



# Decision-boundary aware undersampling via Adversarial Influence for imbalanced software defect prediction<sup>☆</sup>

Shuo Feng<sup>a</sup>, Rongping Li<sup>a</sup>, Jacky Keung<sup>b</sup>, Xiao Yu<sup>c</sup>, Yucheng Shi<sup>a</sup>,<sup>\*</sup>, Mingliang Xu<sup>a</sup>

<sup>a</sup> School of Computer Science and Artificial Intelligence, Zhengzhou University, Zhengzhou, 450001, China

<sup>b</sup> Department of Computer Science, City University of Hong Kong, Hong Kong, 999077, China

<sup>c</sup> State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, 310058, China

## ARTICLE INFO

### Keywords:

Software defect prediction  
Class imbalance  
Undersampling  
Adversarial influence  
Adversarial perturbation

## ABSTRACT

Class imbalance severely impairs the capacity of software defect prediction (SDP) models to distinguish between defective (minority class) and non-defective (majority class) software instances. Conventional undersampling methods alleviate this problem by reducing the number of non-defective instances. However, they frequently eliminate informative non-defective instances essential for constructing reliable decision boundary, thereby compromising prediction performance. The key challenge lies in determining which non-defective instances should be retained to preserve effective decision boundary while balancing class distributions. To address this challenge, we propose Adversarial Influence-Based Intelligent Undersampling (AIUS), a decision-boundary-aware undersampling for imbalanced SDP. In AIUS, we introduce the Adversarial Influence metric to quantify the importance of non-defective instances to the model's decision boundary and formulate non-defective instance selection as an Adversarial Influence-based ranking problem. Specifically, we compute Adversarial Influence scores via adversarial perturbations on a surrogate model to rank non-defective instances and select the top-ranked ones most critical to decision boundary for constructing a balanced training dataset. Experiments on 18 public defect datasets and five commonly used classifiers show that AIUS consistently improves imbalance-robust performance, achieving average gains of 9.50% in Balance, 3.43% in Area Under the ROC Curve (AUC), and 17.02% in Matthews Correlation Coefficient (MCC) over representative resampling baselines, with statistically significant improvements in most settings (Wilcoxon signed-rank test with Cliff's  $\delta$ ). These results suggest that decision-boundary-aware instance selection is an effective and practical strategy for handling class imbalance in software defect prediction.

## 1. Introduction

Software defect prediction (SDP) supports reliability assurance by prioritizing code entities that are most likely to contain defects (Kumar and Sagar, 2025; Tian and Du, 2026; Tang et al., 2025), enabling practitioners to allocate limited inspection and testing effort more effectively (Qiao et al., 2020; Yu et al., 2024). In operational settings, SDP models are judged not by overall accuracy but by their ability to detect defect-prone instances with acceptable false-alarm rates, since missed defects increase post-release failure risk while excessive false positives waste verification effort. This trade-off is drastically amplified in mission-critical systems like spacecraft fault diagnosis software (Devarajulu et al., 2026), where overlooked anomalies may result in catastrophic mission failures such as the Mars Climate Orbiter crash.

A persistent obstacle in SDP is severe class imbalance (Japkowicz and Stephen, 2002), where non-defective instances far outnumber defective ones (Nikpour et al., 2026). Under such skew, many learners become biased toward the non-defective instances, producing deceptively optimistic accuracy yet poor defect recall or unstable decision boundaries (Zhang et al., 2026). This issue is amplified by the frequent class overlap in software metrics: defective and non-defective instances often occupy adjacent regions in the feature space, so preserving the right non-defective instances is crucial for learning a reliable separator.

### 1.1. Motivations

Data-level resampling is widely adopted to address imbalance because it is classifier-agnostic and easy to integrate into existing SDP

<sup>☆</sup> Editor: Professor Tao Zhang.

<sup>\*</sup> Corresponding author.

E-mail addresses: [fengshuo@zzu.edu.cn](mailto:fengshuo@zzu.edu.cn) (S. Feng), [lirongping@gs.zzu.edu.cn](mailto:lirongping@gs.zzu.edu.cn) (R. Li), [jacky.keung@cityu.edu.hk](mailto:jacky.keung@cityu.edu.hk) (J. Keung), [xiao.yu@zju.edu.cn](mailto:xiao.yu@zju.edu.cn) (X. Yu), [ieychshi@zzu.edu.cn](mailto:ieychshi@zzu.edu.cn) (Y. Shi), [ixumingliang@zzu.edu.cn](mailto:ixumingliang@zzu.edu.cn) (M. Xu).

<https://doi.org/10.1016/j.jss.2026.113105>

Received 4 June 2026; Received in revised form 1 September 2026; Accepted 5 September 2026

Available online 12 September 2026

0164-1212/© 2026 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

pipelines. In particular, undersampling can be effective because it avoids synthesizing defective instances (Feng et al., 2021b), but most existing undersampling approaches remove non-defective instances using randomness, clustering heuristics (Liu et al., 2008), or distribution-level redundancy measures (Lin et al., 2017). These strategies rarely evaluate whether a specific non-defective instance is boundary-defining for the downstream classifier. Consequently, undersampling may inadvertently discard informative non-defective instances near the class boundary (e.g., borderline cases Han et al., 2005), distort the learned boundary, and degrade performance on imbalance-robust measures such as MCC and Balance (Feng et al., 2024).

Recent advances in adversarial machine learning provide a complementary perspective (He et al., 2023): Studies of adversarial vulnerability (Ren et al., 2022) suggest that instances whose predictions change substantially under small perturbations (Lin et al., 2025) tend to lie near locally sensitive decision regions (Dong et al., 2026; Zhao et al., 2025). This observation motivates a boundary-aware undersampling strategy for SDP: instead of treating all non-defective instances as equally removable, we prioritize keeping instances whose exclusion would substantially alter the classifier's decision boundary, as these instances are essential for preserving the model's discriminative power.

A practical challenge, however, is that most adversarial perturbation generation techniques (Lu et al., 2025; Yang et al., 2023) rely on gradient information of the model. In SDP, commonly used classifiers such as k-nearest neighbors (KNN) and Random Forests (RF) are typically treated as gradient-unavailable black-box targets (Ran et al., 2025), which render direct gradient-based adversarial generation infeasible. On the one hand, they lack continuously differentiable prediction functions and cannot yield input gradients required for adversarial perturbations. On the other hand, a real-world constraint arising from industrial deployment practices is that predictive models are encapsulated as closed inference service endpoints. Practitioners can only submit feature instances to retrieve model prediction outputs via these interfaces, without any access to the underlying model architectures and trainable weights. Within this oracle-only query paradigm, such predictive models are conventionally categorized as black-box models. It is critical to clarify that our definition of black-box models differs fundamentally from that in explainable artificial intelligence (XAI) (Kaya et al., 2026). In adversarial security, the term black-box does not mean the model is uninterpretable. Instead, it only indicates the inquirer has no access to the target's architecture and parameters.

Fortunately, prior work has demonstrated the transferability of adversarial examples across models (Papernot et al., 2016a). Adversarial samples generated using a substitute model can expose similar vulnerabilities in a target black-box model, even when the two models differ substantially in architecture and learning mechanisms, provided that they are trained for the same task (e.g., identifying defective instances). This transferability property allows adversarial perturbations to be generated using a surrogate model and subsequently evaluated through the response of the target black-box classifier, thereby enabling the assessment of instance importance in practical SDP scenarios.

## 1.2. Our contributions

Motivated by these observations, we repurpose the surrogate-based adversarial perturbation mechanism as a constructive analytical tool and propose an Adversarial Influence-Based Intelligent Undersampling (AIIUS) method for software defect prediction. AIIUS introduces a novel metric, termed Adversarial Influence, to quantify the importance of non-defective instances. Specifically, the Adversarial Influence of a non-defective instance is defined as the squared difference between the prediction probabilities produced by the target model for the original instance and its corresponding adversarial counterpart. A larger value indicates greater sensitivity to perturbation and a stronger influence on the decision boundary, thereby implying higher instance importance. The AIIUS framework proceeds as follows. First, non-defective instances

are clustered and representative instances closest to cluster centroids are selected to reduce redundancy while preserving the overall data distribution. Second, a substitute model is trained to approximate the behavior of the target classifier, and perturbed instances are generated to evaluate the Adversarial Influence of the selected instances. Finally, non-defective instances are ranked according to their Adversarial Influence scores, and the most influential instances are retained to construct a balanced training set by matching the number of defective instances. Through this design, AIIUS both evenly balances class distribution and explicitly retains non-defective instances, which serve as a proxy for local boundary relevance. To comprehensively validate the effectiveness and rationale of AIIUS, our experimental evaluation is designed to answer the following three research questions (RQs):

**RQ1: How does AIIUS perform compared to baseline methods?** We compare AIIUS against six representative baseline methods, including the Undersampling approach based on Feature Importance and Double Side Filter (UFIDSF) (Zheng et al., 2025), Generative Adversarial Network (SB-GAN) (Akritidis et al., 2023), Cluster-based undersampling (Cluster) (Lin et al., 2017), Random Undersampling (RUS) (Liu et al., 2008), Borderline-SMOTE (Han et al., 2005), and Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002) across 18 publicly available datasets from four authoritative repositories (AEEEM D'Ambros et al., 2012, PROMISE Menzies et al., 2012, SOFTLAB Turhan et al., 2009, and NASA Shepperd et al., 2013), using five widely-adopted classifiers: K-Nearest Neighbors (KNN), Random Forest (RF), Decision Tree (Tree), Logistic Regression (LR), and Naïve Bayes (NB). Performance is assessed using five metrics: Matthews Correlation Coefficient (MCC), Area Under the Curve (AUC), Balance, Probability of Detection (PD) and Probability of False Alarm (PF). Results show that AIIUS outperforms the competing methods in terms of balance, AUC, and MCC by 9.50%, 3.43%, and 17.02% on average.

**RQ2: What is the impact of the clustering strategy on the effectiveness and stability of AIIUS?** We investigate the necessity and impact of the clustering step by evaluating AIIUS under different clustering ratios (no-cluster, 1×, 2×, 3×, 3.5× the number of defective instances). Results show that clustering enhances both the effectiveness and stability of AIIUS, with a moderate 3× clustering ratio providing the best trade-off across classifiers.

**RQ3: Why was a standard MLP chosen as the surrogate model in the experiments?** We justify this design choice by comparing the performance of AIIUS when using different surrogate architectures, including a deeper MLP, a CNN, and a ResNet. Results show that AIIUS is not sensitive to the surrogate architecture as long as it reasonably approximates the target decision function, and a standard MLP is sufficient for SDP datasets.

The main contributions of this study are summarized as follows:

- We introduce Adversarial Influence, a novel metric that quantifies instance importance based on decision-boundary sensitivity, thereby bridging adversarial machine learning and data resampling for class imbalance.
- We propose a novel class imbalance handling method, AIIUS, which for the first time incorporates adversarial vulnerability analysis into data-level resampling for software defect prediction.
- We conduct extensive experiments on multiple datasets and classifiers. The results demonstrate that AIIUS achieves competitive and often superior performance compared to representative baseline methods, and additional analyses are provided to justify key design choices.
- We have open-sourced the full implementation of AIIUS on GitHub to facilitate reproduction of our experiments (<https://github.com/pistachio528/AIIUS>).

**Table 1**  
Abbreviations

| Abbreviation | Full name                                             |
|--------------|-------------------------------------------------------|
| AIUS         | Adversarial Influence-Based Intelligent Undersampling |
| AUC          | Area Under the ROC Curve                              |
| CNN          | Convolutional Neural Network                          |
| DE           | Differential Evolution                                |
| GAN          | Generative Adversarial Network                        |
| IR           | Imbalance Ratio                                       |
| MCC          | Matthews Correlation Coefficient                      |
| MLP          | Multilayer Perceptron                                 |
| PCA          | Principal Component Analysis                          |
| PGD          | Projected Gradient Descent                            |
| ResNet       | Residual Network                                      |
| SB-GAN       | Semantic-Balanced Generative Adversarial Network      |
| SDP          | Software Defect Prediction                            |
| XAI          | Explainable Artificial Intelligence                   |

### 1.3. Organization

The remainder of this paper is organized as follows: Section 2 introduces the background and related work; Section 3 details the proposed methodology; Section 4 describes the experimental design; Section 5 presents the experimental results; Section 6 discusses the research work; Section 7 presents the threats to validity; Section 8 concludes the paper and suggests future research directions.

For ease of reference, the main abbreviations used in this paper are summarized in Table 1 in alphabetical order.

## 2. Related works

### 2.1. Class imbalance

In addressing class imbalance, existing approaches can be broadly categorized into algorithm-level methods — such as cost-sensitive learning (Gupta et al., 2022; Guido et al., 2023; Petrides and Verbeke, 2022) and ensemble learning (Abedin et al., 2023; Bai et al., 2022; Khan et al., 2024; Zhao et al., 2023) — and data-level resampling techniques. Among them, resampling has been extensively adopted in software defect prediction (SDP) (Feng et al., 2021a; Goyal, 2022; Bennin et al., 2018), aiming to balance class distributions through oversampling minority (defective) instances or undersampling majority (non-defective) instances.

Random Oversampling (ROS) duplicates minority samples and is prone to overfitting (García et al., 2012). SMOTE (Chawla et al., 2002) alleviates this issue by interpolating between minority samples, yet it may introduce noise and exacerbate class overlap. Subsequent variants focus on boundary-aware synthesis: Borderline-SMOTE (Han et al., 2005) generates samples near decision boundaries, while ADASYN (He et al., 2008) adaptively increases synthetic samples for hard-to-classify instances. These methods have shown effectiveness in SDP (Hossen et al., 2020), but still rely on heuristic interpolation strategies.

