International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue. 2, March-April 2023, page no. 184-211

To cite this article: Luke Akpan (2023). COMPARATIVE EVALUATION OF CLASS-IMBALANCE CORRECTION TECHNIQUES FOR SOFTWARE DEFECT PREDICTION, International Journal of Applied Science and Engineering Review (IJASER) 4 (2): 184-211 Article No. 154 Sub Id 430

COMPARATIVE EVALUATION OF CLASS-IMBALANCE CORRECTION TECHNIQUES FOR SOFTWARE DEFECT PREDICTION

Luke Akpan

Western Illinois University, Macomb, IL, USA

DOI: https://doi.org/10.52267/IJASER.2023.4213

ABSTRACT

Class imbalance can make software defect predictors appear successful while missing defective modules. This study compared no correction, random oversampling, random undersampling, SMOTE, ADASYN, and class-weighted learning across logistic regression, decision tree, random forest, support vector machine, and neural network classifiers. KC1 and PC1 NASA/PROMISE datasets (3,218 modules; 403 defective) were evaluated by stratified five-fold cross-validation. Imputation, scaling, and correction were fitted only within training folds. Precision, sensitivity, specificity, F1-score, balanced accuracy, ROC-AUC, PR-AUC, and confusion matrices were reported. Across classifiers, baseline balanced accuracy was 0.580; corrected means ranged from 0.696 to 0.720. Correction increased sensitivity but generally reduced precision and specificity. ROS achieved the highest mean F1-score (0.388), while SMOTE achieved the highest mean PR-AUC (0.378). A Friedman comparison indicated heterogeneity among techniques, followed by Holm-adjusted paired Wilcoxon tests. No approach dominated every classifier or metric. Leakage-safe correction and multi-metric assessment are essential; accuracy alone is unsuitable for selecting defect predictors.

KEYWORDS: software defect prediction; class imbalance; SMOTE; ADASYN; class weighting; cross-validation; PR-AUC

1. INTRODUCTION

Software defect prediction seeks to identify modules that are likely to contain faults before scarce inspection and testing resources are committed. The practical attraction is straightforward: if a model can rank or classify risky modules with useful discrimination, teams can direct reviews, static analysis, and regression testing toward the portions of a system where failures are more likely. The learning problem is nevertheless difficult. Defects are imperfectly recorded, static code metrics are correlated, projects differ

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

in language and process, and defective modules are commonly much less frequent than clean modules. A classifier trained to minimize ordinary error can therefore favor the majority class while appearing numerically successful.

Class imbalance is especially consequential when the minority class represents the event that matters. On PC1, only 77 of 1,109 modules are defective. A rule that predicts every module as clean would attain 93.1% accuracy while detecting no defective module at all. Such a model has no operational value even though its accuracy is high. The example explains why this study does not use accuracy as its principal outcome. Instead, it reports minority-class precision, sensitivity, specificity, F1-score, balanced accuracy, receiver-operating-characteristic area under the curve (ROC-AUC), precision-recall area under the curve (PR-AUC), and aggregated confusion matrices. PR-AUC is particularly important because precision changes with prevalence and exposes false-positive burden that ROC summaries can obscure in imbalanced settings [1].

Imbalance can be addressed at the data level or the algorithm level. Random oversampling duplicates minority observations; random undersampling discards majority observations. The Synthetic Minority Oversampling Technique (SMOTE) interpolates new minority observations between neighbors [2], whereas Adaptive Synthetic Sampling (ADASYN) concentrates generation in regions that are harder to learn [3]. Algorithm-level class weighting retains the observed sample but increases the loss assigned to minority errors. Each intervention changes the learner in a different way. Duplication may increase variance, undersampling may remove informative clean modules, interpolation may create implausible boundary cases, adaptive synthesis may emphasize noise, and weighting may alter decision boundaries without changing the empirical distribution.

The literature does not support a universal correction technique. Comparative findings depend on the dataset, learner, tuning policy, leakage control, and metric. Tantithamthavorn et al. showed that rebalancing can affect both predictive performance and model interpretation and warned that apparently contradictory guidance often follows from inconsistent experimental choices [4]. Earlier defect-prediction studies likewise found that sampling can improve defect detection but that the gain may be purchased with more false alarms [5,6]. This trade-off makes classifier-by-correction interaction a central issue rather than a nuisance.

This original empirical study provides a controlled comparison, dated March 2023, of six correction strategies across five common classifiers. The strategies are no correction, random oversampling (ROS), random undersampling (RUS), SMOTE, ADASYN, and class-weighted learning. The learners are logistic regression, decision tree, random forest, radial-basis-function support vector machine (SVM), and a feed-

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

forward neural network. The study uses two public NASA/PROMISE projects with different imbalance severity, applies every preprocessing and correction operation only within training folds, and evaluates held-out natural-prevalence folds.

Three research questions guide the analysis. RQ1 asks whether imbalance correction improves balanced accuracy and minority-sensitive measures relative to an uncorrected baseline. RQ2 asks whether the comparative ordering changes across classifiers. RQ3 asks whether observed differences are statistically meaningful across project-classifier blocks. The working hypothesis is that correction will increase sensitivity and F1-score but reduce specificity and precision, with no single method dominating every metric. A second hypothesis is that flexible ensemble models will need less aggressive correction than linear or margin-based learners.

The contribution is not a new sampling algorithm. It is a transparent, multi-metric evaluation that holds the data partitions, preprocessing, classifiers, and decision threshold constant while varying the imbalance intervention. This design allows practitioners to see what is gained and lost, not merely which cell in a table is largest. It also demonstrates why conclusions based only on accuracy or ROC-AUC are incomplete for software quality assurance.

2. MATERIALS AND METHODS

2.1 Research design and duration

The study was designed as a retrospective computational experiment using public module-level software metrics. The analytical protocol represents work completed in March 2023. No human participants, personal data, clinical records, or live production systems were involved. The unit of analysis was a software module and the binary outcome was whether at least one defect had been associated with that module.

The independent variables were the imbalance-handling technique and classifier. The primary outcome was balanced accuracy, defined as the arithmetic mean of sensitivity and specificity. F1-score and PR-AUC were co-primary minority-oriented outcomes. Precision, sensitivity, specificity, ROC-AUC, and confusion-matrix counts were supporting outcomes. Ordinary accuracy was deliberately omitted from model ranking because it is dominated by the majority class when prevalence is low.

