Improving the Performance of TFS with Ensemble Learning for Cross-Project Software Defect Prediction

Pathiah Abdul Samat[1], Yahaya Zakariyau Bala[2], Nur Hamizah Hamidi[3]
Department of Software Engineering and Information System-Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Malaysia[1, 3]
Department of Computer Science, Federal University of Kashere, Gombe 771103, Nigeria[2]

***Abstract*—Software defect prediction (SDP) plays a key role in improving software quality by identifying defect-prone modules early in the development cycle. While within-project prediction has been widely studied, cross-project defect prediction (CPDP) remains challenging due to differences in datasets, high feature dimensionality, and poor model generalization. To address these challenges, this study enhances the Transformation and Feature Selection (TFS) approach by integrating ensemble learning techniques. Three methods, Gradient Boosting Machine (GBM), stacking, and hybridization, were explored to evaluate their effectiveness in improving CPDP performance. Experiments were conducted using the AEEEM datasets, with preprocessing steps including normalization, feature reduction, and the Synthetic Minority Oversampling Technique (SMOTE) to handle data imbalance. The models were trained on source projects and tested on separate target projects, with the F1 score used as the main evaluation metric. Results show that the TFS × Stacking model achieved the highest overall performance, with a mean F1 score of 0.963, outperforming both TFS × GBM (0.958) and TFS × Hybridization (0.920). Compared to the original TFS × Random Forest method, the stacking approach consistently provided significant improvements across all project pairs. These findings highlight the potential of combining TFS with ensemble learning to enhance defect prediction in projects with limited or no historical data. This work not only advances CPDP research but also offers practical value to software teams by enabling more accurate identification of defect-prone modules and better allocation of testing resources.**

Keywords*—Software; defect prediction; cross-project; ensemble learning; feature selection

I. Introduction

Delivering high-quality software is one of the biggest challenges in software engineering [1]. A huge amount of time and money goes into testing and debugging, yet defects often slip through and cause failures in real-world systems [2]. These failures can be costly, damage user trust, and reduce the overall reliability of the software. To address this, researchers and practitioners have turned to software defect prediction (SDP), which uses historical project data and code metrics to identify parts of the system that are most likely to contain bugs. By highlighting defect-prone modules early, teams can focus their testing and maintenance where it matters most, saving both time and resources.

Traditional SDP models are usually built with supervised machine learning techniques that learn from past projects [3-4]. They look for patterns in features such as lines of code, complexity, and other process metrics that often signal potential defects. While these models have proven useful, they depend heavily on having high-quality labeled data from the project being developed. Unfortunately, many new projects lack this data, making it difficult to train reliable models.

This is where cross-project defect prediction (CPDP) becomes important. CPDP allows models to be trained using data from other projects and then applied to a new project with little or no labeled data [5]. In theory, this makes defect prediction more widely usable. In practice, however, CPDP faces major hurdles: the source and target projects often differ in their features and distributions, data is high-dimensional, and models trained on one project often perform poorly when applied to another.

To tackle these problems, researchers have explored techniques such as feature transformation and feature selection. Transformation methods reduce differences between projects, while feature selection eliminates irrelevant or redundant data to make models simpler and more accurate. The Transformation and Feature Selection (TFS) approach has shown promise in this regard, but it still struggles with complex, non-linear patterns and imbalanced datasets.

Recently, ensemble learning has emerged as a powerful way to improve prediction models. Instead of relying on a single learner, ensemble methods combine multiple models to produce more accurate and robust results [6-7]. Gradient Boosting Machine (GBM), stacking, and hybridization are three such techniques that have been particularly effective. GBM is strong at capturing complex relationships and handling imbalanced data; stacking leverages the strengths of diverse models through a meta-learner; and hybridization blends models to capture both linear and non-linear dependencies.

Building on these observations, this study proposes an improved CPDP framework by integrating TFS with GBM, stacking, and hybridization. The goal is to investigate whether combining these methods can lead to better generalization across heterogeneous projects. The F1 score is used to measure predictive performance, with the goal of identifying which

ensemble approach works best with TFS for cross-project defect prediction.

