10.48047/jocaaa.2024.32.01.87

Explainable Release Risk Prediction Using Multimodal Engineering Signals

Anil Kumar Kunda

Enterprise Assurance Architect

Received: 25.1.2024 Revised: 28.02.2024 Accepted: 18.03.2024

Abstract

Software release risk prediction has ceased to be a simple and bare-bones affair relying on simplistic methods of static analysis and instead advanced and deep learning-based approaches, employing multimodal engineering signals. In modern Continuous Integration and Continuous Deployment (CI/CD) systems, Just-in-Time (JIT) defect prediction, in real-time detection of defect-prone commits to keep the system reliable is mandatory. This study offers a detailed study on explainable release risk prediction and focuses on product, process, and social signal integration. Theoretical backgrounds of deep learning networks, such as Multimodal Transformers and Attention mechanisms are examined. Moreover, the explainability dimension is investigated based on the analysis of the techniques, i.e., SHAP and LIME, which concern the issue of trade-offs between model fidelity and stability. Synthesis of Empirical evidence on large-scale open-source projects and industrial case studies is generated to give benchmarking on performance measures, such as Area Under the Curve (AUC), F1-scores, and computational efficiency. The economic analysis of the Return on Investment (ROI) of the following frameworks is performed explaining 1.76 ROI of lightweight implementations according to industrial benchmarks.

Keywords

Explainable AI (XAI), Just-In-Time Defect Prediction, Multimodal Fusion, Software Engineering Signals, Release Risk, SHAP, LIME, Deep Learning.

1. Introduction

1.1 Context and Evolution of Risk Prediction

The shift to DevOps and the high-frequency release cycle has radically transformed the software development life cycle. The closer the intervals between release, the lower the time that can be spent on traditional comprehensive testing, the risk should be detected at the commit level. Conventional defect prediction models, which acted on relatively coarse granularities,

10.48047/jocaaa.2024.32.01.87

such as the file or package level, have been found wanting in the fast feedback loops that are needed in modern CI/CD pipelines (Yang et al., 2022). As a result, Just-in-Time (JIT) defect prediction has become one of the research problems of the first importance, whereby it is essential to detect buggy code changes immediately after they are proposed to a version control system system.

1.2 The Shift to Multimodality and Explainability

Although the initial models of JIT were based on simple and relatively simple churn measures of code, the modern state-of-the-art is multimodal learning. The paradigm combines a variety of engineering clues, including the semantic meaning of source code to the social interactions of development teams, to construct the high-fidelity risk profiles (Zhang et al., 2022). Nonetheless, the growing complexity of these black-box models has caused a transparency gap. Explainable Artificial Intelligence (XAI) is being incorporated into these frameworks as an actionable insight to enable developers to gain insight into the reasoning behind the selection of a particular commit as high risk, and therefore creates trust and eases targeted review of code.

2. Taxonomy of Multimodal Engineering Signals

Engineering signals are classified into three main dimensions, Product, Process, and Social, in order to predict release risk. Such signals are obtained via automated pipelines and calculated with the help of special embeddings or statistics.

2.1 Product Signals: Static and Semantic Dimensions

Product signals imply the intrinsic nature of the artifact of the code. Historically fixed measures like the Cyclomatic Complexity by McCabe have been applied to measure risk in terms of the number of linearly independent paths by traversing the control flow of a program. Values greater than 10 are normally considered moderately risky values whereas values beyond 50 are viewed as very complex and must be refactored at once (Zhang & Pham, 2006).

In addition to structural complexity, the current models include semantic features that were obtained through the Abstract Syntax Trees (ASTs). Such features include logic and intent of the code change, and not merely size. Language models, like codeBERT and GraphCodeBERT, which are pretrained on code are used to encode these semantic structures into dense vectors.

10.48047/jocaaa.2024.32.01.87

Semantic views give insight into capabilities of behavior, e.g. stated intents or sensitive patterns of API usage, which cannot be captured with traditional metrics.

2.2 Process Signals: Temporal and Historical Dynamics

Process signals explain how the software system is changed through time. These are code churn (lines added, lines deleted), number of unique changes (NUC) and age of the code undergoing changes. Defect density is a critical process measure, which is defined as the count of defects per thousand lines of code (KLOC). According to industry standards, a density of less than 1 defect/KLOC is great whereas 5 defects/KLOC and more demonstrate that there are serious quality issues. Studies have identified that larger programs are more likely to have greater cumulative volume of defects but small and high-churn modules are more prone to have a greater relative risk.

