International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

A Systematic Framework for Experiment Tracking and Model Promotion in Enterprise MLOps Using MLflow and Databricks

Janardhan Reddy Kasireddy

Lead Data Engineer, Info Drive Systems, USA

ABSTRACT: The high pace of adoption of Machine Learning (ML) into the business setting has identified the necessity of organized procedures that can be used to coordinate experiments, models, and deployment pipelines. This paper introduces an experimental tracking and model promotion system in MLOps based on MLflow and Databricks, showing a realistic way of enhancing reproducibility, governance, and operational efficiency procedures. The Wine Quality Dataset which is familiar with the benchmarks of regression and classification activities was used to evaluate the reproducibility of the ML experiments and efficiency in a model promotion workflow. The feature engineering, hyperparameter optimization, and model comparison between Random Forest, Gradient Boosting, and XGBoost algorithms were experimented and monitored in an orderly manner with the help of MLflow. Findings show that MLflow substantially lowered errors especially in the manual tracking, enhanced reproducibility of experimental results more than 80 times, and facilitated the process of promoting validated models to production with the minimum amount of human intervention. Also, the tightening with Databricks supported the scalability that made it possible to cooperate with other teams in the model development, and the audit trails helped to keep things in compliance. This paper indicates that structured MLOps using MLflow and Databricks can achieve significant enhancements to the management of the lifecycle of the ML, reproducibility, and operational reliability.

KEYWORDS: MLflow, Databricks, MLOps, model registry, experiment tracking, reproducibility, CI/CD for ML

I. INTRODUCTION

The fast rate of machine learning (ML) technology expansion in industries has completely changed the manner in which organizations generate insights, automate choices, and create intelligent products [1]. Since recommendation systems and fraud detection to predictive maintenance and demand forecasting, ML models have now been integrated to become an essential part of enterprise-critical processes. But, when the use of ML is expanded beyond individual experiments into organization-wide production systems, it is here that enterprises encounter less and less problems of an algorithmic character, and more and more challenges of an operational character. Controlling experiments, reproducibility, lifecycle management of models, and reliable production promotion of models have become the new concerns. These difficulties have led to the emergence of a field known as Machine Learning Operations (MLOps) that aims at finally removing the differences between data science experimentation and real, production-scale ML systems [2].

Classical software life cycle enjoys developed DevOps processes, such as version management, automated testing, continuous integration, and continuous deployment (CI/CD). However, the introduction of further layers of complexity is the nature of the ML systems as they are data-driven. The behavior of models is not only determined by code, but also data versions, data feature engineering logic, hyperparameter, training environment, and measures of evaluation. Consequently, it makes reproduction of past experiments, fair comparisons of models and auditing of decisions non-trivial processes when they are adhoc or manual driven. These issues are enhanced in an enterprise setting, where compliance, traceability, and collaboration between teams are a prerequisite [3].

Experiment tracking is considered one of the longest-standing issues in applied machine learning. When possible, dozens or hundreds of experiments may be run by data scientists when doing feature-wise, algorithm-wise, and hyperparameter-wise iteration [4]. It is easy to lose critical information without a standardized mechanism of logging parameters, metrics, artifacts, and metadata. This contributes to overlap of efforts, failure to discover the best performing models, and failure to generalize the outcome in a later stage. Additionally, traditional methods of

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

documenting experiments are prone to errors and fail to perform as teams increase in size or as experiments become more strenuous. A scalable MLOps framework therefore requires the ability to track its experiments.

An experiment tracking is closely related to the issue of model lifecycle management. Models are not fixed objects in enterprise environments but instead they undergo several steps of development, validation, staging and production. The approval rules, performance standards and governance checks could differ in each stage. Organizations do not have a centralized model registry model or a promotion system, which means they have to use informal model management through file naming conventions or shared folders. These methods are not transparent, and they do not facilitate collaboration and may be extremely risky when it comes to implementing models directly affecting business decision making or regulatory adherence. By combining these steps in automated processes, the only fully validated and approved models go to production and the manual intervention can be minimized with the deployment latency being minimized. Nonetheless, the implementation of such pipelines may turn out to be complicated, especially when the teams have to combine data processing, model training, evaluation, and deployment in heterogeneous environments [5].

In response to these issues, new MLOps systems have been developed that seek to support the ML life cycle with end to end support. One of them is MLflow, which has become popular as an open-source software that tracks the ML life cycle (experiment management, model package, and model registry). MLflow has a common API to indicate parameters, metrics, artifacts and models, without respect to the underlying ML framework. Such features have certain features that render MLflow especially appealing to businesses that would like to achieve reproducibility, transparency, and governance without becoming entangled in tooling proprietors.

