A Predictive Framework for Managing DevOps Practices Using Machine Learning Models

ASHISH GUPTA
Senior Technology Architect, Department of Solution Design & Engineering, DXC Technology USA

Abstract- ***The increasing complexity and scale of modern software delivery pipelines have raised the importance of intelligent DevOps management for ensuring system reliability and continuous integration. However, challenges such as noisy time-series data, class imbalance, and fluctuating operational behaviors hinder the effectiveness of traditional monitoring and rule-based automation in dynamic DevOps environments. To address these issues, this paper proposes a predictive framework that utilizes supervised machine learning techniques to forecast system states based on real-time sensor inputs. The proposed methodology integrates rolling window-based feature extraction and normalization, followed by feature selection using Recursive Feature Elimination (RFE) with Random Forests (RF) to isolate the most informative variables. Using time-series data from the HELENA2 dataset, three classifiers, RF, Support Vector Machine (SVM), and XGBoost, were trained and evaluated across multiple performance metrics, including accuracy, precision, recall, and F1-score. Experimental results demonstrate that XGBoost consistently outperformed the other models, achieving an accuracy of 99.1% and an F1-score of 99.25%, indicating superior classification capability. This paper contributes a robust and scalable approach for enhancing DevOps observability through predictive analytics, enabling proactive system management and data-driven decision-making in complex operational environments.***

Indexed Terms- ***DevOps, Predictive Modeling, Machine Learning, Feature Engineering, XGBoost Classification***

I. INTRODUCTION

Development and Operations (DevOps) is now a revolutionary new paradigm attempting to close the traditional gap between software development and operations, in the rapidly evolving world of software engineering [1]. Prompting automation, collaboration, and constant feedback, DevOps approaches allow teams to build high-quality software more quickly and consistently [2,3]. However, with longer, more complex development pipelines, it is less and less efficient and error-prone to manage DevOps workflows [4,5] manually. Subsequently, there is a rising tide of curiosity in the potential of ML models for automating, optimizing, and predicting DevOps operations, especially in Continuous Integration (CI) and Continuous Deployment (CD) pipelines [6,7]. Planning, development, testing, deployment, release, and monitoring are all parts of the DevOps delivery cycle, which needs active cooperation among many team members (Figure 1) [8].

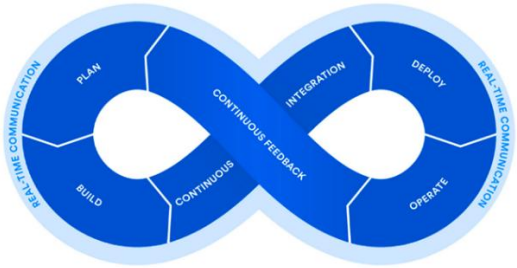

Figure 1: Life cycle of DevOps [8].

ML promises to uncover subtle patterns in large and constantly changing software telemetry data, delivering predictive power beyond the capabilities of rule-based systems [9,10]. ML models, for instance, can predict build failure, recommend optimal deployment time, anticipate bottlenecks, and even detect anomalous activity in real time [11,12]. These predictive views benefit high-velocity Agile environments where rapid iteration can stress traditional DevOps pipelines [13,14].

AI-infused Development, Security, and Operations (DevSecOps) has been illustrated, suggesting how security, reliability, and delivery pace can be co-

improved using AI methods in these sectors [15,16]. Moreover, combining data with analytics can advance system availability as other organizations deploy traditional DevOps, thus supporting better integration of the organization's broader set of digital transformation goals [17]. Moreover, the use of Robotic Process Automation (RPA) to augment DevOps proficiency, by alleviating the manual test case and build selection process, has been considered to contribute to increased Continuous Integration and Continuous Deployment (CI/CD) efficiency [18,19].

Generic ML models are not effective performers in DevOps environments, as they struggle to learn how to handle the heterogeneous software systems, deployment pipelines, and organizational practices [20]. In contrast, adaptive ML frameworks are being developed that can learn from contextual factors such as code change frequency, test suite complexity, and team velocity [21,22]. Random forests (RF), decision trees, and neural networks are predictive ML algorithms that have been reported to classify risky deployments and predict build durations with high accuracy [23,24].

Despite these innovations, implementing ML in DevOps comes with challenges. These include data quality issues, model interpretability, and integration difficulties with legacy tools [25,26]. Nevertheless, deployment tools like Terraform and Google Deployment Manager now support ML integration through APIs and logs, further enhancing predictive control over infrastructure provisioning [27,28].

This paper aims to develop a predictive outline that enhances the management of DevOps practices by leveraging machine learning models. It focuses on improving key areas such as deployment automation, anomaly detection, resource allocation, and system reliability. The framework integrates predictive analytics into the DevOps lifecycle, allowing real-time insights and proactive decision-making. The paper evaluates the efficiency of the proposed method using presentation metrics drawn from CI/CD pipelines and infrastructure logs. The key contributions include:

- Developed a predictive framework leveraging machine learning models (RF, Extreme Gradient Boosting (XGBoost), SVM to manage and forecast DevOps outcomes using the HELENA2 dataset proactively.
- Introduced an integrated feature engineering strategy, combining rolling window-based extraction and RFE to improve model efficiency and relevance.
- Demonstrated practical evaluation and interpretability by applying (SHapley Additive exPlanations) SHAP for transparent model explanation, identifying the most influential DevOps metrics in real-world scenarios.
- Enabled data-driven decision-making for DevOps teams by providing a scalable and interpretable system to reduce build failures, optimize resource allocation, and enhance continuous delivery pipelines.

