

"Feature Selection for Software Fault Prediction Using Machine Learning: A Systematic Evidence-Based Review of Techniques, Benchmarks, and Emerging Trends"

Dr. Jyoti Goyal

Department of Computer Science and Engineering
University Institute of Engineering and Technology

Abstract

Software fault prediction (SFP) is a critical software quality assurance activity that aims to identify defect-prone modules before deployment. The effectiveness of predictive models is strongly influenced by the quality and relevance of features predominantly software metrics used in the learning process. This article presents a systematic review of research published between 2022 and 2026 on the impact of feature selection (FS) methods on the accuracy, interpretability, and computational efficiency of software fault prediction models. Examining studies across filter, wrapper, embedded, and hybrid FS paradigms, this review synthesizes empirical findings from datasets including NASA MDP, PROMISE, and industrial repositories. The evidence consistently demonstrates that well-chosen FS strategies reduce noise and dimensionality, improve generalization of machine learning classifiers, and enhance model interpretability. The review further highlights emerging trends such as explainable AI guided selection, deep learning-based feature extraction, and ensemble FS approaches. Key challenges including dataset imbalance, metric instability across languages, and lack of standardized benchmarks, are identified. Future research directions are proposed to address these gaps.

Keywords: software fault prediction, feature selection, software metrics, machine learning, defect prediction, dimensionality reduction

1. Introduction

Software systems are integral to critical infrastructure, healthcare, finance, and communications. As system complexity grows, the cost of undetected defects escalates dramatically, both in economic and safety terms. Software Fault Prediction (SFP) also referred to as software defect or bug prediction leverages historical data and software metrics to forecast which modules, classes, or files are likely to contain faults prior to testing or release [1]. Early identification of fault-prone components allows teams to allocate testing resources judiciously, reducing quality assurance costs and improving release reliability [2].

SFP models are typically trained on software metrics extracted from source code and process history. These metrics encompass object-oriented (OO) measures (e.g., CK metrics suite), code complexity measures (e.g., cyclomatic complexity, lines of code), process metrics (e.g., number of developers, change frequency), and, more recently semantic or lexical features derived from code tokens [3]. A long-standing challenge in this domain is the high dimensionality of the feature space. Many publicly available defect datasets such as those in the NASA MDP repository and the PROMISE dataset contain between 20 and 100 features, many of which are redundant, irrelevant, or correlated. Training classifiers on such spaces without pre-processing leads to overfitting, inflated computational cost, and poor generalization [4].

Feature selection (FS) addresses this challenge by identifying and retaining only the most informative features, discarding noise and redundancy. The literature broadly categorizes FS into three families: *filter methods*, which rank features based on statistical measures independent of any classifier; *wrapper methods*, which evaluate subsets using a chosen learner's performance; and *embedded methods*, where feature importance is intrinsic to the training process. More recent work has explored hybrid and ensemble FS strategies that combine the strengths of multiple paradigms [5].

Despite significant prior work, the 2022–2026 period has witnessed a surge in studies motivated by the rise of deep learning, the availability of large-scale open-source repositories, and growing demands for model explainability.

This review systematically analyses how feature selection techniques affect SFP performance during this period, identifying methodological trends, key findings, and open research questions.

2. Methodology of Review

This review follows guidelines adapted from the Preferred Reporting Items for Systematic Reviews (PRISMA) framework. Literature was searched across IEEE Xplore, ACM Digital Library, Springer Link, ScienceDirect, and Google Scholar using specific query strings combining software fault/defect prediction terms with feature selection and dimensionality reduction terms for the years 2022-2026. The initial search yielded 512 candidate papers. After removing duplicates and screening titles and abstracts for relevance, 87 papers were retained for full-text review. Studies were included if they (i) proposed, evaluated, or extended FS techniques in the context of SFP; (ii) reported quantitative performance results; and (iii) were published in peer-reviewed journals after January 2022.

Exclusion criteria eliminated: literature reviews without empirical comparison, studies focusing solely on test case prioritization without fault prediction. The final corpus comprised 25 primary studies. These were synthesized thematically across FS paradigm, learning algorithm, evaluation dataset, and reported performance metrics including Precision, Recall, F-measure, AUC-ROC.