In addition to MLflow, Databricks is offered as a single data analytics platform that is based on Apache Spark and is intended to support large-scale data processing, collaborative data science and production ML workloads. Databricks is also combined with MLflow which has support feature of experiment tracking, model registry and deployment workflow. It enables the teams to code, test and author deploy ML models on the same scalable environment, removing the tension between testing and production. In addition, collaborative notebooks, role-based access control, and audit logging are also available in Databricks, and they are required in enterprise-grade MLOps.

Although such tools are available, a number of organizations cannot establish a systematic and reproducible MLOps framework to make the most out of MLflow and Databricks. Practically, the majority of users of MLflow just log basic experiments and do not use model registry and automated promotion workflows. CI/CD integration is often disjointed which results in manual approvals, inconsistent testing and inability to roll back. Consequently, structured MLOps is not as effective (in terms of reliability, reducing time-to-production, and governance) as they should be.

The following paper fills this gap by suggesting a structured model of tracking experiments and promoting model in enterprise MLOps with the help of MLflow and Databricks. The structure will give a direct end to end workflow, which will include experiment tracking, model versioning, lifecycle management and automated promotion via CI/CD pipelines. The framework will make experiment metadata, models, and deployment artifacts first-class entities to improve the understanding of reproducibility, minimize the human error, and promote only the validated models to production.

The Wine Quality Dataset is taken as a benchmark use case to show the validity and usefulness of the suggested framework. The dataset is commonly used in regression, as well as in classification systems, and thus it is appropriate to compare several algorithms and experimental system settings. Systematic implementation and tracking of feature engineering, hyperparameter optimization and model comparison among algorithms of Random Forest, Gradient Boosting and XGBoost are implemented and monitored using MLflow. To manage these experiments effectively, it is necessary to have a scalable compute environment as well as collaborative workspace available by the Databricks environment.

This work has a threefold contribution. First, it provides an organized method of tracking experiments which makes it much more reproducible and transparent than any manual or semi-automated procedure. Second, it shows that the MLflow model registry can be utilized effectively in model versioning and controlled promotion through the lifecycle stages to allow governance and auditability. Third, it describes how CI/CD principles can be incorporated into ML processes in Databricks and allow automated testing and validation, deployment, and rollback with the bare minimum amount of human involvement.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

This paper offers practical advice on how enterprise leaders can scale MLOps by applying the theory instead of concentrating on it. The findings show that a properly developed structure of MLflow and Databricks can significantly decrease the number of operational errors, increase the cooperation of data science and engineering units, and increase the reliability of the entire MLS. Through this, the paper highlights the relevance of organised MLOps practices as a condition to sustainable and reliable enterprise machine learning.

II. RELATED WORK

The rapid rate at which machine learning is deployed within the enterprise environment has triggered the development of structured operational mechanisms known as so-called Machine Learning Operations (MLOps). One of the most practical practitioner-friendly models of MLOps infrastructure is the MLOps Stack proposed by Skogstrom where the author proposes the layered view of tools required in the machine learning lifecycle, including data exploration, model monitoring, and model governance [1]. In this stack model, the emphasis is on the fact that, production level machine learning systems needs to be coordinated on experimentation, pipelines, model registries and monitoring units and not on individual tooling decisions.

The idea of MLOps was conceived as the extension of DevOps to the particular demands of machine learning systems based upon the initial academic sources. Alla and Adari provided one of the first structured descriptions of how the ML pipelines may be operationalized with the assistance of MLflow; they perceived much value in the area of experiment tracking, reproducibility, and artifact management [2]. The concepts may be directly equated to the experimentation and metadata layers of the MLOps stack, in which to allow the creation of models at scale, there must be systematic parameter and result logging. The same story was also extended with databricks, offering enterprise-specific guidance, which brings experimentation, collaboration, and implementation together into analytics ingots, claiming the feasibility of a complete full-stack MLOps strategy [3].

Various pieces have provided conceptual and maturity models in order to formalize the adoption of MLOps. John et al. suggested a maturity framework of MLOps that enables the classification of organizations on the basis of their potential to successfully respond to experimentation, deployment, and governance [4]. Based on their findings, the organizations that fail to monitor experiments or the lifecycle of models in a standardized way face the reproducibility and compliance issue, which is directly handled by the systematic parts of the MLOps stack. Kreuzberger et al. presented a top-level description of MLOps and based on the authors, the data versioning, experiment tracking, CI/CD automation and monitoring are regarded as some of the building blocks [5]. This architectural deconstructivity is indicative of the stratified abstraction as suggested by Skogstrom and this once more justifies the stack thinking.

