International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

# Automated Machine Learning Deployment System Using MLOps

*Thati Venkata M. Lakshmi[1], D. Sai Prabath[2], P. Vignesh[3], K. Bala Krishna[4], J. Kumar Vardhan[5]*
*[1]Assistant Professor, Department of CSE - Artificial Intelligence and Machine Learning, SRK Institute of Technology, Vijayawada - 521108, Andhra Pradesh, India*
*[2,3,4,5]Students, Department of CSE - Artificial Intelligence and Machine Learning, SRK Institute of Technology, Vijayawada - 521108, Andhra Pradesh, India*
***Emails:*** *srktmlcsm4@gmail.com[1], saiprabathdamarla@gmail.com[2], vigneshmohith2@gmail.com[3], kazabalakrishna4@gmail.com[4], jkumarvardhan@gmail.com[5]*

## Abstract
*The increasing adoption of Machine Learning (ML) in real-world applications has exposed significant challenges in deploying models reliably and reproducibly at scale. While most ML workflows primarily focus on model development and evaluation, operational aspects such as experiment tracking, version control, automated retraining, and continuous deployment are often inadequately addressed. This paper presents an Automated Machine Learning Deployment framework grounded in MLOps principles to bridge the gap between model development and production deployment. The proposed system automates the end-to-end ML lifecycle, including data ingestion, multi-model training, performance evaluation, and best-model selection. Data Version Control (DVC) is employed to manage dataset versions, while batch-based data drift monitoring is implemented using Evidently AI to analyze distributional changes across training iterations. MLflow is utilized for systematic experiment tracking and model versioning, ensuring reproducibility and traceability of results. Continuous Integration and Continuous Deployment (CI/CD) are implemented using GitHub Actions to enable automated retraining and seamless deployment upon code or data updates. The selected model is containerized using Docker and deployed as a FastAPI-based REST service on an AWS EC2 instance. The proposed framework demonstrates improved deployment reliability, scalability, and reproducibility through automated MLOps practices, making it suitable for real-world machine learning applications.*

***Keywords:*** *Continuous Integration and Continuous Deployment (CI/CD); Data Drift Detection; Experiment Tracking; Machine Learning Deployment; MLOps.*

## 1. Introduction

The rapid adoption of Machine Learning (ML) across diverse application domains has led to an increased demand for reliable and scalable deployment of ML models in production environments. While significant progress has been achieved in model development and algorithmic performance, deploying ML models in real-world systems remains a complex and error-prone process. Challenges such as experiment reproducibility, data versioning, model traceability, automated retraining, and deployment consistency are often insufficiently addressed in traditional ML workflows.Conventional ML pipelines are largely manual and fragmented, requiring separate handling of data preparation, model training, evaluation, deployment, and monitoring. As a result, models that perform well during experimentation may fail to maintain reliability when exposed to evolving data distributions or production constraints. In particular, changes in input data characteristics, commonly referred to as data drift, can significantly degrade model performance if left undetected. These limitations highlight the need for systematic approaches that integrate development and operational concerns throughout the ML lifecycle. Machine Learning Operations (MLOps) has emerged as a paradigm to address these challenges by combining best practices from software engineering, DevOps, and ML system design. However, many existing implementations focus either on isolated components such as model tracking or deployment automation, while comprehensive end-to-end lifecycle management remains underexplored in practical academic implementations. This paper

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

presents an automated machine learning deployment framework based on MLOps principles that aims to bridge the gap between model development and production deployment. The proposed system enhances reproducibility, scalability, and maintainability, making it suitable for real-world machine learning applications.

