IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

A Comparative Study of Traditional and Ensemble Classifiers for Software Defect Prediction on the NASA PROMISE Repository

Sanjay I[1], Dr.P.Durga Devi[2]

M.Tech 2nd Year, Department of Computer Science and Engineering, SRMIST, Vadapalani[1]

Assistant Professor, Department of Computer Science and Engineering, SRMIST, Vadapalani[2]

Abstract: Software defect prediction (SDP) directs limited testing effort toward the modules most likely to fail, and the NASA PROMISE datasets are its most common benchmark. Published accuracies on these datasets differ widely, from about 76% to more than 97%, and studies rarely report the majority-class baseline or state how resampling and data splitting were organised. This paper presents a controlled comparison of ten traditional and ensemble classifiers on CM1, KC1, KC2, PC1 and JM1, using the ADASYN-based study of Kabir et al. (2025) as the reference benchmark. We vary three factors: the model family (logistic regression, k-nearest neighbours, tuned SVM, random forest, extra trees, gradient boosting, histogram gradient boosting, AdaBoost, voting and stacking), the evaluation protocol (single split, oversampling before the split, oversampling inside training folds, repeated stratified cross-validation) and the data version (original versus deduplicated). Alongside accuracy, we report balanced accuracy, sensitivity and AUC. Three findings emerge. First, 11% to 43% of modules in the original datasets duplicate another module, and removing duplicate and inconsistent rows lowers the best accuracy by up to nine points. Second, oversampling the whole dataset before splitting raises accuracy by two to eight points over a raw split, but on a test set that is half synthetic; applied inside training folds, ADASYN lowers accuracy by roughly one to five points while raising sensitivity, and the area under the ROC curve is almost unchanged. Third, model choice matters little: top-ranked models differ from logistic regression by at most 1.1 accuracy points, and on CM1 and PC1 no model clearly exceeds the majority baseline. Under repeated stratified cross-validation our best accuracy exceeds the reference study on PC1 and JM1 for both data versions, while its CM1 result is not matched. We therefore recommend reporting accuracy with the baseline, balanced accuracy and sensitivity under a leakage-free protocol.

Keywords: Software defect prediction; NASA PROMISE repository; class imbalance; ADASYN; ensemble learning; data leakage; duplicate data; evaluation protocol

1. INTRODUCTION
2.

2.1 Background and Motivation

Software defects are costly to find late in the development cycle, and testing budgets are rarely large enough to examine every module with equal care. Software defect prediction (SDP) addresses this by learning, from static code metrics and historical defect records, which modules are most likely to contain faults so that review and testing effort can be prioritised [1]. Since the work of Menzies et al. [2], who showed that simple learners trained on static code attributes can produce useful predictors, SDP has become one of the most widely studied applications of machine learning in software engineering.

Large benchmarking studies have found that the choice of classifier often matters less than expected, and that many reported differences between models fall within the variation caused by data and experimental design [3]. This observation motivates the present work: when published results on the same datasets differ by ten points or more, it is worth asking how much of that difference reflects modelling and how much reflects the way the experiment was run.

2.2 The NASA PROMISE Repository

The PROMISE repository [4] hosts the NASA Metrics Data Program (MDP) datasets, which are among the most frequently reused benchmarks in SDP research. Each record describes a software module through 21 static metrics, including McCabe cyclomatic complexity measures [5], Halstead software science measures [6], line counts and branch counts, together with a binary label indicating whether a defect was reported. The five datasets used here are CM1 (498 modules), KC1 (2,109), KC2 (522), PC1 (1,109) and JM1 (13,204 in the version used in this study). Their defect rates range from about 7% to about 21%, so every dataset is imbalanced and a classifier that always predicts the non-

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151089

defective class already achieves high accuracy.
The repository has well-known limitations. Shepperd et al.
[7] documented duplicated and inconsistent records in the NASA datasets, and Gray et al. [8] showed that careless handling of such records can inflate reported performance. In our copy of CM1, 57 of the 498 modules share their entire feature vector with an earlier module, which means that a random split can place identical records in both the training and test sets.

2.3 Class Imbalance and Resampling

Class imbalance is a central difficulty in SDP. When defective modules form only a small fraction of the data, standard learners tend to favour the majority class, and overall accuracy can look strong while most defective modules are missed [9]. A common remedy is to resample the training data. The Synthetic Minority Over-sampling Technique (SMOTE) creates new minority samples by interpolating between a minority instance and its neighbours [10]. Adaptive Synthetic Sampling (ADASYN) extends this idea by generating more synthetic samples for minority instances that are harder to learn, that is, those surrounded by more majority-class neighbours [11].
Resampling is valid only when it is applied to the training data. If synthetic samples are generated before the dataset is divided, they are interpolated from records that later appear in the test set, so information from the test set reaches the training set and the measured performance is optimistic. In addition, a test set that has itself been oversampled no longer reflects the class distribution encountered in practice.

2.4 Evaluation Protocols and Metric Choice

Reported SDP results differ not only in the models they use but also in how they split data, whether and when they resample, and which metrics they report. Accuracy is easy to compare across papers and is often required by institutional or venue conventions, but on imbalanced data it must be read against the majority-class baseline. Balanced accuracy, sensitivity (recall on the defective class) and the area under the ROC curve give a fuller picture of how well defective modules are actually detected. A single random train-test split on a dataset with only a few dozen defective modules, such as CM1 with 49, is also a noisy estimate; repeated stratified cross-validation gives a more stable one.

2.5 Research Gap

Kabir et al. [12] recently compared traditional classifiers and bagging, boosting, stacking and voting ensembles on PC1, JM1, KC1 and CM1, applying ADASYN and an 80/20 split, and reported best accuracies between 76% and 94.5%. Other recent studies report accuracies as high as 97% to 98% on related datasets (Section 2.4). The evaluation protocols behind these figures differ, and the figures are seldom set against the majority-class baseline, so it remains unclear how much of the reported improvement from resampling, feature engineering and ensembling survives a controlled comparison. To our knowledge, no study places an ADASYN-based benchmark next to raw-data, pre-split and leakage-free results, on original and deduplicated versions of the same PROMISE datasets, with class-aware metrics and statistical tests.