By doing so, this study not only contributes to the academic field of defect prediction but also offers practical benefits. Development teams working on new projects, especially those with little labeled data, can use these enhanced models to predict software defects more reliably and make better decisions about where to direct testing efforts.

II. Related Work

Software defect prediction (SDP) is widely recognized as an important technique for improving software quality, as it helps identify components that are likely to contain bugs before a system is released [8]. Traditional SDP approaches largely depend on within-project defect prediction (WPDP), where a model is trained and tested using data from the same project. However, WPDP requires a large amount of labeled defect data, something that new or early-stage projects typically do not have [9]. This limitation has encouraged researchers to explore cross-project defect prediction (CPDP), where models trained on one project are applied to another project that has little or no historical defect data [10].

Over the years, CPDP research has proposed various strategies to address challenges such as data inconsistency, high dimensionality, and differences in metric distributions across projects. However, many studies address these issues in isolation, focusing on either feature transformation, feature selection, or ensemble learning without examining how these techniques can be combined and jointly optimized to improve cross-project transfer performance.

A. Feature Transformation Approaches

Feature transformation has been widely used to reduce the differences between source and target datasets. For example, Zhao et al. [11] introduced a manifold-based transformation method that projects data into a shared feature space, increasing similarity between datasets and improving prediction performance. Although their work showed the value of aligning data distributions, it did not consider how ensemble learning or feature selection could be integrated to further reduce noise and redundancy.

Similarly, Shekhawat et al. [12] proposed the Binary SALp Swarm Algorithm (BSSA), which performs both data transformation and feature selection. While the method improved computational efficiency, it was mainly evaluated in WPDP settings and did not examine its potential in CPDP contexts, where the domain differences are more substantial.

B. Feature Selection Methods

Feature selection techniques reduce dimensionality by eliminating irrelevant or redundant metrics. Ali et al. [13] compared several feature selection methods and demonstrated that removing unstable features can significantly improve prediction accuracy. However, their analysis treated feature selection as a standalone step and did not explore its interaction with feature transformation or ensemble learning.

Pandey et al. [14] proposed a hybrid method that combines data transformation with the Binary Binomial Cuckoo Search algorithm for feature selection. Although effective, the approach emphasized heuristic optimization and did not investigate ensemble-based improvements or its suitability for cross-project scenarios.

C. Ensemble Learning in Defect Prediction

Ensemble learning methods such as Random Forest, Gradient Boosting, and stacking have become increasingly popular due to their robustness and ability to capture complex, nonlinear patterns in defect datasets. Matloob et al. [15] conducted a systematic review showing the advantages of ensemble learning, particularly improvements in recall and performance on imbalanced data. However, their work did not examine how ensemble techniques might interact with feature transformation in CPDP environments.

Tang et al. [16] introduced an adaptive ensemble algorithm based on the Sparrow Search Algorithm, showing strong results on imbalanced datasets. Nonetheless, their method required extensive dataset-specific tuning, limiting its generalizability across projects.

Hybrid approaches that combine gradient boosting and deep learning models, such as those examined by [17], [18], and [19], have also demonstrated improved accuracy by leveraging both linear and nonlinear learners. Yet, these approaches do not incorporate feature selection specifically tailored for cross-project prediction, leaving a gap in understanding how transformation, feature reduction, and ensemble learning can work together within CPDP.

D. Gap in Existing Literature

From the reviewed literature, a clear gap emerges: most CPDP studies evaluate feature transformation, feature selection, and ensemble learning separately. There is no unified framework that systematically integrates these techniques or examines how their combination provides conceptual or technical benefits beyond existing hybrid models.

E. Contribution of this Study

This study addresses that gap by proposing an integrated framework that combines Transformation and Feature Selection (TFS) with multiple ensemble learning strategies, including Gradient Boosting Machines (GBM), stacking, and hybrid ensemble models, and evaluates their cross-project transferability using diverse datasets. The contribution lies not only in the integration itself but also in analyzing how each ensemble approach utilizes TFS outputs to improve robustness against domain divergence in CPDP.

