



DARE
DIGITAL LIFELONG PREVENTION

CODE NO. PNC0000002

Spoke 1 Deliverable

S1.D4.2

Best practices and tools for building ML-based systems

This research is co-funded by the Ministry of University and Research within the Complementary National Plan PNC-I.1 "Research initiatives for innovative technologies and pathways in the health and welfare sector"
D.D. 931 of 06/06/2022, PNC0000002 DARE - Digital Lifelong Prevention



S1.D4.2 Best practices and tools for building ML-based systems

Deliverable information

Spoke number and title	Spoke 1 - Enabling factors and technologies for digital prevention
WP number and title	WP4 - Technology and Analytics
Related task(s)	Task 4.2 - Model integration and deployment
Lead beneficiary	UNIBA
Contributing beneficiaries	-
Dissemination level	Public, fully open
Due date	14/05/2024
Actual date of delivery	11/04/2024
Author(s)	Andrea Basile (UNIBA), Fabio Calefato (UNIBA), Filippo Lanubile (UNIBA), Giulio Mallardi (UNIBA), Luigi Quaranta (UNIBA)
Contributors	-
Quality Assurance	Stefano Cagnoni (UNIPR), Federico Chesani (UNIBO)

Document history

Version	Date	Author(s) /Reviewer(s) (Beneficiary)	Description
0.1	15/03/2024	Andrea Basile, Fabio Calefato, Filippo Lanubile, Giulio Mallardi, Luigi Quaranta (UNIBA)	First draft
0.2	08/04/2024	Stefano Cagnoni (UNIPR)	Revision
0.3	09/04/2024	Federico Chesani (UNIBO)	Revision
1.0	11/04/2024	Andrea Basile, Fabio Calefato, Filippo Lanubile, Giulio Mallardi, Luigi Quaranta (UNIBA)	Final document

Disclaimer

This publication reflects only the authors' views, and the Funding Agency is not liable for any use that may be made of the information contained herein.

Table of Contents

<i>Publishable summary</i>	6
1. <i>Systematic Literature Review on MLOps in Healthcare</i>	7
1.1. Background	7
1.2. Research Questions	8
1.3. Query Definition	8
1.3.1. Identification of search terms.	8
1.3.2. Initial set of search terms	9
1.3.3. Refinement of the search terms	9
1.3.4. Definition of the final query.....	9
1.4. Data Collection	10
1.4.1. Selection criterion	11
1.5. Data Extraction & Analysis	12
1.6. Results	15
1.6.1. RQ1: Which MLOps practices are adopted in healthcare?	15
1.6.2. RQ2: Which MLOps tools are used in healthcare?	16
1.6.3. RQ3: Which workflow stages are supported by MLOps in healthcare?	17
1.6.4. RQ4: Which medical specialties are covered by MLOps?.....	17
1.7. MLOps adoption in Healthcare: Challenges and Mitigations	18
1.7.1. Protected data management.....	18
1.7.2. Limited data retention.....	19
1.7.3. Clinical validation report	19
1.7.4. Product Validation.....	19
2. <i>MLOps solutions</i>	20
2.1. Ensuring the reproducibility of ML experiments	21
2.2. Fostering quality assurance in ML projects	23



2.3. Development of APIs for ML models	25
2.4. Delivery of ML components	25
2.5. Monitoring of ML components	26
<i>List of selected papers</i>	28
<i>Bibliography</i>	30

Publishable summary

An ML-based system is a software artifact that incorporates machine learning (ML) components to enable various functions (Sculley et al., 2015). A core part of these systems is the ML model. Significant effort is generally required to integrate ML models into larger systems, which goes well beyond model training and testing. Other important tasks to consider include data collection and versioning, experiment tracking, model deployment, and model monitoring. Amershi et al. (2019) describe these activities as a pipeline to build ML models (see Figure 1).

The complexity of this model-building pipeline has resulted in efforts to develop mature practices and tools for the deployment and maintenance of ML components, giving rise to a research and practice field known as MLOps (Machine Learning Operations). Rooted in software engineering and inspired by DevOps (Bass et al., 2015), MLOps emphasizes process automation to achieve continuous delivery of ML models within ML-enabled systems (Treveil et al., 2020). Among the several objectives of MLOps, there is the facilitation of several non-functional requirements of ML-enabled systems (e.g., reproducibility, explainability, fairness), among which system security is considered of growing importance, especially in safety- or mission-critical domains such as healthcare.

Task 4.2 aims to analyze the current state of the art and identify (i) recommended best practices and (ii) technical solutions for implementing machine learning pipelines that support the development of ML-based systems. To achieve the first goal, we conducted a systematic review of the literature on MLOps practices and tools specifically focused on the healthcare domain. To fulfill the second objective, we reviewed the most popular MLOps solutions, focusing on those available as Free and Open-Source Software (FOSS). A well-designed combination of the reviewed tools has the potential to improve the development and maintenance of a wide range of ML-based components, ensuring that they meet the stringent quality standards required in safety-critical domains such as healthcare.

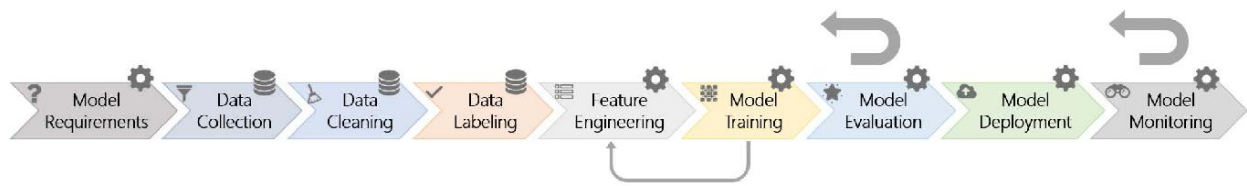


Figure 1 The stages of the typical machine learning pipeline (Amershi et al., 2019)

1. Systematic Literature Review on MLOps in Healthcare

A systematic review of the literature is a comprehensive and structured approach to summarizing and analyzing existing research on a specific topic aimed at providing an overview of the literature (Kitchenham et al. 2009). Systematic reviews are useful for both researchers and practitioners, since they provide summaries of the state of the art in a given area from both viewpoints.