## 2. Related Work

The increasing complexity of deploying machine learning systems in production environments has highlighted significant challenges related to reproducibility, scalability, and technical debt. Early research identified the issue of hidden technical debt in ML systems, emphasizing the risks associated with poorly managed pipelines and ad hoc deployment practices [1]. To address production readiness, structured evaluation frameworks such as the ML Test Score were introduced to assess reliability and maintainability of ML systems [2]. Further studies explored deployment challenges including operational scalability, and system integration in real-world applications [3]. As ML systems evolved beyond experimentation, robust model development and optimization became critical. Foundational tools such as Scikit-learn supported structured model development workflows [4]. Research on hyperparameter optimization improved systematic tuning strategies [5], while studies on overfitting emphasized the importance of reliable model evaluation and generalization performance [6]. Managing the full lifecycle of models became critical. Research on ML pipeline automation addressed scalability and workflow orchestration challenges [7]. MLflow was introduced to streamline experiment tracking, artifact management, and model registry operations, enabling reproducible experimentation and systematic comparison of model runs [8]. Alongside experiment tracking, dataset versioning emerged as a key requirement for reproducible ML workflows. Data Version Control (DVC) provides mechanisms to track and manage dataset revisions in synchronization with source code changes [9]. Automation through Continuous Integration and Continuous Deployment (CI/CD) has also been adapted for ML systems to ensure consistent training, validation, and deployment processes. Research highlights the importance of automated pipelines to reduce manual intervention and maintain workflow reliability [10]. Tools such as GitHub Actions facilitate automated orchestration of build and deployment stages within ML projects [11]. Containerization technologies have further enhanced deployment consistency. Docker enables packaging of ML applications with their dependencies to ensure uniform behavior across development and production environments [12]. The role of containerization within cloud-native architectures has been extensively discussed as a foundation for scalable ML deployment [13]. Cloud infrastructure plays a crucial role in enabling remote model hosting and scalable inference services. Cloud-based ML deployment architectures support flexible resource allocation and operational scalability [14]. Services such as Amazon EC2 provide virtualized infrastructure for hosting containerized ML applications in production settings [15]. Finally, monitoring and concept drift detection have become essential components of operational ML systems. Concept drift adaption techniques address changes in data distributions over time [16]. Modern monitoring frameworks such as Evidently AI provide tools for tracking model performance and detecting data drift in deployed environments [17]. Together, these studies establish the foundational principles of MLOps, highlighting the importance of lifecycle management, automation, containerization, cloud deployment, and monitoring. The proposed framework builds upon these concepts to deliver an integrated and reproducible end-to-end ML deployment architecture.

## 3. Proposed Methodology

The proposed system is an automated machine learning deployment framework designed to operationalize machine learning models using MLOps principles. The framework integrates data management, experiment tracking, monitoring, continuous integration, and deployment automation into a unified and reproducible pipeline. The overall architecture is modular, enabling independent development and maintenance of each component while ensuring seamless interaction across the machine learning lifecycle.The workflow begins with data ingestion and version management. Raw datasets are ingested from source repositories and

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

managed using Data Version Control (DVC), which enables systematic tracking of dataset versions across experiments. Each dataset update is uniquely versioned and linked to corresponding training runs, ensuring full traceability between data, models, and experimental outcomes. This approach supports reproducible model training and facilitates comparison across different data iterations.Following data versioning, the system performs automated model training and evaluation. Multiple machine learning models are trained using the versioned datasets, and their performance is evaluated based on predefined metrics, and trained model artifacts are generated. This enables systematic comparison of model performance and supports automated selection of the best-performing model for deployment.To address data quality and reliability concerns, the proposed system incorporates batch-based data drift monitoring. Evidently AI is used to analyze statistical differences between reference datasets and newly ingested data. Drift reports are generated during pipeline execution to identify significant distributional changes across training iterations. Based on drift detection outcomes, the system supports informed retraining decisions as part of the automated workflow.Automation of retraining and deployment is achieved through continuous integration and continuous deployment (CI/CD) pipelines implemented using GitHub Actions. The pipelines are triggered by code changes or dataset updates tracked through version control. Upon successful completion of training and validation stages, the selected model is automatically packaged and prepared for deployment, reducing manual intervention and minimizing deployment errors.For deployment, the selected model is containerized using Docker to ensure environment consistency and portability. The containerized model is deployed as a RESTful service using FastAPI, providing a standardized interface for inference requests. The service is hosted on an AWS EC2 instance, enabling remote access and scalable deployment in a cloud-based environment.Overall, the proposed system delivers an end-to-end automated MLOps framework that integrates data versioning, batch-based monitoring, experiment tracking, CI/CD automation, and containerized deployment. By unifying these components within a single pipeline, the system enhances reproducibility, maintainability, and deployment reliability, making it suitable for real-world machine learning applications. Figure 1 shows System Architecture for the proposed system

