ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 4 Issue 3 September 2024 / Page No: 200-208
Paper Id: JETA-V4I3P121 / Doi: 10.56472/25832646/JETA-V4I3P121

Original Article

A Survey of DevOps Practices for Machine Learning and Artificial Intelligence Workflows in Modern Software Development

Yeshwanth Macha[1], Sunij Kumar Pulichikkunnu[2]
[1,2]*Independent Researcher*

Received Date: 29 July 2024 Revised Date: 05 September 2024 Accepted Date: 28 September 2024

Abstract: *The rapid pace of the evolution of digital technologies has altered the software development industry, and Artificial Intelligence (AI) and Machine Learning (ML) have become the innovators of several industries. Compared to conventional software systems, AI/ML processes require complex data pipelines, continuous model training, validation, and deployment, necessitating the integration of development practices with strong operational practices. MLOps, as the application of DevOps principles to AI/ML systems, provides a formalized approach to automating, monitoring, and controlling these processes while ensuring scalability, reliability, and maintainability. This paper summarizes potential DevOps practices that can support AI/ML processes in contemporary software development and discusses how continuous integration and delivery, infrastructure as code, containerization, automated testing, and version control contribute to successful model lifecycle management. It also highlights the importance of cross-functional cooperation, monitoring, and governance to address challenges such as data and model drift, reproducibility issues, and operational inefficiencies. The study reports key enablers and gaps by integrating current practices and methods in organizations, offering guidance to researchers and practitioners to adopt effective, scalable, and responsible AI/ML systems. The findings reinforce the role of DevOps practices in balancing development and operations to build reliable, data-driven applications in dynamic environments.*

Keywords: *DevOps, MLOps, Continuous Integration, Automation, Software Development, Artificial Intelligence, DevOps Practices.*

I. INTRODUCTION

The rapid growth of the digital technologies has transformed the procedure of development, deployment, and maintenance of software in industries. Some of these technologies include machine learning (ML) and artificial intelligence (AI) which have turned out to be a source of innovation and can be applied in the form of predictive analytics and autonomous systems [1]. The AI and ML workflows are not straightforward since they involve high-status data pipes, training, validation, and deployment of models in a constant manner in contrast to conventional software. In order to have an effective management of such workflow, there is need to integrate software development practices with an effective operational process that results in the development of MLOps as a form of DevOps to AI/ML systems. MLOps tries to simplify the lifecycle of the ML models in a way that can guarantee scale, reliability, and maintainability.

MLOps frequently inflict infrastructure management, model reproducibility, and software engineer data scientist collaboration issues on organizations that deploy it [2][3]. Multi-project and multi-environment approaches are some of the solutions that can be used to manage multiple projects with microservice based solutions that can share the same codebases to reduce the overheads of running them and improve the allocation of resources. Moreover, the implementation of ML frameworks into industrial operations can result in the improved product quality, the stabilization of the processes, and the support of the environmentally friendly practices.

The integration of AI and ML workflows into the conventional software engineering practices has gained more significance. The use of structured engineering processes assists in the reduction of ad-hoc practices, bettering of models, and the management of data [4]. Another use of AI is software testing optimization and improving efficiency at both stages of the development lifecycle, which proves the possibility of DevOps principles to resolve the operational challenges of intelligent systems.

Regardless of the increased interest, there is still the necessity to conduct surveys that would summarize the existing practices, challenges, and organizational disparities in the application of DevOps to AI/ML workflow. The present study contends to fill this gap by examining the current practices and determining the central strategies, as well as providing a comparative overview. The results are to offer practical findings on the implementation of efficient, scalable and maintainable DevOps procedures within the current AI and ML software development.

This is an open access article under the CCBY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

A. Structure of the Paper

The paper organized in such a way that Section II introduce the background and history of DevOps and MLOps. The third section III is about AI/ML processes in software development. In section IV, the practices that facilitate AI/ML are described as DevOps practices. Section V a literature review of challenges and organizational strategies, whereas the conclusion of the research and future research directions presented in Section VI.

II. FOUNDATION OF DEVOPS AND MLOPS

This part comprises the changes of DevOps in software engineering, how DevOps has changed to become MLOps with important concepts, and a comparison between the classical DevOps and MLOps practices.