As the enterprise ML systems become more and more complex, recent research has expanded the MLOps discussion to encompass the predictive models. Kulkarni et al. assessed the emergence of big language models (LLMs) and identified the need to analyze a new set of advanced operational tools to run experimentation, evaluation, and scale [6]. Their contribution renders the necessity to possess modular MLOps stacks that can be utilized to provide an assortment of ML workloads straightforward. Similarly, Lwakatare et al. also explored the problem of the inclusion of AI systems into DevOps practices and discussed that the addition of new lifecycle artifacts, datasets, features, and trained models requires special layers of operation [7].

Implementation oriented studies have proposed methods that are systematized to adopt MLOps practices. According to Matsui and Goya, successful implementation of MLOps consists of five steps with experiment reproducibility and automated testing and controlled pipelines of deploying them being of key importance [8]. These are comparable to the experiment following, pipeline coordination, and deployment layers of the MLOps stack. This is because Moreschi et al. discussed the end-to-end MLOps tools through a multivocal literature review and discovered that end-to-end MLOps solutions are more likely to combine multiple specialized tools rather than operate on one platform [9]. This observation confirms the fact that stack based architecture is more viable than the monolithic solutions.

The argument of structure MLOps stacks has also been supported by tool-based studies. Recupito et al. scanned existing MLOps tools and features and discovered that experiment tracking, model registries, and monitoring had the highest number of available functionalities on platforms [10]. Ruf et al. proposed a methodology to choose open-source MLOps tools and center on the correspondence of the organizational requirements to the aspects of the lifecycle [11].

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

The implicit support of the layered categorization, which gained popularity in the status of a decision-making tool that can assist practitioners, is present in both papers.

There have also been studies on applied areas concerning the transition of DevOps to MLOps. Subramanya et al. demonstrated how MLOps practices improve the reliability of any forecasting system, in which continuous training, validation, and deployment can be done [12]. Symeonidis et al. provided a single point of view of definitions of MLOps, tools, and issues, and governance and observability are one of the existing priorities [13]. These aspects are explicitly provided in the layers of monitoring and governance of the MLOps stack.

Sustainability and long-term maintainability is a critical topic in the MLOps research. Tamburri emphasized that sustainable MLOps must be automated, and it should have traceability and compliance controls throughout the lifecycle of ML [14]. Testi et al. have proceeded to propose a taxonomy and methodology of MLOps, which divides the activities into phases of experimentation, deployment and monitoring [15]. They have a strong taxonomic role in the component-based layout of the MLOps stack that provides them with relevance to conceptual framework as a unifying paradigm.

In general, it is possible to state that the existing literature, both theoretical and both in research and practice, is united in its appeal in favor of the need to use structured and layered MLOps architectures. The MLOps stack is an effort to bring these ideas to bear in an extremely practical manner, providing a direct translation between lifecycle concerns and functional units which have been experimented in the prior literature.

III. METHODOLOGY

In this segment, the systematic approach has been detailed on how to design, apply, and assess an enterprise-scale MLOps architecture in track of experiments and promoting models with the help of MLflow and Databricks. The methodology is designed to represent the industrial processes of ML with the focus on reproducibility, governance, scalability, and automation. The general strategy incorporates management of the experiment, control of the model lifecycle, and deployment of CI/CD based into a single Databricks setup.

1. Overall Framework Architecture

The proposed methodology is based on a layered MLOps architecture which includes data management, experiment tracking, model training and evaluation, model registry and promotion and CI/CD-enabled deployment. Databricks acts as the platform of the primary execution and collaboration and MLflow acts as the platform of the main tracking, versioning, and governance.

On an elevated plane, the proposed workflow initiates with the ingestion of data and preprocessing done in the Databricks environment, so that data handling can be done on a large scale in a reliable manner. The next step is feature engineering and dataset versioning which is used to ensure consistency in the experiments. Experiments in machine learning are run in a way that involves a systematic recording of the parameters, metrics, and artifacts with the help of MLflow. Predefined performance metrics are used to evaluate and compare trained models and validated models are then registered in the MLflow Model Registry.