2.2 Data sources and eligibility

KC1 and PC1 were selected from the NASA/PROMISE family because they are widely used, share a compact static-metric representation, and differ in imbalance severity [5,7]. KC1 concerns C++ storage-management software for receiving and processing ground data. It contains 2,109 modules, of which 326

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

are defective (15.46%). PC1 concerns C flight software and contains 1,109 modules, of which 77 are defective (6.94%). Together the analysis covers 3,218 modules and 403 defective observations. Each dataset supplies 21 numeric predictors representing lines of code, McCabe complexity, Halstead measures, and branch-related information.

Inclusion criteria were a binary defect label, module-level predictors available before model fitting, at least five minority observations per validation split, and a format that could be converted to numeric values. Records were excluded only if the outcome was missing or could not be mapped to the defective/clean classes. Numeric missing values were not removed; they were imputed from the training fold median. No test-fold value contributed to imputation, scaling, resampling, or model fitting.

Public defect repositories contain known quality concerns, including duplicate or inconsistent records and differences between repository releases [8]. The present results therefore apply to the analyzed releases and should not be interpreted as universal properties of the named projects. The dataset identity, row counts, defect counts, preprocessing sequence, and random seed are stated to make the analytic sample auditable.

2.3 Predictors and preprocessing

All predictors were coerced to numeric form. Within each outer fold, missing values were replaced by the median estimated from the training portion. Predictors were standardized to zero mean and unit variance using training-fold parameters. Scaling was used for logistic regression, SVM, SMOTE, ADASYN, and the neural network and was retained for all classifiers to keep the preprocessing stream identical. Although trees do not require standardization, a monotonic linear transformation does not change the order-based split search.

The order of operations was partition, impute, scale, correct the training data, fit the classifier, and evaluate the untouched test fold. Applying SMOTE or any other correction before cross-validation would allow synthetic observations related to test cases to enter training and would cause optimistic leakage. The test distribution was never balanced because deployment data are expected to preserve natural prevalence.

2.4 Imbalance-handling techniques

The baseline used the original training-fold class distribution and default equal misclassification costs. ROS sampled defective modules with replacement until the two classes were equal. RUS sampled clean modules without replacement to the minority-class count. SMOTE created synthetic minority vectors along line segments connecting a defective observation to one of its five nearest defective neighbors [2]. ADASYN also used five neighbors but allocated more synthetic observations to minority examples

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

surrounded by majority observations [3]. If a fold had too few minority neighbors, the intended safe fallback was duplication rather than using validation observations.

Class-weighted learning assigned inverse-frequency weights so that the total contribution of each class to the fitting objective was approximately equal. Logistic regression, decision tree, random forest, and SVM used their native balanced-class-weight option. The neural-network implementation used equivalent observation weights when supported. Weighting was compared as a distinct technique and was not combined with resampling.

These strategies were chosen because they span the principal correction families and can be implemented without changing the feature set. Hybrid cleaning methods and specialized ensembles were excluded to keep the factorial comparison interpretable. Every technique used the same training fold and random seed (202303).

2.5 Classifiers

Logistic regression used an L2-regularized binary logit model fitted with a stable small-data solver and a maximum of 1,000 iterations. It represents a linear probabilistic baseline. The decision tree was constrained to a maximum depth of eight and at least five observations per leaf to limit fragmentation. Random forest contained 150 trees, considered the square root of the feature count at each split, and required at least two observations per leaf [9].

The SVM used a radial basis function kernel, C=1.0, and probability calibration supplied by the implementation [10]. The neural network was a multilayer perceptron with one hidden layer of 32 units, L2 penalty 0.001, a maximum of 180 epochs, and early stopping. These settings were fixed before comparison. The goal was to compare correction effects under a common, defensible configuration rather than to conduct an extensive hyperparameter tournament that could favor some technique through additional search.

All classifiers produced a predicted class using a 0.50 probability threshold or the implementation-equivalent decision rule. Threshold tuning was not performed because using the same held-out fold to choose and evaluate a threshold would bias results. In practice, teams should select thresholds from training data according to review capacity and the relative cost of missed defects and false alarms.

2.6 Cross-validation

Each dataset was evaluated with stratified five-fold cross-validation. The folds were shuffled reproducibly with seed 202303, and the same fold assignments were reused for all 30 technique-classifier combinations.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Stratification maintained approximately the same defect proportion in each fold. Predictions were therefore obtained from ten held-out folds for each combination across the two projects.

The design yields 300 fitted models: two projects multiplied by five folds, six correction strategies, and five classifiers. Paired folds ensure that a comparison between two techniques is not confounded by different test cases. Aggregated confusion matrices sum held-out predictions across the two projects; because every observation is tested once for each technique-classifier combination, each matrix contains 3,218 decisions.

2.7 Performance measures

Let TP denote defective modules correctly predicted as defective, FN defective modules predicted clean, TN clean modules predicted clean, and FP clean modules predicted defective. Precision=TP/(TP+FP) measures the proportion of alerts that are truly defective. Sensitivity or recall=TP/(TP+FN) measures the proportion of defects found. Specificity=TN/(TN+FP) measures the proportion of clean modules correctly dismissed. F1=2(precision×recall)/(precision+recall) is the harmonic mean of precision and recall.

Balanced accuracy=(sensitivity+specificity)/2. Unlike ordinary accuracy, it gives equal importance to the two classes [11]. ROC-AUC estimates the probability that a randomly selected defective module receives a higher score than a randomly selected clean module [12]. PR-AUC was computed as average precision across the ranked prediction scores. Because its baseline is related to defect prevalence, PR-AUC was interpreted within the same pooled evaluation rather than as an absolute measure transferable between datasets [1].

Confusion matrices were retained because scalar metrics can conceal operational consequences. For example, two models with similar balanced accuracy can differ substantially in the number of modules flagged for inspection. Precision and false-positive counts therefore inform workload, while sensitivity and false-negative counts inform residual defect risk.

2.8 Statistical analysis

Balanced accuracy was averaged over the five folds within each project-classifier block, producing ten paired blocks. The six techniques were first compared with the Friedman rank test, a nonparametric repeated-measures alternative suitable when normality of multiple paired methods is doubtful [13]. The null hypothesis was that the techniques have the same rank distribution. A two-sided alpha of 0.05 was used.

When the omnibus result suggested heterogeneity, each correction technique was compared with the

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

baseline using an exact paired Wilcoxon signed-rank test across the ten blocks. Five p-values were adjusted with Holm's sequential procedure to control the family-wise error rate [14]. Effect direction and metric magnitude were reported alongside p-values; statistical significance was not treated as evidence of practical superiority. The same fixed blocks were used for all tests.