A. Evolution of DevOps in Software Engineering

The development of DevOps in software engineering shows how the field has changed into a disciplined area of strict, linear methods to a team-based, automated, and scaled philosophy. This timeline highlights the major phases and is illustrated in Figure 1 discussed below:

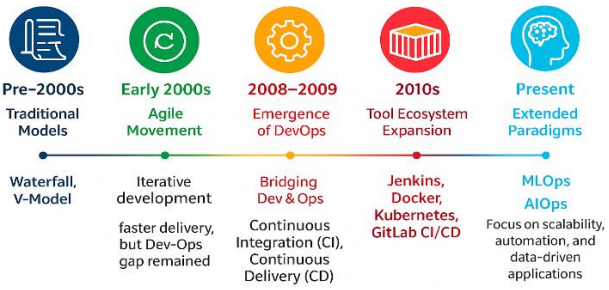

Figure 1 : Evolution of Devops in Software Engineering

a) Pre-2000s: Traditional Models

Software engineering was primarily characterized by structured methodologies, including the Waterfall and V-Model, during the pre-2000s era. Development, testing, and deployment were treated as separate and sequential stages, which resulted in lengthy release cycles and minimal feedback loops [5][6]. Although these models provided discipline and order, they were often inflexible and ill-suited for rapidly changing business requirements.

b) Early 2000s: Agile Movement

With the emergence of Agile methodologies in the early 2000s, the change has occurred. Agile promoted close customer contact, the use of iterative development, incremental delivery. This model helped the teams to provide updates more quickly and to react to the evolving requirements of the user. However, despite all the success, Agile introduced a division between the development teams and the IT operations and integration and deployment remained a problem.

c) 2008 – 2009: Emergence of DevOps

DevOps is a trend of the period between 2008 and 2009 that tried to fill the gap that existed between the team of operations and the developers team. CI, CD, IaC and automated testing were some of the practices that were embraced by DevOps. DevOps increased efficacy of the workflow, reduced mistakes and reduced time to market by teamwork and emphasis on automation.

d) 2010s: Tool Ecosystem Expansion

As of the 2010s, DevOps became a mainstream activity, and an ecosystem of tools and platforms has been growing quickly. Jenkins, Docker, Kubernetes, and GitLab CI/CD solutions provided the organizations with excellent automation, containers and orchestration functionality [7]. It is this decade in which DevOps became an experiment and a standard practice in the software engineering profession.

e) Present: Extended Paradigms

Nowadays, DevOps is not only a collection of practices but also a philosophy of constant enhancement, cooperation, and automation. The new paradigms of MLOps and artificial intelligence use IT operations with AIOps to apply the principles of DevOps to the machine learning processes. The trends mentioned above highlight the necessity to make bigger, data-driven, and intelligent systems, and DevOps becomes a platform to offer future software innovation.

B. Transition from DevOps to MLOps: Key Concepts

The history of software development practice has led to so-called MLOps as a variation of DevOps applied to the machine learning (ML) and artificial intelligence (AI) process in particular. In contrast to DevOps, when automated and

simplified software development life cycle is an issue, MLOps is worried about the specific issue of implementing data-driven model in production [8]. MLOps is not a change of DevOps to a technological one, but a change of adapting to the complexity degree in the form of data, model training, and continuous learning systems.

a) Key Concepts Driving this Transition Include:

The section provides an overview of the key concepts underlying the transition between DevOps and MLOps, such as the practices, tools, and teamwork strategies that become pivotal in the successful management of the AI and ML processes.

i) Data-Centric Pipelines

The idea of MLOps is that data is a first-class citizen in contrast to DevOps that emphasizes the provision of a code and applications. The requirements of the preprocessing, versioning, validation and data collection are required in the objective of precise model accuracy and reproducibility.

ii) Model Lifecycle Management

The subscription of the conventional CI/CD pipelines to CI/CD/CT (Continuous Training) is referred to as MLOps [9]. This guarantees that the models are not only deployed efficiently but also retrained and updated as new information is available and the performance is preserved over time.