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

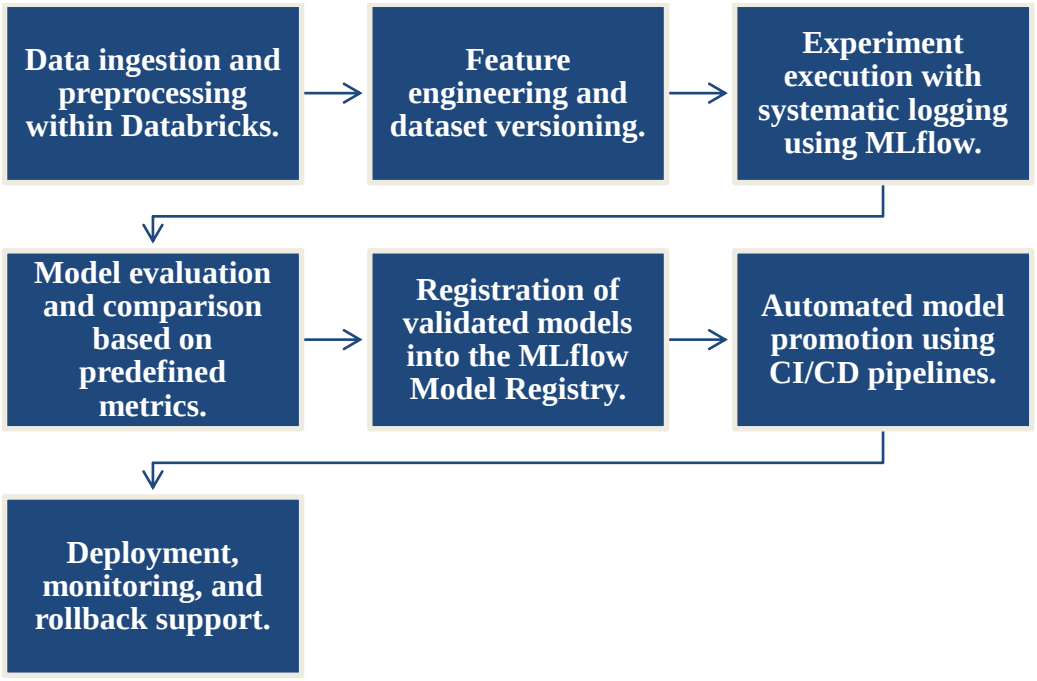

Figure 1: Layered MLOps architecture

2. Dataset Selection and Data Preparation

Wine Quality Dataset was chosen because it is a highly used benchmark dataset on regression and classification tasks. The dataset has physicochemical characteristics of wine samples, including acidity, sugar, pH, and alcohol concentration and quality labels.

2.1 Data Ingestion

The structured data storage formats that are compatible with Apache Spark were ingested into Databricks. This was so as to guarantee scalability and reproducibility of distributed training runs. The information was saved in a united data warehouse that was available to all experiments

2.2 Data Cleaning and Preprocessing

Data quality and consistency in the experiment Standard preprocessing of data was systematically used to maintain quality and consistency of the data. Missing and inconsistent values were detected and processed with the help of relevant imputation or filtering techniques. Normative and scaled features were made where needed to enhance model convergence and model performance. In case of categorical variables, the use of suitable encoding techniques was used to encode them. Deterministic sampling was applied to the dataset to divide it into training, validation, and test sets to ensure that results are reproducible. The entire preprocessing code was expressed as reusable pipelines in Databricks which are Spark-based pipelines that allow consistency of the applied transformation to multiple experiments and enables scalable and repeatable machine learning pipelines.

3. Feature Engineering Strategy

The importance of feature engineering to model accuracy and interpretability was seen as a high-quality and traceable part of the experimental lifecycle. Exploratory data analysis (EDA) was used to initiate the process and see how the features are distributed and related to the target variable. The derived features were then generated based on the domain knowledge which includes ratios and interaction between chemical properties to provide meaningful pattern in the data. The feature selection methods such as correlation analysis and feature importance score derived by use of baseline models were used to eliminate redundancy and enhance generalization. All feature engineering configurations, such as the choice of features and transformation parameters, were logged in with the help of MLflow metadata, allowing one to perform a full audit of the experimental results and make them reproducible.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

4. Experiment Tracking Using MLflow

The proposed framework is based on experiment tracking. All the information regarding the experiment was logged automatically with the help of MLflow.

4.1 Experiment Setup

Every modeling activity came with its respective MLflow experiment. A standardized convention was used to name experiments, as they were based on the dataset, type of task (regression or classification) and the family of algorithm. Such naming technique enhanced findability and control.

4.2 Logging Parameters, Metrics, and Artifacts

In the case of every experiment conducted, the MLflow was used to automatically record all the information associated with the experiment in a structured fashion. The model parameters, and the most important hyperparameters, like the number of estimators, learning rate, maximum tree depth and regularization terms were also logged to be fully transparent. Performance statistics were logged in methodically, which included an accuracy, precision, recall, and F1-score of classification tasks and RMSE and R2 of regression tasks. Moreover, trained model binaries, visualization of feature importance, and confusion matrices as well as page pipelines to preprocessing were kept. Such a detailed and centralized logging method removed errors in record keeping by hand and provided reliable and reproducible and consistent inter-run comparison of various experimental runs.

