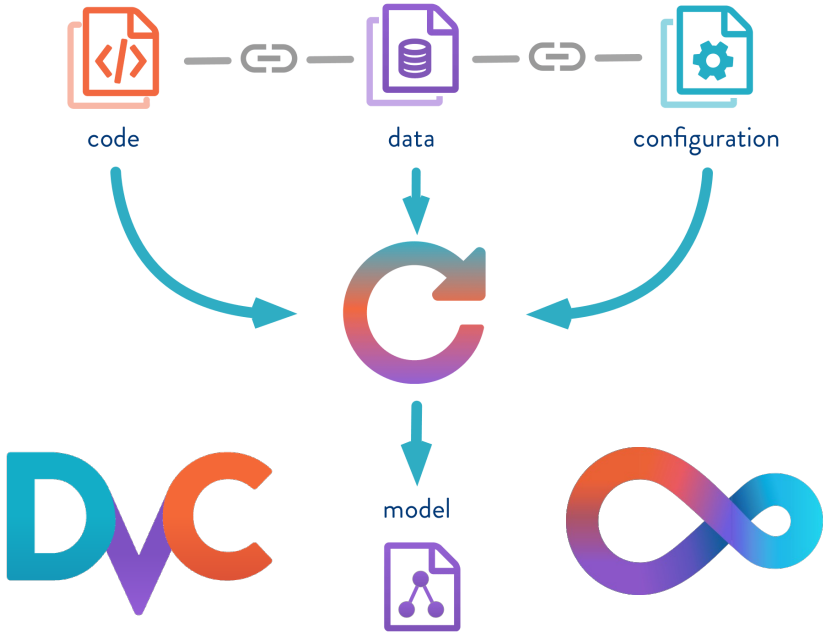
University of Stuttgart
Cluster of Excellence „Data-integrated Simulation Sciences“



Building reproducible workflows using Data Version Control and Continuous Machine Learning

