UDC 004.9:004.4

Oleshchenko L.M., assoc. prof., PhD; Tan Huibin, student

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

SOFTWARE TESTING METHOD USING MACHINE LEARNING ALGORITHMS

Анотація

К.т.н., доцент Олещенко Л. М., студент Тан Хуібін

У статті досліджується застосування алгоритмів машинного навчання в тестуванні програмного забезпечення та пропонуються алгоритми Temporal Causal Forest (TCF) для прогнозування дефектів програмного забезпечення та Temporal Deep Isolation Forest (TDIF) для виявлення аномалій. Запропоновані алгоритми реалізовано в програмне забезпечення та порівняно їх ефективність з класичними алгоритмами машинного навчання, які використовуються для тестування програмного забезпечення.

Introduction

As the scale and complexity of modern software systems continue to increase, the demand for intelligent testing solutions is also growing. Traditional testing methods, such as manual testing and rule-based automated testing, often struggle to ensure sufficient coverage, efficiency, and adaptability in dynamic environments. Machine learning (ML) offers a data-driven alternative by learning from historical project data, system logs, and software metrics to predict potential defects and detect anomalies. Existing ML-based methods ignore the inherent temporal and causal structures in software evolution.

The research proposes a comprehensive ML-driven software testing approach that utilizes multiple algorithms, focusing on two proposed algorithms: Temporal Causal Forest (TCF) for defect prediction and Temporal Deep Isolation Forest (TDIF) for anomaly detection. This approach explores aspects such as interpret-ability, adaptability and sustainable improvement.

Related work

Machine learning has become an essential foundation for intelligent software testing, providing automation, prediction, and adaptability beyond traditional manual and automated methods. Early approaches, such as Logistic Regression and Support Vector Machines, offered baseline predictive capabilities for defect classification but lacked adaptability to evolving datasets and complex

software contexts [1, 2]. These models treated input features as static and independent, overlooking temporal dependencies critical for modeling software evolution. The emergence of ensemble learning introduced Random Forests, which improved prediction accuracy through aggregation of multiple decision trees. Despite their robustness, Random Forests exhibited correlational bias and static feature interpretation, leading to poor adaptability in continuous integration and agile environments [3, 4]. To overcome these shortcomings, causal inference techniques such as the Generalized Random Forest (GRF) and Causal Forest were introduced to estimate heterogeneous treatment effects and uncover causal relationships among software metrics and defect generation mechanisms [5], [6].

These approaches enabled transition from correlation-based to causation-based reasoning, helping identify true defect drivers such as code churn, developer workload, and review frequency [7]. Causal Forests still failed to account for time-dependent interactions among variables, limiting their ability to ensure long-term software reliability [8]. Parallel to defect prediction research, anomaly detection has advanced from statistical threshold-based techniques to machine learning approaches. Isolation Forest algorithm detects anomalies by recursively partitioning feature spaces, isolating rare or abnormal instances based on path length [9]. Although effective in low-dimensional domains, traditional Isolation Forests struggle with scalability and interpret-ability in high-dimensional and time-varying environments. Deep Isolation Forest extended this method by incorporating neural representation learning and adversarial optimization to capture nonlinear boundaries and enhance robustness against subtle anomalies [10]. Despite these improvements, both methods treat data points as temporally independent, neglecting sequential correlations and contextual dependencies crucial for modern distributed systems.

To address these limitations, the present research proposes the Temporal Causal Forest (TCF) and Temporal Deep Isolation Forest (TDIF) frameworks, which integrate causal reasoning with temporal modeling and neural representation learning. These models combine interpretability, adaptability, and temporal awareness, while maintaining scalability for real-time, data-driven software testing pipelines.

Proposed software testing method and ML-algorithms

The proposed framework consists of five primary stages: data preprocessing, feature extraction, defect prediction, anomaly detection, and results visualization. Data preprocessing involves cleaning, normalization, and transformation of software metrics and system logs. The feature extraction phase employs natural language processing and statistical methods to derive meaningful representations from both structured and unstructured data sources.