2.6 Research Questions and Contributions

This paper addresses four research questions. **RQ1:** how much do duplicate and inconsistent records in the original datasets affect measured accuracy and the majority baseline? **RQ2:** how do accuracy, balanced accuracy, sensitivity and AUC change for the same classifiers when the evaluation protocol changes from a raw single split to oversampling before the split, to oversampling inside the training data only, and to repeated cross-validation? **RQ3:** do traditional and ensemble classifiers differ meaningfully once leakage is removed and variance is controlled? **RQ4:** how do our results compare with those of the reference study [12]?

The contributions are as follows. First, we provide a side-by-side comparison of ten classifiers on five PROMISE datasets under five explicitly defined evaluation protocols and two data versions. Second, we quantify how far headline accuracy lies above the majority-class baseline and what it conceals about defect sensitivity. Third, we measure the effect of ADASYN applied correctly, inside the training data, on accuracy, balanced accuracy, sensitivity and AUC. Fourth, we supply statistical tests of the classifier comparison and a leakage-free protocol that other researchers can adopt for fair comparison on these datasets.

2.7 Organisation of the Paper

Section 2 reviews related work on SDP benchmarks, data quality, imbalance handling and recent studies. Section 3describes the datasets, cleaning, classifiers, protocols, metrics and statistical analysis. Section 4 presents the results. Section 5 discusses their meaning and limitations, and Section 6 concludes and outlines future work.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

3. RELATED WORK

3.1 Benchmarking Classifiers for Defect Prediction

Menzies et al. [2] showed on NASA MDP data that Naive Bayes with log-transformed attributes could outperform decision-tree learners on probability of detection, which helped establish detection and false-alarm rates as common evaluation measures. A systematic review by Hall et al. [1] of fault prediction studies reported that simple techniques such as Naive Bayes and logistic regression perform comparatively well and that modelling context and reporting practice vary considerably. Lessmann et al. [3] compared 22 classifiers on ten NASA datasets using the area under the ROC curve and found no statistically significant differences among most of the best-performing classifiers. Ghotra et al. [13] later revisited the question with a larger set of techniques and reported that techniques can fall into statistically distinct groups, which shows that conclusions depend on the data version and on the evaluation set-up. et al. [14] showed that automated parameter optimization can change both the performance and the ranking of classifiers, so that default settings should not be treated as a neutral baseline. These studies motivate our use of repeated cross-validation, rank-based statistical tests [15] and a check on the effect of tuning.

3.2 Data Quality in the NASA Datasets

Shepperd et al. [7] documented implausible values, inconsistent records and duplicate modules in the NASA MDP datasets, and observed that different cleaned versions circulate, so that results obtained on one version cannot always be compared with those from another. Gray et al. [8] showed that large numbers of duplicated modules, including modules whose identical features carry different labels, can distort the measured performance of defect predictors when random splits place the same record in training and test sets. Despite this, many later studies state only that "NASA datasets" were used. We therefore treat the data version as an explicit experimental factor.

3.3 Oversampling in Defect Prediction

Surveys of imbalanced learning [9] describe resampling, cost-sensitive learning and ensemble approaches as the main remedies. SMOTE [10] and ADASYN [11] are the oversampling methods most often combined with defect predictors. Both interpolate between minority-class neighbours; ADASYN differs by allocating more synthetic samples to minority instances with more majority-class neighbours. Because these methods generate artificial records, they must be fitted on the training data only. This point is easy to overlook in a pipeline that balances a dataset once and then splits it, and it is one of the questions examined here.

3.4 Literature Review of Recent Studies (2024 to 2026)

Table 1 summarises six peer-reviewed journal studies from 2024 to 2026 that use NASA or PROMISE data, including the reference study [12]. The entries are based on the abstracts and publisher records of each paper.

Table 1. Literature review of six recent studies on NASA/PROMISE defect prediction.

No.	Paper / Author	Year	Core Idea	Key Mathematical Idea	Reason for Choosing	Limitation / Gap Findings
1	Ali et al. [16]	2024	Genetic-algorithm feature selection with RF, SVM and NB combined by soft voting on seven cleaned NASA datasets.	Fitness-driven search over metric subsets; averaging of predicted class probabilities (soft voting).	Peer-reviewed, uses PROMISE/NASA data; feature selection plus ensemble is a common pipeline.	Reports up to 95.1% accuracy (PC1); majority baseline not stated in the abstract; order of resampling and splitting not stated.
2	Pradhan et al. [17]	2024	PCA and LDA reduction; extreme learning machine (ELM) tuned by an enhanced JAYA optimiser; 10×5 cross-validation.	Population-based search that minimises validation error; ELM with random hidden layer and least-squares output weights.	Reports 94.87% to 98.08% accuracy on six datasets, the top of the published range.	Majority baseline not stated in the abstract; highest values come from heavily tuned pipelines, so the gain from each step is unclear.

DOI: 10.17148/IJARCCE.2026.151089

3	Aflaha et al. [18]	2024	14 SMOTE variants combine d with three boosting algorith ms on NASA MDP data.	Interpola tion between minority neighbou rs (SMOTE and variants); boosting re-weigh ts misclassi fied samples.	Directly studies oversam pling, the same question as our ADASY N comparis on.	Best accuracy 0.9161 (DSMOTE with gradient boosting); the abstract does not say whether resampling was confined to training data.

No.	Paper / Autho r	Ye ar	Core Idea	Key Mathem atical Idea	Reason for Choosin g	Limitation / Gap Findings
4	Mustaq eem et al. [19]	202 5	SMERK P-XGB: SMOTE-ENN resampli ng, RFE-CV feature selection and kernel PCA, followed by XGBoost .	SMOTE plus edited-ne arest-nei ghbour cleaning; nonlinear projectio n by kernel PCA; gradient-boosted trees.	Reports 97.53% on CM1 and 95.65% on KC1, among the highest recent figures.	Many stacked steps; the CM1 value is above the 90.2% majority baseline of the original file, but the protocol is not stated in the abstract.
5	Saima et al. [20]	202 6	Stack of RF, XGBoost and CatBoost with an MLP meta-lear ner; SMOTE or ADASY N with RFE or Boruta selection on ten datasets.	Stacked generalis ation (meta-lea rner trained on base-mo del predictio ns); recursive and all-releva nt feature selection.	Most recent study; combine s ADASY N with ensembli ng, close to our setting.	Up to 96.52% accuracy; the abstract names limited cross-project generalisatio n as a challenge; baselines not stated.
6	Kabir et al. [12] (refere nce)	202 5	ADASY N with tradition al classifier s and bagging, boosting, stacking and voting ensemble s on PC1, JM1, KC1 and CM1; 80/20 split.	ADASY N: synthetic samples weighted by the share of majority neighbou rs; ensemble aggregati on of base predictio ns.	Referenc e benchma rk for our replicatio n; reports PC1 87.08%, JM1 76.49%, KC1 81.45% and CM1 94.5%.	Reported PC1 and JM1 values lie below the majority baselines of the original data; the ordering of steps changed CM1 accuracy by roughly 9 to 13 points.