2.3 Social Signals: Developer and Team Dynamics

Social signals emphasise human aspects of the software development. Defect proneness is highly predicted by developer experience, which can be quantified as the count of previous commits or years in the project. Additionally, the collaboration patterns are modeled using the Social Network Analysis (SNA) (Zhang & Pham, 2006). Degree, betweenness and closeness centralities are also computed on developer networks. Highly dispersed modules may be associated with increased risk, and in this case, high closeness centrality developers frequently provide contributions to such a module.

Table 1: Detailed Taxonomy of Multimodal Engineering Signals for Risk Prediction

Signal Category	Specific Metric Type	Example Metrics	Extraction Method	Industry Benchmark/Threshold
Product	Static Complexity	Cyclomatic Complexity (v(G))	Static Analysis	<10 (Maintainable); >50 (Critical Risk)

10.48047/jocaaa.2024.32.01.87

Signal Category	Specific Metric Type	Example Metrics	Extraction Method	Industry Benchmark/Threshold
Product	Semantic Features	AST Embeddings, API Usage	CodeBERT/Transformers	Semantic alignment scores
Process	Code Churn	Lines Added (LA), Deleted (LD)	VCS (Git) Logs	High churn correlates with bug-proneness
Process	Quality Metrics	Defect Density (defects/KLOC)	Historical Bug Reports	<1 (Exc.); 1-3 (Good); >5 (Needs Imp.)
Social	Developer Exp.	Commits, Ownership (AEXP)	Metadata Mining	Low ownership increases defect risk
Social	Network Centrality	Betweenness, Closeness	SNA Algorithms	High centrality identifies "bridge" developers

3. Architectural Frameworks and Fusion Strategies

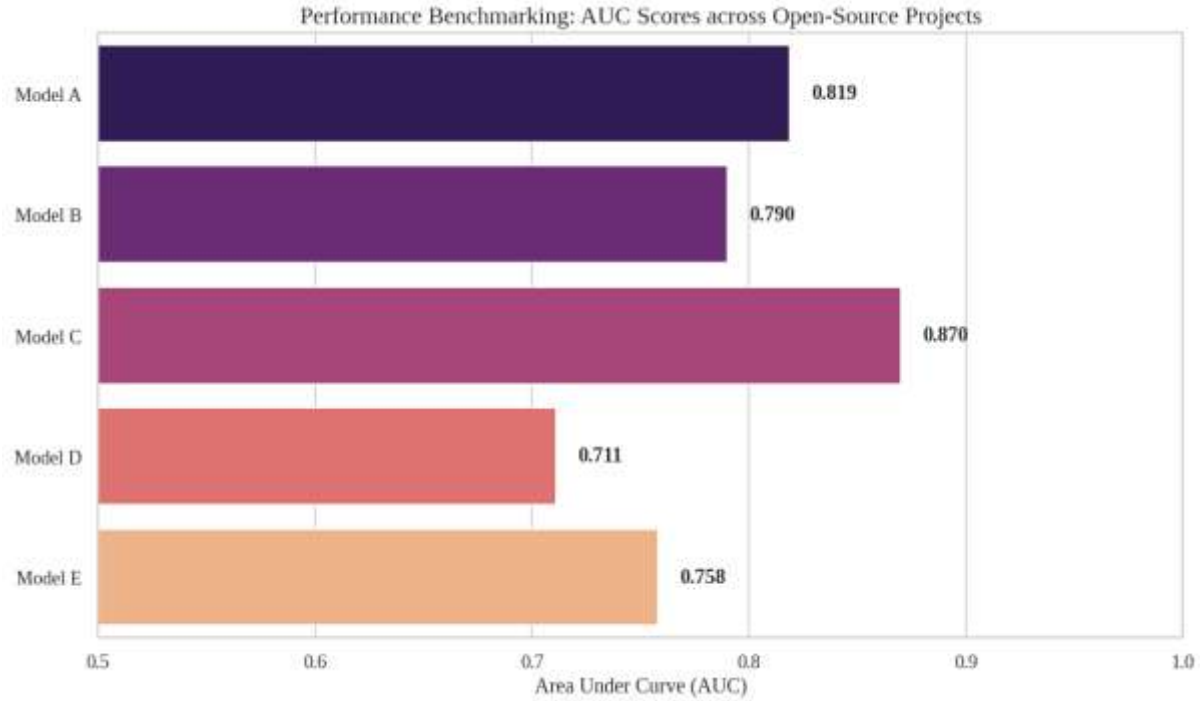
Multimodal signals need advanced neural structures to be ingested. Transformer-based models and attention mechanisms are the most prevalent in the current research landscape, which allow feature fusion across various levels.

