Vol. 8, No. 3, June 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Transfer Performance and LIME Explanation of Ensemble Classifiers in Cross-Project Defect Prediction

Bassey Isong

Computer Science Department, North-West University, Mafikeng, South Africa

Received:
October 15, 2025
Revised:
May 15, 2026
Accepted:
June 6, 2026
Published:
June 26, 2026

Corresponding Author:
Author Name*:
Bassey Isong
Email*:
bassey.isong@nwu.ac.za

DOI:
10.63158/journalisi.v8i3.1667

Abstract. Ensemble methods are widely used in cross-project defect prediction (CPDP), particularly in projects that lack sufficient historical data by training on external source projects. However, no prior study has compared Bagging, Boosting, and Stacking directly under a Leave-One-Project-Out (LOPO) protocol or examined whether within-project performance rankings carry over to the cross-project setting. We evaluated three ensemble classifiers on five NASA MDP datasets sharing a common Halstead and McCabe feature schema. SMOTE is applied exclusively to pooled source data to prevent leakage into the target. A no-SMOTE baseline isolates the contribution of source-only SMOTE. LIME explanations are aggregated over thirty instances per model to assess feature importance consistency across the project boundary. Within-project evaluation shows Stacking achieves the highest F1 on four of five datasets, peaking at 0.503 on KC1. Under LOPO, these rankings reverse as Bagging and Boosting transfer more reliably, while Stacking's F1 drops by up to 0.258 points. Source-only SMOTE consistently improves transfer across all targets and ensembles. LIME consistency analysis produces undefined Spearman rank correlations, indicating that thirty-instance aggregation is insufficient to produce stable rank vectors for 21-feature datasets. To the best of our knowledge, this is the first study to compare all three ensemble strategies under LOPO on a shared NASA dataset feature schema. Particularly with a no-SMOTE control, aggregated LIME analysis, and a pilot meta-feature study identifying dataset size as the most actionable label-free predictor of ensemble suitability for CPDP deployment.

Keywords: SDP, Cross-project defect prediction, Ensemble Learning, SMOTE, LIME, Explainability, Leave-One-Project-Out, NASA MD

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

1. INTRODUCTION

In software quality assurance (SQA) and software maintenance, software defect prediction (SDP) identifies fault-prone modules before testing begins, so that inspection effort can be channelled where it is most likely to find defects [1], [2], [3]. To achieve this, most prediction models are trained and tested on data from the same project, an approach known as within-project defect prediction (WPDP) [4], [5]. WPDP works well when a project has accumulated several releases of labelled defect data. This is because training and test sets share the same metric distributions and the same defect-introduction patterns [4], [6]. However, many projects reach a testing phase before any usable defect history exists. New development initiatives, first-version releases, and teams that do not maintain defect records all face this problem [7], [8]. Unlike WPDP, cross-project defect prediction (CPDP) addresses it by training on labelled data from external source projects and applying the resulting model to the target [9], [10], [11]. This study focuses on homogeneous CPDP, where source and target projects share the same feature schema, specifically Halstead, McCabe, and LOC metrics common to all five NASA MDP datasets used here. This removes feature-space heterogeneity as a confound and isolates the effect of distribution shift. Even within a shared schema, however, source and target projects rarely share the same data distribution: differences in team size, development methodology, application domain, and coding conventions, all of which widen the gap between them [7], [12], [13]. Empirical studies consistently show cross-project models underperform their within-project counterparts in most source-target combinations. Reliable transfer is achieved only when projects share favourable conditions of project similarity [12], [13].

To close this performance gap, ensemble classifiers have been widely adopted [3]. Bagging reduces prediction variance across varied source distributions. Boosting reweights difficult instances sequentially and adapts to imbalanced class ratios. Stacking trains a meta-learner on top of diverse base classifiers and can capture interactions that homogeneous ensembles miss [2], [3], [4], [14]. Class imbalance compounds the problem as defective modules typically account for fewer than 20% of any given dataset and sometimes fewer than 7% [7], [15]. The Synthetic Minority Oversampling Technique (SMOTE) among others, is the most widely used remedy, though its interaction with cross-project transfer has received limited study [16], [17].

Another challenge is that the decision logic of ensemble models is opaque to practitioners. A developer who receives a defect-prone prediction needs to know which features drive it before deciding how to respond. When the prediction comes from a model trained on a different project, a further question arises: are those feature attributions valid for the target setting? Local Interpretable Model-Agnostic Explanations (LIME) generate per-instance feature importance weights for any classifier. Whether those weights remain stable when a model transfers across projects has not been examined. Chen et al. [10] found that feature rankings produced by a global cross-project model were consistent with individual local models in only 55% of cases, which points to a real risk that transferred explanations mislead practitioners. No previous work has quantified this instability across multiple ensemble types using aggregated LIME explanations. In the same vein, a related unresolved question is how to select the appropriate ensemble for a given target project without running a full experiment. Dataset meta-features are statistics computed before any model is trained, such as class imbalance ratio, skewness, and feature correlation. They have been used in other machine learning settings to predict algorithms performance on unseen datasets.

Therefore, this paper evaluates ensemble-based homogeneous CPDP on five NASA MDP datasets: CM1, KC1, JM1, KC2, and PC1, using a Leave-One-Project-Out (LOPO) evaluation design. These datasets are used because they provide a publicly available, well-documented benchmark widely adopted in CPDP research and directly comparable with prior work [18], [19]. All five share the same Halstead and McCabe feature schema. This removes feature-space heterogeneity as a confound and isolates distributional shift as the primary source of cross-project difficulty. SMOTE is applied exclusively to pooled source data inside the training pipeline to prevent leakage into the target. The findings are scoped to the homogeneous setting and should not be generalised to heterogeneous CPDP without further validation. The four research questions (RQs) addressed are: RQ1: Which ensemble learning technique (Bagging, Boosting, or Stacking) transfers most reliably under the LOPO cross-project setting? RQ2: Does applying SMOTE to exclusively pooled source data improve cross-project transfer, and does the effect vary across targets? RQ3: Are LIME feature importance rankings stable when the same ensemble is assessed within a project versus across projects? and RQ4: Can dataset meta-features predict which ensemble will perform best on an unseen target project?

The main contributions are as follows:

1) A controlled LOPO comparison of Bagging, Boosting, and Stacking on five NASA MDP datasets under a fixed 21-feature homogeneous schema. F1, AUC-ROC, MCC, and Brier score are reported alongside a no-SMOTE control baseline.
2) A LIME consistency analysis in which explanations are accumulated over thirty test instances per model, and Spearman rank correlation quantifies explanation stability between the within-project and cross-project settings. This study is the first to apply this diagnostic across multiple ensemble types on a shared NASA dataset schema.
3) A pilot meta-feature analysis identifying imbalance ratio and dataset size as preliminary predictors of ensemble suitability for CPDP. This is achieved with an explicit distinction between label-free and label-dependent meta-features relevant to unlabelled deployment scenarios.

The remainder of this paper is structured as follows: Section 2 presents related work on CPDP, class imbalance, ensemble learning, and explainability in SDP. Section 3 describes the methodology used. Section 4 interprets and discusses the findings, compares them against prior work, and addresses threats to validity, while Section 5 concludes and provides specific future research directions.

2. RELATED WORKS

This section covers WPDP, CPDP, class imbalance handling, ensemble architecture in SDP literature, and the use of explainability methods in defect prediction. Table 1 summarizes relevant studies against these dimensions.

2.1. Within-project and cross-project defect prediction

In SDP, WPDP trains and tests a classifier using data from the same project or its prior releases. Because both sets come from the same distribution, WPDP models tend to capture project-specific defect patterns and generally outperform cross-project models when labelled history is available [4], [6]. The practical constraint is straightforward: projects in early development lack the defect records needed to train a reliable within-project model [7], [8]. On the other hand, CPDP reframes prediction as a transfer task, substituting labelled data from external source projects for missing local history [9], [10].

Homogeneous CPDP, where source and target share the same metric suite, is the more tractable variant because no feature mapping is required [4], [20]. The five NASA datasets used in this study fall into this category, all sharing the Halstead and McCabe schema. Heterogeneous CPDP, where feature spaces differ, requires additional alignment steps such as encoder networks [20] or canonical correlation analysis [16], which introduces extra sources of error beyond distributional shift.

Accordingly, the distributional gap between source and target is the central technical difficulty. Several approaches have been proposed to reduce it. For example, instance weighting based on source-target similarity [7], [12], [21] source project selection by collaborative filtering [22] or clustering [23], and domain adaptation using kernel methods [24]. Tong et al. [12] combined triple feature weighting with an adaptive fallback to unsupervised methods when source data is unreliable, reporting improvements over transfer learning baselines on 34 datasets. Haonan et al. [11] pooled data from multiple source projects, applied KL divergence and mutual information to weight source instances, and found that multi-source transfer outperforms single-source transfer across 30 projects. The LOPO design in this study follows the same multi-source logic, where for each target, the remaining four datasets are pooled as the source. Evaluation protocols vary across the CPDP literature. All-pairs evaluation trains on one project and tests on each of the others in turn. LOPO is a stricter variant in which each dataset serves as a target exactly once, with the rest forming the source pool. LOPO reduces the risk that results are driven by a few favourable source-target combinations and is increasingly adopted in comparative CPDP studies [6], [25].

