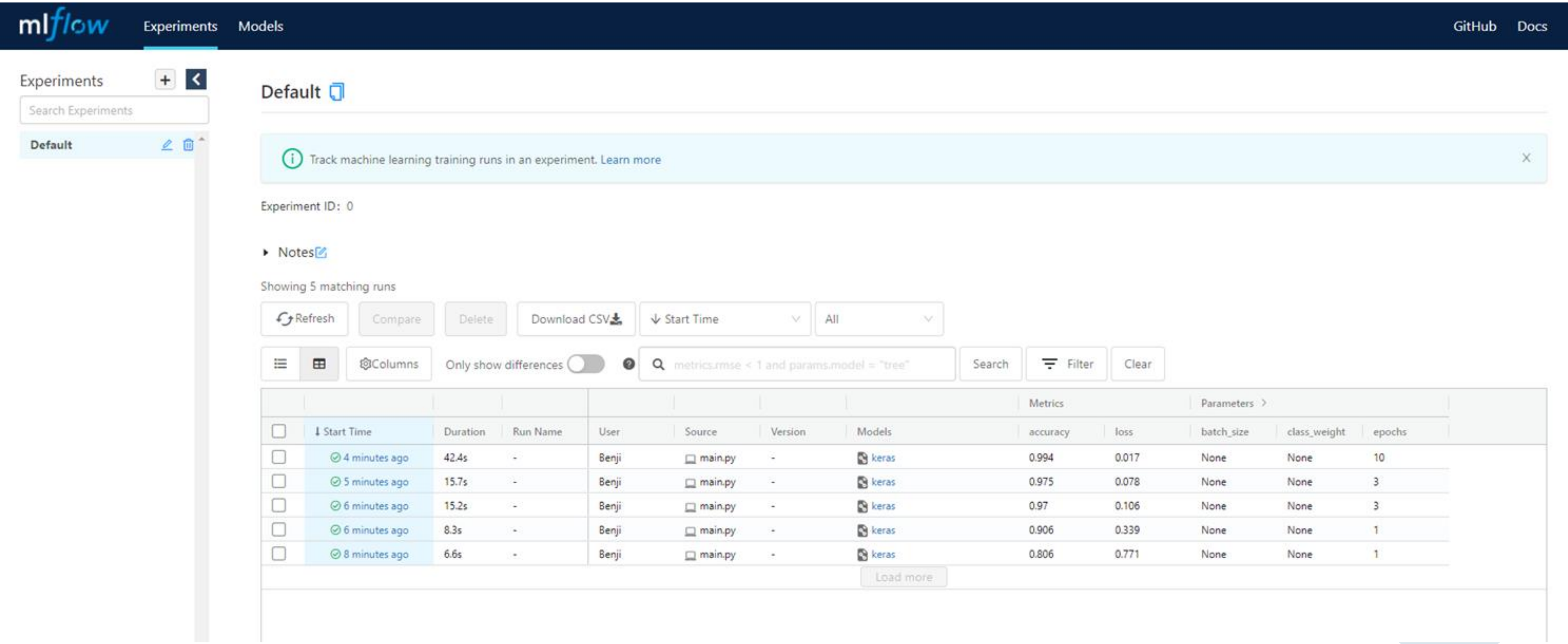
- Versionierung von Datensätzen
- Prinzip: Versionierung des Codes über Git
- Datensätze zu groß für GitHub Repos
- Trainingsdaten werden über DVC auf einen Storage geladen (z.b. direkt auf gdrive)
- Im Git Repository wird eine dvc/config file abgelegt welche das Git-Repo mit dem DVC-Repo verknüpft