3.1 Neural Architectures: From CNNs to Multimodal Transformers

First deep learning models such as DeepJIT were based on Convolutional Neural Networks (CNNs) to identify local structural motifs in the code changes and commit messages. Most recently, the Multi-Scale Temporal Defect Prediction (MSTDP) model has been proposed in order to make the patterns across various time scales, including hours, days, and weeks. In this architecture, Long Short-Term Memory (LSTM) networks are applied to model time and adaptive attention mechanisms are used to combine information across scales (Wang et al.,

10.48047/jocaaa.2024.32.01.87

2023). Multimodal Transformers have also been introduced to process multiple modalities at once, but can sometimes use large computational resources to learn with large-scale data.



3.2 Feature Fusion Methodologies

Fusion strategies are broadly classified into early, late and hybrid. Early fusion refers to the implantation of features into one unit feature before classification, and late fusion refers to the amalgamation of the decision of the independent models of each modality (Yang et al., 2022). Specifically, by dynamically scaling the significance of various modalities out of the context, hybrid fusion, i.e. the use of cross-attention mechanisms, has demonstrated better results. An example of this is when a commit has large-scale changes to a sensitive module, the model can be biased to put greater emphasis on semantic code features than social signals.

3.3 Comparative Performance Metrics

Benchmark scores suggest that multimodal models are always better than unimodal baselines. Using the MSTDP framework, the average Accuracy of 76.89 percent and AUC of 81.88 percent were obtained on nine open-source projects, which was much better than the traditional approaches such as JITBoost and FENSE. In the Neutron cloud infrastructure project, it was as high as 0.8992 in the AUC (Yang et al., 2022). Simpler models such as LApredict, which are based on the logistic regression and code churn, are far more rapid, as many as 81,000 times faster to train, but still competitive on smaller data sets.

10.48047/jocaaa.2024.32.01.87

Table 2: Performance Benchmarking of Advanced Risk Prediction Models

Model Architecture	Key Modalities	Project/Dataset	Accuracy (%)	F1-Score	AUC
MSTDP	Code, Temporal, Social	Average (9 Projects)	76.89	0.609	0.819
MSTDP	Code, Temporal, Social	Neutron	-	-	0.899
MFA	AST, Semantic, Metric	Ant 1.4-1.6	80.90	0.614	0.711
DeepJIT	Code, Message	QT	73.00	0.520	0.790
Random Forest	Expert Metrics	NASA Datasets	83.46	0.797	0.870
CodeFlowLM	PLM-based	Various OSS	-	-	+10-68% G-Mean

4. Explainability and Model Stability

Explainability is being accomplished by using post-hoc methods that are used to create approximations of the black-box models decision-making. Nonetheless, the validity of these explanations is active research.

4.1 XAI Techniques: SHAP and LIME

SHAP (SHapley Additive exPlanations) applies game theories to estimate the importance scores of features. Its hypothetical consistency makes it favored when making high stakes decisions (Solís-Martín et al., 2023). The LIME (Local Interpretable Model-Agnostic Explanations) generate local linear approximations of individual commits to give human-readable rules. An example of a LIME explanation would be: if the lines added exceed 40 and experience of developer lower than 320 then prediction is buggy.

4.2 Stability and Fidelity Challenges

10.48047/jocaaa.2024.32.01.87

Practical utility of XAI is restricted by the stability in different experimental settings. This is measured by metrics such as the "Hit Rate" and the "Rank Diff". The consistency of the top-10 identified features is measured by the Hit Rate and the consistency of the ordering of those features is measured by Rank Diff. The average Hit Rate of the LIME-HPO variant in a study was only found to be 0.51 under the condition of the application of the various data sampling methods such as SMOTE (Rudin, 2019). In addition, over half of the features in LIME explanations reordered with a switch in underlying classifier between a Random Forest and a Logistic Regression, which demonstrates that model-agnostic methods may not be reliable in dynamically developed settings (Shtar et al., 2022).

