Expert Interview

Methodology Proposal
for Active Learning Projects

Universität
Augsburg
University

GET TO KNOW

1 Your Profile

- Current team role / position / job-activities
- ML project scope (team size / company size)
- Experience with Active Learning in the past
- Active Learning influence in the future

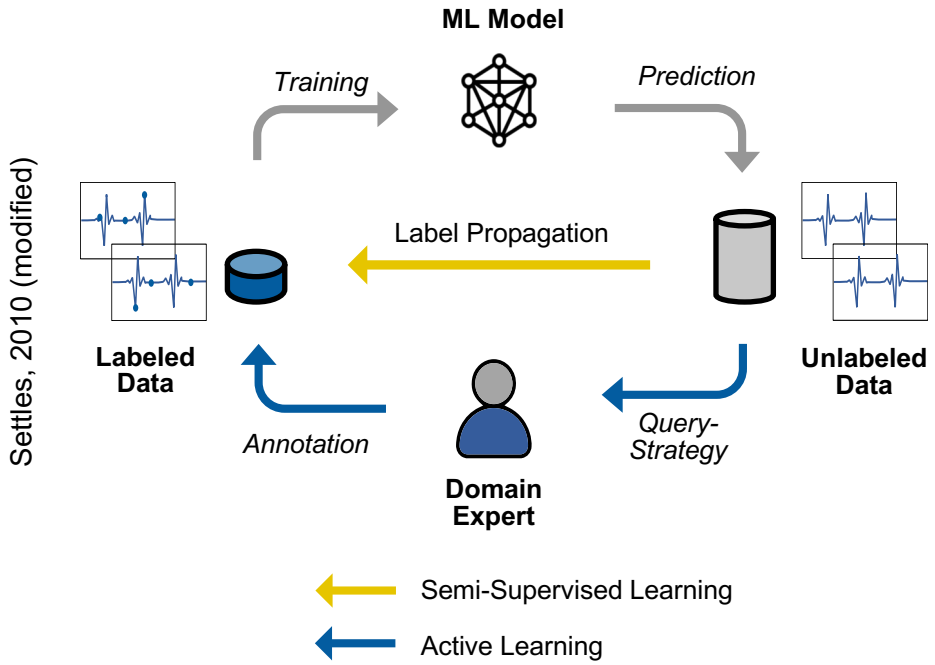
2 Maturity Level Inventory

		Level A	Level B	Level C	Level D
TECHNOLOGY	**What does your current toolchain look like?**	• Collaboration Tools to Chat • Poor SCM • Untracked Artifacts • No Automation • Manual Build	• Enhanced Use of Collaboration Tools • Monitoring Tools • Standardized SCM • Using Artifact Management Tools • Standardized Builds	• Full Use of Collaboration Tools • Integrated Monitoring • Team/Toolset Integration • Automated Builds & Deployment • Using Analytics-Tools	• Toolchain / Pipeline as Product • Fully Automated Deployments • Automated Test-Environments • Integrated Resilience
PROZESS	**Can you outline your current MLOps process?**	• Much Manual / Hand-Off Work • Ad Hoc Development • Heavily Data Science driven • No CI/CD, manual Delivery • Stand-alone solutions • “Trial and error” Testing	• Requirement Management • Waterfall • Standardized Integration and Delivery, Manual Release • Modularity • Manual Testing	• Agile Development • Automated Deliveries (Code, Data & Model) • Integrated Testing • Integrated Reporting	• Lean Development • Continuous Deliveries • Predictive Pipeline Maintenance
PEOPLE	**Do you / your team work according to a certain methodology?**	• Skillset - Organized • Knowledge-Silos • Poor Communication • No Priority-Awareness • Low Innovation	• Issue-based working • Semi-Cooperative • Written Knowledge • Regular Communication • Innovation by Requirement	• Automated Documentation • Knowledge Management • Fast Feedback-Loops • Continuous Education • Innovation Strategy	• Intra-Team Knowledge Transfer • Consulting other Teams • Ownership Mindset • Innovation is business vision

Maturity Level Classification follows Microsoft’s Azure Proposal 2021;
Expanded and restructed by several other sources

QUICK BRIEFING
Active Learning and Challenges in its Implementation

Active Learning: Interactively query new labels from user



Software/ML-Engineering Concepts for Active Learning Projects

- Uniting traditional Software Development and Active Learning Requirements
- Need for a methodology that enables Software Developers and ML-Experts to collaborate efficiently

