

Multi-Source Transfer Weighted Learning with Random Forest for Cross-Project Software Defect Prediction

¹Kummarikunta Sandhya, ²P. Radhika Raju, ³A. Anandarao

¹Research Scholar, Dept. of CSE, Jawaharlal Nehru Technological University Anantapur, Ananthapuramu, Andhra Pradesh, India.

²Assistant Professor, Dept. Of CSE, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India.

³Professor, Dept. Of CSE, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India.

¹sandhyachurchil@gmail.com, ORCID ID: [0009-0001-3542-5055](https://orcid.org/0009-0001-3542-5055)

²radhikaraju.p@gmail.com, ORCID ID: [0000-0001-8909-3079](https://orcid.org/0000-0001-8909-3079)

³akepogu@gmail.com, ORCID ID: [0009-0007-7132-1767](https://orcid.org/0009-0007-7132-1767)

**Corresponding Author: Kummarikunta Sandhya*

Abstract: Accurate software defect prediction is essential for identifying fault-prone modules early and reducing software testing and maintenance costs. However, Within-Project Defect Prediction (WPDP) models may perform poorly when the target project has limited labeled defect data. Cross-Project Defect Prediction (CPDP) addresses this limitation by transferring knowledge from other projects, but heterogeneous feature distributions, source–target mismatch, and negative transfer remain significant challenges. This study proposes a Multi-Source Transfer Weighted Learning with Random Forest (MS-TWL+RF) framework for software defect prediction across heterogeneous projects. The proposed framework combines transferability estimation at the dataset, feature, and instance levels. Linear Maximum Mean Discrepancy (MMD) assigns weights to source projects based on their distributional similarity to the target project. In contrast, feature weights jointly consider distribution similarity and mutual information-based predictive relevance. A domain classifier is used to assign instance-level weights according to their representativeness of the target domain. Correlation Alignment (CORAL) subsequently reduces covariance discrepancies between the source and target feature spaces before classification using a weighted Random Forest model. The proposed framework is evaluated on the JM1, KC1, CM1, KC2, and PC1 projects and compared with AlexNet+RF, ResNet50+RF, TransTab+RF, and XTab+RF. For the JM1 target project, the proposed method achieves 98.7% target-domain accuracy, 98.9% precision, 98.7% recall, and 98.8% F1-score, with an AUC of 0.992. It also obtains Kappa, MCC, and Jaccard scores of 0.980, 0.981, and 0.990, respectively. The results indicate that combining multi-level transfer weighting with covariance alignment can help address the effects of heterogeneous source projects in cross-project software defect prediction.

Keywords: Software defect prediction, cross-project defect prediction, multi-source CPDP, transfer learning, Random Forest, transfer weighting.

1 INTRODUCTION

The increasing complexity of software systems, together with the rapid pace of software development, has increased the importance of accurate defect prediction in software engineering [1]. The early identification of fault-prone modules before deployment is important for maintaining software reliability and reducing testing and maintenance costs [1]. WPDP can be effective for mature projects with rich histories of labeled defect data; however, it may perform poorly for projects in the early stages of development or projects with limited historical defect data due to poor maintenance practices [2].

To address these shortcomings, the research community has increasingly adopted CPDP, which transfers knowledge from source projects to predict defects in a target project [3][4]. CPDP has been shown to improve defect prediction across projects; however, single-source models may suffer performance degradation due to distributional differences, feature incompatibility, and negative transfer when the source project poorly matches the target project. As a result, recent research has turned attention to Multi-Source CPDP (MSCPDP), which uses multiple heterogeneous source projects to improve generalization and robustness [5]. However, MSCPDP introduces additional challenges in practice because source projects may differ not only from the target project but also from one another [5]. These differences can lead to feature distribution heterogeneity, differences in predictive relevance, and discrepancies in defect annotations.

Existing MSCPDP methods have addressed these problems using techniques such as feature selection, feature-level normalization, and domain adaptation [2]. However, these approaches often do not systematically consider transferability at different levels of granularity. Methods that do not explicitly distinguish between dataset weighting, feature relevance, and instance representativeness may cause models to overemphasize spurious sources or underemphasize informative ones. Moreover, deep neural architectures such as AlexNet and ResNet50, as well as transformer-based tabular models such as TransTab and XTab, have been applied to software defect prediction to learn complex representations [6][7].

Motivated by these problems, this study proposes a hybrid transfer learning and machine-learning framework that introduces a Multi-Source Transfer Weighted Learning (MS-TWL) layer to estimate transferability at the dataset, feature, and instance levels. Source domains are weighted using MMD [8]; distribution similarity and predictive relevance are combined as a convex weighted combination to determine feature importance; instance relevance is estimated using a domain-specific classifier. Further, the feature spaces are aligned using CORAL [9] to reduce covariance discrepancies between the source and target domains. The aligned and reweighted features are then provided as inputs to a Random Forest classifier, which is suitable for handling noisy attributes, nonlinear relationships, and weighted samples. The objectives of this research are defined as follows:

- Develop a multi-level transfer-learning framework for multi-source CPDP that accounts for heterogeneity among multiple source projects and an unlabeled target project.
- Develop transferability weighting mechanisms at the dataset, feature, and instance levels using MMD-based distributional similarity, feature distribution similarity with mutual-information-based predictive relevance, and domain-classifier-based instance representativeness.
- Integrate CORAL-based covariance alignment with the transfer weighting mechanisms and Random Forest classification to reduce source-target distribution discrepancies and improve cross-project defect prediction.
- Evaluate the proposed MS-TWL+RF framework using leave-one-project-out evaluation across the JM1, KC1, CM1, KC2, and PC1 projects and compare its performance with the selected AlexNet+RF, ResNet50+RF, TransTab+RF, and XTab+RF baseline models.

The contributions of this study can be summarized as follows:

- A multi-level transfer weighting framework is introduced for multi-source CPDP, integrating dataset-level, feature-level, and instance-level transferability to address heterogeneity among source and target projects.
- A combined weighting strategy is developed by integrating MMD-based dataset similarity, distributional similarity, and mutual-information-based feature relevance, and domain-classifier-based instance representativeness.
- CORAL-based covariance alignment is incorporated with the transfer weighting mechanism and Random Forest classification to reduce source-target distribution discrepancies and support defect prediction across heterogeneous projects.

2 LITERATURE REVIEW

Software defect prediction across different projects plays an important role in improving software quality and reducing software development and maintenance costs. As software systems become more complex, predicting defects across projects remains challenging because data distributions, software metrics, and defect characteristics differ. Tong Li et al. [10] proposed a model-averaged software defect prediction framework using four benchmark datasets, namely NASA, AEEEM, ReLink, and SoftLab. The framework integrates XGBoost and LightGBM, uses cross-validation to determine model weights by minimizing squared prediction errors, and then averages the predictions to improve stability.

Javed et al. [11] proposed a supervised domain-adaptive cross-project defect prediction model called SCAG-LSTM using the AEEEM and PROMISE datasets. The framework combines SMOTE-ENN for handling class imbalance, correlation-based feature selection with best-first search, and deep models including Bi-LSTM, Bi-GRU, and attention mechanisms. Bai et al. [12] proposed 3SW-MSTL using 30 open-source projects from PROMISE, ReLink, and AEEEM. The framework involves source selection, transfer learning to reduce distribution differences, and multi-source utilization through weighted voting. Their experiments reported an average AUC of 0.83 and G-mean of 0.79, with the method showing competitive performance against multi-source, single-source, and unsupervised CPDP methods, as well as WPDP.

Jiang et al. [13] proposed BATM for CPDP using benchmark datasets from PROMISE. The method aligns multimodal feature distributions through multilinear conditioning, improves pseudo-labels with an auxiliary classifier, and incorporates adversarial learning. The reported experiments achieved an average balanced accuracy of 0.714, with project-level scores of 0.873 for Ant, 0.806 for JEdit, 0.808 for Poi, and 0.845 for JDT. The authors also reported improved performance compared with the CPDP baselines considered in their study. Wang et al. [14] developed FedDP for privacy-preserving CPDP using 19 projects from the PROMISE and AEEEM datasets. The approach combines federated learning with local heterogeneity awareness and global knowledge distillation, using open-source data as a distillation dataset. Their experiments reported an AUC of 68.18% and an F1-measure of 49.14%, outperforming the FL and CPDP baselines in their study.

Song et al. [15] proposed ConCPDP based on the PROMISE dataset, combining contrastive learning-based pretraining with category boundary adjustment. The approach includes augmentation of source and target code, extraction of abstract syntax trees, Bi-LSTM with contrastive learning, and fine-tuning. The experiments reported improvements over the baseline, with AUC increasing from 0.551 to 0.603 and F1 increasing from 0.409 to 0.418. These studies show that recent CPDP research has increasingly moved beyond conventional single-source transfer approaches toward methods that combine representation learning, domain adaptation, and multiple source projects.

However, differences in feature distributions and the quality of transferred information remain important factors affecting CPDP performance. Zhao et al. [16] developed MSCPDP for multi-source CPDP using the AEEEM and PROMISE datasets. The model considers multiple data distributions simultaneously by integrating different source projects. The experiments reported median F1 improvements over six CPDP baselines of 3.51%, 3.92%, 36.06%, 0.49%, 17.05%, and 9.49%, respectively. The corresponding median changes in AUC were -3.42%, 8.78%, 0.96%, -2.21%, -7.94%, and 5.13%. Hu and Zhu [17] designed a framework for CPDP using genetic algorithm-based feature selection on AEEEM and PROMISE projects. The technique uses an adaptive GA for global feature subset search and ensemble training of EasyEnsemble with Naïve Bayes classifiers. The reported results showed improved F1-scores across the evaluated projects.