Recently, Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) have been explored for imbalance mitigation by learning minority class distributions and generating more realistic synthetic samples. Empirical studies in SDP demonstrate that GAN-based oversampling can outperform traditional techniques such as SMOTE (Eivazpour and Keyvanpour, 2019; Chouhan and Rathore, 2021). To improve training stability and sample quality, SB-GAN (Akritidis et al., 2023) incorporates prior filtering of borderline and noisy samples. However, GAN- and VAE-based methods generally require large training datasets, suffer from training instability, and incur substantially higher computational costs, which limits their applicability in SDP scenarios with small and sparse datasets.

Undersampling methods reduce majority-class instances but risk discarding informative samples. Random Undersampling (RUS) (Liu et al., 2008) is simple yet information-destructive. Clustering-based

methods, such as K-means undersampling (Lin et al., 2017), retain cluster representatives but are sensitive to parameter selection and may distort data distributions (Feng et al., 2024). To address these issues, more sophisticated undersampling strategies have been proposed. UFIDSF (Zheng et al., 2025) integrates feature importance estimation with double-sided noise filtering to eliminate low-contribution and noisy instances. CBR (Akritidis and Bozanis, 2024) combines hierarchical clustering-based outlier removal with centroid-based oversampling to jointly perform undersampling and oversampling. Learning-based methods further formulate undersampling as an optimization problem, including WCP (Vashisht and Rizvi, 2022), and LTRUS (Feng et al., 2024), which rank majority instances by complexity or contribution and remove the least informative ones.

Different from existing resampling strategies, the proposed AIUS generates adversarial variants of non-defective instances via PGD attacks using a differentiable MLP surrogate. By quantifying instance importance through Adversarial Influence with respect to the decision boundary, AIUS performs decision-boundary-aware and fine-grained undersampling, enabling more effective retention of informative non-defective while alleviating class imbalance.

This formulation also distinguishes AIUS from instance-selection methods based on hardness, uncertainty, or margin criteria. Hardness-based approaches identify difficult instances via geometric neighborhood structure—an instance is considered hard if it resides near examples of the opposite class. Uncertainty sampling selects instances for which the classifier’s predicted probabilities are closest to the decision threshold, while margin-based selection retains instances with the smallest gap between the top-two class probabilities. These criteria share a common limitation: they are static and classifier-agnostic, reflecting where an instance sits in feature space or what a classifier currently outputs at a single point, without probing how the model’s prediction would change under a local perturbation.

Adversarial Influence addresses this limitation directly. Rather than measuring a classifier’s instantaneous indecision, it quantifies the squared change in predicted defect probability when a small, targeted adversarial perturbation is applied to an instance. This makes it a dynamic, model-response-based signal: it does not ask “is the classifier unsure here?” but “does perturbing this instance actively shift the classifier’s decision?” The distinction matters in SDP, where class overlap is prevalent. A high-uncertainty instance may reside in a flat, noisy region where the classifier is broadly indecisive but no single instance is boundary-critical; a high-influence instance, by contrast, is one whose presence or absence would meaningfully destabilize the learned boundary. Furthermore, because Adversarial Influence is computed via a differentiable surrogate model and evaluated through black-box queries to the target classifier, the importance score is explicitly coupled to the target classifier’s behavior rather than derived from data geometry alone—yielding a task-adaptive and classifier-aware notion of instance importance that static criteria cannot provide.

### 2.2. Surrogate-based attack

Surrogate-based attack, also known as black-box substitute attack, is a powerful paradigm for generating adversarial examples when the internal information of the target model is unknown (black-box setting). The core idea of this paradigm is that an attacker trains a local, fully accessible substitute model by repeatedly querying the target model and observing its input-output pairs to approximate the target model’s decision boundary. Subsequently, the attacker uses white-box attack methods to generate adversarial examples on the substitute model. Due to the transferability of adversarial examples — meaning samples effective against one model are often effective against another — these generated examples have a high probability of deceiving the original black-box target model (Papernot et al., 2016b).

Early pioneering work by Papernot et al. (2017) systematically elaborated on substitute model training, where they used models with

the same architecture as the target or models with strong fitting capabilities (e.g., MLP) as substitutes, trained on synthetic or public datasets labeled by the target model, to effectively mimic the target model's decision behavior. Regarding attack methods, various white-box techniques for generating adversarial examples on the substitute model exist, among which gradient-based attacks are most common, such as the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) and its iterative variant, Projected Gradient Descent (PGD) (Madry et al., 2018; Yang et al., 2023). PGD generates stronger adversarial examples with higher attack success rates by adding small perturbations along the gradient direction over multiple iterations and projecting the result into a specified norm ball. To enhance the transfer success rate of adversarial examples from the substitute model to the target model, researchers have proposed various transferability enhancement techniques, such as input transformation (Xie et al., 2019) and momentum integration (Tao et al., 2025), which make the attack process more stable and generalizable, thereby enhancing transferability.

Traditionally, substitute model attacks have been widely applied to assess the security and robustness of machine learning models, particularly in the field of image recognition. However, its core component — the concept of probing and exploiting the vulnerabilities of a model's decision boundary through controlled adversarial perturbations — holds broader implications.

It is important to clarify that this study does not aim to attack or compromise SDP models. Instead, we repurpose the surrogate-based adversarial perturbation mechanism as a constructive analytical tool: by measuring how sensitively the target classifier responds to controlled perturbations of individual non-defective instances, we obtain a principled quantification of their importance to the decision boundary. Specifically, we employ a differentiable MLP as the substitute model and use PGD attacks to generate adversarial counterparts for each non-defective instance. An instance that exhibits substantial fluctuations in prediction probability under minor perturbations is likely to lie in a locally boundary-sensitive region and is therefore may provide useful information for defining the class separation. Based on this insight, we propose the Adversarial Influence metric, transforming techniques from the adversarial attack domain into an advanced measure of sample informativeness, thereby offering a novel perspective for addressing the issues of class imbalance and overlap in SDP. Notably, the generated adversarial examples are fed into the target classifier solely for probability prediction to compute the Adversarial Influence score; they are never used to retrain the target classifier. Consequently, evaluating or guaranteeing adversarial robustness against malicious inputs is not addressed here, as it would require adversarial training or robust optimization techniques that are orthogonal to our data-level resampling objective.

### 3. Proposed method

This paper proposes AIIUS, an Adversarial Influence-based undersampling for software defect prediction. The core idea is to identify and retain the most informative non-defective instances (i.e., those that exert a strong influence on the decision boundary of the classifier) while removing redundant majority-class samples. The overall procedure consists of three sequential stages:

(1) **Representative non-defective instances selection.** The non-defective instances are first clustered, and the centroid-nearest instance from each cluster is selected to form a representative subset of the majority class.

(2) **Surrogate-based Adversarial Influence computation.** A differentiable surrogate model is trained to approximate the decision behavior of the target classifier. Adversarial perturbations are then generated for the representative non-defective instances, and their Adversarial Influence is quantified based on the prediction variations of the target classifier.

(3) **Influence-guided undersampling.** Non-defective instances are ranked according to their Adversarial Influence scores, and the most

**Table 2**

Notation used in the AIIUS framework.

| Symbol                    | Description                                                                                   |
|---------------------------|-----------------------------------------------------------------------------------------------|
| $D$                       | Imbalanced dataset                                                                            |
| $N$                       | Number of instances in $D$                                                                    |
| $\mathbf{x}_i$            | Feature vector of the $i$ th instance                                                         |
| $y_i$                     | Class label of $\mathbf{x}_i$ , where $y_i = 1$ denotes defective and $y_i = 0$ non-defective |
| $S_{\text{maj}}$          | Set of non-defective (majority-class) instances                                               |
| $S_{\text{min}}$          | Set of defective (minority-class) instances                                                   |
| $F_{\text{target}}$       | Target black-box classifier                                                                   |
| $F_{\text{surrogate}}$    | Differentiable surrogate model approximating $F_{\text{target}}$                              |
| $\mathbf{x}^{\text{adv}}$ | Adversarial example of instance $\mathbf{x}$                                                  |
| $I(\mathbf{x})$           | Adversarial Influence of instance $\mathbf{x}$                                                |

influential ones are retained to construct a balanced and informative training dataset.

Table 2 summarizes the notations used throughout this paper. Unless otherwise specified, all symbols follow the definitions given therein. In addition, an algorithmic description (Algorithm 1) and a workflow illustration (Fig. 1) are provided to further facilitate understanding of the proposed methodology.

#### 3.1. Representative non-defective instances selection

In SDP, non-defective instances typically dominate the dataset and often exhibit high redundancy, forming dense regions in the feature space. Processing all such instances is both computationally inefficient and unnecessary. Moreover, instances that are identified as critical through adversarial analysis are usually concentrated near the classifier's decision boundary. Relying exclusively on these boundary-adjacent samples may lead to an under-representation of majority-class safe regions, thereby distorting the original data distribution. Prior studies have emphasized that effective preprocessing for imbalanced learning should reduce redundancy while preserving the intrinsic distribution characteristics of the data (Krawczyk, 2016). Clustering-based strategies provide a principled way to uncover substructures within a class and partition the feature space into regions of varying difficulty, which helps maintain the global data distribution (Appalaraju et al., 2020).

In this work, as shown in Fig. 1(1), we employ the K-means algorithm (Likas et al., 2003) due to its simplicity, efficiency, and especially its ability to explicitly control the number of clusters by specifying  $K$ . Specifically, clustering is applied to the set of non-defective instances  $S_{\text{maj}}$ , and the number of clusters is defined as

$$K = \lambda \cdot |S_{\text{min}}|, \quad (1)$$

where  $\lambda > 1$  is a scaling factor. This design choice is motivated by three considerations. First, Our goal is not to discover semantically optimal clusters, but to obtain a coarse partition of the majority class for representative instance selection. Second, setting  $\lambda > 1$  ensures that the number of representative non-defective instances exceeds that of defective instances, providing a sufficiently expressive candidate pool for subsequent importance evaluation. Third, since clustering is not the primary focus of this study, we avoid complex within-cluster resampling (Hoffman et al., 2001) strategies and retain only the centroid-nearest instance from each cluster.

This centroid-nearest selection strategy effectively preserves the global distribution of the non-defective instances while yielding a compact and representative subset (Li et al., 2022), denoted as  $S'_{\text{maj}}$ . This subset serves as the foundation for subsequent Adversarial Influence computation. Based on empirical evaluations reported in Section 5.2, we set  $\lambda$  to 3 in our experiments, as it provides a favorable trade-off between performance and stability.

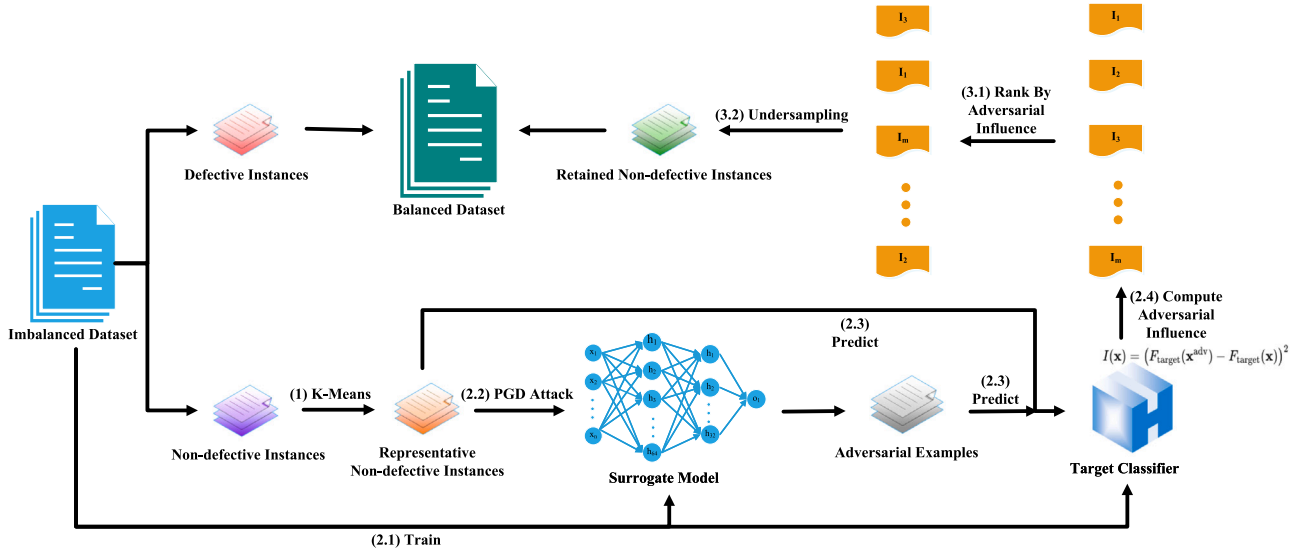


**Fig. 1.** The overall workflow of the proposed AIUS method, consisting of three stages: (1) K-Means clustering of non-defective instances to obtain representative candidates, (2) surrogate model training and PGD-based adversarial perturbation generation followed by Adversarial Influence computation via the target classifier's prediction change, and (3) influence-guided ranking and undersampling to construct a balanced training dataset.

### 3.2. Surrogate-based adversarial influence computation

To enable gradient-based adversarial perturbation generation for black-box target classifiers, we introduce a differentiable surrogate model  $F_{\text{surrogate}}$  to approximate the decision behavior of the target classifier  $F_{\text{target}}$ . In this work, a multi-layer perceptron (MLP) is adopted as the surrogate model, as its representational capacity is sufficient to capture the decision patterns of the target classifier given the moderate feature dimensionality and dataset scale typical of software defect prediction tasks. The surrogate model is trained on the original imbalanced dataset  $D$  by minimizing the binary cross-entropy loss (as shown in Fig. 1(2.1)).