Jan Range | DINI Workshop Stuttgart | 16-09-22

Agenda

1. Introduction

- Workflows in scientific projects
- Version Control using Git

2. Data Version Control

- Reproducible git-based workflows
- Iterative Studio – Comparing experiments

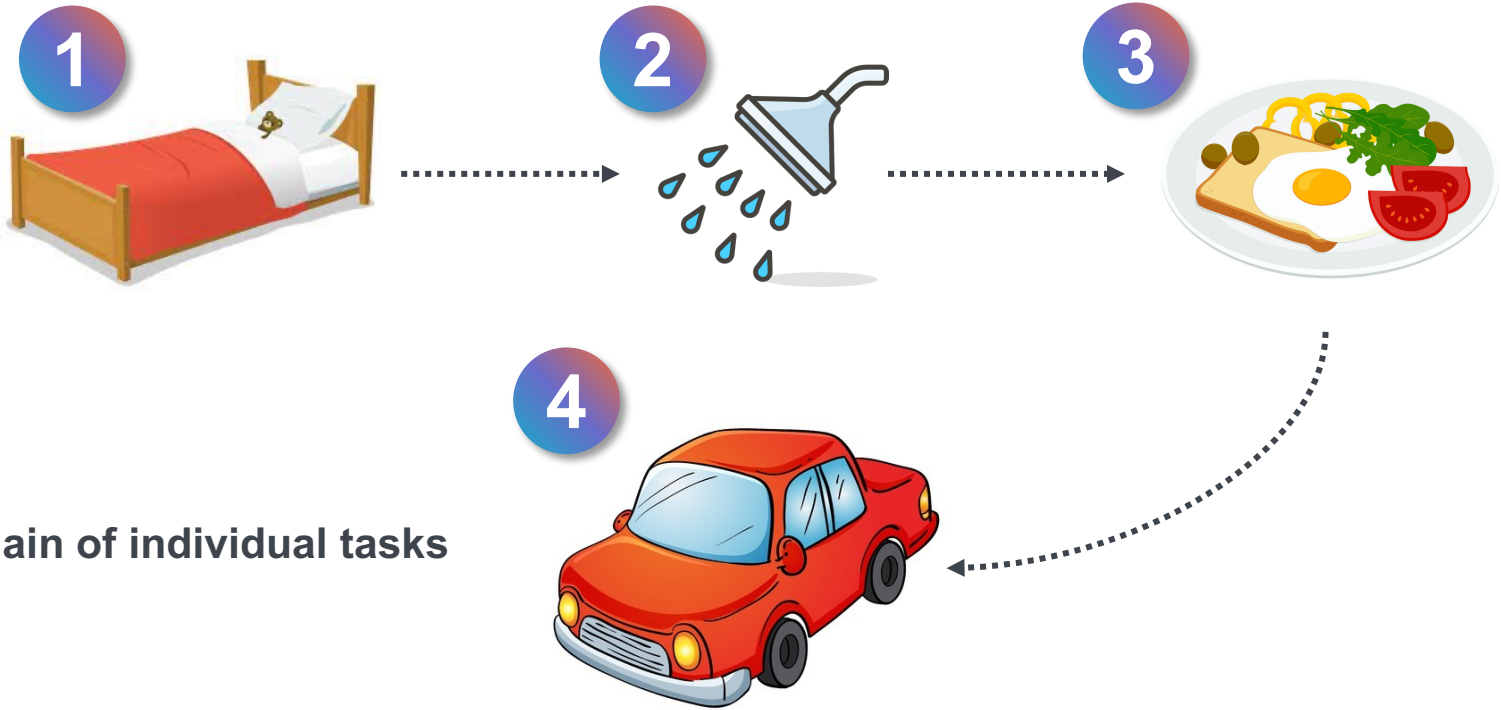
3. Continuous Machine Learning

- Automatic execution of workflows

4. Outlook

Introduction

Workflows



→ **Chain of individual tasks**

Introduction
Workflows

- Workflows **standardize** processes
  - Processing measurement data into a database
  - Machine learning and simulation pipelines
- Enable **automisation** of work
  - Focus on parameters rather than implementation
  - Ensure reproducibility of experiments
- Prevents **human errors** and **saves times**