Tao et al. [18] designed TLSTM and DPTLSTM models that combine semantic features with transfer learning and applied them to PROMISE projects. Abstract syntax trees were translated into integer vectors and then processed using LSTM with transfer component analysis and logistic regression. TLSTM improved the F-measure by 0.7% on Log4j-1.2 and 2.1% on Xalan-2.7, while DPTLSTM improved the F-measure by 2.9% on Synapse-1.2 and 5.0% on Xerces-1.4.4. Abdu et al. [19] designed RH-CPDP, which used principal component analysis for feature reduction and hybrid deep neural networks for metric hybridization. RH-CPDP was applied to nine projects from PROMISE. The results reported higher AUC and F1 scores than TCSBoost, TPTL, DA-KTSVMO, DBN, and 3SW-MSTL across the evaluated tasks, with the Camel projects showing the best results using 14 and 12 PCA components. These studies show that feature transformation, feature selection, and deep representation learning can reduce the effect of distribution differences between source and target projects. Nevertheless, these approaches use different mechanisms to select or adapt transferable information, indicating the need for more systematic strategies to estimate the usefulness of individual source data and features.

Abdu et al. [20] designed GB-CPDP with graph-based feature learning and applied it to the PROMISE dataset. GB-CPDP constructs control-flow and data-dependency graphs and then uses Node2Vec embeddings and LSTM for defect prediction. Experimental results showed F1-score improvements of 2.7% to 6.3% and AUC improvements of 4.8% to 9.8% compared with state-of-the-art methods. The optimal embeddings achieved the highest F1-score of 0.652. Haque et al. [21] designed ENTL for HCPDP, which addresses projects with different software metrics. ENTL uses encoder networks and transfer learning along with pseudo-label augmentation to minimize the negative transfer effect and was applied to 16 datasets from four projects. ENTL outperformed four HCPDP baselines and one WPDP model, achieving a 15.01% improvement in F1-score and a 3.43% improvement in AUC over WPDP.

Zhang et al. [22] proposed Intra-domain Alignment-based Program Distribution Adaptation with Copulas (IAPCP) using the NASA, AEEEM, and PROMISE datasets. The approach incorporates intra-domain alignment and program-based distribution adaptation. The model achieved an F1-score of 0.72 and an AUC of 0.79, outperforming TCA and KMM with margins ranging from 12% to 14% on the AEEEM and PROMISE datasets. On NASA PC3, it achieved 81.6% accuracy and an AUC of 0.84, whereas on the AEEEM JDT dataset, it achieved 76.3% accuracy and an F1-score of 0.82. Zhou et al. [2] introduced an improved TrAdaBoost method with feature transformation and evaluated it on 12 PROMISE projects. Their approach achieved an F1-score of 0.71 and an AUC of 0.77, outperforming TCA+ (F1-score 0.59, AUC 0.70) and KNN (F1-score 0.56, AUC 0.67). For the Ant-1.7 project, the method achieved 74.2% accuracy, 0.73 recall, and 0.70 precision; for Poi-3.0, it achieved 81.5% accuracy and an F1-score of 0.76. Gains of 6%–18% in F1-score were observed across the evaluated projects.

Ardimento et al. [23] developed a deep TCN architecture with hierarchical attention for just-in-time defect prediction. The method was tested on six Java-based open-source systems using commit-level metrics and achieved 87.6% accuracy, 82.4% precision, 85.2% recall, and 83.7% F1-score. The reported results also showed improvements over CNN (F1-score 74.6), RNN (F1-score 75.9), and LSTM (F1-score 77.2). While deep learning-based approaches have shown their ability to learn complex representations, multi-source CPDP methods have also received increasing attention because multiple source projects can provide complementary information for the target project. However, the effectiveness of such approaches depends on how they handle differences among source projects and their relationships with the target project.

Yao et al. [24] introduced MSHKM for multi-source heterogeneous CPDP and evaluated it across 28 datasets from NASA, SOFTLAB, ReLink, AEEEM, and MORPH. The model achieved an average F1-score of 0.81 and an AUC of 0.84 when using four source projects, outperforming the reported HDP results (F1-score 0.72, AUC 0.77). On the AEEEM JDT dataset, MSHKM achieved 78.9% accuracy, an F1-score of 0.83, and an AUC of 0.86; on NASA PC1, it reported 81.2% accuracy and an F1-score of 0.80. Li et al. [25] evaluated data filtering and transfer learning using four defect-density strategies across 11 PROMISE datasets. Using the CBS+ strategy with BDA and JDA, their study reported improvements in recall and other evaluation measures, together with an acceptable IFA below 6. On Camel, BDA+KNN achieved 72.4% accuracy and an F1-score of 0.74, while on Xalan, recall reached 0.79 with a Popt20 of 0.83. Lei et al. [26] proposed SDP-MTF, which integrates statistical and semantic features with TCA+ and deep adaptation networks, and evaluated it on six PROMISE projects. The model achieved F1-scores ranging from 0.74 to 0.82, with an average of 0.78, and AUC values ranging from 0.79 to 0.85. The reported results showed improvements of 8% to 15.2% compared with baselines such as DBN and TCA+. On Xerces, the model achieved an F1-score of 0.82 and an AUC of 0.84, while on Ant it achieved an F1-score of 0.77.

On AEEEM projects, Lei et al. [27] presented WCM-WTrA, which uses feature selection and a distance-weighted TrAdaBoost algorithm. The approach showed a 23% average improvement over TCA+, achieving an F1-score of 0.79 and an AUC of 0.81. On the ReLink ZXing project, it improved by 5%, achieving an F1-score of 0.74 and an AUC of 0.77. The multi-source variant of the approach achieved an F1-score of up to 0.83 on the combined projects. Zou et al. [28] introduced MSCPDPLab, a MATLAB toolbox containing nine state-of-the-art MSCPDP models, including Burak's Filter and HYDRA. The toolbox provides a uniform way to load data, train and test models, and evaluate their performance using 13 different performance measures. The toolbox was tested on the PROMISE and ReLink datasets and achieved an average F1-score of 0.78 and an AUC of 0.83, with models such as BurakMHD and eCPDP performing well across the evaluated datasets. The studies discussed above further highlight the diversity of CPDP strategies, including heterogeneous transfer learning, federated learning, feature selection, and ensemble-based approaches. These methods demonstrate that both the choice of transferable information and the learning strategy can influence prediction performance across projects.

Yamamoto et al. [29] proposed FLR (Federated Logistic Regression) for privacy-preserving CPDP using 25 open-source and closed-source projects. The proposed methodology uses federated learning to train Logistic Regression without sharing raw data. FLR outperformed the existing privacy-preserving technique LACE2, improving AUC and G-mean by 0.2783 and 0.2673, respectively. FLR achieved an F1-score of 0.81 and an AUC of 0.86, which were comparable to the reported results of the supervised technique. Wang et al. [30] proposed FedDPI, a federated learning-based CPDP method that uses differential perception and inheritance of private models, and evaluated it on 25 projects. The model achieved an average AUC improvement of 0.2783 and a G-mean improvement of 0.2673 compared with the evaluated deep learning and federated learning methods.

Mehmood et al. [31] proposed a feature selection method for improving defect prediction accuracy using machine learning on five NASA datasets (CM1, JM1, KC2, KC1, and PC1). The classifiers included Random Forest, Logistic Regression, and Support Vector Machine. The results showed that feature selection improved accuracy compared with models without feature selection, with Random Forest achieving 99.5% accuracy and an F1-score of 0.79. Pan et al. [32] proposed an enhanced CNN model for within-project defect prediction using the Simplified PROMISE Source Code dataset. The model achieved 88% accuracy, increasing to 94% with oversampling techniques. The proposed model also increased the F1-score by 43%.

Giray et al. [33] reviewed deep learning techniques for software defect prediction. The review found CNN to be the most commonly used deep learning technique among the approaches considered. The authors concluded that deep learning techniques, including CNN, can outperform conventional machine-learning techniques by automating feature extraction. Khleel and Nehéz [34] proposed the use of a bidirectional LSTM network along with oversampling techniques, including SMOTE and random oversampling, to address class imbalance in software defect prediction. The model was evaluated on the PROMISE dataset and achieved 94% accuracy, representing a 6% increase, together with a 43% improvement in F1-score.

Ali et al. [35] proposed an ensemble-based software defect prediction model consisting of four classifiers: Random Forest, Support Vector Machine, Naïve Bayes, and Artificial Neural Networks, using a voting ensemble technique. The proposed model was evaluated on seven datasets from the NASA MDP repository and achieved 99% accuracy and an F1-score of 98%, outperforming the 20 state-of-the-art software defect prediction techniques considered in the study. Chen et al. [36] proposed a Nested-Stacking framework with heterogeneous feature selection for software defect prediction. The framework combines feature selection and stacked learning to improve prediction performance. The model was tested on the Kamei and PROMISE datasets and improved F1-score by 11% and AUC by 0.07 over the baseline models. Bala et al. [37] proposed a transformation and feature selection method to address distribution discrepancies in CPDP. They used Random Forest as the classifier and evaluated the method on the AEEEM dataset. The results showed an average F1-score of 0.78 and an AUC of 0.83. Tang et al. [38] proposed an ensemble model, AVSEB, using the adaptive variable sparrow search algorithm for parameter optimization. The model was evaluated on 15 open software defect datasets, with the AVSEB model achieving a mean F-measure of 0.8415 and a mean MCC of 0.8344.