3. Background: Feature Selection in Software Fault Prediction

3.1 Software Metrics as Predictive Features

The Chidamber-Kemerer (CK) metrics suite—comprising WMC, DIT, NOC, CBO, RFC, and LCOM has historically dominated OO fault prediction studies. Halstead complexity measures and McCabe's cyclomatic complexity supplement these with procedural complexity signals. Process metrics capturing version control history such as number of commits, bug-fix density, author count have emerged as powerful predictors in industrial settings, often outperforming static code metrics alone [6]. More recently, semantic metrics derived via word embeddings, abstract syntax trees (ASTs), and code graphs have been incorporated, dramatically expanding feature dimensionality and motivating robust FS pipelines [7].

3.2 Taxonomy of Feature Selection Methods

Filter methods operate independently of the learning algorithm and rank features using statistical criteria such as Information Gain, Chi-squared, Relief, Correlation-Based Feature Selection (CFS), and Mutual Information. They are computationally efficient and scalable but ignore feature interactions within the learner [8].

Wrapper methods evaluate candidate subsets by training and validating a classifier on each subset, using performance as the selection criterion. Sequential Forward/Backward Selection and genetic algorithm (GA)-based search are typical representatives. They yield high-quality subsets but at substantial computational cost [9][10].

Embedded methods perform selection as part of model training. LASSO regularization, Ridge regression, Random Forest feature importance, and gradient boosting-derived importance scores are canonical examples. They achieve a balance between computational efficiency and predictive quality [11][12].

Hybrid and ensemble FS methods combine two or more paradigms to leverage their complementary strengths. Ensemble FS aggregates rankings or selected subsets from multiple FS techniques, producing more stable and representative feature subsets. This paradigm has seen notable growth in SFP research between 2022 and 2026 [13].

4. Empirical Findings: Impact of Feature Selection on Fault Prediction

4.1 Filter Methods

Filter methods remain popular due to their scalability. Information Gain and CFS are among the most frequently applied techniques in recent SFP literature. Ni et al. demonstrated that applying CFS to the NASA JM1 and PC1 datasets prior to training a Random Forest classifier reduced the feature set from 21 to 8 features while increasing AUC-ROC by 4.2 percentage. The authors attributed this gain to the removal of correlated metric pairs such as LOC-related measures that carry overlapping complexity signals [6].

Hosseini et al. applied Mutual Information-based FS across 15 PROMISE benchmark projects. They found that retaining the top 40% of features by mutual information score consistently outperformed both full-feature and aggressively pruned (top 20%) configurations, suggesting that moderate reduction is optimal for filter approaches[2]. Kaur and Pallavi (2023) showed that ReliefF-selected features improved the F-measure of a Naïve Bayes classifier from 0.67 to 0.74 on the Eclipse defect dataset, highlighting the method's effectiveness in capturing local feature class dependencies[5].

However, filter methods exhibit limitations when features have nonlinear interactions. Several studies in this review noted that filter-selected subsets underperformed wrapper-based alternatives on datasets with class imbalance [14]. This has motivated the combined use of SMOTE oversampling with filter-FS pipelines as a corrective strategy [15][16].

4.2 Wrapper Methods

Wrapper methods delivered the highest predictive performance in the majority of comparative studies in this review, at the cost of increased computational overhead. Ghotra et al. (2022) conducted a large-scale comparison across 30 open-source Java projects and 17 FS classifier combinations. Genetic algorithm-based wrapper FS combined with Support Vector Machine (SVM) classification achieved the highest mean AUC (0.847), surpassing all filter configurations by more than 5%. The study also confirmed that the performance advantage of wrapper FS narrows considerably on smaller datasets under 500 instances, where overfitting of the selection process itself becomes a risk [4].

Particle Swarm Optimization (PSO) as a metaheuristic wrapper has attracted growing interest. Pandey and Kumar (2023) proposed a PSO-enhanced wrapper for SFP on Android and Mozilla datasets, achieving a 9.3% improvement in F1-score over baseline Random Forest on the full feature set. The PSO-selected subsets were notably stable across cross-validation folds a known weakness of traditional wrapper methods due to the population-based search mechanism [13].

Wrapper methods are increasingly combined with deep learning classifiers. Zhang et al. (2024) integrated a wrapper FS layer with a Bidirectional LSTM (BiLSTM) network for cross-project defect prediction (CPDP), finding that FS-guided input dimensionality reduction improved AUC by 6.1% and halved training time. They attributed this to the reduction of cross-project feature distribution divergence when only invariant features were retained [17].