Introduction

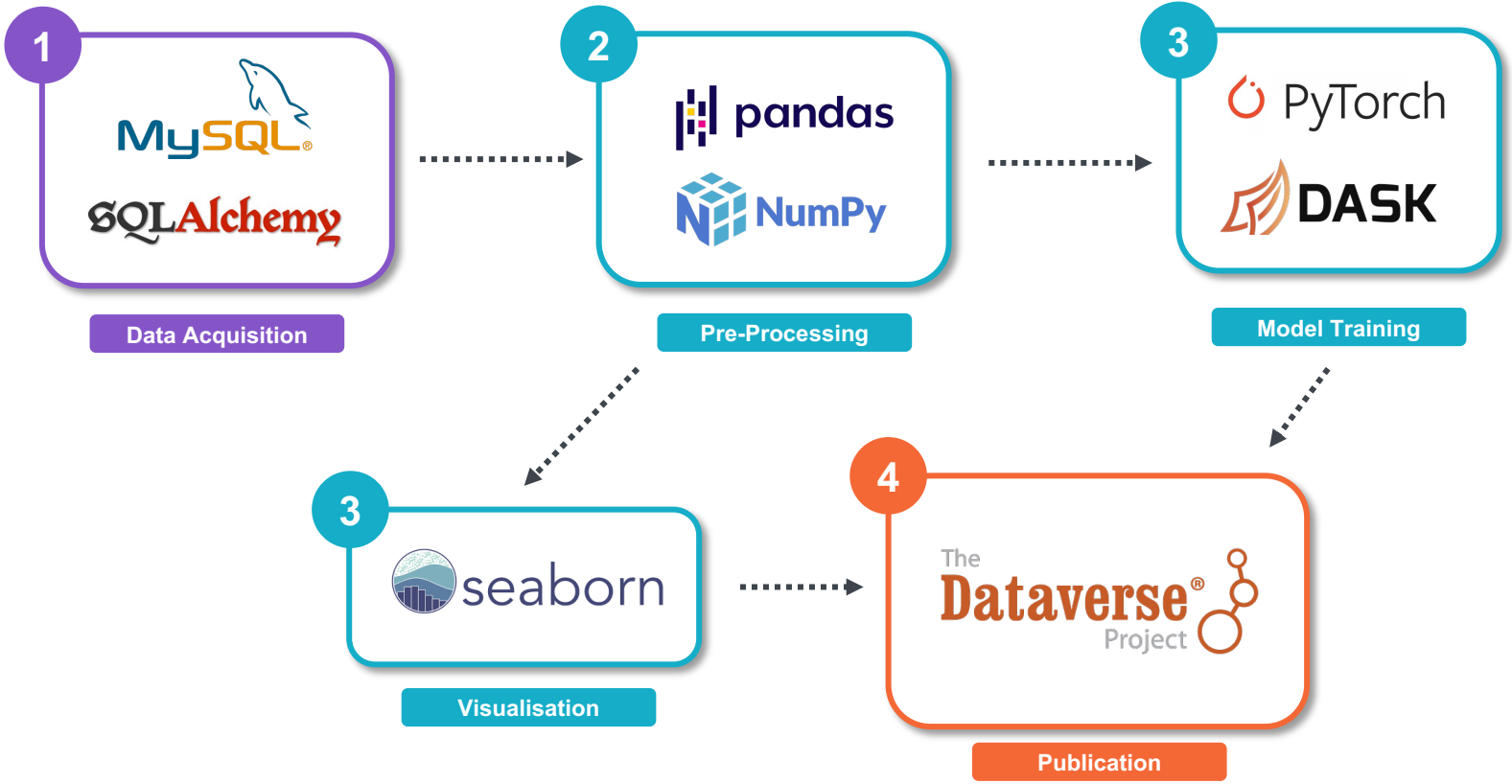
Workflow Requirements

- Keep **track** of each process step
  - Inputs and outputs of the workflow
  - Configurations, metrics and plots
- **Reproducibility** of results
  - Version control of source code
  - Workflow state when it was executed

Introduction
Workflow Requirements

- Application in **different contexts**
  - Re-usable by other scientists
  - Compatible in other environments
- **Modular** design
  - Quickly exchange or add tasks
  - Prevent redundant calculations