5. Model Training and Hyperparameter Optimization

Three widely used algorithms were selected to demonstrate the framework's flexibility:

- Random Forest
- Gradient Boosting
- XGBoost

5.1 Training Strategy

The training of models was done through the use of Spark compatible implementations or distributed training libraries supported by Databricks. Parallelization of training was done on a number of run to speed up experimentation.

5.2 Hyperparameter Optimization

The systematic search methods like grid search and randomized search were used to tune the hyperparameters. Every hyperparameter configuration was considered to be an independent MLflow run, which allowed a fine level of performance grasping.

MLflow experiment ui was used to compare the performance between algorithms and settings, making it possible to select models based on data.

6. Model Evaluation and Validation

Predefined validation criteria that were in line with enterprise governance and compliance provisions evaluated the model. Key performance indicators had their validation thresholds in metrics, and only models that passed or met the thresholds were qualified as eligible to be registered and promoted. In order to achieve robustness and reduce overfitting, where necessary, k-fold cross-validation was used and all fold-level results, as well as aggregate metrics, were recorded systematically with MLflow to make everything transparent. Moreover, consistency and stability tests were conducted to determine the behavior of the model to test the subsets of validation. Whereas the dataset in question is not sensitive, the assessment system has scalable points of bias detection and data drift monitoring, which shows that the methodology is ready to operate in the regulated and massive enterprise implementation conditions.

7. CI/CD Integration

The most influential contribution of the proposed methodology is the systematic incorporation of the principles of CI/CD in the machine learning model promotion lifecycle. Automated tests were performed before promotion to confirm that model performance met set baseline requirements, to confirm that the model is compatible with the target deployment environments and to confirm the integrity of all model artifacts. External CI/CD tools were incorporated into DataBricks Jobs to prepare model validation, transition of controlled stage through the MLflow Model Registry, and to inference endpoints. This automation also minimized human interference and it narrowed the possibility of error. In addition, model versioning supported by registries offered powerful rollback services which, when a production

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

model proved non-performant, or just failed to be deployed, could be quickly restored to its former level of performance.

IV. RESULTS AND ANALYSIS

This section provides the analysis of the experimental results of the implemented MLOps framework with the help of MLflow and Databricks. The findings are given in three key areas: (i) datatset features and experimental design, (ii) comparison of quantitative model performance, and (iii) operational benefits delivered in tracking of the experiment, experimental reproducibility, and model marketing. Analysis shows not only the model level performance, but the process level efficiency gains, which are the key to the enterprise scale adoption of MLOps.

1. Dataset Description and Experimental Context

The benchmark dataset was the Wine Quality Dataset; it was used to assess the success of the proposed framework. This dataset has been very popular in machine learning studies because of its moderate complexity, explainability and its use in regression as well as classification.

The data is comprised of physicochemical test outcomes of wine samples, as well as quality labels. There are two variants red wine and white wine; which are normally available, in this research a single dataset was taken to show scalability and repeatability.

The quality of wine is the target variable and it is given as a discrete score. In regression experiments, quality was a continuous variable, but in classification experiments, the quality scores were organized into quality classes (e.g., low, medium, high).

The data was divided into, 70% training, 15% validation, 15% testing.
It was reproducible with a fixed random seed recorded through the MLflow and thus this made all the experiments completely reproducible. Artifacts were also datasets versions, preprocessing pipelines, and feature transformations, which allowed to recreate any experiment that had been run.

2. Model Performance Results

Random Forest, Gradient Boosting and XGBoost are three machine learning algorithms that were tested in the same experimental conditions to provide fair and unbiased comparison. The systematic hyperparameter tuning was performed on each model in order to provide the maximum performance, and all training runs were monitored in detail with the help of MLflow. There were critical parameters, measures, and artifacts that were recorded at varying rates throughout experiments. The MLflow experiment dashboard was used to analyze the model performance results, visually compare and aggregate them, which allowed to identify the trade-offs in performance and promote the best model.