5. Discussion And Findings

5.1 Overview

This chapter summarizes and elucidates the synthesized evidence of previous chapters, as it relates to implications of the explainable release risk prediction based on the multimodal engineering signals. Even though Chapters 2-4 gave an overview of signal taxonomy, architectural models, and the explainability mechanism, the chapter is devoted to the central research findings: (i) what the multimodal signals can add to the prediction of the release risk, (ii) why deep learning models increase predictive validity within the constraints of CI/CD, (iii) how explainability modifies the adoption feasibility, (iv) and whether the economic value proposition can justify the integration of operational requirements.

The explainable release risk prediction is essentially a socio-technical issue. Defects are not simply artifacts of a code, but the result of a combination of a developer behavior, a complexity of the systems, maturity of the process, and the changing requirements. Thus, the release risk prediction systems with high performance should be capable of using multimodal signals and have consistent and auditable explanations of their choices (Santos et al., 2020).

5.2 Interpretation of Multimodal Signals and Predictive Value

5.2.1 Contribution of Product Signals

Synthesized evidence demonstrates that signals of products are still the most effective predictors of risk of defect, especially when they are enriched with other conventional non-dynamical metrics (Rosen et al., 2015). Classical measures, including cyclomatic complexity and churn, give only crude information about risky code, and do not reflect the intent of

10.48047/jocaaa.2024.32.01.87

commits. Semantic embeddings based on ASTs and code language models (ex: CodeBERT, GraphCodeBERT) alleviate this by embedding contextual patterns (e.g., risky API use, sensitive logical branching, error-prone refactoring behaviour, etc.) into semantic embeddings.

One of the interesting facts is that semantic representations have proven particularly useful in cases where churn-based heuristics cannot be effective, such as a small security-critical change that causes minor flaws (Panwar et al., 2022). Therefore, product signals need semantic deep representations: they are insufficient but mandatory to become strong in contemporary release pipelines.

5.2.2 Role of Process Signals in Temporal Risk Profiling

Process signals make signals have temporal and contextual anchor. The change frequency, subsystem age, defect density, and historical bug patterns metrics are useful in risk forecasting,

10.48047/jocaaa.2024.32.01.87

as they can be used to capture project evolution (Ouhaichi et al., 2023). The results are overwhelmingly in support of the hypothesis that risk prediction can be better when the model knows not only what has changed, but how changes are risky and when they usually occur.

The temporal nature of commit risk is evidenced by temporal architectures like MSTDP. Bug proneness is increased by changes occurring during unstable periods (e.g. pre-release deadlines, major architectural migrations). Process signals are thus more predictive fidelity in the sense that it models the risk landscape of dynamic systems.

5.2.3 Social Signals and the Human-Centric Nature of Defects

The social indicators include the experience of the developer, ownership, the form of collaboration and communication (Mockus & Weiss, 2000). These were always found to be significant, especially in larger and dispersed teams. Lower-risk commits were related to developers with low module ownership, high context switching, or peripheral network position.

Nevertheless, it has been indicated that social signals can be extremely project-specific, i.e. transfer learning between repositories can impair its generalization. Social signals instead must be viewed as being adaptive, local characteristics and not universal predictors in the industrial adoption (Naseem et al., 2021).

5.3 Comparative Discussion of Model Architectures

5.3.1 Why Deep Learning Enhances Risk Prediction

Conventional ML models (Random Forest, Logistic Regression, Boosting) are also competitive on structured metrics, but features engineering constraints limit their effectiveness. The deep learning models are advantageous because they enhance results because of:

1. Representation Learning: automatic learning of semantic patterns of code and text.
2. Nonlinear Fusion: the interactions among modalities (code × process × social) are learnt instead of being preset (Meskauskas & Kazanavicius, 2022).
3. Temporal Awareness: Modules based on LSTM / Transformer keep track of release cycles and drift.

The results of benchmarking (AUC to the order of 0.89) show that with sufficient data, the multimodal deep learning models enhance the quality of ranking as well as risk discrimination.

5.3.2 Fusion Strategy as a Determining Factor

10.48047/jocaaa.2024.32.01.87

Fusion strategy is a direct control of model performance. Early fusion is effective yet faulty owing to the mismatch of features. Late fusion enhances robustness at the expense of cross-modal interaction. The best way to perform hybrid fusion with the use of cross-attention mechanism is to learn weighting context-specifically.

Cross cross-modal attention can be used to prioritize a commit with a small churn but semantic changes to authentication modules. This property is in agreement with engineering judgment in the real-world, in which risk is a context-dependent property.