ML Flow

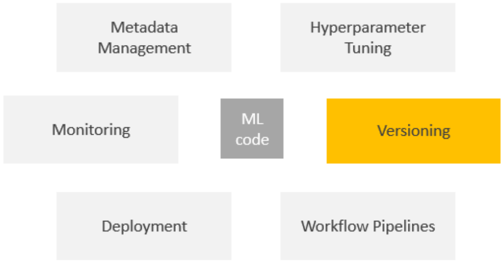
Versioning



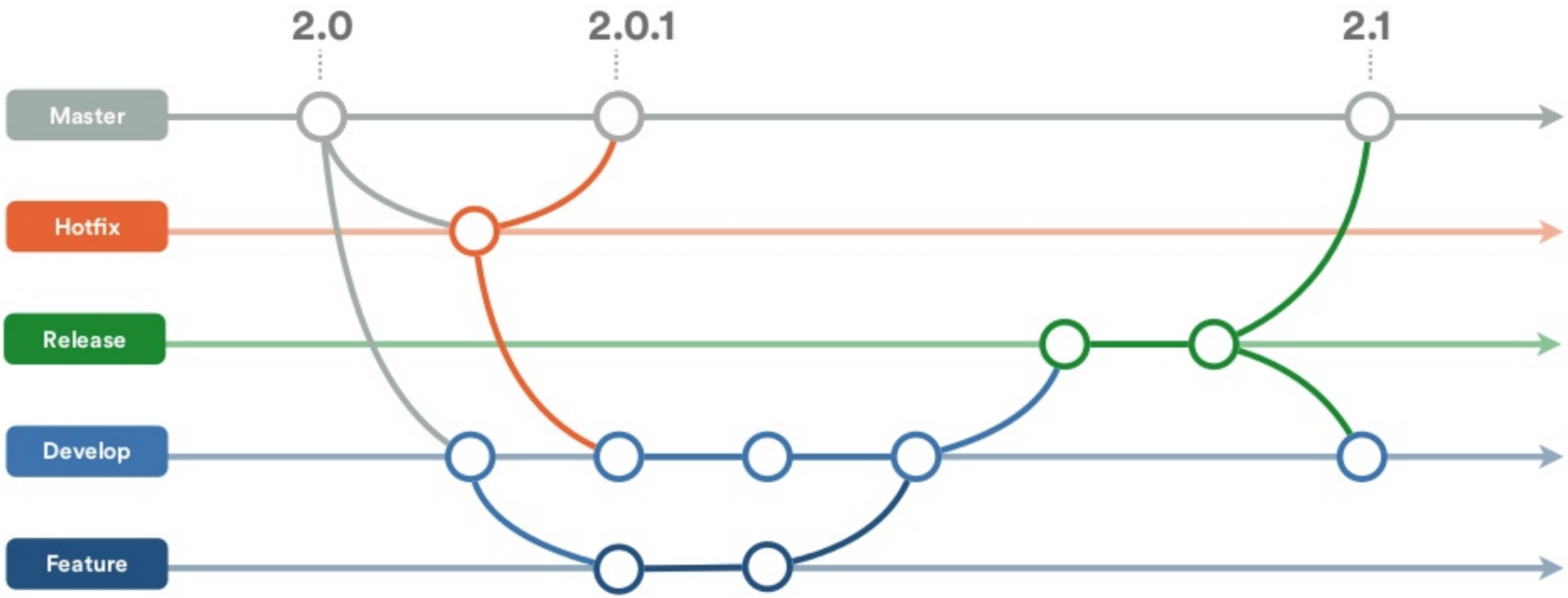
Versionierung von Modellen



GIT
Versioning



Versionierung von Code und Pipelines

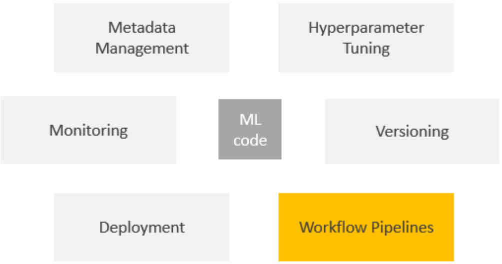


Workflow Pipelines

Kedro

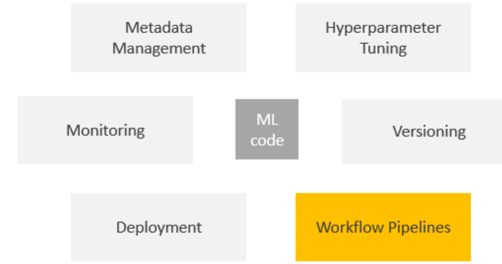
Kedro

Workflow Pipelines

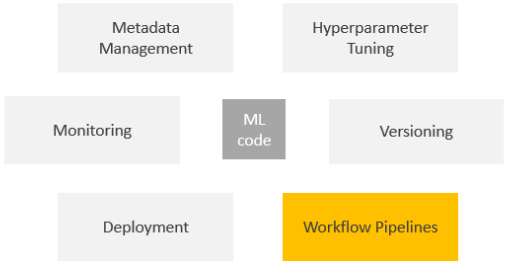


- **Python library**
- **Erstellen von Noden**
  - Funktion
  - Inputs
  - Outputs
  - Name
- **Verbinden von Nodes zu einer Pipeline**
- **Verbinden von mehreren Pipelines**
- **Speichern/Teilen von Pipelines**
- **Visualisieren von Pipelines**

Kedro
Workflow Pipelines



- Neues Projekt: >> kedro new
- Datensets importieren
- Catalog definieren
- Funktionen/Nodes definieren
- Pipelines registrieren
- Sequentiell/Parallel laufen lassen



Kedro

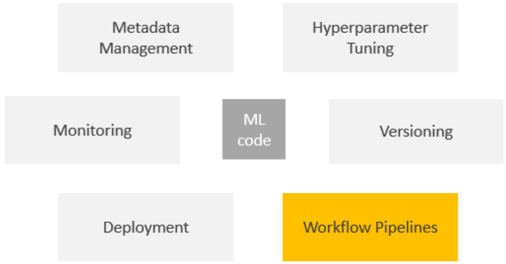
Workflow Pipelines

Datensets/Artefakte definieren

```yaml
companies:
  type: pandas.CSVDataSet
  filepath: data/01_raw/companies.csv
  layer: raw

reviews:
  type: pandas.CSVDataSet
  filepath: data/01_raw/reviews.csv
  layer: raw

shuttles:
  type: pandas.ExcelDataSet
  filepath: data/01_raw/shuttles.xlsx
  layer: raw
```

Kedro

Workflow Pipelines

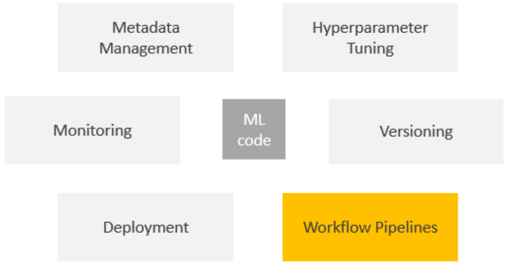


(Node)-Funktion

```python
def preprocess_companies(companies: pd.DataFrame) -> pd.DataFrame:
    """Preprocesses the data for companies.

    Args:
        companies: Raw data.
    Returns:
        Preprocessed data, with `company_rating` converted to a float and
        `iata_approved` converted to boolean.
    """
    companies["iata_approved"] = _is_true(companies["iata_approved"])
    companies["company_rating"] = _parse_percentage(companies["company_rating"])
    return companies
```