To execute our review of the literature of MLOps practices in the healthcare domain, we first identified a comprehensive set of pertinent search terms. Subsequently, we collected the search results and meticulously sifted them in alignment with our research objectives. Finally, we conducted a thorough analysis of the refined corpus, focusing on discerning the MLOps practices and tools adopted in healthcare, delineating the machine learning workflow stages influenced by MLOps, and extracting insights regarding the medical specialties highlighted in each article.

1.1. Background

In the context of our research, we draw on insights from several key studies to inform our understanding of MLOps practices and tools. Amershi et al. 2019 provide a comprehensive overview of the typical stages of machine learning workflows, which we use to as a reference model for MLOps practices and tools. By doing so, we gain a deeper understanding of how MLOps fits within the broader landscape of machine learning development and deployment. Additionally, Kreuzberger et al. 2022 offer an in-depth exploration of MLOps practices, including their definition, architecture, and key components. This foundational knowledge serves as a framework for understanding the principles and methodologies that underlie the implementation of MLOps. Furthermore, Symeonidis et al. 2022 present a detailed taxonomy of MLOps tools and challenges, which

helps us categorize and analyze the diverse range of tools identified in our systematic literature review (SLR). Using these insights, we can systematically assess the landscape of MLOps tools and practices, considering their functionalities and the challenges they address.

1.2. Research Questions

We define the following research questions, which we aim to answer by analyzing the articles retrieved.

RQ1: *Which MLOps practices are adopted in healthcare?*

RQ2: *Which MLOps tools are used in healthcare?*

RQ3: *Which workflow stages are supported by MLOps in healthcare?*

RQ4: *Which are the medical specialties covered by MLOps?*

The first two questions aim to understand respectively the techniques and the available software that can help practitioners achieve better management of machine learning workflow catching possible gaps or peculiarities in the context of healthcare.

The goal of the third question is to identify which stages of a typical machine learning workflow are supported by MLOps practices and tools; understanding the stages that receive less support would give us some hints on future directions to further develop MLOps in healthcare.

The fourth research is mostly explorative and aims to assess what are the specialties encountered in the literature giving a general overview of the real case scenario faced by the MLOps framework.

1.3. Query Definition

1.3.1. Identification of search terms.

The initial phase of this SLR involved exploring articles using a preliminary set of terms composed of words commonly encountered in articles related to software engineering and their synonyms. Subsequently, we refined our search by identifying additional terms to specifically target papers focused on operationalizing systems incorporating machine learning. Finally, we formulated the final query to ensure the selection of precise articles that meet our criteria.

1.3.2. Initial set of search terms

Our initial set of search terms is shown below. The terms included in the query serve the purpose of filtering articles relevant to the intersection of software engineering practices and machine learning applications within the healthcare domain.

We chose two search engines. We opted for Scopus because it provides convenient access to papers and their metadata via API, thus simplifying the process of conducting and replicating SLRs. PubMed, on the other hand, is primarily oriented towards biomedical publications.

```
TITLE-ABS-KEY (("AIOps OR "DataOps" OR "MLOps" OR "MLHOps" OR "data  
pipeline" OR "model pipeline" OR "model deployment" OR "deployment  
pipeline" OR "ML pipeline" OR "machine learning pipeline"))
```

AND

```
("healthcare" OR "health" OR "clinical" OR "medicine" OR "medical" OR  
"wellness")) AND PUBYEAR > 2019 AND PUBYEAR < 2025 AND (LIMIT-TO (LANGUAGE,  
"English"))
```

With this search query, we initially obtained a total of 690 results on Scopus and 271 on PubMed. However, because our review of the literature is specifically centered on MLOps, we opted to refine the query to extract data more precisely, incorporating terms related to operationalizing machine learning within the healthcare domain.

1.3.3. Refinement of the search terms

We removed terms that could be easily confused with MLOps potentially yielding out-of-scope results, thereby undermining the review. We specifically focused on terms that explicitly refer to operationalizing machine learning workflows within the healthcare domain. To enhance precision, we added terms sourced from the reviewed articles and supplemented them with insights from our background knowledge.

1.3.4. Definition of the final query

After several iterations, the final query for our SLR became:

TITLE-ABS-KEY (("MLOps" OR "CD4ML" OR "Operational Machine Learning" OR ("CI/CD" AND ("ML" OR "Machine Learning"))))

AND

("healthcare" OR "health" OR "clinical" OR "medicine" OR "medical" OR "wellness")) AND PUBYEAR > 2014 AND PUBYEAR < 2025

1.4. Data Collection

The search with the final query was conducted on January 22, 2024, yielding 34 results on Scopus and 3 on PubMed. However, the 3 results were found on PubMed were already include in the set retrieved from Scopus. Thus, the total number of unique results is 34. To retrieve only relevant information, we decided to fetch only peer-reviewed papers, not including gray literature papers (i.e., preprints). Details on the number of search results retrieved from each query execution are reported in Figure 2. The initial analysis phases returned 22 potentially interesting results based on the analysis of title, abstract, and keywords.

Given the limited number of results returned by the final query, we decided not to apply any further filtering criterion and, therefore, to analyze each of the 22 articles.

1.4.1. Selection criterion

The selection and exclusion criteria that we used to decide, after reading the entire document, whether to accept each search result are summarized in Table 1.

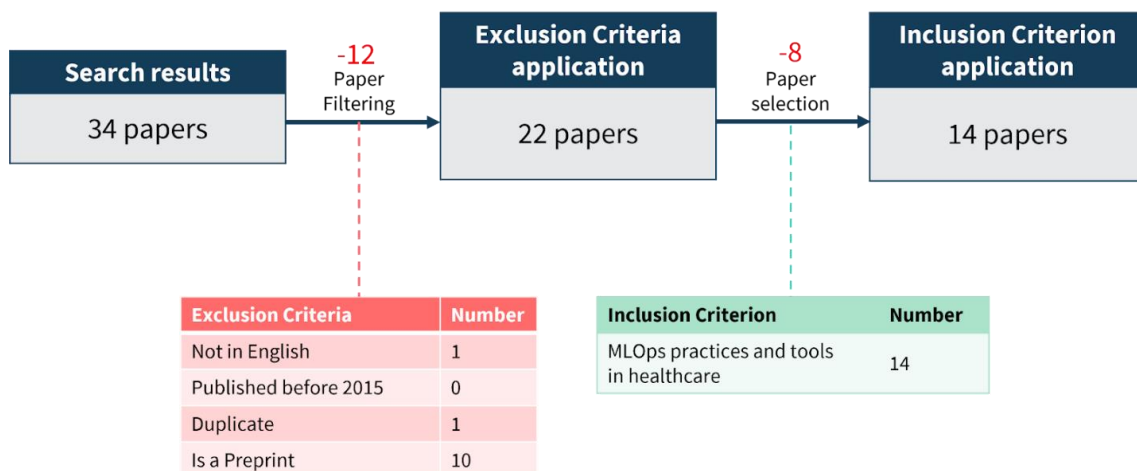


Figure 2 Infographic summarizing the data collection process for the SLR on MLOps in healthcare.