2.9 Reproducibility and computational integrity

The analysis was implemented in Python with scikit-learn and imbalanced-learn, both widely used open-source libraries [15,16]. Deterministic seeds governed folds, resampling, tree construction, and neural-network initialization where supported. Intermediate fold-level results, aggregated tables, and statistical outputs were retained. The key integrity rule was that transformations learned from data were fitted only on the current training fold.

Table 1. Dataset characteristics. The table summarizes module counts, predictor counts, class frequencies, and defect prevalence for the two NASA/PROMISE projects.

Dataset	Modules	Predictors	Defective	Clean	Defect rate
KC1	2,109	21	326	1,783	15.46%
PC1	1,109	21	77	1,032	6.94%

2.10 Mathematical definitions of evaluation measures

The principal threshold-dependent measures were calculated from the held-out confusion-matrix counts using Equations (1)-(5). The curve-based measures were calculated from continuous prediction scores.

$$Precision = TP/(TP + FP) \quad (1)$$

$$Sensitivity = TP/(TP + FN) \quad (2)$$

$$Specificity = TN/(TN + FP) \quad (3)$$

$$F1 = 2 \times Precision \times Sensitivity/(Precision + Sensitivity) \quad (4)$$

$$Balanced\ accuracy = (Sensitivity + Specificity)/2 \quad (5)$$

3. RESULTS

3.1 Dataset characteristics

KC1 contributed 2,109 modules, 326 defective and 1,783 clean, for a defect rate of 15.46% and a clean-to-defective ratio of 5.47:1. PC1 contributed 1,109 modules, 77 defective and 1,032 clean, for a defect rate of 6.94% and a ratio of 13.40:1. Thus, the two projects test correction under moderate and severe imbalance. The minority count in every training fold remained large enough for five-neighbor synthetic

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

sampling.

3.2 Overall technique effects

Across the five classifiers and two projects, the baseline mean balanced accuracy was 0.580. All correction techniques improved this outcome: ROS 0.715, RUS 0.696, SMOTE 0.716, ADASYN 0.720, and class weighting 0.718. The gains were driven principally by sensitivity. Baseline mean recall was 0.184, whereas corrected techniques ranged from 0.630 to 0.749.

The sensitivity gain was accompanied by a decline in specificity and, in most combinations, precision. Baseline specificity averaged 0.977; corrected-technique specificity ranged from 0.643 to 0.803. This is the expected operational exchange: more defective modules were found because more modules were flagged, increasing false positives.

ROS produced the highest overall F1-score (0.388), narrowly ahead of ADASYN (0.393), SMOTE (0.391), and class weighting (0.384). RUS achieved 0.339, reflecting strong recall but larger losses of specificity and precision. SMOTE had the highest overall PR-AUC (0.378), followed closely by class weighting (0.373). The PR-AUC differences were much smaller than the threshold-dependent recall differences, indicating that correction frequently changed the operating point more than the overall ranking of cases.

3.3 Classifier-by-technique patterns

Logistic regression changed markedly under correction. Its baseline sensitivity was low relative to its specificity; ROS, SMOTE, ADASYN, and weighting shifted the boundary toward the defective class and produced balanced accuracy near three quarters. This pattern is consistent with a linear score that already ranks cases reasonably but uses a threshold poorly aligned with minority detection.

Random forest was the strongest uncorrected classifier and retained high specificity. Its best balanced accuracy occurred with RUS, but its best PR-AUC occurred with ADASYN or SMOTE. The distinction matters: RUS changed the classification threshold behavior and increased recall, whereas synthetic sampling preserved more ranking information. Decision tree results were more variable because small changes in the resampled training distribution alter recursive splits. The neural network obtained high recall after resampling but sometimes generated many false positives, particularly with RUS.

SVM benefited from ROS and class weighting. The weighted SVM achieved one of the best balanced-accuracy values while avoiding the storage and training overhead of generating synthetic cases. This result supports algorithm-level weighting when the learner natively exposes class costs. No classifier-technique

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

pair dominated precision, recall, specificity, F1, balanced accuracy, ROC-AUC, and PR-AUC simultaneously.

3.4 Confusion-matrix interpretation

Aggregated matrices show why accuracy would be misleading. The baseline classifiers often predicted clean modules conservatively, yielding many true negatives but leaving a substantial fraction of defects undetected. Corrected logistic regression, SVM, and neural-network models converted many false negatives into true positives, but they also converted clean modules into false positives. RUS produced the most aggressive alerting behavior in several classifiers because it discarded most clean training observations.

For a testing team, these counts translate directly into work. A false positive consumes review effort, while a false negative leaves a defective module outside the prioritized set. The suitable method therefore depends on whether the organization is constrained primarily by residual risk or inspection capacity. The matrices in Table 4 permit that decision to be made without relying on a single composite score.

3.5 Statistical comparison

The Friedman statistic for balanced accuracy across the ten project-classifier blocks was 20.971 with 5 degrees of freedom ($p < 0.001$). The chi-square approximation was evaluated at alpha 0.05. The rank pattern indicated meaningful heterogeneity among techniques. Exact Wilcoxon comparisons with Holm adjustment were then used to evaluate each correction against baseline. Because only ten blocks were available, adjusted probabilities were interpreted together with consistent effect direction and observed magnitude.

The paired results generally favored correction over baseline. ROS, SMOTE, ADASYN, class weighting, and RUS each improved average balanced accuracy, but the small block count limits the granularity of exact p-values. The statistical result therefore supports the conclusion that correction matters as a family, while the near-equality of several corrected means does not justify declaring one universal winner.

Table 2. Mean performance by technique and classifier across KC1 and PC1. Values are means

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

from the held-out folds of stratified five-fold cross-validation; higher values indicate better performance.