III. Methodology

The methodology focuses on improving the Transformation and Feature Selection (TFS) approach proposed [7] by integrating advanced machine learning ensemble techniques for cross-project defect prediction (CPDP). Specifically, this research evaluates three ensemble methods, such as stacking, hybridization, and Gradient Boosting Machine (GBM), to determine which approach best enhances the performance of TFS.

The main challenges addressed include high-dimensional feature spaces, poor generalization across projects, and

imbalanced datasets. Our proposed approach mitigates these limitations by applying feature transformation, feature selection, and ensemble learning techniques.

In the proposed solution, the performance of the TFS model is enhanced through the application of ensemble learning. Three methods are implemented for comparison:

1) Stacking Model, which combines predictions from multiple base learners using a meta-learner.

2) Hybridization Model, which integrates different models to capture both linear and nonlinear patterns.

3) Gradient Boosting Machine (GBM), which builds sequential decision trees that iteratively correct errors.

The most widely adopted evaluation metric, the F1 score, is used to assess model performance due to its effectiveness in handling imbalanced datasets. The experiments are conducted to identify the best-performing ensemble method when combined with TFS for CPDP.

B. Datasets

The AEEEM datasets are some of the open-source datasets commonly used in SDP studies [3]. Table I shows the datasets that contain features gathered from previous software projects, which represent various aspects of software modules, such as lines of code, complexity measures, and fault labels. All the datasets contain defective data from a few software projects, labeled as either "Defective" or "Non-Defective".

TABLE I. CHARACTERISTICS OF DATASETS

Project Name	No. Of Module	No. Of Features	No. Of Defect	Defect Ratio
EQ	324	61	129	39.81%
JDT	997	61	206	20.66%
LC	691	61	64	9.26%
ML	1862	61	245	13.16%
PDE	1497	61	209	13.96%

C. Feature Selection

There are 61 features across all datasets. Only the top ten features, determined by significance scores, were chosen for training before the evaluation to reduce dimensionality and improve model interpretability. The mean and standard deviation of feature values were used to get the relative relevance score. Calculate the average and standard deviation for each characteristic. Determine the difference between the feature mean and the overall dataset mean. Order the features according to their deviation scores or other computed metrics, with lower deviations suggesting greater importance. The relevant features are extracted from every dataset.

D. Evaluation Metrics

The F1 score is the main evaluation metric for machine learning models in this study. This metric is especially useful for imbalanced datasets since it provides a balanced evaluation by merging precision and recall into a harmonic mean [20-21]. By accounting for both false positives and false negatives, the F1 score ensures that the model's prediction capabilities are thoroughly evaluated. The F1 score, which is commonly employed in binary and multi-class classification problems, combines precision and recall, providing a reliable measure of the model's overall performance, making it an important criterion for this study.

Precision: An indicator of a machine learning model's performance – the accuracy of its positive predictions.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

Recall: A quantitative measure that assesses the frequency with which a machine learning model accurately detects positive occurrences.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

F1-score: A measure of the harmonic mean of precision and recall.

$$\text{F1\_score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

E. Implementation

The implementation of the proposed approach consists of four major phases: data preprocessing, feature transformation and selection (TFS), model training, and evaluation, as shown in Fig. 1. First, datasets from multiple software projects (AEEEM) were collected and preprocessed to ensure consistency. Preprocessing included handling missing values, normalizing feature values, and addressing class imbalance using the Synthetic Minority Oversampling Technique (SMOTE).

Next, the TFS was applied to reduce dimensionality and improve the quality of the feature space. Feature scaling and selection techniques were used to identify the most relevant metrics, allowing the models to focus on the most influential predictors.

Three ensemble learning techniques were then implemented to enhance the performance of TFS for cross-project defect prediction (CPDP):

1) TFS × GBM: Gradient Boosting Machine was integrated with TFSM to capture complex non-linear relationships by iteratively correcting errors through sequential decision trees.