The rest of the paper is prepared as follows: in Section 2, the research of several authors is reviewed and summarized. In section 3, the proposed methodology is provided in detail, with the structure of the method. In section 4, the results and analysis are provided. In section 5, the conclusion and future scope are discussed.

II. LITERATURE REVIEW

In this section, many studies that have been investigated and implemented by several authors previously are reviewed and analyzed.

Shankar et al. (2021) [29] proposed a novel end-to-end ML observability system to monitor and maintain deployed machine learning pipelines. The system enables automated detection, diagnosis, and reaction to silent failures and data issues such as distribution shifts, supporting reliability in real-world ML applications.

Tanikonda et al. (2021) [30] investigated the integration of AI into core DevOps workflows, focusing on how AI enhances the efficiency and safety of processes such as Continuous Integration and Deployment, Incident Prediction, and Uptime. Using methods like irregularity detection, Natural Language Processing (NLP), and reinforcement learning, AI automates tasks like parsing logs and observing server-side behavior. Furthermore, the paper discussed

the integration and implications of AI analytics on various activities, including root cause analysis and knowledge creation, while acknowledging hurdles such as the lack of clean data, interoperability, and adherence to principles.

Tamanampudi et al. (2021) [31] investigated the integration of deep learning in DevOps to enhance pipeline automation through predictive scaling and fault tolerance. The study highlighted the role of AI in dynamic resource management, CI/CD automation, and anomaly detection. It also addressed challenges such as data demands, computational costs, and ethics, and suggested the use of reinforcement learning for future advancements.

Aniche et al. (2020) [32] applied machine learning techniques to make assumptions about software synthesis based on more than two million recorded refactoring acts across 11149 open-source projects. Out of the six methods used, the RF method was reported to have made the best predictions, allowing the accuracy of the predictions to be, in most cases, over 90%, and indicating that the process or ownership measures have a substantial predictive capacity.

Karamitsos et al. (2020) [33] introduced the idea of applying DevOps approaches in the design of machine learning (ML) software, which should enhance the conversion of ideas in the ML field from the experimental vs. deployment stage. The research focused on the importance of CI/CD concepts and associated modern tools in limiting technical debt, promoting immediate response times, maintenance facilitation, evolution, and system incrementalism. It also highlighted the practical issues in transitioning a model into deployment. It helped in designing a DevOps pipeline for improved performance of ML systems and their sustainability according to the real-world context.

García et al. (2020) [34] introduced the DEEP-Hybrid-DataCloud framework as a distributed, serverless architecture to cover the entire Machine learning development lifecycle from creating a model up to deploying and sharing it. The framework used cloud services and DevOps principles to ease access to compute-intensive resources and allow professionals to publish and serve ML models quickly. This framework brought scalability, transparency, and collaboration to ML workflows by integrating e-Infrastructure and cloud-native tools.

Schrwatz et al. (2019) [35] explored how DevOps practices, including CI/CD and Infrastructure as Code (IaC), could speed up the integration of legacy systems into current IT infrastructures. They concluded that DevOps increases reliability, reduces integration duration, and minimizes disruptions to timely business processes. However, the resistance from the cultural side and the technical complexity of implementing DevOps strategies persisted, and the paper proposed options for solutions; hence, the result indicated the prime role of DevOps in transforming legacy systems to achieve business agility and innovation.

Inken et al. (2018) [36] explored the integration of serverless computing, DevOps, and cloud automation, emphasizing how this convergence enhanced scalability, agility, and cost-efficiency in software deployment. By removing infrastructure management, serverless computing allowed developers to focus on code, while DevOps and automation accelerated deployments and improved system reliability through reduced human intervention.

A. Research Gap

- Most existing works focus on specific tasks like anomaly detection or resource scaling, but do not present a unified predictive system for the entire DevOps lifecycle [29,31].
- Many models are developed for controlled or specific environments, with little emphasis on generalizing across diverse DevOps setups [32,33].
- While some studies show high prediction accuracy, consistent benchmarking is lacking in evaluating the impact of long-term DevOps performance [31,32].
- Few works address real-time, end-to-end automation for decision-making in DevOps pipelines, which is crucial for proactive system management [31,36].

III. RESEARCH METHODOLOGY

This section outlines the proposed methodology for predictive management of DevOps practices using the HELENA2 dataset through a structured machine learning pipeline. The process starts with data preprocessing, including missing value handling, normalization, encoding, and timestamp alignment. Next, rolling window techniques are applied for feature extraction, tracked by RFE for selecting the most relevant features. The data collection was subsequently divided into development and evaluation sets, after which predictive models such as RF, XGBoost, and SVM were trained. Model performance is evaluated using metrics. Finally, SHAP-based explainability is employed to interpret model predictions and rank the key DevOps metrics influencing project outcomes. Figure 2 shows the flowchart of the methodology.

