International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V7I1P118
Eureka Vision Publication | Volume 7, Issue 1, 124-131, 2026

Original Article

Accelerating Defect and Vulnerability Discovery with ML + HPC: High-Throughput Simulation Analytics for Software Quality Engineering

Aditi Mishra[1], Harsh Vardhan[2], Rohan Shetty[3], Pallavi Deshmukh[4]
[1]Information Technology Manipal University Jaipur Jaipur, Rajasthan, India.
[2]Artificial Intelligence Manipal University Jaipur Jaipur, Rajasthan, India.
[3]Information Technology Manipal University Jaipur Jaipur, Rajasthan, India.
[4]Artificial Intelligence Manipal University Jaipur Jaipur, Rajasthan, India.

Received On: 22/12/2025 ***Revised On: 21/01/2026*** ***Accepted On: 30/01/2026*** ***Published On: 08/02/2026***

Abstract - Modern software systems evolve under tight delivery cycles, heterogeneous cloud deployments, and increasingly stringent security and compliance requirements. While machine learning (ML) has demonstrated promise for predicting defect-prone and vulnerability-prone components, many deployments remain bounded by data sparsity, limited execution context, and constrained throughput of dynamic testing. Meanwhile, high-performance computing (HPC) infrastructures provide abundant parallelism, yet they are often underutilized for software quality engineering workflows that require large-scale simulation, test amplification, and telemetry-driven analytics. This manuscript proposes an integrated ML + HPC methodology for accelerating defect and vulnerability discovery through high-throughput simulation analytics. The approach treats quality discovery as a throughput-optimized, data-centric pipeline: (i) generate execution diversity via scalable simulation and test amplification, (ii) capture and normalize multi-level telemetry from builds, tests, runtime traces, and security checks, (iii) learn ranking and classification models that prioritize code regions and change sets for deeper analysis, and (iv) close the loop with root-cause triage and remediation signals. We formalize the research gap as the missing coupling between predictive models and scalable execution diversity, and we outline an evaluation framework spanning prediction quality, defect and vulnerability yield, and end-to-end cost-per-finding. The resulting methodology enables software teams to scale discovery beyond conventional CI constraints, improving early warning capability and accelerating remediation in complex enterprise-grade systems.

Keywords - Defect Prediction, Vulnerability Detection, High-Performance Computing, Simulation Analytics, Test Amplification, Cloud-Native Software Engineering, Explainable AI, Federated Learning.

1. Introduction

Defects and vulnerabilities remain persistent sources of operational risk as software systems evolve toward distributed micro services, regulated data flows, and continuous delivery. In many organizations, discovery is constrained by CI runtime budgets and limited execution diversity, yielding shallow evidence for triage and delayed detection. A central premise of this paper is that discovery should be optimized for throughput: maximize the number of high-value, reproducible findings per unit time while respecting compute and human triage limits.

Large-scale cluster computing established that independent work items can be processed in parallel at scale, transforming throughput economics for data-intensive tasks [1]. Analogously, software-quality discovery contains a large fraction of embarrassingly parallel work configuration sweeps, test variants, fuzzing campaigns, and instrumented simulationsyet these are rarely scheduled as first-class workloads.

Empirical software engineering has established that defects can be predicted from measurable signals, but performance varies by context and evaluation design. A systematic review emphasizes that study design, feature choice, and validation methodology materially affect reported fault prediction performance [2]. In enterprise Java systems, automated testing frameworks and test strategy selection influence reliability outcomes, reinforcing that discovery depends on both predictive analytics and execution practice [3]. Benchmark datasets with traceable links between defects and source artifacts enable comparison across methods, but they generally do not encode the cost structure of execution diversity or the operational decision of which analyses to run next [4].

Security complicates the problem because vulnerabilities are rarer and may not manifest as functional failures. Complexity-based indicators can correlate with vulnerability presence but are insufficient alone in many settings [6]. Consequently, vulnerability discovery typically blends static analysis, dynamic testing, fuzzing, and manual review, each limited by throughput and triage capacity. Comparative studies of defect prediction models suggest that model selection, feature engineering, and validation choices influence practical outcomes [7], [16].