2) TFS × Stacking: Multiple base learners (e.g., Random Forest and GBM) were combined, with a Logistic Regression meta-learner integrating their predictions to optimize decision boundaries.

3) TFS × Hybridization: A hybrid voting-based approach was used to integrate Random Forest and GBM, capturing both linear and non-linear dependencies for more balanced performance.

The models were trained on source projects and tested on distinct target projects to simulate realistic CPDP scenarios. For each source–target pair, the F1 score was computed as the primary performance metric, as it balances precision and recall and is particularly suitable for imbalanced datasets. Comparative experiments across stacking, hybridization, and GBM identified the best-performing ensemble method when integrated with TFS.

F. Data Preprocessing

Preprocessing the dataset is an important step for ensuring the GBM model's quality and applicability [22]. Preprocessing tasks included correcting class imbalances, scaling numerical characteristics, encoding categorical variables, and handling missing data. Under- and oversampling techniques, notably the Synthetic Minority Oversampling Technique (SMOTE), were utilized to address class imbalances induced by skewed distributions of defect labels. These metrics were required for the model to efficiently analyze the dataset and provide reliable training and testing.

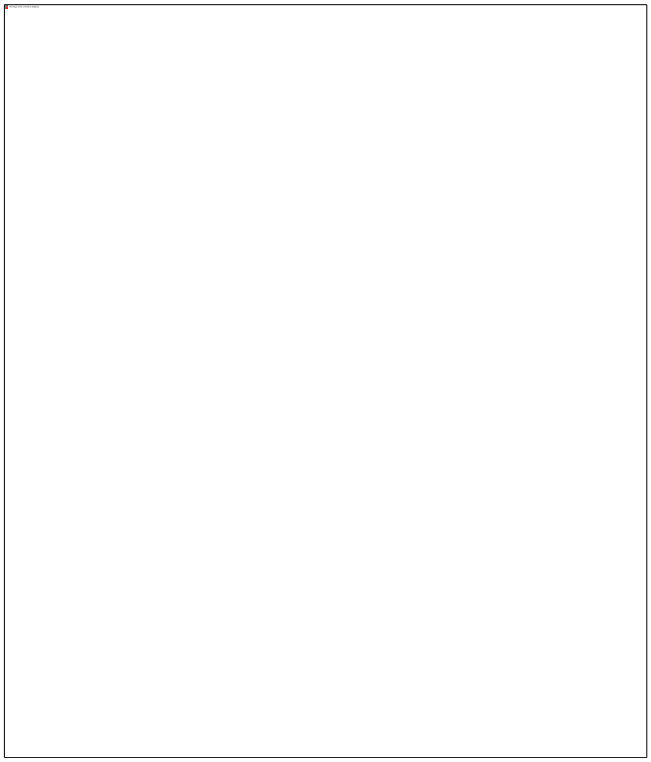

Fig. 1. Cross-project defect prediction workflow.

IV. Results and Discussion

This section describes the outcomes of integrating TFS and GBM in cross-project defect prediction settings. The stacking model combines the predictive power of Gradient Boosting and Random Forest as base learners, with Logistic Regression serving as the meta-model. This section evaluates the model's performance using its F1 score, which is a weighted statistic that accounts for class imbalances in the dataset. The results are analyzed to evaluate how efficient the stacking model method is in comparison to other methods.

A. Evaluation of TFS x GBM, TFS x Stacking and TFS x Hybridization

Table II compares the performance of three ensemble approaches (Stacking Model, Sequential Model, and Hybridization Model) using F1 scores across multiple cross-project software defect prediction scenarios. The models were evaluated by training on one source project (e.g., EQ) and testing on another target project (e.g., JDT).