Once trained, it enables efficient gradient-based adversarial attacks on the representative non-defective instances in  $S'_{\text{maj}}$ , (as shown in Fig. 1(2.2)). Specifically, for each  $x \in S'_{\text{maj}}$ , adversarial examples are generated using the Projected Gradient Descent (PGD) attack (Yang et al., 2023), a widely adopted iterative adversarial method. The PGD attack is parameterized by the perturbation bound  $\epsilon$ , step size  $\alpha$ , and the number of iterations  $T$ . Rather than manually tuning these hyperparameters, we employ the differential evolution (DE) algorithm to automatically identify effective configurations, using cross-validated Matthews correlation coefficient (MCC) as the optimization objective. Given a parameter configuration, PGD iteratively searches for a minimal perturbation  $\delta$  under an  $\ell_\infty$  constraint that maximally alters the surrogate model's prediction. Starting from  $x^0 = x$ , the update rule is defined as

$$x^{t+1} = \Pi_{|\delta|_\infty \leq \epsilon} \left( x^t + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(F_{\text{surrogate}}(x^t))) \right), \quad (2)$$

where  $\mathcal{L}(\cdot)$  denotes the surrogate model loss function and  $\Pi(\cdot)$  is the projection operator that enforces the perturbation constraint. After  $T$  iterations, the final adversarial example  $x^{\text{adv}}$  is obtained (Madry et al., 2018).

After generating adversarial examples for all representative non-defective instances, their Adversarial Influence is quantified using the prediction outputs of the target classifier (as shown in Fig. 1(2.3) and (2.4)). For a given instance  $x$  and its adversarial counterpart  $x^{\text{adv}}$ , the Adversarial Influence is defined as

$$I(x) = (F_{\text{target}}(x^{\text{adv}}) - F_{\text{target}}(x))^2. \quad (3)$$

This measure captures the sensitivity of the target classifier's prediction to adversarial perturbations in the local neighborhood of  $x$ , thereby reflecting the instance's influence on the decision boundary.

### 3.3. Influence-guided undersampling

Based on the computed Adversarial Influence scores, we perform influence-guided undersampling to construct a balanced and informative training dataset. The underlying intuition is that non-defective instances with higher Adversarial Influence values are more sensitive to small, adversarially induced perturbations, indicating that they lie closer to critical decision regions and is associated with stronger local prediction sensitivity on the classifier's decision boundary. Consequently, such instances are more informative for learning the discrimination between defective and non-defective classes and should be preferentially retained during undersampling.

Formally, all instances in  $S'_{\text{maj}}$  are ranked in descending order according to their Adversarial Influence scores (as shown in Fig. 1(3.1)). The top-ranked instances are then selected to form the retained majority-class subset  $\tilde{S}_{\text{maj}}$  (as shown in Fig. 1(3.2)), satisfying

$$|\tilde{S}_{\text{maj}}| = |S_{\text{min}}|. \quad (4)$$

Finally, the balanced training dataset is constructed as

$$D_{\text{balance}} = \tilde{S}_{\text{maj}} \cup S_{\text{min}}. \quad (5)$$

Through this selection strategy, the proposed method achieves class balance while explicitly preserving non-defective instances that are most critical to the target classifier's decision boundary, thereby improving the effectiveness of software defect prediction models.

To visually demonstrate the effectiveness of AIUS, we present PCA visualization using the NASA PC4 dataset with the KNN classifier as an example, as depicted in Fig. 6, illustrating the transition from the original imbalanced data to the final balanced dataset.

## 4. Experiments design

### 4.1. Datasets

We evaluate the proposed method on 18 widely used software defect prediction datasets collected from four public repositories: AEEEM (D'Ambros et al., 2012), NASA (Shepperd et al., 2013), PROMISE (Menzies et al., 2012), and SOFTLAB (Turhan et al., 2009). For the PROMISE repository, each project contains multiple versions. We adopt only the first version of each project (Gong et al., 2019). In addition, the NASA datasets include extra metrics such as parameter counts and comment

**Algorithm 1** AIIUS: Adversarial Influence-based Intelligent Undersampling

**Input:** Imbalanced dataset  $D$   
 Target classifier  $F_{\text{target}}$   
 Scaling factor  $\lambda$

**Output:** Balanced dataset  $D_{\text{balance}}$

- 1: Split  $D$  into  $S_{\text{maj}}$  and  $S_{\text{min}}$ ;
- 2:  $K \leftarrow \lambda \cdot |S_{\text{min}}|$ ;
- 3: Run K-means on  $S_{\text{maj}}$  with  $K$  clusters;
- 4: Select centroid-nearest instances to form  $S'_{\text{maj}}$ ;
- 5: Train surrogate model  $F_{\text{surrogate}}$  on  $D$ ;
- 6: Optimize PGD parameters  $(\epsilon, \alpha, T)$  via DE;
- 7: **for all**  $x \in S'_{\text{maj}}$  **do**
- 8:   Generate adversarial example  $x^{\text{adv}}$  via PGD on  $F_{\text{surrogate}}$ ;
- 9:   Compute  $I(x) \leftarrow (F_{\text{target}}(x^{\text{adv}}) - F_{\text{target}}(x))^2$ ;
- 10: **end for**
- 11: Rank  $S'_{\text{maj}}$  in descending order of  $I(x)$ ;
- 12: Select top  $|S_{\text{min}}|$  instances as  $\tilde{S}_{\text{maj}}$ ;
- 13:  $D_{\text{balance}} \leftarrow \tilde{S}_{\text{maj}} \cup S_{\text{min}}$ ;
- 14: **return**  $D_{\text{balance}}$ ;

**Table 3**

Statistics of 18 datasets from four repositories (AEEEM, NASA, PROMISE, SOFTLAB).

| Repository | Dataset     | Features | $N_{\text{total}}$ | $N_{\text{min}}$ | $N_{\text{maj}}$ | IR    |
|------------|-------------|----------|--------------------|------------------|------------------|-------|
| AEEEM      | JDT         | 61       | 997                | 206              | 791              | 3.84  |
| AEEEM      | LC          | 61       | 691                | 64               | 627              | 9.80  |
| AEEEM      | ML          | 61       | 1862               | 245              | 1617             | 6.60  |
| AEEEM      | PDE         | 61       | 1497               | 209              | 1288             | 6.16  |
| NASA       | CM1         | 37       | 327                | 42               | 285              | 6.79  |
| NASA       | MW1         | 37       | 250                | 25               | 225              | 9.00  |
| NASA       | PC1         | 37       | 679                | 55               | 624              | 11.35 |
| NASA       | PC3         | 37       | 1053               | 130              | 923              | 7.10  |
| NASA       | PC4         | 37       | 1270               | 176              | 1094             | 6.22  |
| PROMISE    | ant-1.3     | 20       | 125                | 20               | 105              | 5.25  |
| PROMISE    | camel-1.0   | 20       | 339                | 13               | 326              | 25.08 |
| PROMISE    | synapse-1.0 | 20       | 157                | 16               | 141              | 8.81  |
| PROMISE    | xalan-2.4   | 20       | 723                | 110              | 613              | 5.57  |
| SOFTLAB    | AR1         | 29       | 121                | 9                | 112              | 12.44 |
| SOFTLAB    | AR3         | 29       | 63                 | 8                | 55               | 6.88  |
| SOFTLAB    | AR4         | 29       | 107                | 20               | 87               | 4.35  |
| SOFTLAB    | AR5         | 29       | 36                 | 8                | 28               | 3.50  |
| SOFTLAB    | AR6         | 29       | 101                | 15               | 86               | 5.73  |

percentages. Since previous studies have reported the presence of mis-labeled instances in the original versions, we use the cleaned versions of the NASA datasets (Shepperd et al., 2013). All public datasets used in our experiments are fully annotated with ground-truth defect labels. We uniformly convert all annotations into the standard binary format  $y_i = 0$  (non-defective) and  $y_i = 1$  (defective) as defined in Table 2. The detailed statistics of all datasets are presented in Table 3, where the IR is the imbalance ratio, calculated by Eq. (6), all results are reported with two decimal places.

$$IR = \frac{N_{\text{maj}}}{N_{\text{min}}} \quad (6)$$

#### 4.2. Classifiers

In this study, the performance of the proposed AIIUS framework is evaluated using five established machine learning classifiers: K-Nearest Neighbors (KNN), Random Forest (RF), Decision Tree (Tree), Logistic Regression (LR), and Naive Bayes (NB). All classifiers are implemented with their default parameters via the scikit-learn library in Python (Pedregosa et al., 2011). This consistent use of out-of-the-box configurations streamlines the experimental process and facilitates the reproducibility and comparability of the results.

#### 4.3. Evaluative metrics

In software defect prediction (SDP) tasks, the class imbalance problem can easily lead to the distortion of traditional performance metrics. To objectively evaluate the performance of the proposed method, this study selects five key metrics that are robust to imbalanced data and aligned with the requirements of defect detection. All metrics are computed based on the confusion matrix, which defines four core outcomes: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The definitions and calculations of the metrics are as follows:

- **PD/Recall:**  $PD = \frac{TP}{TP+FN}$ . It represents the proportion of actual defective instances that are correctly identified by the model.
- **PF:**  $PF = \frac{FP}{TN+FP}$ . It denotes the proportion of actual non-defective instances that are incorrectly classified as defective by the model.
- **Balance:**  $Balance = 1 - \frac{\sqrt{(0-PF)^2 + (1-PD)^2}}{\sqrt{2}}$ . This metric comprehensively considers both PD and PF. Its value ranges between [0, 1], where a higher value indicates a better balance between the PD and PF.
- **AUC:** This metric evaluates the model's overall ability to distinguish between defective and non-defective modules across all possible classification thresholds. It is independent of the class distribution.
- **MCC (Yuan et al., 2023):** This is a balanced measure that yields a high score only if the prediction performs well across all four categories of the confusion matrix. It is particularly informative on imbalanced datasets.  $MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$ .

#### 4.4. Baselines

The effectiveness of AIIUS is demonstrated through comparisons with six baseline methods. These baselines are selected to represent major categories of approaches for handling class imbalance, including oversampling, undersampling and generative methods.

**UFIDSF (Zheng et al., 2025).** A recent undersampling framework that innovates through a double-side filter to cleanse noise from both majority and minority classes, and a feature importance-guided projection to select the most informative majority instances.

**SB-GAN (Akritidis et al., 2023).** A Semantic-Balanced Generative Adversarial Network specifically designed for imbalanced tabular data. Its key innovation lies in proactively identifying and filtering out borderline and noisy minority instances before training the generator.

**Cluster (Lin et al., 2017).** A representative clustering-based undersampling method that uses K-means to cluster the majority class into  $k$  groups (where  $k$  equals the minority count) and then retains only the centroids of these clusters.

**RUS (Liu et al., 2008).** RUS is an undersampling method that balances imbalanced datasets by randomly discarding majority class instances until the class sizes are equal.

**Borderline-SMOTE (Han et al., 2005).** An adaptive variant of SMOTE that focuses on generating synthetic instances only in the borderline region where minority instances are closer to the decision boundary.

**SMOTE (Chawla et al., 2002).** The seminal Synthetic Minority Oversampling Technique generates synthetic minority class instances by interpolating between existing minority instances and their nearest neighbors. It is included as the fundamental oversampling benchmark and is widely used as baseline data resampling techniques in SDP.

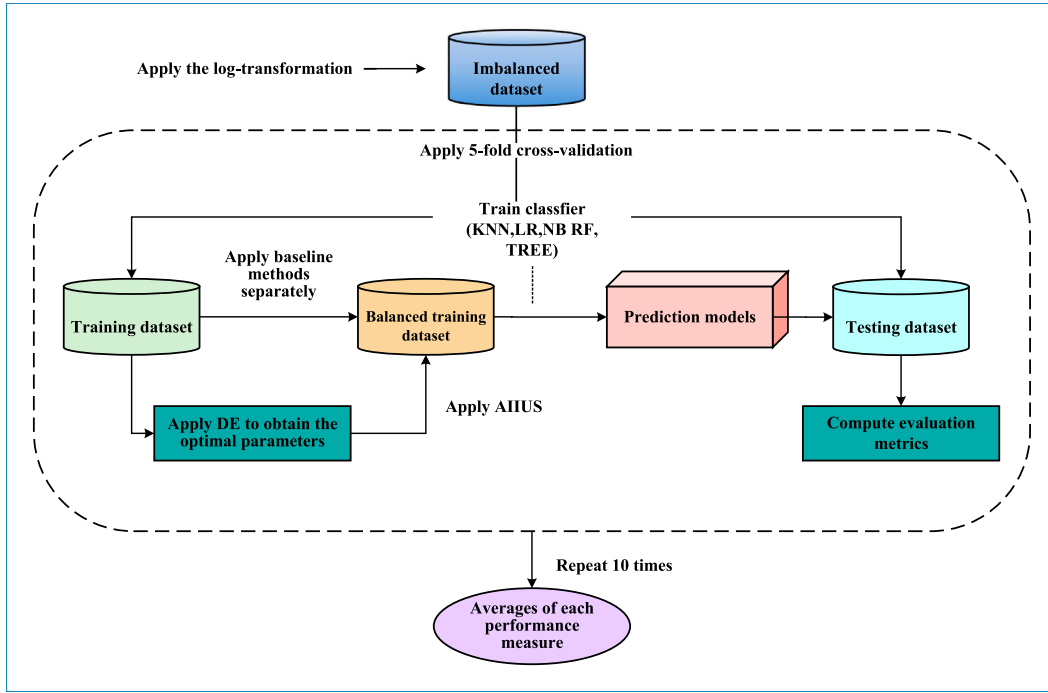


**Fig. 2.** Experimental procedure illustrating the evaluation pipeline: log-transformation is applied to all features, followed by stratified 5-fold cross-validation; baseline methods and AIIUS are independently applied to training folds to produce balanced datasets, which are used to train five classifiers (KNN, LR, NB, RF, TREE); performance is evaluated on test folds and averaged over 10 repetitions.