From a systems perspective, iterative analytics workloads benefit from retaining working sets in memory and reusing intermediate artifacts across repeated computations [8]. Software-quality pipelines similarly benefit from caching compiled artifacts, dependency graphs, coverage maps, and extracted features, reducing the marginal cost of additional executions. Privacy-preserving collaboration is increasingly relevant: federated learning demonstrates how multiple parties can learn shared models without centralizing sensitive data, suggesting a pathway for cross-team quality learning when code and telemetry cannot be freely shared [9], [19].

Enterprise cloud-native deployments introduce additional constraints. Secure microservices architectures for regulated processing emphasize segmentation, observability, and repeatable deployment across platforms such as AWS and OpenShift [10]. Explainable, auditable decision pathways are increasingly demanded in regulated domains, motivating interpretable prioritization and evidence aggregation when ML influences engineering actions [11], [34]. Cloud-native deployment optimization and monitoring practices affect reproducibility and stability of large-scale pipelines, particularly when orchestrating thousands of short-running executions [25], [29].

Modern operational pipelines create upstream signal sources that influence software quality. Predictive monitoring in change-data-capture (CDC) pipelines can reduce error propagation and accelerate mitigation [13]. In regulated healthcare automation, OCR and microservice orchestration introduce data-quality and integration risks that can be reflected in telemetry signals used for quality analytics [14], [36]. Architecture-centered decision intelligence for agile governance motivates connecting defect prediction and automated testing to sprint planning and release control rather than treating prediction as a standalone report [15], [31].

The technical opportunity is to couple ML-based prioritization with HPC-capable execution diversity and simulation analytics. Deep learning-based vulnerability detection demonstrates the potential of representation learning on code, but it requires diverse training signals and robust data pipelines [17]. Pre-trained code-language models provide a foundation for semantic feature extraction, complementing traditional metrics and process signals [24], [32]. Graph-based representations can capture dependency context relevant to propagation paths and systemic risk in distributed systems [40].

Finally, enterprise modernization projects illustrate that large refactors and platform migrations introduce new quality risks. SAP S/4HANA transitions and SAP Fiori adoption highlight integration, performance, and operational workflow risks during transformation [22]. In-memory database layer behavior can shape the latency and failure modes of dependent services, motivating memory-centric performance awareness in quality evaluation [26].

2. Motivation and Related Work

Fault prediction research shows that predictive models can identify defect-prone artifacts, but robustness and generalization remain central challenges. The systematic review in [2] highlights variability in reported performance due to dataset composition, validation choices, and feature engineering. Benchmark datasets that link defects to source artifacts are valuable for reproducibility and comparability, but they typically abstract away operational constraints such as test budget and triage capacity [4].

Security prediction is harder due to rarity, semantic specificity, and adversarial context. Complexity metrics can provide weak signals for vulnerability proneness [6], while representation-learning approaches require large and diverse labeled corpora and careful handling of noise [17].

From a compute and systems standpoint, cluster paradigms established scalable patterns for distributing independent tasks [1]. Iterative analytics systems improved throughput by retaining working sets and reusing intermediate artifacts across repeated computations [8]. These ideas motivate high-throughput execution of test and simulation variants, plus caching of compiled artifacts, dependency graphs, and extracted features.

Cloud-native environments introduce operational requirements for reproducibility, monitoring, and controlled deployments. Secure, compliant microservices emphasize segmented architecture, strong identity controls, and disciplined deployment practices [10]. Deployment and monitoring optimization on OpenShift with Helm-based packaging illustrates the operational need for repeatable, observable pipelines at scale [25]. Governance and observability tradeoffs across platforms influence reproducible analytics and are a practical consideration for high-throughput execution [29].

Governance and explainability are salient. Auditable decision pathways are demanded in regulated decision systems [11], and explainable AI frameworks motivate interpretable evidence aggregation even for complex models [34]. Decision intelligence for agile governance and ML-driven CI/CD risk detection emphasize integration into operational workflows and evaluation under deployment constraints rather than offline accuracy alone [15], [35].

Privacy-preserving learning provides a model for cross-team discovery without raw-data centralization, relevant when code and telemetry are sensitive. Federated learning architectures show how distributed parties can collaborate on detection without centralizing raw data [9], [19].