TABLE II. Comparison of Three Ensemble Methods

Source > Target	Stack Model	Sequentially	Model Hybridization
EQ > JDT	0.85	0.91	0.90
EQ > ML	0.88	0.92	0.81
EQ > PDE	0.85	0.86	0.82
EQ > LC	0.94	0.92	0.90
JDT > EQ	0.97	0.95	0.90
JDT > LC	0.95	0.94	0.91
JDT > ML	0.97	0.95	0.91
JDT > PDE	0.98	0.94	0.91
LC > EQ	0.99	1.00	1.00
LC > JDT	0.98	1.00	0.98
LC > ML	0.99	0.99	0.98
LC > PDE	0.99	0.99	1.00
ML > EQ	0.98	0.96	0.91
ML > JDT	0.98	0.96	0.91
ML > LC	0.98	0.97	0.92
ML > PDE	0.98	0.94	0.90
PDE > EQ	1.00	1.00	0.93
PDE > JDT	1.00	0.98	0.96
PDE > LC	1.00	0.99	0.91
PDE > ML	1.00	0.98	0.94
Mean	0.963	0.958	0.920

The Stacking Model achieves the highest mean F1 score (0.963), indicating its robustness and generalizability across different project pairs. It performs exceptionally well in scenarios such as PDE > EQ, PDE > LC, and PDE > ML, with perfect F1 scores of 1.00. This demonstrates that combining multiple models through stacking improves the utilization of decision boundaries, resulting in higher prediction accuracy.

The Sequential Model follows closely with an average F1 score of 0.958. In some cases, it even outperforms the Stacking Model, such as LC > EQ and LC > JDT, where it achieves flawless F1 scores (1.00). This approach leverages the probabilistic outputs of one model (Random Forest) as inputs to another (Gradient Boosting), which proves highly effective in certain scenarios.

The Hybridization Model lags slightly behind, with an average F1 score of 0.920. While it achieves reasonable results overall, its performance is less consistent. For example, lower F1 scores in scenarios such as EQ > ML (0.81) and EQ > PDE (0.82) highlight its limited ability to generalize across different project pairs.

The results demonstrate that the Stack Model regularly has the greatest mean F1 score (0.963), exceeding both the Sequential Model (0.958) and the Hybridization Model (0.920). However, the Sequential Model performs competitively, occasionally outperforming the Stack Model in select circumstances, such as LC > EQ and LC > JDT. The Hybridization Model consistently underperforms the other two,

implying that it may be less effective in handling cross-project defect prediction jobs. Overall, the Stack Model shows higher robustness in all cases

B. Evaluation of TFS x Stacking Against Original Method

From Table III below, we can see that the TFS x Stacking Model consistently outperforms the TFS x RF Model across all source-target pairs, with a mean F1 score of 0.963 for the stacking model and 0.791 for the RF. Additionally, the stacking model performs well in pairs like EQ > LC and LC > EQ, with F1 scores of 0.94 and 0.99, respectively, compared to RF's 0.5 and 0.72. This demonstrates a significant improvement, particularly for specific source-target pairings. Furthermore, the PDE > EQ and PDE > JDT couples have flawless F1 scores of 1.00, demonstrating the outstanding performance of the TFS x GBM model.

The stacking detects complicated patterns and relationships in data, whereas the Random Forest ensures robustness and reduces overfitting. By merging various models, the stacking approach minimizes bias and instability, resulting in the improvement of prediction performance. Particularly, the stack model performs better than the stand-alone model, RF in combinations such as EQ > LC (0.94 vs. 0.5) and PDE > LC (1.00 vs. 0.87), implying that stacking better manages data dependencies and correlations. Furthermore, the stacked model performs consistently well across several domains, with high F1 scores in combinations such as LC > ML, ML > PDE, and JDT > ML, demonstrating its great ability to generalize and robustness in defect prediction across various software projects.

