

Enhancing software fault prediction through data balancing techniques and machine learning

Akshat Raj, Durva Mahadeo Chavan, Priyal Agarwal, Jestin Gigi, Madhuri Rao, Vinayak Musale,
Akshita Chanchlani, Murtaza Shabbirbhai Dholkawala, Kulamala Vinod Kumar

Department of Computer Engineering and Technology, Dr. Vishwanath Karad MIT World Peace University, Pune, India

Article Info

Article history:

Received May 6, 2024

Revised Oct 27, 2025

Accepted Nov 8, 2025

Keywords:

Generative adversarial networks

Imbalanced data

NearMiss

SMOTE

Software fault prediction

ABSTRACT

Software fault prediction is essential for ensuring the reliability and quality of software systems by identifying potential defects early in the development lifecycle. However, the presence of imbalanced datasets poses a significant challenge to the effectiveness of fault prediction models. In this paper, we investigate the impact of different data balancing techniques, including generative adversarial networks (GANs), synthetic minority over-sampling technique (SMOTE), and NearMiss, on machine learning (ML) model performance for software fault prediction. Through a comparative analysis across multiple datasets commonly used in software engineering research, we evaluate the efficacy of these techniques in addressing class imbalance and improving predictive accuracy. Our findings provide insights into the most effective approaches for handling imbalanced data in software fault prediction tasks, thereby advancing the state-of-the-art in software engineering research and practice. An extensive experimentation is performed and analyzed in this study here that includes 8 datasets, 4 data balancing techniques, and 4 ML techniques in order to demonstrate the efficacy of various models in software fault prediction.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Madhuri Rao

Department of Computer Engineering and Technology, Dr. Vishwanath Karad MIT World Peace University
Kothrud, Pune, Maharashtra, India

Email: madhuri.rao@mitwpu.edu.in

1. INTRODUCTION

Software fault prediction aims at identifying impending faults or bugs in software modules before they are visible operational issues, thereby enhancing the quality of the built software [1], [2]. With the increasing complexity of software systems and the demand for reliable and efficient software, the importance of accurate fault prediction techniques cannot be overstated. Traditional methods of fault prediction often rely on machine learning (ML) models trained on historical data to classify software modules as defective or non-defective based on various code metrics and characteristics [3]–[6]. However, one of the significant challenges in software fault prediction is dealing with imbalanced datasets. Imbalanced datasets occur when the classes of interest (defective vs. non-defective modules) are not evenly distributed, leading to biased model performance [7], [8]. The research aim of this study is to investigate the effectiveness of different data balancing techniques in improving ML model performance for software fault prediction. Specifically, we will explore techniques such as generative adversarial networks (GANs) [9]–[11], synthetic minority over-sampling technique (SMOTE) [12], [13], and NearMiss [14] to address the imbalance in the dataset. By comparing the performance of ML models trained on balanced and unbalanced datasets, we seek to identify the most effective approach for handling imbalanced data in software fault prediction tasks. ML techniques could be classified as supervised, unsupervised and semi-supervised. ML techniques have

been very useful in detecting issues and bugs in software [15], [16]. However, the efficacy of ML techniques depends on the nature of datasets that are often linked to be imbalanced. Addressing issues of data imbalance is therefore highly essential [17] in every area where data distribution has a significance in decision-making.

2. LITERATURE REVIEW AND RESEARCH TARGETS

Software fault prediction is a critical aspect of software engineering, aiming to identify potential defects in software modules before they lead to operational issues. Addressing the challenges posed by imbalanced datasets is crucial for improving the accuracy and effectiveness of fault prediction models. In this literature survey, we review previous research in software fault prediction, with a focus on data balancing techniques such as GANs, SMOTE, and NearMiss. We also discuss the limitations of earlier approaches and how the proposed research aims to address them. In the realm of software fault prediction, addressing the challenge of imbalanced datasets is paramount for achieving accurate and reliable predictive models. Various data balancing techniques have been proposed and explored in the literature to mitigate the impact of class imbalance on model performance. In this section, we review previous research focusing on data balancing techniques such as GANs, SMOTE, and NearMiss, highlighting their strengths, limitations, and gaps, as well as discussing how the proposed research aims to address these shortcomings. Engelman and Lessmann [18] present conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning for oversampling imbalanced data in credit scoring. Alqarni and Aljamaan [19] propose a novel approach that combines GAN-based methods with boosting ensembles to improve software defect prediction performance. While the study offers a promising solution, it lacks specific validation on the quality of generated synthetic data and the scalability of the proposed approach. Our research aims to address these gaps by conducting rigorous performance evaluation and scalability testing of GAN-based oversampling techniques, thus providing empirical evidence of their effectiveness and practical feasibility.