Table 1 List of inclusion and exclusion criteria applied to filter search results in the SLR on MLOps in healthcare, along with their rationale.

Search results inclusion criterion.
The search result contains specific recommendations or take-away points to implement an ML pipeline in the healthcare domain.
Search results exclusion criteria.
The search result is not written in English.
The search result is not a peer-reviewed article, book (e.g., preprint).
The search result is a duplicate.
The search result is published before 2015

The inclusion criterion involved selecting articles reporting studies that use MLOps practices and tools in the healthcare domain. Conversely, the exclusion criteria encompassed papers not written in English, published before 2015—referring to Sculley et al. 2015’s paper, which we consider the first to highlight the problem of hidden technical debt in machine learning—duplicates, or preprints.

The application of the filtering criteria described so far to the search results retrieved from each search engine left us with a corpus of 14 articles, reported in Table 2.

Table 2 List of inclusion and exclusion criteria applied to filter search results in the SLR on MLOps in healthcare.

Search results inclusion criterion.
The search result contains specific recommendations or take-away points to implement an ML pipeline in the healthcare domain.
Search results exclusion criteria.
The search result is not written in English.
The search result is not a peer-reviewed article, book (e.g., preprint).
The search result is a duplicate.
The search result is published before 2015

1.5. Data Extraction & Analysis

In our review, we collected text excerpts based on specific criteria to answer our research questions. We searched for mentions of MLOps practices and tools, focusing on evidence of their adoption or use in healthcare (RQ1, RQ2). We also noted where these practices or tools were referenced within the machine learning workflow stages (RQ3). To ensure relevance, we excluded vague references unsupported by evidence. Furthermore, we recorded the medical specialties discussed in each article (RQ4), providing information on where MLOps is applied in healthcare.

Two authors read the identified articles and extracted the relevant excerpts, tool mentions, and medical specialties from each article. Then, several plenary meetings involving the five authors were conducted, during which all the information extracted was scrutinized and potential disagreements among participants resolved through discussion until consensus was reached.

After consolidating the relevant excerpts, we employed a qualitative analysis method that involves the development of a codebook through open or closed coding. Codes were applied to relevant excerpts, except for medical specialties, which were assigned to the entire article rather than to excerpts.

Table 3 Articles that meet the inclusion criteria for the systematic review of the literature on MLOps in healthcare.

ID	Reference
[R1]	Bai, E., et al. "A Graphical Toolkit for Longitudinal Dataset Maintenance and Predictive Model Training in Health Care". Applied Clinical Informatics, 2021
[R2]	Granlund, T., et al. "Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product". SN Computer Science, vol. 2, fasc. 5, 2021
[R3]	Subramanya, K. N., et al. "Application of MLOps in Prediction of Lifestyle Diseases". ECS Transactions, vol. 107, fasc. 1, Institute of Physics, 2022, pp. 1191-98
[R4]	Berezsky, O., O. Pitsun, G. Melnyk, Y. Batko, B. Derysh, et al. "Application Of MLOps Practices For Biomedical Image Classification". CEUR Workshop Proc., a cura di Shakhovska N. et al., vol. 3302, CEUR-WS, 2022, pp. 69-77
[R5]	Mathew, N., e C. T. Joseph. "Applying Transfer Learning on 3D Brain Images and an MLOPS Study for Deployment". Int. Conf. Smart Comput. Commun.: Intell. Technol. Appl., ICSCC, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 541-47
[R6]	Jain, A., et al. "Brain Tumor Detection Using MLOps and Hybrid Multi-Cloud". IEEE Int. Conf. Blockchain Distrib. Syst. Secur., ICBDS, Institute of Electrical and Electronics Engineers Inc., 2022
[R7]	Stirbu, V., et al. "Continuous Design Control for Machine Learning in Certified Medical Systems". Software Quality Journal, vol. 31, fasc. 2, 2023, pp. 307-33
[R8]	Karácsony, T., et al. "Deepepil: Towards an Epileptologist-Friendly AI Enabled Seizure Classification Cloud System Based on Deep Learning Analysis of 3D Videos". BHI - IEEE EMBS Int. Conf. Biomed. Health Informatics, Proc., Institute of Electrical and Electronics Engineers Inc., 2021
[R9]	Lombardo, G., et al. "Digital Twin for Continual Learning in Location Based Services". Engineering Applications of Artificial Intelligence, vol. 127, 2024
[R10]	Koutsopoulos, K., et al. "Federated Machine Learning through Edge Ready Architectures with Privacy Preservation as a Service". Proc. - IEEE Future Networks World Forum, FNWF, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 347-50
[R11]	Verbiest, J. R., et al. "Gait Stride Length Estimation Using Embedded Machine Learning". Sensors, vol. 23, fasc. 16, 2023
[R12]	Meel, V., e A. Bodepudi. "Melatect: A Machine Learning Approach for Identifying Malignant Melanoma in Skin Growths". Proc SPIE Int Soc Opt Eng, a cura di Osten W. et al., vol. 12084, SPIE, 2022
[R13]	Berezsky, O., O. Pitsun, G. Melnyk, Y. Batko, P. Liashchynskyi, et al. "MLOps Approach for Automatic Segmentation of Biomedical Images". CEUR Workshop Proc., a cura di Shakhovska N. et al., vol. 3609, CEUR-WS, 2023, pp. 241-48
[R14]	Fergus, P., et al. "Pressure Ulcer Categorization and Reporting in Domiciliary Settings Using Deep Learning and Mobile Devices: A Clinical Trial to Evaluate End-to-End Performance". IEEE Access, vol. 11, 2023, pp. 65138-52

The practices were identified using closed coding based on the principles outlined by Kreuzberger et al. (2022), while the tools were identified through open coding, with tool names selected and then clustered according to a taxonomy adapted from Symeonidis et al. (2022). The excerpts referring to MLOps practices and tools were classified using closed coding, aligning with the machine learning workflow stages described by Amershi et al. (2019) and, lastly, the analyzed papers were classified according to the taxonomy presented by the Union of European Medical Specialists (UEMS) - Medical specialties.¹

1.6. Results

The analysis was conducted to address the research questions and provide information on potential challenges and mitigations affecting specific MLOps practices.