ACTIVE LEARNING OPERATIONS LIFE CYCLE
Synthesize DevOps, DataOps and MLOps

1

Continuous Integration (CI)

- Mapping Devs' working to a shared Mainline
- Standardized workflows and code pipelines, automated builds, …

2

Continuous Delivery (CD)

- Reliably released software at any time
- Highly automated deployment workflow

3

Continuous Labelling (CL)

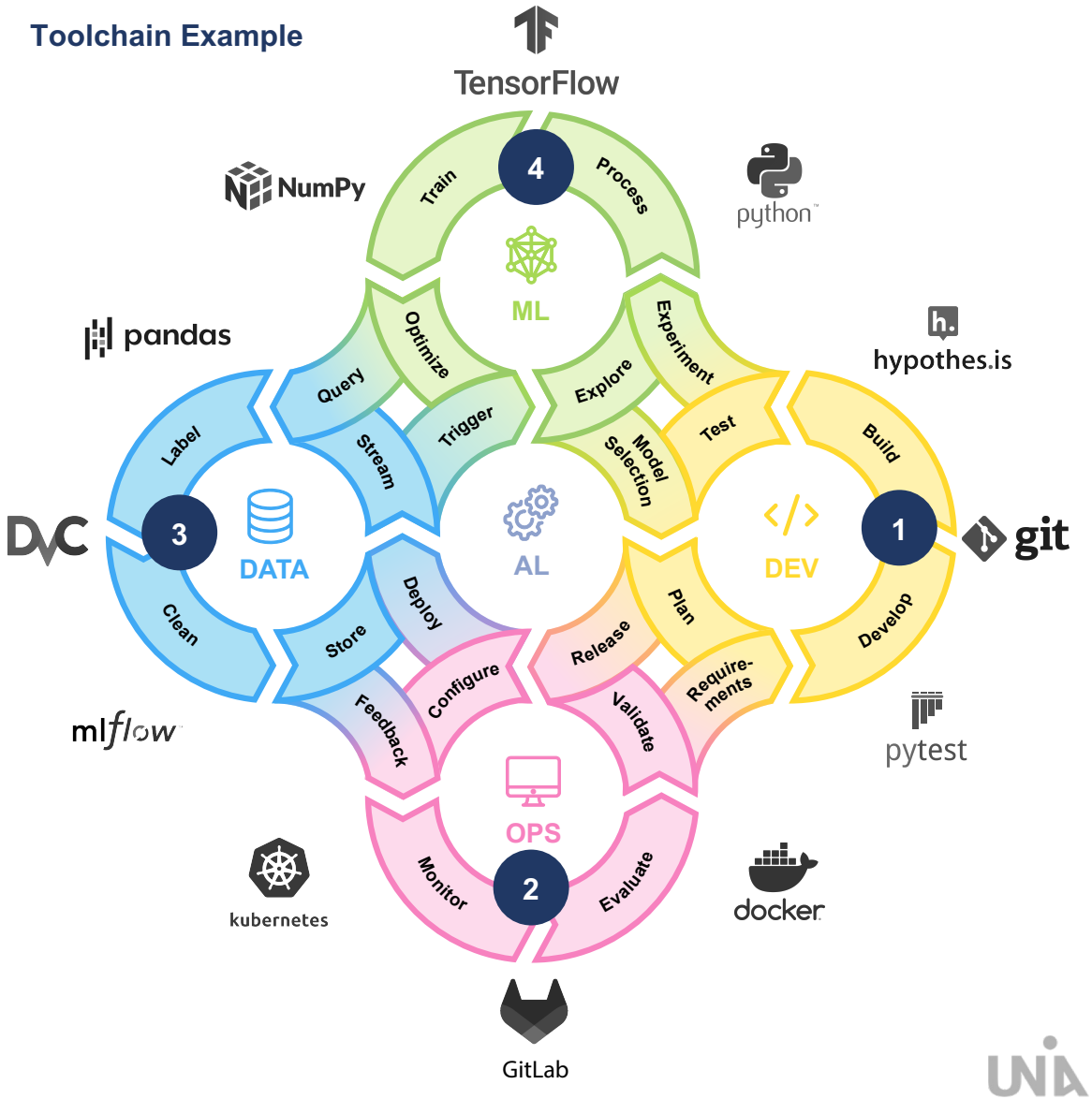
- Automated and standardized integration of (human as well as self-supervised) label acquiese
- Recurring feedback