TABLE III. COMPARISON BETWEEN ENHANCED AND ORIGINAL MODEL

Source > Target	TFSM x Stacking	TFSM x RF
EQ > JDT	0.85	0.71
EQ > ML	0.88	0.81
EQ > PDE	0.85	0.65
EQ > LC	0.94	0.5
JDT > EQ	0.97	0.71
JDT > LC	0.95	0.74
JDT > ML	0.97	0.81
JDT > PDE	0.98	0.85
LC > EQ	0.99	0.72
LC > JDT	0.98	0.86
LC > ML	0.99	0.85
LC > PDE	0.99	0.87
ML > EQ	0.98	0.74
ML > JDT	0.98	0.86
ML > LC	0.98	0.89
ML > PDE	0.98	0.87
PDE > EQ	1.00	0.74
PDE > JDT	1.00	0.85
PDE > LC	1.00	0.92
PDE > ML	1.00	0.86
Mean	0.963	0.791

C. Advantages Over Previous Approaches

A major strength of this study lies in the integrated TFS + Ensemble framework, which performs better than previous CPDP methods evaluated in earlier studies. Unlike earlier CPDP approaches that applied feature transformation, feature selection, or ensemble techniques independently, the proposed method unifies these processes to more comprehensively address dataset heterogeneity. This integrated structure reduces noise, improves feature stability, and enhances the ability of ensemble models to learn cross-project patterns.

V. CONCLUSION

In this study, we set out to improve cross-project software defect prediction (CPDP) by combining Transformation and Feature Selection (TFS) with three ensemble learning techniques: Gradient Boosting Machine (GBM), stacking, and hybridization. The results show that using ensemble learning with TFS makes defect prediction more accurate and reliable by reducing issues such as high dimensionality, imbalance in the datasets, and weak generalization across projects. Of the three approaches, the TFS × Stacking model performed best, achieving the highest mean F1 score (0.963). This confirms that stacking, which combines the strengths of different base learners through a meta-learner, is particularly effective for capturing complex defect patterns. GBM and hybridization also showed good performance, but stacking consistently provided more robust results. When compared to the original TFS × Random Forest method, the ensemble-based models, especially stacking, showed significant improvements across all project combinations.

In real-world settings where labeled data is scarce or unavailable, common in early project phases, the proposed approach provides a practically viable defect prediction mechanism.

Despite its promising results, the study has several limitations. First, the experiments are limited to five datasets from the AEEEM repository, which may not represent the full diversity of real-world software systems. Second, the feature selection method used is relatively simple and may not capture deeper relationships among metrics. Third, the ensemble learners used were standard implementations; more advanced or domain-specific variations were not explored. Finally, the study does not evaluate the computational cost of the integrated pipeline, which may be a concern in large-scale industrial settings

Looking ahead, future research could build on this work by incorporating transfer learning or domain adaptation techniques to reduce distributional discrepancies more systematically. Another potential area is the integration of deep learning models or graph-based neural architectures to capture structural relationships within software components. Additionally, more sophisticated feature selection or feature synthesis techniques, such as metaheuristic optimization or automated feature engineering, could be integrated into the TFS pipeline.

ACKNOWLEDGMENT

This work was supported by Universiti Putra Malaysia.

REFERENCES

[1] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, "Software defect prediction analysis using machine learning techniques," Sustainability, vol. 15, no. 6, p. 5517, 2023, doi: 10.3390/su15065517.

[2] F. Matloob et al., "Software defect prediction using ensemble learning: A systematic literature review," IEEE Access, vol. 9, pp. 98754–98771, 2021, doi: 10.1109/ACCESS.2021.3095559.

[3] S. Angamuthu and K. G. Maheswari, "A comprehensive review of SDP using machine and deep learning techniques in software testing," in Proc. 2024 2nd Int. Conf. Computing Data Anal. (ICCDA), 2024, pp. 1–6, doi: 10.1109/ICCDA64887.2024.10867332.

[4] M. Nevendra and P. Singh, "A survey of software defect prediction based on deep learning," Arch. Comput. Methods Eng., vol. 29, no. 7, pp. 5723–5748, 2022.

[5] M. S. Saeed and M. Saleem, "Cross project software defect prediction using machine learning: A review," Int. J. Comput. Innov. Sci., vol. 2, no. 3, pp. 35–52, 2023.

[6] B. Gezici and A. Tarhan, "Explainable AI for software defect prediction with gradient boosting classifier," in Proc. 2022 7th Int. Conf. Comput. Sci. Eng. (UBMK), 2022, doi: 10.1109/UBMK55850.2022.9919490.