1.6.1. RQ1: Which MLOps practices are adopted in healthcare?

From the analyzed articles, it is evident that there is full coverage of MLOps practices, meaning that the practices applied are aligned with the ones taken from Lombardo et al. More details are available in the Practices and References (Table 3).

The most prominent insight that potentially influences MLOps practices is the strict dependence on regulations [R2, R7, R9]. *“When developing a medical device product, it is essential to clearly understand applicable regulatory requirements and determine a regulatory strategy accordingly from the beginning of the project [...] However, such continuous practices are not immediately compatible with regulatory requirements that may need authority involvement”* [R2]. These impacts result from numerous factors, including the need to manage parts of the pipeline in a private on-premises environment, stringent regulations on patient data, and the imperative for a stable, explainable, and deterministic model suitable for safe use in this sensitive domain. Moreover, Granlund et al. [R2] underscored the lack of explicit directives from regulatory bodies on compliance requirements, highlighting the novelty of these considerations.

¹ <https://www.uems.eu/about-us/medical-specialties>

Table 4 MLOps practices adopted in healthcare and references.

Practice	References
CI/CD automation	[R1, R4, R5, R7, R8, R9, R13, R2]
Workflow orchestration	[R1, R4, R5, R7, R9, R13, R14, R2]
Reproducibility	[R4, R5, R9, R12]
Versioning of data, code, model	[R10, R12, R2]
Collaboration	[R7, R8, R2]
Continuous ML training & evaluation	[R1, R7, R9, R11, R12, R2]
ML meta data tracking	[R2, R11, R1, R4, R5, R9, R13]
Continuous monitoring	[R4, R5, R7, R9, R12, R2]
Feedback loops	[R4, R7, R8, R2]

1.6.2. RQ2: Which MLOps tools are used in healthcare?

In Figure 3, we present the tools that emerged from the analyzed articles, grouped according to the taxonomy proposed by Symeonidis et al. 2022. In the exploration of tools, MLFlow emerges prominently as the most applied solution, as evidenced by its frequent references [R1, R5, R12, R2]. This underscores its importance in facilitating various aspects of the modeling and operationalization phase. However, it should be noted that despite the prevalence of MLFlow, a diverse array of tools exists that cater to this operationalization

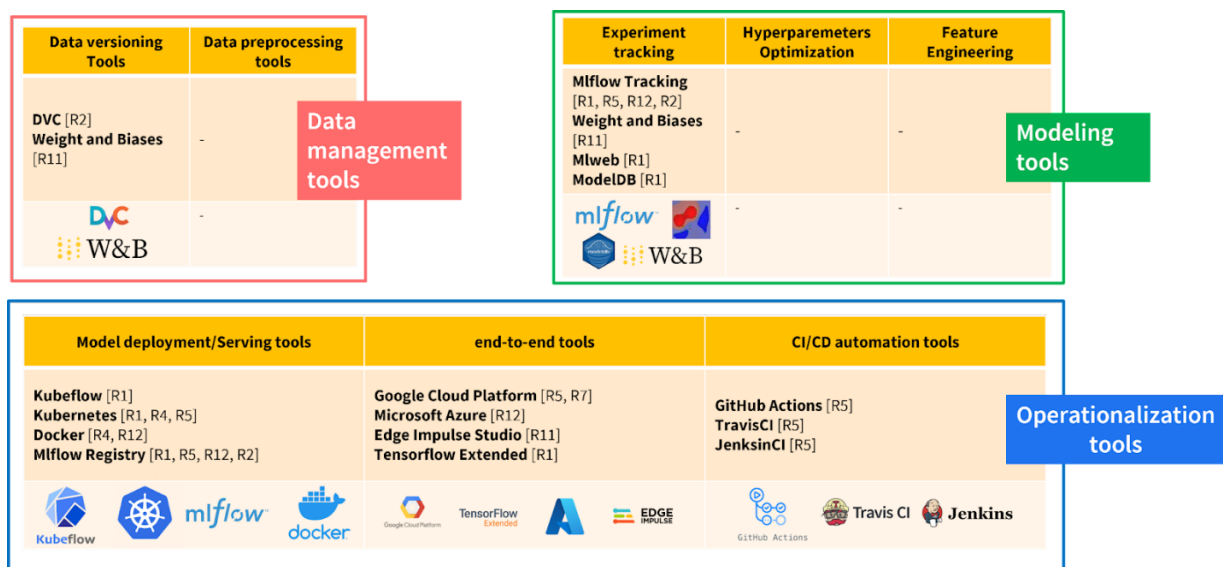


Figure 3 Tools found in the literature and grouped according to the adapted taxonomy from Symeonidis et al. 2022.

domain. Despite the abundance of options, it is apparent that the literature lacks significant mention of data preprocessing tools. Furthermore, one can notice the absence of hyperparameter optimization and feature engineering tools within the discourse, suggesting potential avenues for further investigation and development in these critical areas.

1.6.3. RQ3: Which workflow stages are supported by MLOps in healthcare?

As shown in Table 4, the ML workflow stages supported by MLOps practices and tools in healthcare seem to focus primarily on the phases from Model Training to Model Monitoring. Conversely, fewer articles address the initial stages of the ML workflow, particularly Data Labeling.

1.6.4. RQ4: Which medical specialties are covered by MLOps?

The European Union of Medical Specialists (UEMS) represents over 50 medical disciplines through various bodies and structures. The most important are the 43 Specialist Sections, representing independently recognized specialties. However, in our investigation, we could only fetch seven specialties represented in Figure 4 with related references.

Table 5 ML workflow stages supported by MLOps practices and references.