4

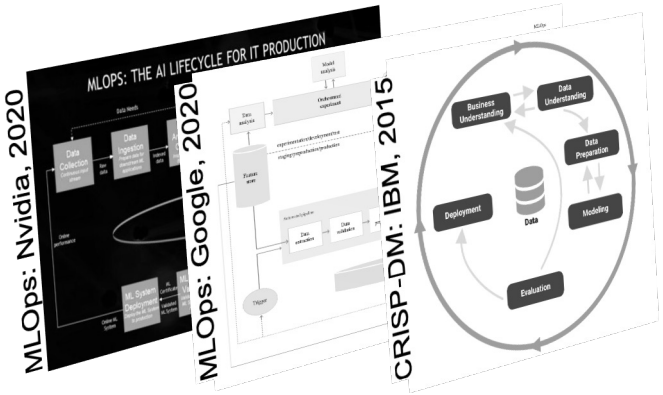
Continuous Training (CT)

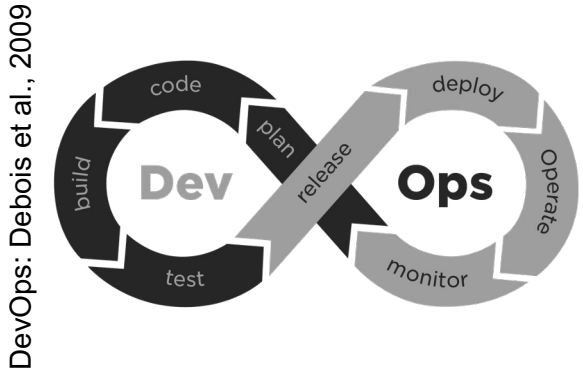
- Automatically (Re-)Trigger (scheduled) Training Pipelines
- Machine-Learning Experiments

Toolchain Example



ACTIVE LEARNING OPERATIONS LIFE CYCLE
Transform ALOps into an agile Development Process





ML development process
"What" has a team to do next
but not on the *"How"* to do it

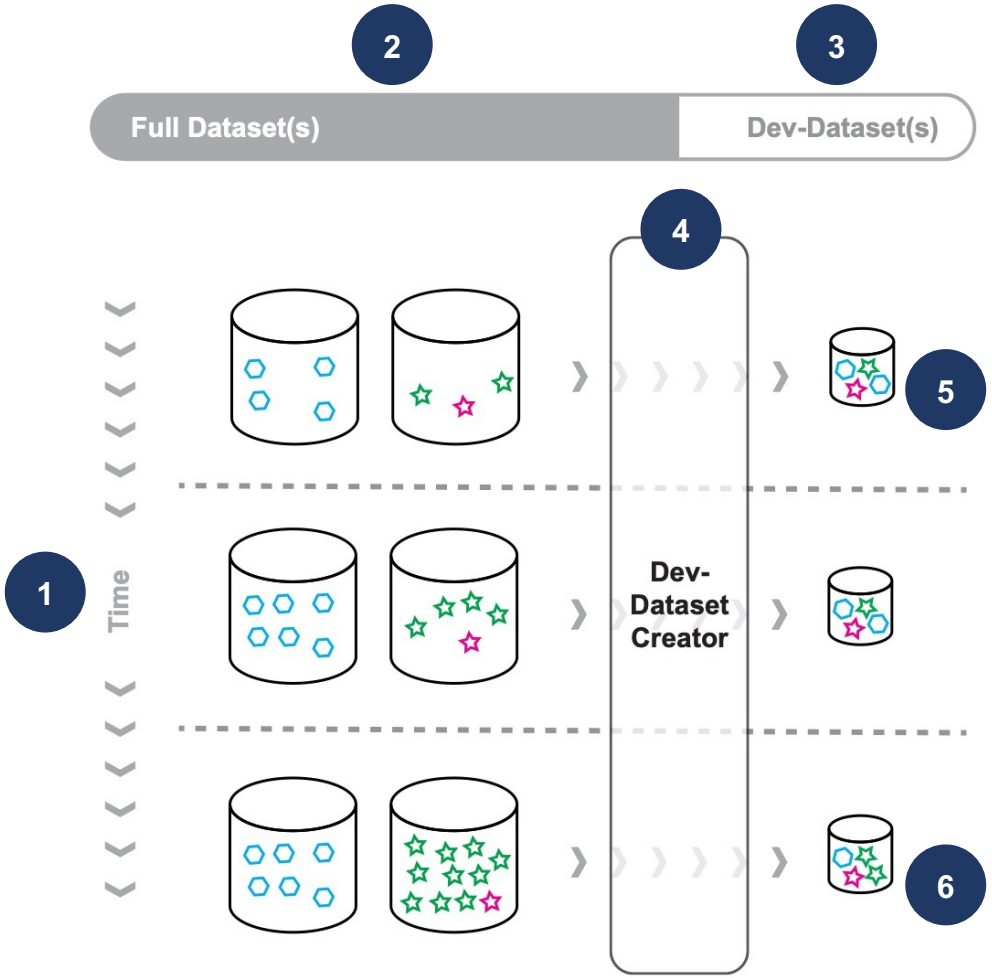
SE development process
"How" is already answered
by methodologies like *"Git Flow"*