#### 4.5. Statistical test

To rigorously evaluate the performance differences between AIIUS and baseline methods, we adopt a comprehensive statistical analysis strategy, including hypothesis testing, effect size analysis, and win-draw-loss comparison.

The Wilcoxon signed-rank test (Rey and Neuhäuser, 2011) is employed to assess the statistical significance of pairwise performance differences between AIIUS and each baseline method. As a non-parametric test, it does not assume normality and is well suited to the performance distributions commonly observed in software defect prediction studies. The significance level is set to 0.05.

To quantify the magnitude of performance differences, Cliff's  $\delta$  (Macbeth et al., 2010) is used as the effect size measure. A positive value of  $\delta$  indicates superior performance of AIIUS, while a negative value favors the baseline method. Following the guidelines of Kampenes et al. (2007), the effect size is interpreted as negligible, small, moderate, or large.

In addition, a win-draw-loss analysis based on MCC is conducted to provide an intuitive comparison of AIIUS against baseline methods across datasets, reflecting its performance stability under diverse data scenarios.

#### 4.6. Experimental procedure

The experimental workflow, as illustrated in Fig. 2, begins with a logarithmic transformation applied to all numerical features in the imbalanced dataset to mitigate the effect of skewed distributions, without further feature normalization or standardization. Specifically, for each instance  $\mathbf{x}_i$ , each feature is transformed as

$$\mathbf{x}_i^{(j)} \leftarrow \log(\mathbf{x}_i^{(j)} + 1), \quad j = 1, \dots, d, \quad (7)$$

Subsequently, stratified 5-fold cross-validation is conducted to partition the dataset into training and testing sets while preserving the original class distribution. For baseline methods, each resampling technique is independently applied to the training data to construct balanced

**Table 4**

Hyperparameter configurations.

| Process      | Hyperparameter                             | Values        |
|--------------|--------------------------------------------|---------------|
| Differential | Population size                            | 10            |
| Evolution    | Maximum iterations                         | 8             |
| Algorithm    | Mutation factor                            | 0.3           |
|              | Recombination probability                  | 0.9           |
|              | Maximum perturbation magnitude, $\epsilon$ | [0.01, 0.3]   |
|              | Iteration step size, $\alpha$              | [0.001, 0.05] |
|              | Number of iterations, $N$                  | [5, 20]       |
| Surrogate    | Learning rate                              | 0.001         |
| Model        | Training epochs                            | 30            |
| Training     | Batch size                                 | 32            |

datasets. In contrast, for the proposed AIIUS method, differential evolution is performed on the training data to determine the optimal adversarial perturbation parameters. The resulting balanced training datasets are then used to train five classifiers, namely KNN, LR, NB, RF, and TREE. Model performance is evaluated on the testing set using five metrics: MCC, AUC, Balance, PD, and PF. To enhance statistical reliability, the entire procedure is repeated ten times with different random seeds, and the final results are reported as the average performance across all repetitions.

Hyperparameter settings for all classifiers are summarized in Table 4.

## 5. Experimental results

We present the experimental findings through a detailed analysis of the following research questions.

### 5.1. RQ1: How does AIIUS perform compared to baseline methods?

As mentioned in Section 4, we compare the proposed AIIUS method with two recently proposed approaches (UFIDSF and SB-GAN) and four widely used resampling techniques (Cluster, RUS, Borderline-SMOTE,

and SMOTE) across 18 software defect prediction datasets using five classifiers (KNN, LR, NB, RF, and TREE).

Table 5 reports the average PD, PF, and Balance obtained by seven sampling methods across five classifiers. W/D/L, short for Win/Draw/Loss, presents the number of datasets, on which AIIUS performs better than, the same as, or worse than another method, in terms of Balance. As shown in the table, AIIUS achieves the highest average Balance on four classifiers (KNN, LR, RF, and TREE), and ranks third on NB, with a marginal gap of only 0.010 from the best-performing method. These results indicate that AIIUS is effective in maintaining a favorable trade-off between defect detection capability and false alarm control. In contrast, although UFIDSF attains the highest PD on several classifiers, this improvement is accompanied by a substantial increase in PF (e.g., PD = 0.813 and PF = 0.435 on KNN), suggesting that its gains in detection capability come at the expense of significantly higher false alarm rates. Methods such as SB-GAN and Borderline-SMOTE tend to yield lower PF but also exhibit reduced PD, implying a higher risk of false negatives (FN). Furthermore, the W/D/L statistics show that AIIUS outperforms competing methods on the majority of datasets across different classifiers, suggesting that its performance advantages are consistent rather than driven by a small subset of datasets.

Fig. 3 further illustrates the distribution of balance values across datasets using box plots where the black triangle and black line denote the mean and median values, respectively. AIIUS generally exhibits higher central tendency and more stable distributions compared to most baselines.

Table 6 reports the AUC results of the seven sampling methods, including average AUC values, W/D/L statistics, statistical significance test p-values, and Cliff's  $\delta$  effect sizes. In terms of AUC, which reflects the overall discriminative capability of classifiers, AIIUS achieves the best performance on the RF and TREE, with relative improvements of 1.1% and 7.5% over the second-best baselines, respectively. On KNN and NB, the AUC gaps between AIIUS and the best-performing methods remain within 3.2%, indicating strong competitiveness. Although AIIUS does not achieve the highest AUC on LR, it still outperforms several traditional sampling techniques by a clear margin.

Tables 7, 8, 9, 10, and 11 present the detailed MCC results on 18 datasets using KNN, LR, NB, RF, and TREE classifiers, respectively, together with corresponding averages and statistical comparisons. With respect to this core evaluation metric, AIIUS exhibits more pronounced advantages. Specifically, it achieves the highest average MCC on KNN, NB, RF, and TREE, outperforming the corresponding second-best baselines by 8.6%, 3.8%, 19.4%, and 13.2%, respectively. These improvements are further supported by statistical significance tests ( $p < 0.05$ ). On LR, although AIIUS ranks slightly below the best-performing method with a relative gap of approximately 2.4%, it still surpasses the representative baseline UFIDSF by 14.6%. Moreover, the W/D/L statistics indicate that AIIUS consistently outperforms most competing methods on the majority of datasets, particularly under the RF classifier, highlighting its robustness and cross-dataset stability.

Figs. 4 and 5 present box plots of the AUC and MCC results across all datasets. As shown, AIIUS generally exhibits higher median values and more compact interquartile ranges than most baselines, particularly under RF and Tree, suggesting both strong performance and improved stability.

To conclude the above observation, AIIUS outperforms the competing methods in terms of balance, AUC, and MCC by 9.50%, 3.43%, and 17.02% on average.

## 5.2. RQ2: What is the impact of the clustering strategy on the effectiveness and stability of AIIUS?

To investigate the effect of clustering in AIIUS and to determine an appropriate clustering ratio, we evaluate its performance under four clustering ratios (1 $\times$ , 2 $\times$ , 3 $\times$ , and 3.5 $\times$  the number of defective instances) as well as a no-clustering setting. This design is motivated

**Table 5**

Average PD, PF, and Balance of seven methods across five classifiers with W/D/L statistics.

|      | Metric  | AIIUS        | UFIDSF       | SB-GAN       | Cluster      | RUS    | Border       | SMOTE        |
|------|---------|--------------|--------------|--------------|--------------|--------|--------------|--------------|
| KNN  | PD      | 0.783        | <b>0.813</b> | 0.598        | 0.627        | 0.737  | 0.622        | 0.678        |
|      | PF      | 0.305        | 0.435        | <b>0.230</b> | 0.265        | 0.334  | 0.247        | 0.281        |
|      | Balance | <b>0.688</b> | 0.633        | 0.646        | 0.651        | 0.673  | 0.653        | 0.676        |
|      | W/D/L   |              | 15/0/3       | 13/0/5       | 16/0/2       | 11/0/7 | 13/0/5       | 12/0/6       |
| LR   | PD      | 0.724        | <b>0.776</b> | 0.658        | 0.662        | 0.735  | 0.615        | 0.667        |
|      | PF      | 0.274        | 0.341        | 0.217        | 0.273        | 0.299  | <b>0.204</b> | 0.222        |
|      | Balance | <b>0.698</b> | 0.611        | 0.689        | 0.660        | 0.691  | 0.668        | 0.693        |
|      | W/D/L   |              | 15/0/3       | 11/0/7       | 13/0/5       | 8/0/10 | 12/0/6       | 7/0/11       |
| NB   | PD      | 0.641        | <b>0.765</b> | 0.637        | 0.655        | 0.657  | 0.611        | 0.653        |
|      | PF      | 0.259        | 0.327        | 0.251        | 0.306        | 0.282  | <b>0.249</b> | 0.259        |
|      | Balance | 0.650        | 0.616        | 0.651        | 0.652        | 0.646  | 0.644        | <b>0.660</b> |
|      | W/D/L   |              | 13/0/5       | 10/0/8       | 13/0/5       | 9/0/9  | 11/0/7       | 9/0/9        |
| RF   | PD      | <b>0.768</b> | 0.764        | 0.330        | <b>0.768</b> | 0.747  | 0.430        | 0.443        |
|      | PF      | 0.189        | 0.350        | 0.168        | 0.382        | 0.299  | <b>0.105</b> | 0.107        |
|      | Balance | <b>0.732</b> | 0.567        | 0.520        | 0.652        | 0.699  | 0.581        | 0.589        |
|      | W/D/L   |              | 18/0/0       | 18/0/0       | 15/0/3       | 17/0/1 | 17/0/1       | 17/0/1       |
| TREE | PD      | 0.699        | <b>0.723</b> | 0.372        | 0.671        | 0.670  | 0.406        | 0.407        |
|      | PF      | 0.321        | 0.371        | 0.229        | 0.427        | 0.346  | <b>0.153</b> | 0.157        |
|      | Balance | <b>0.662</b> | 0.555        | 0.542        | 0.574        | 0.636  | 0.560        | 0.560        |
|      | W/D/L   |              | 15/0/3       | 17/0/1       | 15/0/3       | 16/0/2 | 18/0/0       | 17/0/1       |

**Table 6**

Average AUC of seven methods across five classifiers with statistical test results.

|                  | AIIUS        | UFIDSF          | SB-GAN          | Cluster         | RUS             | Border          | SMOTE           |
|------------------|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| KNN              | 0.723        | 0.699           | 0.719           | <b>0.748</b>    | 0.702           | 0.687           | 0.699           |
| W/D/L            |              | 14/0/4          | 10/0/8          | 5/0/13          | 13/0/5          | 14/0/4          | 14/0/4          |
| p-value          |              | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.556           | 0.111           | -0.444          | 0.444           | 0.556           | 0.556           |
| LR               | 0.755        | 0.771           | <b>0.797</b>    | 0.772           | 0.718           | 0.705           | 0.722           |
| W/D/L            |              | 9/0/9           | 7/0/11          | 6/0/12          | 12/0/6          | 12/0/6          | 11/0/7          |
| p-value          |              | <b>&gt;0.05</b> | <b>&lt;0.05</b> | <b>&gt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.000           | -0.222          | -0.333          | 0.333           | 0.333           | 0.222           |
| NB               | 0.691        | 0.649           | 0.704           | <b>0.713</b>    | 0.688           | 0.687           | 0.699           |
| W/D/L            |              | 14/0/4          | 8/0/10          | 7/0/11          | 11/0/7          | 10/0/8          | 9/0/9           |
| p-value          |              | <b>&lt;0.05</b> | <b>&gt;0.05</b> | <b>&gt;0.05</b> | <b>&gt;0.05</b> | <b>&gt;0.05</b> | <b>&gt;0.05</b> |
| Cliff's $\delta$ |              | 0.556           | -0.111          | -0.222          | 0.222           | 0.111           | 0.000           |
| RF               | <b>0.760</b> | 0.752           | 0.739           | 0.747           | 0.725           | 0.669           | 0.673           |
| W/D/L            |              | 9/0/9           | 11/0/7          | 10/0/8          | 17/0/1          | 17/0/1          | 17/0/1          |
| p-value          |              | <b>&gt;0.05</b> | <b>&gt;0.05</b> | <b>&gt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.000           | 0.222           | 0.111           | 0.889           | 0.889           | 0.889           |
| TREE             | <b>0.689</b> | 0.641           | 0.629           | 0.615           | 0.662           | 0.635           | 0.633           |
| W/D/L            |              | 13/0/5          | 15/0/3          | 16/0/2          | 16/0/2          | 17/0/1          | 17/0/1          |
| p-value          |              | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.444           | 0.667           | 0.778           | 0.778           | 0.889           | 0.889           |

by the fact that the imbalance ratios of the studied datasets range from 3.5 to 25.08.

Fig. 7 illustrates the impact of different clustering ratios on AIIUS in terms of MCC across five classifiers. In this figure, the line plots depict the average MCC values, reflecting overall predictive effectiveness, while the bar charts represent the corresponding standard deviations, indicating performance stability across datasets.

From the perspective of average performance, moderate clustering generally leads to improved MCC for most classifiers. In particular, LR and RF exhibit a clear upward trend as the clustering ratio increases, with their performance peaking around the 3 $\times$  setting. This indicates that clustering the non-defective (majority) instances into a moderate number of subregions helps regularize their distribution, thereby enabling classifiers relying on global decision boundaries or ensemble mechanisms to more effectively exploit structured majority-class information. NB shows relatively stable average performance across different clustering ratios, with slight improvements under moderate clustering, suggesting limited but consistent benefits. For KNN and TREE, the impact of clustering on average MCC is more nuanced.