Values are as stated in each paper's abstract or record; protocols differ between studies and are not directly comparable.

Three patterns stand out. First, most pipelines stack several choices, namely feature selection or dimensionality reduction, resampling, hyperparameter search and an ensemble or meta-learner, each of which adds degrees of freedom to the experiment. Second, accuracy is the headline metric throughout, and the best reported values range from 76.49% to 98.08%, a spread of more than twenty points on overlapping sets of datasets. Third, from the abstracts and records we reviewed, few studies state whether resampling was applied before or after the split, whether duplicate modules were removed, or how the reported accuracy compares with the majority-class baseline. We do not claim that any individual result is invalid; rather, these details are needed before the figures can be compared.

3.5 Positioning of the Present Study

The present study differs from the work above in that it holds the data, the models and the metrics fixed and varies one factor at a time: the data version, the evaluation protocol and the model family. It includes an explicit replication of the pre-split resampling that would inflate results, reports the majority baseline next to every accuracy, and applies rank-based tests, so that the sources of variation in published accuracies can be separated.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

4. MATERIALS AND METHODS

4.1 Datasets

We used five NASA MDP datasets from a public mirror of the PROMISE repository: CM1, KC1, KC2, PC1 and JM1. Each module is described by 21 static code metrics (McCabe complexity, Halstead measures, line counts and branch counts) and a binary defect label (the label column is named "defects" in four files and "problems" in KC2). All attributes are numeric. The datasets differ in size from 522 to 13,204 modules and in defect rate from 6.9% to 20.5%.

4.2 Data Versions and Cleaning

Two versions of each dataset were analysed. The **original** version is the data as distributed. The **deduplicated** version was produced in two steps: (i) all rows whose feature vectors occur more than once with conflicting labels were removed, since the correct label cannot be determined; (ii) among the remaining rows with identical features and identical labels, one copy was kept. No other cleaning was applied, so the deduplicated version is a conservative, rule-based reduction rather than a replacement for curated versions such as the one described by Shepperd et al. [7].

4.3 Classifiers

Ten classifiers were compared, five of them traditional or single-model learners and five ensembles, using the scikit-learn library [21] (Table 2). Logistic regression, k-nearest neighbours and the support vector machine were applied to standardised features; tree-based models were applied to raw features. The tuned SVM selected C and gamma by three-fold grid search inside each training fold. The soft-voting and stacking ensembles combined random forest [22], extra trees and histogram gradient boosting, with logistic regression as the stacking meta-learner. The SVM was not run on JM1 and stacking was not run on the original JM1 because of run time.

Table 2. Classifier configurations.

Classifier	Group	Main settings
Logistic Regression (LR)	Traditio nal	Standardised features; L2 penalty (C = 1); up to 3,000 iterations
k-Nearest Neighbours (KNN)	Traditio nal	Standardised features; k = 5; Euclidean distance
SVM (tuned)	Traditio nal	RBF kernel; C in {1, 10, 100}; gamma in {scale, 0.01}; 3-fold grid search
Random Forest (RF)	Ensembl e	300 trees; random seed 42
Extra Trees (ET)	Ensembl e	400 randomised trees
Gradient Boosting (GB)	Ensembl e	200 stages; maximum depth 4
Hist. Gradient Boosting (HGB)	Ensembl e	Maximum depth 4; learning rate 0.05; 200 iterations
AdaBoost	Ensembl e	200 estimators
Voting	Ensembl e	Soft vote of RF (150 trees), ET (200 trees), HGB
Stacking	Ensembl e	Same base learners; logistic regression meta-learner; 3-fold internal CV

For the tuning experiment, random forest and gradient boosting were also tuned by randomised search (12 candidates, 3-fold CV) on the training portion only.

4.4 ADASYN and Evaluation Protocols

ADASYN [11] was implemented directly because the dedicated library was not available in the execution environment. Features are standardised and, for each minority instance x_i, the share $r_i = \Delta_i / k$ of majority-class instances among its $k = 5$ nearest neighbours (Δ_i of them) is computed and normalised to $\hat{r}_i$ (Eq. 1). The number of synthetic samples g_i generated for x_i is proportional to $\hat{r}_i$, and the total G is chosen to balance the classes (Eq. 2). Each synthetic sample interpolates between x_i and a randomly chosen minority neighbour $\tilde{x}_i$ (Eq. 3).

$$r_i = \Delta_i / k, \quad \hat{r}_i = r_i / \Sigma\square\ r\square \qquad (1)$$



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151089

$$G = N(maj) - N(min), \quad g_i = \hat{r}_i \cdot G \qquad (2)$$

$$s = x_i + \lambda (\tilde{x}_i - x_i), \quad \lambda \in [0, 1] \qquad (3)$$

Five evaluation protocols were applied. **P1:** a single stratified 80/20 split of the raw data (seed 42). **P2:** ADASYN applied to the whole dataset before the 80/20 split, which leaks information and leaves a test set of which half is synthetic; it is included to measure the effect of this practice, not as a valid protocol. **P3:** ADASYN applied to the 80% training portion only, with an untouched test set. **P4:** repeated stratified five-fold cross-validation on the raw data. **P5:** repeated stratified five-fold cross-validation with ADASYN fitted inside each training fold only. For the ten-model comparison (P4) we used 3 repeats for CM1, KC2 and PC1, 2 for KC1 and 1 for JM1, giving 15, 15, 15, 10 and 5 folds. For the ADASYN comparison (P4 against P5, four models) we used 10 repeats for CM1, KC2 and PC1, 5 for KC1 and 2 for JM1. The number of repeats was limited by run time on the larger datasets.