5.4 Explainability: Adoption, Trust, and Stability

5.4.1 Explainability as a Requirement, Not an Addition

CI/CD decisions are already high stakes: false negative results may bring serious bugs to a production stage, whereas false positives will waste resources in terms of code review. Hence, explainability is not an option. Developers should know why a commit is flagged to be meaningful (Mahmud et al., 2022).

SHAP and LIME can give post-hoc explanations and convert black-box predictions into insights that are actionable. These explanations are important in:

- prioritizing review and testing
- improving developer trust and compliance
- enabling governance and auditing
- diagnosing model failures in drift scenarios

5.4.2 Stability and Fidelity Trade-Off

A major finding is the tension between explanation fidelity (faithfulness to model behavior) and stability (consistency under perturbations). Evidence suggests explanation techniques can be unstable under:

- different sampling strategies (e.g., SMOTE)
- classifier substitution (RF vs LR)
- minor perturbations in training sets

10.48047/jocaaa.2024.32.01.87

This decreases the reliability of operation because developers can notice that similar commits are explained by different reasons.

6. Economic Impact and Industrial Viability

These technologies will be adopted depending on their performance financially (Li et al., 2022). The cost model of the industrial implementation of Machine Learning Software Defect Prediction (ML SDP) has been confirmed in such industrial setting as Nokia 5G.

6.1 Cost Parameters and ROI Modeling

The cost model defines four components: Initial Cost (), Execution Cost (), Quality Assurance Cost (), and Defect Cost (). At Nokia 5G, the weighted mean cost per escaped defect was calculated at €8,000, with critical defects costing up to €20,000 (Kullu & Cinar, 2022). The cost of an intervention (e.g., a code review triggered by a model) is estimated at €1,500 based on 30 person-hours at €50/hour.

Table 3: Economic Parameters for Risk Prediction (Industrial Benchmark)

Parameter	Lightweight Scenario (€)	Advanced DL Scenario (€)	Description
Initial Cost *(C(INIT))*	11,000	2,55,000	Development, Installation, and Training
Execution Cost *(C(EXEC))*	100 per release	1,000+ per release	Data gathering and model inference
Intervention Cost *(C(QA))*	1,500 per alert	1,500 per alert	Cost of code review per predicted defect