Example workflow



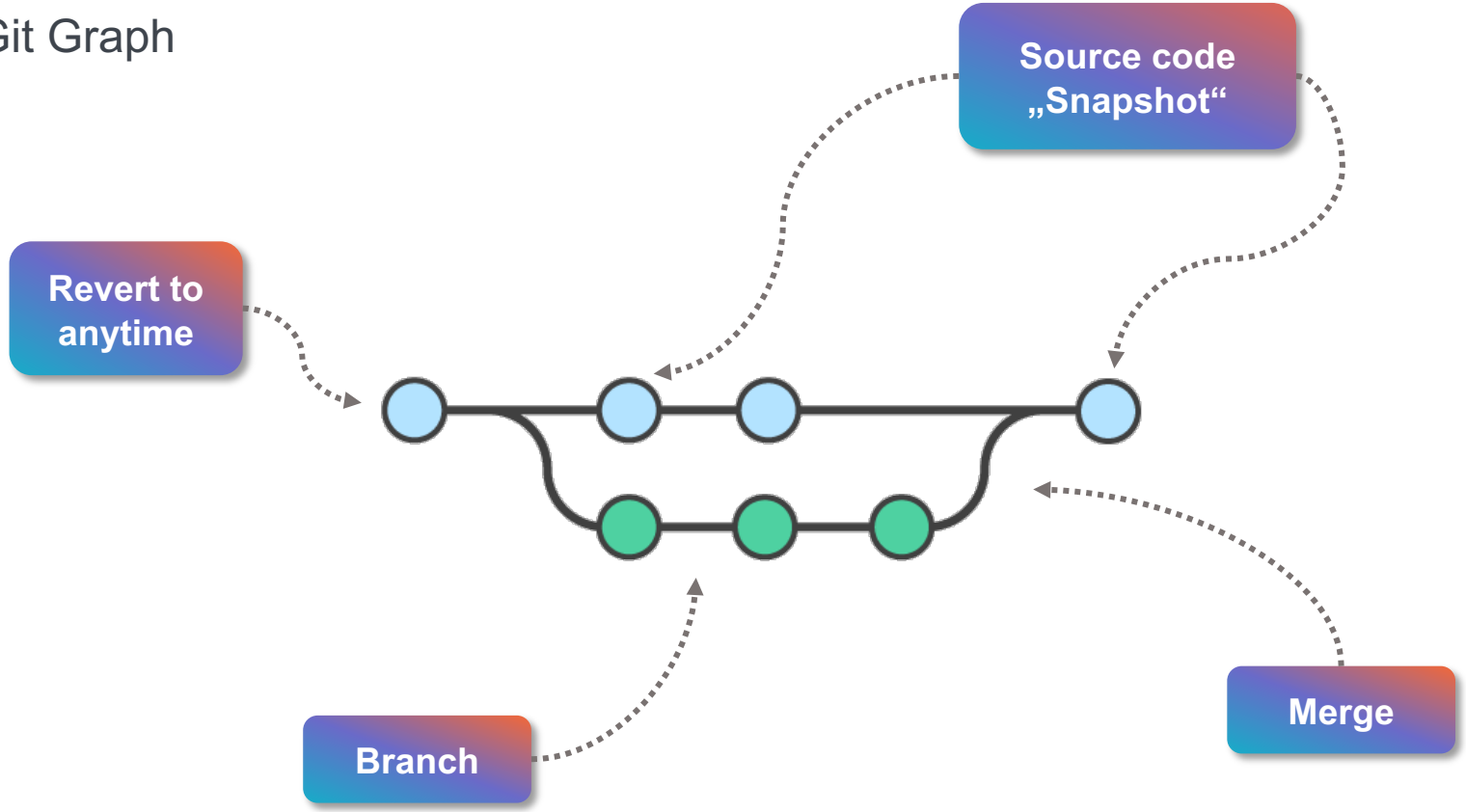
Basics of Git

Overview

- Essential tool for **software development**
  - Keeps track of source code „snapshots“
  - Reversion to any state
- **Collaborative** environment
  - Parallel development
  - Structured due to branching

