An MLOps Solution Framework for Transitioning Machine Learning Models into eHealth Systems

Andrea Basile, Fabio Calefato, Filippo Lanubile*, Giulio Mallardi and Luigi Quaranta

Dept. of Computer Science, University of Bari, Via Edoardo Orabona 4, 70125 Bari BA, Italy

Abstract

Over the past few years, there has been a growing experimentation of machine learning (ML)-based technologies in the healthcare domain. However, most related initiatives struggle to progress beyond the prototypical research stage and transition to clinical use. Although this problem affects the adoption of ML across all industries, it is largely exacerbated in the highly regulated medical domain. Lately, MLOps has emerged as a new discipline encompassing practices and tools to streamline the development and maintenance of ML-enabled systems. Rooted in software engineering and inspired by DevOps, it places great emphasis on the automation of ML pipelines and model lifecycle. In this paper, we present an MLOps-based solution framework designed to streamline the transition of experimental ML models to production-ready components for eHealth systems. Our approach is designed to support the reliable integration and clinical deployment of ML-enabled tools that can assist healthcare professionals. The solution framework is being developed and validated in the context of "DARE – Digital Lifelong Prevention", an Italian research project aimed at leveraging the potential of data to improve health promotion and prevention throughout the life course.

Keywords

ML pipeline reproducibility, ML model deployment, ML-enabled component, ML for healthcare, health informatics

1. Introduction

The integration of data-driven artificial intelligence (AI) into eHealth systems has recently emerged as a promising avenue to enhance healthcare delivery and improve patient outcomes [1]. Consequently, in the last few years, there has been a growing experimentation of healthcare solutions based on machine learning (ML) and deep learning (DL). These data-driven AI techniques have already shown remarkable capabilities in key areas of medicine, from diagnostics to treatment [2].

However, most research initiatives struggle to progress beyond the prototypical research stage and transition to clinical use. On the one hand, the primary focus of research teams is typically on optimizing the model building process and advancing the state of the art in terms of model performance. On the other hand, moving beyond experimentation, towards the clinical application of machine learning, poses significant challenges. These include ensuring the end-to-end reproducibility and traceability of ML pipelines, verifying the quality of all involved artifacts, and making ML-enabled components interoperable. As a result — after the dissemination of scientific findings — ML models typically remain confined within laboratories and never find practical application. This precludes a broader societal impact of ML research in the medical domain, representing a significant dispersion of valuable resources and potential.

To address the key challenges hindering the practical application of ML in the healthcare domain, we propose a solution framework that integrates a set of best practices and a selection of software tools for their implementation. The solution framework is based on MLOps (short for 'Machine Learning Operations'), an emerging discipline in the area of AI engineering. Inspired by DevOps, MLOps places great emphasis on the automation of ML pipelines and the lifecycle of machine learning models.

Our solution framework is designed to support the end-to-end process for building and maintaining ML-enabled components to be integrated into eHealth systems. On the one hand, it aims to improve the practices adopted by data scientists in the laboratory. To this aim, it comprises tools to organize the requirements of an ML project, ensure the reproducibility and traceability of ML experiments, and verify the quality of code, data, and models. On the other hand, it assists the transition of ML models to production environments. To this aim, it supports activities such as model API development, model containerization, deployment, and monitoring. In all phases, it leverages workflow automation tools to make the process reproducible and reduce the margin for human error.

The solution framework described in this paper has been developed as part of "DARE – Digital Lifelong Pre-

Ital-IA 2024: 4th National Conference on Artificial Intelligence, organized by CINI, May 29-30, 2024, Naples, Italy

*Corresponding author.

andrea.basile@uniba.it (A. Basile); fabio.calefato@uniba.it (F. Calefato); filippo.lanubile@uniba.it (F. Lanubile); giulio.mallardi@uniba.it (G. Mallardi); luigi.quaranta@uniba.it (L. Quaranta)

0009-0007-2381-4575 (A. Basile); 0000-0003-2654-1588 (F. Calefato); 0000-0003-3373-7589 (F. Lanubile); 0000-0001-5847-117X (G. Mallardi); 0000-0002-9221-0739 (L. Quaranta)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3762/524.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

vention," an Italian research project aimed at leveraging the potential of data to improve health promotion and prevention throughout the life course. We are currently in the process of validating the benefits of the solution framework through a few case studies, both within and outside DARE. In the future, we plan to further extend the scope of our proposal, by leveraging automation to support further aspects of ML projects. For instance, we envision the automated creation of documentation and validation reports needed to comply with healthcare regulations and certify the resulting ML-enabled systems as medical devices.