iii) Experiment Tracking and Reproducibility

During machine learning procedures, multiple experiments are conducted by using different parameters, data sets, and algorithms. Experiment tracking, model versioning and reproducing results regardless of environment are also suggested in MLOps.

iv) Monitoring and Model Drift Detection

Whereas DevOps focuses on the monitoring of systems and uptime, MLOps uses model performance monitoring in production [10][11]. When the real-life data is not consistent with the training data, it is important to detect the model drift so as to avoid prediction accuracy degradation.

v) Cross-Functional Collaboration

MLOps facilitates collaboration within the domain of data scientists as well as between data scientists and operations as well as business stakeholders. The multidisciplinary approach to integration makes sure that the models are consistent with business objectives and are both technical and scalable.

vi) Tooling Ecosystem

The MLOps ecosystem is based on DevOps tooling (e.g. Docker, Kubernetes) and has ML-specific platforms, including MLflow, Kubeflow, TensorFlow Extended (TFX) and DVC (Data Version Control). These tools offer end-to-end pipeline management.

C. Comparison of Traditional DevOps vs. MLOps Practices

Table 1 : Comparison of Devops Vs MLOPS Practices

Aspect	Traditional DevOps	MLOps
Focus & Scope	Application code, deployment pipelines, infrastructure automation	Includes code, data, feature engineering, model training, and retraining pipelines
Continuous Integration/Delivery	CI/CD for building, testing, and deploying applications	CI/CD/CT (Continuous Training) to update and redeploy models as data evolves
Version Control	Git-based systems for source code management	Versioning of code, datasets, features, and models for reproducibility
Testing	Unit, integration, and system testing	Software tests + model validation, data quality checks, and performance benchmarking
Monitoring	System performance, uptime, and resource utilization	Tracks accuracy, fairness, bias, and model drift in real-world deployments
Collaboration	Developers, testers, and operations teams	Developers, data scientists, ML engineers, and domain experts
Tooling Ecosystem	Jenkins, Docker, Kubernetes, GitLab CI/CD	MLflow, Kubeflow, TFX, DVC alongside DevOps tools

DevOps and MLOps may have similar objectives, including automation, collaboration, and continuous improvement, but still are very different in scope, processes and challenges. DevOps is mainly focused on software development lifecycle whereas MLOps expands these concepts to include data, models and experiments [12]. It is essential to comprehend the

distinctions between these two paradigms when the organization is heading to an AI-driven system. Table I gives an overview of the comparison between DevOps and MLOps Practices:

III. AI/ML WORKFLOWS IN SOFTWARE DEVELOPMENT

The software development systems of AI and Machine Learning (ML) processes are more complicated than the traditional software lifecycle because they include not only software, but also data, models, and adaptation in changing situations. With the appropriate architectural workflow, data driven systems (see in Figure 2) have a high degree of reliability, scalability and can be maintained over time [13][14]. The core components of AI/ML workflows include data engineering, model development, and lifecycle management.

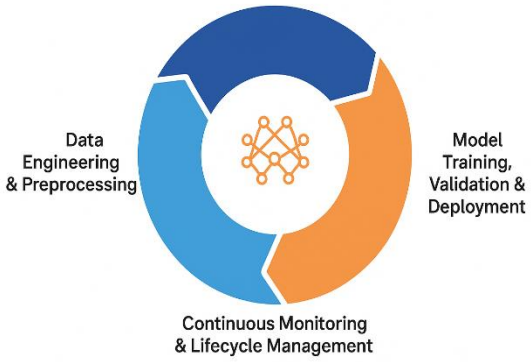

Figure 2 : AI/ML Workflows in Software Development

A. Data Engineering and Preprocessing

Data serves as the foundation of any AI/ML system. Unlike traditional software, where code is the primary asset, ML models are heavily dependent on the quality and consistency of input data [15]. Data engineering ensures that raw data from diverse sources (databases, IoT devices, logs, etc.) is collected, cleaned, and transformed into structured formats suitable for training. Key aspects include:

- Data Collection & Integration: Aggregating structured and unstructured data from multiple heterogeneous sources.
- Data Cleaning & Transformation: Handling missing values, noise, inconsistencies, and duplicates to maintain accuracy.
- Feature Engineering: Creating meaningful features from raw attributes to improve model performance.
- Scalability: Using data pipelines in the cloud and big data frameworks like Apache Spark and Hadoop to process massive amounts of data.