## 4. System Architecture

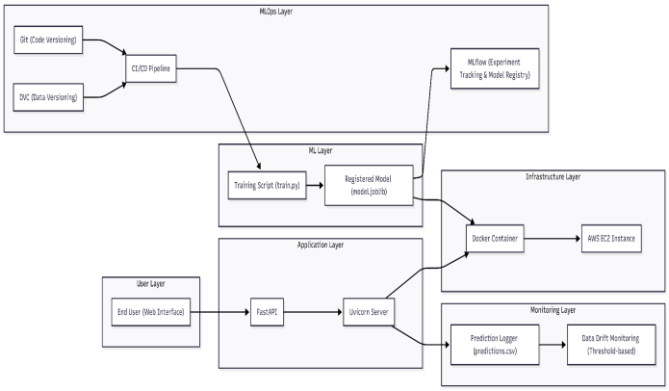


**Figure 1 System Architecture for the proposed system**

The system architecture of the proposed automated machine learning deployment framework is designed using a layered approach to ensure modularity, scalability, and clear separation of responsibilities. As illustrated in Fig. 1, the architecture is composed of six primary layers: User Layer, Application Layer, ML Layer, MLOps Layer, Monitoring Layer, and Infrastructure Layer. Each layer plays a distinct role in supporting the end-to-end machine learning lifecycle.The User Layer represents external users or client applications that interact with the system by submitting inference requests. User inputs are forwarded to the application layer through a RESTful interface, enabling seamless communication between users and the deployed machine learning service. The Application Layer is responsible for handling request routing and inference execution. It consists of a FastAPI-based web service served by a Uvicorn application server. FastAPI provides a lightweight and efficient REST interface, while Uvicorn ensures high-performance asynchronous request handling. Incoming user requests are validated and forwarded to the trained machine learning model for prediction. The ML Layer encapsulates the trained machine learning model, which is stored as a serialized artifact (e.g., model.joblib). This layer is responsible for

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

loading the selected model and generating predictions based on incoming requests from the application layer. The ML layer is decoupled from the application logic, enabling independent model updates without affecting the service interface. The MLOps Layer forms the core of the automated deployment framework. It integrates Git for source code versioning, Data Version Control (DVC) for dataset version management, and MLflow for experiment tracking and model registry management. During model development, DVC tracks changes in datasets, while MLflow logs training parameters, evaluation metrics, and model artifacts. The CI/CD pipeline orchestrated through GitHub Actions automates model retraining, validation, and deployment whenever code or data updates are detected. The Monitoring Layer is responsible for tracking prediction outputs and detecting data drift. Prediction logs are stored in structured files (e.g., predictions.csv), which serve as input for drift analysis. Batch-based data drift monitoring is implemented using threshold-based techniques to identify distributional changes between reference data and newly observed data. This monitoring process supports informed retraining decisions without requiring continuous real-time monitoring infrastructure. The Infrastructure Layer provides the runtime environment for deployment. The application and model are containerized using Docker to ensure environment consistency and portability. The Docker container is deployed on an AWS EC2 instance, enabling remote access, scalable execution, and cloud-based hosting of the inference service. Overall, the proposed architecture integrates application serving, model management, monitoring, and deployment automation within a unified MLOps framework. By clearly separating concerns across layers and automating interactions between them, the system achieves reproducibility, maintainability, and reliable production deployment of machine learning models.

## 5. Results

This section presents the experimental results of the automated machine learning deployment system.The results validate the effectiveness of the model comparison, experiment tracking, and end-to-end deployment in a production-ready environment.

### 5.1.Model Performance and Experiment Tracking

Multiple model training runs were executed and systematically logged using an experiment tracking framework.Each run recorded evaluation metrics and parameters, enabling consistent comparison and reproducibility. **Fig. 2** illustrates the complete history of recorded experiment runs, confirming successful experiment tracking. The performance of the individual models is illustrated in **Fig. 3.** and **Fig. 4**. While both models achieved comparable accuracy, noticeable differences were observed in precision, recall, F1-score. Based on these metrics, the system automatically selected and registered the best-performing model, as shown in Fig. 5. A consolidated comparison of evaluation metrics is presented in Fig. 6, clearly justifying the final model selection decision.