[7] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Improving cross-project software defect prediction method through transformation and feature selection approach," IEEE Access, vol. 11, pp. 2318–2326, 2023, doi: 10.1109/ACCESS.2022.3231456.

[8] N. Rane, S. P. Choudhary, and J. Rane, "Ensemble deep learning and machine learning: Applications, opportunities, challenges, and future directions," Stud. Med. Health Sci., vol. 1, no. 2, pp. 18–41, 2024.

[9] M. Ali et al., "Analysis of feature selection methods in software defect prediction models," IEEE Access, vol. 11, pp. 145954–145974, 2023, doi: 10.1109/ACCESS.2023.3343249.

[10] R. Malhotra and S. Das, "Exploring advanced techniques for software defect prediction: A comprehensive review," in Proc. 2024 1st Int. Conf. Adv. Comput., Commun. Netw. (ICAC2N), 2024, pp. 175–182.

[11] Y. Zhao, Y. Zhu, Y. Qiao, and X. Chen, "Cross-project defect prediction method based on manifold feature transformation," Future Internet, vol. 13, no. 8, p. 216, 2021, doi: 10.3390/fi13080216.

[12] S. S. Shekhawat, H. Sharma, S. Kumar, A. Nayyar, and B. Qureshi, "BSSA: Binary SALp Swarm algorithm with hybrid data transformation for feature selection," IEEE Access, vol. 9, pp. 14867–14882, 2021, doi: 10.1109/ACCESS.2021.3049547.

[13] M. Z. Ali et al., "Advances and challenges in feature selection methods: A comprehensive review," J. Artif. Intell. Metaheuristics, vol. 7, no. 1, pp. 67–77, 2024.

[14] A. C. Pandey, D. S. Rajpoot, and M. Saraswat, "Feature selection method based on hybrid data transformation and binary binomial cuckoo search," J. Ambient Intell. Humaniz. Comput., vol. 11, pp. 719–738, 2020, doi: 10.1007/s12652-019-01330-1.

[15] F. Matloob, T. M. Ghazal, N. Taleb, and S. Aftab, "Software defect prediction using ensemble learning: A systematic literature review," IEEE Access, vol. 9, pp. 98754–98771, 2021, doi: 10.1109/ACCESS.2021.3095559.

[16] Y. Tang, Q. Dai, M. Yang, T. Du, and L. Chen, "Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm," Int. J. Mach. Learn. Cybern., vol. 14, no. 6, pp. 1967–1987, 2023, doi: 10.1007/s13042-022-01740-2.

[17] S. Gupta, "A hybrid machine learning framework of gradient boosting decision tree and sequence model for predicting escalation in customer support," in Proc. 2020 IEEE Int. Conf. Big Data, 2020, doi: 10.1109/BigData50022.2020.9377831.

[18] A. Abdu et al., "Semantic and traditional feature fusion for software defect prediction using hybrid deep learning model," Sci. Rep., vol. 14, no. 1, p. 14771, 2024.

[19] A. M. Akbar, R. Herteno, S. W. Saputro, M. R. Faisal, and R. A. Nugroho, "Optimizing software defect prediction models: Integrating hybrid grey wolf and particle swarm optimization for enhanced feature selection with gradient boosting algorithm," J. Electron. Electromed. Eng. Med. Informatics, vol. 6, no. 2, pp. 169–181, 2024.

[20] S. Pal and A. Sillitti, "Cross-project defect prediction: A literature review," IEEE Access, vol. 10, pp. 118697–118717, 2022.

[21] W. Wen et al., "Cross-project software defect prediction based on class code similarity," IEEE Access, vol. 10, pp. 105485–105495, 2022.

[22] E. Daniel and G. R. Jainish, "Boosting software quality: A machine learning technique for predicting defects," in Proc. 2025 6th Int. Conf. Mobile Comput. Sustain. Informatics (ICMCSI), 2025, pp. 1062–1070.