The remainder of this paper is organized as follows. In Section 2, we provide a definition of MLOps and report about existing MLOps experimentation in the healthcare domain. In Section 3, we introduce the DARE research project. In Section 4, we provide details about the practices and tools included in our solution framework. In Section 5, we outline future research directions and in Section 6 we conclude the paper.

2. Background

2.1. MLOps definition

MLOps is an umbrella term that encompasses a set of practices and tools to streamline the creation and maintenance of ML-enabled systems. It primarily aims to automate ML pipelines and workflows, facilitating the deployment of models into production environments. The ultimate objective of MLOps is to implement the continuous integration and deployment of models (CI/CD), mirroring and extending the DevOps approach used in conventional software systems.

Kreuzberger et al. provide a comprehensive definition of MLOps, which they define as *"a paradigm, including aspects like best practices, sets of concepts, as well as a development culture when it comes to the end-to-end conceptualization, implementation, monitoring, deployment, and scalability of machine learning products"* [3].

With its growing popularity, MLOps is emerging as a distinct discipline in the area of AI engineering. This is evidenced by the recent addition of courses on this topic at some universities [4, 5].

2.2. MLOps in healthcare

The application of data-driven AI in healthcare faces several challenges, ranging from regulatory compliance and data privacy concerns to the interoperability of systems and the integration of AI-driven insights into clinical decision-making processes. Additionally, the high-stakes nature of healthcare demands rigorous validation, monitoring, and interpretability of machine learning models to ensure patient safety and trust among healthcare providers.

To address these challenges, an increasing number of researchers are exploring the use of MLOps to integrate ML models into eHealth systems. In [6], Granlund et al. introduce a certified medical software for the risk assessment of joint replacement interventions, exploring the use of MLOps in a highly regulated context. A similar effort is reported by Stirbu et al., who present an approach that leverages pull requests as design controls and applies it to integrate ML models in certified medical systems [7].

Lombardo et al. leverage a digital twin technology to provide Location Based Services (LBS) with intelligent functionalities [8]. In doing so, they leverage MLOps to facilitate model evolution and adaptation to changes in the physical world.

To address a similar problem, Toivakka et al. propose an efficient software delivery model, based on DevOps, which ensures compliance with medical device standards [9]. Specifically, they align medical device software regulatory requirements from standards IEC 62304 and IEC 82304-1 into the software delivery pipeline.

3. DARE Project

The DARE project is a wide-ranging initiative funded by the Italian Ministry of University and Research. It has fostered the development of a distributed knowledge community dedicated to digital preventive healthcare research. This community encompasses a network of around 250 researchers from universities, hospitals, healthcare companies, and other organizations.

The primary goal of the project is to produce the knowledge and multidisciplinary solutions necessary to establish Italy as a leading country in digital prevention. Specifically, the project aims to promote preventive actions enabled by digital technologies and big data to improve the readiness and accuracy of key public health tasks such as forecasting, surveillance, early diagnosis, and response to acute and chronic diseases, including comorbidities. A peculiarity of the project is the adoption of a 'life-course' perspective to address health-related conditions in general.

Ultimately, DARE aims to leverage digital technologies to bridge social and geographic disparities in access to integrated health services, benefiting the most vulnerable segments of the population.

4. MLOps Solution Framework

To support the transition of prototypical ML-based solutions developed within DARE to production-grade eHealth systems, we have proposed a solution framework based on state-of-the-art MLOps practices and tools. Our

framework has a general-purpose design and is meant to support the development and maintenance of a variety of ML-based eHealth software. However, it can be easily customized to support specific research initiatives within DARE and beyond.

In the following paragraphs, we describe the main ideas behind the solution framework. Specifically, we report on the MLOps practices encompassed by the framework, as well as the tools that we recommend for their practical implementation.

Several MLOps tools have been developed so far. Most of them are commercial solutions, typically integrated into end-to-end MLOps or cloud-computing platforms. Open-source options are available as well, and some of the commercial tools – typically provided as Software-as-a-Service (SaaS) – are based on an open-source core which can be independently deployed on-premises. In our solution framework, we recommend adopting open source software whenever possible. Not only is it typically more cost-effective, but it also offers independence from cloud infrastructures, enabling on-premises deployments. This is particularly important in the healthcare domain, in which hospitals and other research institutions need to comply with stringent patient data management requirements, which typically cannot leave the institution's computing facilities. In such cases, our MLOps solution framework can be fully deployed on-premises.