2.2. Class Imbalance in defect datasets

Defect datasets are skewed, as non-defective modules outnumber defective ones in most projects, with imbalance ratios ranging from roughly 2:1 to above 13:1 in the NASA datasets collection. Standard classifiers trained on such data tend to assign most instances to the majority class, producing high overall accuracy but low recall on defective modules [7],[15]. This makes accuracy unreliable as a primary metric, like F1, and AUC-ROC gives a more informative picture of minority-class performance [4], [26]. Furthermore, SMOTE generates synthetic minority instances through linear interpolation between existing defective samples and is the most widely applied imbalance remedy in SDP [16], [17]. Mustaqeem et al. [17] combined SMOTE-ENN with XGBoost and recursive feature

elimination on four NASA datasets, including CM1 and KC1, reporting F1 improvements over classifiers trained on the unbalanced data. Fan et al. [16] paired SMOTE with Deep Canonical Correlation Analysis for cross-project transfer and improved AUC across 27 projects.

A methodological concern specific to CPDP is where SMOTE is applied relative to the source-target boundary. Fitting SMOTE on data that includes target instances, or applying it to the whole dataset before splitting, introduces synthetic samples derived from the target distribution into training, inflating performance estimates. In this study, SMOTE is embedded inside the training pipeline and fitted only on source data, so the target set is never modified.

2.3. Ensemble methods in SDP

Ensemble methods appear throughout SDP literature because they address two recurring problems: the variance that comes from training on small or noisy defect datasets, and the class bias that individual classifiers develop on imbalanced data [2], [3], [4], [14]. Bagging trains multiple base classifiers on bootstrap samples and aggregates by majority vote. Random Forest (RF) is the most cited bagging variant in SDP and appears in several studies on NASA datasets as a reliable baseline [1], [2], [3]. Its variance-reduction mechanism makes it comparatively stable when training data comes from multiple heterogeneous projects. On a similar note, boosting train classifiers sequentially, reweighting instances that earlier learners misclassified. AdaBoost and XGBoost are the most common variants in SDP. Wang et al. [19] adapted AdaBoost with an instance migration step for cross-project transfer and reported F1 improvements over standard baselines. Nikravesh and Keyvanpour [7] used a transfer boosting algorithm with Newton-gravity-based instance weighting and obtained the best AUC result in 67% of project pairs across three benchmark repositories.

Stacking trains diverse base classifiers and feeds their out-of-fold predictions to a meta-learner. Chen et al. [14] applied a two-layer stacking architecture on eight NASA datasets and reported improvements over single-ensemble baselines in AUC and MCC. Likewise, Xin et al. [4] compared Bagging, Boosting, and Stacking directly on 22 NASA and PROMISE datasets and recommended Boosting for time-sensitive projects. Neither study evaluated the three strategies under LOPO conditions, and neither asked whether within-project

rankings carry over to the cross-project setting. Chen et al. [10] framed ensemble selection as a multi-objective bi-level optimization problem and searched over pipeline configurations simultaneously. Their MBL-CPDP tool outperformed AutoML baselines on AUC across 20 projects, though the computational cost is high. This study takes a different approach. Three fixed ensemble strategies are evaluated under controlled conditions, and dataset meta-features are examined as a lower-cost mechanism for guiding ensemble selection.

2.4. Explainability in defect prediction

In SDP, prediction accuracy alone does not satisfy practitioners who need to act on a defect-prone label. A developer must know which features steered the prediction prior to deciding whether and how to respond. Post-hoc explanation methods, particularly LIME and SHAP as XAI methods, address this by assigning signed importance weights to features for individual predictions [10], [13]. Most SDP applications of these tools report feature importance for a single instance or compute global importance over the full test set. Single-instance LIME is unstable. This is because small perturbations to the input can change the rank order of features substantially [27]. Consequently, it is not reliable as evidence of a model's general behaviour. This study collects LIME explanations over thirty test instances per model and uses Spearman rank correlation to compare the resulting rankings between the within-project and cross-project settings.

Whether explanations transfer reliably across project borders has received almost no attention. Chen et al. [10] noted that the most important features identified by a global cross-project model were consistent with individual local models in only 55% of cases. If true at the ensemble level, a model with acceptable cross-project F1 could still produce misleading feature attributions for the target project. No prior study has tested this across multiple ensemble types, which is the focus of RQ3.

Table 1. Summary of related studies

Ref.	CPDP	Method	Ensemble	Datasets	XAI
[2]	No	None	Voting ensemble	7 NASA MDP	No
[4]	No	SMOTE	Bagging/Boosting/Stack	22 (NASA, PROMISE)	No

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Ref.	CPDP	Method	Ensemble	Datasets	XAI
[7]	Yes	SMOTE + cost	Boosting (transfer)	AEEEM, NASA, SOFTLAB	No
[10]	Yes	Various	AutoML pipeline	20 (AEEEM, JURECZKO)	Partial
[11]	Yes	SMOTE	Weighted ensemble	30 (AEEEM, PROMISE)	No
[12]	Yes	SMOTE	Boosting (transfer)	34 (AEEEM, PROMISE, NASA)	No
[14]	No	Feature filter	Stacking (2-layer)	8 NASA MDP	No
[16]	Yes	SMOTE	SVM via DCCA	27 (NASA, PROMISE)	No
[17]	No	SMOTE-ENN	XGBoost	CM1, PC1, KC1 (NASA)	Partial
[21]	Yes	None stated	Boosting (adapted)	Multiple	No
This Study	Yes	SMOTE (src)	Bagging/Boosting/Stack	5 NASA MDP, LOPO	LIME (RQ3)

Table 1 maps related studies against cross-project transfer, imbalance handling, ensemble type, and XAI, the four dimensions that the present study addresses. Three gaps emerge. As shown, studies that compare all three ensemble types directly do so within a single project. The cross-project studies that use ensemble methods fix one type or embed it in a broader framework. This makes it impossible to isolate the ensemble strategy's contribution. No study uses LOPO on the five-dataset NASA subset with all three strategies under the same protocol. On XAI, studies that report any feature importance do so as a global ranking or a single-instance example; none compare within-project and cross-project rankings for the same model. The stability of LIME explanations across the project border is left entirely unaddressed. Moreover, no study extracts the dataset meta-features to predict which type will transfer best to a given target. Although Chen et al. [10] use meta-features implicitly through AutoML search, they do not examine them as predictors of ensemble suitability. This study focuses on answering that question.

3. METHODS

This paper adopts an empirical software engineering (SE) research design structured around four research questions (RQs) in its study. It covers predictive performance, class-imbalance treatment, explainability consistency, and meta-learning for model selection. The experimental pipeline proceeds through four phases: a within-project baseline, a LOPO cross-project evaluation, an aggregated LIME consistency analysis, and a meta-feature correlation analysis. Fig. 1 presents an overview of the full experimental process. The experiment was conducted on a Windows OS 64-bit system with an AMD Ryzen 8-core processor (4.10 GHz), 32 GB RAM, and a 466 GB hard drive, using a Python virtual environment to ensure reproducibility. The main packages included scikit-learn for ensemble classifier implementation, hyperparameter tuning, and evaluation metrics; imbalanced-learn for SMOTE-based oversampling within the training pipeline; and LIME (lime library) for local feature importance explanation. pandas and NumPy handled data loading, pre-processing, and numerical operations. Matplotlib was used for figure generation and report output. The OS, time, collections, and warnings standard library modules managed file operations, execution timing, and pipeline logging. scipy.stats provided the Spearman rank correlation function used in the LIME consistency and meta-feature analyses. All experiments were run under Python 3.14.3.

3.1. Datasets and pre-processing

Five software defect datasets are used, all sourced from the NASA Metrics Data Program (MDP) through the PROMISE SE Repository. All five share the same Halstead complexity measures, McCabe cyclomatic complexity metrics, and lines-of-code (LOC) features. Table 2 summarizes their features. We employed NASA datasets because they have been widely used by researchers and practitioners and are publicly available for SDP study [18], [19].

Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

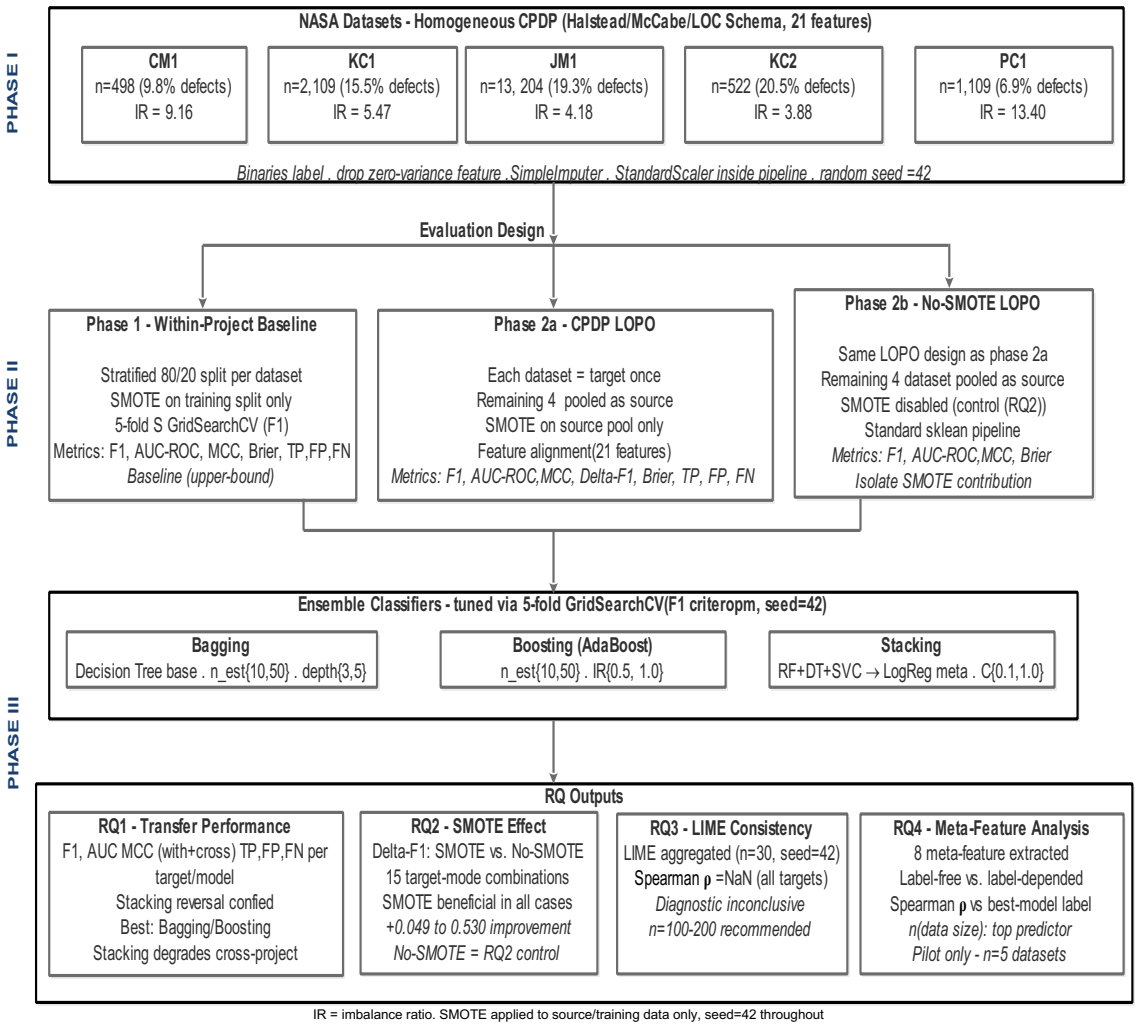

Figure 1. Experimental process

Table 2. Summary of NASA benchmark datasets

Dataset	Project Type	Instances	Features	Defective (%)	Imbalance Ratio
CM1	NASA Spacecraft	498	21	9.8	9.16
KC1	Storage Management	2,109	21	15.5	5.47
JM1	Real-Time System	13,204	21	19.3	4.18
KC2	Storage Management	522	21	20.5	3.88
PC1	Flight Software	1,109	21	6.9	13.40

Note that JM1 is known to contain duplicate module entries. The pipeline loads all rows as supplied by the PROMISE repository without deduplication, resulting in 13,204 instances used in training and evaluation. This is consistent with prior work, Al-Fraihat et al. [5], who reported duplicate entries in this dataset.

The imbalance ratio (IR) for each dataset is defined as the ratio of the majority class (non-defective) instances to the minority class (defective) instances and expressed in Eq.1.

$$\mathrm{IR} = \frac{|\mathrm{N}_0|}{|\mathrm{N}_1|} \tag{1}$$

where N_0 is the set of non-defective instances, and N_1 is the set of defective instances. All datasets are pre-processed uniformly. We removed non-numeric columns, zero-variance features dropped, missing values imputed with zero, and label columns containing string booleans (TRUE/FALSE) or integer defect counts binarized such that any count greater than zero maps to the positive class. Standardisation is applied inside each pipeline fit, not globally, to prevent data leakage across folds or across the source-target boundary.

3.2. Ensemble classifiers

In this study, we used three ensemble methods evaluated as follows: Bagging (with a Decision Tree (DT) base estimator), AdaBoost (Boosting), and Stacking (RF, DT, and SVC as base learners; Logistic Regression (LR) as meta-learner). Each is implemented in an imbalanced-learn pipeline with SMOTE as the first resampling step and a SimpleImputer ahead of the scaler to handle any NaN values that arise after feature alignment in the CPDP pool. The SVC base learner is initialised with 'probability=True' to enable Platt-scaled probability outputs. These are required for the Stacking meta-learner to receive calibrated input features and for LIME to compute prediction probabilities. Moreover, hyperparameters are tuned via five-fold Stratified K-Fold GridSearchCV using F1 as the optimisation criterion. StandardScaler is also applied to all models within the pipeline for consistency, although tree-based classifiers such as Bagging and Stacking's RF and DT base learners are scale-invariant. Uniform scaling ensures no pre-processing difference between models, which confounds the cross-model comparison. Table 3 presents the search space.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Table 3. Hyperparameter search space for each ensemble

Ensemble	Hyperparameter	Values Searched
Bagging	Number of estimators	10, 50
Bagging	Base estimator max depth	3, 5
Boosting	Number of estimators	50, 100
Boosting	Learning rate	0.5, 1.0
Stacking	Meta-learner C (LogReg)	0.1, 1.0

Bagging trains T base classifiers h_1, h_2, …, h_T, each on a bootstrap sample of the source training data. The final prediction for an instance x is determined by majority vote as expressed in Eq. 2.

$$H(x) = \text{argmax}_c \sum_i \mathbb{1}[h_i(x) = c] \quad (2)$$

where c ∈ {0, 1} is the class label, $\mathbb{1}[\cdot]$ is the indicator function, and T bootstrap classifiers h_i each vote for one class.

Boosting, which is the AdaBoost, maintains a weight distribution D_t over training instances. At each iteration t, a weak learner h_t is trained on D_t. Its weight α_t in the final ensemble is computed from its weighted error ε_t:, is expressed in Eq. 3.

$$\varepsilon_t = \sum_i D_t(i) * \mathbb{1}[y_i \neq h_t(x_i)] \quad (3)$$

where $D_t(i)$ is the weight of instance i at iteration t, y_i is the true label, and $h_t(x_i)$ is the weak learner's prediction.

$$\alpha_t = ½ * \ln((1 - \varepsilon_t) / \varepsilon_t) \quad (4)$$

The instance weights are then updated: $D_{\{t+1\}}(i) = D_t(i) * \exp^{(-\alpha_t * y_i * h_t(xi))}$, normalised so that $\sum_i D_{\{t+1\}}(i) = 1$. The final prediction is the sign of the weighted sum of weak learner outputs: $h(x) = \text{sign}(\sum_t \alpha_t * h_t(x))$.

In the same vein, stacking trains K diverse base classifiers on the source training data using five-fold cross-validation and uses their out-of-fold predictions as input features for a meta-learner. For instance, x, the meta-learner input is given by:

$$z(x) = [h_1(x), h_2(x), \dots, h_k(x)] \quad (5)$$

where $h_k(x)$ is the predicted probability from base classifier k. The meta-learner f (LR) produces the final output: $\hat{y} = f(z(x))$.

Because the meta-learner is trained on out-of-fold predictions from the source pool, its combination weights are calibrated to the source distribution. This calibration is strong within the source but may not transfer to a target project with a different marginal distribution, which is the mechanism behind the cross-project degradation observed in Section 4.

3.3. Class imbalance handling

SMOTE addresses class imbalance by generating synthetic minority instances through linear interpolation in feature space [4], [11], [12], [15], [16]. For a minority instance x_i, a synthetic sample $x_{synthetic}$ is generated as:

$$x_{Synthetic} = [x_i + \lambda * (x_{neigbhour} - x_i)] \quad (6)$$

where $x_{neighbour}$ is a randomly selected minority-class nearest neighbour of x_i and $\lambda \in [0, 1]$ is drawn uniformly at random.

We applied SMOTE completely to the training data in both evaluation settings. In the within-project setting, SMOTE is fitted on the 80% training split. In the LOPO setting, SMOTE is fitted on the pooled source data. The target dataset is never resampled. This is enforced structurally, where SMOTE is a pipeline step fitted only during training, so it cannot access test or target instances.