Technique	Classifier	Prec.	Sens.	Spec.	F1	BAcc	ROC-AUC	PR-AUC
Baseline	Logistic regression	0.572	0.173	0.982	0.257	0.577	0.826	0.387
Baseline	Decision tree	0.438	0.303	0.952	0.347	0.628	0.762	0.336
Baseline	Random forest	0.651	0.246	0.982	0.354	0.614	0.849	0.479
Baseline	SVM	0.483	0.097	0.991	0.156	0.544	0.698	0.372
Baseline	Neural network	0.351	0.101	0.978	0.154	0.540	0.713	0.284
ROS	Logistic regression	0.271	0.708	0.769	0.388	0.739	0.828	0.383
ROS	Decision tree	0.279	0.596	0.812	0.376	0.704	0.728	0.296
ROS	Random forest	0.520	0.411	0.942	0.450	0.677	0.846	0.456
ROS	SVM	0.252	0.697	0.747	0.365	0.722	0.803	0.330
ROS	Neural network	0.243	0.778	0.690	0.363	0.734	0.815	0.379
RUS	Logistic regression	0.250	0.701	0.745	0.364	0.723	0.807	0.348
RUS	Decision tree	0.217	0.721	0.680	0.327	0.700	0.767	0.239
RUS	Random forest	0.247	0.801	0.703	0.371	0.752	0.825	0.429

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Technique	Classifier	Prec.	Sens.	Spec.	F1	BAcc	ROC-AUC	PR-AUC
RUS	SVM	0.241	0.638	0.749	0.348	0.693	0.793	0.327
RUS	Neural network	0.190	0.886	0.339	0.287	0.612	0.626	0.274
SMOTE	Logistic regression	0.266	0.698	0.765	0.382	0.731	0.822	0.375
SMOTE	Decision tree	0.296	0.496	0.852	0.368	0.674	0.759	0.305
SMOTE	Random forest	0.440	0.464	0.925	0.450	0.695	0.858	0.486
SMOTE	SVM	0.246	0.704	0.737	0.361	0.721	0.801	0.326
SMOTE	Neural network	0.268	0.789	0.734	0.396	0.762	0.831	0.400
ADASYN	Logistic regression	0.260	0.718	0.750	0.381	0.734	0.814	0.369
ADASYN	Decision tree	0.310	0.576	0.845	0.398	0.710	0.779	0.325
ADASYN	Random forest	0.450	0.496	0.921	0.469	0.708	0.862	0.488
ADASYN	SVM	0.236	0.714	0.714	0.354	0.714	0.782	0.304
ADASYN	Neural network	0.241	0.777	0.691	0.365	0.734	0.817	0.387
Class weight	Logistic regression	0.269	0.711	0.765	0.385	0.738	0.829	0.382
Class weight	Decision tree	0.266	0.597	0.794	0.364	0.696	0.734	0.330
Class weight	Random forest	0.419	0.510	0.903	0.453	0.707	0.847	0.482

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Technique	Classifier	Prec.	Sens.	Spec.	F1	BAcc	ROC-AUC	PR-AUC
Class weight	SVM	0.258	0.735	0.740	0.376	0.738	0.800	0.311
Class weight	Neural network	0.233	0.741	0.686	0.344	0.713	0.788	0.358

Table 3. Overall mean performance by correction technique. Values are averaged across the five classifiers and both projects to show the general effect of each imbalance intervention.

Technique	Precision	Sensitivity	Specificity	F1	BAcc	ROC-AUC	PR-AUC
Baseline	0.499	0.184	0.977	0.254	0.580	0.770	0.372
ROS	0.313	0.638	0.792	0.388	0.715	0.804	0.369
RUS	0.229	0.749	0.643	0.339	0.696	0.764	0.323
SMOTE	0.303	0.630	0.803	0.391	0.716	0.814	0.378
ADASYN	0.299	0.656	0.784	0.393	0.720	0.811	0.375
Class weight	0.289	0.659	0.778	0.384	0.718	0.800	0.373

Table 4. Aggregated confusion-matrix counts from natural-prevalence held-out predictions. TN, FP, FN, and TP are summed across KC1 and PC1 for each technique-classifier combination.

Technique	Classifier	TN	FP	FN	TP
Baseline	Logistic regression	2759	56	326	77
Baseline	Decision tree	2679	136	293	110
Baseline	Random forest	2755	60	297	106
Baseline	SVM	2783	32	353	50

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Technique	Classifier	TN	FP	FN	TP
Baseline	Neural network	2752	63	350	53
ROS	Logistic regression	2136	679	119	284
ROS	Decision tree	2251	564	166	237
ROS	Random forest	2625	190	225	178
ROS	SVM	2086	729	120	283
ROS	Neural network	1932	883	87	316
RUS	Logistic regression	2075	740	117	286
RUS	Decision tree	1903	912	120	283
RUS	Random forest	1965	850	84	319
RUS	SVM	2061	754	114	289
RUS	Neural network	1205	1610	68	335
SMOTE	Logistic regression	2122	693	116	287
SMOTE	Decision tree	2364	451	201	202
SMOTE	Random forest	2579	236	209	194
SMOTE	SVM	2045	770	115	288
SMOTE	Neural network	2024	791	88	315
ADASYN	Logistic regression	2060	755	106	297
ADASYN	Decision tree	2351	464	175	228

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Technique	Classifier	TN	FP	FN	TP
ADASYN	Random forest	2570	245	205	198
ADASYN	SVM	1947	868	95	308
ADASYN	Neural network	1869	946	71	332
Class weight	Logistic regression	2129	686	117	286
Class weight	Decision tree	2188	627	156	247
Class weight	Random forest	2503	312	186	217
Class weight	SVM	2063	752	104	299
Class weight	Neural network	1953	862	104	299

Table 5. Paired Wilcoxon comparisons of balanced accuracy against baseline. Exact paired probabilities are reported with Holm adjustment for the five planned comparisons.

Comparison	W	Exact p	Holm-adjusted p
ROS vs baseline	0.0	0.0020	0.0098
RUS vs baseline	1.0	0.0039	0.0098
SMOTE vs baseline	0.0	0.0020	0.0098
ADASYN vs baseline	0.0	0.0020	0.0098
Class weight vs baseline	0.0	0.0020	0.0098

3.6 Selected conventional confusion matrices

Tables 6-9 present representative held-out results in conventional 2x2 form. Rows denote the observed class and columns denote the predicted class; the matrices make the false-positive inspection burden and false-negative defect burden directly visible.

International Journal of Applied Science and Engineering Review

IJASER

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Table 6. Confusion matrix for Baseline with Random Forest. Counts aggregate natural-prevalence held-out predictions across KC1 and PC1.

Observed class	Predicted clean	Predicted defective
Actual clean	2755	60
Actual defective	297	106

Table 7. Confusion matrix for ROS with Logistic regression. Counts aggregate natural-prevalence held-out predictions across KC1 and PC1.

Observed class	Predicted clean	Predicted defective
Actual clean	2136	679
Actual defective	119	284

Table 8. Confusion matrix for SMOTE with Random Forest. Counts aggregate natural-prevalence held-out predictions across KC1 and PC1.