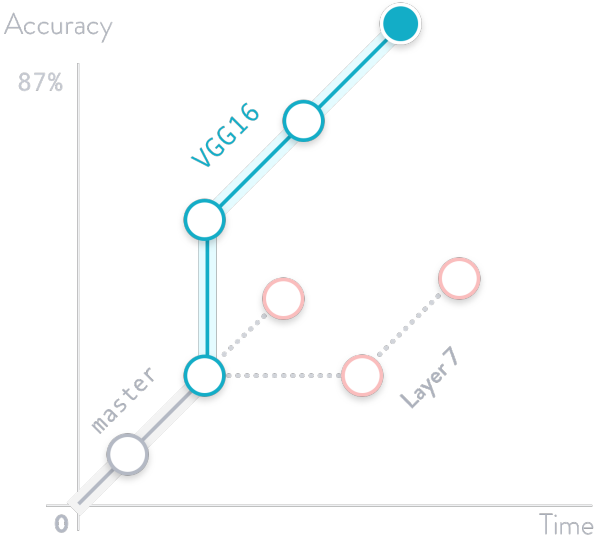
Basics of Git

Git Graph



Data Version Control
Overview

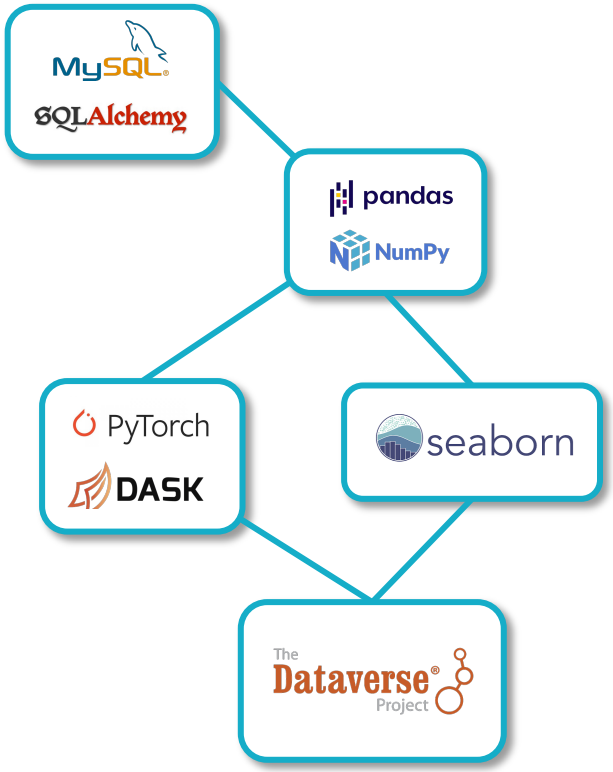
- Git-based **workflow management** tool
  - Open-source software
  - Workflow version control
- Comprehensive **process report**
  - → *Input and Output data*
  - → *Parameters and Experiments*
  - → *Metrics and Plots*



Data Version Control

Building workflows

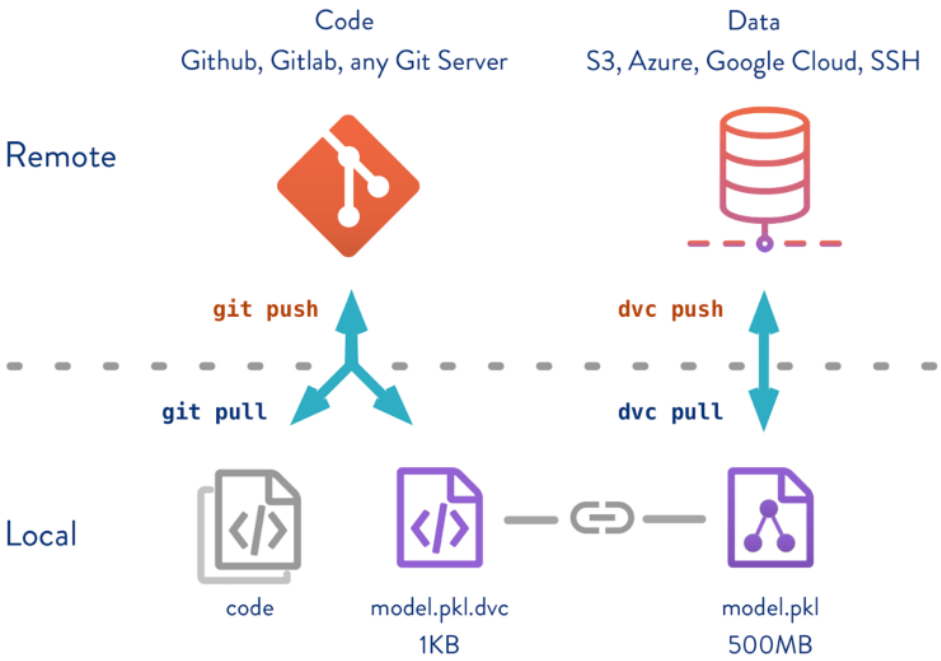
- Workflows are described as a **DAG**
  - Each task/application is a node
  - DVC keeps track of the structure
- Managed through **YAML** files
  - Stores checksum of code and parameters
  - Specifies what has to be executed per stage



Data Version Control

Reproducibility

- Workflow state is **tracked via Git**
  - Parameters / Nodes / Metrics / Code
- **Data** is tracked via DVC
  - Inputs / Intermediates / Outputs