Main difference for Active Learning Projects
Not model- but data-centric ML
Data more volatile than the code
High complexity due to flowing & constantly changing data

Lots of Requirements:

- Versioning and Reproducibility of code and (ever changing) data
- Quality assurance requirements of code and (ever changing) data
- General software-development requirements like fast feedback loops
- …

DEVELOPMENT METHODOLOGY PROPOSAL
Principle I: Development Dataset(s)



1 Basics
- Tracked by a data version control system
- Separate data repositories for each data source
- Data resides on a data remote

2 Full Dataset(s)
- May consist of one or more data sources
- Not not only “evolve” over time, but they change all the time
- In the Active Learning case, this includes both unlabeled and labeled samples

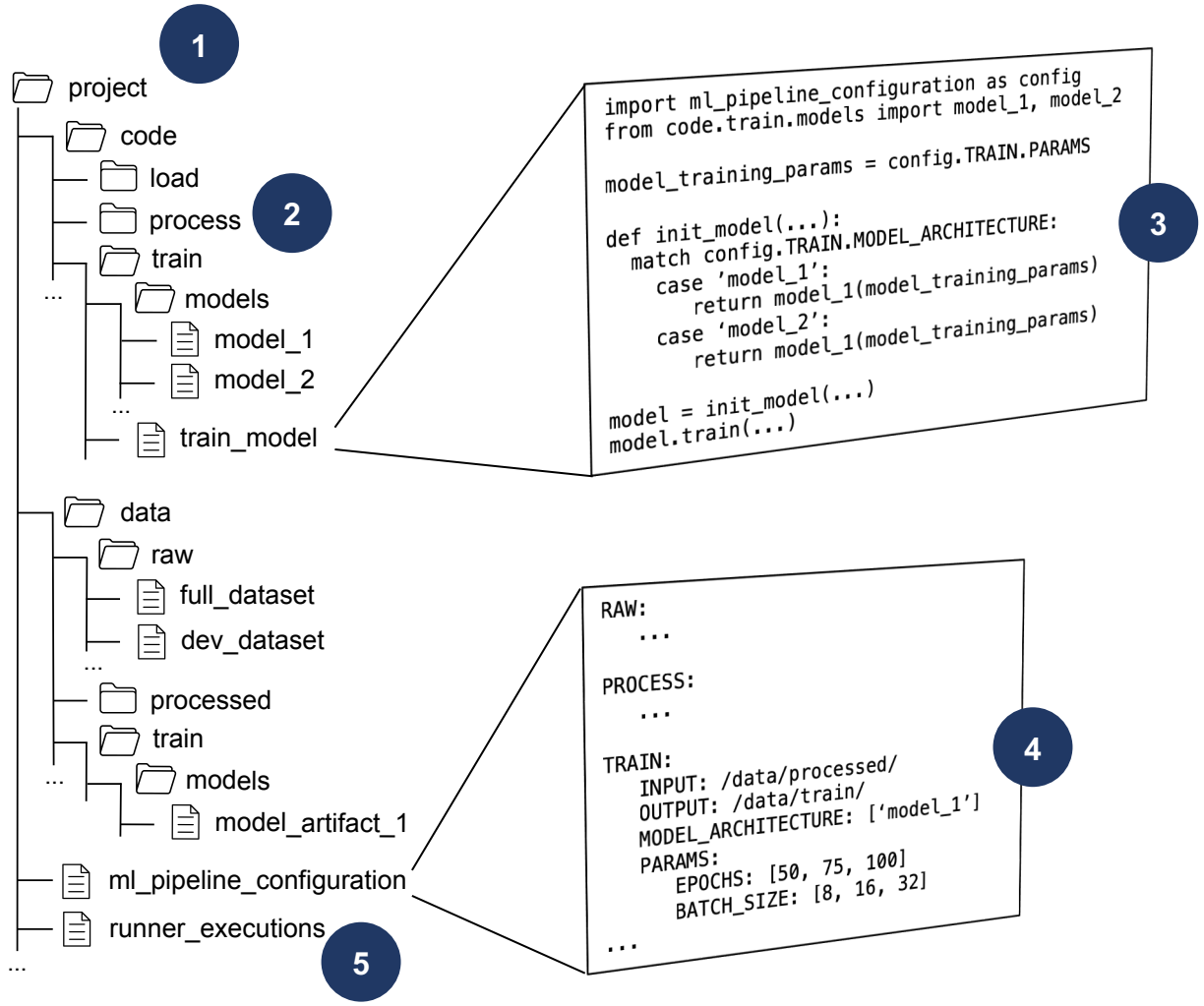
3 Development Dataset(s)
- Small subset of the original data for fast feedback-loops
- ML pipeline code can be executed on it in just a few minutes
- Dev Dataset is intended not only for specific runners (more on later slides) but mainly for the developers local client