4.1. Scoping the ML Problem

When planning to build an ML-enabled system or component, the initial challenge is properly defining the underlying machine learning problem, if one exists. Indeed, while machine learning offers optimal solutions for a wide range of problems, it is always crucial to assess whether using it is sensible and feasible for the specific problem at hand, considering factors like availability of labeled data and computing resources.

Inspired by the Business Model Canvas, the Machine Learning Canvas by Goku Mohandas [10] can serve as a useful template to facilitate this decision-making process. It encourages thinking on both product and system design aspects, clarifying the motivation, key objectives, feasibility, and high-level strategy for building the proposed ML-enabled solution.

4.2. Ensuring the Reproducibility and Traceability of ML Pipelines

Once the basic requirements for the desired product have been specified, data engineers and data scientists can start working together to build the ML models that will power the final product. In doing so, they should take care of defining a reproducible and traceable pipeline.

Reproducibility is a key requirement for ML pipelines. It is essential not only for achieving consistent model performance across production and lab environments but also for enabling the recovery and timely retraining of deployed models. Nonetheless, the inherent nondeterministic nature of most ML and DL techniques, coupled with the complexity of ML pipelines, makes attaining reproducibility in practice a significant challenge.

Similarly, ensuring the full traceability of model building processes is of paramount importance. Healthcare is a safety-critical domain in which decisions can have life-altering consequences. Thus, for models aimed at supporting healthcare professionals in decision-making activities it is essential to be able to trace back any unexpected behavior to the model training process, enabling root cause analysis. This ensures the transparency and accountability of the overall system. Moreover, traceability helps in meeting healthcare regulations.

As a first step towards ensuring the reproducibility and traceability of ML pipelines, we propose the use of git as a version control system (VCS) for code artifacts and of DVC[1] as a specialized VCS for data and models. By adopting these tools in conjunction, it is always possible to understand which specific version of a dataset and of a training script were used to build a particular version of a machine learning model.

A further step towards ensuring the full traceability of the training process is adopting an experiment tracking solution. In this regard, we recommend using MLflow,[2] a popular open-source platform featuring a dedicated experiment tracking module (MLflow Tracking). With MLflow, data scientists can track all relevant details of an ML experiment, including the training algorithm, the hyperparameters, the dataset version, and the selected features. Similarly, the metrics selected for model evaluation can be logged into MLflow, together with any experimental output. The outcomes of experimental runs can then be visually compared in a dashboard offered through a web application. Once the best run has been determined, the resulting model can be registered in a model registry within the dedicated MLflow module (MLflow Registry). If used consistently to register models and update their status, the model registry becomes the centralized store of production-grade models and related metadata – i.e., if a deployed model is pulled from a model registry, it is easy to trace back the particular experimental run that produced it.

[1] https://dvc.org
[2] https://mlflow.org

4.3. Fostering Quality Assurance of ML Artifacts

A major criticism raised by software engineers towards data scientists concerns the poor code quality of experimental ML artifacts, particularly computational notebooks [11, 12]. Integrating data science tools with static analyzers and testing utilities could significantly improve code quality. In this regard, our framework promotes the adoption of pytest[3] as a testing framework and ruff[4] as a static analyzer for Python scripts. Moreover, in projects that include computational notebooks, we recommend the use of Pynblint,[5] i.e., a specialized linting solution for Jupyter Notebook documents.

Nonetheless, the quality of ML-enabled systems extends beyond code and is largely determined by the quality of data and models. It is widely acknowledged that model performance can be substantially impacted by the quality of training data, which often fails to meet ideal standards in real-world scenarios. Addressing data quality issues, such as biases, noise, and scarcity, is crucial to developing reliable and effective ML-enabled systems. To this aim, our solution framework provides for the use of Deepchecks,[6] a commercial tool with an open-source core. Deepchecks can be used to test training data for outliers and other anomalies; moreover, it reveals issues like the leakage of test data in training datasets.

In addition to assessing performance metrics, the quality of models can be further evaluated using dedicated testing approaches. Where applicable, our solution framework recommends the development of behavioral model tests. Originally proposed by Ribeiro et al. [13], these tests are designed to ensure specific model capabilities. For instance, in the case of an NLP model, data scientists might want to verify that the model can handle negations appropriately. These tests can be implemented using the same testing framework employed for verifying code correctness (in our framework, pytest).

4.4. Developing APIs for ML Components

To enable seamless integration of models into larger systems, they are typically encapsulated within dedicated APIs. Specifically, given the widespread adoption of microservices and serverless architectures, which predominantly rely on the HTTP protocol for inter-component communication, a common pattern is to expose ML models through web APIs, using either REST or RPC approaches. By wrapping models with standardized web APIs, they can be more easily consumed and orchestrated within distributed architectures, facilitating their deployment and scalability.