B. Model Training, Validation, and Deployment

Once high-quality data is available, the next stage focuses on building ML models. This involves experimentation, rigorous testing, and deployment strategies to integrate models into production environments. Key elements include:

- Model Training: Applying algorithms (e.g., Deep Learning, Decision Trees, Ensemble Models) on training datasets using frameworks such as TensorFlow, PyTorch, or Scikit-learn.
- Hyperparameter Tuning: Optimizing learning rates, batch sizes, and architectures to maximize model accuracy.
- Validation and Testing: Ensuring generalizability and avoiding overfitting by dividing data into training, validation, and testing sets [16][17]. Many evaluation metrics and cross-validation procedures are utilized, including precision, recall, F1-score, and ROC-AUC.
- Deployment Strategies: Deploying models into scalable production systems using CI/CD pipelines, containerization (Docker), and orchestration tools (Kubernetes).

C. Continuous Monitoring and Lifecycle Management

The degradation of ML models over time might be caused by concept drift, data drift, or shifting user behaviour, unlike traditional applications [18][19]. Hence, continuous monitoring and lifecycle management are crucial to maintain model effectiveness. Core practices include:

- Performance Monitoring: Tracking model accuracy, latency, fairness, and bias during real-world usage.
- Model Retraining: Updating models with fresh data to adapt to changes in patterns.
- Drift Detection: Identifying when input data distribution or target variables shift significantly.
- Automation with MLOps: Integrating automated pipelines for retraining, testing, and redeployment (CI/CD/CT) to ensure agility.
- Governance & Compliance: Maintaining audit trails, version control, and regulatory compliance for responsible AI use.

IV. DEVOPS PRACTICES ENABLING AI/ML

The integration of DevOps practices into AI and ML workflows has given rise to MLOps, a discipline that adapts the principles of continuous integration, delivery, and automation to data-driven systems illustrate in Figure 3. Traditional DevOps focused primarily on code, whereas AI/ML introduces additional complexity through data pipelines, model training, validation, and monitoring [20][21]. Consequently, DevOps practices have emerged as key enables in the effort to see AI/ML applications scale to be reliable and agile.