3.4. Within and cross-project LOPO evaluation

We split each dataset into 80/20 using stratified sampling. GridSearchCV tunes hyperparameters on the training split, while the best estimator is evaluated on the held-

out test set. We performed tuning independently for each evaluation setting. In within-project tuning, the 80% training split of each dataset. In the LOPO setting, tuning uses the full pooled source pool for each target. In both cases, the tuned configuration is applied to the held-out test set or target, respectively, with no parameter sharing between settings. To assess the predictive performance in both settings, four metrics are reported. The main metric is F1, which is the harmonic mean of precision and recall on the defective class [3], [5]:

$$\mathrm{F1} = \frac{2\times \mathrm{precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{7}$$

where Precision = TP / (TP + FP) and Recall = TP / (TP + FN), computed with respect to the defective (positive) class.

The secondary metric is the Matthews Correlation Coefficient (MCC), which offers a single balanced measure across all four confusion matrix cells and is robust to class imbalance.

$$MCC = \frac{\mathrm{TP}*\mathrm{TN}-\mathrm{FP}*\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{8}$$

where TP, TN, FP, and FN are true positives, true negatives, false positives, and false negative respectively. MCC returns a value -1 and +1, where +1 indicates perfect prediction, 0 indicates performance no better than random, and -1 indicates complete disagreement between prediction and observation [4], [23].

Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is also reported. It measures the probability that the model ranks a randomly chosen defective instance above a randomly chosen non-defective one, independent of the decision threshold [3]:

$$\mathrm{AUC} = \mathrm{P}(\, \mathrm{f}(\mathrm{x}^{+}) > \mathrm{f}(\mathrm{x}^{-})) \tag{9}$$

where x^{+} is a defective instance, x^{-} is a non-defective instance, and $f(\cdot)$ denotes the model's predicted probability of the positive class.

The Brier score (BS) is also reported in this study. It measures probability calibration, particularly, the mean squared difference between the predicted probability of the positive class and the actual binary outcome [28].

$$\mathrm{BS} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i=i}}^{\mathrm{N}} (\mathrm{f}(\mathrm{x_i}) - \mathrm{y_i})^2 \tag{10}$$

Where $f(x_i)$ is the model's predicted probability of the defective class for instance i, $y_i \in \{0,1\}$ is the true label, and N is the number of instances. It ranges from 0 to 1, with lower values indicating better-calibrated probability estimates. A score of 0 means perfect calibration and 1 denotes worst-case miscalibration [28]. Unlike F1 and MCC, which assess classification decisions at a fixed threshold, BS assesses the quality of the main probability estimates independent of any threshold. We included it in this study to distinguish between threshold miscalibration, where probabilities are well-estimated, but the decision boundary is misplaced, and probability miscalibration, where the predicted probabilities themselves are inaccurate. However, we excluded accuracy as a primary metric due to the presence of class imbalance in all datasets. This is because a classifier that always predicts non-defective achieves accuracy between 80 and 93 percent across the five datasets while producing zero correct defect detections. We did not include Precision-Recall Area Under the Curve (PR-AUC) as a primary metric because MCC already provides a balanced measure across all four confusion matrix cells. Reporting both would introduce redundancy without additional interpretive value.

In terms of cross-project based on LOPO, each dataset serves as the target exactly once. The remaining four datasets are pooled as the source. Features are aligned by retaining the intersection of numeric columns common to both pool and target. Given the shared NASA schema, this intersection is the full 21-feature set in all cases. SMOTE is applied to the source pool while the target is untouched. Hyperparameter tuning uses five-fold StratifiedKFold on the source pool. The tuned model is assessed on all instances of the target dataset.

To quantify the change in predictive performance between within-project and cross-project settings, the F1 drop and delta-F1 (ΔF1) are computed for each dataset-model combination:

$$\Delta F1 = F1_{within} - F1_{cross} \quad (11)$$

In this case, a positive ΔF1 indicates degradation under cross-project transfer, while a negative ΔF1 indicates that cross-project performance exceeds the within-project baseline. Two delta measures are used throughout. $\Delta F1_{transfer}$ = $F1_{cross}$ - $F1_{within}$ quantifies transfer degradation relative to the within-project baseline (where negative = degradation, positive = improvement). $\Delta F1_{SMOTE}$ = $F1_{SMOTE}$ - $F1_{noSMOTE}$ quantifies the contribution of source-only SMOTE to cross-project performance (always positive in this study).

3.5. LIME consistency analysis

In terms of XAI, LIME approximates the behaviour of a black-box model f in the local neighbourhood of a given instance x by fitting a surrogate linear model g that minimises the following objective [28]:

$$\xi(x) = \operatorname{argmin}_{\{g \in G\}} \ell(f, g, \pi_x) + \Omega(g) \quad (12)$$

where G is the class of linear models, ℓ measures the fidelity of g to f in the neighbourhood defined by proximity measure π_x, and Ω(g) penalises model complexity. The signed coefficient of each feature in g constitutes its local importance weight.

In this study, LIME explanations are generated for thirty randomly selected test instances per model per setting. Feature weights are averaged across the thirty explanations, and features are ranked by absolute mean weight. We used a sample of thirty instances selected uniformly at random from the full test set, without stratification by predicted class. This is because single-instance LIME explanations are unstable since slight input perturbations can shift feature rank orders greatly. A fixed random seed of 42 was applied throughout all experiments, including dataset splitting, SMOTE sampling, model initialization, and LIME instance selection, to ensure reproducibility. The same thirty instances were used consistently across all three ensemble models for each dataset, to allow direct comparison of feature rank vectors. The choice of thirty instances is consistent with the minimum sample recommended for aggregated local explanation studies in previous XAI literature and represents a pragmatic balance between

explanation stability and computational cost. A sensitive analysis using larger samples(100 and 200 instances) is identified as future work.

To further quantify explanation stability between the within-project and cross-project settings, Spearman's rank correlation coefficient (ρ) is computed between the two feature rank vectors for each dataset-model combination as expressed in Eq. 13.

$$\rho = 1 - \frac{6 \sum d_i^2)}{n*(n^2 - 1)} \quad (13)$$

where d_i is the difference between the rank of feature i in the within-project ranking and its rank in the cross-project ranking, and n is the total number of features. $\rho \in [-1, 1]$: values near +1 indicate stable feature attribution across the project boundary; values near 0 or −1 indicate divergent attribution.

3.6. Meta-feature analysis

Eight meta-features are extracted from each dataset before any model is trained. These are statistics that characterize the distributional properties of a dataset and are available as a priori predictors without requiring any labelled target data. **Table 4** presents each meta-feature.

Table 4. Dataset meta-features used in analysis

Meta-Feature	Definition	Formula / Notes
n	Number of instances	Count of rows after pre-processing
p	Number of features	Count of numeric columns retained
IR	Imbalance ratio	$\|N_0\| / \|N_1\|$ (Eq. 1)
DR	Defect rate	$\|N_1\| / (\|N_0\| + \|N_1\|)$
Skewμ	Mean feature skewness	μ of skewness across all p features
Skewσ	Std. dev. of skewness	σ of skewness across all p features
Kurtμ	Mean feature kurtosis	μ of excess kurtosis across all p features
Corrμ	Mean pairwise correlation	μ of |Pearson r| over all p(p−1)/2 feature pairs

The best-performing ensemble for each target project is identified from the LOPO F1 results. Spearman's ρ (Eq. 13) is then computed between each meta-feature and the best-model label, encoded as an ordinal variable (Bagging = 0, Boosting = 1, Stacking = 2). A meta-feature with a high absolute ρ is a candidate predictor of ensemble suitability for an unseen target project. Given n = 5 datasets, we then treated the analysis as a pilot identification of candidate predictors rather than a statistically validated meta-model. A Spearman correlation computed over five data points has very low statistical power, and the findings are presented as a hypothesis for validation in future work with larger dataset collections.

4. RESULTS AND DISCUSSION

This section presents the experimental results for all RQs following the method outlined in Fig. 1. We reported the findings for metrics F1, MCC, and AUC-ROC throughout. Moreover, the Brier score is reported separately in the discussion section to support the calibration analysis. Also, TP, FP, and FN counts for the cross-project setting are provided in Table 8 to show the type of transfer error made by each ensemble. LIME explanations were generated using thirty test instances selected uniformly at random from the full test set. This includes both defective and non-defective instances, without filtering by true or predicted class label.