Kedro

Workflow Pipelines



Pipeline Definition

```python
def create_pipeline(**kwargs):
    return Pipeline(
        [
            node(
                func=preprocess_companies,
                inputs="companies",
                outputs="preprocessed_companies",
                name="preprocess_companies_node",
            ),
            node(
                func=preprocess_shuttles,
                inputs="shuttles",
                outputs="preprocessed_shuttles",
                name="preprocess_shuttles_node",
            ),
            node(
                func=create_model_input_table,
                inputs=["preprocessed_shuttles", "preprocessed_companies", "reviews"],
                outputs="model_input_table",
                name="create_model_input_table_node",
            ),
        ]
    )
```

Kedro

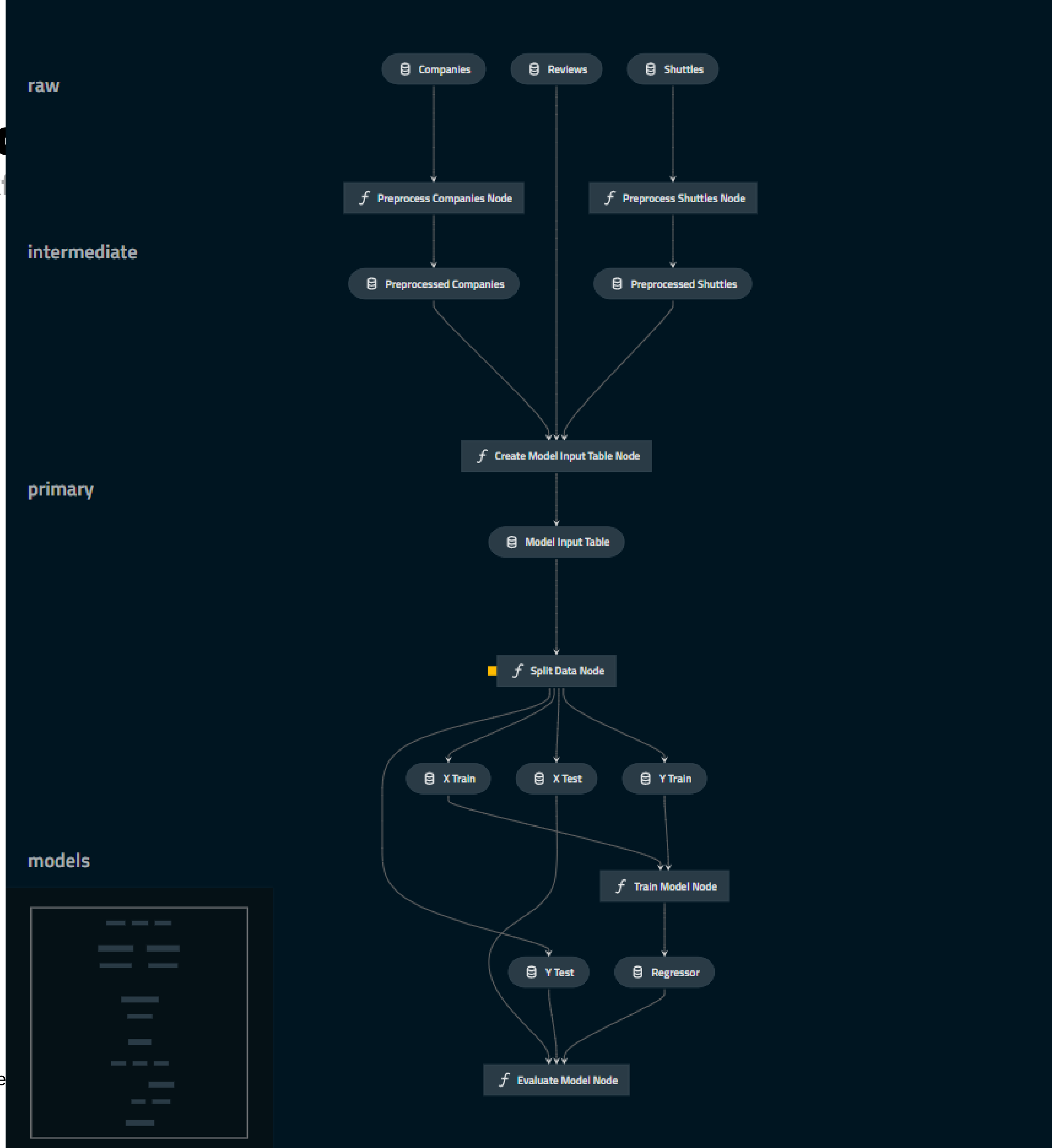
Workflow Pipelines

Pipelines verbinden

```python
def register_pipelines() -> Dict[str, Pipeline]:
    """Register the project's pipeline.

    Returns:
        A mapping from a pipeline name to a ``Pipeline`` object.

    """
    data_processing_pipeline = dp.create_pipeline()
    data_science_pipeline = ds.create_pipeline()

    return {
        "__default__": data_processing_pipeline + data_science_pipeline,
        "dp": data_processing_pipeline,
        "ds": data_science_pipeline,
    }
```