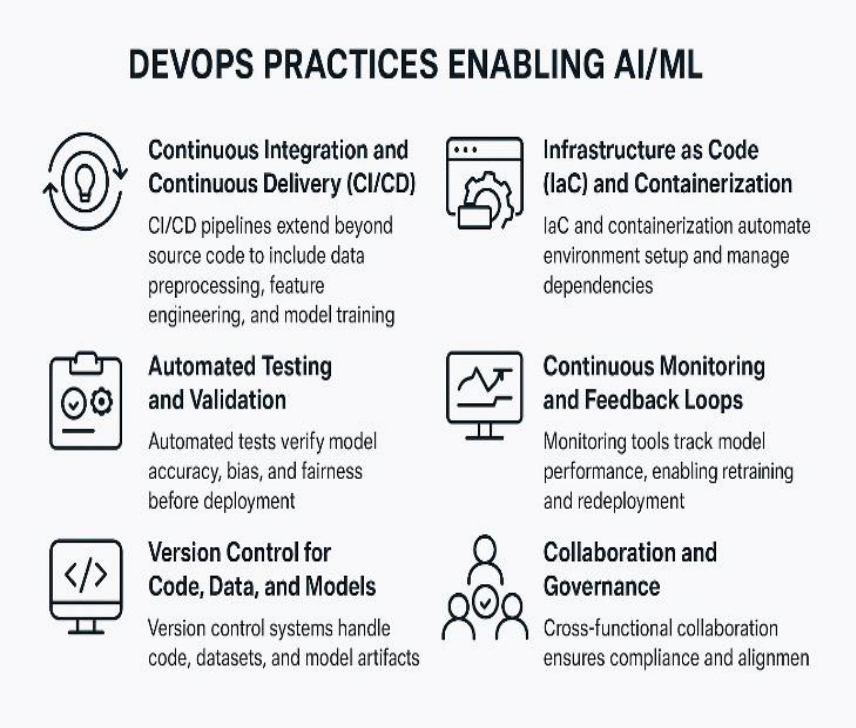

Figure 3 : Devops Practices Enabling AL/ML

A. Continuous Integration and Continuous Delivery (CI/CD)

The source code is not limited to CI/CD pipelines, which also contain data preprocessing scripts in AI/ML projects, feature engineering, and trained models [22]. The automated pipelines offer the assurance that the construction, testing, or introducing of datasets and models allude to every alteration in their parameters. This reduces human interference and accelerates the experimentation process and improves the reproducibility of the results.

B. Infrastructure as Code (IaC) and Containerization

AI/ML tasks are CPU-intensive and need the consistency in the execution environment in the development and testing phases, as well as in production. IaC and containerization (e.g. Docker, Kubernetes) are some of the DevOps practices that facilitate automatization of setting up an environment, scalability of GPU clusters, and complex dependencies. This ensures that ML models can be trained and deployed consistently across heterogeneous platforms.

C. Automated Testing and Validation

Testing in AI/ML differs from traditional software because it must validate not only functionality but also model accuracy, bias, and fairness. DevOps pipelines enable automated unit tests for data preprocessing, integration tests for APIs, and validation metrics for models before deployment [23]. These automated practices reduce the risk of deploying underperforming or biased models into production

D. Continuous Monitoring and Feedback Loops

Data drift and concept drift cause ML models to degrade over time, unlike static software. DevOps practices extend into observability by integrating monitoring tools that track accuracy, latency, fairness, and business KPIs in real time. Automated feedback loops allow teams to detect performance degradation, retrain models, and redeploy updated versions seamlessly.

E. Version Control for Code, Data, and Models

A critical enabler in AI/ML workflows is the extension of version control systems (e.g., Git, DVC, MLflow) to handle not just source code but also datasets, model artifacts, and experiment configurations [23][24]. This enables traceability, reproducibility, and rollback capabilities, ensuring that teams can audit and reproduce AI/ML outcomes with precision.

F. Collaboration and Governance

DevOps emphasizes cultural alignment between development and operations. In AI/ML, this principle extends to data scientists, ML engineers, and business stakeholders. Practices such as governance frameworks, audit trails, and model registries ensure compliance with regulatory standards while fostering cross-functional collaboration.

V. LITERATURE REVIEW

Previous studies have mainly focused on examining challenges in adopting DevOps practices. In contrast, this survey of DevOps practices for machine learning and artificial intelligence workflows investigates organizational differences and provides a comparative summary in Table II:

Erdenebat et al. (2023) suggest a new method to overcome this limitation by outlining a multi-project, multi-environment (MPME) strategy. Businesses and developers can put more energy into creating the product itself and less into managing the DevOps infrastructure when they use this method to organize multiple microservice-based projects running on a shared code base in a large organization using self-hosted Kubernetes clusters [25].

Hanzelik, Kummer and Abonyi (2022) This study details a methodology for automating the creation, upkeep, and management of software sensors for modelling complicated chemical processes, with a focus on machine learning (ML) methods. Utilizing ML and edge computing, wants to provide the chemical industry with cutting-edge answers to the problem of measuring variables in the lab that are notoriously difficult. Software sensor models are designed to continuously predict product quality in order to provide effective quality control, and to support efficient, eco-friendly, and safe laboratory work while also maintaining consistent plant production conditions [26].

Jha and Popli (2021) organization has been forced to reevaluate its software development procedures due to the purpose. Validating the quality of software is an essential function of software testing. Innovations to solve these integration difficulties must be led by researchers and practitioners in both artificial intelligence and software testing. Software testing might be taken to the next level with the help of artificial intelligence and machine learning (ML). Providing a concise overview of current research on AI applications in software testing is the primary goal of this article. Also covered are the software testing tasks associated with AI and the difficulties that come with them [27].

Shafiq et al. (2021) provide research on the application of ML to different phases of the SDLC. Examining the connection between the many phases of the software development life cycle and various types, approaches, and tools of machine learning is the overarching goal of this article. To find out if machine learning prefers specific phases or methods, undertake an all-encompassing analysis [28].