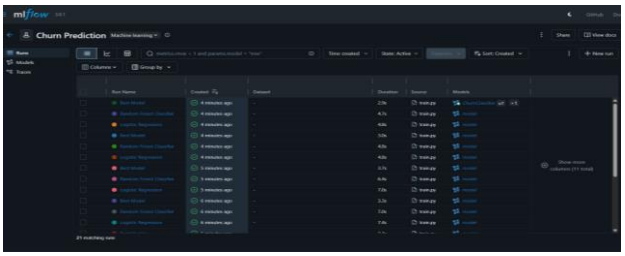


**Figure 2 ML flow Experiment Runs History**

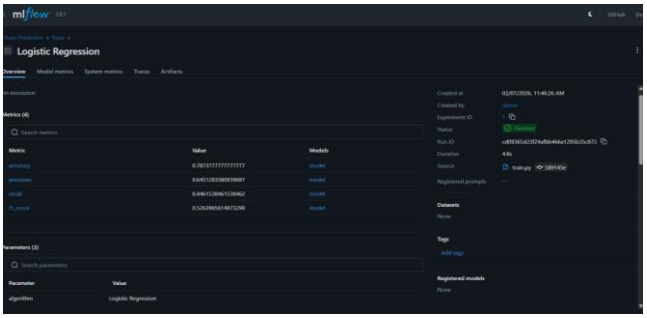


**Figure 3 Logistic Regression Performance Metrics**

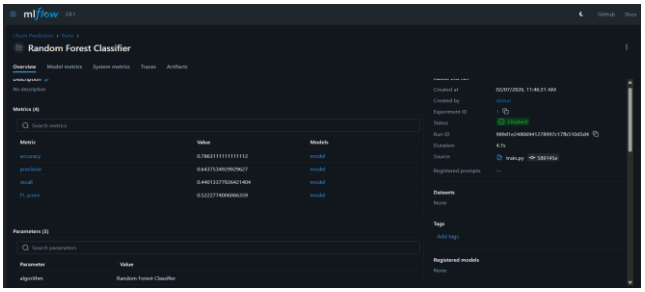


**Figure 4 Random Forest Performance Metrics**

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

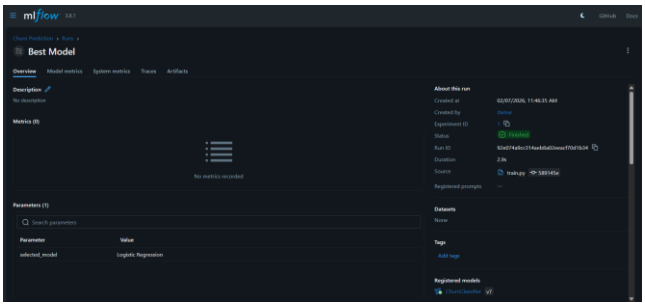

**Figure 5** Best Model Selection And Registration

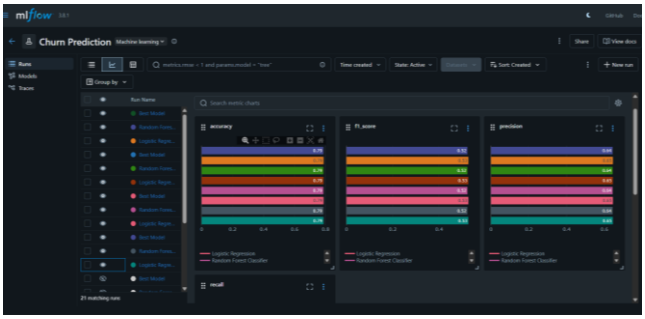

**Figure 6** Comparative Model Performance Metrics

### 5.2. Deployment and Inference Validation

The selected model was containerized using Docker to ensure consistent execution across environments. The containerized application was deployed on a cloud-based virtual machine to support remote access and scalable inference. Fig. 7 presents the FastAPI Swagger interface, illustrating the available GET and POST endpoints for inference. Fig. 8 shows the successful execution of a prediction request and the corresponding response, validating real-time inference through the deployed service. The active cloud deployment is shown in Fig. 9, confirming that the application is running on an AWS EC2 instance. These results demonstrate reliable containerized deployment, cloud hosting, and real-time prediction serving, indicating the system's readiness for practical use.