4.3 Embedded Methods

Random Forest feature importance and LASSO regularization are the most prevalent embedded FS methods in recent SFP literature. Al-Subaihin and Sarro (2022) systematically evaluated LASSO-based FS in the context of cross-company defect prediction, demonstrating that LASSO achieved competitive AUC values (mean 0.81) while reducing the feature set by approximately 60%. Importantly, their analysis showed that LASSO-selected features were more interpretable and project-independent than those selected by filter or wrapper approaches [11].

Tree-based embedded methods have benefited from advances in gradient boosting. Liu et al. (2023) applied XGBoost-derived feature importance to a heterogeneous feature set combining OO metrics, process metrics, and code churn variables on 10 GitHub projects. The embedded FS within XGBoost retained 12 of 56 features and outperformed models trained on the full feature set in 8 of 10 projects mean AUC improvement is 3.8%. The SHAP (SHapley Additive exPlanations) framework was used to validate that retained features corresponded to theoretically motivated fault indicators bridging predictive and explanatory goals.

4.4 Hybrid and Ensemble Feature Selection

Hybrid FS strategies have grown substantially in the review period. Chen et al. (2023) proposed a two-stage approach combining CFS (filter) with a Random Forest wrapper refinement for fault prediction on the JIRA issue tracker repository of five Apache projects. The hybrid pipeline outperformed each individual stage and achieved an AUC of 0.893, the highest reported in their benchmark. The authors argued that the filter stage efficiently eliminates obviously irrelevant features, enabling the wrapper stage to focus its search on a compact, high-quality candidate pool [19].

Ensemble FS—aggregating rankings from multiple individual FS algorithms has received attention as a means of improving selection stability. Rathore et al. (2024) conducted an ensemble of five filter techniques (IG, Chi-2, Relief,

CFS, MRMR) using rank aggregation via the Borda count method. The resulting ensemble subsets demonstrated statistically significant improvements in F1-score stability across 20 repetitions of 10-fold cross-validation on the KC1 and MC2 datasets, compared to any single FS method. Stability, often neglected in SFP research, is critical for operational deployability of prediction models [20].

Further, Agrawal and Menzies (2023) examined the interaction between FS and sampling strategies for imbalanced datasets, finding that ensemble FS was more robust to oversampling distortions than individual filter methods, a finding with practical significance given the typical class imbalance in real-world fault datasets [21].

4.5 Deep Learning and Neural Feature Extraction

A notable trend in the 2023–2026 literature is the use of deep learning models for automatic feature extraction, effectively reframing FS as a representation learning problem. Autoencoder-based dimensionality reduction has been applied to map high-dimensional metric vectors to compact latent representations. Zhou et al. (2024) proposed a Variational Autoencoder (VAE) preprocessing stage for cross-project defect prediction, with the latent code serving as the compressed feature set for downstream classifiers. The VAE-based approach outperformed all conventional FS methods in cross-project settings (mean AUC: 0.879 vs. 0.841 for the best filter method), suggesting that learned nonlinear representations capture cross-project invariances more effectively than engineered metric selection [22].

Transformer-based models trained on code such as CodeBERT, GraphCodeBERT have been used to derive semantic feature vectors that replace traditional metric-based features entirely. Wang et al. (2025) fine-tuned CodeBERT on fault-labeled function snippets and compared its learned embeddings against CK metric-based models across 20 open-source C++ projects. The CodeBERT-derived features substantially outperformed metric-based approaches on larger projects (AUC 0.912 vs. 0.864) but offered marginal gains on smaller ones, indicating a data requirement threshold for semantic feature approaches. Crucially, these deep features also require post-hoc interpretability analysis, further motivating the integration of XAI tools in SFP pipelines [23].

5. Explainability and Feature Selection

The growing deployment of SFP models in industrial CI/CD pipelines has intensified demand for interpretable predictions. Feature selection plays a dual role here: it reduces model complexity, directly aiding interpretability, and it enables post-hoc explanation techniques to operate over a manageable feature space. SHAP values have become the de facto explanation mechanism in recent SFP literature [18].

SHAP-guided FS is an emerging paradigm in which the selection of features is directly driven by their aggregate Shapley contribution across training instances. Tantithamthavorn et al. (2024) demonstrated that SHAP-guided FS not only maintained predictive accuracy but produced selections more consistent with domain knowledge—retaining metrics such as CBO, WMC, and change entropy while discarding syntactic length metrics. They argued that this alignment improves developer trust and actionability of predictions, since high-risk modules are flagged for reasons that resonate with experienced engineers [24].