Lorenzoni et al. (2021) One problem with current machine learning development processes is that many in the field are unaware that the Software Engineering Development Lifecycle includes steps that might greatly enhance their work. Naturally, given the distinct nature of machine learning systems compared to more conventional software systems, it is reasonable to anticipate that their development processes differ. From a software engineering vantage point, this paper seeks to examine the difficulties and solutions that crop up throughout ML model development. Specifically, it seeks to deduce how software engineers might reap the benefits of incorporating the conventional software engineering process into a Machine Learning workflow [29].

Nascimento et al. (2020) examines the use of software engineering (SE) in the creation of artificial intelligence (AI) and machine learning (ML) systems, finds relevant problems and solutions, and finds out if these match the demands of experts. These systems were developed in a lab or by a large firm using a research-driven development approach, according to their results. Testing, data management, and artificial intelligence software quality are the primary areas where experts encounter difficulties. Guidelines, lessons learnt, and tools are the most common sorts of contributions made by suggested SE practices [30].

Barenkamp, Rebstadt and Thomas (2020) research organizes the findings throughout the SDLC. The analysis indicates that the great success and possibilities of AI lie in a) the possibility to create long routine jobs in the software development and testing with the help of algorithms. AI therefore helps accelerate the development processes, achieve a reduction in development costs and efficiency. The AI that is currently available is also based on artificial design and is mostly reproductive, however, the software engineering process is going to be automated, and the major advantage of it lies in the fact that Human developers able to increase their creativity when it comes to using AI devices [31].

The literature review discusses the literature critically in relation to their objectives, their key results, limitations, as well as prospects. An overview of these aspects is therefore put in Table II in a clear and comparative manner. The research trends of applying DevOps concepts in AI and ML pipelines to the modern software development process are put into focus in this methodology.

Table 2 : Comparative Analysis of Devops Practices In Modern Software Development

References	Study On	Objectives	Findings	Challenges	Future Directions
Erdenebat et al. (2023)	Multi-project, multi-environment (MPME) approach for microservice-based projects	Helping big businesses use self-hosted Kubernetes clusters to manage numerous microservice projects sharing the same codebase	Helps developers focus on product development and reduces DevOps infrastructure management effort	Organizing and coordinating environments and numerous tasks.	Additional efficiency to multi-project orchestration and automation of DevOps.
Hanzelik, Kummer & Abonyi (2022)	ML framework for software sensors in chemical processes	Develop, maintain, and manage the lifecycle of ML-based software sensors	Constant prediction of product quality, good quality control, good stable operation of plants, green work in laboratories.	The use of correct ML models of variables that are hard to measure.	Integration of edge computing with ML for real-time process optimization
Jha & Popli, (2021)	AI in software testing	Analyse the current level of artificial intelligence (AI) used for software testing	AI and ML are able to improve software testing power and productivity.	The difficulties of AI in software test workflows integration.	Innovations in AI-driven testing strategies and automated validation
Shafiq et al. (2021)	ML in software development life cycle (SDLC)	Look at the connection between the various SDLC steps and the various ML methods, tools, and types	ML prefers some SDLC steps and methods and makes the processes more efficient.	Identifying the best ML to use at each stage of SDLC.	Holistic integration of ML across all SDLC stages
Lorenzoni et al. (2021)	Machine learning (ML) development with an emphasis on software engineering	Investigate challenges and practices in ML model development using software engineering processes	A lot of professionals engage in impromptu procedures that could be improved with the use of SE techniques.	Modifying conventional SE practice to ML practices.	Development of standardized ML engineering processes based on SE principles
Nascimento et al. (2020)	SE practices in AI/ML system development	Identify challenges and practices in AI/ML system development	The majority of systems are research based; guidelines, lessons learnt and tools are all good donations.	Testing, data management, quality of AI software.	Enhance SE practices tailored for AI/ML systems in industrial contexts

