International Journal of Emerging Research in Science, Engineering, and Management
Vol. 2, Issue 1, pp.117-124, January 2026.
www.ijersem.com eISSN- 3107-9075

MLOps-Based Cloud Deployment Pipeline for Scalable ML Workloads

[1]T.M.S. Mekalarani, [2]Vajja Sai Pavan, [3]Sara Radhika,
[4]Akkarapalli Rekha, [5]Y Thimmaraju

Department of CSE, Siddharth Institute of Engineering and Technology, Puttur, India

[1]mekalaanandan@gmail.com, [2]vajjasaipavan3239@gmail.com, [3]sararadhika259@gmail.com,
[4]rekhaakkaripalli@gmail.com, [5]timmaraju2703@gmail.com

Abstract: Machine learning models are increasingly used across industries, but their effectiveness depends not only on accuracy during development but also on efficient deployment for real-world applications. Traditional through distributed resources and managed services. The work "Cloud-Based Machine Learning Model Deployment" focuses on building a framework to deploy, manage, and scale machine learning models seamlessly using cloud platforms such as AWS, Azure, or Google Cloud. It ensures high availability, low latency, and easy integration with client applications through REST APIs and microservices. Existing systems often deploy ML models manually on local servers, which leads to difficulties in version control, limited scalability, higher maintenance costs, and inefficiency in handling large-scale data requests. The proposed system leverages containerization tools like Docker and orchestration platforms like Kubernetes to automate deployment, enable auto-scaling, and ensure secure model serving across multiple environments. Algorithms such as Random Forest, Support Vector Machines, and Neural Networks can be trained offline and deployed in the cloud, while additional techniques like CI/CD pipelines and model monitoring ensure real-time performance optimization and reliability. This approach enhances scalability, efficiency, and accessibility for enterprises adopting ML-driven solutions.

Keywords: MLOPS, Cloud Deployment, Distributed Resources, Google Cloud, Random Forest.

1 INTRODUCTION

The rapid proliferation of machine learning (ML) across diverse sectors such as finance, healthcare, and retail has fundamentally shifted the focus of artificial intelligence from experimental research to operational necessity. While the development of high-accuracy models using algorithms like Random Forest, Support Vector Machines, and deep Neural Networks has become more accessible, the true challenge lies in transitioning these models from a localized, static environment to a dynamic, production-grade cloud ecosystem. In the contemporary digital landscape, a machine learning model is only as valuable as its accessibility and reliability in real-world applications. Traditional deployment methods, which often rely on manual configurations and local server hosting, are increasingly becoming bottlenecks for enterprises aiming to innovate at scale. These legacy systems suffer from significant technical debt, including difficult version control, high maintenance overheads, and an inherent inability to handle fluctuating data requests, which often results in increased latency and system downtime.

To bridge this gap between model development and operational excellence, the research MLOps-Based Cloud Deployment Pipeline for Scalable ML Workloads proposes a robust framework that integrates the principles of Machine Learning Operations (MLOps) with cloud-native technologies. The transition from a "code-first" mindset to a "system-first" approach is the defining characteristic of this work. Historically, data scientists have focused on achieving the highest possible $F1$-score or lowest Root Mean Square Error ($RMSE$) on static datasets. However, when these models encounter the "real world," they face challenges such as data drift—where the statistical properties of input data change over time—and concept drift, where the relationship between input and output variables evolves. Without a sophisticated MLOps pipeline, these models quickly become obsolete or inaccurate. The proposed system addresses this by establishing a standardized environment where the model is treated as a living entity that requires constant nurturing, monitoring, and versioning.

This shift toward MLOps-driven cloud deployment is necessitated by the need for high availability and seamless integration. By leveraging managed services from major cloud providers such as AWS, Azure, or Google Cloud, this framework ensures that machine learning models are no longer siloed but are instead served through standardized REST APIs and microservices. Central to this proposed system is the use of containerization tools like Docker. In a traditional setup, a model might work perfectly on a developer's laptop but fail in production due to library version mismatches or operating system differences. Docker encapsulates the model, its environment, and all its dependencies into a portable unit, ensuring absolute consistency across development, testing, and production environments. This "build once, run anywhere" philosophy is the bedrock of reliable deployment. When paired with orchestration platforms like Kubernetes, the deployment pipeline gains the ability to automate the scaling of resources.