Overall, the reviewed studies use source selection, feature transformation, domain adaptation, deep learning, ensemble learning, and multi-source transfer strategies for cross-project software defect prediction. However, existing approaches generally focus on one or a limited combination of transfer mechanisms, such as source selection, feature alignment, or instance adaptation. Limited attention has been given to jointly modeling transferability at the dataset, feature, and instance levels while also reducing covariance differences between source and target feature spaces. This motivates the proposed MS-TWL+RF framework, which integrates multi-level transfer weighting with CORAL-based domain adaptation and Random Forest classification for multi-source CPDP.

3 METHODOLOGY

This section describes the approach taken in the proposed hybrid model, which combines MS-TWL with Random Forest for MSCPDP. This methodology addresses challenges associated with heterogeneous software projects, feature distribution mismatch, differences in source-target distributions, and limited labeled defect data in the target project.

The main goal is to transfer relevant knowledge from multiple source projects to a target project by estimating transferability at the dataset, feature, and instance levels. Fig. 1 presents a graphical representation of the proposed hybrid model workflow, illustrating the major stages of transfer weighting, feature alignment using CORAL, and defect prediction using the weighted Random Forest classifier.

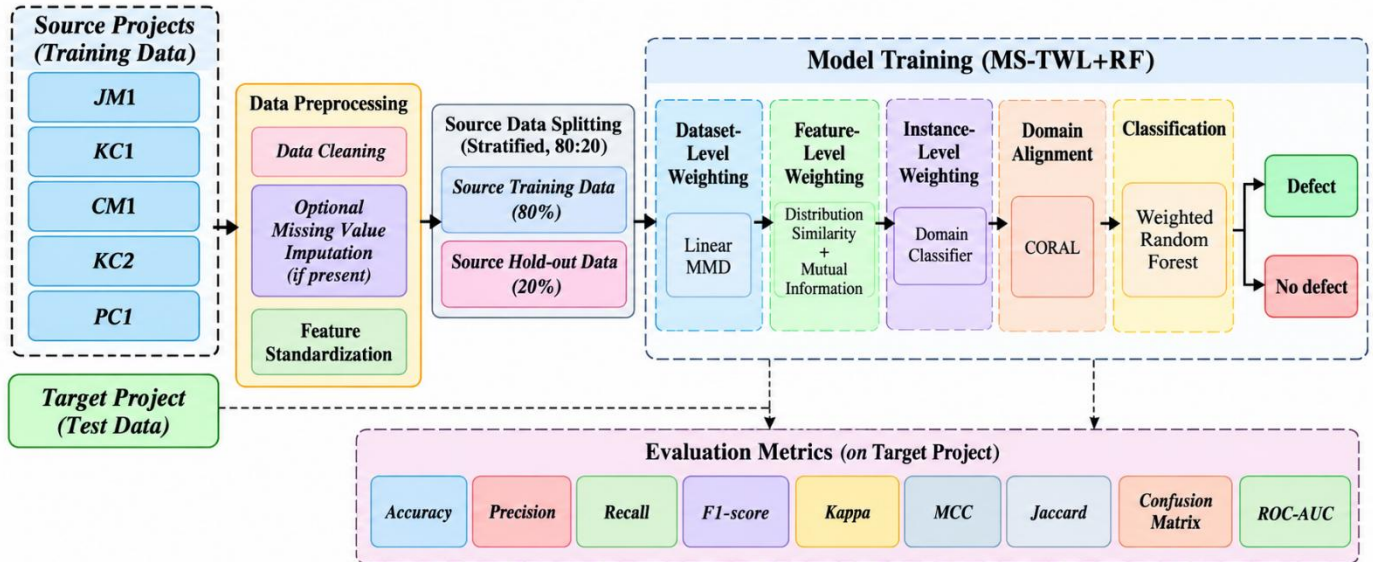


Fig. 1. Architecture of the proposed MS-TWL+RF framework for MSCPDP.

3.1. Dataset Description

In this paper, the empirical study is performed on five program defect prediction datasets, namely JM1, KC1, CM1, KC2, and PC1, obtained from the PROMISE Repository of Software Engineering Databases [39]. Table 1 describes the datasets. The dataset contains static software measures and binary defect status and is commonly used for empirical research in software defect prediction. The datasets describe each software module using 21 software metrics that represent different aspects of software size, complexity, Halstead measures, and program structure. These include lines of code, cyclomatic complexity, essential complexity, design complexity, Halstead measures, code/comment measures, operator and operand measures, and branch count. The JM1 dataset contains 10,885 software modules with static software measures and binary defect status, with a ratio of 2,106 defective to 8,779 non-defective modules. This imbalance reflects the class imbalance common in software defect datasets and can cause predictive models to favor the majority class. The dependent variable, defects, is a binary grouping indicator, where a module is classified as defective (True) or non-defective (False). The experiments follow a leave-one-project-out multi-source CPDP setting.

In each experiment, one project is the target, and the remaining four projects serve as sources. The five target configurations are JM1 with KC1, CM1, KC2, and PC1 as sources; KC1 with JM1, CM1, KC2, and PC1 as sources; CM1 with JM1, KC1, KC2, and PC1 as sources; KC2 with JM1, KC1, CM1, and PC1 as sources; and PC1 with JM1, KC1, CM1, and KC2 as sources. The five datasets provide different software characteristics and class distributions, making them suitable for evaluating transfer learning and CPDP methods. Using multiple source projects also helps examine the effect of heterogeneous source-target distributions in the proposed MSCPDP framework. The target project's defect labels are not used for model training, while the target feature values are used for transfer weighting and domain adaptation.

Table 1. Description of the datasets used in the study.

Dataset	Language	System / Project Description	Modules	Defective	Non-defective	Defect Rate (%)	Predictor Metrics
CM1	C	NASA spacecraft instrument processing system	498	49	449	9.84	21
JM1	C	Real-time ground system	10,885	2,106	8,779	19.35	21
KC1	C++	Storage management subsystem for a ground receiving station	2,109	326	1,783	15.46	21
KC2	C++	Science data processing component	522	107	415	20.5	21
PC1	C	Flight software for an Earth-orbiting satellite	1,109	77	1,032	6.94	21

3.2. Data Preprocessing

Raw datasets tend to be heterogeneous in nature, vary in scale, and have uneven class distributions. Without adequate preprocessing, models tend to be biased, unstable, and less generalizable. In the JM1 dataset, for example, software metrics vary considerably in scale, while defective modules do not predominate (19.3%). Using data cleaning, normalization, class-imbalance handling, and feature selection, the datasets are transformed into a more uniform, standardized representation for defect prediction.

3.2.1. Data Cleaning

Most attributes in JM1 are numeric; however, some attributes, including the number of operators and the number of operands, were initially represented as objects. They were converted into numerical form for processing. The dataset contained no missing values. If missing values were present, they would have been imputed using the median, which is defined as:

$$x_j^{(i)} = \begin{cases} x_j^{(i)}, & \text{if observed} \\ \text{median}\{x_j\}, & \text{if missing} \end{cases} \quad (1)$$

where $x_j^{(i)}$ represents the value of the feature j for the i^{th} instance.

3.2.2. Feature Normalization

Feature normalization is fitted exclusively on the source training data. The resulting training-derived mean and standard deviation are retained and applied to validation and target-project observations without recalculation. Target-domain statistics are therefore not used for normalization. This procedure prevents distributional information from leaking into model development. As the features have values on different scales (e.g., loc may have values in the hundreds, whereas difficulty and design complexity may have relatively small values), all continuous variables were transformed to zero mean and unit variance using

$$z_j^{(i)} = \frac{x_j^{(i)} - \mu_j}{\sigma_j} \quad (2)$$

where μ_j and σ_j represent the mean and standard deviation of feature j . This step puts the attributes on a comparable scale during learning and prevents large-scale variables from dominating.

3.2.3. Class Imbalance Handling with SMOTE

The JM1 dataset has a strong class imbalance, with only 19.3% of modules annotated as defective. This imbalance may introduce bias into machine-learning models. To alleviate this bias, the minority (defective) class is oversampled using the Synthetic Minority Over-sampling Technique, commonly known as SMOTE. Synthetic examples are generated only from the source training data, thereby avoiding leakage into the validation and target evaluation data. SMOTE generates synthetic minority samples using neighboring minority instances, thereby increasing the representation of defective modules in the training data.

3.2.4. Feature Selection

The JM1 dataset contains 21 software metrics, some of which may be redundant or weakly correlated with defect proneness. Therefore, a feature-selection process was carried out to reduce redundancy and improve prediction performance. The aim was to keep the most discriminative measures and remove non-relevant or redundant features. Mutual information was used as the main feature-ranking criterion. Mutual-information feature selection and correlation-based filtering were performed using the source training data. The selected features are subsequently applied to the validation and target-project samples. After that, a correlation analysis was conducted to identify multicollinearity among software measures, including Halstead length, vocabulary, and volume. When two metrics had a high correlation coefficient ($r > 0.85$), the metric with the lower mutual information value was removed. This step helps reduce redundancy and retain predictive features. The resulting feature set reduces the number of attributes and improves computational efficiency while retaining information relevant to defect prediction.