3. Problem Statement and Research Gap

3.1. Problem Statement

Given an evolving codebase C with frequent changes ΔC and bounded CI resources (time, compute, and human triage), the objective is to maximize actionable discovery of defects and vulnerabilities. Let E be a set of candidate executions (tests, simulations, fuzzing campaigns,

configuration sweeps, and instrumented runs). Each execution e ∈ E has cost(e) and yields evidence obs(e) such as failures, traces, coverage deltas, anomaly scores, and security alerts. Let B denote triage capacity per iteration. The objective is to schedule a subset E* ⊆ E that maximizes expected discovery utility under compute and triage constraints, while preserving reproducibility and explainability.

3.2. Research Gap

Current practice frequently separates prediction (offline scoring) from discovery (limited CI executions). Predictors often rely on static signals and do not control which executions are run next, leaving discovery bounded by default test suites and limited configuration coverage. Evaluation practices can overstate performance when they do not match deployment realities such as temporal drift, cross-release shift, and architectural refactors [2], [16]. Vulnerability discovery is additionally constrained by rarity and semantic complexity, where metrics alone are insufficient and learned detectors require diverse evidence and careful evaluation [6], [17]. The core gap is the missing coupling between scalable execution diversity (HPC-capable simulation analytics) and ML-driven scheduling that allocates compute to the highest expected-yield executions. A second gap concerns governance: when ML influences release decisions, evidence must be auditable and interpretable under organizational and regulatory constraints [11], [34].

4. Proposed Ml + Hpc Simulation Analytics Pipeline

We propose H-DSA (High-Throughput Defect and Security Analytics), a pipeline that unifies scalable execution diversity generation, telemetry normalization, ML-based prioritization, and closed-loop remediation.

4.1. Architecture Overview

H-DSA comprises: (1) ingestion and build normalization; (2) execution diversity generation (configuration sweeps, test amplification, targeted fuzzing); (3) telemetry capture and schema normalization; (4) feature store and representation learning (metrics, process signals, semantic embeddings); (5) decision layer with ranking and classification models; and (6) closed-loop triage and remediation feedback.

4.2. HPC Execution Model

The execution layer schedules large numbers of independent runs across a cluster, adopting scalable patterns inspired by distributed task processing [1]. Intermediate artifacts and features are cached to reduce marginal cost, consistent with working-set acceleration principles [8]. Caching and reuse are treated as first-class design elements, aligning with evidence that predictive analytics and Redis-backed caching improve responsiveness in repeated processing workloads [18].

4.3. Cloud-Native Reproducibility and Compliance

The pipeline is deployed as containerized services with controlled environments and repeatable execution. Helm-based packaging and monitoring practices support operational reproducibility and observability [25]. Data handling follows regulated microservices patterns (segmentation, identity controls, and audit logging) [10]. Telemetry channels are protected via encryption and anomaly monitoring to reduce leakage and tampering risk [21].

5. Scheduling and Evidence-Driven Prioritization

5.1. Budgeted Scheduling

Discovery acceleration requires converting predictions into scheduling decisions. Let U(e) be expected discovery utility of execution e, capturing probability of novel findings and expected severity. Under compute budget K and triage budget B, H-DSA selects E* to maximize total utility while respecting constraints. This operationalizes decision intelligence by explicitly connecting ML outputs to CI/CD governance and resource allocation [15].

5.2. Evidence Fusion

Evidence fusion combines static code and process features, semantic representations derived from source code, dynamic signals (coverage deltas, crash signatures, trace anomalies), and architectural context (service boundaries, gateways). Fault-aware microservice transitions motivate including gateway and orchestration context because new failure modes emerge at service seams [33]. Graph-based context can be incorporated via dependency and call graphs to represent propagation paths and systemic risk [40]. For privacy-constrained collaboration, federated aggregation enables local training and model sharing without centralizing sensitive artifacts [9], [19].

5.3. Explainability and Auditability

When the decision layer influences engineering action, explanations must accompany scores. H-DSA produces feature-attribution summaries, counterfactual guidance, and evidence pointers to trace and test artifacts. This aligns with regulatory-grade requirements for auditable decision pathways [11] and interpretability frameworks for high-stakes contexts [34].