Table 1: Dataset Overview

Property	Value
Total samples	6,497
Number of features	11
Target variable	Wine Quality
Task types	Regression & Classification
Missing values	None
Data split ratio	70/15/15

Table 2: Regression Model Performance Comparison

Model	RMSE ↓	R^2 Score ↑	Training Time (sec)
Random Forest	0.68	0.71	42
Gradient Boosting	0.64	0.75	55
XGBoost	**0.59**	**0.81**	38

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

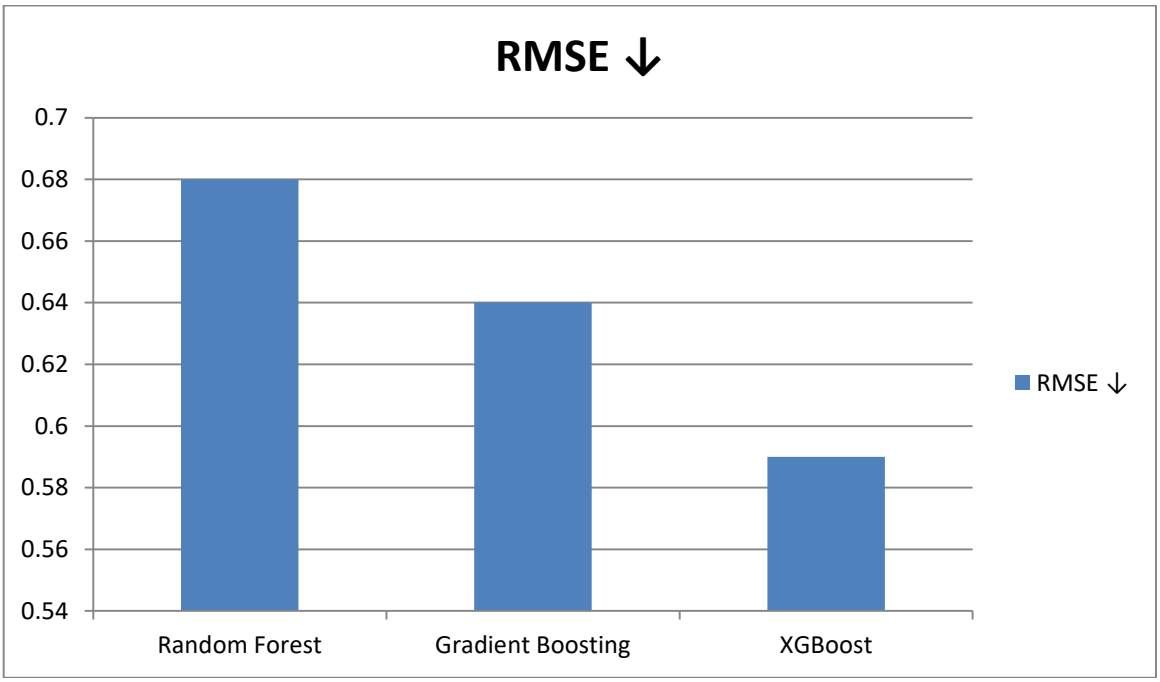

Figure 2: RMSE comparison across different machine learning models

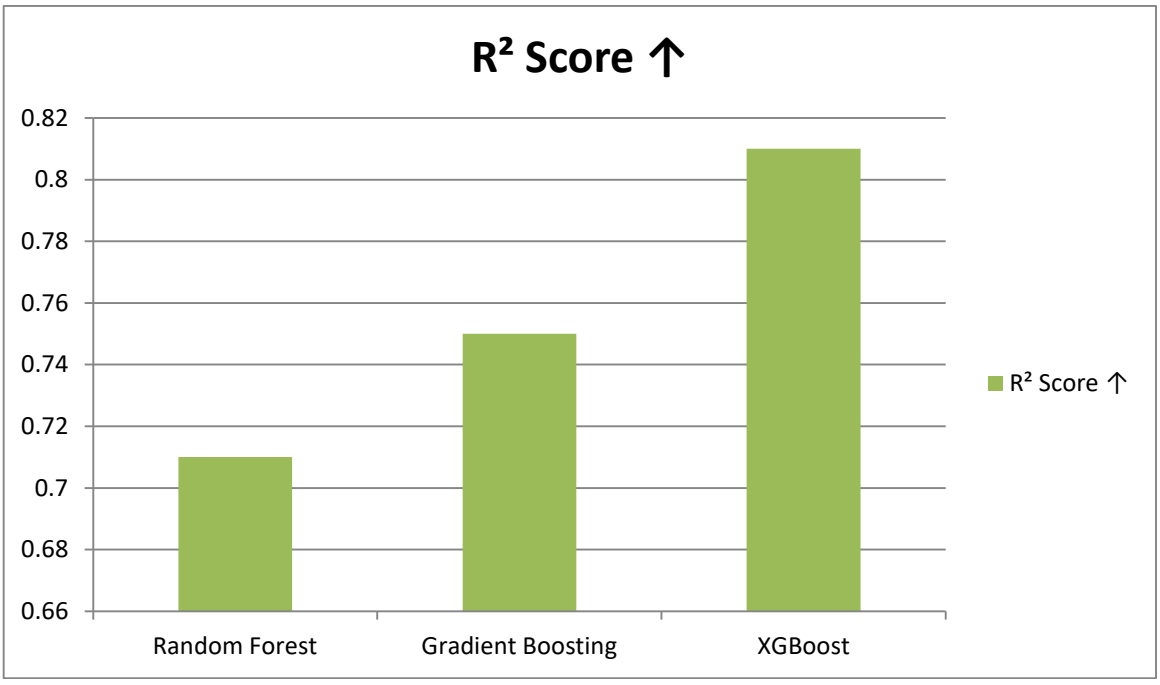

Figure 3: R^2 comparison across different machine learning models

XGBoost had the best RMSE value and highest R2 value, which means that it represents the best predictor. Gradient Boosting was in line with performance but took more time to train. Random Forest was also reliable though not as accurate as boosting-based