Observed class	Predicted clean	Predicted defective
Actual clean	2579	236
Actual defective	209	194

Table 9. Confusion matrix for Class weight with SVM. Counts aggregate natural-prevalence held-out predictions across KC1 and PC1.

Observed class	Predicted clean	Predicted defective
Actual clean	2063	752
Actual defective	104	299

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

4. DISCUSSION

4.1 Principal findings

The experiment answers RQ1 affirmatively but conditionally. Every correction strategy improved mean balanced accuracy and minority recall relative to no correction. The improvement was not free: precision and specificity usually declined. Consequently, a claim that resampling "improved performance" would be incomplete unless the metric and cost trade-off were named. ROS offered the highest mean F1-score, SMOTE the highest mean PR-AUC, and class weighting a strong balance without creating new records. RQ2 was also supported. The correction ranking changed across learners. Random forest had the strongest uncorrected performance and gained less uniformly from balancing than logistic regression or SVM. RUS was particularly aggressive for the neural network, where high recall was offset by poor specificity. Class weighting was attractive for SVM and logistic regression, while synthetic oversampling was competitive for random forest. These interactions show that imbalance correction is part of model selection, not a preprocessing switch that can be chosen independently.

For RQ3, the omnibus rank analysis indicated differences across techniques, and paired follow-up comparisons generally showed correction improving balanced accuracy over baseline. However, the corrected methods were close to one another. The practical evidence is stronger for "use a minority-aware strategy and evaluate it honestly" than for "always use SMOTE" or any other single prescription.

4.2 Relationship to prior work

The findings agree with studies showing that imbalance correction can raise probability of detection while increasing false alarms [4-6]. The increase in recall after ROS, SMOTE, ADASYN, RUS, or weighting is mechanistically plausible because every method raises the effective influence of defective modules during fitting. The corresponding fall in precision is also expected at fixed prevalence: when more modules are labeled positive, false positives rise unless ranking discrimination improves enough to compensate.

The small difference in PR-AUC between baseline and several corrected strategies deserves attention. Resampling can alter fitted probabilities and the default threshold while leaving rank ordering relatively stable. This explains why threshold-dependent metrics may improve dramatically while ROC-AUC or PR-AUC changes little. Saito and Rehmsmeier's warning that ROC analysis can look overly optimistic under imbalance remains relevant [1]. Reporting both ROC-AUC and PR-AUC prevents the conclusion from depending on one view of discrimination.

Tantithamthavorn et al. found that rebalancing can change not only performance but also the interpretation of defect models [4]. The present study did not estimate feature importance, yet the classifier-specific results are compatible with their broader message: correction changes the empirical learning problem. A

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

coefficient, split, or importance value obtained after aggressive undersampling describes the altered training distribution and should not automatically be interpreted as if it came from the full project population.

The results also align with broad evaluations of imbalanced classification showing that sampling methods are context dependent [17,18]. SMOTE reduces simple duplication but assumes that linear interpolation between minority neighbors is meaningful. Static software metrics are continuous, making interpolation numerically convenient, but synthetic combinations may not correspond to realizable modules. ADASYN further emphasizes difficult neighborhoods, which can help boundary learning or amplify mislabeled observations. RUS avoids synthetic points but throws away information, a cost visible in the reduced specificity of some models.

4.3 Why accuracy is inadequate

PC1 illustrates the accuracy paradox precisely. Predicting every module clean would be correct for 1,032 of 1,109 modules. Yet sensitivity, F1, and balanced accuracy for that rule would be 0, 0, and 0.5, respectively. An accuracy-led model selection process could therefore prefer a useless detector over a model that finds most defects while generating manageable false alarms.

Balanced accuracy corrects the equal-class weighting problem but does not measure alert quality. A model can obtain reasonable balanced accuracy through high recall and mediocre specificity while still producing low precision in a rare-event dataset. F1 adds precision but ignores true negatives. ROC-AUC is threshold independent but can appear favorable when the negative class is abundant. PR-AUC focuses on the positive class but is prevalence sensitive. Confusion matrices remain indispensable because they show the absolute inspection burden and missed-defect count.

The implication is not that one metric should replace accuracy. It is that software defect prediction requires a dashboard of complementary measures selected before results are examined. In the present study, balanced accuracy served as the statistical endpoint, PR-AUC and F1 supplied minority-sensitive confirmation, and matrices supplied operational meaning.

4.4 Implications for practice

Organizations with native class-weight support should include weighting as a first-line candidate. It preserves all observations, avoids synthetic-data assumptions, and performed competitively here. ROS is similarly simple and produced strong F1 results, but duplicated minority records can increase training cost and overfit idiosyncratic modules. SMOTE is reasonable when minority neighborhoods are sufficiently populated and continuous predictors make interpolation defensible. ADASYN should be treated

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

cautiously when labels are noisy because it concentrates attention on ambiguous regions.

RUS is useful when the majority class is extremely large or computation is constrained. Its information loss is most acceptable when clean modules are redundant. In small software projects, however, discarding most clean modules can destabilize a tree or neural network. Repeated sampling or balanced ensembles may reduce that instability, although those extensions were outside this comparison.

The choice should be tied to a deployment objective. If missing a defect is very costly, a high-recall corrected model may be preferred even with lower precision. If manual review capacity is scarce, PR-AUC, precision at a fixed inspection budget, or effort-aware measures may be more important. Teams should set a threshold using training data and a documented cost ratio rather than accept 0.50 automatically.

Correction must occur inside the validation loop. Performing SMOTE before partitioning is a common but serious error because related synthetic and original observations can appear on opposite sides of the evaluation. The resulting score answers how well the model recognizes contaminated neighbors, not how well it predicts unseen modules. The leakage-safe sequence used here should be treated as a minimum reproducibility requirement.

4.5 Implications for research

Future comparisons should use nested validation when tuning hyperparameters or thresholds. The outer loop should estimate generalization, while the inner loop selects classifier settings, correction ratios, neighbor counts, and thresholds. Repeated cross-validation would yield more stable variance estimates than a single five-fold partition. When many datasets are available, project should remain the independent unit for statistical inference rather than treating hundreds of correlated folds as independent observations. Probability calibration deserves separate study. Oversampling and weighting alter the effective class prior, so raw predicted probabilities may no longer estimate deployment prevalence. Calibration on untouched training-validation data can improve probability interpretation, but it should not reuse the final test fold. Evaluation should include calibration slope, intercept, Brier score, and decision curves when probabilities drive resource allocation.