6. System Deployment Considerations in Enterprise Environments

6.1. Regulated Microservices and Data Quality

Regulated systems require strict boundaries and controlled data flows. Secure microservices architectures for HIPAA-compliant processing highlight segmentation, identity enforcement, and auditability [10]. OCR-driven workflows illustrate that extraction errors and integration faults can dominate reliability; therefore, pipeline data-quality metrics should be included as first-class signals [14], [36].

6.2. Platform Choice, Monitoring, and Repeatability

Large-scale execution pipelines require repeatable deployments and consistent monitoring. Comparative studies of OpenShift with Helm-based deployments emphasize deployment optimization and observability requirements [25]. Platform comparisons (e.g., Pivotal Cloud Foundry vs. OpenShift) underscore governance and operational tradeoffs that influence reproducible analytics [29].

6.3. CI/CD Governance Integration

H-DSA integrates with CI/CD gates by exposing risk scores and ranked execution plans. Real-world ML-driven CI/CD risk detection highlights the need for evaluating detection under real constraints and governance policies [35]. Agile early fault prediction supports sprint planning by identifying risk hotspots before sprint closure [31].

7. Evaluation Framework

The evaluation objective is to quantify discovery acceleration, not only predictive accuracy.

7.1. Workloads

Evaluation uses defect datasets with traceable defect-to-artifact mapping [4] and time-aware validation aligned with best-practice guidance [2], plus microservice scenarios motivated by modernization effects and gateway behavior [33]. Security evaluation combines metric-based vulnerability indicators [6] with learned vulnerability detection models [17].

7.2. Metrics

Primary metrics include discovery yield (confirmed findings per compute-hour), cost-per-finding (compute plus triage time), precision-at-budget (top-k precision at fixed triage capacity), and operational reproducibility (rerun determinism, failure isolation). Explainability is evaluated by time-to-triage and faithfulness checks aligned with interpretability expectations [34].

7.3. Statistical and Operational Validity

Comparisons should include baselines from comparative defect prediction studies [7], [16] and must control for leakage and temporal drift [2]. Training stability influences calibration and downstream scheduling quality; convergence-aware techniques should be monitored and reported [39].

8. Discussion

8.1. Converting Prediction into Discovery

Offline risk scoring is insufficient unless it changes execution allocation. By coupling ML with scalable execution diversity, H-DSA increases the probability of surfacing rare failures and high-impact vulnerabilities earlier in the lifecycle.

8.2. Security-Specific Challenges

Because vulnerability signals are sparse and adversarial, H-DSA emphasizes multi-evidence fusion: metrics and process signals [6], learned representations, dynamic evidence from simulation, and interpretability for auditability [11], [34]. Secure telemetry channels and anomaly monitoring reduce pipeline exposure [21].

8.3. Data Engineering and Remediation Loops

Large telemetry volumes require robust integration patterns and schema normalization. Closed-loop learning benefits from integrating root-cause analysis and automated remediation patterns for multi-system integrity issues [28].

8.4. Cross-Domain Signals and Operational Context

In practice, quality analytics often benefit from incorporating signals from adjacent operational domains. Examples include monitoring signals from CDC pipelines that indicate upstream data integrity risk [13], platform-layer behavior during in-memory database operations [26], and modernization-induced integration risks in SAP S/4HANA and SAP Fiori transitions [22].

9. Threats to Validity

Internal validity threats include biased execution diversity generation and artificial failure modes. Mitigations include held-out configuration families and reproducibility checks aligned with guidance in fault prediction reviews [2]. Construct validity threats include label noise and proxy outcomes; mitigations include confirmation workflows and feedback-loop tracking. External validity threats include limited generalization across domains and architectures; mitigations include cross-release evaluation and comparative baselines [7], [16]. Conclusion validity requires statistically meaningful comparisons and careful monitoring of training dynamics and calibration [39].

10. Conclusion

This paper presented H-DSA, an ML + HPC approach for accelerating defect and vulnerability discovery through high-throughput simulation analytics. By treating discovery as a throughput-optimized pipeline with evidence-driven scheduling, caching, and explainable decision support, the approach scales beyond conventional CI constraints while aligning with governance and compliance needs.