Ke

Work



Hyperparameter Tuning

ML ...de

Versioning

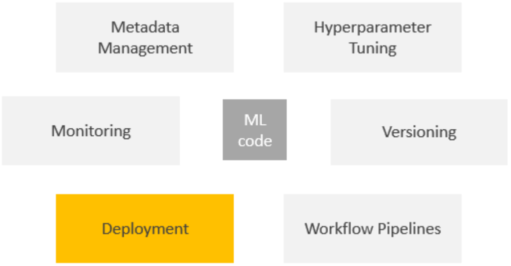
Workflow Pipelines

Visualisierung

Deployment

Bento

Bento
Deployment



1. Model

```
# train.py
from sklearn import svm
from sklearn import datasets

# Load training data
iris = datasets.load_iris()
X, y = iris.data, iris.target

# Model Training
clf = svm.SVC(gamma='scale')
clf.fit(X, y)
```

Bento

Deployment

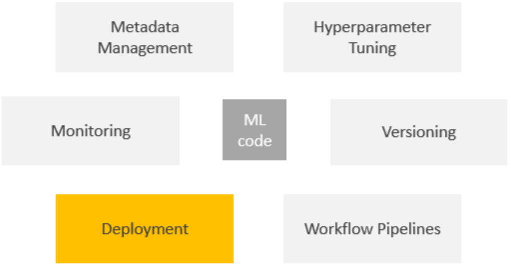
Metadata Management | Hyperparameter Tuning | Monitoring | ML code | Versioning | Deployment | Workflow Pipelines

2. Create API

```python
# bento_service.py
import pandas as pd

from bentoml import env, artifacts, api, BentoService
from bentoml.adapters import DataframeInput
from bentoml.frameworks.sklearn import SklearnModelArtifact

@env(infer_pip_packages=True)
@artifacts([SklearnModelArtifact('model')])
class IrisClassifier(BentoService):
    """
    A minimum prediction service exposing a Scikit-learn model
    """

    @api(input=DataframeInput(), batch=True)
    def predict(self, df: pd.DataFrame):
        """
        An inference API named `predict` with Dataframe input adapter, which codifies
        how HTTP requests or CSV files are converted to a pandas Dataframe object as the
        inference API function input
        """
        return self.artifacts.model.predict(df)
```

Bento

Deployment

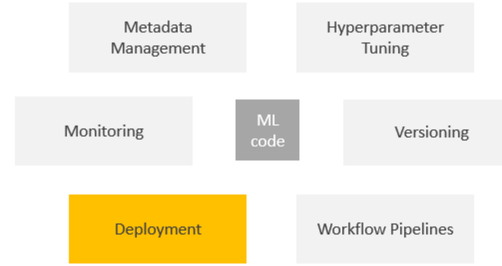


3. Pack Model and API

```python
# bento_packer.py

# import the IrisClassifier class defined above
from bento_service import IrisClassifier

# Create a iris classifier service instance
iris_classifier_service = IrisClassifier()

# Pack the newly trained model artifact
iris_classifier_service.pack('model', clf)

# Save the prediction service to disk for model serving
saved_path = iris_classifier_service.save()
```

Bento

Deployment

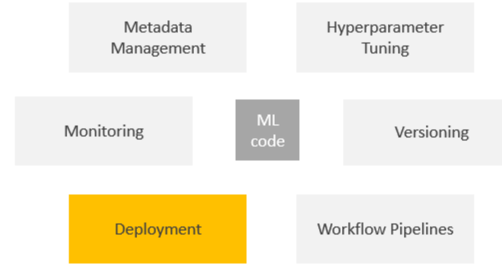


4. Serve API

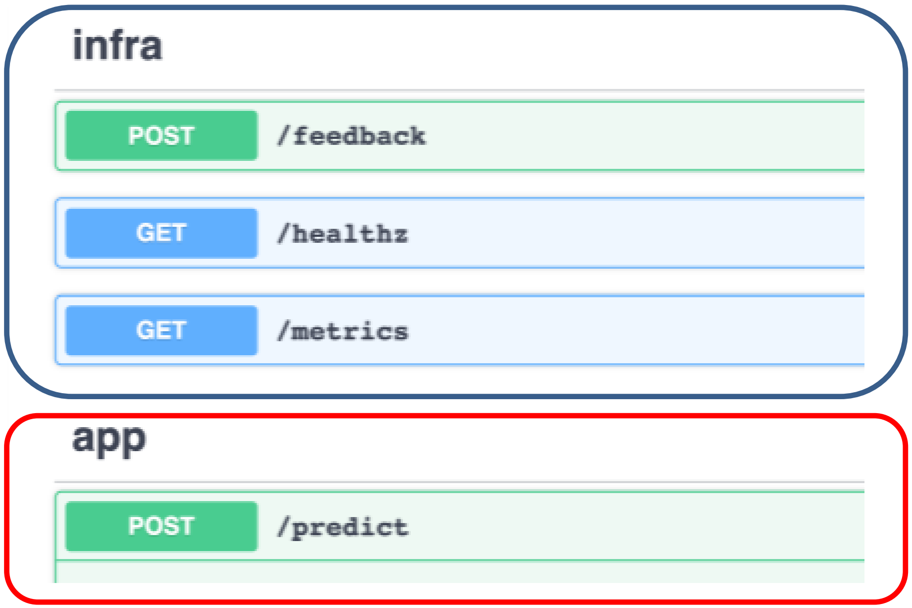
```
bentoml serve IrisClassifier:latest
```