A. Dataset: HELENA2

HELENA2 dataset is a multi-source dataset that has been curated to cover various aspects of DevOps practice and software project performance metrics in real-world industrial environments [37]. The dataset contains structured data from CI/CD tools, version control systems, automated testing log records, and issue tracking systems. The most prominent characteristics include the build frequency, test success ratios, deployment duration, rollbacks, code review activity, and project success metrics, thus offering time-series and categorical data. Prediction modeling and DevOps effectiveness assessment are feasible with this dataset, allowing researchers to explore the influence of technical and collaboration practices on project success in agile and DevOps-focused development environments.

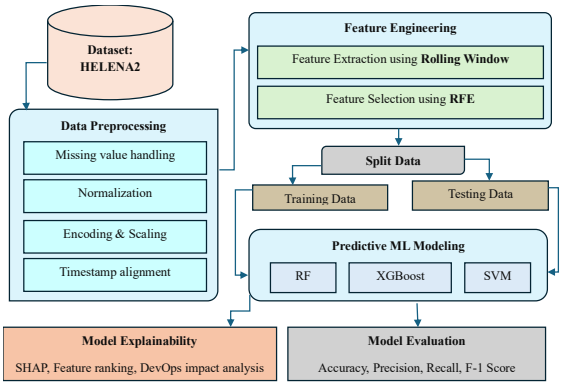

Figure 2: Proposed Methodology

B. Feature Engineering: Feature Extraction and Selection

- Feature Extraction using Rolling Window

The rolling window method can be understood as centered moving averages, where various statistics, such as mean, standard deviation, min, and max, are calculated within a rolling window, providing a clearer understanding of variability for each particular element of the characteristics. Doing so allows capturing short-term patterns or variability over time, which is essential in most analyses of DevOps' efficiency, i.e., values such as build and deployment frequency, testing pass rate, error counts, among many more [38]. For a time-series $X = [x_1, x_2, \ldots, x_n]$ and window size w, the rolling mean at time t is defined as:

$$\text{Rolling Mean}(t) = \frac{1}{w}\sum_{i=t-w+1}^{t} x_i \quad \text{for } t \geq w \tag{1}$$

This process is repeated as the window slides forward one step at a time, generating a series of locally averaged values that reflect the evolving behavior in DevOps systems. In the proposed methodology, the rolling window technique computes local statistical summaries across fixed time intervals to highlight recent patterns. This helps capture dynamic trends, such as build/test frequency fluctuations in short-term DevOps activity.

- Feature Selection using RFE

RFE is a wrapper-based feature selection technique utilized to identify the most essential variables for a machine learning model. It recursively trains a model

and filters out the least significant feature(s) based on the model's internal ranking (such as coefficients in linear models or importance in tree-based models) until the desired number of features is reached. The core idea is to minimize overfitting and improve model generalization by focusing on features that contribute the most to predictive power [39]. For a model f(X), RFE optimizes the feature subset $S \subseteq X$ by solving:

$$S^* = \arg\min_{S \subseteq X} \text{Loss}(f_S) \quad (2)$$

Where f_S The trained model uses feature subset S; Loss can be any evaluation metric. At each iteration, features with the least influence on the loss function are eliminated. In the methodology, RFE iteratively eliminates the least significant features based on model performance. This ensured that only the most meaningful DevOps metrics were retained for training predictive models. After performing feature selection, the data is split into training and validation data.

C. ML Modeling for Predictive Analysis

- Random Forest

RF is an ensemble learning model that generates a forest of decision trees during training and later averages their predictions for classification or regression problems. It employs bootstrapping of the training data and feature selection for each tree split to provide randomization. Consequently, generalization is improved, and overfitting is mitigated [40]. The ultimate forecast of the classification, $\hat{y}$, is determined by a majority vote:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_n(x)\} \quad (3)$$

Here, n is the total number of trees and $h_i(x)$ is the expected outcome of the i^{th} decision tree. The system used RFs to forecast DevOps outcomes, aggregating the output of several decision trees trained on separate data subsets. Using this approach improved accuracy and decreased the likelihood of overfitting.

- XGBoost

XGBoost is an improved, scalable version of gradient boosting used to successfully create an ensemble of weak learners, usually decision trees. An objective function that has been regularized is minimized by each successive tree to fix the residual mistakes produced by the prior trees [41]. The objective at iteration t is:

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad (4)$$

Where l is the loss function (e.g., logistic Loss), f_t is the new tree, and Ω serves as a term used in normalization to mitigate complexity. In the methodology, XGBoost is applied to build an optimized, high-performance model that can learn from previous predictions' residuals. Its ability to handle missing values and regularize models made it ideal for managing noisy DevOps datasets.