ML workflow stage	References
Model Requirements	[R7, R2]
Data Collection	[R1, R7, R13, R2]
Data Cleaning	[R4, R2]
Data Labeling	-
Feature Engineering	[R9]
Model Training	[R1, R4, R5, R7, R9, R10, R11, R12, R2]
Model Evaluation	[R2, R11, R1, R4, R5, R9, R13]
Model Deployment	[R4, R5, R7, R9, R11, R13, R14, R2]
Model Monitoring	[R7, R12, R2, R4, R5, R9, R8]

- Allergology
- Anaesthesiology
- Cardiology
- Cardiothoracic Surgery
- Child and Adolescent Psychiatry
- Clinical Neurophysiology
- Dermatology and Venereology
- Emergency Medicine
- Endocrinology
- Gastroenterology & hepatology
- **Geriatrics** [R14]
- Gynaecology & Obstetrics
- Infectious Diseases
- **Internal Medicine** [R5, R6, R12]
- Laboratory Medicine
- Medical Biopathology
- Medical Genetics
- Medical Microbiology
- Medical Oncology
- Nephrology
- **Neurology** [R2, R11]
- Neurosurgery
- Nuclear Medicine
- Occupational Medicine
- Ophthalmology
- Oro-Maxillo-Facial Surgery
- **Orthopaedics & Traumatology** [R2]
- Otorhinolaryngology
- Paediatric Surgery
- Paediatrics
- **Pathology** [R4, R13]
- Pharmacology
- Physical and Rehabilitation Medicine
- Plastic, Reconstructive and Aesthetic Surgery
- Pneumology
- Psychiatry
- **Public Health Medicine** [R3]
- **Radiology** [R10]
- Radiation Oncology and Radiotherapy
- Rheumatology
- Surgery
- Thoracic Surgery
- Urology
- Vascular Surgery

Figure 4 Medical Specialties Identified by UEMS. In bold, medical specialties emerged from the literature with corresponding references.

1.7. MLOps adoption in Healthcare: Challenges and Mitigations

In the following, we summarize our findings and elaborate on them in terms of the challenges currently faced when introducing MLOps in healthcare settings and related mitigation strategies.

1.7.1. Protected data management

Managing protected healthcare data presents significant challenges for MLOps practitioners, as exemplified by the Oravizio project [R2]. Due to legal and organizational policies, access to data for model development is highly limited. These constraints required developing and deploying MLOps pipelines within tightly controlled, on-premises environments, limiting scalability and agility. Restricted access to data affects data versioning, continuous learning, and evaluation. Access to historical data hinders the tracking of changes over time, while obtaining new data for continuous learning requires considerable time and effort. Evaluation may also suffer from limited access to labeled data. Adapting MLOps processes within these constraints requires careful planning and collaboration to ensure compliance while still deriving valuable insights from the data.

1.7.2. Limited data retention

Continuous learning and evaluation in MLOps are crucial to maintaining and improving machine learning models over time. However, they can be significantly affected by limited data retention, particularly in industries such as healthcare, where strict regulations govern the handling of patient data. Regulations often require that patient data used for retraining be available only for a specific amount of time, which can hinder the ability to continuously train and evaluate models using historical data. An approach to address this challenge is to generate synthetic data from historical data [R9]. By doing so, organizations can continue to train and evaluate models even when the original data is no longer available due to retention policies. This approach also helps avoid catastrophic forgetting, where models forget previously learned tasks when trained on new data. Implementing a robust system for managing and generating synthetic data can help ensure that MLOps practices like continuous learning and evaluation remain effective in environments with limited data retention policies.

1.7.3. Clinical validation report

Integrating clinical validation reports into continuous integration/continuous deployment (CI/CD) automation processes represents a crucial advancement in MLOps practices, particularly in the healthcare sector. Using model cards and automated generation techniques, such as those proposed in the references [R2, R7], MLOps teams can ensure that their machine learning models are technically competent and clinically validated. So, the performance of these models is evaluated not just in terms of traditional metrics but also through a rigorous clinical evaluation, which is essential to ensure patient safety and regulatory compliance in healthcare applications. However, a significant challenge in implementing such practices is managing protected health data, as clinical validation reports often contain sensitive patient information. Integrating clinical validation reports into CI/CD automation is a critical step forward, bridging the gap between technical excellence and clinical relevance in AI applications for healthcare.

1.7.4. Product Validation

With the term “locked” both FDA and the COCIR European Trade Association define an AI model *“that provides the same result each time the same input is applied to it. The*

design of the AI remains unchanged. Examples of locked AI are static look-up tables, decision trees, and complex classifiers”²

The requirement for ‘locked’ algorithms in AI-enabled medical devices poses challenges to CI/CD automation in several ways. Primarily, CI/CD automation relies on continuously integrating changes, testing them, and deploying them into production environments. However, the immutability of locked algorithms means that once a model is deployed, it cannot be modified or updated, which is against the iterative nature of CI/CD processes. Additionally, CI/CD automation typically involves automated testing at various stages of the pipeline to ensure the quality and reliability of the software being developed. With locked algorithms, the testing process may become more complex, as developers need to ensure that the model behaves as expected in all scenarios without the ability to adjust them based on feedback from testing. The requirement for locked algorithms in AI-enabled medical devices introduces constraints that can disrupt the continuous integration and delivery processes typically used in CI/CD automation. Therefore, careful planning and adaptation of CI/CD pipelines are required to meet regulatory requirements while ensuring efficient and reliable software delivery.

2. MLOps solutions

After conducting our systematic literature review on MLOps in healthcare, we embarked on a comprehensive search for MLOps tools. Our goal was to identify and review the most widely adopted solutions that could enhance MLOps workflows in the medical domain. Whenever possible, we placed particular emphasis on analyzing Free and Open Source Software (FOSS). In particular, we choose to focus on open-source solutions as they prevent vendor lock-in issues and ensure on-premises installation, two aspects that are critical in data privacy management. The results of our search are presented below, organized according to the MLOps practices facilitated by each solution:

- ensuring the reproducibility of ML experiments;
- fostering quality assurance (QA) in ML projects;
- development of Application Programming Interfaces (APIs) for ML models;

² [Discussion Paper “Proposed Regulatory Framework for Modifications to Artificial Intelligence/Machine Learning \(AI/ML\)-Based Software as a Medical Device \(SaMD\)”](#), published by US Food and

- delivery of ML components;
- monitoring of ML components.