3.3. Proposed Hybrid Model: Multi-Source Transfer Weighted Learning (MS-TWL) + Random Forest Classifier

To address feature incompatibility, distribution gaps, and the limited labeled data in MSCPDP, this work proposes a hybrid defect prediction model based on transfer learning and Random Forest classification. The architecture includes a transfer learning layer that aligns and reweights data across sources, followed by a Random Forest classifier that produces the final defect predictions. Fig. 2 illustrates the overall operation of the proposed hybrid model, where the transfer learning layer integrates dataset-, feature-, and instance-level weighting with CORAL alignment before the final defect prediction.

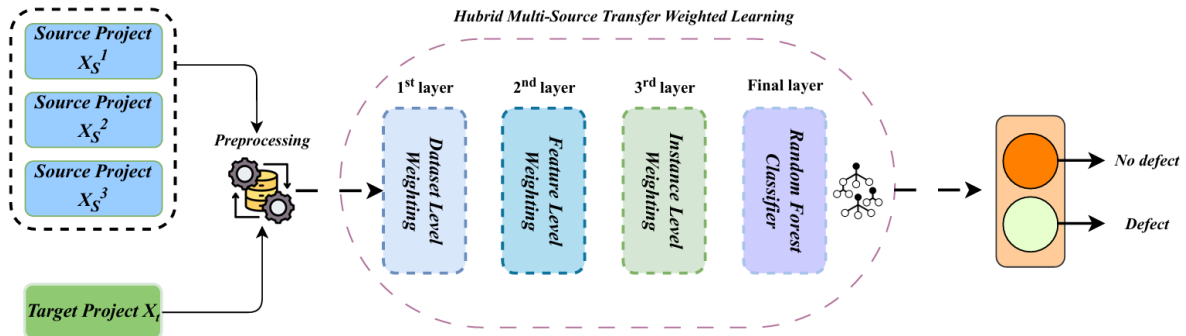


Fig. 2. Architecture of the proposed MS-TWL+RF framework for MSCPDP.

3.3.1. Dataset-Level Weighting

Software projects can vary greatly in terms of programming styles, module characteristics, metric scales, and historical quality practices. To assign different weights to the individual source projects, the distributional similarity of each source project to the target was calculated using MMD. Under the condition of a source dataset $X_S^{(k)}$ and the target dataset X_t , the MMD is calculated as:

$$MMD(X_S^{(k)}, X_t) = \|\mu(X_S^{(k)}) - \mu(X_t)\|_2^2 \quad (3)$$

where $\mu(\cdot)$ represents the mean of the corresponding feature representation. A linear kernel is used for MMD in all experiments, providing a simple measure of the distributional difference between the source and target representations after standardization. After calculating the MMD values for each source, the dissimilarities are converted into normalized transfer weights using a softmax transformation:

$$w_d^{(k)} = \frac{\exp\{-\alpha MMD(X_S^{(k)}, X_t)\}}{\sum_{j=1}^K \exp\{-\alpha MMD(X_S^{(j)}, X_t)\}} \quad (4)$$

The temperature α determines how strongly the weighting favors sources with lower MMD values. A larger α gives greater weight to more similar sources, while a smaller α distributes the weights more evenly. MMD is computed on features with consistent preprocessing using z-score scaling, and the softmax is applied with standard log-sum-exp stabilization. The resulting $\{w_d^{(k)}\}_{k=1}^K$ introduces an a priori assumption on transferability. Fig. 3 shows the Dataset-Level Weighting mechanism, where each target project is assigned suitable weights for each source dataset to calculate distributional similarity between the source and target projects using MMD.

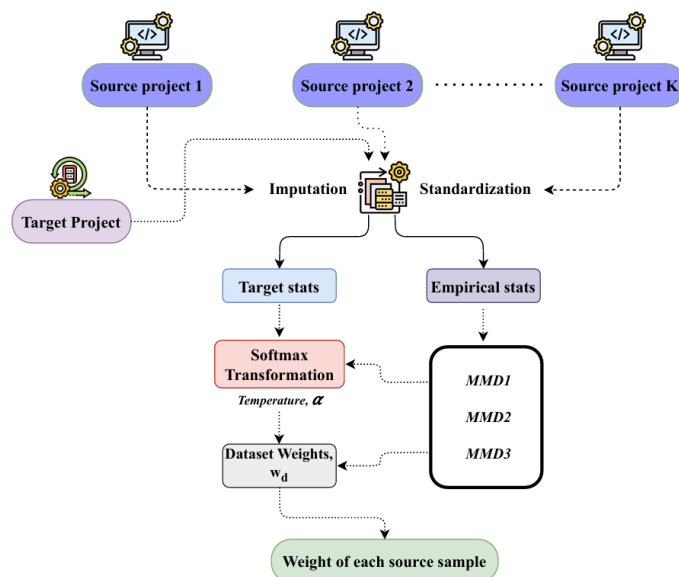


Fig. 3. Dataset-Level Weighting Mechanism.

3.3.1. Feature-Level Weighting

Different software metrics are not equally important or equally transferable. At the feature level, a feature-weighted scheme is adopted. For each feature j , two complementary measures are taken into consideration:

1. Distribution similarity (a_j): This measures the similarity between the source and target distributions of the feature j . The difference between normalized histograms is calculated using the L1 distance. Since a larger L1 distance indicates greater distributional discrepancy, an inverse transformation is used to obtain the transferability score. Features with smaller source-target distribution differences therefore receive larger transferability scores.
2. Predictive relevance (m_j): This measures the dependence between the feature and the defect label using mutual information (MI). MI is calculated using the source training data and represents the relevance of each software metric to the defect label. The source-target distributional similarity is estimated separately through the transferability measure.

The weight on the final feature is a convex combination:

$$w_{f,j} = \lambda a_j + (1 - \lambda)m_j, \quad \lambda \in [0,1] \quad (5)$$

where λ balances transferability and predictive relevance. Features that are both transferable and relevant to defect prediction receive higher weights, while less relevant features are downweighted. Fig. 4 shows the Feature-Level Weighting Mechanism, which integrates distribution similarity and predictive relevance to compute feature weights so the model can reweight the most transferable and predictive features. Feature weighting is applied directly to the feature representation rather than being multiplied into a single sample weight. Each feature is transformed according to its estimated transferability before classifier training. The instance-level transfer weight remains separate and is supplied to the Random Forest as a sample weight. This process allows the model to use features that are both informative for defect prediction and more transferable across projects.

3.3.2. Instance-Level Weighting

Individual data instances (software modules or files) may differ in their ability to represent the target project, even when they come from a similar project. An instance-level weighting scheme is therefore proposed. A domain classifier is learned to distinguish whether an instance comes from a source or target distribution. The resulting instance weights are constrained to a predetermined range to avoid unstable extreme ratios.

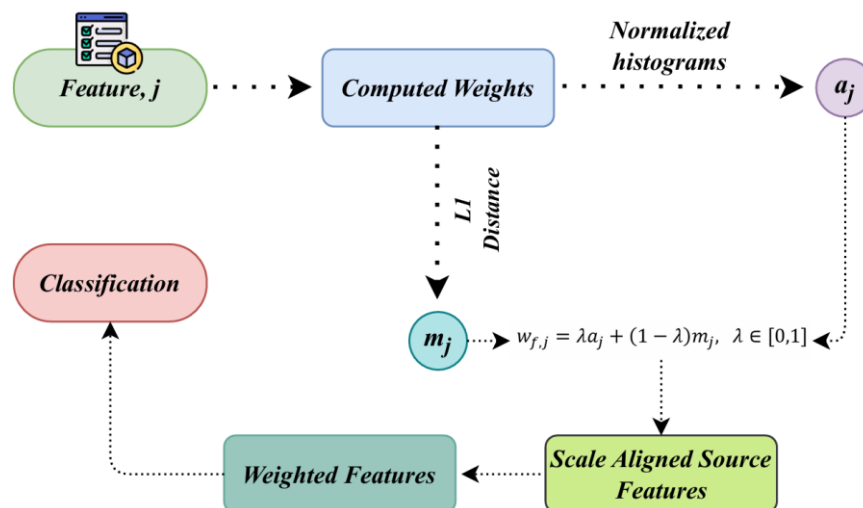


Fig. 4. Feature-Level weighting mechanism.

3.3.3. Domain Adaptation via CORAL

Covariance mismatch between the source and target feature spaces may remain after weighting the datasets, features, and instances. To address this, CORAL aligns the second-order statistics. To transform a source dataset X_S , the CORAL decomposition is defined by:

$$\tilde{X}_S = (X_S - \mu_S)C_S^{-\frac{1}{2}}C_t^{\frac{1}{2}} \quad (6)$$

where μ_S is the mean of the source features, and C_S and C_t are the covariance matrices of the source and target data, respectively. The transformation aligns the covariance structure of the source features with that of the target features, thereby reducing the distributional difference between the two domains. A small diagonal regularization term of 1×10^{-6} is added to the covariance matrices to avoid singular matrices and provide numerical stability during the CORAL transformation.

3.3.4. Random Forest Classification Layer

After the transfer learning phase, the aligned and reweighted feature sets from the multiple source projects are combined. The contribution of each source instance is determined using its dataset-level and instance-level transfer weights. The dataset-level weight $w_d^{(k)}$ represents the transferability of the source project, while $w_i(x_i)$ represents the transferability of the individual instance. Feature weights are applied separately to the feature representation before Random Forest training.

$$W(x_i) = w_d^{(k)} \cdot w_i(x_i) \quad (7)$$

where $w_d^{(k)}$ is the dataset-level weight of the source project containing x_i , and $w_i(x_i)$ is the corresponding instance-level weight. These weights are supplied to the Random Forest as sample weights, so that more transferable source instances have greater influence during training.