Effort-aware defect prediction is another extension. A module-level false positive does not have the same cost when modules differ greatly in size. Measures such as recall at 20% of lines inspected or cost-effectiveness curves can supplement classification metrics [19]. Temporal validation is also preferable when release chronology is available because random folds may place modules from related development periods in training and test sets.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Finally, data quality should be examined explicitly. Public NASA datasets have multiple releases and documented inconsistencies [8]. Replication across cleaned NASA variants, PROMISE Java projects, AEEEM projects, and industrial systems would test external validity. Reporting exact file hashes and row-level cleaning rules would make disagreements traceable. [20,21]

4.6 Strengths

The study has five main strengths. First, it compares six techniques spanning no correction, duplication, deletion, synthetic generation, adaptive generation, and loss weighting. Second, it applies the same techniques to five classifiers with different inductive biases. Third, all transformations and sampling occur within training folds, preventing leakage. Fourth, evaluation includes seven scalar measures and confusion matrices rather than accuracy alone. Fifth, paired nonparametric tests respect the repeated comparison structure.

The use of natural-prevalence test folds is particularly important. Balancing the test data would artificially change precision and PR-AUC and would no longer represent the alert burden faced by a project. Fixed thresholds and fixed classifier configurations also make the correction effect easier to interpret, even though they do not maximize every learner.

4.7 Limitations

Only two projects were included, and both come from the NASA/PROMISE ecosystem. The effective statistical sample is ten project-classifier blocks, so exact tests have limited power and cannot establish small differences among corrected techniques. Results may differ for object-oriented metrics, process metrics, just-in-time prediction, cross-project transfer, or modern learned representations.

The experiment used one random five-fold partition and one fixed correction ratio. Repeated partitions could reveal sampling variance. SMOTE and ADASYN used five neighbors without tuning, and all corrections targeted approximate class equality. Partial balancing may yield better precision-recall trade-offs. Similarly, classifier hyperparameters were defensible but not extensively tuned. The study estimates average behavior under a controlled setup, not the maximum achievable score for each pairing.

The neural-network weighting path depends on implementation support for observation weights; software versions should therefore be recorded in a formal replication package. Neural networks are also sensitive to initialization and early stopping, so one seed understates training variability. Deep architectures were intentionally excluded because the available tabular samples, especially PC1's 77 defective modules, do not support a fair high-capacity comparison.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Confusion matrices were aggregated across projects for compact reporting. Aggregation aids operational interpretation but can hide project heterogeneity. Fold-level files should be consulted for replication, and project-level matrices should be reported when a deployment decision concerns one codebase. PR-AUC is prevalence dependent, so its pooled mean should not be compared directly with studies using different defect rates.

The binary label indicates whether a module is defective, not defect severity, number of faults, or repair cost. Some labels may be incomplete or noisy. No causal conclusion can be drawn: correction changes predictive fitting but does not reveal why defects occur. Finally, the analysis does not evaluate interpretability, fairness across components, or downstream developer behavior after alerts.

5. Robustness considerations and decision framework

5.1 Reading the results as trade-offs

The central practical error in imbalance research is to read a performance table as a horse race. The largest number in one column is not automatically the best model because the columns encode different consequences. Precision asks whether a warning deserves attention; sensitivity asks how many defective modules escape; specificity asks how much clean code can be safely deprioritized. F1 combines the first two but does not recognize correct rejections. Balanced accuracy recognizes both classes but treats a percentage-point increase in sensitivity as exchangeable with the same increase in specificity. ROC-AUC and PR-AUC assess rankings rather than one operating decision. A decision maker must therefore begin with a cost model, even if that model is qualitative.

Consider two organizations. A safety-critical flight project may prefer a model that finds nearly every defective module, accepting an extensive review queue. A small commercial team preparing a minor release may be able to inspect only a fixed fraction of modules. For the first organization, sensitivity and false negatives dominate; RUS or an aggressively weighted model might be defensible. For the second, precision, PR-AUC, or recall at a fixed alert budget matters more; random forest with synthetic sampling may provide a better ordering. Neither preference is visible from ordinary accuracy.

The confusion matrices also prevent a misleading percentage-only interpretation. A ten-percentage-point specificity reduction affects many more modules than a ten-percentage-point sensitivity gain when clean modules greatly outnumber defective modules. That does not make the trade undesirable, but it makes the workload explicit. Before deployment, teams should translate FP into reviewer-hours and FN into expected defect exposure. If approximate costs are available, a threshold can be chosen to minimize expected cost on validation data. Correction and threshold selection should then be nested inside the training process.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

5.2 Baseline and random oversampling

An uncorrected baseline remains necessary even when imbalance is obvious. It reveals whether the classifier already ranks minority cases well and provides the reference needed to quantify the effect of intervention. In this experiment, random forest was comparatively capable without correction, whereas linear and margin-based models had conservative default classifications. Omitting the baseline would have concealed the fact that some gains came primarily from moving the operating point rather than improving ranking discrimination. [22-24]

ROS is appealing because it is transparent: no observations are discarded and no feature vectors are invented. Its primary effect is to make minority errors appear more often during optimization. For deterministic learners, duplicating rows is closely related to assigning larger case weights, although stochastic optimization and regularization can make the fitted result differ. ROS performed well on F1 in this study, suggesting that simple re-expression of the training distribution was sufficient to shift several boundaries.

The weakness of ROS is that it repeats any minority noise. A mislabeled or exceptional defective module may appear many times, encouraging a tree to build a branch around it or a neural network to memorize it. Duplication also enlarges the training set, which matters for more expensive learners. A responsible ROS workflow should examine minority duplicates and influential observations, use repeated validation, and compare against class weighting. If weighting achieves similar results, it may be the more economical option.

5.3 Random undersampling

RUS changes the problem from learning among all observed clean modules to learning among a selected subset. It can be extremely efficient when the majority class contains millions of redundant observations. It may also expose a decision boundary that is otherwise overwhelmed by majority density. The high recall obtained in several combinations is consistent with these properties.

The information cost is substantial for KC1 and especially PC1. Equalizing PC1 training folds requires discarding most clean modules. Those discarded cases may include legitimate boundary examples that distinguish risky from safe code. A classifier trained without them can label broad regions as defective, raising FP and lowering specificity. A single RUS sample also introduces selection variance; another seed may retain a different clean subset and yield different splits or support vectors.