– Model API wird auf localhost:5000 bereitgestellt

Bento
Deployment

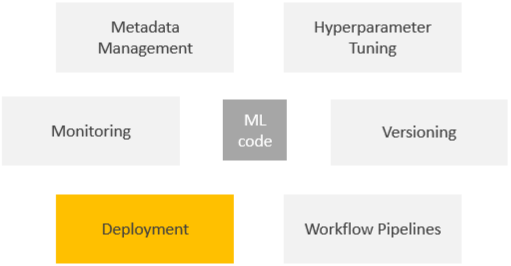


Running API Services



API Monitoring

Model Predictions



Bento

Deployment

Request Body

```
[
  [5.1, 3.5, 1.4, 0.2],
  [5.0, 3.5, 1.6, 0.3],
  [4.9, 4.5, 1.2, 0.2]
]
```

Execute | Clear

Responses

Curl

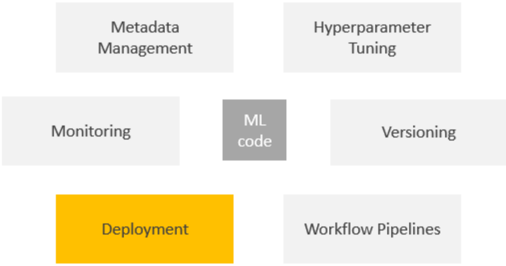
```
curl -X POST "http://127.0.0.1:5000/predict" -H "accept: */*" -H "Content-Type: application/json" -d "
[[5.1,3.5,1.4,0.2],[5,3.5,1.6,0.3],[4.9,4.5,1.2,0.2]]"
```

Request URL

```
http://127.0.0.1:5000/predict
```

Bento

Deployment



Response

Code	Details
200	**Response body**

```
[
  0,
  0,
  0
]
```

Download

Response headers

```
content-length: 9
content-type: application/json
date: Wed, 06 May 2020 00:08:59 GMT
request_id: cc0257bd-5299-439e-9f38-27cc731f887c
server: Werkzeug/0.16.0 Python/3.7.5
```

Responses

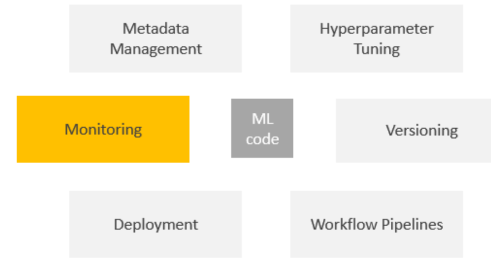
Code	Description	Links
200	success	*No links*

Monitoring

Fiddler

Fiddler

Monitoring



Model Details

Properties

BASIC

Fiddler ID
random_forest_26

Name
Random Forest Model

Description
This is models customer bank churn

TECHNICAL

Algorithm
-

Framework
-

Model Task
binary_classification

SCHEMA DETAILS

Input Type
structured

Class Labels
-

Targets
churn(category): [no,yes]

Dataset
bank_churn

Outputs

1 output(s) found in model schema

NAME	DATA TYPE
probability_yes	float

Fiddler

Monitoring

Metadata Management
Hyperparameter Tuning
Monitoring
ML code
Versioning
Deployment
Workflow Pipelines

Model Decisions

Fiddler

Monitoring

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

Workflow Pipelines

Model Accuracy

Metadata Management

Hyperparameter Tuning

Monitoring

ML code

Versioning

Deployment

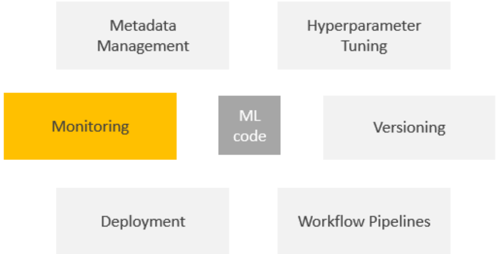
Workflow Pipelines

Fiddler

Monitoring

Model Drift

Fiddler
Monitoring



Feature Drift 1. April

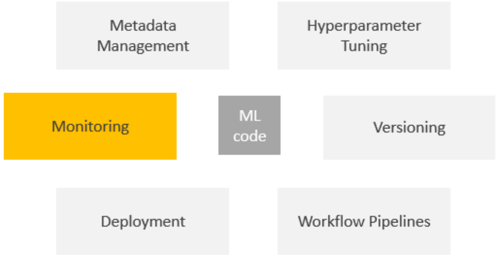
Drift Analytics

Apr 01, 2021 5PM - 1 Day

FEATURE	PREDICTION DRIFT IMPACT	FEATURE DRIFT	FEATURE IMPACT
age	45.472%	0.08	33.91%
balance	19.467%	0.09	13.56%
numofproducts	15.149%	0.03	27.13%
isactivemember	9.280%	0.04	14.45%