4.5 Evaluation Metrics

Accuracy is the primary metric, because it is the one reported by the reference study. It is always read against the **majority-class baseline**, the accuracy of a model that always predicts "not defective" (Eqs. 4 and 5). Sensitivity is the recall on the defective class, specificity the recall on the non-defective class, and balanced accuracy is their mean (Eqs. 6 to 8). The area under the ROC curve (AUC) is computed from predicted probabilities and is independent of the decision threshold.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \qquad (4)$$

$$\text{Baseline} = \max(p, 1 - p) \qquad (5)$$

$$\text{Sensitivity} = TP / (TP + FN) \qquad (6)$$

$$\text{Specificity} = TN / (TN + FP) \qquad (7)$$

$$\text{Balanced acc.} = (\text{Sens.} + \text{Spec.}) / 2 \qquad (8)$$

Here p is the proportion of defective modules in the evaluated data, and TP, TN, FP and FN are the entries of the confusion matrix with the defective class as positive.

4.6 Statistical Analysis

For each dataset and data version, the fold-level accuracies of the classifiers were compared with the Friedman test, with folds as blocks and classifiers as treatments, and average ranks were computed [15] (Eq. 9, where N is the number of folds, k the number of classifiers and R□ the average rank of classifier j). The best-ranked classifier by mean accuracy was then compared with logistic regression using a paired Wilcoxon signed-rank test. Because folds from repeated cross-validation share training data, the resulting p-values are indicative rather than exact.

$$\chi^2F = 12N/(k(k+1)) \cdot [\Sigma\square\, R\square^2 - k(k+1)^2/4] \qquad (9)$$

4.7 Implementation and Reproducibility

All experiments were run in Python with scikit-learn [21], with fixed random seeds. The splits, folds and models are deterministic given the seed, so the reported numbers can be reproduced from the public datasets. The datasets are public; the analysis scripts are available from the author on request.

5. RESULTS

5.1 Dataset Characteristics and the Effect of Cleaning (RQ1)

Table 3 shows the size of each dataset before and after cleaning. Between 11% (CM1) and 43% (KC1) of the modules in the original datasets duplicate an earlier module, with JM1 at 33%, KC2 at 29% and PC1 at 15%. Removing conflicting and duplicate rows reduces KC1 from 2,109 to 1,172 modules, JM1 from 13,204 to 8,732 and KC2 from 522 to 363, while PC1 (1,109 to 942) and CM1 (498 to 440) change less. Because duplicated modules are mostly non-defective, the defect rate rises after cleaning, for example from 15.5% to 25.2% on KC1 and from 15.9% to 21.9% on JM1, and the majority baseline falls accordingly (84.5% to 74.8% on KC1). Cleaning therefore lowers both the baseline and the best attainable accuracy, so accuracy figures from the two data versions should not be compared directly.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

Table 3. Dataset characteristics before and after cleaning.

Dataset	Mod ules	Defective (original)	Duplicates of earlier rows	Conflcting rows removed	Modules after cleaning	Defective (clean ed)	Majority baseline % (origclean)
CM1	498	49 (9.8%)	57 (11%)	2	440	47 (10.7 %)	90.2 / 89.3
KC1	2,109	326 (15.5 %)	917 (43%)	253	1,172	295 (25.2 %)	84.5 / 74.8
KC2	522	107 (20.5 %)	153 (29%)	118	363	99 (27.3 %)	79.5 / 72.7
PC1	1,109	77 (6.9%)	161 (15%)	13	942	64 (6.8%)	93.1 / 93.2
JM1	13,20 4	2,103 (15.9 %)	4,384 (33%)	1,116	8,732	1,916 (21.9 %)	84.1 / 78.1

Duplicates: rows whose 21-metric feature vector equals that of an earlier row. Conflicting rows: rows whose features occur with both labels. Cleaned: conflicting rows removed and one copy of each exact duplicate kept.

5.2 Effect of the Evaluation Protocol (RQ2)

Table 4 and Fig. 1 give the best accuracy among logistic regression, k-nearest neighbours, random forest and gradient boosting under each protocol on the original data. Under a raw single split (P1), the best accuracy lies between 0.2 points below and 2.5 points above the majority baseline: 90.0% against 90.2% on CM1, 87.0% against 84.5% on KC1, 81.9% against 79.5% on KC2,94.6% against 93.1% on PC1 and 85.1% against 84.1% on JM1. High accuracy on PC1 and CM1 therefore reflects mainly the class balance of the data.

Applying ADASYN to the entire dataset before splitting (P2) raises the best accuracy over P1 by 1.8 points on PC1, 3.3 on CM1, 6.2 on KC1, 6.9 on JM1 and 7.9 on KC2, reaching 93.3% on CM1 and 96.4% on PC1. These values are measured on a test set in which half of the modules are defective and many are synthetic, so they are neither comparable with the baselines in Table 4 nor free of leakage. When ADASYN is applied to the training portion only (P3), the best accuracy falls back to 80.0% (CM1), 86.5% (KC1), 79.0% (KC2), 93.2% (PC1) and 84.1% (JM1). Repeated cross-validation (P4) yields 84.1% to 93.8%, and fitting ADASYN inside the training folds (P5) lowers accuracy by roughly one to five points relative to P4 (from 89.4% to 84.1% on CM1, for example).

Table 4. Best accuracy (%) among LR, KNN, RF and GB under each protocol, original data.

Protocol	CM1	KC1	KC2	PC1	JM1
Majority baseline	90.2	84.5	79.5	93.1	84.1
Raw data, single 80/20 split	90.0	87.0	81.9	94.6	85.1
ADASYN before split, single 80/20 split (leaky)	93.3	93.2	89.8	96.4	92.0
ADASYN on training data only, single 80/20 split	80.0	86.5	79.0	93.2	84.1
Repeated 5-fold CV, raw data	89.4	86.1	84.1	93.8	85.2
Repeated 5-fold CV, ADASYN in training folds	84.1	84.3	80.1	92.0	84.0

Shaded: P2 (ADASYN before split), evaluated on a test set that is half synthetic and therefore not comparable with the other rows or with the majority baseline.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

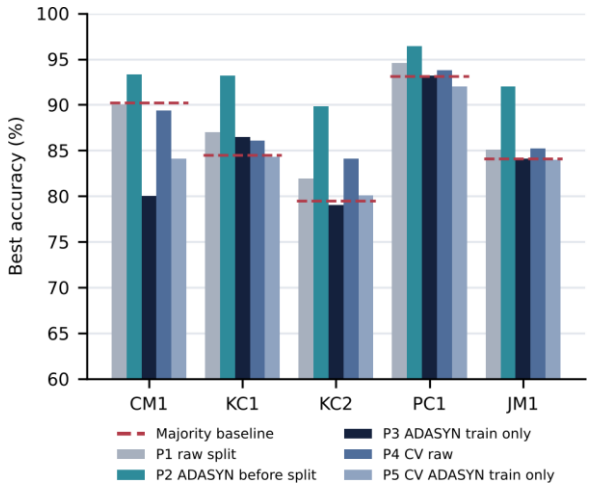


Fig. 1. Best accuracy (%) among LR, KNN, RF and GB under five evaluation protocols on the original data. Dashed lines mark the majority baseline. The pre-split ADASYN bars (P2) use a half-synthetic test set.