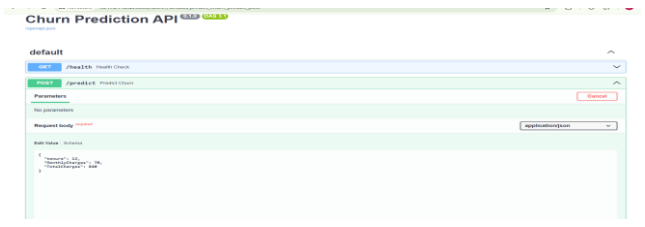

**Figure 7** Fastapi Swagger API Endpoints

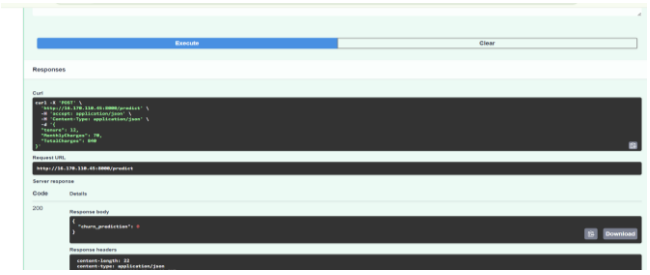

**Figure 8** Prediction Output From Deployed Service

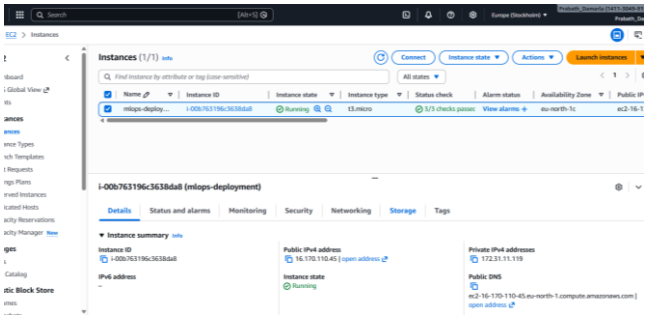

**Figure 9** AWS EC2 Instance Running The Application

### 5.3. CI/CD Automation Results

An automated CI/CDpipeline was implemented to support continuousintegration and deployment of the system. The pipeline is triggered automatically on code updates and orchestrates tasks including dependency installation, model training, evaluation, container build, and deployment. Fig. 10 presents the workflow execution history, showing both successful and failed runs, demonstrating pipeline robustness, traceability, and error visibility. Fig. 11 illustrates a complete end-to-end successful pipeline execution, from job setup to deployment completion. These results confirm reliable automation of the training and deployment process with minimal manual intervention.

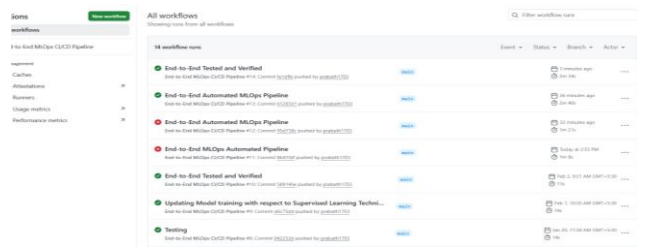

**Figure 10** CI/CD workflow execution history

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

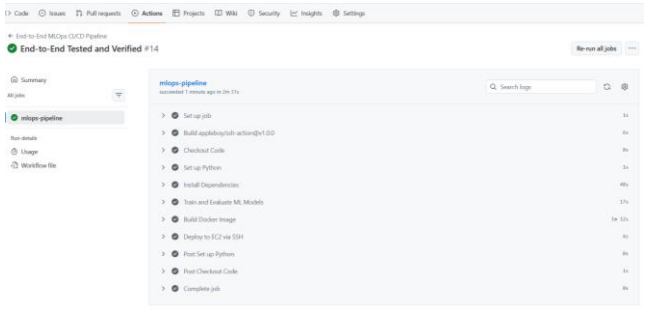

**Figure 11 Successful end-to-end CI/CD pipeline execution**

### 5.4. Monitoring and Drift Detection

Batch-based monitoring was implemented to track changes in data distribution over time. Prediction outputs were compared with reference training data to detect drift. Fig. 12 presents the data drift report generated during monitoring, demonstrating the system's capability to identify drift and support informed retraining decisions.