2.1. Ensuring the reproducibility of ML experiments

Reproducibility is a crucial requirement in machine learning projects. Not only is it essential to achieve consistent performance in both production and laboratory environments, but it also enables the recovery and timely retraining of deployed models. However, achieving reproducibility in ML is a challenging task. In particular, in the healthcare domain, the reproducibility problem is exacerbated by the availability of datasets, which may become unavailable after a certain time (see Sect. 1.7.2 on limited data retention). This peculiarity makes it necessary to generate of privacy-preserving synthetic datasets that ensure data retention, and hence replicability, even after their expiration.

One of the first steps towards reproducibility is embracing version control. For code, the standard solution is Git³ – the de-facto standard version control system – employed in conjunction with popular code hosting platforms like GitHub⁴, GitLab⁵, and Bitbucket⁶. While GitHub is the most widely used platform for hosting Git repositories, GitLab offers the additional benefit of on-premises deployment solutions, providing organizations with the flexibility to host their code repositories internally. However, versioning data is more challenging than versioning code. Different data formats (e.g., text and images) require specialized versioning mechanisms, and storing and retrieving data is more difficult due to larger file sizes. These challenges can be overcome by using specialized tools such as DVC⁷, Git LFS⁸, LakeFS⁹, Dolt¹⁰, and Pachyderm¹¹. DVC, an open-source platform, is widely used to version large data files (datasets and models) and backing them up to cloud remotes. Git LFS (Large File Storage) is a Git extension that enables the versioning of large files by storing them outside of the Git repository. LakeFS is a data versioning platform that provides Git-like operations to manage data lakes. Dolt is an SQL database that supports

³ <https://git-scm.com>

⁴ <https://github.com>

⁵ <https://about.gitlab.com>

⁶ <https://bitbucket.org>

⁷ <https://dvc.org>

⁸ <https://git-lfs.com>

⁹ <https://lakefs.io>

¹⁰ <https://www.dolthub.com>

¹¹ <https://www.pachyderm.com>

version control and data collaboration using Git-like operations. Finally, Pachyderm is a data versioning and lineage platform specifically designed to manage machine learning data pipelines.

Experiment tracking is another key component of reproducibility. Being able to trace back experimental decisions is crucial to identifying and reproducing the best experimental paths. To support this practice, MLflow¹² Tracking, a popular open-source solution, is widely used. In addition to the Tracking module, MLflow provides a Registry module to save models in a centralized store. DagsHub¹³, a cloud hosting platform specifically designed for data scientists, can be employed as a remote for both DVC and MLflow, facilitating the management and storage of data files and experimental results. Alternatively, all MLflow modules can be installed and run on-premises. Several commercial alternatives are available for experiment tracking, such as Comet ML¹⁴, Neptune.ai¹⁵, and Weights & Biases.¹⁶ These platforms offer similar functionality to MLflow Tracking, including the ability to log and visualize various metrics, parameters, and artifacts during the model training process. Additionally, they also provide model registry features, allowing users to store and manage their trained models in a centralized location. Besides, if we consider library-specific solutions, a further popular option for experiment tracking is TensorBoard¹⁷, which is sometimes favored due to its tight integration with the TensorFlow library.

MLOps pipelines can be reproduced automatically using workflow orchestrators. Other than expediting the overall process, these tools are useful as they reduce the margin for human error. Workflow orchestrators allow data scientists to specify the computational steps included in ML pipelines and the related dependencies, information that is typically modeled as a directed acyclic graph (DAG). Besides supporting data and model versioning, DVC enables the definition of ML pipelines, which can be straightforwardly reproduced by executing a single terminal command. However, the automation of intricate ML pipelines in complex deployment scenarios can require the use of specialized tools. Apache Airflow¹⁸ is an open-source platform that uses DAGs to define the sequence and dependencies of

¹² <https://mlflow.org>

¹³ <https://dagshub.com/>

¹⁴ <https://www.comet.com>

¹⁵ <https://neptune.ai/>

¹⁶ <https://wandb.ai/>

¹⁷ <https://www.tensorflow.org/tensorboard>

¹⁸ <https://airflow.apache.org>

tasks, enabling efficient parallel processing and reduced manual intervention. It offers automation, scalability, and real-time monitoring capabilities, making it a powerful tool for managing complex ML pipelines. Kubeflow¹⁹ is a machine learning toolkit dedicated to simplifying the deployment and management of ML workflows on Kubernetes. It provides a set of services and UI aimed at creating and managing ML pipelines, focusing on end-to-end orchestration, easy experimentation, and easy reuse. Dagster²⁰ is another workflow management system that supports the creation, deployment, and management of complex data pipelines. As DVC, Dagster is a commercial tool with an open-source core. Finally, Taipy²¹ is a tool for building large data pipelines that optimizes performance through parallel processing and efficient resource allocation. It is particularly useful for teams that need to integrate data pipelines with a full-fledged GUI application, as it provides easy integration and seamless interaction capabilities.

2.2. Fostering quality assurance in ML projects

Previous research has revealed that the quality of code in experimental machine learning (ML) artifacts is generally poor, particularly in the case of computational notebooks (Pimentel et al., 2019; Wang et al., 2020). At the same time, model performance is known to be significantly affected by the quality of training data, which is often suboptimal in real-world scenarios. To ensure production-grade quality for artifacts developed in the laboratory, data scientists must modularize and test their code, as well as employ static analyzers.

Various quality assurance (QA) tools can be integrated into machine learning (ML) pipelines to improve the overall quality of code, data, and models. For static code analysis, linters like Pylint²², flake8²³, and Ruff²⁴ can be used to identify potential issues, enforce coding standards, and maintain code quality. These tools offer different features and configurations, allowing Python developers to choose the most suitable option based on their preferences and project requirements. For code testing, frameworks such as Pytest²⁵,

¹⁹ <https://www.kubeflow.org>

²⁰ <https://dagster.io>

²¹ <https://www.taipy.io>

²² <https://pylint.readthedocs.io/en/stable/>

²³ <https://flake8.pycqa.org/en/latest/>

²⁴ <https://docs.astral.sh/ruff/>

²⁵ <https://docs.pytest.org/en/8.0.x/>

unittest²⁶, nose²⁷, and doctest²⁸ provide robust solutions for writing and executing unit tests, enabling data scientists to validate their code's functionality and behavior. While Pytest and unittest are widely adopted testing frameworks in the Python ecosystem, nose and doctest offer alternative approaches with unique features and capabilities. Nose, for instance, provides more granular control over test discovery and execution, while doctest allows for testing code examples directly from docstrings. Additionally, specialized tools such as Pynblint²⁹ can be employed for static analysis of Jupyter notebooks, a widely used format for interactive code development and exploration in ML projects. Furthermore, to optimize energy efficiency, practitioners can track the CO2 emissions of their ML pipelines using dedicated libraries like Code Carbon³⁰, promoting sustainable practices in the development and deployment of ML solutions.