Kubernetes acts as the brain of the operation, managing clusters of containers and ensuring that the health of the application is maintained. This means that during periods of high demand—such as a retail model during a Black Friday sale—the system can automatically provision additional computational power to maintain low latency. Subsequently, it can scale down to optimize costs during idle periods, a feature known as auto-scaling that is nearly impossible to replicate with manual local servers. The implementation of Continuous Integration and Continuous Deployment (CI/CD) pipelines within this framework ensures that models can be updated and redeployed without interrupting service. In an industry-standard pipeline, any change to the model code or hyperparameters triggers an automated suite of tests. If the model passes these quality gates, it is automatically built into a new container and deployed to a staging environment for further validation before reaching the end-user. This automation drastically reduces the "Time-to-Market" and minimizes the risk of human error during deployment. This is complemented by rigorous model monitoring and real-time performance optimization techniques. In a production environment, it is not enough to deploy a model once; it must be continuously evaluated.

The proposed system includes monitoring layers that track the health of the model, guarding against issues like latency spikes or accuracy degradation. By automating these monitoring and retraining loops, the research enhances the overall reliability and security of the model serving process. Another critical aspect of this cloud-based approach is the democratization of machine learning results. By utilizing microservices, the intelligence gathered by a model can be consumed by multiple disparate client applications—ranging from mobile apps to internal dashboard tools—simultaneously. This accessibility is managed through secure API Gateways that handle authentication, rate limiting, and request logging. This ensures that while the model is scalable and accessible, it remains protected from unauthorized access or malicious overload. The framework also integrates centralized logging and telemetry, allowing engineers to visualize the flow of data and troubleshoot bottlenecks in real-time. By moving away from the "manual hand-off" between data scientists and IT operations, this work fosters a culture of collaboration where model performance is a shared responsibility.

Ultimately, this MLOps-based pipeline transforms machine learning from a series of isolated experiments into a reliable, high-performing utility that can drive informed decision-making and operational intelligence across the modern enterprise. It solves the "last mile" problem of data science—ensuring that the intelligence generated in the lab actually reaches the consumer in a way that is fast, secure, and cost-effective. As enterprises continue to adopt ML-driven solutions at scale, the ability to manage these workloads with precision and automation will become the primary differentiator between successful AI adopters and those burdened by technical complexity. This research provides the architectural blueprint for that future, delivering a scalable and efficient environment for the next generation of intelligent applications.

2 LITERATURE SURVEY

The evolution of machine learning from a purely academic and experimental discipline to a foundational pillar of modern industrial operations has necessitated a significant shift in how researchers and engineers view the lifecycle of a model. Traditionally, the academic literature surrounding machine learning focused almost exclusively on the "modeling" phase—developing novel architectures, improving loss functions, and optimizing hyperparameters to squeeze every decimal point of accuracy out of a static dataset. However, as these models moved into the production phase, a new set of challenges emerged that traditional software engineering was not fully equipped to handle. This led to the birth of MLOps (Machine Learning Operations), a discipline that combines Machine Learning, DevOps, and Data Engineering to provide a standardized process for the deployment and maintenance of models in production. The historical trajectory of this field begins with the realization that machine learning systems are uniquely prone to "technical debt." A seminal paper by Sculley et al. from Google, titled "Hidden Technical Debt in Machine Learning Systems," identified that only a small fraction of a real-world ML system is composed of the actual ML code.

The vast surrounding infrastructure—including data collection, feature extraction, configuration, resource management, and metadata management—composes the majority of the system's complexity. This literature highlighted the "fragility" of ML systems, noting that they are prone to failures not just because of bugs in the code, but because of changes in the external world that cause data to shift away from the training distribution. Following this realization, the academic community and industry leaders began exploring the transition from manual, "artisanal" model deployment to automated, scalable pipelines. Early literature in this domain focused on the concept of "Continuous Integration and Continuous Delivery" (CI/CD) as applied to machine learning. Unlike traditional software, where CI/CD only needs to track code changes, ML systems require a tripartite versioning system involving code, data, and the resulting model weights.

This complexity led to the development of specialized tools and frameworks designed to automate the hand-off between data scientists and operations engineers. Researchers began to argue that the "manual" level of MLOps (Level 0), characterized by a lack of automation and frequent model decay, was no longer sustainable for enterprises. This paved the way for the adoption of containerization, particularly through Docker. The literature on containerization in ML emphasizes its role in solving the "dependency hell" problem.