The defect prediction stage uses the Temporal Causal Forest (TCF), which extends the Generalized Random Forest with temporal causality modeling via Double Machine Learning and Granger causality tests. This model identifies causal influences between software metrics, such as code complexity and defect density, across time, thus producing interpretable predictions.

For anomaly detection, the Temporal Deep Isolation Forest (TDIF) introduces neural feature encoding and reinforcement-based tree-depth optimization to improve sensitivity to temporal and contextual variations. It extends the Deep Isolation Forest by dynamically adjusting the isolation boundaries based on real-time log evolution. Together, these two algorithms form a comprehensive testing mechanism capable of detecting both pre-release defects and post-deployment anomalies.

Proposed software system is implemented using a modular microservices architecture. The frontend, developed with React.js, provides an interactive dashboard for uploading data, configuring algorithms, and visualizing results. The backend, powered by FastAPI, handles asynchronous model execution and database communication. Data are managed through PostgreSQL for structured metrics and MongoDB for log data, with Redis ensuring high-speed caching.

Research results

The performance of the proposed software framework was evaluated using the NASA PROMISE dataset for defect prediction and a large-scale operational log dataset for anomaly detection (3,125 distinct software module samples for defect prediction and 100,000 log event records for anomaly detection). Metrics included Accuracy, Precision, F1 Score, and Execution Time.

Core implementations relied on well-established frameworks Scikit-learn, TensorFlow, and FastAPI, enabling both flexibility and reproducibility of results. Baseline algorithms (Logistic Regression, Random Forest, Neural Network, Causal Forest, Deep Isolation Forest) were compared with the proposed TCF and TDIF models under identical preprocessing and validation settings.

The research results indicate that the proposed Temporal Causal Forest (TCF) and Temporal Deep Isolation Forest (TDIF) outperform all baseline algorithms in both prediction accuracy and execution stability (Tables 1-2). TCF improved F1 scores by approximately 15.7% over Causal Forests, while TDIF achieved a 2.6% increase over Deep Isolation Forests (Fig. 1). These results validate the advantage of combining temporal and causal analysis for enhanced software testing reliability and predictive performance.

Table 1.

Comparison of machine learning algorithms for defect prediction

Model	Precision	Recall	F1 Score	Accuracy
Logistic Regression	0.273	0.506	0.355	0.541
Neural Network	0.642	0.668	0.654	0.685
Random Forest	0.498	0.540	0.518	0.598
Causal Forest	0.712	0.538	0.613	0.830
Temporal Causal Forest (proposed)	0.856	0.790	0.77	0.869