Single-split results are also noisy. On CM1 the test set of 100 modules contains about ten defective ones, so a change of one prediction moves accuracy by one point. Comparisons between models or protocols on a single CM1 split can therefore change by several points from one split to another, which is why we rely on repeated cross-validation.

5.3 Comparison of Traditional and Ensemble Classifiers (RQ3)

Table 5 reports the mean fold-level accuracy of all ten classifiers on the deduplicated data, and Fig. 2 shows accuracy relative to the majority baseline. Within each dataset the spread between the best and worst classifier is small: 2.6 points on CM1 (86.7% to 89.3%), 2.2 on KC1(75.3% to 77.5%), 2.6 on KC2 (76.7% to 79.3%), 1.2 on PC1 (92.4% to 93.6%) and 1.5 on JM1 (77.7% to 79.2%).

No family is consistently ahead: logistic regression ties for highest on KC1 (77.5%, with histogram gradient boosting and voting) and on KC2 (79.3%, with voting and stacking), stacking is highest on CM1, random forest and voting tie on PC1 (93.6%), and voting and extra trees tie on JM1 (79.2%). The results on the original data (Appendix, Table A1) show the same pattern, except that k-nearest neighbours lags the best model by 4.3 points on KC1 and 3.9 on JM1.

Table 5. Mean accuracy (%) ± standard deviation over folds, deduplicated data, repeated stratified 5-fold CV, no resampling.

Classifier	CM1	KC1	KC2	PC1	JM1
LR	88.3 ± 1.3	77.5 ± 1.3	79.3 ± 3.7	93.2 ± 0.9	78.8 ± 0.5
KNN	88.1 ± 1.2	75.3 ± 1.6	76.8 ± 3.9	92.7 ± 1.2	77.7 ± 1.0
RF	88.1 ± 1.3	76.8 ± 1.7	78.6 ± 3.5	93.6 ± 0.8	79.0 ± 0.5
GB	86.9 ± 1.7	75.9 ± 2.1	76.7 ± 4.4	93.4 ± 1.1	78.9 ± 0.7
ET	87.9 ± 1.3	76.9 ± 1.9	78.3 ± 3.4	93.3 ± 0.9	79.2 ± 0.5
HGB	86.7 ± 2.1	77.5 ± 1.8	78.5 ± 4.4	93.1 ± 1.3	78.8 ± 0.5
AdaBoost	87.0 ± 2.0	76.7 ± 1.2	78.3 ± 3.4	92.4 ± 1.2	78.7 ± 0.3
Voting	87.9 ± 1.1	77.5 ± 1.7	79.3 ± 3.9	93.6 ± 1.0	79.2 ± 0.5
Stacking	89.3 ± 0.6	76.0 ± 0.8	79.3 ± 3.0	93.2 ± 0.4	79.0 ± 0.4
SVM tuned	88.8 ± 1.1	77.2 ± 1.4	79.0 ± 2.8	93.5 ± 0.5	n/a

Folds: 15 (CM1, KC2, PC1), 10 (KC1), 5 (JM1). SVM was not run on JM1.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

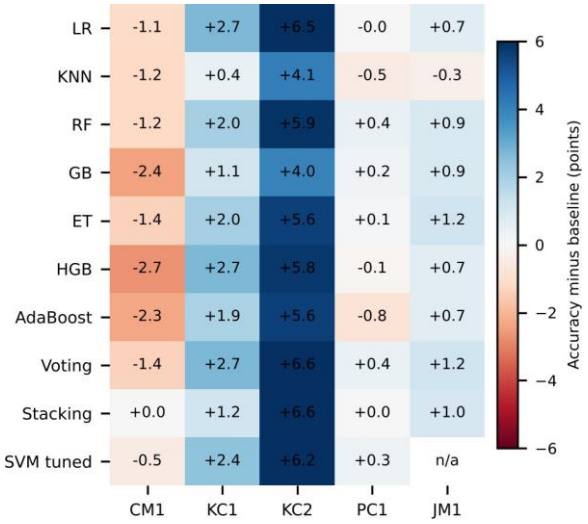


Fig. 2. Mean accuracy minus the majority-class baseline (percentage points), deduplicated data. Cells on PC1 and JM1 are within about one point of zero, CM1 is at or below zero, and KC2 is the only dataset on which every classifier clearly exceeds the baseline.

Measured against the baseline, the picture is clearer than the raw accuracies suggest. On KC2 every classifier exceeds the baseline by 4.0 to 6.6 points, and on KC1 by 0.4 to 2.7 points. On JM1 the margin is between −0.3 and +1.2 points, on PC1 between −0.8 and +0.4, and on CM1 between −2.7 and 0.0. Only on KC2 and, to a lesser degree, KC1 do the models learn enough from the metrics to improve clearly on predicting the majority class.

The Friedman test (Table 6) detects significant differences among the classifiers in eight of ten dataset and data-version combinations at the 0.05 level; KC2 is the exception in both versions ($p = 0.386$). The differences are small in size, however: in the eight cases where the best classifier by mean accuracy is not logistic regression itself, it exceeds logistic regression by only 0.1 to 1.1 points, and the paired Wilcoxon test is significant only on CM1 in both data versions and on KC1 and PC1 in the original data. The CM1 result needs particular care. Stacking ranks first on CM1 because it predicts the non-defective class almost exclusively (sensitivity 0.00 in both data versions), so its accuracy equals the baseline (89.3% cleaned, 90.2% original). A ranking by accuracy rewards such behaviour, which is a further reason not to rely on accuracy alone.