VI. CONCLUSION AND FUTURE WORK

The continuous evolution of digital technologies has fundamentally reshaped software development, placing AI and ML at the forefront of innovation across industries. As AI/ML systems become increasingly complex, the integration of DevOps practices through MLOps has proven essential for ensuring efficient, reliable, and scalable workflows. Data and model drift, reproducibility, operational inefficiencies, and continuous integration and delivery are some of the specific difficulties that AI/ML pipelines present. This study has shown how important practices like infrastructure as code, containerization, automated testing, version control, and cross-functional collaboration help organizations manage these issues. Practical insights into creating organized, repeatable, and robust AI/ML workflows are offered by the study's comparative analysis of organizational approaches. Improved automation of model retraining and deployment, standardised MLOps frameworks that are suitable for different types of organizations, and operational pipelines that incorporate explainable AI and ethical governance should be the goals of future research. Better optimization of AI/ML system performance, reduction of operational risks, and support for intelligent, responsible, and sustainable software development in ever-changing environments can be achieved by studying how new technologies like AI-driven monitoring, federated learning, and edge computing affect DevOps practices.

VII. REFERENCE

[1] M. Steidl, M. Felderer, and R. Ramler, "The pipeline for the continuous development of artificial intelligence models—Current state of research and practice," J. Syst. Softw., vol. 199, 2023, doi: 10.1016/j.jss.2023.111615.

[2] D. E. Rzig, F. Hassan, and M. Kessentini, "An empirical study on ML DevOps adoption trends, efforts, and benefits analysis," Inf. Softw. Technol., 2022, doi: 10.1016/j.infsof.2022.107037.

[3] G. Modalavalasa, "The Role of DevOps in Streamlining Software Delivery: Key Practices for Seamless CI/CD," Int. J. Adv. Res. Sci. Commun. Technol., vol. 1, no. 12, pp. 258–267, Jan. 2021, doi: 10.48175/IJARSCT-8978C.

[4] C. Watson, N. Cooper, D. N. Palacio, K. Moran, and D. Poshyvanyk, "A Systematic Literature Review on the Use of Deep Learning in Software Engineering Research," ACM Trans. Softw. Eng. Methodol., vol. 31, no. 2, pp. 1–58, Apr. 2022, doi: 10.1145/3485275.

[5] F. H. Alshammari, "Trends in Intelligent and AI-Based Software Engineering Processes: A Deep Learning-Based Software Process Model Recommendation Method," Comput. Intell. Neurosci., vol. 2022, pp. 1–11, Oct. 2022, doi: 10.1155/2022/1960684.

[6] A. Goyal, "Driving Continuous Improvement in Engineering Projects with AI-Enhanced Agile Testing and Machine Learning," Int. J. Adv. Res. Sci. Commun. Technol., vol. 3, no. 3, pp. 1320–1331, Jul. 2023, doi: 10.48175/IJARSCT-14000T.

[7] Z. Kotti, R. Galanopoulou, and D. Spinellis, "Machine Learning for Software Engineering: A Tertiary Study," ACM Comput. Surv., vol. 55, no. 12, 2023, doi: 10.1145/3572905.

[8] A. R. Munappy, J. Bosch, H. H. Olsson, A. Arpteg, and B. Brinne, "Data management for production quality deep learning models: Challenges and solutions," J. Syst. Softw., vol. 191, p. 111359, Sep. 2022, doi: 10.1016/j.jss.2022.111359.

[9] N. Lokiny, "The Role of AI and Machine Learning in DevOps Automation," vol. 7, no. 2, pp. 328–333, 2020.

[10] R. Miñón, J. Diaz-De-arcaya, A. I. Torre-Bastida, and P. Hartlieb, "Pangea: An MLOps Tool for Automatically Generating Infrastructure and Deploying Analytic Pipelines in Edge, Fog and Cloud Layers," Sensors, 2022, doi: 10.3390/s22124425.

[11] A. R. Bilipelli, "End-to-End Predictive Analytics Pipeline of Sales Forecasting in Python for Business Decision Support Systems," Int. J. Curr. Eng. Technol., vol. 12, no. 6, pp. 819–827, 2022.

[12] G. Sriraman and S. R., "A machine learning approach to predict DevOps readiness and adaptation in a heterogeneous IT environment," Front. Comput. Sci., vol. 5, Oct. 2023, doi: 10.3389/fcomp.2023.1214722.