4 Automated Creation
- Sub-Sampling must be automated
- Triggering can be either continuous or manual
- However, the procedure is automated, ideally a separate project or module in the respective data registries

5 Requirements to Dev-Dataset(s)
- Data distribution should approximately match to the data distribution of the full data set(s)
- Labeled & unlabeled data, as well as train/test/validation splits are also given
- Outliers and corrupt data points must be included

6 Approx. current Live Data
- If requirements are not longer met, dev-dataset(s) must be redefined
- A significantly changed data distribution must be mapped in the dev-dataset

DEVELOPMENT METHODOLOGY PROPOSAL
Principle II: Modular ML Pipeline Code



1 **Basics**
- Project code is tracked by a Source Control System
- Simplifies collaboration and provide basis for reproducibility

2 **ML Pipeline Stages**
- ML data flow consists of several phases (load, process, train, evaluate, …) or more complex (encode, query, transform, …)
- Each of this phases refines the data
- These phases must be identified in each ML-Project and divided into appropriate stages
- Stages should be reflected in respective code modules
- Developers can work on different stages in parallel

3 **Configurable Source Code**
- Stages should be configurable via function flags
- Single modules must therefore be implemented in a way they can handle parameterization and configuration from their related ML Pipeline Stage

4 **ML Pipeline Definition and Configuration**
- ML Pipeline Configuration must be stored in files tracked by the source control system
- Contains pipeline stages with dependencies between them
- Contains adjustable parameters, like a ”control panel” for pipeline
- List of allowed values for each parameter (and configuration combinations) enable automated testing

5 **Pipeline Execution**
- Goal is, that pipeline code runs on developers client locally and on remote through runner
- Execution instructions (data set(s) specification, …) and environment variables are tracked by the SCM

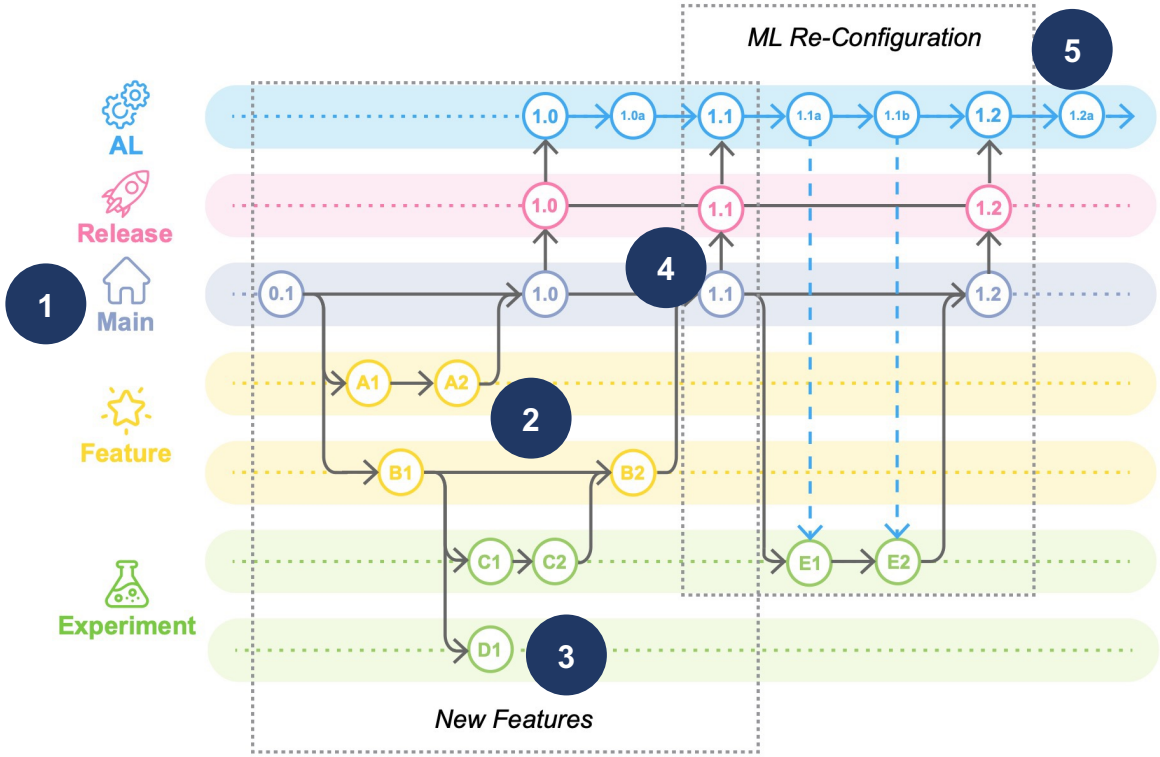
DEVELOPMENT METHODOLOGY PROPOSAL
Principle III: CI/CD/CT Runner Usage