Table 6. Friedman test on fold-level accuracy and best/worst average ranks (rank 1 = best).

Data	Datas et	Fol ds	Mod els	χ^2	p	Best-rank ed (rank)	Worst-ra nked (rank)
Origin al	CM1	15	10	39.0	<0.00 1	Stacking (3.2)	GB (7.7)
Origin al	KC1	10	10	28.8	<0.00 1	Voting (3.3)	KNN (9.4)
Origin al	KC2	15	10	9.6	0.386	LR (4.1)	GB (6.5)
Origin al	PC1	15	10	26.9	0.001	Voting (3.6)	LR (7.0)
Origin al	JM1	5	8	23.5	0.001	Voting (1.8)	KNN (8.0)
Clean ed	CM1	15	10	38.4	<0.00 1	Stacking (2.4)	HGB (7.1)
Clean ed	KC1	10	10	18.7	0.028	Voting (3.9)	KNN (7.8)
Clean ed	KC2	15	10	9.6	0.386	Stacking (4.6)	GB (6.9)
Clean ed	PC1	15	10	23.5	0.005	RF (4.0)	AdaBoost (8.0)
Clean ed	JM1	5	9	16.8	0.033	ET (2.6)	KNN (8.2)

Shaded p-values are below 0.05. JM1 has only five folds, so its p-values are weak evidence.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

5.4 Effect of ADASYN Applied Inside the Training Folds (RQ2)

Table 7 and Fig. 3 compare repeated cross-validation without resampling (P4) and with ADASYN in the training folds (P5) on the deduplicated data for four representative models. The same pattern appears on all five datasets: ADASYN lowers accuracy and raises sensitivity. Accuracy falls by 5.4 to 12.5 points for logistic regression, 5.7 to 19.0 for k-nearest neighbours, 2.0 to 6.7 for random forest and 1.1 to 4.7 for gradient boosting. Sensitivity rises by 0.31 to 0.60 for logistic regression,0.12 to 0.22 for random forest and 0.04 to 0.13 for gradient boosting. Balanced accuracy increases most for logistic regression (by 0.06 to 0.20) and least for the tree ensembles (between −0.01 and +0.09).

The AUC changes by at most 0.05 in any cell. Resampling therefore moves the operating point of the classifier along a similar ROC curve rather than improving the ranking of modules by risk, which explains why accuracy falls while sensitivity rises. A plausible reason why tree ensembles keep more of their accuracy is that they fit the training data closely and shift their decision boundary less in response to synthetic minority samples than logistic regression and k-nearest neighbours do, although we did not test this explanation. Whether the trade is worthwhile depends on the relative cost of a missed defect and of an unnecessary inspection, which a single accuracy figure cannot express.

Table 7. Repeated 5-fold CV without resampling (P4) versus ADASYN in training folds (P5), deduplicated data.

Dataset	Mo del	Acc. % P4	Acc. % P5	Bal. acc. P4	Bal. acc. P5	Sens. P4	Sens. P5	AUC P4	AUC P5
CM 1	LR	88.1	76.6	0.53	0.73	0.08	0.68	0.78	0.76
	KN N	88.0	69.0	0.52	0.65	0.07	0.60	0.64	0.69
	RF	88.1	81.4	0.50	0.55	0.02	0.21	0.73	0.72
	GB	86.7	82.0	0.53	0.54	0.10	0.19	0.68	0.69
KC 1	LR	77.2	66.3	0.59	0.65	0.23	0.64	0.71	0.71
	KN N	75.1	63.2	0.59	0.61	0.27	0.58	0.65	0.65
	RF	77.1	74.4	0.62	0.65	0.32	0.46	0.73	0.73
	GB	75.8	74.5	0.62	0.63	0.33	0.40	0.70	0.70
KC 2	LR	79.2	73.8	0.68	0.74	0.44	0.75	0.83	0.82
	KN N	76.4	70.7	0.69	0.73	0.51	0.78	0.79	0.79
	RF	78.8	74.7	0.70	0.71	0.51	0.63	0.83	0.82
	GB	77.2	75.0	0.70	0.69	0.53	0.57	0.81	0.80
PC 1	LR	93.1	80.6	0.57	0.76	0.15	0.72	0.84	0.84
	KN N	92.8	79.5	0.55	0.70	0.12	0.59	0.71	0.75
	RF	93.4	90.9	0.58	0.67	0.18	0.40	0.83	0.85
	GB	93.0	91.3	0.60	0.65	0.21	0.34	0.81	0.83
JM 1	LR	78.8	68.5	0.55	0.65	0.12	0.58	0.70	0.70
	KN N	77.4	61.1	0.59	0.61	0.26	0.62	0.66	0.65

Dataset	Model	Acc. % P4	Acc. % P5	Bal. acc. P4	Bal. acc. P5	Sens. P4	Sens. P5	AUC P4	AUC P5
	RF	79.0	77.0	0.58	0.62	0.21	0.36	0.71	0.71
	GB	78.8	77.7	0.57	0.60	0.19	0.29	0.71	0.71

Repeats: 10 (CM1, KC2, PC1), 5 (KC1), 2 (JM1). Values are means over folds.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

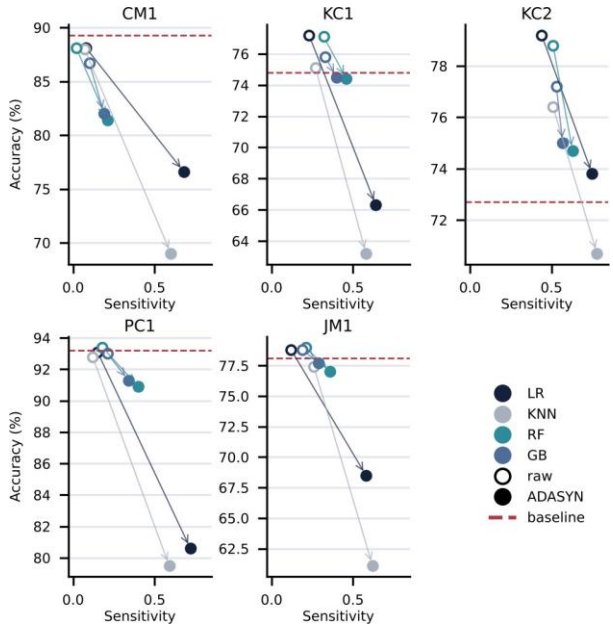


Fig. 3. Accuracy and sensitivity before (open markers) and after (filled markers) applying ADASYN inside the training folds, deduplicated data. Arrows show the shift; the dashed line is the majority baseline.