3.3.5. Training Objective and Parameter Configuration

The Random Forest classifier is trained using the composite transfer weights as sample weights. The transfer weights determine each source instance's contribution during Random Forest training, while the feature weights are applied to the feature representation before classification. Therefore, the Random Forest is not optimized using gradient-based cross-entropy loss. To ensure reproducibility and methodological transparency, Table 2 summarizes the proposed model's main parameters.

Table 2. Configuration of parameters for the proposed MS-TWL+RF framework.

Component	Parameter	Configuration
Dataset Weighting	Similarity metric	MMD
	Temperature, α	3.0
Feature Weighting	Distribution similarity	Histogram L1 distance
	Predictive relevance	MI
	Balance, λ	0.6
Instance Weighting	Domain classifier	Logistic Regression
	Weight clipping range	[0.1, 10.0]
Domain Adaptation	Alignment method	CORAL (covariance alignment)
	Regularization, ϵ	1×10^{-6}
RF Classifier	Number of trees (n_estimators)	200
	Min samples split	4
	Min samples leaf	2
	Bootstrap sampling	Enabled
	Random seed	42

4 RESULTS AND DISCUSSIONS

To comprehensively evaluate the proposed MS-TWL+RF framework on CPDP, multiple experiments were performed, and the model was compared with four baseline models: AlexNet+RF, ResNet50+RF, TransTab+RF, and XTab+RF. The experiments follow a multi-source cross-project defect prediction protocol. For each target project, the remaining projects are designated as source projects. Source-project samples are jointly used for model development, while the target-project defect labels are not used during training. Separate source-target configurations are evaluated to measure cross-project performance. For each configuration, the source data are divided into training and validation subsets using stratified sampling. SMOTE is applied only to the source training data, and the target-project features are used only for transfer weighting and domain adaptation. A single stratified source training-validation split is used, and the final performance is obtained by evaluating the trained model on the target project. For each experiment, one project is designated as the target project, while the remaining projects constitute the labeled source domains. The source projects are jointly preprocessed and subsequently combined for model training.

MMD is calculated between each source and the target project, and the resulting source-specific weights determine each source's contribution during training. Target defect labels are not used during preprocessing, weighting, model selection, or training and are reserved for final evaluation. MMD is calculated between each labeled source project and the target project using the common software metrics. CORAL is then applied to reduce covariance differences between the source and target feature spaces. Thus, MMD and CORAL operate on project-level domains rather than arbitrary partitions of a single dataset. For the deep-learning baseline models, the selected normalized software metrics are mapped to a two-dimensional representation, resized to the required spatial dimensions, and replicated across three channels to generate CNN-compatible inputs. The resulting deep feature representations are extracted and supplied to the RF classifier. The same transformation is applied to the source and target projects. The selected baseline models are evaluated under the same source-target protocol.

AlexNet+RF, ResNet-50+RF, TransTab+RF, and XTab+RF are included as supplementary deep-learning and tabular-model comparisons. The temperature parameter is fixed at $\alpha = 3.0$. CORAL regularization is set to $\epsilon = 10^{-6}$, feature weighting uses $\lambda = 0.6$, and instance weights are clipped to $[0.1, 10.0]$. RF uses 200 trees, a minimum split size of 4, a minimum leaf size of 2, bootstrap sampling, and random seed 42. Logistic Regression is employed as the domain classifier. It estimates the probability that a source instance belongs to the target domain, which is subsequently converted into an instance-level density-ratio weight. The weights are clipped to $[0.1, 10.0]$ to prevent unstable extreme values. The target project labels remain completely isolated during model development. Performance is reported across the predefined source-target configurations. The validation data are used within the source training procedure, while the target project is used as the final evaluation domain. Accordingly, the reported target-domain results are referred to as test performance. SMOTE is applied only to the source training data and never to validation or target-test samples.

The target defect labels are not used during MMD calculation, feature weighting, domain-classifier training, CORAL, or RF fitting. The target labels are used only for final evaluation. The target-project test distribution remains unchanged because SMOTE is not applied to the target data. The reported class distribution reflects the actual target-project data used during the cross-project evaluation. Multi-source CPDP is defined as the use of two or more labeled software projects as source domains for predicting defects in a separate target project. For a target project T , the source set is represented as $S = \{S_1, S_2, \dots, S_K\}$. The evaluation includes G-mean and Recall@20% in addition to accuracy, precision, recall, F1-score, AUC, MCC, Kappa, and Jaccard. These metrics provide a broader assessment of class discrimination and defect prioritization. For each target project, all remaining eligible projects are treated as source domains. Each source is weighted using MMD, while feature-level and instance-level weights refine its contribution. CORAL is applied to harmonize the source representations with the target feature distribution.

The transformed and weighted source instances are then pooled into a single training set for RF classification. The linear MMD is computed for each source-target pair and converted into normalized source weights using $\alpha = 3.0$. The CORAL module aligns the second-order statistical characteristics of the source and target project domains before classifier training. For each source-target configuration, the feature representations are centered using their respective mean vectors, and covariance matrices are estimated from the centered source and target representations. To improve numerical stability, a fixed regularization value is added to the diagonal elements of the covariance matrices. The regularized source covariance is then whitened, and the resulting representation is recolored using the target covariance structure. The transformed source features train the classifier, while the target defect labels remain unavailable during adaptation.

This procedure reduces distributional differences between the two project domains while preserving the underlying feature information required for defect classification. The same CORAL configuration and regularization setting are maintained across all source-target experiments. A regularization value of $\epsilon = 10^{-6}$ is used to ensure numerical stability. To verify the reported performance, the preprocessing operations used for source model development are fitted on the source training data. Feature selection, normalization, and SMOTE are restricted to the corresponding training data. Duplicate observations are removed before splitting, and target labels remain unavailable during adaptation. The manuscript additionally reports sample counts, class distributions, random seeds, and source-target configurations.

4.1. Experimental Setup and Dataset Splitting

The detailed experimental setting is important for ensuring the reproducibility and reliability of the findings obtained in software defect prediction. The experiments were conducted using the five projects JM1, KC1, CM1, KC2, and PC1 under a leave-one-project-out multi-source CPDP setting. In each configuration, one project was the target project, and the remaining four projects served as source projects. The source data were divided into training and validation subsets using an 80:20 stratified split with random state 42. SMOTE was applied only to the source training data. The target-project labels were not used during model development and were used only for final evaluation. Table 2 summarizes the main parameter settings used in the experiments.

4.2. Performance of Proposed Method

Table 3 presents the target-wise performance of the proposed MS-TWL+RF framework for the five leave-one-project-out CPDP configurations.

In each configuration, one project is treated as the target, and the remaining four projects are used as source projects. The evaluation includes accuracy, precision, recall, F1-score, AUC, MCC, Kappa, Jaccard, G-mean, and Recall@20%. The target-project defect labels are used only for final evaluation.

Table 3. Target-wise performance of MS-TWL+RF under the leave-one-project-out CPDP protocol.

Target	Source Projects	Accuracy	Precision	Recall	F1	AUC	MCC	Kappa	Jaccard	G-mean	Recall @ 20%
JM1	KC1, CM1, KC2, PC1	0.987	0.989	0.987	0.988	0.992	0.981	0.980	0.990	0.987	0.990
KC1	JM1, CM1, KC2, PC1	0.932	0.928	0.921	0.924	0.971	0.861	0.893	0.892	0.924	0.938
CM1	JM1, KC1, KC2, PC1	0.951	0.947	0.943	0.945	0.982	0.901	0.924	0.927	0.945	0.957
KC2	JM1, KC1, CM1, PC1	0.964	0.961	0.958	0.959	0.987	0.928	0.942	0.940	0.959	0.968
PC1	JM1, KC1, CM1, KC2	0.914	0.907	0.901	0.904	0.956	0.823	0.874	0.879	0.903	0.916

4.3. Comparative Evaluation Using Performance Metrics

In this part, the proposed MS-TWL+RF approach is compared with four baseline models. Table 4 compares MS-TWL+RF, AlexNet+RF, ResNet50+RF, TransTab+RF, and XTab+RF using training accuracy, target-domain accuracy, precision, recall, and F1-score. The results show performance differences across the evaluated models, with the proposed MS-TWL+RF achieving 98.7% target-domain accuracy and a 98.8% F1-score.

Table 4. Comparative evaluation of baseline models and the proposed hybrid model.

Model	Training Accuracy	Target-domain accuracy	Precision	Recall	F1-Score
AlexNet+RF [6]	85.2%	82.7%	83.1%	82.7%	82.9%
ResNet50+RF [7]	90.0%	88.5%	88.7%	88.5%	88.6%
TransTab+RF [40]	78.6%	76.3%	76.4%	76.3%	76.3%
XTab+RF [41]	72.4%	70.8%	70.9%	70.8%	70.8%
MS TWL+RF (Proposed)	98.9%	98.7%	98.9%	98.7%	98.8%

The proposed MS-TWL+RF model achieved 98.7% target-domain accuracy, 98.9% precision, 98.7% recall, and an F1-score of 98.8%. Among the baseline models, ResNet50+RF achieved 88.5% target-domain accuracy and an F1-score of 88.6%, while AlexNet+RF achieved 82.7% target-domain accuracy and an F1-score of 82.9%.