Fiddler

Monitoring



Feature Drift 12. April

Drift Analytics

Apr 12, 2021 5PM - 1 Day

Sear

FEATURE	PREDICTION DRIFT IMPACT	FEATURE DRIFT	FEATURE IMPACT
numofproducts	69.306%	0.32	27.13%
age	15.815%	0.06	33.91%
balance	7.564%	0.07	13.56%
isactivemember	2.299%	0.02	14.45%

Fiddler
Monitoring

Metadata Management
Hyperparameter Tuning
Monitoring
ML code
Versioning
Deployment
Workflow Pipelines

Feature Distribution

Technologie-Stack

Integrated Solutions

Azure Machine Learning

Metadata Management
- Python SDK für Azure Machine Learning

Hyperparameter Tuning
- hyperdrive Paket

Monitoring
- Azure Monitor

ML code

Versioning
- Datesets

Deployment
- Azure Pipelines

Workflow Pipelines
- Azure Machine Learning Pipelines
- Azure Data Factory-Pipelines
- Azure Pipelines

Amazon SageMaker

Prepare →

SageMaker Ground Truth
Label training data for machine learning

SageMaker Data Wrangler NEW
Aggregate and prepare data for machine learning

SageMaker Processing
Built-in Python, BYO R/Spark

SageMaker Feature Store NEW
Store, update, retrieve, and share features

SageMaker Clarify NEW
Detect bias and understand model predictions

Build →

SageMaker Studio Notebooks
Jupyter notebooks with elastic compute and sharing

Built-in and Bring-your-own Algorithms
Dozens of optimized algorithms or bring your own

Local Mode
Test and prototype on your local machine

SageMaker Autopilot
Automatically create machine learning models with full visibility