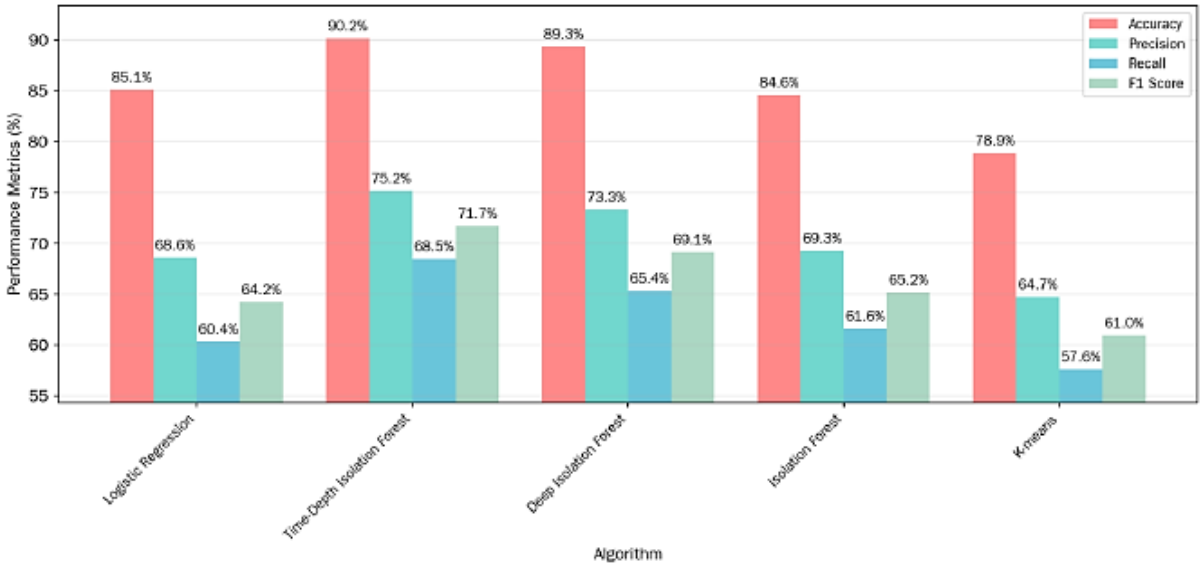

Fig. 1. Comparison of machine learning algorithms for anomaly detection

In terms of resource utilization, both TCF and TDIF maintained stable CPU usage within the range of 50–70%, reflecting efficient multi-core utilization without excessive hardware strain. Memory consumption remained moderate, averaging 1.2–1.9 GB per complete training cycle, indicating a well-balanced trade-off between model complexity and hardware demands.

Conclusions

This paper presented a unified software testing framework that leverages multiple machine learning algorithms to enhance defect prediction and anomaly detection processes. The proposed Temporal Causal Forest and Temporal Deep Isolation Forest incorporate causal reasoning and temporal learning, achieving higher accuracy, better interpret-ability, and improved scalability.

The modular architecture supports continuous feedback, retraining, and adaptive deployment. These results confirm the scalability of the proposed models

for large-scale and real-time software testing environments, where computational constraints are a critical factor. Future optimization directions include continuous model retraining, hybrid ensemble integration, and deployment within CI/CD pipelines to ensure real-time adaptability and industrial applicability.

References

1. Rajbahadur G. K., Wang S., Kamei Y., & Hassan A. E. (2022). The Impact of Using Regression Models to Build Defect Classifiers. arXiv preprint arXiv:2202.06157.
2. Ali H. M., Hamza M. Y., & Rashid T. A. (2024). A Comprehensive Study on Automated Testing with Software Lifecycle. arXiv preprint arXiv:2405.01608.
3. Khalid A., Badshah G., Ayub N., Shiraz M., & Ghouse M. (2023). Software Defect Prediction Analysis Using Machine Learning Techniques. Sustainability, 15(6), 5517.
4. Pandey S. K., Mishra R. B., & Tripathi A. K. (2021). Machine learning based methods for software fault prediction: A survey. Expert Systems with Applications, 172, 114595.
5. Athey Susan, Julie Tibshirani, and Stefan Wager. Generalized random forests. (2019): 1148-1178.
6. Sheetlani J., & Keswani P. (2023). Utilizing Machine Learning For Predicting Software Defects. Journal of Survey in Fisheries Sciences, 10(1), 1–8.
7. Athey Susan, and Guido Imbens. Recursive partitioning for heterogeneous causal effects. Proceedings of the National Academy of Sciences 113.27 (2016): 7353-7360.
8. Daneshjou R., Smith M. P., Sun M. D., Rotemberg V., & Zou J. (2021). Lack of transparency and potential bias in artificial intelligence data sets and algorithms: a scoping review. JAMA dermatology, 157(11), 1362-1369.
9. Al Farizi W. S., Hidayah I., & Rizal M. N. (2021, September). Isolation forest based anomaly detection: A systematic literature review. In 2021 8th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE) (pp. 118-122). IEEE.
10. Xu H., Pang G., Wang Y., & Wang Y. (2023). Deep isolation forest for anomaly detection. IEEE Transactions on Knowledge and Data Engineering, 35(12), 12591-12604.