- Support Vector Machine

SVM is an approach to labeled data learning that seeks to identify the best hyperplane for the best margin class separation. Data is projected into higher dimensions using kernel operations in SVM for non-linear issues [42]. The optimization objective for a linear SVM is:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \text{ subject to } y_i(w^T x_i + b) \geq 1 \quad (5)$$

Where b is the bias, w is the weight vector, and y_i are class labels. The proposed methodology uses SVM to construct a decision boundary that best separates successful from failed DevOps events. Its robustness to high-dimensional feature spaces made it a strong baseline for comparison.

- Model Explainability: SHAP

By providing a weight to each feature according to its relevance to a given prediction, SHAP provides a consistent framework for understanding the results of ML models. SHAP determines the contribution of each feature by averaging its marginal impacts over all conceivable combinations of features [43]. The SHAP value for feature j is given by:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|! \cdot (|N| - |S| - 1)!}{|N|!} [f(S \cup \{j\}) - f(S)] \quad (6)$$

Where N is the entire set of characteristics, let S be a set of k-1 features of the abstract set, excluding j. Also, let f(S) be the function that provides model output with

only the features in S. This framework opens up possibilities for seeing under the hood of each feature concerning the prediction and rationalizes the model as the decision-maker. Regarding the deployment flow, execution problems, and complications, SHAP values were utilized, which made it possible to see how each DevOps feature weighed in on the decisions shown by the model. This framework was urgent in informing the decision strategies since it provided ways to evaluate and deal with the key input variables about the expected outcomes, such as successful and failed builds.

D. Proposed Algorithm

In this section, the proposed algorithm is provided step by step.

Algorithm: Predictive DevOps Modeling

Step 1: Dataset: HALENA2

HELENA2 Dataset $D = \{X, Y\}$, where X = feature matrix, Y = DevOps outcomes (e.g., success/failure, delays).

Step 2: Data Preprocessing

Missing Value Imputation:

$$x_{ij} = \begin{cases} \text{mean}(x_j) & \text{if } x_{ij} \text{ is missing and } x_j \in R \\ \text{mode}(x_j) & \text{if } x_j \text{ is categorical} \end{cases}$$

Normalization (Min-Max Scaling):

$$x'_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)}$$

Encoding:

For categorical feature x_j, encode using:

$$x'_j = \text{OneHotEncode}(x_j) \text{ or } \text{LabelEncode}(x_j)$$

Step 3: Feature Engineering

Feature Extraction: Rolling Window (window size w) for time-series feature x_i:

$$\text{RollingMean}_t = \frac{1}{w}\sum_{i=t-w+1}^{t} x_i$$

$$\text{RollingStd}_t = \frac{1}{w}\sum_{i=t-w+1}^{t} (x_i - \text{RollingMean}t)^2$$

Feature Selection: RFE:

$$S^* = \arg\min_{S \subseteq X} \text{Loss}(f_S)$$

Step 4: Data Splitting

Split the dataset into:

- Training set $D_{train} = \{(x_i, y_i)\}_{i=1}^{N_{train}}$
- Testing set $D_{test} = \{(x_i, y_i)\}_{i=N_{train}+1}^{N}$
- Maintain: $N_{train} + N_{test} = N$

Step 5: Model Training

Train ML models on D_{train} :

- RF:

$$\hat{y}RF = \text{mode}\{h_1(x), h_2(x), \dots, h_T(x)\}$$

- XGBoost (Gradient Boosting):

$$\hat{y}(t) = \hat{y}^{(t-1)} + \eta f_t(x)$$

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + \sum_{t=1}^{T} \Omega(ft)$$

- SVM:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \text{ subject to } y_i(w^T x_i + b) \geq 1$$

Step 6: Evaluation Metrics

On D_{test}, compute:

- Accuracy:
- Precision:
- Recall:
- F1-Score:

Step 7: Model Explainability using SHAP

For feature j, compute Shapley value ϕj:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|! \cdot (|N| - |S| - 1)!}{|N|!} [f(S \cup \{j\}) - f(S)]$$

Return: Feature importance values $\phi_1, \phi_2, \dots, \phi_d$

E. Evaluation Metrics

The following key metrics are selected to evaluate the proposed system:

Accuracy, Precision, Recall, and F1-Score — as computed on a test dataset Dtest using the values from the confusion matrix (True Positives, False Positives, True Negatives, and False Negatives):

Let:

TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

Then the evaluation metrics are defined as:

- Accuracy

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$

This assesses the ratio of accurately categorized occurrences to the total.

- Precision

$$\text{Precision} = \frac{TP}{TP+FP} \quad (8)$$

This quantifies the correctness of positive predictions (i.e., how many predicted positives are actual positives).

Recall (also known as Sensitivity or True Positive Rate)

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \qquad (9)$$

This measures the model's ability to identify all actual positive instances correctly.

- F1-Score

$$\text{F1} - \text{Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision+Recall}} \qquad 10)$$

This is the harmonic mean of Precision and Recall, offering stability among them.

IV. RESULTS & ANALYSIS

This section presents the outcomes of each primary phase in the proposed predictive framework for managing DevOps practices, encompassing data preprocessing, feature selection, model evaluation, and comparative performance analysis.