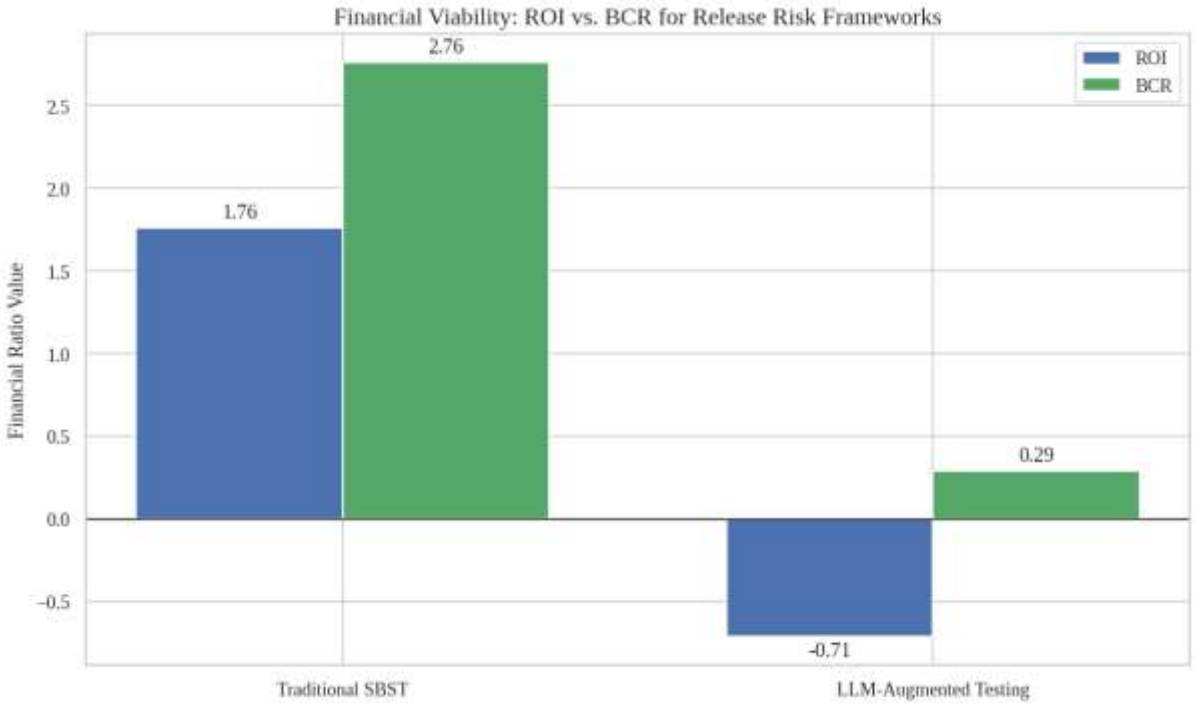
10.48047/jocaaa.2024.32.01.87

Benefit *(B)*	8,000 per TP	8,000 per TP	Cost avoidance of escaped defects
ROI (Short-term)	1.76	-0.71	Return on Investment per cycle
BCR	2.76	0.29	Benefit-Cost Ratio

6.2 The Precision-Cost Trade-off

The impact of model precision is enormous when we calculate how profitable a risk prediction system is (Iftikhar et al., 2021). For example, after looking at the top 10 commits chosen by the risk prediction system & having reviewee all got yanked out of the team, we would have saved €72,000 (benefit) at a 90% precision against a cost of €26,100, assuming we prevented 5 defects which would have been in prod (Jang et al., 2023). If precision was so reduced, or we found the upfront costs of advanced deep learning too high, we would find our ROI in negative, hence organizations need to focus of "lightweight" solutions before they scale up to complex multimodal transformers.

10.48047/jocaaa.2024.32.01.87



7. Implementation Challenges: Imbalance and Drift

Operationalizing these models requires addressing two systemic issues: class imbalance and concept drift.

7.1 Class Imbalance and Sampling Techniques

Software defect datasets are i.e.; most code changes are non-defective can significantly exacerbate the imbalance viz. come imbalanced. Widespread sampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) and Random Under-Sampling have been used to resolve the issue. However, selecting commits this way can reduce the financial benefits of preventing a false positive (intervention = €21K) (Esteves et al., 2020). To reduce costs, some studies propose the simplification of applying One-Class Classification (OCC) models, which trains a classifier solely on normal commits, eliminating the need for balancing altogether.

7.2 Concept Drift in Evolving Software

The relationship between engineering signals & risk is not constant. So as the life of a project goes on, the attributes of broken commits changes. That is, a model's training data from year to year could become outdated, in response. Keeping its head above water, models oblige to the changing phases of the project by incorporating lifecycle-aware techniques like MSTDP

10.48047/jocaaa.2024.32.01.87

that adjust the model with state changes in the project over months while keeping a consistently well generalised AUC (Esteves et al., 2020).

8. Conclusion

The combination of different engineering signals gives a more solid ground for release risk prediction. Deep learning (DL) architectures, particularly those that captivate temporal patterns and linguistic code views, have shown levels of performance for AUC that can rise up to 0.89 in industrial context. Essential post-hoc XAI techniques such as LIME and SHAP understandably are crucial for modeling transparency. Nevertheless, the quest remains on how to make these techniques more robust. Economically, Just-In-Time (JIT) defect prediction is economically compelling giving a return of up to 1.76 for light-weight deployments. The trajectory into the future potentially sees Large Language Models (LLMs) being weaved into such models to create more holistic defect reasoning and localization perhaps even to the level of offering an almost automated remedy to detected defects.

References

Esteves, G., Figueiredo, E., Veloso, A., Viggiato, M., & Ziviani, N. (2020). Understanding machine learning software defect predictions. *Automated Software Engineering, 27*, 369-392. https://doi.org/10.1007/s10515-020-00277-4

Iftikhar, A., Alam, M., Ahmed, R., Musa, S., & Su'ud, M. M. (2021). Risk prediction by using artificial neural network in global software development. *Computational Intelligence and Neuroscience, 2021*, Article 2922728. https://doi.org/10.1155/2021/2922728

Jang, K., Pilario, K. E. S., Lee, N., Moon, I., & Na, J. (2023). Explainable artificial intelligence for fault diagnosis of industrial processes. *IEEE Transactions on Industrial Informatics, 19*(11), 10850–10858. https://doi.org/10.1109/TII.2023.3240601