With respect to this, our solution framework endorses FastAPI,[7] a specialized Python framework for developing OpenAPI-compliant web APIs. By leveraging FastAPI, data scientists can efficiently build standardized and well-documented APIs for their ML models, benefiting from its high performance capabilities and first-class support for asynchronous code.

4.5. ML Component Delivery

In addition to exposing API endpoints, models must be packaged in a portable way and automatically deployed to production environments. To accomplish this, our MLOps solution framework embraces Infrastructure as Code (IaC), a well-established DevOps methodology. The typical approach involves packaging ML models, along with their web API components, into software containers leveraging IaC techniques. In our solution framework, we advocate for the use of Docker,[8] which has established itself as the de facto standard containerization technology during the last decade. Using Docker, ML models can be shipped as immutable and portable software packages that are consistently reproducible across different deployment environments. This containerized approach aligns with modern cloud-native architectures.

Deployed models need to be properly documented. As a standardized format to consistently document the delivered machine learning components, model cards can be adopted to report essential model attributes. Model cards are simple Markdown documents describing the model, its intended uses and potential limitations, including biases and ethical considerations, the training parameters and experimental information, the datasets used for training, and the model evaluation results. This type of documentation was first proposed by Mitchell et al. in [14] and gained popularity through its adoption by Hugging Face,[9] a prominent AI community and machine learning hub. Beyond model cards, Hugging Face users typically document the datasets used with Dataset Cards, which outline basic details about the data as well as information on how to use the data responsibly (e.g., potential biases within the dataset). Dataset cards help users understand the contents of the dataset and provide context for how it should be used.

A further step towards the end-to-end automation of ML pipelines is adopting CI/CD (Continuous Integration/-Continuous Deployment) solutions for the automated deployment of containerized ML components. Automating this step of the workflow offers two key benefits: expediting the deployment of model updates and minimizing

[3] https://docs.pytest.org
[4] https://docs.astral.sh/ruff/
[5] https://github.com/collab-uniba/pynblint
[6] https://deepchecks.com
[7] https://fastapi.tiangolo.com
[8] https://www.docker.com
[9] https://huggingface.co

human errors through consistent, rigorous quality assurance checks before deployment. Several CI/CD tools with similar capabilities are currently available. To reduce friction in adopting this practice, our framework recommends leveraging the CI/CD service integrated with the chosen code hosting platform for sharing Git repositories, such as GitHub Actions[10] for GitHub or GitLab CI/CD[11] for GitLab. A notable advantage of GitLab CI/CD is the ability to deploy the entire code hosting platform on-premises, which can be beneficial for institutions with policies prohibiting external code hosting.

4.6. ML Component Monitoring

To ensure the continued availability and performance of deployed ML-enabled components, continuous monitoring is essential. A comprehensive monitoring system should track both the resource utilization of ML components and the performance of the underlying ML models themselves, as model performance often degrades over time. Establishing robust monitoring practices maintains a crucial feedback loop, enabling ML engineers to promptly identify and replace underperforming models as needed.

A wide range of solutions can be leveraged for this purpose, ranging from general-purpose monitoring tools like Prometheus,[12] Grafana,[13] and the ELK stack[14] to specialized software specifically designed for monitoring ML systems, such as the monitoring module of Deepchecks. Within our solution framework, we favor the adoption of the popular open-source stack of Prometheus and Grafana. Their general-purpose nature allows for setting up custom monitoring services that holistically track the overall health of ML components, encompassing resource utilization metrics as well as ML model performance indicators. Moreover, being open-source, both Prometheus and Grafana can be seamlessly deployed on-premises using their official Docker containers, aligning with our framework's emphasis on open solutions and on-prem deployability for healthcare use cases.

5. Future Work

While the proposed MLOps framework lays a robust foundation for deploying machine learning models in healthcare, there are still additional issues that require careful consideration. In particular, the complexity of security and regulatory requirements in the healthcare sector poses significant challenges that need to be thoroughly addressed.

Accordingly, our future work will prioritize enhancing the security of the MLOps framework. Robust authentication, authorization, and encryption protocols will safeguard patient data and ensure system integrity through API security.

In addition, we aim to explore automated report generation as a means to facilitate compliance with regulatory bodies and enable efficient auditing processes. By leveraging CI/CD workflows to generate comprehensive reports and documentation, we expect to streamline compliance efforts, thereby facilitating the certification of ML models and ML-enabled eHealth systems as medical devices.