**Table 7**

The MCC values for AIIUS and baselines using the KNN classifier across 18 datasets.

| KNN              | AIIUS        | UFIDSF          | SB-GAN          | Cluster         | RUS             | Border          | SMOTE           |
|------------------|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| JDT              | 0.482        | 0.421           | 0.446           | 0.436           | 0.471           | 0.432           | 0.441           |
| LC               | 0.250        | 0.199           | 0.284           | 0.212           | 0.215           | 0.214           | 0.223           |
| ML               | 0.232        | 0.215           | 0.266           | 0.242           | 0.291           | 0.326           | 0.330           |
| PDE              | 0.233        | 0.220           | 0.218           | 0.219           | 0.240           | 0.218           | 0.189           |
| CM1              | 0.226        | 0.184           | 0.177           | 0.015           | 0.165           | 0.181           | 0.204           |
| MW1              | 0.241        | 0.239           | 0.218           | 0.255           | 0.242           | 0.219           | 0.211           |
| PC1              | 0.321        | 0.312           | 0.341           | 0.282           | 0.266           | 0.276           | 0.278           |
| PC3              | 0.327        | 0.299           | 0.301           | 0.343           | 0.303           | 0.322           | 0.335           |
| PC4              | 0.433        | 0.397           | 0.525           | 0.404           | 0.400           | 0.491           | 0.491           |
| ant-1.3          | 0.449        | 0.298           | 0.298           | 0.333           | 0.403           | 0.398           | 0.317           |
| camel-1.0        | 0.199        | 0.061           | 0.050           | 0.134           | 0.124           | -0.006          | 0.144           |
| synapse-1.0      | 0.260        | 0.282           | 0.239           | 0.302           | 0.267           | 0.291           | 0.234           |
| xalan-2.4        | 0.313        | 0.259           | 0.255           | 0.306           | 0.294           | 0.289           | 0.284           |
| AR1              | 0.238        | 0.135           | 0.153           | 0.180           | 0.089           | 0.108           | 0.194           |
| AR3              | 0.297        | 0.351           | 0.230           | 0.391           | 0.442           | 0.357           | 0.332           |
| AR4              | 0.328        | 0.236           | 0.398           | 0.225           | 0.210           | 0.207           | 0.236           |
| AR5              | 0.604        | 0.593           | 0.505           | 0.578           | 0.611           | 0.617           | 0.624           |
| AR6              | 0.253        | 0.189           | 0.168           | 0.105           | 0.220           | 0.184           | 0.171           |
| Average          | <b>0.316</b> | 0.272           | 0.282           | 0.276           | 0.292           | 0.285           | 0.291           |
| W/D/L            |              | 16/0/2          | 13/0/5          | 13/0/5          | 12/0/6          | 13/0/5          | 13/0/5          |
| p-value          |              | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.778           | 0.444           | 0.444           | 0.333           | 0.444           | 0.444           |

**Table 8**

The MCC values for AIIUS and baselines using the LR classifier across 18 datasets.

| LR               | AIIUS | UFIDSF          | SB-GAN | Cluster         | RUS   | Border | SMOTE        |
|------------------|-------|-----------------|--------|-----------------|-------|--------|--------------|
| JDT              | 0.483 | 0.420           | 0.482  | 0.370           | 0.474 | 0.466  | 0.485        |
| LC               | 0.233 | 0.176           | 0.239  | 0.146           | 0.255 | 0.262  | 0.254        |
| ML               | 0.328 | 0.334           | 0.378  | 0.243           | 0.359 | 0.361  | 0.364        |
| PDE              | 0.252 | 0.201           | 0.272  | 0.180           | 0.269 | 0.300  | 0.283        |
| CM1              | 0.321 | 0.257           | 0.272  | 0.243           | 0.251 | 0.293  | 0.297        |
| MW1              | 0.257 | 0.205           | 0.277  | 0.266           | 0.236 | 0.229  | 0.208        |
| PC1              | 0.339 | 0.326           | 0.344  | 0.333           | 0.288 | 0.280  | 0.283        |
| PC3              | 0.374 | 0.375           | 0.373  | 0.346           | 0.342 | 0.355  | 0.357        |
| PC4              | 0.523 | 0.426           | 0.573  | 0.527           | 0.532 | 0.575  | 0.579        |
| ant-1.3          | 0.278 | 0.228           | 0.350  | 0.294           | 0.335 | 0.311  | 0.358        |
| camel-1.0        | 0.117 | 0.102           | 0.102  | 0.126           | 0.155 | 0.036  | 0.164        |
| synapse-1.0      | 0.289 | 0.276           | 0.317  | 0.267           | 0.257 | 0.389  | 0.348        |
| xalan-2.4        | 0.337 | 0.322           | 0.331  | 0.317           | 0.318 | 0.312  | 0.314        |
| AR1              | 0.198 | 0.195           | 0.089  | 0.014           | 0.129 | 0.039  | 0.173        |
| AR3              | 0.225 | 0.206           | 0.235  | 0.212           | 0.267 | 0.368  | 0.313        |
| AR4              | 0.345 | 0.352           | 0.381  | 0.356           | 0.383 | 0.441  | 0.461        |
| AR5              | 0.586 | 0.550           | 0.591  | 0.567           | 0.613 | 0.603  | 0.610        |
| AR6              | 0.250 | 0.239           | 0.352  | 0.318           | 0.194 | 0.221  | 0.232        |
| Average          | 0.330 | 0.288           | 0.331  | 0.285           | 0.314 | 0.324  | <b>0.338</b> |
| W/D/L            |       | 15/0/3          | 6/0/12 | 12/0/6          | 9/0/9 | 9/0/9  | 7/0/11       |
| p-value          |       | <b>&lt;0.05</b> | >0.05  | <b>&lt;0.05</b> | >0.05 | >0.05  | >0.05        |
| Cliff's $\delta$ |       | 0.667           | -0.333 | 0.333           | 0.000 | 0.000  | -0.222       |

Their performance curves exhibit noticeable fluctuations across clustering ratios rather than a monotonic trend. For KNN, MCC increases as the clustering ratio grows and reaches its local maximum at the 3 $\times$  setting, followed by a slight decline at 3.5 $\times$ . Although the no-cluster setting yields the highest average MCC for KNN, it is accompanied by substantially larger variance, indicating unstable behavior.

When stability is taken into account, the advantage of moderate clustering becomes more evident. The no-cluster setting exhibits noticeably larger standard deviations for several classifiers, especially KNN and RF, indicating higher sensitivity to data partitioning and training sample variability. In comparison, clustering ratios around 3 $\times$  generally result in reduced or well-controlled variance across classifiers, suggesting that moderate clustering contributes to more stable and reliable learning behavior.

Considering both average performance and stability, the 3 $\times$  clustering ratio emerges as a well-balanced configuration. It achieves peak or near-peak MCC values for LR, RF, and NB, provides competitive

**Table 9**

The MCC values for AIIUS and baselines using the NB classifier across 18 datasets.

| NB               | AIIUS        | UFIDSF          | SB-GAN | Cluster         | RUS             | Border | SMOTE           |
|------------------|--------------|-----------------|--------|-----------------|-----------------|--------|-----------------|
| JDT              | 0.466        | 0.479           | 0.452  | 0.469           | 0.453           | 0.449  | 0.459           |
| LC               | 0.394        | 0.365           | 0.346  | 0.393           | 0.303           | 0.344  | 0.343           |
| ML               | 0.238        | 0.215           | 0.235  | 0.071           | 0.244           | 0.242  | 0.249           |
| PDE              | 0.321        | 0.304           | 0.311  | 0.304           | 0.294           | 0.303  | 0.299           |
| CM1              | 0.256        | 0.198           | 0.199  | 0.187           | 0.194           | 0.193  | 0.197           |
| MW1              | 0.215        | 0.243           | 0.264  | 0.223           | 0.243           | 0.277  | 0.237           |
| PC1              | 0.241        | 0.221           | 0.237  | 0.233           | 0.221           | 0.218  | 0.218           |
| PC3              | 0.320        | 0.299           | 0.296  | 0.278           | 0.285           | 0.281  | 0.282           |
| PC4              | 0.378        | 0.366           | 0.351  | 0.385           | 0.345           | 0.355  | 0.358           |
| ant-1.3          | 0.355        | 0.320           | 0.350  | 0.312           | 0.368           | 0.397  | 0.333           |
| camel-1.0        | 0.078        | 0.071           | 0.149  | 0.122           | 0.160           | 0.070  | 0.062           |
| synapse-1.0      | 0.163        | 0.095           | 0.232  | 0.171           | 0.211           | 0.315  | 0.281           |
| xalan-2.4        | 0.348        | 0.269           | 0.329  | 0.324           | 0.319           | 0.327  | 0.323           |
| AR1              | 0.073        | 0.067           | 0.064  | 0.140           | 0.073           | 0.018  | 0.038           |
| AR3              | 0.280        | 0.224           | 0.264  | 0.251           | 0.205           | 0.240  | 0.216           |
| AR4              | 0.334        | 0.308           | 0.385  | 0.402           | 0.375           | 0.368  | 0.385           |
| AR5              | 0.662        | 0.579           | 0.592  | 0.557           | 0.647           | 0.601  | 0.624           |
| AR6              | 0.269        | 0.257           | 0.218  | 0.250           | 0.160           | 0.243  | 0.278           |
| Average          | <b>0.299</b> | 0.271           | 0.293  | 0.282           | 0.275           | 0.283  | 0.288           |
| W/D/L            |              | 16/0/2          | 14/0/4 | 11/0/7          | 12/0/6          | 13/0/5 | 13/0/5          |
| p-value          |              | <b>&lt;0.05</b> | >0.05  | <b>&lt;0.05</b> | <b>&lt;0.05</b> | >0.05  | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.778           | 0.556  | 0.222           | 0.333           | 0.444  | 0.444           |

**Table 10**

The MCC values for AIIUS and baselines using the RF classifier across 18 datasets.

| RF               | AIIUS        | UFIDSF          | SB-GAN          | Cluster         | RUS             | Border          | SMOTE           |
|------------------|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| JDT              | 0.550        | 0.548           | 0.500           | 0.231           | 0.498           | 0.561           | 0.563           |
| LC               | 0.294        | 0.273           | 0.416           | 0.091           | 0.275           | 0.348           | 0.346           |
| ML               | 0.347        | 0.343           | 0.324           | 0.116           | 0.362           | 0.391           | 0.400           |
| PDE              | 0.313        | 0.280           | 0.306           | 0.075           | 0.287           | 0.316           | 0.318           |
| CM1              | 0.301        | 0.175           | -0.014          | 0.169           | 0.224           | 0.040           | 0.074           |
| MW1              | 0.290        | 0.225           | 0.349           | 0.152           | 0.190           | 0.368           | 0.326           |
| PC1              | 0.386        | 0.325           | 0.392           | 0.305           | 0.308           | 0.282           | 0.281           |
| PC3              | 0.392        | 0.356           | 0.266           | 0.351           | 0.354           | 0.310           | 0.299           |
| PC4              | 0.572        | 0.480           | 0.541           | 0.498           | 0.519           | 0.574           | 0.576           |
| ant-1.3          | 0.453        | 0.355           | 0.322           | 0.249           | 0.363           | 0.310           | 0.339           |
| camel-1.0        | 0.156        | 0.016           | -0.021          | 0.076           | 0.125           | 0.072           | 0.169           |
| synapse-1.0      | 0.339        | 0.229           | 0.311           | 0.293           | 0.261           | 0.248           | 0.235           |
| xalan-2.4        | 0.400        | 0.243           | 0.327           | 0.339           | 0.341           | 0.281           | 0.286           |
| AR1              | 0.300        | 0.163           | 0.113           | 0.175           | 0.213           | 0.243           | 0.283           |
| AR3              | 0.587        | 0.417           | 0.466           | 0.455           | 0.430           | 0.442           | 0.362           |
| AR4              | 0.408        | 0.385           | 0.459           | 0.422           | 0.367           | 0.278           | 0.318           |
| AR5              | 0.637        | 0.565           | 0.567           | 0.628           | 0.592           | 0.660           | 0.654           |
| AR6              | 0.241        | 0.209           | 0.197           | 0.225           | 0.141           | 0.253           | 0.230           |
| Average          | <b>0.387</b> | 0.310           | 0.323           | 0.269           | 0.325           | 0.335           | 0.337           |
| W/D/L            |              | 18/0/0          | 14/0/4          | 17/0/1          | 17/0/1          | 10/0/8          | 10/0/8          |
| p-value          |              | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 1.000           | 0.556           | 0.889           | 0.889           | 0.111           | 0.111           |

average performance for KNN and TREE, and simultaneously improves robustness by reducing performance variance. Therefore, from a combined effectiveness–stability perspective, selecting 3 $\times$  the minority class size as the default clustering configuration represents a practical and robust choice for AIIUS.

### 5.3. RQ3: Why was a standard MLP chosen as the surrogate model in the experiments?

First, it should be clarified that the primary and sole purpose of the MLP is to address the black-box constraint of the target classifiers (e.g., Random Forest, KNN, Decision Tree). Since these classifiers do not provide accessible gradients, we cannot directly apply gradient-based attacks like PGD (Projected Gradient Descent) to them. Therefore, we train a differentiable MLP surrogate to approximate the decision boundary of the target black-box model. This allows us to compute the necessary gradients in our manuscript to generate adversarial perturbations.