4.1. Within-project baseline and cross-project LOPO

Table 5 reports F1, AUC-ROC, and MCC for each ensemble on each dataset in the within-project setting. Bold values indicate the best F1 per dataset. As shown in Table 5, stacking achieves the highest F1 and MCC on four of five datasets, confirming its within-project advantage. The MCC values are consistent with the F1 rankings. Stacking leads on CM1 (MCC=0.0278), KC1 (0.406), and PC1 (0.447). On KC2, Bagging and Boosting tie at F1 (0.500) and MCC (0.355), while Stacking tails at F1=0.359 and MCC=0.218. AUC-ROC values are consistently higher than F1 across all models, which is expected given the class imbalance. PC1 achieves the highest AUC values (up to 0.870 for Bagging) despite its low defect rate of 6.9 percent. This suggests that the Halstead and McCabe features are particularly descriptive for that project. CM1 shows the lowest F1 scores overall, consistent with its small size and severe imbalance.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Table 5. Within-project results

Dataset	Bag F1	Bag AUC	Bag MCC	Bst F1	Bst AUC	Bst MCC	Stk F1	Stk AUC	Stk MCC
CM1	0.256	0.621	0.154	0.176	0.607	0.047	**0.357**	**0.648**	**0.278**
KC1	0.455	0.805	0.349	0.404	0.777	0.282	**0.503**	**0.822**	**0.406**
JM1	0.397	0.725	0.262	0.406	0.734	0.270	**0.413**	**0.776**	**0.318**
KC2	**0.500**	**0.721**	**0.355**	**0.500**	0.707	**0.355**	0.359	0.667	0.218
PC1	0.386	0.870	0.374	0.386	0.857	0.374	**0.483**	**0.848**	**0.447**

**Bag = Bagging, Bst = Boosting, Stk = Stacking

Table 6 reports F1, AUC-ROC, and MCC for each ensemble under LOPO evaluation, answering RQ1. Bold values indicate the best cross-project F1 per target.

Table 6. LOPO cross-project F1, AUC-ROC, MCC, and $\Delta F1_{transfer}$ for the best cross-project model

Target	Bag F1	Bag AUC	Bag MCC	Bst F1	Bst AUC	Bst MCC	Stk F1	Stk AUC	Stk MCC	Best $\Delta F1_{transfer}$
CM1	**0.327**	0.731	**0.243**	0.312	**0.737**	0.226	0.255	0.659	0.170	+0.071 (Bag)
KC1	0.416	0.679	0.300	**0.422**	**0.749**	**0.310**	0.247	0.691	0.141	+0.018 (Bst)
JM1	0.307	0.641	0.124	**0.332**	0.636	**0.161**	0.254	**0.590**	0.126	-0.074 (Bst)
KC2	**0.618**	**0.845**	**0.514**	0.602	0.828	0.492	0.425	0.796	0.320	+0.118 (Bag)
PC1	0.225	0.739	0.162	**0.258**	**0.729**	**0.202**	0.257	0.680	0.194	-0.128 (Bst)

*Bold = best cross-project F1 per target. **$\Delta F1_{transfer}$ = $F1_{cross}$ - $F1_{within}$ for the best cross-project model, ***positive = improvement over within-project baseline, ****negative = degradation.

As presented, Bagging performs best on CM1 (F1=0.327, MCC=0.243) and KC2 (F1=0.618, MCC=0.514), improving from 0.256 to 0.327 on CM1 and from 0.500 to 0.618 on KC2, respectively. Boosting performs best on KC1 (F1=0.422, MCC=0.310), JM1 (F1=0.332, MCC=0.161), and PC1 (F1=0.258, MCC=0.202). Stacking performs worst or near-worst on every target. The largest single drop is Stacking on KC1 ($\Delta F1_{transfer}$ = -0.256), from 0.503 within-project to 0.247 cross-project, and from MCC=0.406 to MCC=0.141. MCC confirms the F1 rankings throughout: no target shows a discrepancy between the best model by F1 and the best model by MCC. KC2 is the only target where all three ensembles improve over the within-project baseline. Fig. plots the mean F1 by ensemble type for both settings, with error bars showing standard deviation across the five datasets. Bagging and Boosting transfer more reliably than Stacking under LOPO evaluation. Stacking's within-project dominance does not carry over to the cross-project setting, a pattern confirmed by both F1 and MCC.

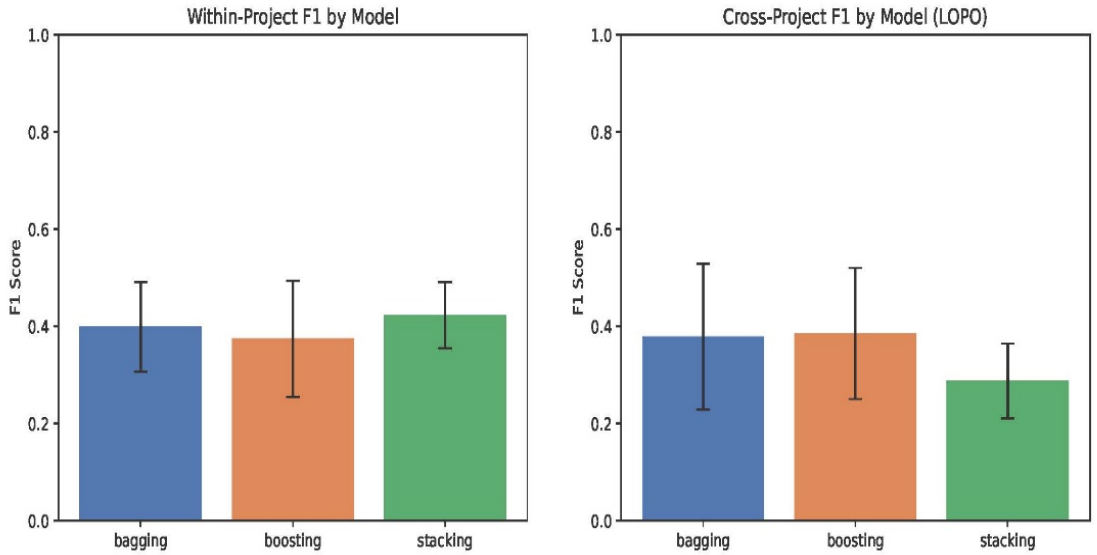

Figure 2. Ensemble generalization

4.2. Effect of SMOTE on cross-project transfer and error analysis

This subsection answered RQ2, and Table 7 compares cross-project F1 with and without SMOTE applied to the source pool. The no-SMOTE condition provides a direct control that isolates the contribution of source-only SMOTE to LOPO performance. SMOTE applied exclusively to the pooled source data improves cross-project F1 for every target-model combination without exception. $\Delta F1_{SMOTE}$ ranges from +0.049 (JM1/Bagging) to +0.530 (KC2/Boosting). The effect is largest on KC1 and KC2, where no-SMOTE Boosting produces near-zero F1 (0.006 and 0.072, respectively), indicating that without SMOTE the model learns almost exclusively from the majority class and fails to detect defects in the target. JM1 shows the smallest SMOTE benefit for Bagging (+0.049), consistent with its large size (13,204 instances) already providing an adequate majority-class signal without synthetic augmentation. Fig. 3 shows the full $\Delta F1_{SMOTE}$ distribution across all 15 target-model combinations; green bars indicate improvement over no-SMOTE.

Table 7. SMOTE VS. no-SMOTE Cross-project F1 under LOPO

Target	Model	SMOTE F1	No-SMOTE F1	$\Delta F1_{SMOTE}$
CM1	Bagging	0.327	0.209	+0.118
CM1	Boosting	0.312	0.038	+0.273
CM1	Stacking	0.255	0.129	+0.126

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Target	Model	SMOTE F1	No-SMOTE F1	$\Delta F1_{SMOTE}$
KC1	Bagging	0.416	0.042	+0.375
KC1	Boosting	0.422	0.006	+0.416
KC1	Stacking	0.247	0.046	+0.201
JM1	Bagging	0.307	0.258	+0.049
JM1	Boosting	0.332	0.168	+0.163
JM1	Stacking	0.254	0.117	+0.137
KC2	Bagging	0.618	0.122	+0.496
KC2	Boosting	0.602	0.072	+0.530
KC2	Stacking	0.425	0.138	+0.287
PC1	Bagging	0.225	0.115	+0.110
PC1	Boosting	0.258	0.051	+0.207
PC1	Stacking	0.257	0.178	+0.079

$\Delta F1_{SMOTE} = F1_{SMOTE} - F1_{noSMOTE}$. All positive values indicate SMOTE improves transfer.

The finding of RQ2, therefore, shows source-only SMOTE consistently improves cross-project transfer across all targets and ensembles. Without SMOTE, most models fail to detect defective modules in the target project. The effect is largest where the target's minority class is most underrepresented relative to the pooled source pool.