Future work includes deeper graph-based reasoning over dependency structures [40], broader federated deployments for privacy-constrained collaboration [9], [19], improved calibration and uncertainty-aware scheduling [39], and governance-aligned release policies that combine interpretability with empirical validation [11], [34].

11. Implementation Blueprint for Ml + Hpc Quality Discovery

This section specifies a concrete blueprint for implementing ML + HPC discovery acceleration in production environments.

11.1. Telemetry Schema and Feature Contracts

H-DSA benefits from explicit feature contracts that define schemas for build events, test outcomes, coverage artifacts, runtime traces, and security alerts. Join keys should include commit identifiers, build identifiers, module/service

names, configuration fingerprints, and environment hashes. Unified schemas reduce integration burden when consolidating signals across heterogeneous pipelines, consistent with real-time integration practices in complex ecosystems [27].

11.2. Deterministic Job Packaging

Each execution task (test variant, configuration sweep, fuzzing run, or instrumented simulation) should be packaged as an immutable container with pinned dependencies, fixed toolchain versions, and explicit resource requests. Helm-based packaging and monitoring are important at scale because operational variance can otherwise dominate outcomes [25]. Platform governance differences across orchestration stacks affect repeatability and security controls [29].

11.3. Scheduling Classes and Backpressure

H-DSA partitions workloads into low-latency gating runs, batch discovery sweeps, and background learning jobs. The scheduler enforces backpressure by limiting the number of findings promoted to triage per iteration (budget B), preventing overload. Cluster dispatch follows scalable patterns for distributing independent tasks and aggregating outputs [1].

11.4. Artifact Caching and Reuse

Caching is a primary throughput lever. Compiled artifacts, dependency graphs, static analysis outputs, and feature vectors should be stored in a content-addressable cache keyed by commit/configuration/toolchain fingerprints. Working-set acceleration concepts motivate keeping frequently reused artifacts close to compute [8]. Cache-backed optimization has been shown to improve responsiveness in repeated processing scenarios [18].

11.5. Compliance and Secure Telemetry

When telemetry may contain sensitive traces or regulated indicators, the pipeline must enforce least-privilege access, audited storage boundaries, and protected data-in-transit channels. Secure microservices designs for regulated processing provide design cues for segmentation and auditability [10]. Encrypted and anomaly-monitored channels reduce the risk of tampering and leakage in high-throughput pipelines that exchange large volumes of artifacts [21].

11.6. Explainability Artifacts

H-DSA persists explainability artifacts as first-class objects: feature attribution vectors, evidence pointers to traces/tests, and change-level rationale summaries. Regulatory-grade decision pathways motivate auditable explanations and empirical validation [11]. Interpretability frameworks further motivate explanation faithfulness checks to reduce misleading rationales [34].

12. Experimental Protocol and Reproducibility Guidelines

To evaluate discovery acceleration credibly, experiments must reflect temporal evolution and operational constraints.

12.1. Time-Aware Splits and Leakage Avoidance

Defect prediction and vulnerability detection should be evaluated with time-aware splits where training precedes testing chronologically, avoiding optimistic estimates from leakage. This aligns with best-practice guidance emphasized in fault prediction reviews [2]. When using defect-to-artifact benchmarks, mappings should preserve temporal ordering and avoid duplicate leakage across releases [4].

12.2. Scheduling Evaluation as an A/B Experiment

Because H-DSA changes which executions are performed, scheduling should be evaluated as the primary intervention. A baseline policy (uniform allocation or heuristic allocation) should be compared to ML-guided allocation under identical compute budgets. Comparative defect prediction studies motivate including multiple model families and reporting stability across settings [7], [16].

12.3. Ground Truth Confirmation and Cost-per-Finding

Findings should be considered actionable only when reproducible and traceable to a minimal failing test case or confirmed security report. Cost-per-finding combines compute consumption with human triage effort. ML-driven CI/CD risk detection studies emphasize measuring impact under real deployment constraints rather than purely offline accuracy [35].

12.4. Calibration and Training Stability

Budgeted scheduling benefits from calibrated probabilities and uncertainty estimates. Convergence behavior influences calibration and downstream decision quality; convergence-aware techniques should be monitored and reported [39].

12.5. Audit Logs and Reproducibility Artifacts

Each run should persist manifests containing commit identifiers, configuration fingerprints, container digests, tool versions, and schema versions. Auditability expectations motivate retaining these artifacts for forensic analysis and governance [11].