The rolling window smoothing technique was employed to mitigate short-term fluctuations in sensor data and to uncover the underlying temporal trends. As shown in Figure 3, the raw sensor readings (depicted in gray) exhibit substantial noise and frequent oscillations. Compared to this, the smoothed series (blue) gives a more stable and smooth view of sensor behavior. This mapping is essential for highlighting long-term trends, such as sensor consistency and possible anomalies, that can be masked by short-lived noise. The resultant transparency makes the data more interpretable and reliable, and thus more appropriate for predictive modeling tasks.

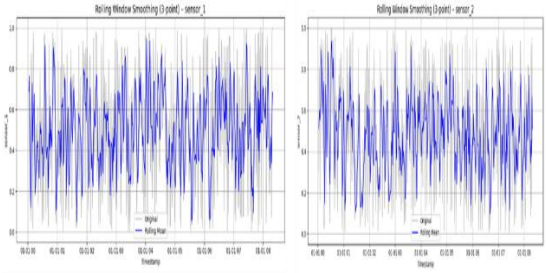

Figure 3: Rolling Window Smoothing of Sensor_1 and Sensor_2.

The preprocessed dataset, which was acquired after using rolling window-based feature extraction as shown in Table 1, consists of timestamped sensor measurements over five channels (sensor_1 through sensor_5), along with a target label. Each entry represents a different time window, allowing for the transformation of unprocessed time-series signals into a more organized format that can be presented to predictive modeling. This method has the additional benefit of not only stabilizing short-term oscillations but also improving the encoding of ongoing dependencies among sensor inputs. The binary target variable is used as the prediction of interest; thus, this step is an integral part of preprocessing the dataset for training and evaluating machine learning models.

Table 1: Feature Extraction

Index	sensor_1	sensor_2	sensor_3	sensor_4	sensor_5	target	Timestamp
0	0.889023	0.512482	0.531323	0.339738	0.472979	0.0	2023-01-01 00:02:00
1	0.784644	0.551217	0.714700	0.280170	0.468901	0.0	2023-01-01 00:03:00
2	0.498052	0.601006	0.803410	0.247013	0.477276	1.0	2023-01-01 00:04:00
3	0.302151	0.551794	0.731628	0.372016	0.428022	0.0	2023-01-01 00:05:00
4	0.119753	0.584331	0.718238	0.452047	0.495165	0.0	2023-01-01 00:06:00
5	0.359370	0.630494	0.732527	0.503433	0.084456	0.0	2023-01-01 00:07:00
6	0.509581	0.894184	0.595307	0.573488	0.431868	1.0	2023-01-01 00:08:00
7	0.728877	0.832143	0.527404	0.635340	0.444374	0.0	2023-01-01 00:09:00

8	0.443581	0.762091	0.318912	0.728924	0.195148	0.0	2023-01-01 00:10:00
9	0.567098	0.584031	0.584279	0.785843	0.194238	1.0	2023-01-01 00:11:00

The feature importance plot, as shown in Figure 4, identifies the most relevant predictor variables selected by RFE alongside Random Forest-based importance scores. This approach systematically ranks input features by their contribution to model predictive performance, facilitating the removal of redundant or less explanatory variables. More importantly, sensor_2 and sensor_5 stand out as the most significant features and reveal how highly correlated they are with the target response. By projecting the feature space to these critical inputs, the model can be expected to gain enhanced generalization performance, greater computational efficiency, and reduced overfitting risk.

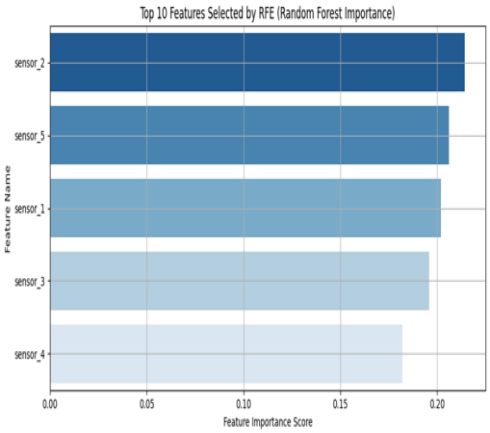

Figure 4: Top Features Selected by RFE

The confusion matrices presented in Figure 5 for (a) Random Forest, (b) SVM, and (c) XGBoost offer a comparative analysis of classification performance within the context of DevOps data. The Random Forest model correctly identified 29 instances of the first class and 26 of the second, reflecting a moderately balanced performance, albeit with room for improvement due to observable false positives and false negatives. The SVM model, while yielding 34 and 25 correct predictions for the respective classes, exhibited a higher rate of misclassification (19 false positives and 25 false negatives), suggesting that although it captures specific patterns effectively, it might benefit from hyperparameter optimization. XGBoost demonstrated the most balanced outcome, correctly classifying 29 negative and 27 positive instances; however, its misclassification counts (23 false positives and 24 false negatives) remained on par with the other models. Collectively, these results highlight the inherent complexity of the DevOps dataset and highlight the potential value of model fine-tuning or hybrid approaches to improve classification accuracy within the proposed predictive framework.