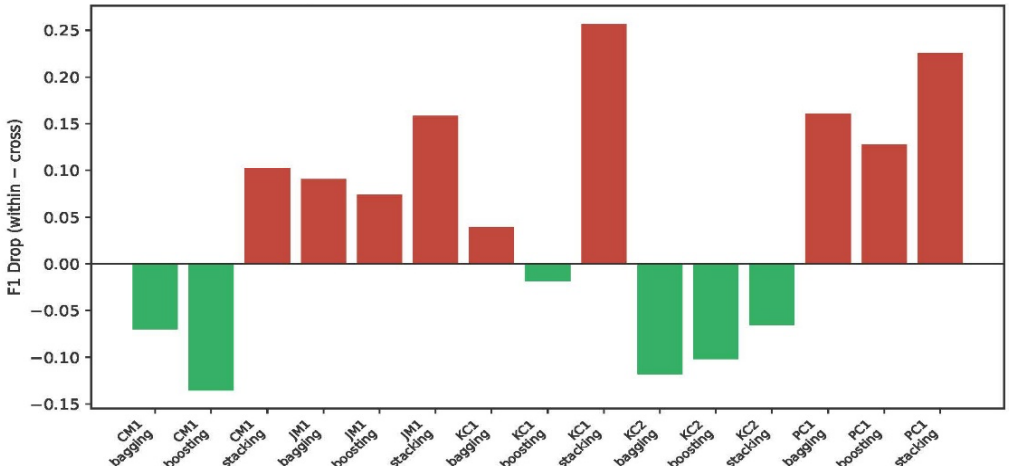

Figure 3. $\Delta F1_{SMOTE}$ distribution across all dataset-model combinations

Table 8. TP, FP, FN, and error ratios under LOPO cross-project evaluation

Target	Model	TP	FP	FN	FP/TP Ratio	FN/TP Ratio
CM1	Bagging	24	74	25	3.08	1.04
CM1	Boosting	24	81	25	3.38	1.04
CM1	Stacking	13	40	36	3.08	2.77
KC1	Bagging	148	237	178	1.60	1.20
KC1	Boosting	146	220	180	1.51	1.23
KC1	Stacking	68	157	258	2.31	3.79
JM1	Bagging	1112	4032	991	3.62	0.89
JM1	Boosting	1303	4455	800	3.42	0.61
JM1	Stacking	496	1303	1607	2.63	3.24
KC2	Bagging	85	83	22	0.98	0.26
KC2	Boosting	84	88	23	1.05	0.27
KC2	Stacking	38	34	69	0.89	1.82
PC1	Bagging	34	191	43	5.62	1.26
PC1	Boosting	36	166	41	4.61	1.14
PC1	Stacking	27	106	50	3.93	1.85

FP/TP = false positive rate relative to true positives; FN/TP = miss rate relative to detections.

We further analysed the transfer errors, and Table 8 reports TP, FP, and FN for each ensemble on each target under LOPO evaluation. In this study context, FN represents missed defects, which is the higher-cost error in defect prediction, while FP represents clean modules incorrectly flagged for inspection. It shows that Stacking consistently produces the highest FN/TP ratios, which indicate it misses proportionally more defects than it detects on KC1 (FN/TP=3.79), JM1 (3.24), and CM1 (2.77). Conversely, Bagging and Boosting maintain FN/TP ratios below 1.3 on KC1 and JM1, showing they detect more defects than they miss. KC2 is again the exception because both Bagging (FN/TP=0.26) and Boosting (0.27) achieve strong recall across projects, detecting about four defective modules for every one missed. The FP/TP ratios are high across all models on PC1 (3.93

to 5.62), reflecting the low defect rate of 6.9 percent and the difficulty of precisely identifying the small defective minority in that project.

4.3. LIME explanation consistency

This subsection answered RQ3, and Table 9 presents the Spearman rank correlation (ρ) between within-project and cross-project LIME feature rank vectors for each target dataset and ensemble.

Table 9. Spearman ρ LIME feature rank vectors

Target	Bagging ρ	Boosting ρ	Stacking ρ
CM1	NaN	NaN	NaN
KC1	NaN	NaN	NaN
JM1	NaN	NaN	NaN
KC2	NaN	NaN	NaN
PC1	NaN	NaN	NaN

NaN = rank correlation undefined due to tied ranks in aggregated weight vectors

The findings show that all values are NaN because the LIME aggregation produced insufficient variance in the feature weight vectors to compute a meaningful rank correlation. That is, when the mean LIME weights across sample size (n=30) converge to near-zero for most features, the rank vectors are effectively tied, and Spearman ρ is undefined. This finding differs from the near-zero interpretation because NaN (Not a number) is a stronger result. It indicates not that rankings are approximately the same, but that thirty-instance LIME aggregation with 21 features produces weight vectors in which most features receive near-zero mean weights. This collapses the rank structure entirely. The Spearman ρ diagnostic cannot be computed under these conditions and therefore cannot serve as a deployment-time trustworthiness filter at this aggregation sample size. Therefore, RQ3 findings suggest that LIME rank correlation is undefined for all models and targets at thirty-instance aggregation. This indicates that the aggregation sample is insufficient to produce stable, non-degenerate rank vectors for 21-feature datasets. The diagnostic requires a larger aggregation sample before it can be applied.

4.4. Meta-feature predictors of ensemble suitability

This subsection presents the findings of RQ4. Table 10 reports the best-performing cross-project ensemble per target by F1 and MCC under the LOPO pipeline, alongside the main

meta-features. Label-free meta-features, i.e., computable without defect labels, are marked with †.

Table 10. Best-performing ensemble per target project and meta-features

Target	Best (F1)	Best (MCC)	n†	IR	DR	feat_corr†	skew_mean†
CM1	Bagging	Bagging	498	9.16	9.8%	0.681	5.44
KC1	Boosting	Boosting	2,109	5.47	15.5%	0.720	4.20
JM1	Boosting	Boosting	13,204	5.28	15.9%	0.587	16.26
KC2	Bagging	Bagging	522	3.88	20.5%	0.775	8.95
PC1	Boosting	Boosting	1,109	13.40	6.9%	0.633	8.01

† = label-free (computable on unlabelled target). IR = imbalance ratio, DR = defect rate

In addition, Fig. 4 plots the Spearman rank correlation between each meta-feature and the best-model label, showing n_samples and IR as the strongest positive predictors.

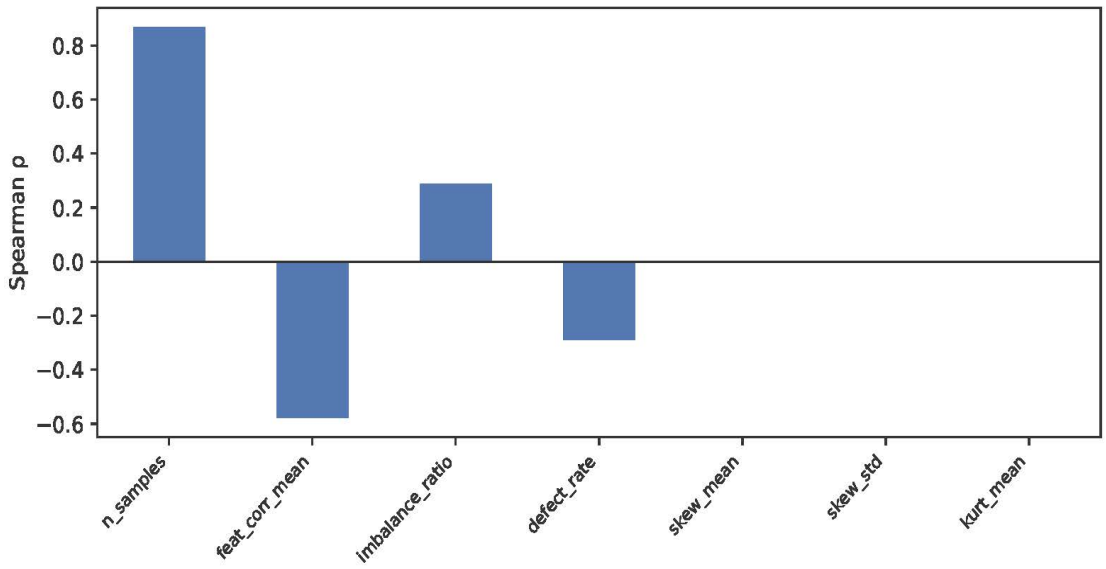

Figure 4. Meta-feature correlation with the best ensemble

As presented, Boosting is the best ensemble on PC1 with MCC=0.202. F1 also favours Boosting on PC1 (0.258 vs 0.257 for Stacking), though the margin is negligible. Bagging is best on the two smaller, more severely imbalanced targets (CM1: IR=9.16, n=498; KC2:

IR=3.88, n=522), while Boosting is best on the three larger datasets with moderate-to-high imbalance (KC1, JM1, PC1). A practical distinction relevant to deployment is that IR and DR require labelled target data and cannot be computed before deployment. Dataset size (n) and feature correlation (feat_corr) are label-free and available on any unlabelled target dataset. The pattern that Bagging is preferred for smaller datasets (n < 600) and Boosting for larger ones (n > 1,000) is observable from label-free meta-features alone. This is directly actionable for practitioners who must select an ensemble before any target labels are available. Overall, RQ4 finding shows that Bagging performs best on smaller, more severely imbalanced targets. Conversely, Boosting performs best on larger datasets with moderate imbalance. Thus, n, a label-free meta-feature, is the most practically useful preliminary predictor of ensemble suitability for CPDP deployment.