5.5 Effect of Hyperparameter Tuning

Table 8 compares default and tuned random forest and gradient boosting on a single stratified split of the deduplicated data. Tuned random forest is 1.1, 1.3, 1.4 and 0.5 points higher on CM1, KC1, KC2 and PC1 and 0.6 points lower on JM1. Tuned gradient boosting changes accuracy by −1.1, +0.8, +6.9, 0.0 and +0.2 points on the same datasets, and the only large change (KC2) comes from a test set of 73 modules. There is no consistent gain, and the single-split design cannot separate the effect of tuning from split noise; tuning inside cross-validation is left for future work.

Table 8. Accuracy (%) of default and tuned models on one stratified 80/20 split, deduplicated data.

Dataset	RF default	RF tuned	GB default	GB tuned
CM1	86.4	87.5	84.1	83.0
KC1	78.7	80.0	76.2	77.0
KC2	82.2	83.6	76.7	83.6
PC1	90.5	91.0	91.5	91.5
JM1	80.0	79.4	78.7	78.9

5.6 Comparison with the Reference Study (RQ4)

Table 9 sets our best accuracy under repeated cross-validation next to the values reported by Kabir et al. [12]. On the original data our best accuracy exceeds the reference value on PC1 (93.7% against 87.08%), JM1 (85.3% against 76.49%) and KC1 (86.5% against 81.45%) and falls short on CM1 (90.2% against 94.5%). On the deduplicated data it exceeds the reference on PC1 (93.6%) and JM1 (79.2%) and falls short on KC1 (77.5%) and CM1 (89.3%). The random-forest column, which does not depend on selecting the best of ten models, gives the same ordering. Because selecting the best of ten classifiers on the same folds is mildly optimistic, the random-forest values are the safer comparison.

Two cautions apply. First, the reference values on PC1 (87.08%) and JM1 (76.49%) lie below the majority baselines of the original data (93.1% and 84.1%), which suggests that its test data had a different class balance or that a different data version was used, so the comparison is not like for like. Second, neither our raw nor our pre-split protocol reproduces the pattern of the reference study: pre-split ADASYN (P2) comes close on CM1 (93.3% against 94.5%) but gives values far above the reference on PC1 (96.4%), JM1 (92.0%) and KC1 (93.2%), and raw evaluation gives values close to the baselines. The reference values on KC1 and JM1 are closer to our deduplicated results (77.5% and 79.2%) than to our original-data results, but the PC1 and CM1 values do not fit that explanation. We cannot determine the cause from the information available.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151089

Table 9. Best accuracy (%) under repeated CV compared with the reference study.

Data set	Reference study [12]	Original : best (model)	Original : RF	Original: baseline	Cleaned : best (model)	Cleaned : RF	Cleaned: baseline
CM 1	94.5	90.2 (Stacking)	89.8	90.2	89.3 (Stacking)	88.1	89.3
KC1	81.45	86.5 (Voting)	86.1	84.5	77.5 (LR)	76.8	74.8
KC2	—	84.0 (LR)	83.4	79.5	79.3 (Stacking)	78.6	72.7
PC1	87.08	93.7 (RF)	93.7	93.1	93.6 (RF)	93.6	93.2
JM1	76.49	85.3 (Voting)	85.3	84.1	79.2 (Voting)	79.0	78.1

Reference values are as reported and may use a different data version and protocol. Folds: 15 (CM1, KC2, PC1), 10 (KC1), 5 (JM1).

6. DISCUSSION

6.1 Interpretation of the Findings

The results point to the data and the protocol as larger sources of variation than the classifier. Duplicates make up between 11% and 43% of the original modules and change the baseline by up to ten points, so the same model can appear to gain or lose several points of accuracy depending on the version of the dataset used (RQ1). The evaluation protocol shifts accuracy by a similar amount: oversampling before the split adds two to eight points over a raw split, while the correct use of ADASYN lowers accuracy and raises sensitivity (RQ2). In contrast, once a leakage-free protocol is used, the ten classifiers differ by at most 2.6 points of accuracy within a dataset on the cleaned data (and by at most 4.3 points on the original data, where k-nearest neighbours lags), and the best-ranked classifier is within 1.1 points of logistic regression (RQ3).

The comparison against the majority baseline qualifies many of the apparent successes. On CM1 and PC1, nearly all models are within one or two points of predicting the majority class, and sensitivity is low in the absence of balancing. Reported accuracies of 90% or more on these datasets, ours included, mostly reflect their class balance. Defect prediction is a detection task, and a figure such as 93.7% accuracy on PC1 says little about how many defective modules are found.

Regarding the reference study (RQ4), our best results exceed its values on PC1 and JM1 for both data versions, but we regard this as a limited finding. The reference values on those datasets are below the majority baselines, so they were probably obtained on test data with a different class balance, and without access to its protocol we cannot say whether a like-for-like comparison would give the same ordering.

6.2 Recommendations for Reporting

Based on these results we suggest the following minimum reporting practice for SDP experiments on PROMISE-type data: (i) state the data version and report the number of duplicate and conflicting modules; (ii) report the majority-class baseline next to every accuracy;
(iii) apply any resampling to the training data only, and say so; (iv) report balanced accuracy, sensitivity and AUC with accuracy; (v) use repeated stratified cross-validation and report the spread across folds; and (vi) compare classifiers with a rank-based test rather than by differences in means alone.

6.3 Threats to Validity

Internal validity. ADASYN was implemented by us, and although it follows the published algorithm, small differences from the library implementation are possible. Hyperparameters were fixed rather than tuned within cross-validation, apart from the SVM; the tuning experiment used a single split. The number of cross-validation repeats was limited on JM1 and KC1 for run-time reasons, so the p-values for JM1 (five folds) are weak. Overlapping training sets make the folds dependent, so the reported p-values are indicative.

External validity. Only five NASA datasets of module-level static metrics were studied, and the findings may not transfer to other languages, cross-project settings, or process metrics. Gradient-boosting libraries such as XGBoost and CatBoost, used in several recent studies, were not included.