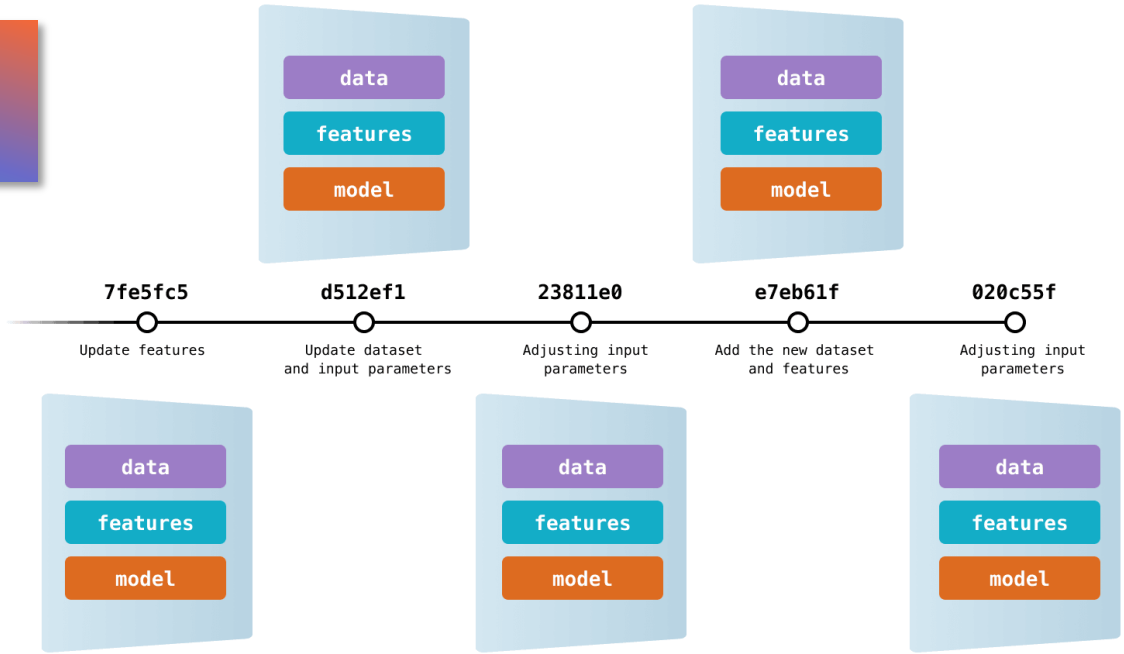


Data Version Control

Reproducibility

→ Workflow history as a Git graph

→ Each node can be reproduced



Data Version Control

How to

1 Initialize DVC inside a Git repository

```
$ dvc init
```

2 Build the DAG by adding stages and their dependencies

```
$ dvc stage add -n prepare \
                -p prepare.seed,prepare.split \
                -d src/prepare.py -d data/data.xml \
                -o data/prepared \
                python src/prepare.py data/data.xml
```

Data Version Control

How to

3 Execute the workflow

```
$ dvc repro
```

4 Commit and push workflow state

```
$ git commit -a -m "updated data after modified featurization"
```

```
$ dvc push
```

Data Version Control

Iterative Studio

- Web-based app to **inspect workflow results**
  - Includes all tracked metadata (parameters, metrics, …)
  - Sortable spreadsheet design to filter best results
- **Comparison** of individual runs
  - Differences between metrics and parameters
  - Combined plots to inspect selected runs at once