Repeated undersampling ensembles are a natural extension. Multiple models can be fitted to different majority subsets, and their scores can be averaged. This retains computational advantages while allowing

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

more majority observations to contribute across the ensemble. Such an extension would no longer be a pure comparison of the six requested techniques, so it was not included, but practitioners should consider it when the majority class is very large.

5.4 SMOTE and ADASYN

SMOTE assumes that the segment between two nearby minority observations lies in a meaningful minority region. For continuous software metrics, the arithmetic operation is well defined. Its semantic validity is less certain. A synthetic module can combine complexity and size values that do not occur together in compiled software, and correlated metric constraints may be violated. Standardization reduces domination by large-scale features but does not guarantee a feasible program representation. [25]

Despite that caveat, SMOTE can broaden sparse minority regions without merely repeating records. Its strong PR-AUC and competitive F1 results here suggest that it preserved or modestly improved ranking quality. The result should not be generalized to categorical predictors without a mixed-data variant, nor to extremely small minority samples where neighbor geometry is unreliable. Feature selection, correlation analysis, and local density diagnostics should precede synthetic generation in a production study.

ADASYN changes the allocation rule by generating more cases near minority observations that are surrounded by majority neighbors. The intended benefit is to focus learning on difficult regions. The danger is that difficulty can reflect overlap, measurement error, or mislabeling rather than a legitimate boundary. When a mislabeled minority point sits inside a clean cluster, ADASYN may manufacture additional misleading points around it. The method's competitive mean results do not eliminate this risk. The comparison between SMOTE and ADASYN therefore supports validation rather than dogma. Similar aggregate scores may arise from different local decision surfaces. Researchers should report the number of synthetic cases, sampling ratio, neighbor count, scaling method, and any failure fallback. Visual inspection of two-dimensional projections is not proof of validity but can reveal gross artifacts. More rigorous checks can examine whether generated vectors violate known metric bounds or multivariate relationships.

5.5 Class weighting

Class weighting operates through the loss function. It preserves the observed feature vectors and changes the penalty attached to errors. For logistic regression and SVM, this directly shifts the fitted boundary toward the minority class. For trees, weighting changes impurity calculations and leaf decisions. For random forests, it interacts with bootstrap samples and split selection. These mechanisms explain why equal class weights do not create identical results across classifiers.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Weighting performed close to the best resampling approaches and was especially useful for SVM. This makes it a strong default candidate when implementation support is reliable. It avoids the geometric assumptions of synthetic methods and the information loss of RUS. It also keeps the training-set size fixed. Nevertheless, it does not solve label noise, dataset shift, or poor feature representation, and it can produce poorly calibrated probabilities when the effective training objective differs from deployment prevalence. Weights should be regarded as hyperparameters rather than immutable inverse-frequency formulas. Equal total weight for each class is a reasonable starting point, but the optimal ratio depends on costs and classifier behavior. A nested validation loop can search a modest grid of ratios. The final reported model should include the selected weight, decision threshold, and calibration procedure so that others can reproduce the operational trade-off.

5.6 Classifier-specific interpretation

Logistic regression offers a useful diagnostic because its linear score is comparatively easy to inspect. A large sensitivity increase with little change in ROC-AUC suggests that weighting or resampling changed the intercept or boundary placement more than the ordering of modules. Coefficients should be interpreted after checking collinearity because Halstead, McCabe, and size measures are related. Standardization makes coefficient magnitudes more comparable but does not eliminate instability.

Decision trees are attractive to practitioners because their rules can be communicated, yet they are unstable under resampling. Duplicating or deleting a few observations can change an early split and all descendants. Depth and minimum-leaf constraints reduced this instability but could not remove it. A corrected tree should be evaluated over repeated samples, and its apparent explanatory rules should not be treated as causal statements about software quality.

Random forest averages many trees and was the strongest baseline in several metrics. Its robustness does not mean imbalance is irrelevant. A minority observation still competes for representation in bootstrap samples and split criteria, while the default threshold may remain conservative. Sampling and weighting can improve detection, but the baseline forest's strong ranking explains why PR-AUC gains were smaller than recall gains.

RBF-SVM learns a nonlinear margin whose placement depends directly on class penalties and local density. Weighting is therefore a natural technique, and the empirical results support it. Probability outputs from SVM are derived through an additional calibration procedure, so ROC-AUC and PR-AUC usefully complement thresholded classifications. For operational probability estimates, calibration should be performed in a nested, untouched validation partition.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

The neural network had enough flexibility to respond strongly to resampled distributions, sometimes too strongly. Early stopping and regularization restrained overfitting, but the limited number of defective PC1 modules remains a fundamental constraint. Multiple initializations and confidence intervals would be required before choosing a neural network over simpler models. High recall alone is insufficient if the resulting review queue is impractical.

5.7 Statistical inference and uncertainty

The paired design is more informative than comparing unpaired averages because every technique encounters the same project-classifier blocks. Friedman ranks reduce sensitivity to scale differences and outliers, while Wilcoxon tests examine paired directions and magnitudes without assuming normality. Holm adjustment limits the chance that one of five baseline comparisons appears significant merely because several tests were performed.

Statistical significance does not identify a deployment winner. With enough blocks, a tiny change can be significant but irrelevant; with ten blocks, a useful change may fail to reach a conventional threshold. The mean balanced-accuracy gain, F1 and PR-AUC behavior, and confusion-matrix burden are therefore as important as adjusted p-values. Confidence intervals from repeated nested validation would improve uncertainty quantification in a larger replication.

Classifier blocks from the same project are not fully independent because they share observations. The rank test treats each project-classifier pairing as a repeated block for technique comparison, a common benchmarking convention, but it does not create new independent projects. This is why the study's external-validity claims remain cautious. A future analysis with many projects should aggregate within project or use a hierarchical model that represents project and classifier effects explicitly.

5.8 Deployment checklist

Before fitting, a team should define the positive class, verify prevalence, inspect label provenance, remove train-test duplicates, and document metric meanings. It should select at least one uncorrected baseline, one data-level correction, and one algorithm-level correction. All imputation, scaling, feature selection, correction, and threshold tuning must occur inside training data. The untouched test fold should retain natural prevalence.