13. Open Challenges and Limitations

Despite the promise of ML + HPC for discovery acceleration, several challenges remain.

13.1. Label Noise and Sparse Vulnerability Signals

Defect labels can be noisy due to incomplete linkage between issues and commits, while vulnerability labels are often sparse and delayed. Representation-learning approaches require careful corpus construction and consistent labeling to avoid unstable estimates [17]. Complexity-based indicators provide weak signals and may not generalize across architectures [6].

13.2. Human Factors and Triage Bottlenecks

Even when compute scales, triage remains limited. Decision intelligence approaches motivate integrating prioritization with planning and governance processes to ensure findings translate into action [15]. Explainability can reduce triage time, but explanations must be faithful and operationally useful [34].

13.3. Privacy, Cross-Team Learning, and Governance

Cross-team learning improves data diversity but raises privacy and governance concerns. Federated learning provides a pathway, yet introduces challenges in aggregation robustness and heterogeneous feature spaces [9], [19]. Regulated microservices contexts further constrain data movement and necessitate strong audit controls [10], [11].

13.4. Multi-Objective Optimization

Discovery acceleration is multi-objective: maximize yield, minimize cost, prioritize severity, and preserve developer trust. Future work should address multi-objective scheduling policies and decision thresholds that remain defensible under governance scrutiny [11], [35].

References

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in Proc. 6th Symp. Operating Systems Design and Implementation (OSDI), 2004, pp. 137–150.

[2] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, “A systematic literature review on fault prediction performance in software engineering,” IEEE Trans. Softw. Eng., vol. 38, no. 6, pp. 1276–1304, Nov.–Dec. 2012, doi: 10.1109/TSE.2011.103.

[3] Gudi, S. R. (2023). Enhancing Reliability in Java Enterprise Systems through Comparative Analysis of Automated Testing Frameworks. International Journal of Emerging Trends in Computer Science and Information Technology, 4(2), 151-160. https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P115

[4] S. K. Gunda, "Enhancing Software Fault Prediction with Machine Learning: A Comparative Study on the PC1 Dataset," 2024 Global Conference on Communications and Information Technologies (GCCIT), BANGALORE, India, 2024, pp. 1-4, https://doi.org/10.1109/GCCIT63234.2024.10862351.

[5] Indrasena Manga, “Edge Software Engineering for Lightweight AI: Real-Time Environmental Data Processing with Embedded Systems ”, Journal of Computational Analysis and Applications (JoCAAA), vol. 34, no. 6, pp. 88–104, Jun. 2025.

[6] Y. Shin and L. Williams, “An Empirical Model to Predict Security Vulnerabilities Using Code Complexity Metrics,” in Proc. Empirical Software Engineering and Measurement (ESEM), 2008.

[7] S. K. Gunda, "Analyzing Machine Learning Techniques for Software Defect Prediction: A Comprehensive Performance Comparison," 2024 Asian Conference on Intelligent Technologies (ACOIT), KOLAR, India, 2024, pp. 1-5, https://doi.org/10.1109/ACOIT62457.2024.10939610.

[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster Computing with Working Sets,” in Proc. 2nd USENIX Conf. Hot Topics in Cloud Computing (HotCloud), 2010.

[9] Thalakanti, R. R. ., Goud Bandari, S. S., & Sivva, S. D. . (2024). Federated Learning for Privacy Preserving Fraud Detection across Financial Institutions: Architecture Protocols and Operational Governance. International Journal of Emerging Research in Engineering and Technology, 5(2), 108-114. https://doi.org/10.63282/3050-922X.IJERET-V5I2P111

[10] Gudi, S. R. (2024). Design and Evaluation of Secure Microservices Architecture for HIPAA-Compliant Prescription Processing on AWS and OpenShift. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(2), 144-149. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I2P116

[11] Bandari, S. S. G. ., Sivva, S. D. ., & Thalakanti, R. R. (2024). Regulatory Grade Fraud Detection using Explainable Artificial Intelligence with Auditable Decision Pathways and Empirical Validation on Banking Data. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(3), 139-147. https://doi.org/10.63282/3050-9262.IJAIDSML-V5I3P115.