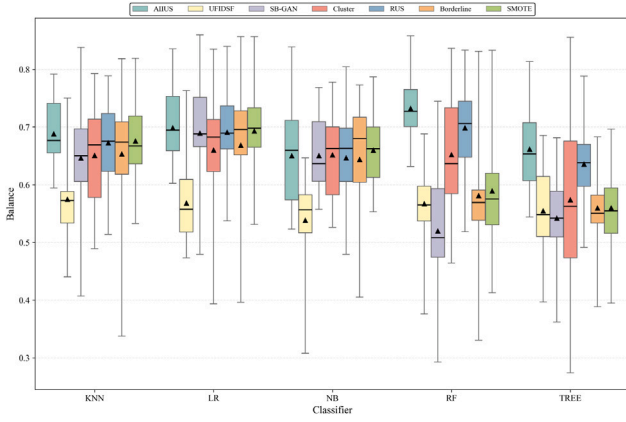


**Fig. 3.** Box-plots comparing the distribution of Balance values achieved by AIIUS and six baseline methods (UFIDSF, SB-GAN, Cluster, RUS, Borderline-SMOTE, SMOTE) across 18 software defect prediction datasets under five classifiers (KNN, LR, NB, RF, TREE). Black triangles and horizontal lines denote mean and median values, respectively.

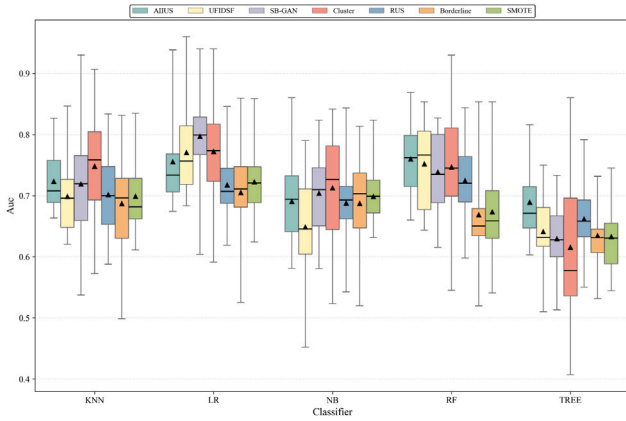


**Fig. 4.** Box-plots comparing the distribution of AUC values achieved by AIIUS and six baseline methods across 18 software defect prediction datasets under five classifiers. Black triangles and horizontal lines denote mean and median values, respectively. Higher median and more compact interquartile ranges indicate better and more stable discriminative performance.

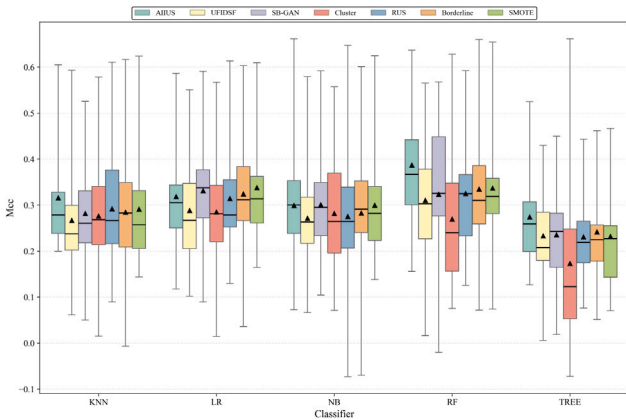


**Fig. 5.** Box-plots comparing the distribution of Matthews Correlation Coefficient (MCC) values achieved by AIIUS and six baseline methods across 18 software defect prediction datasets under five classifiers. AIIUS generally exhibits higher median values and more compact interquartile ranges, particularly under RF and TREE classifiers.

**Table 11**

The MCC values for AIIUS and baselines using the Tree classifier across 18 datasets.

| Tree             | AIIUS        | UFIDSF          | SB-GAN          | Cluster         | RUS             | Border          | SMOTE           |
|------------------|--------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| JDT              | 0.457        | 0.430           | 0.432           | 0.127           | 0.370           | 0.414           | 0.423           |
| LC               | 0.209        | 0.194           | 0.308           | 0.141           | 0.195           | 0.217           | 0.218           |
| ML               | 0.240        | 0.157           | 0.253           | 0.067           | 0.248           | 0.221           | 0.215           |
| PDE              | 0.198        | 0.192           | 0.183           | 0.048           | 0.188           | 0.182           | 0.182           |
| CM1              | 0.202        | 0.150           | 0.019           | 0.164           | 0.167           | 0.150           | 0.102           |
| MW1              | 0.127        | 0.177           | 0.082           | 0.012           | 0.126           | 0.169           | 0.135           |
| PC1              | 0.308        | 0.274           | 0.285           | 0.273           | 0.230           | 0.249           | 0.258           |
| PC3              | 0.303        | 0.252           | 0.276           | 0.247           | 0.271           | 0.241           | 0.271           |
| PC4              | 0.493        | 0.404           | 0.450           | 0.452           | 0.443           | 0.439           | 0.466           |
| ant-1.3          | 0.267        | 0.315           | 0.240           | 0.118           | 0.234           | 0.245           | 0.120           |
| camel-1.0        | 0.141        | 0.005           | 0.055           | -0.072          | 0.090           | 0.051           | 0.070           |
| synapse-1.0      | 0.277        | 0.189           | 0.163           | 0.248           | 0.173           | 0.225           | 0.166           |
| xalan-2.4        | 0.292        | 0.208           | 0.221           | 0.091           | 0.235           | 0.259           | 0.242           |
| AR1              | 0.128        | 0.154           | 0.154           | 0.027           | 0.180           | 0.177           | 0.261           |
| AR3              | 0.251        | 0.287           | 0.264           | 0.089           | 0.208           | 0.224           | 0.236           |
| AR4              | 0.334        | 0.260           | 0.442           | 0.404           | 0.301           | 0.273           | 0.245           |
| AR5              | 0.524        | 0.337           | 0.245           | 0.662           | 0.416           | 0.462           | 0.461           |
| AR6              | 0.181        | 0.208           | 0.169           | 0.021           | 0.076           | 0.155           | 0.131           |
| Average          | <b>0.274</b> | 0.233           | 0.236           | 0.173           | 0.231           | 0.242           | 0.232           |
| W/D/L            |              | 13/0/5          | 13/0/5          | 16/0/2          | 16/0/2          | 15/0/3          | 15/0/3          |
| p-value          |              | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | 0.444           | 0.444           | 0.778           | 0.778           | 0.667           | 0.667           |

**Table 12**

Average MCC of AIIUS under four surrogate architectures with statistical comparisons.

|                  | MLP          | Deep-MLP        | CNN             | ResNet          |
|------------------|--------------|-----------------|-----------------|-----------------|
| KNN              | <b>0.316</b> | 0.316           | 0.314           | 0.315           |
| p-value          |              | <b>&lt;0.05</b> | >0.05           | <b>&lt;0.05</b> |
| Cliff's $\delta$ |              | -0.056          | -0.056          | -0.056          |
| LR               | <b>0.330</b> | 0.298           | 0.316           | 0.311           |
| p-value          |              | <b>&lt;0.05</b> | <b>&lt;0.05</b> | >0.05           |
| Cliff's $\delta$ |              | 0.556           | 0.333           | 0.333           |
| NB               | 0.299        | <b>0.301</b>    | <b>0.301</b>    | 0.299           |
| p-value          |              | <b>&lt;0.05</b> | >0.05           | >0.05           |
| Cliff's $\delta$ |              | -0.056          | -0.222          | 0.111           |
| RF               | 0.387        | <b>0.390</b>    | 0.382           | 0.385           |
| p-value          |              | >0.05           | <b>&lt;0.05</b> | >0.05           |
| Cliff's $\delta$ |              | -0.111          | 0.333           | -0.111          |
| TREE             | 0.274        | 0.279           | <b>0.281</b>    | 0.275           |
| p-value          |              | >0.05           | >0.05           | >0.05           |
| Cliff's $\delta$ |              | -0.222          | -0.333          | -0.056          |

To justify the choice of surrogate architecture, we systematically evaluate four representative models, including a standard MLP, a deeper MLP, CNN, and ResNet, as reported in Table 12.

Across all classifiers, the differences in MCC among surrogate models are generally small, with most comparisons showing non-significant p-values and small Cliff's  $\delta$  effect sizes. More complex surrogates do not provide consistent or substantial improvements over the standard MLP, and in several cases, the MLP achieves the best or near-best MCC.

These results indicate that AIIUS is not sensitive to the specific surrogate architecture, provided that the surrogate can reasonably approximate the target decision function. Given the moderate dimensionality and scale of software defect prediction datasets, the representational capacity of a standard MLP is sufficient. Therefore, the MLP is adopted as a simple, effective, and computationally efficient surrogate model.

## 6. Discussion

### 6.1. The time consumption analysis of AIIUS

The AIIUS method consists of three main stages: representative non-defective instance selection, surrogate-based Adversarial Influence computation, and influence-guided undersampling.

In the first stage, the K-means algorithm is applied to partition  $|S_{\text{maj}}|$  instances into  $K = \lambda \cdot |S_{\text{min}}|$  ( $\lambda = 3$ ) clusters. The time complexity

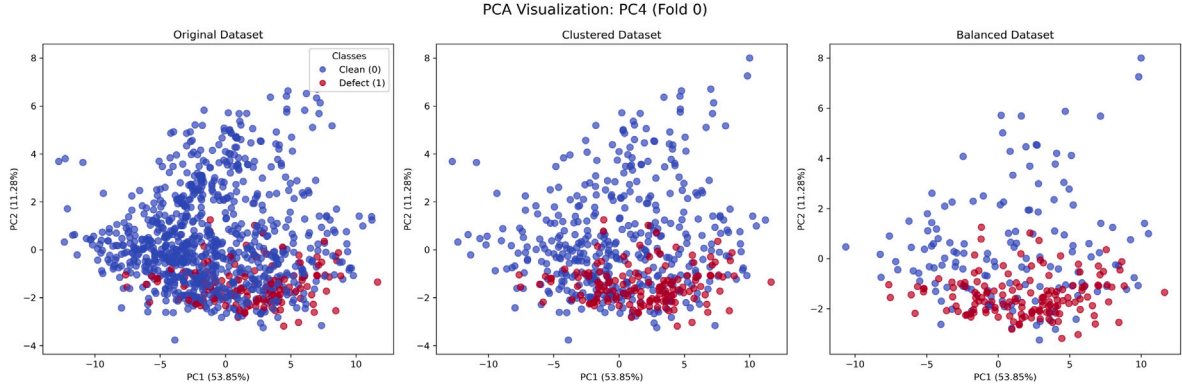


**Fig. 6.** PCA visualization of the NASA PC4 dataset (Fold 0) under the KNN classifier, showing the transition from the original imbalanced dataset (left), through the clustered representative non-defective instances (middle), to the final balanced dataset constructed by AIIUS via Adversarial Influence-guided retention (right). Green and red points denote non-defective and defective instances, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

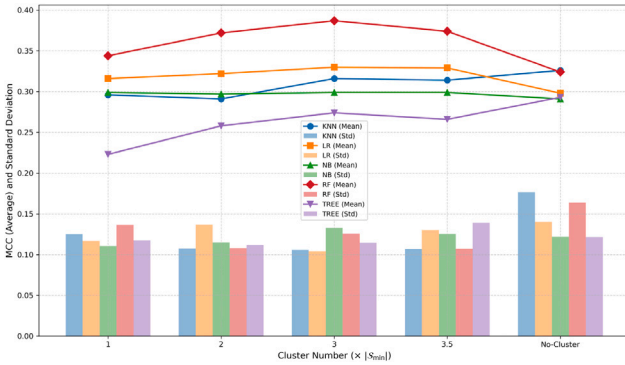


**Fig. 7.** Impact of clustering ratio (1 $\times$ , 2 $\times$ , 3 $\times$ , 3.5 $\times$  the number of defective instances, and no-cluster) on AIIUS performance: line plots show average MCC values (reflecting effectiveness) and bar charts show standard deviations (reflecting stability) across 18 datasets for five classifiers. The 3 $\times$  ratio provides the best trade-off between performance and stability for most classifiers.

of this step is  $O(|S_{\text{maj}}| \cdot K \cdot d \cdot I_{\text{km}})$ , where  $d$  denotes the feature dimension and  $I_{\text{km}}$  the number of K-means iterations. Although this step introduces computational cost, it significantly reduces the number of non-defective instances to be processed in subsequent stages, thereby effectively alleviating the computational burden of the core adversarial evaluation phase.

The second stage begins by training a multilayer perceptron (MLP) as a surrogate model, which has a time complexity of approximately  $O(E \cdot N \cdot d)$ , where  $E$  is the number of training epochs. Subsequently, PGD is employed to generate adversarial examples for each of the representative non-defective instances obtained from the first stage. The PGD attack performs  $T$  iterations per instance, each requiring gradient computation of the surrogate model, with constant cost per iteration. Hence, the complexity of this sub-step is  $O(K \cdot T)$ . Finally, the target classifier is invoked to obtain predictions for both the original and adversarial samples, thereby computing the Adversarial Influence score. This operation costs  $O(K \cdot C_{\text{target}})$ , where  $C_{\text{target}}$  represents the time cost of a single prediction by the target model. Overall, the second stage exhibits a complexity of  $O(E \cdot N \cdot d + K \cdot T + K \cdot C_{\text{target}})$ .

The third stage is dominated by the sorting operation, with a time complexity of  $O(K \log K)$ . Given that  $K$  and  $|S_{\text{min}}|$  are of the same order of magnitude, this overhead is generally negligible in practice.