[13] S. Tatineni, "Applying DevOps Practices for Quality and Reliability Improvement in Cloud-Based Systems." 2023. doi: 10.13140/RG.2.2.25688.88326.

[14] A. Goyal, "Optimising Software Lifecycle Management through Predictive Maintenance : Insights and Best Practices," Int. J. Sci. Res. Arch., vol. 07, no. 02, pp. 693–702, 2022.

[15] F. M. A. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," J. Softw. Evol. Process, vol. 29, no. 6, Jun. 2017, doi: 10.1002/smr.1885.

[16] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023, doi: 10.1109/ACCESS.2023.3262138.

[17] V. Shah, "Next-Gen Emergency Communication Using LowPower Wide-Area and Software-Defined WANS," Int. J. Adv. Res. Sci. Commun. Technol., vol. 2, no. 1, pp. 600–609, 2022, doi: 10.48175/IJARSCT-8349M.

[18] S. Amershi et al., "Software Engineering for Machine Learning Applications," Icse, vol. 2020, pp. 1–10, 2019.

[19] Abhishek and P. Khare, "Cloud Security Challenges: Implementing Best Practices for Secure SaaS Application Development," Int. J. Curr. Eng. Technol., vol. 11, no. 06, pp. 669–676, Nov. 2021, doi: 10.14741/ijcet/v.11.6.11.

[20] L. Faubel, K. Schmid, and H. Eichelberger, "MLOps Challenges in Industry 4.0," SN Comput. Sci., 2023, doi: 10.1007/s42979-023-02282-2.

[21] G. Modalavalasa, "Towards Sustainable Development Based on Machine Learning Models for Accurate and Efficient Flood Prediction," vol. 8, no. 2, pp. 940–944, 2021.

[22] R. Amaro, R. Pereira, and M. M. Da Silva, "Capabilities and Practices in DevOps: A Multivocal Literature Review," IEEE Trans. Softw. Eng., 2023, doi: 10.1109/TSE.2022.3166626.

[23] E. Hechler, M. Oberhofer, and T. Schaeck, Deploying AI in the Enterprise. Berkeley, CA: Apress, 2020. doi: 10.1007/978-1-4842-6206-1.

[24] H. P. Kapadia, "AI Enhanced Web Accessibility Features," Int. J. Res. Anal. Rev., vol. 8, no. 4, pp. 476–483, 2021.

[25] B. Erdenebat, B. Bud, T. Batsuren, and T. Kozsik, "Multi-Project Multi-Environment Approach—An Enhancement to Existing DevOps and Continuous Integration and Continuous Deployment Tools," Computers, vol. 12, no. 12, p. 254, Dec. 2023, doi: 10.3390/computers12120254.

[26] P. P. Hanzelik, A. Kummer, and J. Abonyi, "Edge-Computing and Machine-Learning-Based Framework for Software Sensor Development," Sensors, vol. 22, no. 11, p. 4268, Jun. 2022, doi: 10.3390/s22114268.

[27] N. Jha and R. Popli, "Artificial intelligence for software testing-perspectives and practices," in Proceedings - 2021 4th International Conference on Computational Intelligence and Communication Technologies, CCICT 2021, 2021. doi: 10.1109/CCICT53244.2021.00075.

[28] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," IEEE Access, vol. 9, pp. 140896–140920, 2021, doi: 10.1109/ACCESS.2021.3119746.

[29] G. Lorenzoni, P. Alencar, N. Nascimento, and D. Cowan, "Machine Learning Model Development from a Software Engineering Perspective: A Systematic Literature Review," arvix, 2021.

[30] E. Nascimento, A. Nguyen-Duc, I. Sundbø, and T. Conte, "Software engineering for artificial intelligence and machine learning software: A systematic literature review," arvix.org, 2020, doi: /10.48550/arXiv.2011.03751.

[31] M. Barenkamp, J. Rebstadt, and O. Thomas, “Applications of AI in Classical Software Engineering,” AI Perspect., vol. 2, no. 1, Dec. 2020, doi: 10.1186/s42467-020-00005-4.