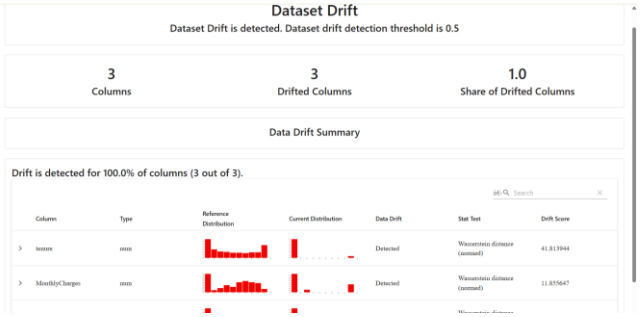


**Figure 12 Data drift detection report**

## Conclusion And Future Scope

This paper presented an automated machine learning deployment framework based on MLOps principles to address challenges in reproducibility, deployment reliability, and operational scalability. The proposed system integrates dataset versioning, multi-model training, experiment tracking, automated CI/CD pipelines, containerized deployment, cloud hosting, and batch-based monitoring within a single end-to-end architecture. Experimental results demonstrate effective model comparison, reproducible experimentation, and reliable real-time inference through a deployed RESTful service. By minimizing manual intervention and ensuring deployment consistency, the framework provides a practical foundation for deploying machine learning models in real-world production environments.

## Future Scope

Future work will focus on extending the framework with continuous data and model monitoring to enable automated retraining and adaptive model updates. Support for additional machine learning algorithms and automated hyperparameter optimization techniques can further improve model performance. The deployment architecture can be enhanced using container orchestration platforms to support high-traffic applications. Additionally, integrating advanced observability, security, and access-control mechanisms will strengthen the framework for enterprise-grade machine learning deployments.

## REFERENCES

[1]. D. Sculley et al., "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2503–2511.

[2]. E. Breck et al., "The ML test score: A rubric for ML production readiness and technical debt reduction," in Proc. IEEE Big Data, 2017, pp. 1123–1132.

[3]. L. M. H. Nguyen, G. Varghese, and A. N. Ananthakrishnan, "Machine learning deployment challenges and solutions," IEEE Software, vol. 37, no. 4, pp. 42–49, 2020.

[4]. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[5]. J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research, vol. 13, pp. 281–305, 2012.

[6]. G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," Journal of Machine Learning Research, vol. 11, pp. 2079–2107, 2010.

[7]. C. Zhang, C. Ré, M. Cafarella, and J. Shin, "ML pipelines: Automation and scalability challenges," in Proc. ACM SIGMOD, 2019, pp. 1–4.

[8]. M. Zaharia et al., "Accelerating the machine learning lifecycle with MLflow," IEEE Data Engineering Bulletin, vol. 41, no. 4, pp. 39–45, 2018.

[9]. DVC.org, "Data Version Control for machine learning projects," 2023. [Online]. Available: https://dvc.org

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 03 March 2026
Page No: 1409-1415
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0193

[10]. S. Ramírez and J. J. Durillo, "Continuous integration and deployment for machine learning systems," in Proc. IEEE ICMLA, 2019, pp. 1–6.
[11]. GitHub, "GitHub Actions: CI/CD for automation," 2023. [Online]. Available: https://docs.github.com/actions
[12]. D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," Linux Journal, vol. 2014, no. 239, 2014.
[13]. S. Pahl, "Containerization and the PaaS cloud," IEEE Cloud Computing, vol. 2, no. 3, pp. 24–31, 2015.
[14]. S. Shen, A. V. Iyengar, and J. Rao, "Cloud-based deployment of machine learning models," IEEE Internet Computing, vol. 24, no. 6, pp. 18–26, 2020.
[15]. Amazon Web Services, "Amazon EC2: Scalable cloud computing," 2023. [Online]. Available: https://aws.amazon.com/ec2
[16]. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys, vol. 46, no. 4, pp. 1–37, 2014.
[17]. Evidently AI, "Evidently: Open-source ML monitoring and observability," 2023. [Online]. Available: https://www.evidentlyai.com