During evaluation, the team should report precision, sensitivity, specificity, F1, balanced accuracy, ROC-AUC, PR-AUC, and the confusion matrix. Fold-level distributions should accompany means. When many methods are compared, a paired omnibus test and multiplicity-controlled follow-up tests should be used. The independent unit should be the project or release, not thousands of correlated module predictions.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Before deployment, stakeholders should convert the matrix into expected workload and risk at the anticipated module volume. The threshold should be chosen under a declared review budget or cost ratio. Probability calibration should be assessed if scores are interpreted as risks. Drift in prevalence, coding practice, tooling, and defect-label collection should be monitored after deployment. A model should be retrained or retired when its calibration or minority detection degrades.

This checklist clarifies the role of imbalance correction. Correction is not a guarantee of useful prediction; it is one controlled intervention inside a broader assurance process. Its value can be established only through leakage-safe validation, complete reporting, and an operational decision rule.

6. CONCLUSION

In this March 2023 comparative experiment, class-imbalance correction materially changed software defect prediction. Across KC1 and PC1 and five classifiers, every evaluated correction increased mean balanced accuracy and recall relative to no correction. The benefit was consistently accompanied by a cost in specificity, precision, or both. ROS yielded the highest average F1-score, SMOTE the highest average PR-AUC, and class weighting offered a strong, computationally simple alternative. RUS was effective for recall but could generate an excessive false-positive burden.

No technique dominated every classifier and metric. The defensible practical recommendation is therefore to evaluate several minority-aware strategies inside leakage-safe cross-validation, preserve the natural class distribution in test folds, and choose using operational costs. Accuracy alone should not be used to select a defect predictor. Balanced accuracy, precision, sensitivity, specificity, F1, ROC-AUC, PR-AUC, and confusion matrices jointly provide the evidence needed to judge whether additional detected defects justify additional inspections.

Compliance with ethical standards

Acknowledgments

The author acknowledges the maintainers of the public NASA/PROMISE datasets and the open-source scientific Python community.

Disclosure of conflict of interest

The author declares no conflict of interest.

Statement of ethical approval

Ethical approval was not required because this study used public, non-personal software-metric datasets and involved no human or animal participants.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Funding

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

Data availability

The KC1 and PC1 datasets are publicly available through established software engineering data repositories. The analytical design, seed, model settings, fold-level metric fields, and aggregation rules are described in the manuscript to support replication.

Author contribution

Luke Akpan conceived the study, designed the analysis, interpreted the results, and prepared the manuscript.

REFERENCES

1. Saito T, Rehmsmeier M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS One. 2015;10(3):e0118432. doi:10.1371/journal.pone.0118432.
2. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321-357. doi:10.1613/jair.953.
3. He H, Bai Y, Garcia EA, Li S. ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of the 2008 IEEE International Joint Conference on Neural Networks. Piscataway: IEEE; 2008. p. 1322-1328. doi:10.1109/IJCNN.2008.4633969.
4. Tantithamthavorn C, Hassan AE, Matsumoto K. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. IEEE Trans Softw Eng. 2020;46(11):1200-1219. doi:10.1109/TSE.2018.2876537.
5. Tomar D, Agarwal S. Prediction of defective software modules using class imbalance learning. Appl Comput Intell Soft Comput. 2016;2016:7658207. doi:10.1155/2016/7658207.
6. Khadijah K, Sasongko PS. The comparison of imbalanced data handling method in software defect prediction. Kinetik. 2020;5(3):193-202. doi:10.22219/kinetik.v5i3.1049.
7. Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng. 2007;33(1):2-13. doi:10.1109/TSE.2007.256941.
8. Shepperd M, Song Q, Sun Z, Mair C. Data quality: some comments on the NASA software defect datasets. IEEE Trans Softw Eng. 2013;39(9):1208-1215. doi:10.1109/TSE.2013.11.
9. Breiman L. Random forests. Mach Learn. 2001;45:5-32. doi:10.1023/A:1010933404324.
10. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20:273-297. doi:10.1007/BF00994018.

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

11. Brodersen KH, Ong CS, Stephan KE, Buhmann JM. The balanced accuracy and its posterior distribution. In: 20th International Conference on Pattern Recognition. Piscataway: IEEE; 2010. p. 3121-3124. doi:10.1109/ICPR.2010.764.
12. Fawcett T. An introduction to ROC analysis. Pattern Recognit Lett. 2006;27(8):861-874. doi:10.1016/j.patrec.2005.10.010.
13. Demšar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res. 2006;7:1-30.
14. Holm S. A simple sequentially rejective multiple test procedure. Scand J Stat. 1979;6(2):65-70.
15. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825-2830.
16. Lemaître G, Nogueira F, Aridas CK. Imbalanced-learn: a Python toolbox to tackle the curse of imbalanced datasets in machine learning. J Mach Learn Res. 2017;18(17):1-5.
17. García V, Sánchez JS, Mollineda RA. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. Knowl Based Syst. 2012;25(1):13-21. doi:10.1016/j.knosys.2011.06.013.
18. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F. Learning from imbalanced data sets. Cham: Springer; 2018. doi:10.1007/978-3-319-98074-4.
19. Mende T, Koschke R. Effort-aware defect prediction models. In: 2010 14th European Conference on Software Maintenance and Reengineering. Piscataway: IEEE; 2010. p. 107-116. doi:10.1109/CSMR.2010.18.
20. Catal C, Diri B. A systematic review of software fault prediction studies. Expert Syst Appl. 2009;36(4):7346-7354. doi:10.1016/j.eswa.2008.10.027.
21. Hall T, Beecham S, Bowes D, Gray D, Counsell S. A systematic literature review on fault prediction performance in software engineering. IEEE Trans Softw Eng. 2012;38(6):1276-1304. doi:10.1109/TSE.2011.103.
22. Lessmann S, Baesens B, Mues C, Pietsch S. Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng. 2008;34(4):485-496. doi:10.1109/TSE.2008.35.
23. Kamei Y, Shihab E. Defect prediction: accomplishments and future challenges. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering. Piscataway: IEEE; 2016. p. 33-45. doi:10.1109/SANER.2016.56.
24. Ghotra B, McIntosh S, Hassan AE. Revisiting the impact of classification techniques on the performance of defect prediction models. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. Piscataway: IEEE; 2015. p. 789-800. doi:10.1109/ICSE.2015.91.
25. Bashir K, Li T, Yohannese CW, Yahaya M, Kahraman C. SMOTEFRIS-INFFC: handling the challenge of borderline and noisy examples in imbalanced learning for software defect prediction. J

ISSN: 2582-6271

Vol. 4, Issue.2, March-April 2023, page no. 184-211

Intell Fuzzy Syst. 2020;38(1):603-622. doi:10.3233/JIFS-179459.