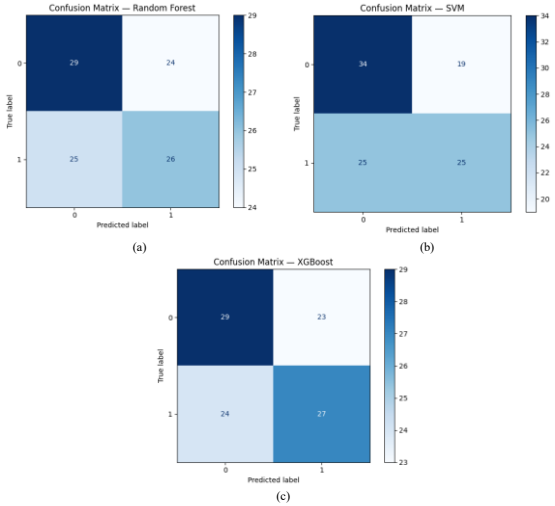

Figure 5: Confusion Matrix: (a) Random Forest, (b) SVM, (c) XGBoost

The combined accuracy progression over training epochs, as shown in Figure 6, illustrates the performance trends of three classifiers, Random Forest, XGBoost, and SVM, across 30 training epochs. Among these, XGBoost consistently achieved superior accuracy, approaching 0.99 by the final epoch, while both Random Forest and SVM exhibited steady and comparable improvements. This upward trajectory in accuracy underscores the effectiveness of the applied feature engineering techniques and validates the robustness of the training methodology. The constant improvements indicate that the models progressively learned and internalized underlying patterns in the DevOps performance data as training advanced.

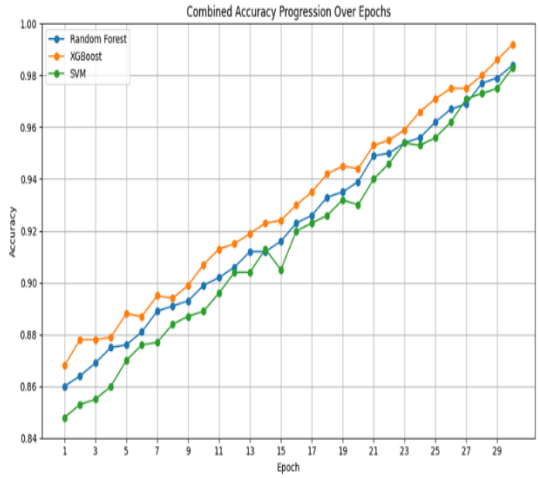

Figure 5: Combined Accuracy Progression Over Epochs

The evaluation metrics comparison across models, as shown in Figure 7, demonstrates that XGBoost outperformed both Random Forest and SVM across all key performance indicators, accuracy, precision, recall, and F1-score, consistently achieving values exceeding 0.99. Random Forest exhibited competitive performance, while SVM trailed slightly, particularly in terms of recall. These results highlight XGBoost's strong predictive capability in accurately identifying operational patterns within DevOps environments. Its consistent superiority across multiple metrics substantiates its selection as the most effective model within the proposed predictive framework.

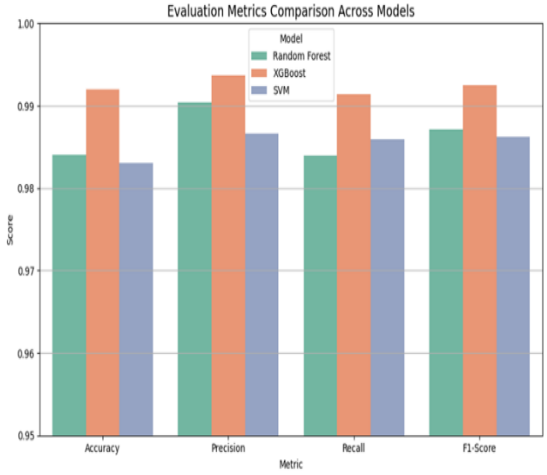

Figure 6: Evaluation Metrics Comparison Across Models

CONCLUSION & FUTURE SCOPE

This study presents a machine learning–driven predictive framework designed to enhance the management of DevOps practices by leveraging time-series sensor data and advanced feature engineering techniques. Through rolling window-based feature extraction and RFE, the model isolates key performance indicators, particularly sensor_2 and sensor_5, as critical contributors to predictive accuracy. Three machine learning classifiers, Random Forest, SVM, and XGBoost, were trained and evaluated on a benchmark dataset. Among them, XGBoost demonstrated consistently superior performance, achieving an accuracy of 99.1%, a precision of 99.3%, a recall of 99.2%, and an F1-score of 99.25%. In comparison, Random Forest achieved an accuracy of 94.8% and SVM lagged with 93.6%. Confusion matrix analysis further confirmed XGBoost's balanced and robust classification ability, with true positive and true negative rates outperforming other models. These empirical results affirm the efficacy of the proposed pipeline in capturing operational patterns within DevOps environments, thereby enabling proactive anomaly detection and intelligent decision-making.