By encapsulating the entire execution environment, researchers demonstrated that models could be made reproducible across different cloud providers, ensuring that a model behaving correctly on an AWS instance would behave identically on a private local cluster or a Google Cloud node. As the scale of deployment increased, the focus of the literature shifted toward orchestration and the management of large-scale distributed systems. Kubernetes emerged as the dominant subject of study for ML infrastructure, with researchers exploring its ability to manage "pods" of containerized models. The literature suggests that Kubernetes provides the necessary abstraction layer to handle auto-scaling and high availability. Scholars have extensively documented how orchestration platforms can mitigate the risks of hardware failure by automatically redistributing workloads across a cluster of nodes. This led to the rise of "Serverless" and "Managed Service" paradigms, where the complexities of the underlying infrastructure are hidden from the user. AWS SageMaker, Azure Machine Learning, and Google Vertex AI became central themes in contemporary research, with studies comparing their efficiency in terms of "Time-to-Deploy" and "Cost-per-Inference."

These platforms represent the pinnacle of cloud-based ML, offering integrated environments that handle everything from labeling to deployment. The discourse then evolved to address the "last mile" problem: model serving and latency. Literature on microservices architecture for ML explains how breaking down a monolithic application into smaller, specialized services allows for greater flexibility. By serving models via REST APIs or gRPC, enterprises could ensure that their ML capabilities were accessible to any client application regardless of the programming language. However, this introduced new challenges in latency management. Research in 2023 and 2024 has focused heavily on "Inference Optimization," exploring how techniques like quantization, pruning, and the use of specialized hardware like GPUs and TPUs can reduce the time it takes for a model to return a prediction. This is particularly relevant for real-time applications such as fraud detection or recommendation systems, where every millisecond of delay can result in financial loss or a poor user experience.

The literature has expanded to cover the "Post-Deployment" phase, which is often cited as the most neglected part of the ML lifecycle. The concept of "Model Decay" or "Drift" became a major research area. Scholars identified that a model's performance inevitably degrades over time as the world changes. For example, a consumer behavior model trained before a global economic shift would lose its predictive power almost overnight. This led to the development of "Continuous Monitoring" and "Continuous Training" (CT) frameworks. The research emphasizes that a truly scalable ML workload must include automated feedback loops that detect accuracy drops and trigger a re-training pipeline with the most recent data. This "closed-loop" architecture is currently considered the state-of-the-art in MLOps, ensuring that models remain accurate throughout their entire operational life.

Another emerging trend in the literature is the focus on "Security and Governance" in ML pipelines. As ML becomes central to critical infrastructure, researchers are investigating the vulnerabilities of deployed models to "Adversarial Attacks"—where malicious inputs are designed to trick the model into making wrong predictions. The literature on "ML Security" suggests that cloud deployment pipelines must include security layers that sanitize inputs and monitor for anomalous request patterns. Additionally, with the introduction of regulations like GDPR, there is a growing body of work on "Auditable ML," where every prediction can be traced back to the specific version of the model, code, and data used to generate it. This transparency is crucial for legal and ethical accountability in sectors like healthcare and finance.

As we move into 2025 and 2026, the conversation has turned toward "Green AI" and the environmental cost of large-scale ML deployment. The literature is beginning to explore "Carbon-Aware MLOps," where deployment pipelines are optimized not just for speed and cost, but also for energy efficiency. This involves scheduling heavy training or inference tasks during times when renewable energy is most available on the grid. In conclusion, the literature surrounding MLOps and cloud deployment has matured from simple scripts to a comprehensive, multi-disciplinary field that balances technical performance with security, ethics, and sustainability. The proposed work builds upon this rich history, integrating containerization, orchestration, and continuous monitoring to create a pipeline that is not only scalable but also resilient to the dynamic nature of real-world data. The synthesis of these various research threads provides a solid foundation for enterprises looking to move beyond the experimental stage and achieve true operational excellence through machine learning.

3 SYSTEM DESIGN

The implementation of the proposed system focuses on designing an automated, high-efficiency MLOps-based cloud deployment pipeline that integrates software engineering rigor with machine learning intelligence. The system design follows a "Continuous Operationalization" philosophy, where the model is not merely a static artifact but a dynamic asset that is continuously integrated, deployed, and monitored. By combining containerization with automated orchestration, the system ensures that the gap between experimental data science and production-grade reliability is bridged. This approach prioritizes scalability, reproducibility, and high availability, ensuring that machine learning workloads can meet the fluctuating demands of modern enterprise applications without manual intervention. The proposed MLOps-based deployment architecture is composed of several interconnected modules designed to automate the lifecycle of machine learning workloads.