Data Version Control

Iterative Studio

JR-1991 > MLSession_Boosting_DVC Public

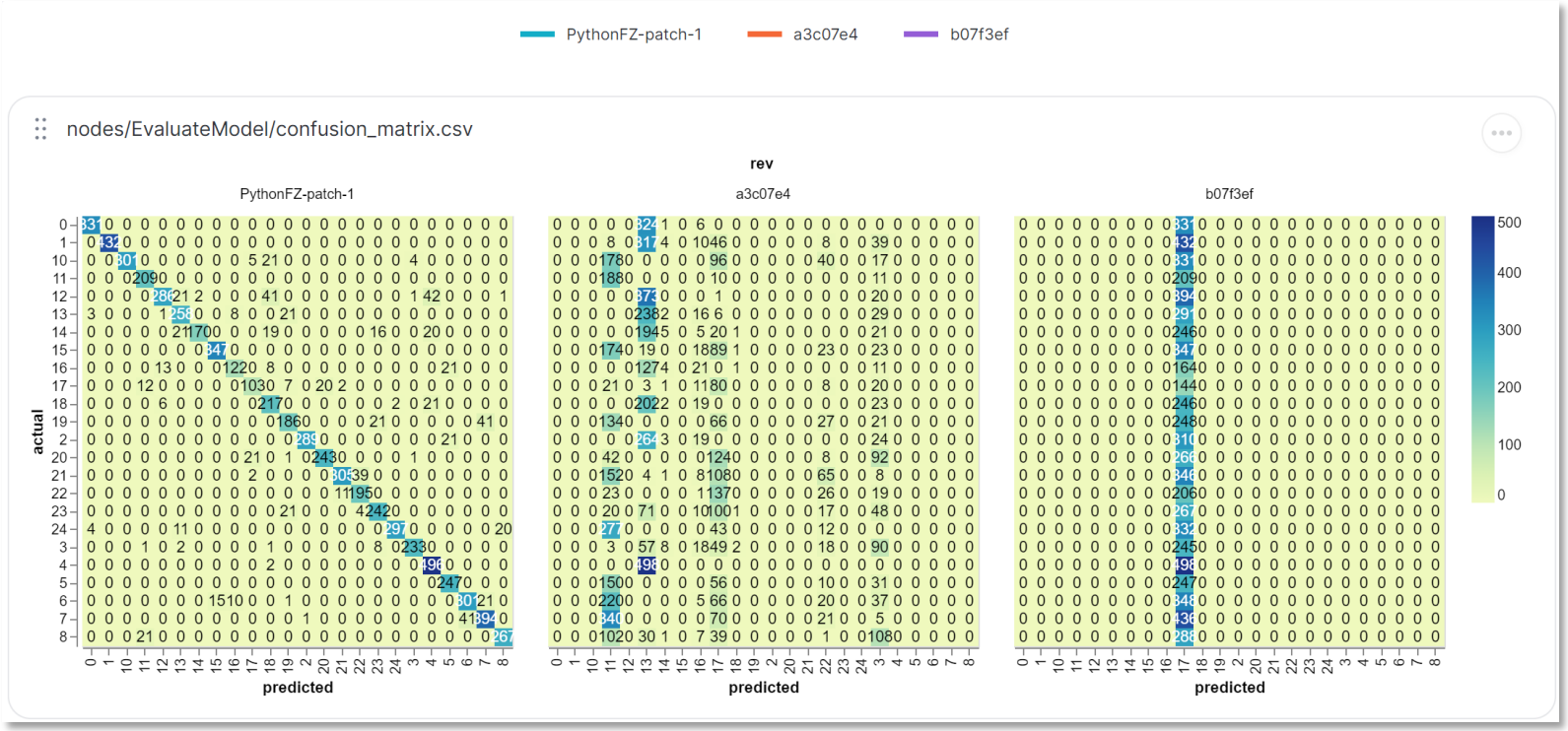
+ Add a model

Filters | Columns | Plots | Compare | Trends

Commit	Created	Message	CML	...nodes/ModelEvaluation/metrics_no_cac		params.yaml GradientBoosting		
				acc	f1_score	learning_rate	max_depth	n_estimators
Model_1-cc6a98b-904ee50-6cff8c4 inherited from main View PR								
Model_1-cc6a...	May 05, 2022	CML auto commit		0.96491	0.96970	0.05	12	1000
e2206b2	May 05, 2022	Update params.yaml:Gradi...		0.97368	0.97710	0.05	12	1000
Model_1-cc6a...	May 03, 2022	CML auto commit		0.97368	0.97710	0.01	5	1000