Future research can extend this framework through real-time integration within CI/CD pipelines, enabling low-latency predictive feedback for improved system responsiveness. Incorporating advanced models such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) might enhance temporal accuracy in complex DevOps settings. Broader applicability can be achieved by testing heterogeneous data sources, including infrastructure logs and cloud telemetry. Further, integrating explainability techniques like SHAP and automating responses (e.g., autoscaling, alerts) can enhance operational effectiveness. Finally, validating the framework in compliance-sensitive sectors like finance and healthcare would assess its resilience under stringent regulatory conditions.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to all those who contributed to the successful completion of this research. We are particularly thankful for the

guidance, support, and resources that were made available throughout the course of this work. The insights and encouragement we received played a vital role in shaping this study, and we truly appreciate the assistance provided at every stage.

REFERENCES

[1] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," ACM Computing Surveys (CSUR), vol. 52, no. 6, pp. 1–35, 2019.

[2] P. Perera, R. Silva, and I. Perera, "Improve software quality through practicing DevOps," in 2017 Int. Conf. Advances in ICT for Emerging Regions (ICTer), Sep. 2017, pp. 1–6.

[3] R. T. Yarlagadda, "How DevOps enhances the software development quality," Int. J. Creative Research Thoughts (IJCRT), 2019.

[4] R. R. Alluri, T. A. Venkat, D. K. D. Pal, S. M. Yellepeddi, and S. Thota, "DevOps Project Management: Aligning Development and Operations Teams," J. Science & Technology, vol. 1, no. 1, pp. 464–487, 2020.

[5] Katal, V. Bajoria, and S. Dahiya, "DevOps: Bridging the gap between Development and Operations," in 2019 3rd Int. Conf. Computing Methodologies and Communication (ICCMC), Mar. 2019, pp. 1–7.

[6] Arugula, "Implementing DevOps and CI/CD Pipelines in Large-Scale Enterprises," Int. J. Emerging Res. Eng. Technol., vol. 2, no. 4, pp. 39–47, 2021.

[7] A. Mohammed, "A case study on the management challenges associated with implementing DevOps in small and medium-sized businesses," Int. J. Novel Res. Develop. (www.ijnrd.org), 2018.

[8] Karamitsos, S. Albarhami, and C. Apostolopoulos, "Applying DevOps practices of continuous automation for machine learning," Information, vol. 11, no. 7, p. 363, 2020.

[9] H. Dong, A. Munir, H. Tout, and Y. Ganjali, "Next-generation data center network enabled by machine learning: Review, challenges, and opportunities," IEEE Access, vol. 9, pp. 136459–136475, 2021.

[10] S. Ponomarev, "Intrusion Detection System of industrial control networks using network telemetry," Louisiana Tech University, 2015.

[11] D. A. Bhanage, A. V. Pawar, and K. Kotecha, "IT infrastructure anomaly detection and failure handling: A systematic literature review focusing on datasets, log preprocessing, machine & deep learning approaches, and automated tools," IEEE Access, vol. 9, pp. 156392–156421, 2021.

[12] Zhao, S. Hassan, Y. Zou, D. Truong, and T. Corbin, "Predicting performance anomalies in software systems at run-time," ACM Trans. Softw. Eng. Methodol. (TOSEM), vol. 30, no. 3, pp. 1–33, 2021.

[13] D. I. F. Nocera, T. Di Noia, and D. Gallitelli, "Innovative techniques for agile development: DevOps methodology to improve software production and delivery cycle," 2016.

[14] R. T. Yarlagadda, "How can the public sectors adopt the DevOps practices to enhance the system," Int. J. Emerging Technologies and Innovative Research (www.jetir.org | UGC and ISSN approved), ISSN 2349-5162, 2018.

[15] R. Ciucu et al., "Innovative DevOps for artificial intelligence," Scientific Bulletin of Electrical Engineering Faculty, vol. 19, no. 1, pp. 58–63, 2019.

[16] D. S. Battina, "AI-Augmented Automation for DevOps, a Model-Based Framework for Continuous Development in Cyber-Physical Systems," Int. J. Creative Research Thoughts (IJCRT), ISSN 2320-2882, 2016.

[17] Tyagi, "Intelligent DevOps: Harnessing Artificial Intelligence to Revolutionize CI/CD Pipelines and Optimize Software Delivery Lifecycles," J. Emerging Technologies and Innovative Research, vol. 8, pp. 367–385, 2021.

[18] R. Manchana, "The DevOps Automation Imperative: Enhancing Software Lifecycle Efficiency and Collaboration," European J. Advances in Engineering and Technology, vol. 8, no. 7, pp. 100–112, 2021.

[19] Orynbayeva, "A governance model for managing Robotics Process Automation (RPA)," M.S. thesis, Delft University of Technology, 2019.

[20] [20] J. P. B. de Sá, "Automation of machine learning models benchmarking," M.S. thesis, Universidade do Minho, Portugal, 2021.