Minimal Infrastructure Setup

1. Developer provide (dev-)dataset(s) and get artifacts from experiments or production deployment
2. Developer commits Code to SCM and get progress information
3. SCM-Server trigger jobs and runner (e.g. on GPU-Server) provide progress-information
4. Runner pulls related dataset(s) (full or dev, belonging to the job) and push artifacts (e.g. trained model, processed data, …)
5. Data server provides queried artifacts and stores new data & labels, can be extended by a feature, model registry, …
6. Runner ensure automated rollout of (re-)deployment, trigger manual training-jobs
7. Provide related release

DEVELOPMENT METHODOLOGY PROPOSAL
Git Flow for Active Learning

Branching Workflow: Software- and Data-Branches



1 Main Branch
- Contains most feature complete version (SW and Model)

2 Feature Branches
- Follow established concept from traditional SW projects
- Implementation of new features, e.g. model architecture, data augmentation, …
- CI-Runner operating in the scope of the development dataset for fast feedback

3 Experiment Branches
- Developer create branch with code of new feature
- Experiment Branches Runners have -access to actual full data set(s)
- Experiment Branches can be used to test new ML-Configurations
- Experiments are reproducible and traceable for the whole team

4 Release Branch
- When the ML configuration or new feature is accepted by a code review (merge request), it is propagated to the release branch and deployed

5 Active Learning Branch
- (Data-) Branch in scope of the current data pools, which is triggered continuously or automatically by defined actions
- Contains data version information that is potentially newer than the current software release version
- Data version information can also be merged into any experiment branches so that the runner pulls this data version state from the data remote

ASSESSMENT
Evaluate proposed Development Methodology

1

TECHNOLOGY
For which MLOps principles do you see an technical-dept enabler through the presented development approach?
- SCM and Artifact Management: Versioning Data, Code and Model
- Automated Testing along Data, Code and Model
- Reproducibility (of Experiments) along Data, Code and ML Artifacts
- Automation of tasks related to Data, Code and Model (e.g. (re-) training, rollout/rollbacks of deployments, …)
- Monitoring of Data, Code and Model
- Methodology has hard dependencies / is agnostic
- Documentation

2

PROCESS
How do you evaluate the proposed development methodology according to process requirements?
- Agility of Development: Facilitate teamwork for groups of multiple developers
- Quality assessments (Code / Data / Model)
- Scheduling, management and scaling of (computing-) resources
- Completeness / Missing Steps in process
- Concerns about principles (modular pipeline code, dev-dataset, runner usage)
- Possible contributes to the audibility of AI/ML applications

3

PEOPLE
Where do you see the biggest gaps and opportunities for you / your team?
- Influencing people mindset / workflow (Data awareness development, …)
- Implementation is helpful / extreme overhead for me and my team
- Can / Can’t imagine use of method in an possible Active Learning Project

THANK YOU FOR YOUR TIME.

Universität Augsburg
Fakultät für Angewandte Informatik

Software Methodologies for Distributed Systems
Institute for Computer Science
University of Augsburg
Universitätstraße 6a
86159 Augsburg

funded by

Federal Ministry of Education and Research