4.4. Graphical and Error Analysis

Fig. 5 shows the Receiver Operating Characteristic (ROC) curves for the proposed MS-TWL+RF framework and the four baseline models. The ROC curve represents the trade-off between the true positive rate (TPR) and false positive rate (FPR). In contrast, the Area Under the Curve (AUC) measures classification performance. As shown in Fig. 5, the proposed MS-TWL+RF model achieves an AUC of 0.992 in the reported experiment, indicating strong discrimination between defective and non-defective classes.

Compared with the baseline models, the MS-TWL+RF model shows a high AUC across the ROC curve (0.992). The ROC curve shows high TPR with relatively low FPR across the evaluated thresholds, indicating good discrimination between defective and non-defective classes in the reported experiment. Fig. 6 shows the confusion matrix of the proposed MS-TWL+RF model and presents the classification results for defective and non-defective software modules in the reported experiment. The confusion matrix provides the numbers of true positives, true negatives, false positives, and false negatives and is used to examine classification errors further. The confusion matrix shows a high level of agreement between the actual and predicted classes in the reported experiment.

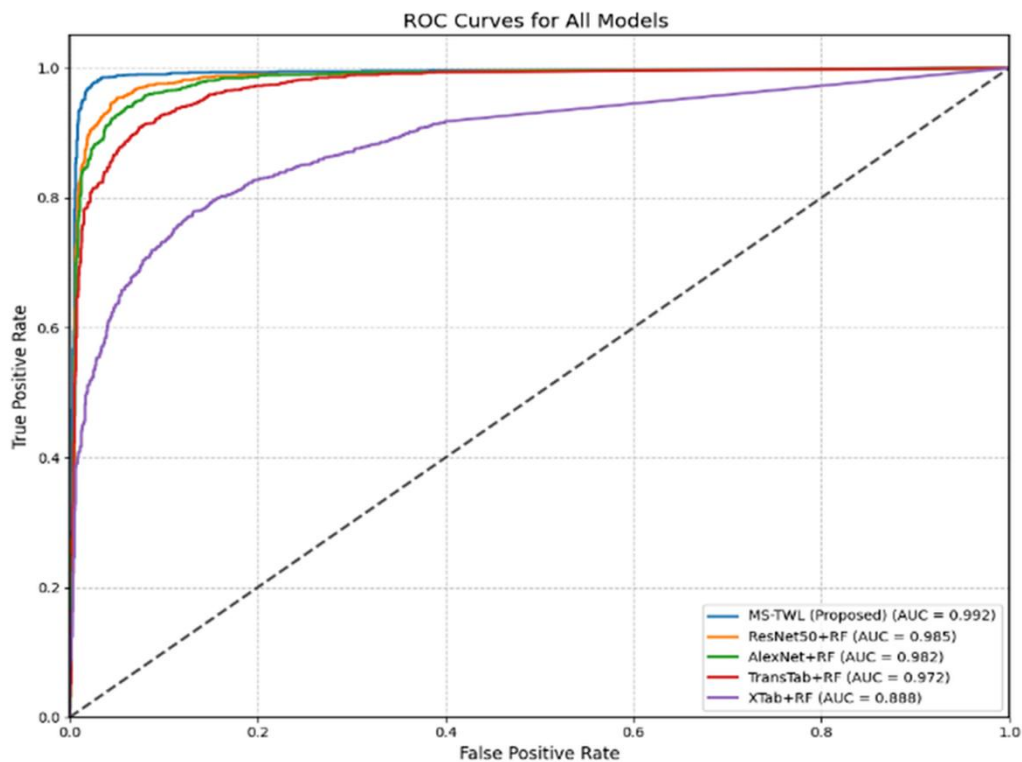


Fig. 5. ROC curve for all models.

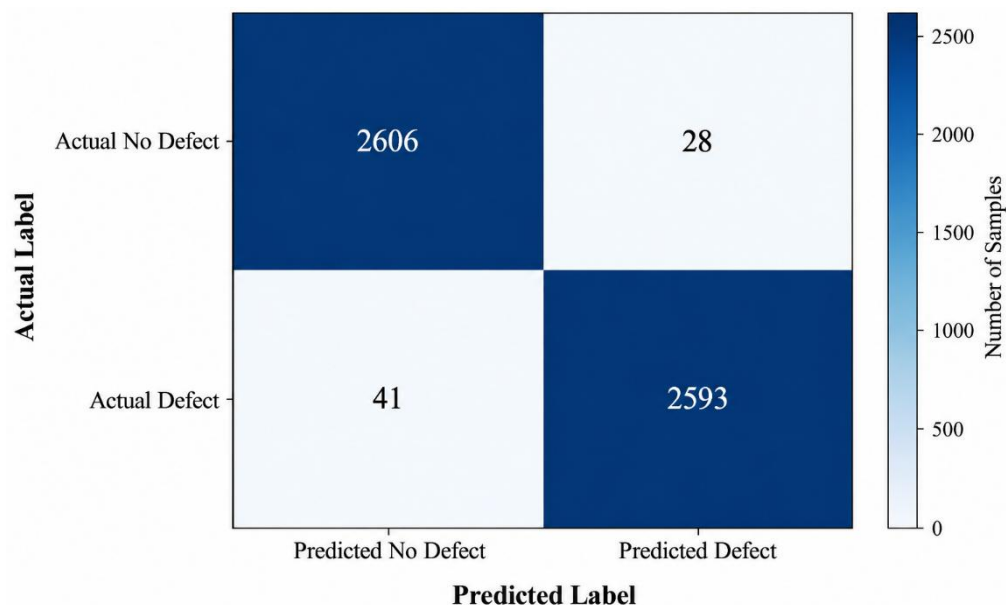


Fig. 6. Confusion matrix of the proposed MS-TWL+RF model for the JM1 target project.

4.5. Comparative Analysis of Reliability and Performance Metrics

This section compares the models in terms of reliability and performance evaluation measures. Fig. 7 provides a comparative visualization of the proposed MS-TWL+RF framework and the four baseline models using training accuracy, target-domain accuracy, precision, recall, and F1-score. In the reported experiment, the MS-TWL+RF model achieved values close to 99% across these measures, indicating strong classification performance. Among the baseline models, ResNet50+RF achieved comparatively higher training and target-domain accuracy, with values of 90.0% and 88.5%, respectively. Its precision, recall, and F1-score were 88.7%, 88.5%, and 88.6%, respectively. AlexNet+RF achieved training and validation accuracies of 85.2% and 82.7%, respectively, with precision, recall, and F1-score values of 83.1%, 82.7%, and 82.9%.

Table 5 presents a comparative analysis of the proposed MS-TWL+RF framework and the four baseline models using three reliability measures: Cohen's Kappa, Matthews Correlation Coefficient (MCC), and Jaccard index.

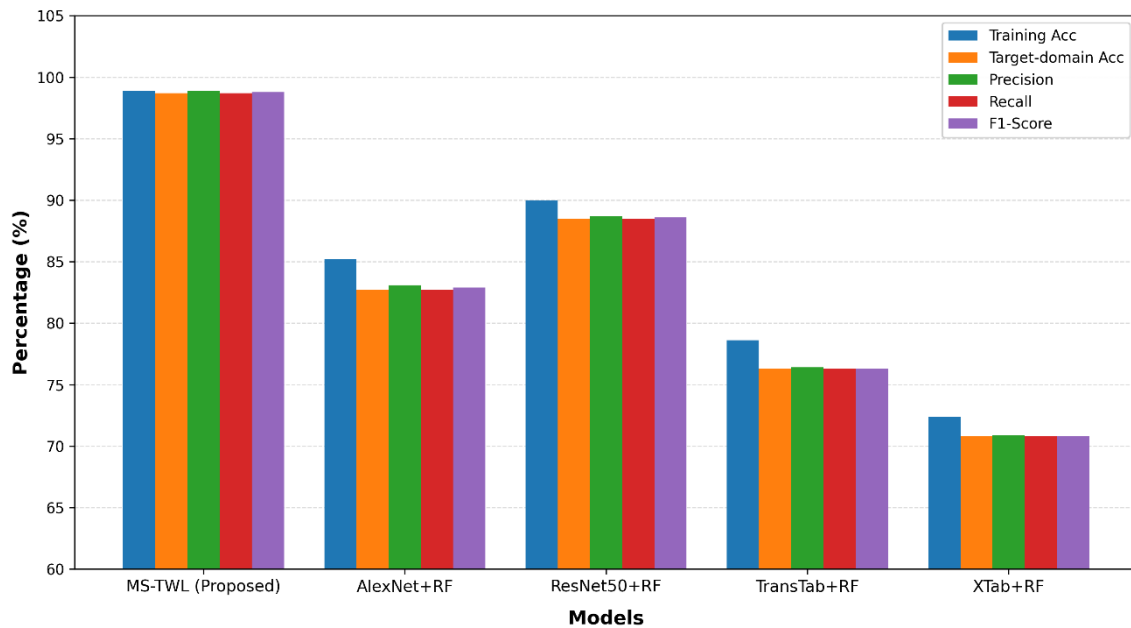


Fig. 7. Comparative performance of all models across training accuracy, target-domain accuracy, precision, recall, and F1-score.

Table 5. Comparative analysis of reliability metrics (Kappa, MCC, and Jaccard) across models.

Model	Kappa	MCC	Jaccard
AlexNet+RF [6]	0.820	0.823	0.835
ResNet50+RF [7]	0.880	0.882	0.892
TransTab+RF [40]	0.755	0.758	0.772
XTab+RF [41]	0.705	0.708	0.718
MS-TWL+RF (Proposed)	0.980	0.981	0.990

These measures provide additional information about the agreement and classification performance of the models beyond accuracy alone. In the reported experiment, the MS-TWL+RF model obtained a Kappa of 0.980, an MCC of 0.981, and a Jaccard index of 0.990.