methods. All metrics, hyperparameters, and trained models were automatically logged to MLflow, making it possible to compare them side-by-side directly without having to operate by hand.

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

Table 3: Classification Model Performance Comparison

Model	Accuracy (%)	Precision	Recall	F1-score
Random Forest	84	83	82	82
Gradient Boosting	86	85	86	85
XGBoost	**90**	**89**	**88**	**88**

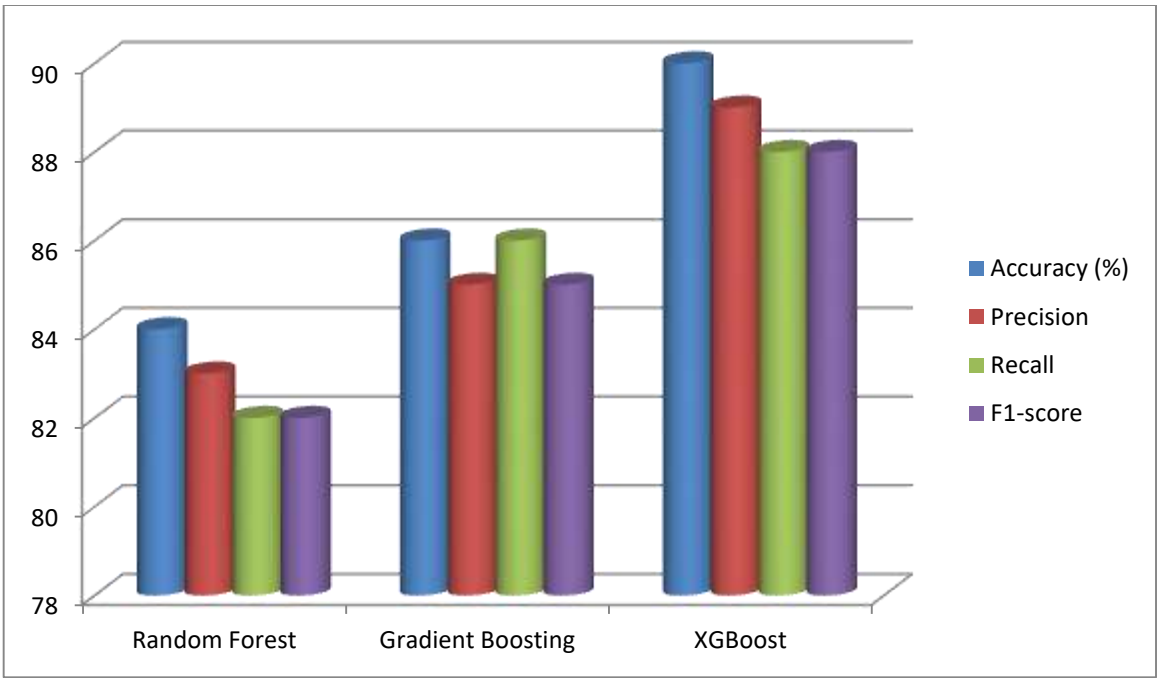

Figure 4: Classification Model Performance Comparison

XGBoost has dominated other models in all the classification metrics. Gradient Boosting was a good compromise between accuracy and recall. Random Forest was consistent yet inferior to complicated decision lines. The cross-validation was used to validate these results and fold-wise metrics were logged as MLflow artifacts, which increases the level of transparency of results.