SageMaker JumpStart NEW
Pre-built solutions for common use cases

Train & tune →

One-click Training
Distributed infrastructure management

SageMaker Experiments
Capture, organize, and compare every step

Automatic Model Tuning
Hyperparameter optimization

Distributed Training Libraries NEW
Training for large datasets and models

SageMaker Debugger NEW
Debug and profile training runs

Managed Spot Training
Reduce training cost by 90%

Deploy & manage →

One-click Deployment
Fully managed, ultra low latency, high throughput

Kubernetes & Kubeflow Integration
Simplify Kubernetes-based machine learning

Multi-Model Endpoints
Reduce cost by hosting multiple models per instance

SageMaker Model Monitor
Maintain accuracy of deployed models

SageMaker Edge Manager NEW
Manage and monitor models on edge devices

SageMaker Pipelines NEW
Workflow orchestration and automation

SageMaker Studio
Integrated development environment (IDE) for ML

Amazon SageMaker – IDE for ML

Metadata Management
- SageMaker

Hyperparameter Tuning
- Auto Tuning

Monitoring
- SM Model Monitor
- Edge Manager

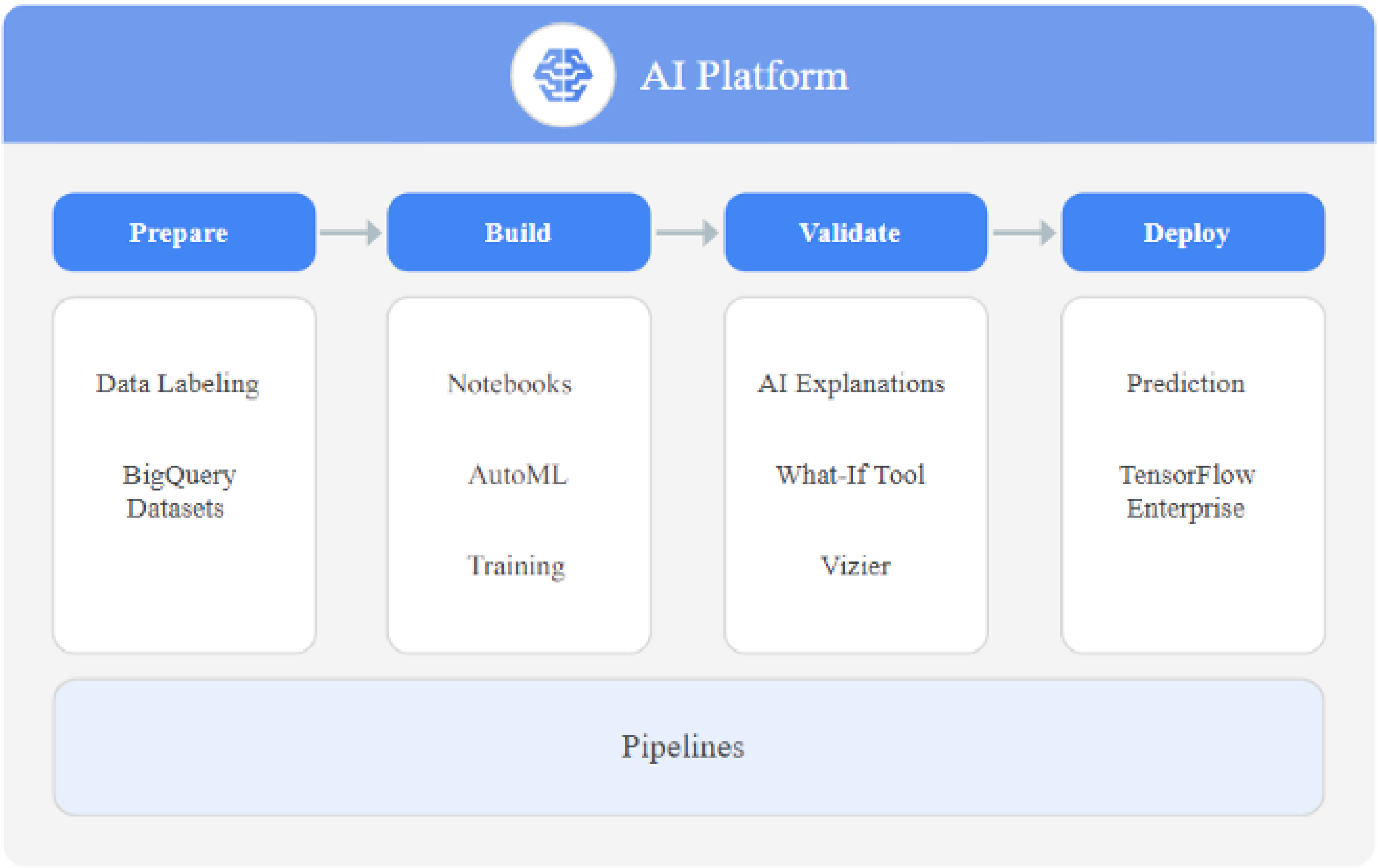
ML code

Versioning
- SM Experiments
- Model Versions

Deployment
- One-click deployment

Workflow Pipelines
- SM Pipelines

Google Cloud AI



Google Cloud AI - Vertex

Metadata Management
- Vertex

Hyperparameter Tuning
- Vertex AI Vizier

Monitoring
- Vertex AI Model Monitoring