The overall time complexity of AIIUS can be approximated as  $O((|S_{\text{maj}}| \cdot |S_{\text{min}}| + N) \cdot d)$ , given that the number of iterations in K-means  $I_{\text{km}}$ , surrogate training epochs  $E$ , and PGD steps  $T$  are relatively

small constants in practice. Among these, the first term (clustering) typically dominates. However, given that the feature dimension  $d$  in software defect prediction datasets is usually limited (between 20 and 61) and the scales of  $|S_{\text{maj}}|$  and  $|S_{\text{min}}|$  range from hundreds to thousands, this cost remains acceptable. More importantly, several computation-intensive components, such as surrogate training and PGD parameter tuning, can be performed offline or reused across experiments, further mitigating practical runtime concerns. Compared with oversampling techniques such as SMOTE, AIIUS avoids the generation of potentially noisy synthetic samples, and compared with conventional clustering-based undersampling, it introduces a measure of instance importance that explicitly accounts for decision boundary sensitivity. The demonstrated performance gains justify the additional computational investment.

## 6.2. Toward distributed surrogate training for AIIUS

The current implementation of AIIUS assumes that the surrogate model  $F_{\text{surrogate}}$  is trained on the centralized imbalanced dataset  $D$ , which may be impractical in cross-organizational or proprietary software development settings where raw data cannot be shared due to privacy regulations or business confidentiality. Fortunately, AIIUS itself does not inherently rely on this centralized training setup. The differentiable surrogate model serves solely to provide gradient information for PGD-based perturbation generation, while its specific training paradigm — whether centralized or distributed — does not affect the remaining stages of the undersampling pipeline. This modular design allows AIIUS to naturally accommodate various privacy-preserving architectures for surrogate training, thereby ensuring that the approach can operate within current data privacy requirements.

Among feasible privacy-preserving paradigms, federated learning (Yurdem et al., 2024) is the most lightweight and industry-friendly solution to train  $F_{\text{surrogate}}$  across disjoint software repositories. It offers a distributed training framework where each participating organization or repository trains a local MLP surrogate on its own non-defective instances, and only model updates (e.g., gradients or weights) are aggregated by a trusted central server using algorithms such as FedAvg (McMahan et al., 2017) to produce a unified global surrogate model. The aggregated global surrogate is then broadcast back to all local sites and serves as the differentiable proxy for the target black-box classifiers. After receiving the shared surrogate, every repository executes PGD adversarial perturbation, Adversarial Influence calculation and final undersampling entirely on local data, without any leakage of proprietary source code metrics.

For ultra-confidential scenarios such as financial, aerospace and embedded military software projects where even encrypted gradient

information cannot be transmitted externally, homomorphic encryption (Cheon et al., 2017) can be adopted as an alternative architecture. Under this scheme, all model training and parameter aggregation operations are conducted purely on ciphertexts, eliminating plaintext data exposure entirely. Nevertheless, this method introduces substantial computational latency, making it less suitable for general industrial pipelines compared to federated learning.

### 6.3. Generalizability to emerging high-stakes domains

The empirical validation of AIIUS is conducted on traditional software defect prediction benchmarks, which primarily represent open-source development scenarios. However, the applicability of AIIUS is not confined to these traditional open-source repositories.

Crucially, the core mechanism of AIIUS does not rely on any domain-specific semantic features. The Adversarial Influence metric, defined as  $I(\mathbf{x}) = (F_{\text{target}}(\mathbf{x}^{\text{adv}}) - F_{\text{target}}(\mathbf{x}))^2$ , operates solely on numerical feature vectors and query-able classifier outputs. This property makes the framework inherently domain-agnostic, provided that two conditions are satisfied: (1) instances can be represented as fixed-length numerical vectors, and (2) the target classifier provides probabilistic predictions accessible via queries.

This theoretical generality positions AIIUS for direct application to several emerging high-stakes domains where class imbalance and boundary ambiguity are even more pronounced than in traditional SDP.

**Smart contract vulnerability detection.** In Ethereum smart contracts, vulnerabilities such as reentrancy and access control flaws have caused substantial financial losses, with the DAO hack alone resulting in losses exceeding \$50 million (M et al., 2024). The class imbalance is extreme, with Ponzi scheme contracts constitute less than 5% of deployed contracts (Liu et al., 2026), and recent work has shown that boundary-aware sampling can effectively mitigate the resulting detection degradation (Wang et al., 2025). Since smart contract bytecode can be encoded as opcode frequency vectors or control-flow graph embeddings, AIIUS can be directly applied without algorithmic modification.

**IoT malware detection.** IoT devices face severe security threats from botnets and malware, yet their resource-constrained nature makes lightweight classifiers such as Random Forest and KNN — which are compatible with AIIUS's black-box probing mechanism — particularly attractive (Darwish and Roy, 2025). Studies have demonstrated that IoT malware datasets exhibit heavy class imbalance and significant feature-space overlap between benign and malicious binaries, posing challenges for boundary-aware classification (Jovanović and Vuletić, 2025). Given that IoT binaries are commonly represented via system call sequences or control-flow graph metrics, the numerical feature prerequisite of AIIUS is readily satisfied.

**Spacecraft diagnostic systems.** Notably, several datasets used in this study—including CM1 (spacecraft instrument software) and PC1/PC3/PC4 (satellite flight software)—originate from NASA spacecraft projects. In such mission-critical systems, the cost of false negatives far outweighs that of false positives, as software failures can lead to mission failure or loss of life. The severe class imbalance in these datasets further motivates influence-guided retention of boundary-defining instances.

### 6.4. Insights and limitations

This work offers paradigm-level insights into handling class imbalance in SDP. Rather than focusing solely on distribution-level resampling, the results indicate that preserving instances that are critical for maintaining discriminative decision boundaries can be more beneficial. By using a surrogate model to capture the decision behavior of the downstream classifier, AIIUS allows data preprocessing to be guided by how the model actually makes decisions, enabling instance selection to better support the learning process. Moreover, by leveraging the

transferability of adversarial examples, the proposed approach enables approximate boundary probing in black-box settings, which improves its applicability to practical SDP scenarios.

This work provides several paradigm-level insights for software defect prediction. It suggests that handling class imbalance in SDP can benefit from moving beyond distribution-level resampling toward preserving instances that are critical for maintaining discriminative decision boundaries. By leveraging a surrogate model and adversarial perturbations, AIIUS explicitly feeds back the decision logic of the downstream classifier into the preprocessing stage, thereby providing a feasible pathway for constructing a model-aligned intelligent data management mechanism. Moreover, the use of adversarial transfer enables boundary probing even in black-box settings, enhancing the practicality of the approach for real-world SDP systems.

Beyond SDP, the underlying principle of evaluating instance importance through model response sensitivity is not task-specific and can be extended to other imbalanced tabular learning problems characterized by majority redundancy and boundary ambiguity, such as fault diagnosis or risk prediction. Nevertheless, this work has limitations. The effectiveness of Adversarial Influence depends on the surrogate model's ability to approximate the target decision boundary, and substantial mismatch may affect instance ranking quality. More targeted surrogate architectures may further improve performance when focusing on specific classifiers. In addition, adversarial evaluation introduces extra computational cost, which may limit scalability in large-scale or time-sensitive scenarios.

## 7. Threats to validity

Like any empirical study, this work is subject to several potential threats to validity. We discuss these threats from the perspectives of internal validity, external validity, construct validity, and conclusion validity, and describe the measures taken to mitigate them.

**Internal validity.** Internal validity concerns whether the observed experimental results can be attributed to the proposed method rather than to uncontrolled factors in the experimental process. In this study, a primary internal validity threat arises from the use of a differentiable surrogate model to approximate the decision boundary of the target black-box classifier. If the surrogate model fails to adequately capture key decision patterns of the target model, the resulting Adversarial Influence may not accurately reflect the true importance of instances. Moreover, since adversarial perturbations are generated using the surrogate rather than the target classifier, the effectiveness of Adversarial Influence relies on the transferability of adversarial examples. To mitigate this threat, we empirically evaluate multiple surrogate model architectures and demonstrate that AIIUS is not sensitive to the specific surrogate choice, provided that the surrogate reasonably approximates the target decision function.

In addition to surrogate modeling, internal validity may also be influenced by the clustering strategy adopted to select representative non-defective instances. Although we evaluate multiple clustering ratios and adopt a balanced setting ( $\lambda = 3$ ), alternative clustering strategies may lead to different representative subsets. In this work, we employ the K-means algorithm due to its simplicity and stability, while acknowledging that exploring more advanced clustering techniques remains an avenue for future research.

Another internal validity threat arises from randomness introduced by clustering initialization and model training. This threat is mitigated by adopting stratified five-fold cross-validation to preserve the class distribution in each fold and repeating the entire experimental process ten times with different random initializations. Performance metrics are averaged over all runs, providing a more stable estimation of model performance.

**External validity.** External validity concerns the generalizability of the experimental findings to other software defect prediction scenarios. A potential threat is that the evaluation is conducted on a finite

set of publicly available datasets and a limited number of learning algorithms, which may not fully capture the diversity of real-world software development environments. To mitigate this threat, we conduct experiments on 18 real-world SDP datasets collected from four widely used and authoritative repositories, covering projects with diverse sizes, domains, and defect distributions. In addition, AIIUS is evaluated using five widely adopted classifiers with fundamentally different learning mechanisms, including instance-based, probabilistic, linear, and ensemble models. The consistent performance improvements observed across datasets and classifiers suggest that AIIUS exhibits good generalizability in practical SDP settings. Nevertheless, datasets or classifiers with characteristics substantially different from those considered in this study may lead to different empirical outcomes.

**Construct validity.** Construct validity concerns whether the selected measurements adequately reflect the intended research concepts. We evaluate defect prediction performance under class imbalance using multiple widely adopted metrics, including MCC, AUC, Balance, PD, and PF, which capture complementary aspects of predictive performance and are well established in the SDP literature.

**Conclusion validity** Conclusion validity concerns the reliability of the statistical inferences drawn from the experimental results. To reduce the impact of random variation, we apply the Wilcoxon signed-rank test for statistical significance and use Cliff's  $\delta$  to assess the magnitude of performance differences.

## 8. Conclusion and future work

This paper introduced AIIUS, a decision-boundary-aware undersampling method for imbalanced software defect prediction. Unlike conventional undersampling that removes non-defective instances using random or distribution-level criteria, AIIUS explicitly prioritizes non-defective instances that are most influential to the target classifier, quantified through an Adversarial Influence score derived from the target model's prediction change under small surrogate-generated perturbations. By combining (i) clustering-based candidate selection to preserve the non-defective instances distribution and reduce redundancy, with (ii) influence-guided retention to preserve boundary-defining instances, AIIUS constructs balanced training sets that better support defect discrimination.

Across 18 datasets and five classifiers, AIIUS achieved consistent improvements on imbalance-robust measures, yielding average gains of 9.50% (Balance), 3.43% (AUC), and 17.02% (MCC) over representative baselines, with statistical testing indicating significant advantages in most settings—particularly for MCC and Balance. Additional analyses showed that moderate clustering (3× the defective size) improves stability, and that a standard MLP surrogate is sufficient in this context.

Beyond empirical performance, this work offers a paradigm-level insight for handling class imbalance in SDP by shifting the focus from quantity-based resampling toward boundary-aware instance preservation. The proposed decision-boundary-aware resampling strategy enables data preprocessing to explicitly account for classifier behavior, including black-box settings, thereby enhancing practical applicability. Although AIIUS introduces additional computational cost compared to simple random undersampling, its design balances efficiency and effectiveness for typical SDP data scales.

Future work will focus on (1) substituting the embedded DE with mature Bayesian/TPE optimizers from Optuna and Auto-sklearn to boost search efficiency on ultra-large-scale software datasets. This upgrade retains the complete adversarial influence-based undersampling logic while absorbing the high-convergence advantage of AutoML tools for large search budgets, strengthening the scalability of our method for industrial big code repositories, (2) implementing distributed surrogate training by integrating federated learning or homomorphic encryption into the AIIUS framework, as discussed in Section 6.3, to enable practical deployment in cross-organizational and distributed software teams without exposing proprietary metrics, (3) incorporating feature

selection as a preprocessing step prior to clustering to reduce feature dimensionality and suppress noise, thereby further enhancing the performance of AIIUS in specific scenarios, and (4) reducing computational overhead via more query-efficient influence estimation and stronger theoretical characterization of the relationship between adversarial influence, boundary geometry, and generalization under imbalance.

## CRedit authorship contribution statement

**Shuo Feng:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Conceptualization. **Rongping Li:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jacky Keung:** Writing – review & editing. **Xiao Yu:** Writing – review & editing. **Yucheng Shi:** Writing – review & editing, Supervision, Conceptualization. **Mingliang Xu:** Supervision, Project administration.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

All datasets are public benchmark datasets cited in references. AIIUS source code will be released on GitHub after acceptance.