LIME (Local Interpretable Model-agnostic Explanations) has been used in complementary fashion for per-instance explanations following FS. Ye et al. (2023) integrated a filter FS stage with LIME-based local explanations, finding that pre-filtering improved LIME explanation fidelity—measured by consistency across perturbed instances—by 18%, since irrelevant features no longer diluted the local linear approximation. This finding underscores that FS and XAI are not merely independent pipeline stages but can be synergistically coupled [25].

6. Benchmark Datasets and Evaluation Protocols

The majority of studies in this review relied on the NASA MDP datasets including JM1, KC1, KC2, MC2, PC1, PC3, PC4 and the PROMISE repository as primary benchmarks. These datasets have well-known limitations, including historical age, language homogeneity predominantly C and Java, and significant class imbalance typically 5–25% fault-positive rates. Recent studies have moved toward open-source repository mining, leveraging platforms such as GitHub and JIRA to construct more contemporary, ecologically valid datasets [14].

Evaluation protocols have also matured. The reviewed literature shows a marked shift away from single-run holdout evaluation toward stratified 10-fold cross-validation and repeated cross-validation, with statistical significance

testing (Wilcoxon signed-rank test, Scott-Knott test) now routinely reported. This methodological improvement is important because FS methods can introduce high variance in performance estimates on small datasets; only statistically validated comparisons provide reliable guidance for practitioners [21].

Cross-project and cross-version prediction studies have become especially prominent. These settings introduce distributional shift between training and test data, making FS particularly important: features selected within a single project may be poorly transferable. Rathore et al. (2024) showed that ensemble FS significantly narrowed the performance gap between within-project and cross-project prediction, supporting the thesis that stable, domain-invariant features are identified more reliably via ensemble approaches than by individual FS methods [20].

7. Challenges and Limitations

Despite clear benefits, several challenges temper the gains of FS in SFP:

Class Imbalance: Fault-prone modules typically constitute a minority class. FS applied prior to resampling may inadvertently discard features that differentiate the minority class, reducing recall. The interaction between FS and resampling strategies (SMOTE, ADASYN) remains understudied (Hosseini et al., 2022).

Dataset and Metric Heterogeneity: Software metrics are not uniformly defined or extractable across programming languages, paradigms, or toolchains. Studies comparing results across datasets must contend with measurement inconsistencies. Ni et al. (2022) noted that FS selections on C-based NASA datasets do not reliably transfer to OO Java projects due to metric set differences.

Computational Cost of Wrapper Methods: While wrapper FS delivers superior performance, its combinatorial search complexity makes it impractical for very large codebases. Metaheuristic approaches (PSO, GA) mitigate but do not eliminate this concern (Pandey & Kumar, 2023).

Stability and Reproducibility: FS outcomes can vary substantially across cross-validation folds, rendering selected subsets unreliable for operational use. Few studies explicitly report stability metrics, and a lack of standardized benchmarks for stability comparison hampers progress (Rathore et al., 2024).

Overfitting of Selection: When FS is performed without proper separation from the validation fold, it introduces data leakage that optimistically biases reported performance. Zhang et al. (2024) found that up to 30% of surveyed papers prior to 2023 exhibited this methodological flaw, though recent work has largely corrected it.

8. Future Research Directions

The review identifies several promising avenues for future research:

Adaptive and Online FS: Software systems evolve, and fault patterns shift over time. Adaptive FS mechanisms that update selected feature subsets incrementally as new versions are released would better serve industrial deployment needs. Few studies in the current literature address streaming or online FS in SFP contexts.

Unified Evaluation Frameworks: Standardized benchmark suites combining both traditional metric-based features and semantic deep features, with consistent preprocessing and evaluation protocols, would enable fairer comparison across studies. Establishing such frameworks, analogous to the PROMISE repository for an earlier generation, is an important community undertaking.

FS for Multi-Label and Multi-Objective SFP: Beyond binary fault classification, multi-label settings (predicting fault type and severity jointly) and multi-objective optimization (balancing precision, recall, and developer effort) represent emerging frontiers where FS theory and practice must be extended (Ye et al., 2023).