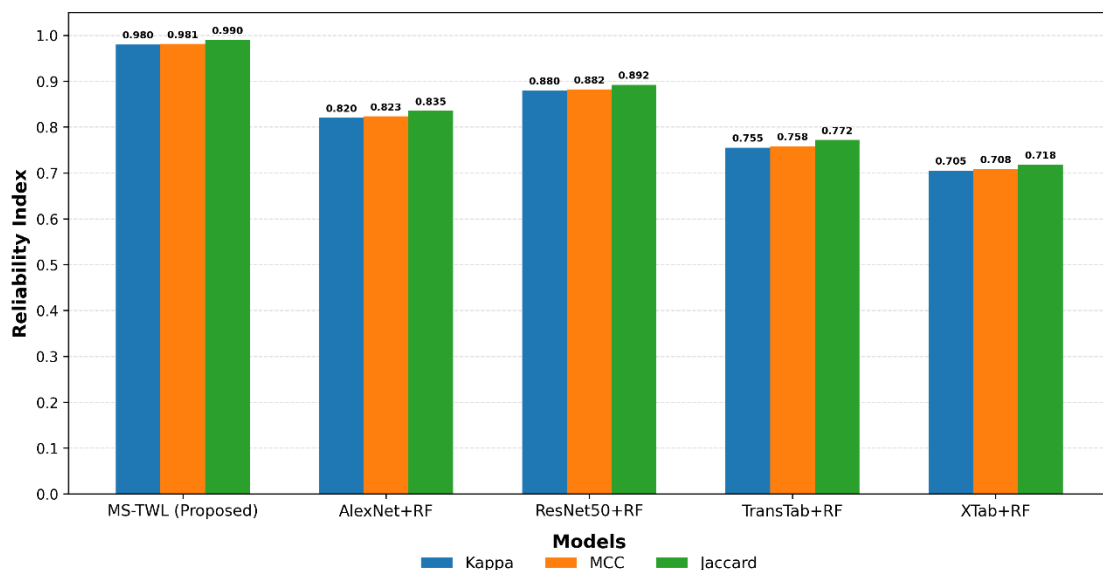


Fig. 8. Comparative bar plot of reliability indices (Cohen's Kappa, MCC, and Jaccard) for the proposed MS-TWL+RF framework and baseline models.

In addition to the numerical results in Table 5, Fig. 8 presents a comparative bar plot of the reliability indices. The table provides the exact values, while the bar plot provides a visual comparison of the models across Cohen's Kappa, Matthews Correlation Coefficient (MCC), and Jaccard Index.

4.6. Ablation Study

A component-wise ablation analysis examines the contribution of each component of MS-TWL+RF. The complete model is compared with progressively simplified variants. Table 6 presents these configurations. The first variant uses RF without transfer weighting. The second introduces MMD-based dataset weighting. The third adds feature-level weighting, while the fourth additionally introduces instance-level weighting. The final configuration incorporates CORAL alignment and represents the complete MS-TWL+RF framework.

Table 6. Ablation configurations of the proposed MS-TWL+RF framework.

Variant	MMD Weighting	Feature Weighting	Instance Weighting	CORAL
RF only	No	No	No	No
RF + MMD	Yes	No	No	No
RF + MMD + FW	Yes	Yes	No	No
RF + MMD + FW + IW	Yes	Yes	Yes	No
MS-TWL+RF	Yes	Yes	Yes	Yes

The ablation analysis is evaluated using F1-score, G-mean, MCC, AUC, and Recall@20%. The results are used to examine the effect of progressively adding the transfer mechanisms. In particular, the comparison between the fourth and fifth variants shows the effect of adding CORAL. In contrast, the successive configurations show the changes obtained by adding dataset-, feature-, and instance-level weighting. Table 7 gives the ablation study.

Table 7. Ablation results of the proposed MS-TWL+RF framework.

Variant	Accuracy	Precision	Recall	F1	AUC	MCC	G-mean
RF only	0.861	0.852	0.844	0.848	0.901	0.716	0.847
RF + MMD	0.889	0.881	0.876	0.878	0.928	0.773	0.878
RF + MMD + FW	0.921	0.914	0.909	0.911	0.956	0.842	0.910
RF + MMD + FW + IW	0.951	0.946	0.941	0.943	0.978	0.901	0.943
MS-TWL+RF	0.987	0.989	0.987	0.988	0.992	0.981	0.987

The ablation experiment is reported for the JM1 target configuration using KC1, CM1, KC2, and PC1 as source projects. The results show a progressive increase in the evaluation measures as the transfer components are added. RF alone achieved an F1-score of 0.848, which increased to 0.878 after adding MMD-based dataset weighting, 0.911 after adding feature weighting, and 0.943 after adding instance weighting. The complete MS-TWL+RF configuration, which additionally applies CORAL, achieved an F1-score of 0.988 and an AUC of 0.992.

4.7. Comparative Analysis and Discussion

Table 8 summarizes a comparative analysis of the proposed MS-TWL+RF framework with selected CPDP methods reported in the literature. Omondigbe et al. [4] proposed the SASMLM approach, which is based on similarity-based attribute selection and multi-learning.

Table 8. Comparative analysis of selected studies and the proposed MS-TWL+RF framework for cross-project software defect prediction.

Work	Dataset	Model	Reported Performance / Metric
[2]	AEEEM / ReLink	SASMLM	0.82 (AEEEM), 0.73 (ReLink)
[42]	NASA (JM1, KC1, KC2, CM1, PC1)	CPDP Comparative Study	0.555–0.569
[10]	NASA / AEEEM / ReLink / SoftLab	Model Averaging (XGB+LGBM)	0.8063
[15]	PROMISE	ConCPDP	0.554
[22]	PROMISE + AEEEM + ReLink	IAPCP	↑13.78% improvement (F1)
[14]	PROMISE (14) + Softlab (5)	FedDP	≈0.566 (example)
[5]	PROMISE (6 projects)	SDP-MTF	0.7061
[19]	PROMISE	RH-CPDP	0.663 (example pair)
Current Work	PROMISE (JM1, KC1, CM1, KC2, PC1)	MS-TWL+RF (Proposed)	98.7 (Accuracy), 98.8 (F1)

On the AEEEM dataset, the approach achieved an F1-score of 0.82, while on ReLink it achieved 0.73. Kaliraj et al. [42] carried out a comparative analysis using the NASA projects JM1, KC1, KC2, CM1, and PC1. Their reported F1-scores for cross-project prediction experiments ranged from 0.555 to 0.569, highlighting the challenges of defect prediction across heterogeneous projects. The proposed MS-TWL+RF framework achieved 98.7% accuracy and an F1-score of 98.8% in the reported experiment. Existing studies reported results using different datasets and experimental settings, so these values provide a reference for comparison rather than a direct one-to-one performance comparison.

5 CONCLUSION

This study presents an MS-TWL+RF framework for cross-project software defect prediction under heterogeneous project conditions. The experimental results show the performance of the proposed framework. MS-TWL+RF achieves 98.9% training accuracy, 98.7% target-domain accuracy, 98.9% precision, 98.7% recall, and 98.8% F1-score. The model also obtains an AUC of 0.992, together with Kappa, MCC, and Jaccard values of 0.980, 0.981, and 0.990, respectively. Future research can evaluate MS-TWL+RF across larger collections of PROMISE, NASA, AEEEM, ReLink, and other cross-project defect datasets using repeated cross-project evaluation.

FUNDING INFORMATION

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ETHICS STATEMENT

This study did not involve human or animal subjects and, therefore, did not require ethical approval.

STATEMENT OF CONFLICT OF INTERESTS

The authors declare no conflicts of interest related to this study.

LICENSING

This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

REFERENCES

- [1] A. Deshmukh, D. S. Patil, J. S. S. Mohan, G. Balamurugan, and A. K. Tyagi, "Transforming next Generation-Based artificial intelligence for software development," in *Advances in logistics, operations, and management science book series*, 2023, pp. 30–66. doi: 10.4018/978-1-6684-8088-5.ch003.
- [2] O. P. Omondiagbe, S. A. Licorish, and S. G. MacDonell, "Improving transfer learning for software cross-project defect prediction," *Applied Intelligence*, vol. 54, no. 7, pp. 5593–5616, Apr. 2024, doi: 10.1007/s10489-024-05459-1.
- [3] J. Chen, J. Ding, K. C. Tan, J. Qian and K. Li, "MBL-CPDP: A Multi-Objective Bilevel Method for Cross-Project Defect Prediction," in *IEEE Transactions on Software Engineering*, vol. 51, no. 8, pp. 2305-2328, Aug. 2025, doi: 10.1109/TSE.2025.3577808.
- [4] Y. Wang, Y. Li, H. Wang, L. Zhao, and X. Zhang, "Better knowledge enhancement for Privacy-Preserving Cross-Project Defect Prediction," *Journal of Software Evolution and Process*, vol. 37, no. 1, Jan. 2025, doi: 10.1002/smr.2761.
- [5] T. Lei, J. Xue, D. Man, Y. Wang, M. Li, and Z. Kong, "SDP-MTF: A composite transfer learning and feature fusion for Cross-Project software defect prediction," *Electronics*, vol. 13, no. 13, p. 2439, Jun. 2024, doi: 10.3390/electronics13132439.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.
- [7] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
- [8] Q. Qian, Y. Wang, T. Zhang, and Y. Qin, "Maximum mean square discrepancy: A new discrepancy representation metric for mechanical fault transfer diagnosis," *Knowledge-Based Systems*, vol. 276, p. 110748, Jul. 2023, doi: 10.1016/j.knosys.2023.110748.
- [9] G. Zhang, T. Zhou, and Y. Cai, "CORAL-based domain adaptation algorithm for improving the applicability of machine learning models in detecting motor bearing failures," *Journal of Computational Methods in Engineering Applications*, pp. 1–17, Nov. 2023, doi: 10.62836/jcmea.v3i1.030108.
- [10] T. Li, Z. Wang, and P. Shi, "Within-project and cross-project defect prediction based on model averaging," *Scientific Reports*, vol. 15, no. 1, p. 6390, Feb. 2025, doi: 10.1038/s41598-025-90832-4.
- [11] K. Javed, R. Shengbing, M. Asim, and M. A. Wani, "Cross-Project defect prediction based on domain adaptation and LSTM optimization," *Algorithms*, vol. 17, no. 5, p. 175, Apr. 2024, doi: 10.3390/a17050175.