DOI: 10.17148/IJARCCE.2026.151089

Construct validity. Our deduplication rule is simple and differs from curated cleaned versions of the NASA data [7]. Comparisons with the reference study rely on its published figures; we could not verify its splitting and resampling protocol, its data version or its random seeds.

7. CONCLUSION AND FUTURE WORK

This study compared ten traditional and ensemble classifiers on five NASA PROMISE datasets under controlled variations of data version and evaluation protocol. Duplicate and conflicting records are a substantial feature of the original data and lower both the baseline and the best attainable accuracy when removed. Oversampling the whole dataset before splitting inflates accuracy on a test set that is half synthetic, whereas ADASYN applied inside the training folds lowers accuracy, raises sensitivity and leaves AUC almost unchanged. Differences between classifiers are small, and on CM1 and PC1 no classifier clearly exceeds the majority baseline. Under repeated cross-validation our best accuracy exceeds the reference study on PC1 and JM1 for both data versions, and not on CM1, but the comparison is limited by uncertainty about the reference protocol.

Future work includes tuning hyperparameters inside nested cross-validation, adding gradient-boosting libraries and cost-sensitive learning, evaluating threshold selection as an alternative to resampling, extending the analysis to cross-project prediction and to datasets with process metrics, and applying the reporting recommendations of Section 5.2 to the studies in Table 1 once their full protocols are available.

REFERENCES

[1]. T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," IEEE Trans. Softw. Eng., vol. 38, no. 6, pp.1276–1304, Nov./Dec. 2012.
[2]. 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.
[3]. S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," IEEE Trans. Softw. Eng., vol. 34, no. 4, pp. 485–496, Jul./Aug. 2008.
[4]. J. Sayyad Shirabad and T. J. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada, 2005.
[5]. T. J. McCabe, "A complexity measure," IEEE Trans.Softw. Eng., vol. SE-2, no. 4, pp. 308–320, Dec. 1976.
[6]. M. H. Halstead, Elements of Software Science. New York, NY, USA: Elsevier North-Holland, 1977.
[7]. 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, Sep. 2013.
[8]. D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "The misuse of the NASA metrics data program data sets for automated software defect prediction," in Proc. 15th Annu. Conf. Eval.Assessment Softw. Eng. (EASE), 2011, pp. 96–103.
[9]. H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Trans. Knowl. Data Eng., vol. 21, no. 9,pp. 1263–1284, Sep. 2009.
[10]. 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.
[11]. H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN:Adaptive synthetic sampling approach for imbalanced learning," in Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN), 2008, pp. 1322–1328.
[12]. S. M. H. Kabir, M. T. Rahman, and A. H. Mridul, "Software defect prediction using traditional machine learning and ensemble learning algorithms," Smart Wearable Technology, 2025, doi: 10.47852/bonviewSWT52025645.
[13]. B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in Proc.37th IEEE/ACM Int. Conf. Softw. Eng. (ICSE), 2015,pp. 789–800.
[14]. C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," in Proc. 38th Int. Conf. Softw. Eng. (ICSE), 2016, pp. 321–332.
[15]. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," J. Mach. Learn. Res., vol. 7, pp. 1–30, 2006.
[16]. M. Ali, T. Mazhar, A. Al-Rasheed, T. Shahzad, Y. Y. Ghadi, 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.
[17]. D. Pradhan, D. Muduli, A. T. Zamani, S. I. Yaqoob, S.M. Alanazi, R. R. Kumar, N. Parveen, and M. Shameem, "Refined software defect prediction using enhanced JAYA optimization and extreme learning machine," IEEE

DOI: 10.17148/IJARCCE.2026.151089

Access, vol. 12, pp. 141559–141579, 2024, doi: 10.1109/ACCESS.2024.3467183.

[18]. R. U. Aflaha, R. Herteno, M. R. Faisal, F. Abadi, and S. W. Saputro, "Effect of SMOTE variants on software defect prediction classification based on boosting algorithm," J. Ilm. Tek. Elektro Komput. Inform. (JITEKI), vol. 10, no. 2, pp. 201–216, 2024,doi: 10.26555/jiteki.v10i2.28521.

[19]. 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. Process, vol. 37, no. 1, 2025, doi: 10.1002/smr.2731.

[20]. S. N. Saima, A. Salam, M. M. Kawsur, and K. M. Imtiaz-Ud-Din, "An advanced stacked ensemble framework for enhanced software bug predictions," J. Inf. Organ. Sci., vol. 50, no. 1, pp. 247–262, 2026, doi: 10.31341/jios.50.1.13.

[21]. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp.2825–2830, 2011.

[22]. L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

APPENDIX

Table A1. Mean accuracy (%) ± standard deviation over folds, original data, repeated stratified 5-fold CV, no resampling.

Classifier	CM1	KC1	KC2	PC1	JM1
LR	89.1 ± 1.3	85.6 ± 0.4	84.0 ± 2.7	92.8 ± 0.9	84.5 ± 0.1
KNN	89.1 ± 1.4	82.2 ± 2.1	82.6 ± 3.1	92.8 ± 1.2	81.4 ± 0.8
RF	89.8 ± 1.0	86.1 ± 1.2	83.4 ± 3.0	93.7 ± 0.8	85.3 ± 0.7
GB	88.2 ± 1.7	85.8 ± 0.8	82.6 ± 2.6	93.1 ± 1.1	84.8 ± 0.4
ET	89.3 ± 1.1	86.2 ± 1.1	83.0 ± 3.1	93.6 ± 0.9	85.1 ± 0.8
HGB	88.2 ± 2.2	85.9 ± 0.9	83.0 ± 3.0	93.1 ± 0.8	84.6 ± 0.3
AdaBoost	88.0 ± 2.1	85.8 ± 0.5	82.6 ± 3.2	93.1 ± 1.0	84.4 ± 0.1
Voting	89.1 ± 1.4	86.5 ± 0.9	83.5 ± 3.1	93.7 ± 0.8	85.3 ± 0.6
Stacking	90.2 ± 0.4	85.4 ± 0.7	83.3 ± 2.0	93.4 ± 0.5	n/a
SVM tuned	90.1 ± 0.5	85.7 ± 0.7	83.7 ± 3.1	93.0 ± 0.6	n/a

SVM and stacking were not run on the original JM1 because of run time.