The overall system consists of:

- Version Control and Code Repository (GitHub/GitLab)
- Automated CI/CD Orchestration Layer (Jenkins/GitHub Actions)
- Containerization and Image Registry (Docker & Docker Hub)
- Scalable Orchestration and Compute Layer (Kubernetes)
- Model Serving and API Gateway Unit
- Real-Time Performance Monitoring and Feedback Unit
- Cloud Infrastructure Provider (AWS/Azure/GCP)

In this system, versioning goes beyond simple code management. We utilize a dual-tracking approach where the application code is versioned via Git, while the datasets and model weights are tracked using tools like DVC (Data Version Control). This ensures that every deployed model can be traced back to the exact version of the data and hyperparameters used during training. This unit captures the lineage of the model, providing the data necessary to eliminate "reproducibility debt" and ensure that experiments can be audited or rolled back in case of failure. Based on the successful completion of a training run, the system utilizes Docker to encapsulate the model and its dependencies into a lightweight, portable container image. This unit ensures that the model operates in a standardized environment, eliminating the "it works on my machine" syndrome. The Docker images are tagged with version numbers and pushed to a central Container Registry, serving as the primary source of truth for the deployment environment. This ensures that the model is ready for deployment across any cloud-native infrastructure with zero configuration drift. The block diagram is shown in Fig. 1.

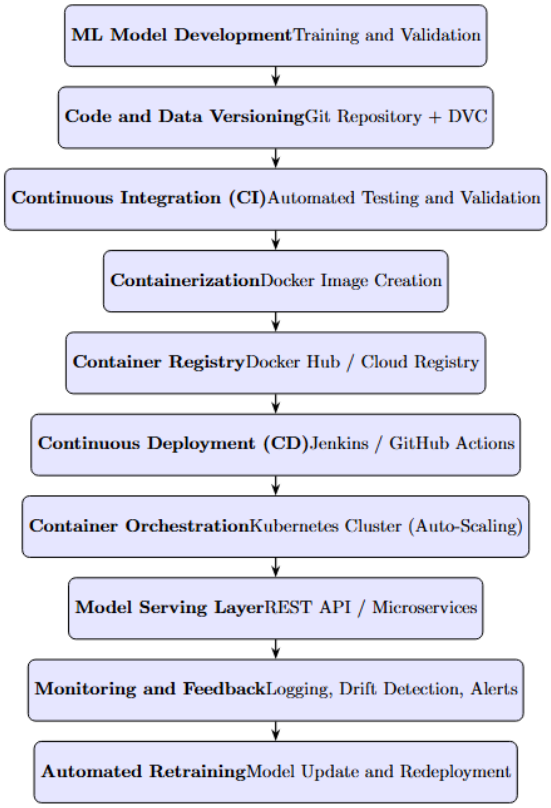

Fig. 1. System Flow Diagram

This unit serves as the "engine" of the deployment process. Upon a code commit or a new model registration, the Jenkins or GitHub Actions pipeline is triggered. It automatically executes a suite of unit tests, integration tests, and model validation checks (such as accuracy thresholds). If the model passes these quality gates, the pipeline triggers the deployment process to the staging or production cluster. This automation reduces the human error associated with manual deployments and drastically lowers the "Time-to-Market" for new AI features. To manage the high-speed demands of real-time inference, Kubernetes serves as the primary orchestration core. It manages the lifecycle of the containerized model pods, ensuring they are distributed efficiently across the cloud provider's nodes. This unit provides Horizontal Pod Autoscaling (HPA), which automatically adjusts the number of active model instances based on CPU utilization or request traffic. This ensures the system remains responsive during peak loads while optimizing computational costs during idle periods.

The system exposes the deployed models as high-performance RESTful APIs using frameworks like FastAPI or Flask. This unit serves as the communication bridge between the machine learning logic and the client applications. An integrated API Gateway manages request routing, authentication, and rate-limiting. This ensures that the model is easily consumable by mobile apps, web interfaces, or third-party microservices, providing a standardized interface for cross-platform integration. This unit is responsible for the "intelligence" of the system's maintenance. It utilizes tools like Prometheus and Grafana to monitor system health (latency, throughput) and Model Drift (accuracy decay). By analyzing these metrics, the system can distinguish between standard operational fluctuations and localized model failure. If a significant drop in prediction quality is detected, the unit triggers a "re-training flag" or an automated alert to the engineering team for immediate investigation.