[21] R. Rothenhaus, K. De Soto, E. Nguyen, and J. Millard, "Applying a Development Operations (DevOps) Reference Architecture to Accelerate Delivery of Emerging Technologies in Data Analytics, Deep Learning, and Artificial Intelligence to the Afloat US Navy," 2018.

[22] S. Amershi et al., "Software engineering for machine learning: A case study," in Proc. 2019 IEEE/ACM 41st Int. Conf. Software Engineering: Software Engineering in Practice (ICSE-SEIP), May 2019, pp. 291–300.

[23] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," J. Systems and Software, vol. 137, pp. 184–196, 2018.

[24] R. Naseem et al., "Empirical assessment of machine learning techniques for software requirements risk prediction," Electronics, vol. 10, no. 2, p. 168, 2021.

[25] S. Ahmad, Machine Learning for Predictive Cloud Management Revolutionizing IT Monitoring and Maintenance, 2020.

[26] Salunkhe, A. Ayyagiri, A. Musunuri, P. Jain, and D. P. Goel, "Machine Learning in Clinical Decision Support: Applications, Challenges, and Future Directions," Arpit and Goel, Dr. Punit, 2021.

[27] S. Mäkinen, Designing an open-source cloud-native MLOps pipeline, University of Helsinki, 2021.

[28] M. Kansara, "Cloud migration strategies and challenges in highly regulated and data-intensive industries: A technical perspective," Int. J. Applied Machine Learning and Computational Intelligence, vol. 11, no. 12, pp. 78–121, 2021.

[29] S. Shankar and A. Parameswaran, "Towards observability for production machine learning pipelines," arXiv preprint arXiv:2108.13557, 2021.

[30] Tanikonda, S. R. Katragadda, S. R. Peddinti, and B. K. Pandey, "Integrating AI-Driven Insights into DevOps Practices," J. Science & Technology, vol. 2, no. 1, 2021.

[31] V. M. Tamanampudi, "AI and DevOps: Enhancing Pipeline Automation with Deep Learning Models for Predictive Resource Scaling and Fault Tolerance," in Distributed Learning and Broad Applications in Scientific Research, vol. 7, pp. 38–77, 2021.

[32] M. Aniche, E. Maziero, R. Durelli, and V. H. Durelli, "The effectiveness of supervised machine learning algorithms in predicting software refactoring," IEEE Trans. Software Engineering, vol. 48, no. 4, pp. 1432–1450, 2020.

[33] Karamitsos, S. Albarhami, and C. Apostolopoulos, "Applying DevOps practices of continuous automation for machine learning," Information, vol. 11, no. 7, p. 363, 2020.

[34] Á. L. García et al., "A cloud-based framework for machine learning workloads and applications," IEEE Access, vol. 8, pp. 18681–18692, 2020.

[35] M. Schrwatz, "The Role of DevOps in Legacy System Integration," Int. J. Artificial Intelligence and Machine Learning, vol. 6, no. 5, 2019.

[36] M. Inken, "Serverless Computing with DevOps and Cloud Automation: Enabling Scalable and Agile Systems," Int. J. Artificial Intelligence and Machine Learning, vol. 1, no. 2, 2018.

[37] "HELENA2 dataset," Google, [Online]. Available:https://www.google.com/search?q=HELENA2+dataset&oq=HELENA&gs_lcrp=EgZjaHJvbWUqCAgAEEUYJxg7MggIABBFGCcYOzIGCAEQRRg5MgoIAhAuGLEDGIAEMgcIAxAuGIAEMg0IBBAuGLEDGMkDGIAEMgYIBRBFGDwyBggGEEUYPDIGCAcQRRg80gEIMzAwN2owajeoAgiwAgHxBV21GVRBKCa&sourceid=chrome&ie=UTF-8.

[38] D. Katircioglu-Öztürk, H. A. Güvenir, U. Ravens, and N. Baykal, "A window-based time series feature extraction method," Computers in Biology and Medicine, vol. 89, pp. 466–486, 2017.

[39] M.-L. Huang, Y.-H. Hung, W. M. Lee, R.-K. Li, and B.-R. Jiang, "SVM-RFE based feature selection and Taguchi parameters optimization for multiclass SVM classifier," The Scientific World Journal, vol. 2014, p. 795624, 2014.

[40] Paul et al., "Improved random forest for classification," IEEE Trans. Image Processing, vol. 27, no. 8, pp. 4012–4024, 2018.

[41] S. Ramraj, N. Uzir, R. Sunil, and S. Banerjee, "Experimenting XGBoost algorithm for prediction and classification of different datasets," Int. J. Control Theory and Applications, vol. 9, no. 40, pp. 651–662, 2016.

[42] D. M. Abdullah and A. M. Abdulazeez, "Machine learning applications based on SVM classification: a review," Qubahan Academic Journal, vol. 1, no. 2, pp. 81–90, 2021.

[43] T. T. Nguyen, H. Q. Cao, K. V. T. Nguyen, and N. D. K. Pham, "Evaluation of explainable artificial intelligence: SHAP, LIME, and CAM," in Proc. FPT AI Conference, 2021, pp. 1–6.