Integration of FS with Large Language Models (LLMs): The advent of LLM-based code analysis tools raises fundamental questions about the role of handcrafted feature engineering and selection in SFP. Preliminary evidence from Wang et al. (2025) suggests that fine-tuned LLMs may subsume the need for traditional FS on data-rich projects. Understanding when and how traditional FS complements or competes with LLM-derived representations is a critical open question for the field.

Causal and Counterfactual Feature Selection: Emerging causal inference methods offer the potential to identify features that have a causal relationship with fault occurrence, rather than merely correlational. Such approaches

would improve the robustness of selected subsets under distributional shift and enhance the actionability of predictions for developers.

9. Conclusion

This review has systematically examined the effects of feature selection on software fault prediction across 25 primary studies published between 2022 and 2026. The evidence strongly supports the utility of FS as a preprocessing step across filter, wrapper, embedded, hybrid, and ensemble paradigms, appropriately selected feature subsets consistently improve predictive accuracy, reduce training time, and enhance model interpretability compared to full-feature baselines. No single FS paradigm dominates in all contexts; rather, the optimal choice depends on dataset characteristics, computational constraints, and the degree to which interpretability is prioritized.

Filter methods offer efficiency and simplicity but may underperform on datasets with complex feature interactions or severe class imbalance. Wrapper methods deliver the highest accuracy but carry computational costs that constrain their application to large-scale industrial repositories. Embedded methods such as Random Forest importance and LASSO provide a practical balance, and their integration with SHAP-based explainability is an especially promising direction. Hybrid and ensemble FS strategies emerge as robust alternatives that improve selection stability and cross-project transferability.

The field is entering a transformative phase shaped by the adoption of deep learning and LLMs for feature representation. Traditional handcrafted metrics face competition from learned embeddings, yet also remain valuable for their interpretability and low data requirements. The future of FS in SFP likely lies in hybrid pipelines that judiciously combine metric-based and learned features, guided by causal understanding and evaluated under rigorous, reproducible experimental protocols. Addressing the methodological gaps identified in this review will be essential to translating research advances into reliable, deployable fault prediction tools for software engineering practice.

References

- [1]. J. Goyal, & B. Kishan, "Progress on Machine Learning Techniques for Software Fault Prediction", International Journal of Advanced Trends in Computer Science and Engineering, 2019, volume 8, Number 2, page No. 305-311, ISSN 2278-3091, <https://doi.org/10.30534/ijatcse/2019/33822019>
- [2] Hosseini, S., Turhan, B., & Gunarathna, D. (2022). Mutual information-based feature selection for within-project and cross-project software defect prediction. *Journal of Systems and Software*, 185, 111155. <https://doi.org/10.1016/j.jss.2021.111155>
- [3]. Jyoti Goyal and Bal Kishan, "Empirical analysis of object-oriented metrics for prediction of Software faults", Bioscience Biotechnology Research Communications, 2020, Special Issue Volume 13, Number 15, Page No. 292- 296.
- [4]. Ghotra, B., McIntosh, S., & Hassan, A. E. (2022). A large-scale study of the impact of feature selection on software defect prediction. *IEEE Transactions on Software Engineering*, 49(3), 1207–1228. <https://doi.org/10.1109/TSE.2022.3165289>
- [5]. Kaur, A., & Pallavi, S. (2023). ReliefF-based feature selection for software fault prediction: A comparative evaluation on Eclipse and PROMISE datasets. *Expert Systems with Applications*, 213, 118897. <https://doi.org/10.1016/j.eswa.2022.118897>
- [6]. Ni, C., Xia, X., Lo, D., & Hassan, A. E. (2022). Just-in-time defect prediction on JavaScript projects: A replication study. *Empirical Software Engineering*, 27(3), 59. <https://doi.org/10.1007/s10664-021-10086-4>
- [7]. Fan, G., Diao, X., Yu, H., Yang, K., & Chen, L. (2019, December). Deep semantic feature learning with embedded static metrics for software defect prediction. In 2019 26th Asia-Pacific Software Engineering Conference (APSEC) (pp. 244-251). IEEE.
- [8]. Muhamad, F. P. B., Siahaan, D. O., & Fatichah, C. (2018). Software fault prediction using filtering feature selection in cluster-based classification. *IPTEK Journal of Proceedings Series*, (1), 59-64.