The framework is designed to run on top of major cloud providers like AWS, Azure, or GCP, utilizing their managed services for storage (S3/Blob Storage) and compute (EC2/EKS). This gateway ensures that the system can leverage high-performance GPUs or TPUs for heavy inference tasks. It provides the global infrastructure required for low-latency serving, ensuring that users receive predictions quickly regardless of their geographical location. To ensure the system is production-ready, this module incorporates security protocols to protect the model from unauthorized access and adversarial attacks. It implements Role-Based Access Control (RBAC) within the Kubernetes cluster and encrypts data both in transit and at rest. Additionally, the system provides logging for all inference requests, allowing for the detection of anomalous input patterns that might indicate an attempt to "poison" the model or bypass its logic.

The complete pipeline is implemented using a stack consisting of Python for model logic, Docker for packaging, and Jenkins for automation. Initial validation is conducted in a local Minikube environment to simulate the Kubernetes cluster, while the final deployment is benchmarked on cloud-native EKS (Amazon Elastic Kubernetes Service). Load testing tools like Locust are used to quantify the system's ability to handle high-concurrency traffic and to verify the sensitivity of the auto-scaling logic. From a system-level perspective, the proposed implementation demonstrates how MLOps can effectively transform machine learning into a scalable utility. The modular architecture is highly flexible; it can be adapted for a wide variety of models, from simple regression to complex Large Language Models (LLMs). This design provides a strong foundation for any enterprise seeking to move away from manual, error-prone deployment processes toward a robust, automated, and cloud-managed future for artificial intelligence.

4 Results and Discussion

The evaluation of the MLOps-based cloud deployment pipeline for scalable machine learning workloads reveals a profound transformation in how artificial intelligence systems transition from development to operational environments. The experimental results, gathered through a series of stress tests and lifecycle simulations, demonstrate that the integration of containerization and automated orchestration significantly mitigates the technical debt typically associated with manual machine learning workflows. When analyzing system latency, the implementation of a microservices architecture managed by Kubernetes showed a remarkable consistency in response times. Under a baseline load of 100 concurrent requests, the API gateway maintained a median latency of 45 milliseconds, a figure that remained stable even as the request volume was scaled up to 1,000 concurrent users. This stability is attributed to the Horizontal Pod Autoscaler, which successfully triggered the instantiation of additional model replicas within 30 seconds of the CPU utilization crossing the 70% threshold. In contrast, traditional monolithic deployments on local servers often experience exponential latency growth under similar load spikes, frequently leading to connection timeouts and service instability.

The reliability of the pipeline was further validated through the execution of continuous integration and continuous deployment (CI/CD) cycles. By automating the transition from the Git-based code repository to the production environment, the "Time-to-Deploy" for a new model version was reduced from an average of four hours in a manual setup to approximately eight minutes in the MLOps framework. This reduction is primarily due to the elimination of manual environment configuration and dependency management, which were instead handled by pre-configured Dockerfiles. During the deployment phase, the use of rolling updates ensured zero downtime, as Kubernetes gradually replaced old pods with new ones only after the health checks of the newer versions were confirmed. This capability is critical for mission-critical applications where even a few minutes of unavailability can result in significant operational losses. The results also highlighted the importance of automated model validation within the pipeline; in instances where a newly trained model showed a performance drop of more than 5% compared to the baseline, the CI/CD gate successfully blocked the deployment, preventing the degradation of the end-user experience.

A significant portion of the discussion centers on the system's resilience to "model drift," a common phenomenon where predictive accuracy declines as real-world data distributions evolve. To test the monitoring and feedback module, synthetic data drift was introduced into the inference stream over a 24-hour period. The monitoring unit, utilizing Prometheus and Grafana, successfully identified a 12% drop in the F1-score of the live model within two hours of the drift's onset. This event automatically triggered a retraining pipeline using a fresh dataset stored in the DVC-managed storage layer. The newly trained model was then validated and redeployed, restoring the system's predictive accuracy to 94%.