6. Conclusion

The approach presented in this paper represents a comprehensive and robust framework for the development, deployment, and monitoring of ML models within eHealth systems. By leveraging industry-standard practices and tools, it addresses the critical aspects of reproducibility, traceability, and quality assurance throughout the entire machine learning lifecycle.

The approach prioritizes a structured foundation for coding, emphasizing clear problem statements, data sources, and evaluation metrics. It integrates version control and experiment tracking for reproducibility and collaboration. Rigorous quality assurance is applied to data and models, ensuring integrity and ethical considerations. MLOps practices streamline deployment for efficient model deployment. Continuous monitoring detects issues early, fostering reliability and trust in developed models.

Ultimately, this comprehensive and methodical approach provides a solid foundation for healthcare organizations to harness the full potential of artificial intelligence while upholding responsible AI principles. It empowers stakeholders to develop and deploy machine learning models that are not only accurate and performant but also interpretable, ethical, and maintainable over time, driving innovation and positive impact across various domains and industries.

Acknowledgments

This study has been realized with the co-financing of the Ministry of University and Research in the framework of PNC "DARE - Digital lifelong prevention project" (PNC0000002 – CUP B53C22006450001). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.

[10] https://github.com/features/actions
[11] https://docs.gitlab.com/ee/ci/
[12] https://prometheus.io
[13] https://grafana.com
[14] https://www.elastic.co/elastic-stack

References

[1] E. J. Topol, High-performance medicine: the convergence of human and artificial intelligence, Nature Medicine 25 (2019) 44–56. doi:10.1038/s41591-018-0300-7.

[2] A. Bohr, K. Memarzadeh, Chapter 2 - the rise of artificial intelligence in healthcare applications, in: A. Bohr, K. Memarzadeh (Eds.), Artificial Intelligence in Healthcare, Academic Press, 2020, pp. 25–60. doi:10.1016/B978-0-12-818438-7.00002-2.

[3] D. Kreuzberger, N. Kühl, S. Hirschl, Machine learning operations (mlops): Overview, definition, and architecture, IEEE Access 11 (2023) 31866–31879. doi:10.1109/ACCESS.2023.3262138.

[4] F. Lanubile, S. Martínez-Fernández, L. Quaranta, Teaching MLOps in Higher Education through Project-Based Learning, in: 2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), IEEE, 2023, pp. 95–100. doi:10.1109/ICSE-SEET58685.2023.00015.

[5] F. Lanubile, S. Martínez-Fernández, L. Quaranta, Training future ML engineers: A project-based course on MLOps, IEEE Software (2023) 1–9. doi:10.1109/MS.2023.3310768.

[6] T. Granlund, V. Stirbu, T. Mikkonen, Towards Regulatory-Compliant MLOps: Oravizio's Journey from a Machine Learning Experiment to a Deployed Certified Medical Product, SN Computer Science 2 (2021) 342. doi:10.1007/s42979-021-00726-1.

[7] V. Stirbu, T. Granlund, T. Mikkonen, Continuous design control for machine learning in certified medical systems, Software Quality Journal 31 (2023) 307–333. doi:10.1007/s11219-022-09601-5.

[8] G. Lombardo, M. Picone, M. Mamei, M. Mordonini, A. Poggi, Digital Twin for Continual Learning in Location Based Services, Engineering Applications of Artificial Intelligence 127 (2024) 107203. doi:10.1016/j.engappai.2023.107203.

[9] H. Toivakka, T. Granlund, T. Poranen, Z. Zhang, Towards regops: A devops pipeline for medical device software, in: International Conference on Product-Focused Software Process Improvement, Springer, 2021, pp. 290–306. doi:10.1007/978-3-030-91452-3_20.

[10] G. Mohandas, Machine learning systems design, https://madewithml.com/, 2023.

[11] J. a. F. Pimentel, L. Murta, V. Braganholo, J. Freire, A large-scale study about quality and reproducibility of jupyter notebooks, in: Proceedings of the 16th International Conference on Mining Software Repositories, MSR '19, IEEE Press, 2019, p. 507–517. doi:10.1109/MSR.2019.00077.

[12] J. Wang, L. Li, A. Zeller, Better code, better sharing: on the need of analyzing jupyter notebooks, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 53–56. doi:10.1145/3377816.3381724.

[13] M. T. Ribeiro, T. Wu, C. Guestrin, S. Singh, Beyond accuracy: Behavioral testing of NLP models with CheckList, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4902–4912. doi:10.18653/v1/2020.acl-main.442.

[14] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, T. Gebru, Model Cards for Model Reporting, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, ACM, 2019, pp. 220–229. doi:10.1145/3287560.3287596.