4.5. Discussion

This paper compares ensemble methods such as Bagging, Boosting, and Stacking directly under an LOPO protocol. This section interprets the findings and situates them against the literature. The analysis reveals Stacking's within-project F1 advantage is real. On four of the five datasets, it leads on both F1 and MCC, but the cross-project results show a different picture entirely. Three mechanisms may plausibly explain the reversal. The Brier scores in Table 11 show Stacking retains well-calibrated probabilities under LOPO. That is, on KC1, its cross-project Brier score (0.152) equals Bagging's, yet its F1 is 0.175 lower. This gap between calibration and classification performance points to threshold miscalibration. The meta-learner learns to weight base-learner outputs based on source-pool patterns. The threshold does not relocate correctly when the target has a different class distribution. A second mechanism is base-learner disagreement. In this case, the RF, DT, and SVC base learners each track different aspects of the source distribution, and distributional shift may produce disagreements that the meta-learner was never trained to resolve. Third, the LOPO source pool is itself a mixture of four projects with different defect rates. As a result, the meta-learner fits weights to a distribution that may poorly approximate any single target. Bagging sidesteps the first problem because it aggregates by majority vote, with no learned combination step that can drift. Boosting's instance reweighting concentrates on defective modules, which carry transferable signal regardless of project origin.

Table 11. Brier scores for within- and cross-project settings

Setting	Model	CM1	KC1	JM1	KC2	PC1
Within	Bagging	0.204	0.153	0.184	0.172	0.098
Within	Boosting	0.198	0.182	0.225	0.177	0.179
Within	Stacking	0.146	0.120	0.127	0.161	0.056
Cross	Bagging	0.217	0.152	0.228	0.157	0.172
Cross	Boosting	0.220	0.193	0.236	0.213	0.215
Cross	Stacking	0.120	0.152	0.161	0.142	0.109

As shown in Table 11, the Brier scores with a lower score signify better calibration. Stacking maintains good probability calibration across projects but fails to convert this into a high F1, pointing to threshold miscalibration. Xin et al. [4] recommended boosting for within-project settings; the present results suggest the recommendation extends to cross-project transfer.

As presented in Table 7, source-only SMOTE improves cross-project F1 for every one of the 15 target-model combinations tested, from +0.049 (JM1/Bagging) to +0.530 (KC2/Boosting). To this end, SMOTE did not harm the transfer to PC1. On the other hand, no-SMOTE F1 on PC1 ranges from 0.051 to 0.178, well below the SMOTE values of 0.225 to 0.257. PC1's cross-project degradation relative to its within-project baseline is caused by the distributional gap, not by oversampling. The SMOTE benefit scales with minority-class underrepresentation in the source pool: KC2 and KC1, where no-SMOTE Boosting yields near-zero F1, and gains the most. JM1 gains the least (+0.049 for Bagging), which is consistent with its 13,204 instances already supplying an adequate majority-class signal without synthetic augmentation. This is a factor calculable before any model is trained. Rashmi and Kaur [29] observed a related inconsistency across projects in cross-project vulnerability prediction. The defect-rate ratio provides a concrete candidate explanation for such a finding.

Across the experiments, the NaN Spearman ρ results do not mean rankings are stable. They mean the aggregation produced degenerate rank vectors. Across n=30 and 21 features, mean LIME weights converge to near zero for most features, leaving no usable rank order to correlate. A larger sample, about 100 to 200 instances, or a count of how often each feature appears in the top five explanations, would produce computable

vectors. The permutation importance plots show LOC, iv(g), and LOCode as the dominant features across all datasets and models. If those features absorb most LIME weight across both settings but inconsistently across individual instances, the n=30 average will wash out to near zero. The finding that Stacking's F1 collapse leaves no trace in LIME attributions, taken alongside the Brier score results (Table 11), supports threshold miscalibration as the primary mechanism. The model shifts its decision boundary toward the source distribution without changing which features it considers informative.

In terms of the meta-feature, Bagging is best on the two smallest targets (CM1: n=498, KC2: n=522). Boosting is best on the three largest (KC1: n=2,109, JM1: n=13,204, PC1: n=1,109). This shows that dataset size is the only label-free meta-feature that cleanly separates the two. A provisional pattern of n < 600 associating with Bagging and n > 1,000 with Boosting is observed in this data. However, this should not be treated as a validated rule; it is based on five datasets and serves only as a hypothesis for future testing. IR adds signal but requires labels. In a fully unlabelled deployment, the dataset size is what a practitioner can act on before predictions are available.

4.5.1. Comparison with existing studies

Table 12 positions this study against the most directly comparable prior work. Within-project F1 scores (0.176 to 0.503) are consistent with comparable studies using NASA datasets. Within-project AUC-ROC values observed here, ranging from 0.721 to 0.870, are broadly consistent with prior ensemble studies on the same datasets [2], [14]. These comparisons are indicative rather than definitive because evaluation protocols, dataset splits, and feature pre-processing differ across studies. Although direct comparison is limited by differences in evaluation protocol, both studies use within-project cross-validation with per-dataset feature optimisation, whereas this study uses a fixed feature set and LOPO evaluation. The lower AUC values observed in the cross-project setting (0.590 to 0.683) reflect the additional difficulty of generalising across project boundaries. The cross-project results are harder to compare directly because most prior CPDP studies use all-pairs evaluation on different dataset collections. Tong et al. [12] reported F1 values that vary considerably across 34 datasets. Here, KC2's cross-project Bagging F1 of 0.618 falls at the stronger end of that range, consistent with KC2 being a well-aligned transfer target. The most directly comparable finding concerns ensemble type rankings. Xin et al. [4] found Boosting to be the most efficient ensemble on their NASA and

PROMISE datasets. These current results extend that finding to the cross-project case. Stacking's reversal was not tested in [4] and was not foreseeable from the within-project rankings alone. No prior study has paired LOPO evaluation of all three ensemble types with a no-SMOTE control, aggregated LIME analysis, and a meta-feature pilot on a shared NASA MDP schema.

Table 12. Comparison with related studies

Study	Setting	Datasets	Best Ensemble	Explainability
Xin et al. [4]	Within-project	4 NASA/PROMISE	Boosting	None
Chen et al. [14]	Within-project	8 NASA MDP	Stacking (2-layer)	None
Misbah et al. [2]	Within-project	7 NASA MDP	Voting	None
Tong et al. [12]	Cross-project	34 mixed	Boosting (tr.)	None
Wang et al. [21]	Cross-project	Multiple	Boosting (ad.)	None
This study	Within + LOPO	5 NASA MDP	Bagging/Boosting	LIME (agg.)

tr. = transfer variant; ad. = adapted; agg. = aggregated over 30 instances

4.5.2. Threats to validity

We performed hyperparameter tuning using cross-validation on training data only, and test data was never used during model selection. SMOTE is embedded inside the pipeline and fitted exclusively on source data, preventing resampling-based leakage into the target. A SimpleImputer precedes SMOTE in the pipeline so that NaN values arising from feature alignment after pooling cannot reach the resampler or the classifier. One internal threat remains: the LOPO evaluation uses a fixed source-target partition rather than repeated random subsampling of the source pool. Results, therefore, reflect one specific pooling arrangement per target. A no-SMOTE baseline was added to isolate the contribution of source-only SMOTE. This partially addresses this concern, but sensitivity to the composition of the source pool, for example, weighting or filtering source datasets by similarity to the target, was not tested.

In terms of external validity, all five datasets come from NASA-embedded and aerospace software. Findings may not transfer to open-source or commercial projects from other domains. The shared Halstead and McCabe schema removes feature heterogeneity as a confound but simultaneously means the study does not cover heterogeneous CPDP, where source and target use different metric sets. Replications with the Jureczko CK-metrics collection, AEEEM, or GitHub-derived defect datasets are, therefore, needed before the results can be treated as general.

For construct validity, we used F1 as the primary metric, with AUC-ROC, MCC, and Brier score as complementary measures. None of these accounts for the asymmetric cost of false negatives versus false positives in defect prediction; a cost-sensitive metric such as Pofb would give a more deployment-relevant evaluation but requires cost estimates unavailable for these datasets. LIME explanations were aggregated over thirty test instances per model. The RQ3 result, undefined Spearman ρ for all models and targets, indicates that n=30 is insufficient to produce non-degenerate rank vectors for 21 features. Whether this reflects genuine feature stability across projects or noise in the aggregated weights cannot be determined from the current design. Aggregation samples of 100 to 200 instances are recommended to test the diagnostic at a larger scale. Furthermore, JM1 is known to contain duplicate module entries, which were retained as supplied by the PROMISE repository. No sensitivity analysis was conducted to test whether deduplication would alter the results. This is acknowledged as a limitation, and replication on a deduplicated version of JM1 is recommended.

In terms of conclusion validity, with five datasets, inferential statistical tests such as the Wilcoxon signed-rank test are not appropriate, as the sample size is too small for reliable non-parametric comparison. Results are reported descriptively. The RQ4 meta-feature analysis is a pilot study. In this case, any causal interpretation of the correlations observed would require a larger dataset collection and statistical controls.