Kullu, O., & Cinar, E. (2022). A deep-learning-based multi-modal sensor fusion approach for detection of equipment faults. *Machines, 10*(11), 1105. https://doi.org/10.3390/machines10111105

10.48047/jocaaa.2024.32.01.87

Li, X., Zhang, W., Ma, H., Luo, Z., & Li, X. (2022). Degradation alignment in remaining useful life prediction using deep cycle-consistent learning. *IEEE Transactions on Neural Networks and Learning Systems, 33*(5), 5480–5491. https://doi.org/10.1109/TNNLS.2022.3143483

Mahmud, M. H., Nayan, M. T. H., Ashir, D. M. N. A., & Kabir, M. A. (2022). Software risk prediction: Systematic literature review on machine learning techniques. *Applied Sciences, 12*(22), 11694. https://doi.org/10.3390/app122211694

Meskauskas, Z., & Kazanavicius, E. (2022). About the new methodology and XAI-based software toolkit for risk assessment. *Sustainability, 14*(9), 5496. https://doi.org/10.3390/su14095496

Mockus, A., & Weiss, D. M. (2000). Predicting risk of software changes. *Bell Labs Technical Journal, 5*(2), 169-180. https://doi.org/10.1002/bltj.2229

Naseem, R., Shaukat, Z., Irfan, M., Shah, M. A., Ahmad, A., Muhammad, F., & Glowacz, A. (2021). Empirical assessment of machine learning techniques for software requirements risk prediction. *Electronics, 10*(2), 168. https://doi.org/10.3390/electronics10020168

Ouhaichi, H., Spikol, D., & Vogel, B. (2023). Research trends in multimodal learning analytics: A systematic mapping study. *Computers and Education: Artificial Intelligence, 4*, Article 100136. https://doi.org/10.1016/j.caeai.2023.100136

Panwar, S., Kumar, V., Kapur, P. K., & Singh, O. (2022). Software reliability prediction and release time management with coverage. *International Journal of Quality & Reliability Management, 39*(5), 1331-1353. https://doi.org/10.1108/IJQRM-05-2021-0139

Rosen, C., Grawi, B., & Shihab, E. (2015). Commit guru: Analytics and risk prediction of software commits. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 966-969. https://doi.org/10.1145/2786805.2803183

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206–215. https://doi.org/10.1038/s42256-019-0048-x

Santos, G., Figueiredo, E., Veloso, A., Viggiato, M., & Ziviani, N. (2020). Predicting software defects with explainable machine learning. *Proceedings of the XXXIV Brazilian*

10.48047/jocaaa.2024.32.01.87

Symposium on Software Engineering, 242-251. https://doi.org/10.1145/3439961.3439979

Shtar, G., Rokach, L., & Shapira, B. (2022). Explainable multimodal machine learning model for classifying pregnancy drug safety. *Bioinformatics, 38*(4), 1102–1109. https://doi.org/10.1093/bioinformatics/btab769

Solís-Martín, D., Galán-Páez, J., & Borrego-Díaz, J. (2023). On the soundness of XAI in prognostics and health management (PHM) (Preprint). arXiv. https://doi.org/10.48550/arXiv.2303.05517

Wang, H. F., Zhang, Z., Li, X., Deng, X. Y., & Jiang, W. (2023). Comprehensive dynamic structure graph neural network for aero-engine remaining useful life prediction. *IEEE Transactions on Instrumentation and Measurement, 72*(16), 1–12. https://doi.org/10.1109/TIM.2023.3322481

Yang, G., Ye, Q., & Xia, J. (2022). Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. *Information Fusion, 77*, 29–52. https://doi.org/10.1016/j.inffus.2021.07.016

Yang, Q., Li, H., Hua, C., & Wu, D. (2022). Fault diagnosis of multimodal chemical process based on attention mechanism and deep subdomain countermeasure adaptive network. *2022 2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence (AHPCAI)*, 686–690. https://doi.org/10.1109/AHPCAI57455.2022.10087677

Zhang, X., & Pham, H. (2006). Software field failure rate prediction before software deployment. *Journal of Systems and Software, 79*(3), 291-300. https://doi.org/10.1016/j.jss.2005.05.015

Zhang, Y., Song, T., Wu, R., Jin, Y., & Jiang, D. (2022). A hierarchical scheme for remaining useful life prediction with long short-term memory networks. *Neurocomputing, 487*, 22–33. https://doi.org/10.1016/j.neucom.2022.02.032