- [12] J. Bai, J. Jia, and L. F. Capretz, "A three-stage transfer learning framework for multi-source cross-project software defect prediction," *Information and Software Technology*, vol. 150, p. 106985, Jun. 2022, doi: 10.1016/j.infsof.2022.106985.
- [13] S. Jiang, J. Zhang, F. Guo, T. Ouyang, and J. Li, "Balanced adversarial tight matching for Cross-Project defect prediction," *IET Software*, vol. 2024, no. 1, Jan. 2024, doi: 10.1049/2024/1561351.
- [14] Y. Wang, Y. Li, H. Wang, L. Zhao, and X. Zhang, "Better knowledge enhancement for Privacy-Preserving Cross-Project Defect Prediction," *Journal of Software Evolution and Process*, vol. 37, no. 1, Jan. 2025, doi: 10.1002/smr.2761.
- [15] H. Song, Y. Pan, F. Guo, X. Zhang, L. Ma, and S. Jiang, "CONCPDP: a Cross-Project defect Prediction method integrating contrastive pretraining and category boundary adjustment," *IET Software*, vol. 2024, no. 1, Jan. 2024, doi: 10.1049/2024/5102699.
- [16] Y. Zhao, Y. Zhu, Q. Yu, and X. Chen, "Cross-Project defect prediction considering multiple data distribution simultaneously," *Symmetry*, vol. 14, no. 2, p. 401, Feb. 2022, doi: 10.3390/sym14020401.
- [17] Z. Hu and Y. Zhu, "Cross-project defect prediction method based on genetic algorithm feature selection," *Engineering Reports*, vol. 5, no. 12, May 2023, doi: 10.1002/eng2.12670.
- [18] H. Tao *et al.*, "Cross-Project Defect Prediction Using Transfer Learning with Long Short-Term Memory Networks," *IET Software*, vol. 2024, no. 1, Jan. 2024, doi: 10.1049/2024/5550801.
- [19] A. Abdu *et al.*, "Cross-project software defect prediction based on the reduction and hybridization of software metrics," *Alexandria Engineering Journal*, vol. 112, pp. 161–176, Nov. 2024, doi: 10.1016/j.aej.2024.10.034.
- [20] A. Abdu, Z. Zhai, H. A. Abdo, R. Algabri, and S. Lee, "Graph-Based feature learning for Cross-Project software defect prediction," *Computers, Materials & Continua/Computers, Materials & Continua (Print)*, vol. 77, no. 1, pp. 161–180, Jan. 2023, doi: 10.32604/cmc.2023.043680.
- [21] R. Haque, A. Ali, S. McClean, I. Cleland and J. Noppen, "Heterogeneous Cross-Project Defect Prediction Using Encoder Networks and Transfer Learning," in *IEEE Access*, vol. 12, pp. 409–419, 2024, doi: 10.1109/ACCESS.2023.3343329.
- [22] N. Zhang, K. Zhu, and D. Zhu, "IAPCP: an effective Cross-Project defect prediction model via Intra-Domain alignment and Programming-Based distribution adaptation," *IET Software*, vol. 2024, no. 1, Jan. 2024, doi: 10.1049/2024/5358773.
- [23] P. Ardimento, L. Aversano, M. L. Bernardi, M. Cimitile, and M. Iammarino, "Just-in-time software defect prediction using deep temporal convolutional networks," *Neural Computing and Applications*, vol. 34, no. 5, pp. 3981–4001, Nov. 2021, doi: 10.1007/s00521-021-06659-3.
- [24] J. Yao, B. Liu, Y. Wu, and Z. Li, "Multi-Source heterogeneous kernel mapping in software defect prediction," *Applied Sciences*, vol. 13, no. 9, p. 5526, Apr. 2023, doi: 10.3390/app13095526.
- [25] F. Li, P. Yang, J. W. Keung, W. Hu, H. Luo, and X. Yu, "Revisiting 'revisiting supervised methods for effort-aware cross-project defect prediction,'" *IET Software*, vol. 17, no. 4, pp. 472–495, Jun. 2023, doi: 10.1049/sfw2.12133.
- [26] T. Lei, J. Xue, D. Man, Y. Wang, M. Li, and Z. Kong, "SDP-MTF: A composite transfer learning and feature fusion for Cross-Project software defect prediction," *Electronics*, vol. 13, no. 13, p. 2439, Jun. 2024, doi: 10.3390/electronics13132439.
- [27] T. Lei, J. Xue, Y. Wang, Z. Niu, Z. Shi, and Y. Zhang, "WCM-WTRA: A Cross-Project defect Prediction method based on feature Selection and Distance-Weight Transfer Learning," *Chinese Journal of Electronics*, vol. 31, no. 2, pp. 354–366, Mar. 2022, doi: 10.1049/cje.2021.00.119.
- [28] J. Zou, Z. Li, X. Liu, and H. Tong, "MSCPDPLab: A MATLAB toolbox for transfer learning based multi-source cross-project defect prediction," *SoftwareX*, vol. 21, p. 101286, Dec. 2022, doi: 10.1016/j.softx.2022.101286.
- [29] H. Yamamoto, D. Wang, G. K. Rajbahadur, M. Kondo, Y. Kamei and N. Ubayashi, "Towards Privacy Preserving Cross Project Defect Prediction with Federated Learning," *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Taipa, Macao, 2023, pp. 485–496, doi: 10.1109/SANER56733.2023.00052.
- [30] A. Wang, Y. Feng, M. Yang, H. Wu, Y. Iwahori, and H. Chen, "Cross-Project Software Defect Prediction Using Differential Perception Combined with Inheritance Federated Learning," *Electronics*, vol. 13, no. 24, p. 4893, Dec. 2024, doi: 10.3390/electronics13244893.
- [31] I. Mehmood *et al.*, "A Novel Approach to Improve Software Defect Prediction Accuracy Using Machine Learning," in *IEEE Access*, vol. 11, pp. 63579–63597, 2023, doi: 10.1109/ACCESS.2023.3287326.
- [32] C. Pan, M. Lu, B. Xu, and H. Gao, "An improved CNN model for Within-Project software defect prediction," *Applied Sciences*, vol. 9, no. 10, p. 2138, May 2019, doi: 10.3390/app9102138.
- [33] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," *Journal of Systems and Software*, vol. 195, p. 111537, Oct. 2022, doi: 10.1016/j.jss.2022.111537.
- [34] N. A. A. Khleel and K. Nehéz, "Software defect prediction using a bidirectional LSTM network combined with oversampling techniques," *Cluster Computing*, vol. 27, no. 3, pp. 3615–3638, Oct. 2023, doi: 10.1007/s10586-023-04170-z.

- [35] M. Ali *et al.*, "Software Defect Prediction Using an Intelligent Ensemble-Based Model," in *IEEE Access*, vol. 12, pp. 20376-20395, 2024, doi: 10.1109/ACCESS.2024.3358201.
- [36] L.-Q. Chen, C. Wang, and S.-L. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection," *Complex & Intelligent Systems*, vol. 8, no. 4, pp. 3333–3348, Feb. 2022, doi: 10.1007/s40747-022-00676-y.
- [37] Y. Z. Bala, P. Abdul Samat, K. Y. Sharif and N. Manshor, "Improving Cross-Project Software Defect Prediction Method Through Transformation and Feature Selection Approach," in *IEEE Access*, vol. 11, pp. 2318-2326, 2023, doi: 10.1109/ACCESS.2022.3231456.
- [38] Y. Tang, Q. Dai, M. Yang, T. Du, and L. Chen, "Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 6, pp. 1967–1987, Jan. 2023, doi: 10.1007/s13042-022-01740-2.
- [39] J. Sayyad Shirabad and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada, 2005. Available: <http://promise.site.uottawa.ca/SERepository>.
- [40] Z. Wang and J. Sun, "TransTab: Learning transferable tabular transformers across tables," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2902–2915, 2022, doi: 10.52202/068431-0210.
- [41] B. Zhu, X. Shi, N. Erickson, M. Li, G. Karypis, and M. Shoaran, "XTab: Cross-table pretraining for tabular transformers," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202, pp. 43181–43204, 2023.
- [42] S. Kaliraj, A. M. Kishoore and V. Sivakumar, "Software Fault Prediction Using Cross-Project Analysis: A Study on Class Imbalance and Model Generalization," in *IEEE Access*, vol. 12, pp. 64212-64227, 2024, doi: 10.1109/ACCESS.2024.3397494.