[12] I. Manga, "AutoML for All: Democratizing Machine Learning Model Building with Minimal Code Interfaces," 2025 3rd International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2025, pp. 347-352, doi: 10.1109/ICSCDS65426.2025.11167529.

[13] Reddy Mittamidi VK. Leveraging AI and ML for Predictive Monitoring and Error Mitigation in Change Data Capture Pipelines. IJETCSIT 2025 Aug. 21;6(3):104-11. Available from: https://ijetcsit.org/index.php/ijetcsit/article/view/515

[14] Gudi, S. R. (2024). AI-Driven Fax-to-Digital Prescription Automation: A Cloud-Native Framework Using OCR, Machine Learning, and Microservices for Pharmacy Operations. International Journal of Emerging Research in Engineering and Technology, 5(1), 111-116. https://doi.org/10.63282/3050-922X.IJERET-V5I1P113

[15] Sivva SD, Thalakanti RR, Bandari SSG, Yettapu SDR. AI-Driven Decision Intelligence for Agile Software Lifecycle Governance: An Architecture-Centered Framework Integrating Machine Learning Defect Prediction and Automated Testing. IJETCSIT 2023 Dec. 30 ;4(4):167-72. Available from: https://ijetcsit.org/index.php/ijetcsit/article/view/554

[16] S. K. Gunda, "Comparative Analysis of Machine Learning Models for Software Defect Prediction," 2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2024, pp. 1-6, https://doi.org/10.1109/ICPECTS62210.2024.10780167.

[17] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong, “VulDeePecker: A Deep Learning-Based

System for Vulnerability Detection," in Proc. Network and Distributed System Security Symposium (NDSS), 2018, doi: 10.14722/ndss.2018.23158.

[18] Gudi, S. R. (2024). Leveraging Predictive Analytics and Redis-Backed Caching to Optimize Specialty Medication Fulfillment and Pharmacy Inventory Management. International Journal of AI, BigData, Computational and Management Studies, 5(3), 155-160. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I3P116

[19] I. Manga, "Federated Learning at Scale: A Privacy-Preserving Framework for Decentralized AI Training," 2025 5th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 2025, pp. 110-115, doi: 10.1109/ICSCSA66339.2025.11170780.

[20] Krishna GV, Reddy BD, Vrindaa T. EmoVision: An Intelligent Deep Learning Framework for Emotion Understanding and Mental Wellness Assistance in Human Computer Interaction. 2025 Oct ;6(4):14-20. https://ijaidsml.org/index.php/ijaidsml/article/view/295

[21] S. R. Gudi, "Ensuring Secure and Compliant Fax Communication: Anomaly Detection and Encryption Strategies for Data in Transit," 2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Tirupur, India, 2025, pp. 786-791, https://doi.org/10.1109/ICIMIA67127.2025.11200537

[22] Raikar, T., & Apelagunta, V. (2025). Implementing SAP Fiori in S/4HANA transitions: Key guidelines, challenges, strategic implications, AI integration recommendations. Journal of Engineering Research and Sciences, 4(11), 1–9. https://doi.org/10.55708/JS0411001

[23] Gunda, S. K. (2025). Accelerating Scientific Discovery With Machine Learning and HPC-Based Simulations. In B. Ben Youssef & M. Ben Ismail (Eds.), Integrating Machine Learning Into HPC-Based Simulations and Analytics (pp. 229-252). IGI Global Scientific Publishing. https://doi.org/10.4018/978-1-6684-3795-7.ch009.

[24] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, and D. Jiang, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," arXiv:2002.08155, 2020.

[25] S. R. Gudi, "Monitoring and Deployment Optimization in Cloud-Native Systems: A Comparative Study Using OpenShift and Helm," 2025 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Tirupur, India, 2025, pp. 792-797, https://doi.org/10.1109/ICIMIA67127.2025.11200594

[26] Raikar, T. (2025). High-Performance In-Memory Computing: A Research Study on SAP S/4 HANA Database Layer. American Journal of Technology, 4(2), 93-113. https://doi.org/10.58425/ajt.v4i2.449

[27] I. Manga, "Unified Data Engineering for Smart Mobility: Real-Time Integration of Traffic, Public Transport, and Environmental Data," 2025 5th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 2025, pp. 1348-1353, doi: 10.1109/ICSCSA66339.2025.11170800.