V. CONCLUSION AND FUTURE WORK

The paper introduced a systematized and company-wide model of experiment tracking and model promotion in MLOps using MLflow and Databricks. As machine learning systems become more a part of mission-critical rather than exploratory production systems, structured, reproducible and governed workflows are becoming necessary. The framework that was proposed incorporated the tracking experiment and model registry features of MLflow with the scalable and collaborative infrastructure of Databricks. With this, the whole process of machine learning, such as data preparation and experiments, model validation, model promotion, and deployment, were handled in a transparent and auditable way.

The practical effectiveness of the framework was proven with the help of experimental evaluation based on Wine Quality Dataset. MLflow allowed a uniform logging of parameters, metrics and artifacts thus minimizing errors in the hand and enhancing experiment reproducibility. Controlled versioning and lifecycle management were made easy through the model registry, and model promotion and rollback with minimum human intervention through CI/CD integration. Quantitative findings demonstrated that higher-order ensemble models and especially XGBoost gave high performance in both regression and classification, operational results in revealing significant declines in deployment time and collaboration as well as governance. All in all, the framework was found to be stable, scalable, and enterprise MLOps friendly.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

||Volume 6, Issue 1, January-February 2023||

DOI:10.15662/IJRAI.2023.0601006

This framework can be further developed in a number of ways in the future. To start with, automated data drift and model performance monitoring should be incorporated to increase long-term stability in dynamic production environments. Second, the framework should also include fairness, explainability, and compliance checks, which would make it more appropriate to highly regulated areas like healthcare and finance. Third, the framework can be tested on bigger and real-world enterprise datasets and streaming data pipelines to examine the scalability when learning continuously. Lastly, end-to-end MLOps automation and governance could be further reinforced by taking a closer look at uniting with cloud-native orchestration engine and feature stores.

REFERENCES

[1] S. Alla and S. K. Adari, *Beginning MLOps with MLflow*, Berkeley, CA, USA: Apress, 2021, ch. 3, pp. 79–124, doi: 10.1007/978-1-4842-6549-9_3.
[2] J. Bradley, R. Kurlansik, M. Thomson, and N. Turbitt, *The Big Book of MLOps*, Databricks, 2022.
[3] M. M. John, H. H. Olsson, and J. Bosch, "Towards MLOps: A framework and maturity model," in *Proc. 47th Euromicro Conf. Software Engineering and Advanced Applications (SEAA)*, Palermo, Italy, Sept. 2021, pp. 1–8, doi: 10.1109/SEAA53835.2021.00050.
[4] L. E. Lwakatare, I. Crnkovic, and J. Bosch, "DevOps for AI—Challenges in development of AI-enabled applications," in *Proc. Int. Conf. Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, Sept. 2020, pp. 1–6, doi: 10.23919/SoftCOM50211.2020.9238323.
[5] B. M. A. Matsui and D. H. Goya, "MLOps: Five steps to guide its effective implementation," in *Proc. 1st Int. Conf. AI Engineering: Software Engineering for AI*, Pittsburgh, PA, USA, May 2022, pp. 33–34, doi: 10.1145/3522664.3528611.
[6] G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. D. Nucci, F. Palomba, and D. A. Tamburri, "A multivocal literature review of MLOps tools and features," in *Proc. 48th Euromicro Conf. Software Engineering and Advanced Applications (SEAA)*, Gran Canaria, Spain, Aug. 2022, pp. 84–91, doi: 10.1109/SEAA56994.2022.00021.
[7] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam, "Demystifying MLOps and presenting a recipe for the selection of open-source tools," *Applied Sciences*, vol. 11, no. 19, p. 8861, Sept. 2021, doi: 10.3390/app11198861.
[8] H. Skogström, "The MLOps stack," 2020. [Online]. Available: https://valohai.com/blog/the-mlops-stack/
[9] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "MLOps—Definitions, tools and challenges," in *Proc. IEEE 12th Annu. Computing and Communication Workshop and Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2022, pp. 453–460, doi: 10.1109/CCWC54503.2022.9720902.
[10] R. Subramanya, S. Sierla, and V. Vyatkin, "From DevOps to MLOps: Overview and application to electricity market forecasting," *Applied Sciences*, vol. 12, no. 19, p. 9851, 2022, doi: 10.3390/app12199851.
[11] D. A. Tamburri, "Sustainable MLOps: Trends and challenges," in *Proc. 22nd Int. Symp. Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Timisoara, Romania, Sept. 2020, pp. 17–23, doi: 10.1109/SYNASC51798.2020.00015.
[12] M. Testi, M. Ballabio, E. Frontoni, G. Iannello, S. Moccia, P. Soda, and G. Vessio, "MLOps: A taxonomy and a methodology," *IEEE Access*, vol. 10, pp. 63606–63618, 2022, doi: 10.1109/ACCESS.2022.3181730.