## References

- Abedin, M.Z., Guotai, C., Hajek, P., Zhang, T., 2023. Combining weighted smote with ensemble learning for the class-imbalanced prediction of small business credit risk. *Complex & Intell. Syst.* 9, 3559–3579.
- Akritis, L., Bozaris, P., 2024. A clustering-based resampling technique with cluster structure analysis for software defect detection in imbalanced datasets. *Inform. Sci.* 674, 120724.
- Akritis, L., Fevgas, A., Alamaniotis, M., Bozaris, P., 2023. Conditional data synthesis with deep generative models for imbalanced dataset oversampling. In: 2023 IEEE 35th International Conference on Tools with Artificial Intelligence. ICTAI, IEEE, pp. 444–451.
- Appalaraju, S., Zhu, Y., Xie, Y., Fehérvári, I., 2020. Towards good practices in self-supervised representation learning. *arXiv preprint arXiv:2012.00868*.
- Bai, J., Jia, J., Capretz, L.F., 2022. A three-stage transfer learning framework for multi-source cross-project software defect prediction. *Inf. Softw. Technol.* 150, 106985.
- Bennin, K.E., Keung, J., Phannachitta, P., Monden, A., Mensah, S., 2018. Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Trans. Softw. Eng.* 44, 534–550.
- Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. *J. Artificial Intelligence Res.* 16, 321–357.
- Cheon, J.H., Kim, A., Kim, M., Song, Y., 2017. Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (Eds.), *Advances in Cryptology – ASIACRYPT 2017*. Springer International Publishing, Cham, pp. 409–437.
- Chouhan, S.S., Rathore, S.S., 2021. Generative adversarial networks-based imbalance learning in software aging-related bug prediction. *IEEE Trans. Reliab.* 70, 626–642.
- D'Ambros, M., Lanza, M., Robbes, R., 2012. Evaluating defect prediction approaches: a benchmark and an extensive comparison. *Empir. Softw. Eng.* 17, 531–577.
- Darwish, R., Roy, K., 2025. Comparative analysis of federated learning, deep learning, and traditional machine learning techniques for iot malware detection. In: 2025 IEEE 4th International Conference on AI in Cybersecurity. ICAIC, pp. 1–10. <http://dx.doi.org/10.1109/ICAIC63015.2025.10849203>.
- Devarajulu, V.S., Lindquist, A.W., Addula, S.R., Kanipakam, S., 2026. NASA simupy-based spacecraft fault reliability diagnostics framework using machine learning. In: 2026 IEEE Aerospace Conference. pp. 1–12. <http://dx.doi.org/10.1109/AERO66936.2026.11519869>.
- Dong, Y., Lei, J., Long, S., Li, S., Zhang, J., 2026. A comprehensive survey of adversarial defense techniques in the visual domain. *Neurocomputing* 660, 131799.
- Eivazpour, Z., Keyvanpour, M.R., 2019. Adversarial samples for improving performance of software defect prediction models. In: The 7th International Conference on Contemporary Issues in Data Science. Springer, pp. 299–310.

- Feng, S., Keung, J., Xiao, Y., Zhang, P., Yu, X., Cao, X., 2024. Improving the undersampling technique by optimizing the termination condition for software defect prediction. *Expert Syst. Appl.* 235, 121084.
- Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K.E., Kabir, M.A., Zhang, M., 2021a. Coste: Complexity-based oversampling technique to alleviate the class imbalance problem in software defect prediction. *Inf. Softw. Technol.* 129, 106432.
- Feng, S., Keung, J., Yu, X., Xiao, Y., Zhang, M., 2021b. Investigation on the stability of smote-based oversampling techniques in software defect prediction. *Inf. Softw. Technol.* 139, 106662.
- García, V., Sánchez, J.S., Mollineda, R.A., 2012. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowl.-Based Syst.* 25, 13–21.
- Gong, L., Jiang, S., Wang, R., Jiang, L., 2019. Empirical evaluation of the impact of class overlap on software defect prediction. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 698–709.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2020. Generative adversarial networks. *Commun. ACM* 63, 139–144.
- Goodfellow, I.J., Shlens, J., Szegedy, C., 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Goyal, S., 2022. Handling class-imbalance with knn (neighbourhood) under-sampling for software defect prediction. *Artif. Intell. Rev.* 55, 2023–2064.
- Guido, R., Groccia, M.C., Conforti, D., 2023. A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers. *Soft Comput.* 27, 12863–12881.
- Gupta, N., Jindal, V., Bedi, P., 2022. Cse-ids: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems. *Comput. Secur.* 112, 102499.
- Han, H., Wang, W.Y., Mao, B.H., 2005. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: *International Conference on Intelligent Computing*. Springer, pp. 878–887.
- He, H., Bai, Y., Garcia, E.A., Li, S., 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). IEEE, pp. 1322–1328.
- He, K., Kim, D.D., Asghar, M.R., 2023. Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Commun. Surv. & Tutorials* 25, 538–566.
- Hoffman, E.B., Sen, P.K., Weinberg, C.R., 2001. Within-cluster resampling. *Biometrika* 88, 1121–1134.
- Hossen, M.A., Islam, M.S., Yusof, N.A.T., Rahman, M.S., Siddika, F., Rahman, M., Khatun, S., Karim, M.S.A., Mahmud, S.H., 2020. Hybrid sampling and random forest based machine learning approach for software defect prediction. In: *InECCE2019: Proceedings of the 5th International Conference on Electrical, Control & Computer Engineering*, Kuantan, Pahang, Malaysia, 29th July 2019. Springer, pp. 541–553.
- Japkowicz, N., Stephen, S., 2002. The class imbalance problem: A systematic study. *Intell. Data Anal.* 6, 429–449.
- Jovanović, D.D., Vuletić, P.V., 2025. Machine learning pipelines for iot botnet detection and behavior characterization in heavily imbalanced settings. *Signal, Image Video Process.* 19, 254.
- Kampenes, V.B., Dybå, T., Hannay, J.E., Sjøberg, D.I., 2007. A systematic review of effect size in software engineering experiments. *Inf. Softw. Technol.* 49, 1073–1086.
- Kaya, O., Shah, A.F.M.S., Karabulut, M.A., Karahan, S.N., Osmanca, M.S., Acir, N., 2026. Explainable artificial intelligence (XAI): concepts, applications, challenges, and future perspectives. *IEEE Access* 14, 27394–27417. <http://dx.doi.org/10.1109/ACCESS.2026.3663161>.
- Khan, A.A., Chaudhari, O., Chandra, R., 2024. A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation. *Expert Syst. Appl.* 244, 122778.
- Krawczyk, B., 2016. Learning from imbalanced data: open challenges and future directions. *Prog. Artif. Intell.* 5, 221–232. <http://dx.doi.org/10.1007/s13748-016-0094-0>.
- Kumar, V.K., Sagar, P.V., 2025. An optimal feature selection based hybrid intelligent model for software defect prediction. *Knowl.-Based Syst.* 114146. <http://dx.doi.org/10.1016/j.knsys.2025.114146>.
- Li, L., Lan, L.Y.L., Huang, L., Ye, C., Andrade, J., Wilson, P.C., 2022. Selecting representative samples from complex biological datasets using k-medoids clustering. *Front. Genet.* 13, 954024. <http://dx.doi.org/10.3389/fgene.2022.954024>.
- Likas, A., Vlassis, N., Verbeek, J.J., 2003. The global k-means clustering algorithm. *Pattern Recognit.* 36, 451–461.
- Lin, C.Y., Lai, B.H., Ng, H.F., Lin, W.Y., Chang, M.C., 2025. Robust defense against adversarial attacks with defensive preprocessing and adversarial training. In: 2025 IEEE International Conference on Consumer Electronics. ICCE, IEEE, pp. 1–6.
- Lin, W.C., Tsai, C.F., Hu, Y.H., Jhang, J.S., 2017. Clustering-based undersampling in class-imbalanced data. *Inform. Sci.* 409–410, 17–26.
- Liu, Y., Chen, Y., Wang, X., Yang, Y., Tan, C., 2026. Ponzi scheme detection for smart contracts based on oversampling. *Comput. Mater. Contin.* 86, 1–21. <http://dx.doi.org/10.32604/cmc.2025.069152>, URL: <http://www.techscience.com/cmc/v86n1/64450>.
- Liu, X.Y., Wu, J., Zhou, Z.H., 2008. Exploratory undersampling for class-imbalance learning. *IEEE Trans. Syst. Man Cybern. B* 39, 539–550.
- Lu, Y., Liu, J., Zhang, R., 2025. An adversarial perturbation generation method for image anti-forensics based on dual-path spatial attention gan. In: *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing*. ICASSP, IEEE, pp. 1–5.
- M, A., R, S., G, S., 2024. Optimizing smart contract security: A cost-sensitive graph neural network approach for vulnerability detection. In: 2024 International Conference on Electronic Systems and Intelligent Computing. ICESIC, pp. 191–195. <http://dx.doi.org/10.1109/ICESIC61777.2024.10846442>.
- Macbeth, G., Razumiejczyk, E., Ledesma, R.D., 2010. Cliff's Delta calculator: A non-parametric effect size program for two groups of observations. *Univ. Psychol.* 10, 545–555.
- Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A., 2018. Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, URL: <https://openreview.net/forum?id=rJzIBfZAb>.
- McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.y., 2017. Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (Eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1273–1282, URL: <https://proceedings.mlr.press/v54/mcmahan17a.html>.
- Menzies, T., Caglayan, B., Kocaguneli, E., Krall, J., Peters, F., Turhan, B., 2012. The promise repository of empirical software engineering data.
- Nikpour, B., Rahmati, F., Mirzaei, B., Nezamabadi-pour, H., 2026. A comprehensive review on data-level methods for imbalanced data classification. *Expert Syst. Appl.* 295, 128920.
- Papernot, N., McDaniel, P., Goodfellow, I., 2016a. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.
- Papernot, N., McDaniel, P., Goodfellow, I., 2016b. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples [arxiv]. p. 13, arXiv.
- Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A., 2017. Practical black-box attacks against machine learning. In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. pp. 506–519.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830.
- Petrides, G., Verbeke, W., 2022. Cost-sensitive ensemble learning: a unifying framework. *Data Min. Knowl. Discov.* 36, 1–28.
- Qiao, L., Li, X., Umer, Q., Guo, P., 2020. Deep learning based software defect prediction. *Neurocomputing* 385, 100–110.
- Ran, Y., Zhang, A.X., Li, M., Tang, W., Wang, Y.G., 2025. Black-box adversarial attacks against image quality assessment models. *Expert Syst. Appl.* 260, 125415.
- Ren, C., Du, X., Xu, Y., Song, Q., Liu, Y., Tan, R., 2022. Vulnerability analysis, robustness verification, and mitigation strategy for machine learning-based power system stability assessment model under adversarial examples. *IEEE Trans. Smart Grid* 13, 1622–1632.
- Rey, D., Neuhauser, M., 2011. Wilcoxon-signed-rank test. In: *International Encyclopedia of Statistical Science*. Springer, Berlin Heidelberg, pp. 1658–1659.
- Shepperd, M., Song, Q., Sun, Z., Mair, C., 2013. Data quality: Some comments on the NASA software defect datasets. *IEEE Trans. Softw. Eng.* 39, 1208–1215.
- Tang, Y., Du, Y., Li, A., Yang, M.s., Xia, Y., 2025. An undersampling method for software defect prediction based on Hilbert curve mapping distance. *Eng. Appl. Artif. Intell.* 162, 112519.
- Tao, W., Long, S., Liu, X., Li, W., Tao, Q., 2025. Optimizing the adversarial perturbation with a momentum-based adaptive matrix. *IEEE Trans. Dependable Secur. Comput.* <http://dx.doi.org/10.1109/TDSC.2025.3646013>.
- Tian, X., Du, Y., 2026. Fmdtde: A novel hybrid sampling framework for addressing class imbalance in software defect prediction. *Inf. Softw. Technol.* 190, 107945.
- Turhan, B., Menzies, T., Bener, A.B., Di Stefano, J., 2009. On the relative value of cross-company and within-company data for defect prediction. *Empir. Softw. Eng.* 14, 540–578.
- Vashisht, R., Rizvi, S.A.M., 2022. Class imbalance learning to heterogeneous cross-software projects defect prediction. *Int. J. Softw. Innov. (IJSI)* 10, 1–18.
- Wang, L., Cheng, H., Sun, Z., Tian, A., Yang, Z., 2025. Pspl: A ponzi scheme smart contracts detection approach via compressed sensing oversampling-based peephole lstm. *Future Gener. Comput. Syst.* 166, 107655. <http://dx.doi.org/10.1016/j.future.2024.107655>, URL: <https://www.sciencedirect.com/science/article/pii/S0167739X24006198>.
- Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L., 2019. Improving transferability of adversarial examples with input diversity. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 2725–2734.
- Yang, Y., Lin, C., Ji, X., Tian, Q., Li, Q., Yang, H., Wang, Z., Shen, C., 2023. Towards deep learning models resistant to transfer-based adversarial attacks via data-centric robust learning. *arXiv preprint arXiv:2310.09891*.
- Yu, X., Rao, J., Liu, L., Lin, G., Hu, W., Keung, J.W., Zhou, J., Xiang, J., 2024. Improving effort-aware defect prediction by directly learning to rank software modules. *Inf. Softw. Technol.* 165, 107250.

- Yuan, X., Chen, S., Zhou, H., Sun, C., Yuwen, L., 2023. Chsmote: Convex hull-based synthetic minority oversampling technique for alleviating the class imbalance problem. *Inform. Sci.* 623, 324–341.
- Yurdem, B., Kuzlu, M., Gullu, M.K., Catak, F.O., Tabassum, M., 2024. Federated learning: Overview, strategies, applications, tools and future directions. *Heliyon* 10, e38137. <http://dx.doi.org/10.1016/j.heliyon.2024.e38137>, URL: <https://www.sciencedirect.com/science/article/pii/S2405844024141680>.
- Zhang, Y., Liu, N., Zhao, Y., Fan, J., Gong, L., 2026. Evaluating the interactions between class overlap and class imbalance for software defect prediction. *Expert Syst. Appl.* 296, 129067.
- Zhao, Y., Huang, W., Liu, W., Yao, X., 2025. Revealing vulnerable regions through diverse adversarial examples. *Mach. Learn.* 114, 163. <http://dx.doi.org/10.1007/s10994-025-06788-z>.
- Zhao, H., Zhao, C., Zhang, X., Liu, N., Zhu, H., Liu, Q., Xiong, H., 2023. An ensemble learning approach with gradient resampling for class-imbalance problems. *INFORMS J. Comput.* 35, 747–763.
- Zheng, M., Wang, F., Hu, X., Hu, L., Yu, Q., Zheng, X., 2025. Ufidsf: An undersampling approach based on feature importance and double side filter for imbalanced data classification. *Future Gener. Comput. Syst.* 167, 107750.