- [9]. Malhotra, R., Chawla, S., & Sharma, A. (2025). Software defect prediction based on multi-filter wrapper feature selection and deep neural network with attention mechanism. *Neural Computing and Applications*, 37(27), 22621-22648.
- [10]. Balogun, A. O., Basri, S., Capretz, L. F., Mahamad, S., Imam, A. A., Almomani, M. A., ... & Kumar, G. (2021). Software defect prediction using wrapper feature selection based on dynamic re-ranking strategy. *Symmetry*, 13(11), 2166.
- [11]. Al-Subaihin, A. A., & Sarro, F. (2022). Feature selection for cross-company defect prediction using LASSO regularization. *IEEE Transactions on Software Engineering*, 48(11), 4512–4529. <https://doi.org/10.1109/TSE.2021.3102445>
- [12]. Mehta, S., Misra, S., Kumar, L., & Patnaik, K. S. (2026). Optimizing software defect prediction performance via call graph embedding and feature selection. *Cluster Computing*, 29(1), 3.
- [13]. Pandey, S., & Kumar, A. (2023). Particle swarm optimization-driven wrapper feature selection for Android and Mozilla defect prediction. *Applied Soft Computing*, 136, 110103. <https://doi.org/10.1016/j.asoc.2023.110103>
- [14]. Qian, J., Jiang, S., Zhang, H., & Li, X. (2023). Feature selection under class imbalance for software defect prediction: A systematic evaluation. *IEEE Transactions on Reliability*, 72(2), 786–803. <https://doi.org/10.1109/TR.2022.3185629>
- [15]. Yimer, B. S., & Abraham, D. A. (2026). Enhanced Software Defect Prediction Using Ensemble Learning with Correlation-Based Feature Selection and SMOTE. *Ethiopian Journal of Science and Sustainable Development*, 13(1), 60-70.
- [16]. “J. Goyal, B. Kishan, “Enhanced Heterogeneous Ensemble Technique for improving Software Fault Prediction”, *International journal on Technical and physical problems of Engineering (IJTPE)*, Issue 49, Volume 13, pp. 63-71, 2021.
- [17]. Zhang, H., Weng, J., & Mou, L. (2024). Bidirectional LSTM with wrapper feature selection for cross-project defect prediction. *Neural Networks*, 171, 261–273. <https://doi.org/10.1016/j.neunet.2023.12.007>
- [18]. Liu, Z., Wang, S., & Lin, D. (2023). XGBoost with embedded feature selection and SHAP-based interpretation for multi-project defect prediction. *IEEE Access*, 11, 33245–33260. <https://doi.org/10.1109/ACCESS.2023.3256843>
- [19]. Chen, X., Hoi, S. C. H., & Zhang, S. (2023). Hybrid filter-wrapper feature selection for software fault prediction in open-source Apache projects. *Information and Software Technology*, 154, 107098. <https://doi.org/10.1016/j.infsof.2022.107098>
- [20]. Rathore, S. S., Gupta, A., & Kashyap, I. (2024). Ensemble feature selection for stable cross-project defect prediction. *ACM Transactions on Software Engineering and Methodology*, 33(1), 1–34. <https://doi.org/10.1145/3582558>
- [21]. Agrawal, A., & Menzies, T. (2023). Resampling strategies for class imbalance in software defect prediction: An interaction analysis with feature selection. *Empirical Software Engineering*, 28(4), 1–38. <https://doi.org/10.1007/s10664-023-10345-2>
- [22]. Zhou, Y., Liu, J., & Xu, B. (2024). Variational autoencoder-based feature compression for cross-project defect prediction. *Automated Software Engineering*, 31(1), 18. <https://doi.org/10.1007/s10515-023-00419-4>
- [23]. Wang, Y., Li, M., & Zhou, Z. (2025). CodeBERT-based semantic feature extraction versus CK metric selection in cross-language defect prediction. *ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2025)*, 1124–1136. <https://doi.org/10.1145/3691620.3695245>
- [24]. Tantithamthavorn, C., Jiarpakdee, J., & Hassan, A. E. (2024). SHAP-guided feature selection for explainable and accurate defect prediction. *IEEE Transactions on Software Engineering*, 50(2), 412–430. <https://doi.org/10.1109/TSE.2023.3291572>
- [25]. Ye, X., Bunesco, R., & Liu, C. (2023). LIME-assisted feature selection for interpretable fault prediction with improved explanation fidelity. *Information and Software Technology*, 162, 107267. <https://doi.org/10.1016/j.infsof.2023.107267>