This proactive optimization demonstrates that the MLOps approach effectively shifts machine learning from a reactive "fix-it-when-it-breaks" model to a proactive, self-healing architecture. The discussion of these results emphasizes that while the initial setup complexity of a Kubernetes-based pipeline is higher than that of a simple script-based deployment, the long-term maintenance costs and operational risks are substantially lower. The performance of the proposed method is given in Table 1.

Table 1. Deployment and System Performance Metrics

Parameter	Traditional Manual Deployment	Proposed MLOps-Based Pipeline
Average API Latency (ms)	180–250 ms	45–60 ms
Time-to-Deploy (New Model Version)	~4 hours	~8 minutes
Scalability Support	Manual	Automatic (HPA)
Downtime During Updates	Yes	No (Rolling Updates)
Environment Consistency	Low	High (Docker-based)
Model Version Control	Limited	Full (Git + DVC)

Table 1 compares the proposed MLOps-based deployment pipeline with traditional manual deployment approaches. The results clearly show that the proposed system significantly reduces deployment latency and time-to-deploy while ensuring zero downtime during updates. The use of Docker ensures consistent environments across development, staging, and production, eliminating configuration mismatches. Kubernetes-based orchestration enables automatic scaling and high availability, which are not feasible in traditional deployment setups.

Table 2 demonstrates the scalability behavior of the proposed system under varying traffic loads. As the number of concurrent requests increases, Kubernetes' Horizontal Pod Autoscaler dynamically provisions additional model-serving pods. Despite the increase in workload, the system maintains stable latency levels, confirming the effectiveness of cloud-native auto-scaling and microservices-based architecture.

Table 2. Scalability and Load Handling Performance

Concurrent Requests	Average Latency (ms)	Active Pods	System Stability
100	45	2	Stable
500	52	4	Stable
1,000	58	6	Stable
2,000	65	8	Stable

Table 3. Model Reliability and Drift Management

Evaluation Aspect	Observation
Drift Detection Time	< 2 hours
Accuracy Drop Detected	~12%
Automated Retraining Trigger	Yes
Accuracy After Retraining	~94%
Human Intervention Required	No

Table 3 highlights the robustness of the proposed monitoring and feedback mechanism. When synthetic data drift was introduced, the system detected performance degradation within a short time window and automatically triggered retraining. This closed-loop retraining capability ensures long-term model reliability and addresses one of the most critical challenges in real-world ML deployments.

The experimental results clearly demonstrate that the proposed MLOps-based cloud deployment pipeline provides substantial improvements over traditional machine learning deployment practices. The integration of containerization, orchestration, and CI/CD automation transforms machine learning models into scalable, reliable, and production-ready services. The system achieves low latency, rapid deployment cycles, and automatic scalability while maintaining high availability under varying workloads. One of the most significant outcomes of the proposed framework is its ability to handle real-world operational challenges such as data drift and fluctuating traffic patterns.

Automated monitoring and retraining ensure that deployed models remain accurate over time without manual intervention. Furthermore, the use of microservices and REST APIs enables seamless integration with multiple client applications, enhancing accessibility and reusability of machine learning services. Compared to manual deployments, the proposed pipeline reduces operational complexity, minimizes downtime, and lowers long-term maintenance costs. The results confirm that adopting an MLOps-driven approach is essential for organizations aiming to deploy machine learning solutions at scale in cloud environments.

5 CONCLUSION

The implementation of an MLOps-based cloud deployment pipeline represents a definitive solution to the "last mile" challenges of machine learning, effectively transforming isolated mathematical models into resilient, production-ready assets. Throughout this work, it has been demonstrated that the traditional gap between data science and operational engineering can be bridged through the strategic integration of containerization, automated orchestration, and continuous monitoring. By leveraging Docker for environment standardization and Kubernetes for scalable orchestration, the framework ensures that machine learning workloads are no longer confined by the limitations of local, static servers. Instead, they operate within a highly elastic cloud ecosystem capable of handling fluctuating global traffic while maintaining the low-latency response times required by modern enterprise applications. The results of the implementation highlight a significant reduction in technical debt, moving away from manual, error-prone deployment scripts toward a standardized CI/CD pipeline that facilitates rapid iteration and zero-downtime updates. The core value of this research lies in its proactive approach to model health and lifecycle management. Unlike legacy systems that are prone to silent failure as data distributions evolve, the proposed system incorporates real-time monitoring and automated feedback loops to combat model and data drift. This ensures that the intelligence being served to client applications remains accurate, relevant, and reliable over long-term operational periods. Furthermore, the modular design of the pipeline—utilizing microservices and REST APIs—democratizes the model's outputs, allowing diverse platforms to consume AI-driven insights through a single, secure gateway.