Data Version Control

Iterative Studio



Data Version Control

Additional features

→ Planning of experiments

Queue multiple runs and execute them sequentally or in parallel

→ Parameters in a YAML file

Workflow configuration in a single file that allows parameters studies

→ Set up a GitHub Action that re-runs the workflow

Continuous Machine Learning

DevOps for workflows

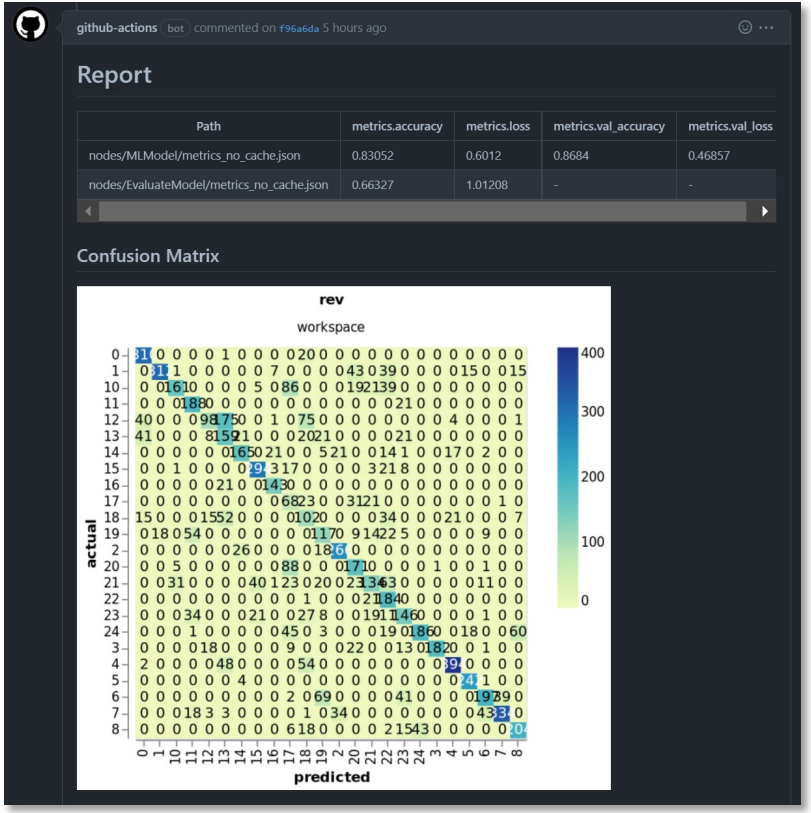
- **CI/CD pipeline** to execute workflow upon changes
  - Utilizes GitHub/GitLab Actions
  - Execution in a web- or self-hosted runner
- Submits a **report** to the repository
  - Contains results of the run
  - Commented on a Pull Request

Continuous Machine Learning

DevOps for workflows

- Reports include
  - → Metrics
  - → Text
  - → Tables
  - → Plots

- Resulting data pushed to file storage
- Workflow state pushed to repository



Outlook

→ Application of DVC

Already used in Molecular Dynamics simulations and Kinetic Modeling of enzyme reactions

→ ZnTrack – A Python wrapper

Library developed by Fabian Zills (ICP Stuttgart) for a seamless Python integration

→ From Web-App to Dataverse

Publish the best experiments from Iterative Studio

Resources

- **Data Version Control** – https://dvc.org
- **CML** – https://cml.dev
- **ML Example** – https://github.com/JR-1991/MLSession_Boosting_DVC
- **ZnTrack** – https://zntrack.readthedocs.io/en/latest/

University of Stuttgart
Germany

Thank you!

Jan Range

e-mail jan.range@simtech.uni-stuttgart.de

phone +49 (0) 711 685-60095

www. simtech.uni-stuttgart.de/exc/research-data-management/

University of Stuttgart
EXC2075 Data-Integrated Simulation Science