[28] Reddy Mittamidi VK. AI/ML Powered Intelligent Root Cause Analysis and Automated Remediation for Multi System Data Integrity Issues. IJAIBDCMS 2025 Nov. 14;6(4):133-41. Available from: https://ijaibdcms.org/index.php/ijaibdcms/article/view/338

[29] Srikanth Reddy Gudi. (2025). A Comparative Analysis of Pivotal Cloud Foundry and OpenShift Cloud Platforms. The American Journal of Applied Sciences, 7(07), 20–29. https://doi.org/10.37547/tajas/Volume07Issue07-03

[30] Kishore Varma Alluri AK. Using Salesforce CRM and Deep Learning (CNN) Techniques to Improve Patient Journey Mapping and Engagement in Small and Medium Healthcare Organizations. IJAIDSML 2025 Nov. 22 ;6(4):101-9. Available from: https://ijaidsml.org/index.php/ijaidsml/article/view/330

[31] Gunda, S. K., Yalamati, S., Gudi, S. R., Manga, I., & Aleti, A. K. (2025). Scalable and adaptive machine learning models for early software fault prediction in agile development: Enhancing software reliability and sprint planning efficiency. International Journal of Applied Mathematics, 38(2s). https://doi.org/10.12732/ijam.v38i2s.74

[32] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, "A Survey of Machine Learning for Big Code and Naturalness," ACM Computing Surveys, 2018.

[33] S. R. Gudi, "Deconstructing Monoliths: A Fault-Aware Transition to Microservices with Gateway Optimization using Spring Cloud," 2025 6th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2025, pp. 815-820, https://doi.org/10.1109/ICESC65114.2025.11212326

[34] I. Manga, "Towards Explainable AI: A Framework for Interpretable Deep Learning in High-Stakes Domains," 2025 5th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 2025, pp. 1354-1360, doi: 10.1109/ICSCSA66339.2025.11170778.

[35] Thalakanti, R. R., & Goud Bandari, S. S. . (2024). Intelligent Continuous Integration and Delivery for Banking Systems using Machine Learning Driven Risk Detection with Real World Deployment Evaluation. International Journal of AI, BigData, Computational and Management Studies, 5(4), 168-175. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V5I4P118

[36] Gudi, S. R. (2025). Enhancing optical character recognition (OCR) accuracy in healthcare prescription processing using artificial neural networks. European Journal of Artificial Intelligence and Machine Learning, 4(6). https://doi.org/10.24018/ejai.2025.4.6.79

[37] Kishore Varma Alluri AK. Salesforce CRM Framework for Real Time DeFi Portfolio Intelligence and Customer Engagement Forecasting in Web3 Based Decentralized Finance Ecosystems Using ML Techniques. IJAIBDCMS 2025 Nov. 6;6(4):99-107. Available from:

https://ijaibdcms.org/index.php/ijaibdcms/article/view/319

[38] S. K. Gunda, "Automatic Software Vulnerabilty Detection Using Code Metrics and Feature Extraction," 2025 2nd International Conference On Multidisciplinary Research and Innovations in Engineering (MRIE), Gurugram, India, 2025, pp. 115-120, https://doi.org/10.1109/MRIE66930.2025.11156601.

[39] R. R. Thalakanti, "Enhancing Convergence in Fully Connected Neural Networks via Optimized Backpropagation," 2025 2nd International Conference on Computing and Data Science (ICCDS), Chennai, India, 2025, pp. 1-6, doi: 10.1109/ICCDS64403.2025.11209625.

[40] I. Manga, "Scalable Graph Neural Networks for Global Knowledge Representation and Reasoning," 2025 9th International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2025, pp. 1399-1404, doi: 10.1109/ICISC65841.2025.11188341.

[41] Gunda, S.K. (2026). A Hybrid Deep Learning Model for Software Fault Prediction Using CNN, LSTM, and Dense Layers. In: Bakaev, M., et al. Internet and Modern Society. IMS 2025. Communications in Computer and Information Science, vol 2672. Springer, Cham. https://doi.org/10.1007/978-3-032-05144-8_21.