Quality assurance for data is more challenging than for code due to the variety of existing data formats, and there is no single tool that covers all possibilities. Great Expectations³¹ is an open-source framework that allows the definition of assertions on various properties of tabular data. When training datasets are not tabular, Great Expectations can still be used to check their quality, but a preliminary extraction of features into a data frame is required. Deequ³² is another open-source library that can be used as an alternative to Great Expectations, particularly in the context of Apache Spark-based data pipelines. It offers a similar functionality to define and validate data quality constraints but with a specific focus on large-scale distributed data processing. For non-tabular data formats, solutions such as Deepchecks³³ and deequ offer native support for various dataset types, including images, text, and other structured or unstructured data. Deepchecks, for instance, provides a comprehensive suite of tools for validating and monitoring the quality of computer vision and natural language processing datasets. These specialized solutions can complement the capabilities of general-purpose data quality frameworks like Great Expectations, enabling data scientists to ensure the integrity and reliability of their training data across a wide range of domains and formats.

²⁶ <https://docs.python.org/3/library/unittest.html>

²⁷ <https://nose.readthedocs.io/en/latest/>

²⁸ <https://docs.python.org/3/library/doctest.html>

²⁹ <https://pynblint.readthedocs.io/en/latest/>

³⁰ <https://codecarbon.io>

³¹ <https://greatexpectations.io>

³² <https://github.com/aws-labs/deequ>

³³ <https://deepchecks.com>

Regarding model QA, the use of quantitative metrics for model evaluation, such as precision and recall, can be complemented by behavioral model testing (BMT). Behavioral tests assess the behavior of models when applied to specific categories of input data. We experimented with BMT in the natural language processing (NLP) domain, as inspired by (Ribeiro et al., 2020). However, the same type of testing can be applied straightforwardly to other domains, such as computer vision.

2.3. Development of APIs for ML models

To enable seamless integration into larger systems, machine learning (ML) models typically expose their predictive capabilities through web APIs. Among various alternatives, using a general-purpose framework such as FastAPI³⁴ allows the maximum degree of customizability. Furthermore, FastAPI itself represents an interesting solution due to its shallow learning curve, concise syntax, production-level robustness, and compliance with the OpenAPI standard. Nevertheless, there are other solutions designed specifically for developing APIs for ML models, such as BentoML³⁵, Cortex³⁶, and Seldon Core.³⁷ BentoML is an open-source framework that simplifies the process of packaging and deploying ML models as web services, while Cortex and Seldon Core are open source platforms that provide end-to-end solutions for deploying, monitoring, and managing ML models at scale. In addition, to showcase the integration of machine learning (ML) models into larger systems through their exposed APIs, special-purpose front-end frameworks like Gradio³⁸ and Streamlit³⁹ can be employed to build web-based prototypical user interfaces with minimal effort.

2.4. Delivery of ML components

Another crucial set of MLOps practices concerns the delivery of machine learning (ML)-based components. Beyond exposing endpoints, models need to be packaged in a portable manner and automatically deployed in cloud-based production environments.

³⁴ <https://fastapi.tiangolo.com>

³⁵ <https://www.bentoml.com>

³⁶ <https://docs.cortexlabs.com>

³⁷ <https://www.seldon.io/solutions/seldon-core>

³⁸ <https://www.gradio.app>

³⁹ <https://streamlit.io>

Therefore, in our review, we focused on achieving this through containerization and CI/CD technologies.

Containerization is a key enabler for packaging and deploying applications in a consistent and portable manner. The *de facto* standard for software containerization is Docker⁴⁰, an open-source platform that allows applications to be packaged with all their dependencies into lightweight, self-contained units called containers. Docker containers can run consistently across different environments, from development to production, ensuring reliable and reproducible deployments. For orchestrating Docker containers in production environments, Kubernetes⁴¹ has emerged as a popular and widely adopted solution that provides automated container deployment, scaling, and management capabilities.

To automate the end-to-end build and deployment process, CI/CD technologies play a crucial role. Due to its seamless integration with GitHub, one of the most popular code hosting platforms, GitHub Actions is a widely used CI/CD solution. However, there are several alternatives available, such as GitLab CI/CD⁴², Jenkins⁴³, Travis CI⁴⁴, and CircleCI⁴⁵. These tools enable developers to define and automate various stages of the software development lifecycle, including building, testing, and deploying applications to different environments. In the context of MLOps, they can also be used to automate the deployment of containerized ML-based components.

2.5. Monitoring of ML components

To ensure service availability and adequate model performance after deployment, it is crucial to continuously monitor ML-enabled components. A comprehensive monitoring system should track both the resource consumption of ML components and the performance of the ML models themselves, as they are typically subject to performance degradation over time. By setting up a robust monitoring system, ML engineers can maintain a feedback loop, enabling them to quickly identify and replace underperforming models as needed.

⁴⁰ <https://www.docker.com>

⁴¹ <https://kubernetes.io/it/>

⁴² <https://docs.gitlab.com/ee/ci/>

⁴³ <https://www.jenkins.io>

⁴⁴ <https://www.travis-ci.com>

⁴⁵ <https://circleci.com>

Popular open-source monitoring stacks – such as Prometheus⁴⁶ and Grafana⁴⁷ or Elasticsearch⁴⁸, Logstash⁴⁹, and Kibana⁵⁰ (also known as the ELK stack) – can be used to set up a full-fledged monitoring system for ML-based components or systems. While Prometheus or a combination of Elasticsearch and Logstash are generally used for tracking system metrics, they can also be customized to store model performance metrics. Grafana or Kibana, on the other hand, are used to visualize the data collected by the monitoring platforms, and present them in monitoring dashboards.