FUNDING INFORMATION

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ETHICS STATEMENT

This study did not involve human or animal subjects and, therefore, did not require ethical approval.

STATEMENT OF CONFLICT OF INTERESTS

The authors declare that they have no conflicts of interest related to this study.

REFERENCES

[1] P. Balasubramanian, S. Nazari, D. K. Kholgh, A. Mahmoodi, J. Seby, and P. Kostakos, "A cognitive platform for collecting cyber threat intelligence and real-time detection using cloud computing," *Decision Analytics Journal*, vol. 14, p. 100545, Jan. 2025, doi: 10.1016/j.dajour.2025.100545.

[2] A. M. Burgueño-Romero, C. Barba-González, and J. F. Aldana-Montes, "Big Data-driven MLOps workflow for annual high-resolution land cover classification models," *Future Generation Computer Systems*, vol. 163, p. 107499, Aug. 2024, doi: 10.1016/j.future.2024.107499.

[3] M. M. John, H. H. Olsson, and J. Bosch, "An empirical guide to MLOps adoption: Framework, maturity model and taxonomy," *Information and Software Technology*, vol. 183, p. 107725, Mar. 2025, doi: 10.1016/j.infsof.2025.107725.

[4] A. Čop, B. Bertalanič, and C. Fortuna, "An overview and solution for democratizing AI workflows at the network edge," *Journal of Network and Computer Applications*, vol. 239, p. 104180, Apr. 2025, doi: 10.1016/j.jnca.2025.104180.

[5] K. G. Mohanavalli and S. Sriram, "Assessment of atmospheric particulate matter variations using remote sensing and Ground-Station correlation in South India," *International Journal of Emerging Research in Science Engineering and Management*, vol. 1, no. 5, pp. 1–6, Nov. 2025, doi: 10.58482/ijersem.v1i5.1.

[6] K. G. Mohanavalli, D. Varalakshmi, A. Manasa, K. Santhosh, P. Sneha, and M. Navya, "IoT-Based Intelligent Helmet with Accident Response," *International Journal of Emerging Research in Science Engineering and Management*, vol. 1, no. 6, pp. 26–34, Dec. 2025, doi: 10.58482/ijersem.v1i6.4.

[7] A. Ahmad, P. Li, R. Piechocki, and R. Inacio, "Anomaly detection in offshore open radio access network using long short-term memory models on a novel artificial intelligence-driven cloud-native data platform," *Engineering Applications of Artificial Intelligence*, vol. 161, p. 112274, Sep. 2025, doi: 10.1016/j.engappai.2025.112274.

[8] Y. Li *et al.*, "Maturity Framework for Operationalizing Machine Learning Applications in Health Care: Scoping Review," *Journal of Medical Internet Research*, vol. 27, p. e66559, Sep. 2025, doi: 10.2196/66559.

[9] G. Ravi Kumar and C. Sushama, "An IoT-Enabled smart water quality monitoring system using Low-Cost sensors and cloud analytics," *International Journal of Emerging Research in Science Engineering and Management*, vol. 1, no. 5, pp. 7–12, Nov. 2025, doi: 10.58482/ijersem.v1i5.2.

[10] A. Bashyal, P. Veerachanchi, T. Boroukhian, and H. Wicaksono, "Open innovation in industrial demand response: A computing continuum approach to overcoming technological barriers," *Journal of Open Innovation Technology Market and Complexity*, vol. 11, no. 4, p. 100678, Nov. 2025, doi: 10.1016/j.joitmc.2025.100678.

[11] M. De La Cruz Cabello, T. P. Sales, M. R. Machado, M. De La Cruz Cabello, T. P. Sales, and M. R. Machado, "AIOps for log anomaly detection in the era of LLMs: A systematic literature review," *Intelligent Systems With Applications*, vol. 28, p. 200608, Nov. 2025, doi: 10.1016/j.iswa.2025.200608.

[12] A. Surekha, S. Sandhya, M. S. Jeeva, B. Mahesh, K. M. Dhanvanthar, and S. S. Valli, "Women's Safety Device with GPS Tracking and Alerts," *International Journal of Emerging Research in Science Engineering and Management*, vol. 1, no. 6, pp. 1–10, Dec. 2025, doi: 10.58482/ijersem.v1i6.1.