5. CONCLUSION

This paper compares Bagging, Boosting, and Stacking under an LOPO design on five NASA MDP datasets. We found a consistent ranking reversal as follows. Stacking leads within-project but transfers poorly, while Bagging and Boosting are more reliable across projects. A no-SMOTE control confirms that source-only SMOTE improves transfer in all 15 target-model combinations. In addition, LIME consistency analysis produces undefined Spearman ρ for all models and targets, meaning explanation stability remains unresolved at the current thirty-instance aggregation level. Dataset size shows a provisional association with ensemble suitability. That is, smaller targets favour Bagging, larger ones Boosting, but this pattern is based on five datasets and should not be treated as a confirmed rule. Three research directions follow this study. First, the LOPO experiment should be replicated with the Jureczko CK-metrics collection and the AEEEM datasets to

test whether the Stacking reversal holds outside the NASA MDP schema. Second, LIME aggregation at 100 to 200 instances to determine whether the diagnostic is viable at a larger scale. Third, a meta-learning study with fifteen or more target projects to validate dataset size and imbalance ratio as predictors of ensemble suitability.

ACKNOWLEDGMENT

This research was supported by the UDSC, the Department of Computer Science at the North-West University Mafikeng campus, South Africa.

REFERENCES

[1] M. Ali, T. Mazhar, A. Al-Rasheed, T. Shahzad, Y. G. Yasin, and M. A. Khan, "Enhancing software defect prediction: A framework with improved feature selection and ensemble machine learning," *PeerJ Comput. Sci.*, vol. 10, p. e1860, 2024, doi: 10.7717/peerj-cs.1860.

[2] M. Ali et al., "Software Defect Prediction Using an Intelligent Ensemble-Based Model," *IEEE Access*, vol. 12, pp. 20376–20395, 2024, doi: 10.1109/ACCESS.2024.3358201.

[3] B. Isong and E. Igo, "Ensemble Learning for Software Defect Prediction: Performance, Practicality and Future Directions," *Journalisi*, vol. 7, no. 3, pp. 2245–2291, Sep. 2025, doi: 10.51519/journalisi.v7i3.1171.

[4] X. Dong, Y. Liang, S. Miyamoto, and S. Yamaguchi, "Ensemble learning based software defect prediction," *J. Eng. Res.*, vol. 11, no. 4, pp. 377–391, 2023.

[5] D. Al-Fraihat, Y. Sharrab, A.-R. Al-Ghuwairi, H. Alshishani, and A. Algarni, "Hyperparameter Optimization for Software Bug Prediction Using Ensemble Learning," *IEEE Access*, vol. 12, pp. 51869–51878, 2024, doi: 10.1109/ACCESS.2024.3380024.

[6] A. Vescan, R. Găceanu, and C. Şerban, "Exploring the impact of data preprocessing techniques on composite classifier algorithms in cross-project defect prediction," *Autom. Softw. Eng.*, vol. 31, p. 47, 2024, doi: 10.1007/s10515-024-00454-9.

[7] N. Nikravesh and M. R. Keyvanpour, "*Cross-project Defect Prediction with an Enhanced Transfer Boosting Algorithm*," in Proc. 12th Int. Conf. Comput. Knowl. Eng. (ICCKE), Mashhad, Iran, 2022, pp. 157–162, doi: 10.1109/ICCKE57176.2022.9960103.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[8] T. Asano et al., "*Using Bandit Algorithms for Project Selection in Cross-Project Defect Prediction*," in Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME), Luxembourg, 2021, pp. 649–653, doi: 10.1109/ICSME52107.2021.00074.

[9] S. Zheng, J. Gai, H. Yu, H. Zou, and S. Gao, "Training data selection for imbalanced cross-project defect prediction," *Comput. Electr. Eng.*, vol. 94, p. 107370, 2021.

[10] J. Chen, J. Ding, K. C. Tan, J. Qian, and K. Li, "MBL-CPDP: A Multi-Objective Bilevel Method for Cross-Project Defect Prediction," *IEEE Trans. Softw. Eng.,* vol. 51, no. 8, pp. 2305–2328, Aug. 2025, doi: 10.1109/TSE.2025.3577808.

[11] H. Tong et al., "MASTER: Multi-Source Transfer Weighted Ensemble Learning for Multiple Sources Cross-Project Defect Prediction," IEEE Trans. Softw. Eng., vol. 50, no. 5, pp. 1281–1305, May 2024, doi: 10.1109/TSE.2024.3381235.

[12] H. Tong, W. Lu, W. Xing, and S. Wang, "ARRAY: Adaptive triple feature-weighted transfer Naive Bayes for cross-project defect prediction," *J. Syst. Softw.,* vol. 202, p. 111721, 2023.

[13] O. P. Omondiagbe, S. A. Licorish, and S. G. MacDonell, "Improving transfer learning for software cross-project defect prediction," *Appl. Intell.,* vol. 54, pp. 5593–5616, 2024.

[14] J. Chen, J. Xu, S. Cai, X. Wang, H. Chen, and Z. Li, "Software Defect Prediction Approach Based on a Diversity Ensemble Combined With Neural Network," *IEEE Trans. Rel.,* vol. 73, no. 3, pp. 1487-1501, Sept. 2024, doi: 10.1109/TR.2024.3356515.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[16] X. Fan, S. Zhang, K. Wu, W. Zheng, and Y. Ge, "Cross-Project Software Defect Prediction Based on SMOTE and Deep Canonical Correlation Analysis," *Comput. Mater. Contin.,* vol. 78, pp. 1687–1711, 2024.

[17] M. Mustaqeem, T. Siddiqui, and S. Mustajab, "A hybrid-ensemble model for software defect prediction for balanced and imbalanced datasets using AI-based techniques with feature preservation: SMERKP-XGB," *J. Softw. Evol. Proc.*, vol. 37, no. 1, p. e2731, Jan. 2025, doi: 10.1002/smr. 2731.

[18] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data Quality: Some Comments on the NASA Software Defect Datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sept. 2013, doi: 10.1109/TSE.2013.11.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[19] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007, doi: 10.1109/TSE.2007.256941.

[20] R. Haque, A. Ali, S. McClean, I. Cleland, and J. Noppen, "Heterogeneous Cross-Project Defect Prediction Using Encoder Networks and Transfer Learning," *IEEE Access*, vol. 12, pp. 409-419, 2024, doi: 10.1109/ACCESS.2023.3343329.

[21] W. Wang, Y. Li, S. Song, J. Lu, B. Chen, and B. Wang, "*Research on Cross-project Defect Prediction Based on Instance Migration Method*," in Proc. 3rd Int. Conf. Comput. Sci. Manage. Technol. (ICCSMT), Shanghai, China, 2022, pp. 300–303, doi: 10.1109/ICCSMT58129.2022.00070.

[22] Z. Sun, J. Li, H. Sun, and L. He, "CFPS: Collaborative filtering based source projects selection for cross-project defect prediction," *Appl. Soft Comput.*, vol. 99, p. 106940, 2021.

[23] M. Patil, M. Bisi, and P. Manchala, "*Source Project Selection for Cross-Project Software Defect Prediction using Clustering Approach*," in Proc. IEEE 20th India Council Int. Conf. (INDICON), Hyderabad, India, 2023, pp. 904–909, doi: 10.1109/INDICON59947.2023.10440956.

[24] C. Jin, "Cross-project software defect prediction based on domain adaptation learning and optimization," *Expert Syst. Appl.*, vol. 171, p. 114637, Jun. 2021, doi: 10.1016/j.eswa.2021.114637.

[25] Y. Khatri and U. R. Saxena, "Dynamic learner selection for cross-project fault prediction," *Int. J. Syst. Assur. Eng. Manage.*, vol. 16, pp. 532–551, 2025, doi: 10.1007/s13198-024-02586-3.

[26] A. M. Ibrahim, H. Abdelsalam, and I. A. T. F. Taj-Eddin, "Software Defects Prediction At Method Level Using Ensemble Learning Techniques," *Int. J. Intell. Comput. Inf. Sci.*, vol. 23, no. 2, pp. 28-49, 2023, doi: 10.21608/ijicis. 2023.189934.1251.

[27] G. Lee, H. Ju, and S. U.-J. Lee, "*Can We Trust the Actionable Guidance from Explainable AI Techniques in Defect Prediction?*" in Proc. IEEE Int. Conf. Softw. Anal. Evol. Reeng. (SANER), Montreal, QC, Canada, 2025, pp. 476–487, doi: 10.1109/SANER64311.2025.00051.

[28] E. A. Felix and S. P. Lee, "Predicting the number of defects in a new software version," *PLoS ONE*, vol. 15, no. 3, Art. no. e0229131, Mar. 2020, doi: 10.1371/journal.pone.0229131.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[29] Rashmi and A. Kaur, "Smadasyn-boosted cross-project transfer learning for effective security fault prediction," Evol. Intell., vol. 18, p. 76, 2025, doi: 10.1007/s12065-025-01060-8.