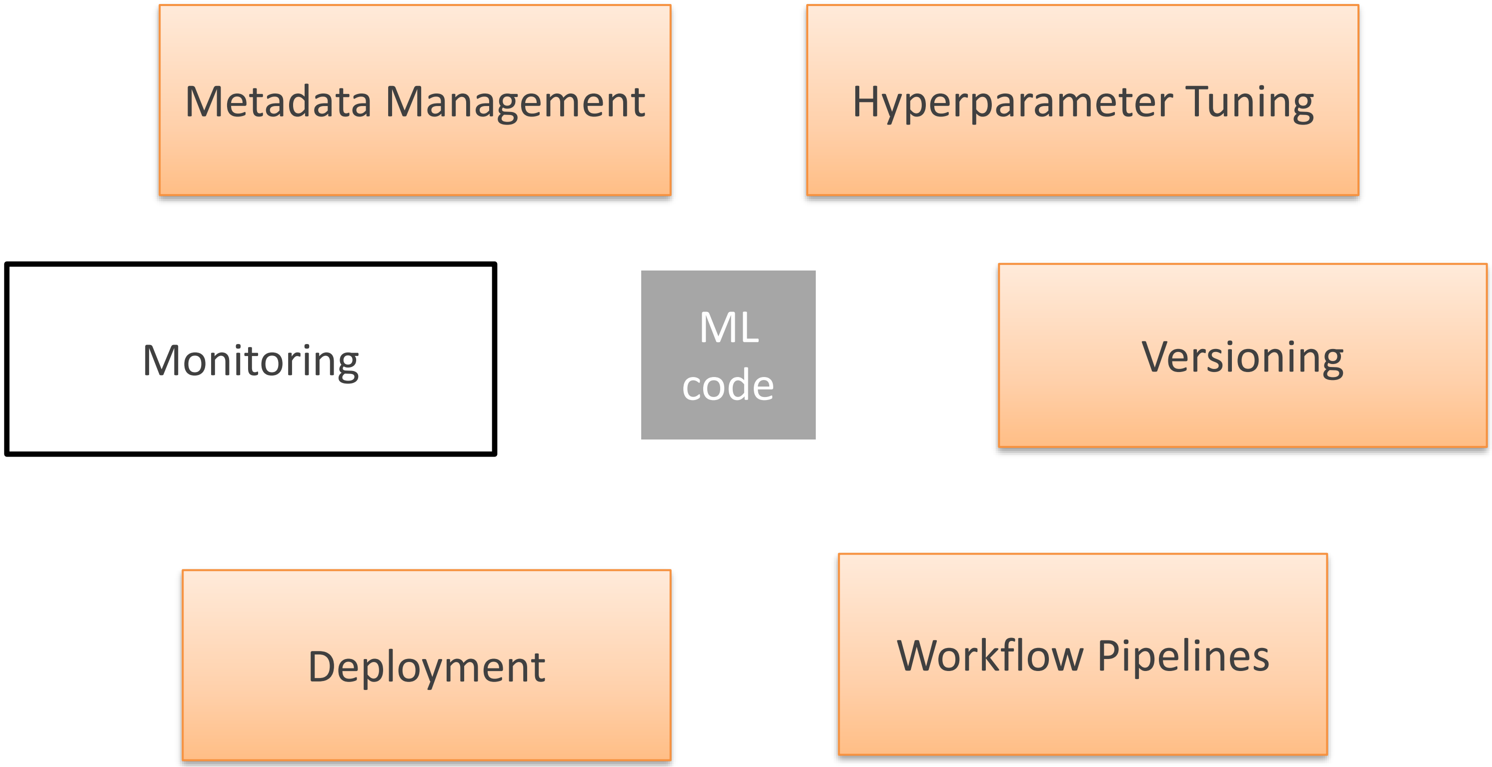
ML code

Versioning

Deployment
- AI Platform Prediction

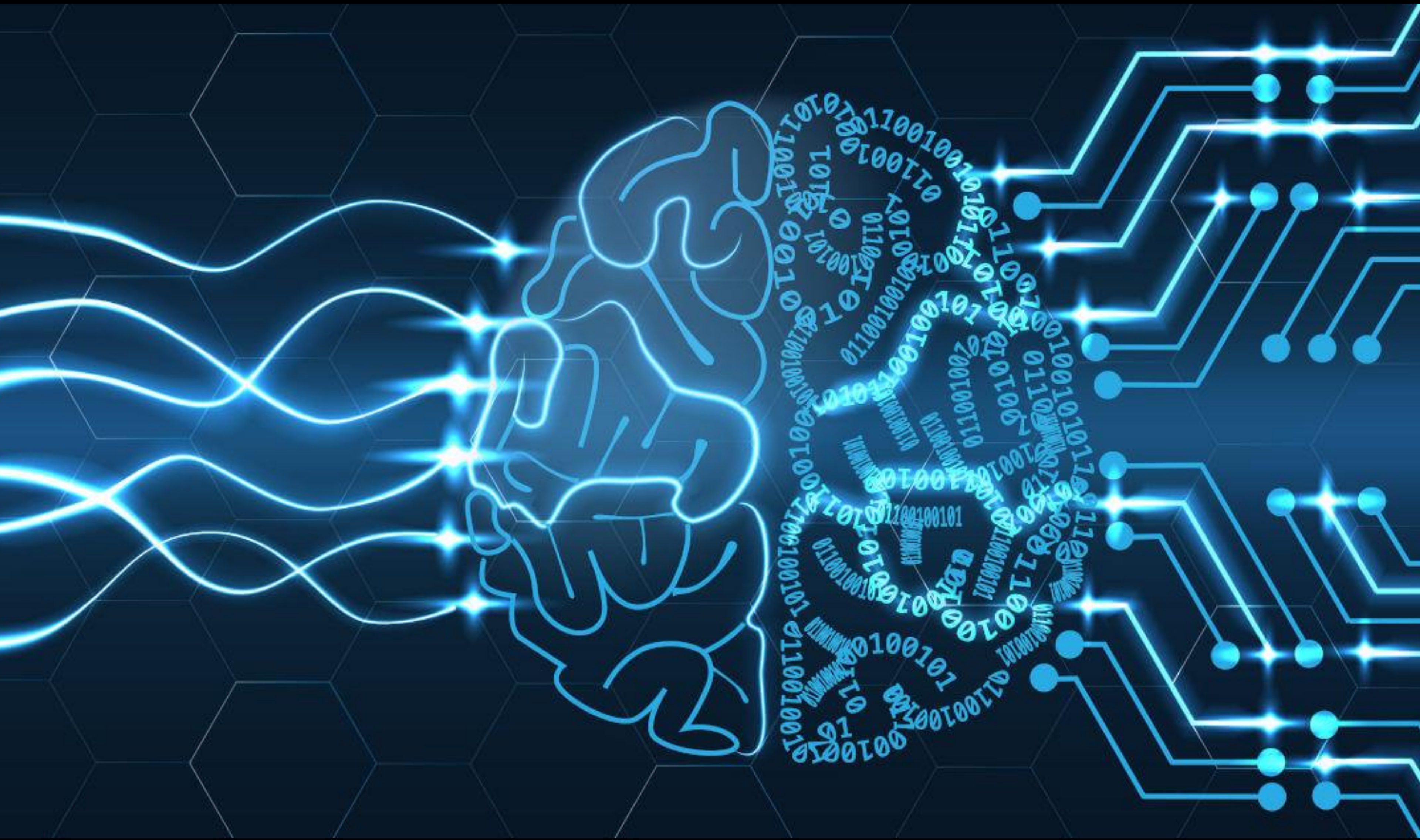
Workflow Pipelines
- Vertex AI Pipelines

TensorFlow Extended



Handlungsempfehlung

- **Flexibilität**
  - Neues Gebiet > Wechsel eines Tools muss einfach möglich sein
- **Stabilität**
  - Untergang von kleinen Tools auf lange Sicht möglich > Auf große Player setzen
- **Priorisierung**
  - Komplexes Gebiet > Da anfangen wo der größte Nutzen liegt



Quellenverzeichnis

- https://www.trustradius.com/mlops
- https://neptune.ai/blog/best-mlops-tools
- https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning#mlops_level_0_manual_process
- https://optuna.readthedocs.io/en/stable/tutorial/index.html
- https://www.youtube.com/watch?v=kLKBcPonMYw
- https://towardsdatascience.com/google-cloud-ai-platform-hyper-accessible-ai-machine-learning-cddd8c3348b3
- https://searchcloudcomputing.techtarget.com/tip/6-Amazon-SageMaker-capabilities-developers-should-know-about
- https://aws.amazon.com/de/machine-learning/
- https://datasolut.com/crisp-dm-standard/#Business-Understanding
- https://kedro.readthedocs.io/en/stable/01_introduction/01_introduction.html
- https://kedro.readthedocs.io/en/stable/03_tutorial/06_visualise_pipeline.html
- https://docs.bentoml.org/en/latest/quickstart.html#run-on-google-colab
- https://www.fiddler.ai/ml-monitoring