In addition to general-purpose stacks, there are also specialized alternatives specifically designed for monitoring ML models in production environments. For example, Evidently AI⁵¹, Deepchecks, and Whylogs⁵² offer specialized solutions for monitoring ML models in production. These tools are designed to track and analyze various aspects of model performance, such as data drift, model bias, and prediction quality, providing ML engineers with valuable insights and alerts to proactively address potential issues.

⁴⁶ <https://prometheus.io>

⁴⁷ <https://grafana.com>

⁴⁸ <https://www.elastic.co>

⁴⁹ <https://www.elastic.co/logstash>

⁵⁰ <https://www.elastic.co/kibana>

⁵¹ <https://www.evidentlyai.com>

⁵² <https://whylogs.readthedocs.io/en/latest/>

List of selected papers

- [R1] Bai, E. et al., 2021, “A Graphical Toolkit for Longitudinal Dataset Maintenance and Predictive Model Training in Health Care”, *Applied Clinical Informatics*, vol. 13, pp. 56–66, <https://doi.org/10.1055/s-0041-1740923>.
- [R2] Granlund T., et al., 2021, “Towards Regulatory-Compliant MLOps: Oravizio’s Journey from a Machine Learning Experiment to a Deployed Certified Medical Product”, *SN Computer Science*, vol. 2, <https://doi.org/10.1007/s42979-021-00726-1>.
- [R3] Subramanya, K. N. et al., 2022, “Application of MLOps in Prediction of Lifestyle Diseases”, *ECS Transactions*, vol. 107, pp. 1191–98, <https://doi.org/10.1149/10701.1191ecst>.
- [R4] Berezsky, O., et al., 2022, “Application of MLOps Practices for Biomedical Image Classification”, *CEUR Workshop Proc.*, vol. 3302, pp. 69–77, <https://ceur-ws.org/Vol-3302/short3.pdf>
- [R5] Mathew, N. and C. T. Joseph, 2023, “Applying Transfer Learning on 3D Brain Images and an MLOPS Study for Deployment”, *Int. Conf. Smart Comput. Commun.: Intell. Technol. Appl., ICSCC*, pp. 541–47, <https://doi.org/10.1109/ICSCC59169.2023.10335014>.
- [R6] Jain, A. et al., 2022, “Brain Tumor Detection Using MLOps and Hybrid Multi-Cloud”, *IEEE Int. Conf. Blockchain Distrib. Syst. Secur., ICBDS*, <https://doi.org/10.1109/ICBDS53701.2022.9936020>.
- [R7] Stirbu, V. et al., 2023, “Continuous Design Control for Machine Learning in Certified Medical Systems”, *Software Quality Journal*, vol. 31, pp. 307–33, <https://doi.org/10.1007/s11219-022-09601-5>.
- [R8] Karácsony, T. et al., 2021, “Deepepil: Towards an Epileptologist-Friendly AI Enabled Seizure Classification Cloud System Based on Deep Learning Analysis of 3D Videos”, *BHI - IEEE EMBS Int. Conf. Biomed. Health Informatics, Proc.*, <https://doi.org/10.1109/BHI50953.2021.9508555>.
- [R9] Lombardo, G. et al., 2024, “Digital Twin for Continual Learning in Location Based Services”, *Engineering Applications of Artificial Intelligence*, vol. 127, <https://doi.org/10.1016/j.engappai.2023.107203>.
- [R10] Koutsopoulos, K. et al., 2022, “Federated Machine Learning through Edge Ready Architectures with Privacy Preservation as a Service”, *Proc. - IEEE Future Networks World Forum, FNWF*, pp. 347–50, <https://doi.org/10.1109/FNWF55208.2022.00067>.
- [R11] Verbiest, J. R. et al., 2023, “Gait Stride Length Estimation Using Embedded Machine Learning”, *Sensors*, vol. 23, <https://doi.org/10.3390/s23167166>.
- [R12] Meel, V. and A. Bodepudi, 2022, “Melatect: A Machine Learning Approach for Identifying Malignant Melanoma in Skin Growths”, *Proc SPIE Int Soc Opt Eng*, vol. 12084, <https://doi.org/10.1117/12.2625005>.



[R13] Berezsky, O. et al., 2023, “MLOps Approach for Automatic Segmentation of Biomedical Images”, CEUR Workshop Proc., vol. 3609, pp. 241–48, <https://ceur-ws.org/Vol-3609/short5.pdf>

[R14] Fergus, P. et al., 2023, “Pressure Ulcer Categorization and Reporting in Domiciliary Settings Using Deep Learning and Mobile Devices: A Clinical Trial to Evaluate End-to-End Performance”, IEEE Access, vol. 11, pp. 65138–52, <https://doi.org/10.1109/ACCESS.2023.3289839>.

Bibliography

- Amershi, S., et al. (2019). Software Engineering for Machine Learning: A Case Study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 291–300). <https://doi.org/10.1109/ICSE-SEIP.2019.00042>
- Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley Professional.
- Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering—a systematic literature review. *Information and Software Technology*, 51(1), 7-15. <https://doi.org/10.1016/j.infsof.2008.09.009>
- Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*, 11, 31866–31879. <https://doi.org/10.1109/ACCESS.2023.3262138>
- Pimentel, J. F., Murta, L., Braganholo, V., & Freire, J. (2019). A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks. In Proc. of the 16th International Conference on Mining Software Repositories (pp. 507–517). <https://doi.org/10.1109/MSR.2019.00077>
- Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond Accuracy: Behavioral Testing of NLP models with CheckList. *Association for Computational Linguistics*. <https://doi.org/10.48550/ARXIV.2005.04118>
- Symeonidis, G., Nerantzis, E., Kazakis, A., & Papakostas, G. A. (2022). MLOps - Definitions, Tools and Challenges. In 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0453–0460). <https://doi.org/10.1109/CCWC54503.2022.9720902>
- Treveil, M., et al. (2020). *Introducing MLOps*. O'Reilly Media, Inc.
- Wang, J., Li, L., & Zeller, A. (2020). Better code, better sharing: On the need of analyzing jupyter notebooks. In Proc. of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results (pp. 53–56). <https://doi.org/10.1145/3377816.3381724>