Several studies have investigated the use of different data balancing techniques to improve software fault prediction models. For example, Kumar and Venkatesan [20] explored the use of GANs for addressing data imbalance in software defect prediction. The authors proposed a novel approach that leverages GANs to generate synthetic data samples, thereby balancing the distribution of defective and non-defective instances in the dataset. Their results showed promising improvements in model accuracy and performance compared to traditional techniques. Similarly, Feng *et al.* [21] focuses on using SMOTE, a popular oversampling technique, to address class imbalance in software fault prediction. The study compared the performance of ML models trained on balanced and unbalanced datasets, demonstrating the effectiveness of SMOTE in improving model performance. However, the study also highlighted limitations in the scalability and computational efficiency of SMOTE, indicating the need for further research in this area. In addition to oversampling techniques like SMOTE and GANs, under sampling methods such as NearMiss have also been explored in the context of software fault prediction. For example, Mqadi *et al.* [14] investigates the use of NearMiss for handling class imbalance in defect prediction datasets. The authors proposed a hybrid approach that combines under sampling with feature selection to improve the performance of ML models. Their experimental results showed promising improvements in model accuracy and generalization capabilities. Furthermore, in [22], a survey of software fault prediction techniques and recent developments is provided, which highlights the need to address class imbalance issues. While the survey offers valuable insights into various supervised ML techniques and sampling methods, it lacks in-depth exploration of emerging techniques beyond SMOTE. Our research aims to extend this survey by investigating the effectiveness of GANs and other advanced data balancing techniques in software fault prediction, thus enriching the existing literature with new insights and empirical evidence.

Our primary research target is to conduct extensive experiments to evaluate the performance of software defect prediction in scenarios where data is imbalanced. Despite the advancements in data balancing techniques for software fault prediction, several limitations exist in earlier approaches. One common limitation is the lack of comprehensive evaluation and comparison of different sampling methods across diverse datasets and ML algorithms. Many studies focus on a limited set of techniques or datasets, which may not fully capture the variability and complexity of real-world software development scenarios. Furthermore, existing research often overlooks the impact of data bias and model interpretability on the effectiveness of data balancing techniques. Imbalanced datasets may contain biased representations of certain classes, leading to skewed model predictions and erroneous conclusions. Moreover, the interpretability of ML models trained on balanced or synthetic data is often overlooked, making it challenging to understand the underlying factors driving model predictions. Here, we study the impact of data imbalance on 8 different datasets: JM1, AR1, CM1, KC2, MW1, PC1, MC2, and KC1. Our research aims to build upon the problems identified in literature review due to data imbalance and by evaluating the applicability of GAN-based oversampling across multiple software fault prediction datasets. We thus extend the scope beyond specific

domains and address the limitations of single-class focus. Our research seeks to bridge this gap by incorporating GAN-based oversampling into the comparative analysis, thus providing a more comprehensive evaluation of sampling techniques' efficacy in software fault prediction.

3. PROPOSED METHODOLOGY

To overcome the limitations of existing research, we adopted a systematic approach that includes thorough experimentation, rigorous evaluation metrics, and extensive validation across diverse software development scenarios exploring GAN, SMOTE, and NearMiss across multiple datasets and ML algorithms. Additionally, we have explored the use of advanced sampling techniques and ensemble methods to further enhance the model performance and reliability. Hence, the proposed research seeks to advance the field of software fault prediction by providing insights into the most effective data balancing techniques for improving model accuracy and reliability. By addressing the limitations of earlier approaches, we aim to contribute to the development of more robust and dependable fault prediction models, ultimately enhancing the quality and reliability of software systems. Figure 1 depicts the logical steps taken in the proposed model.

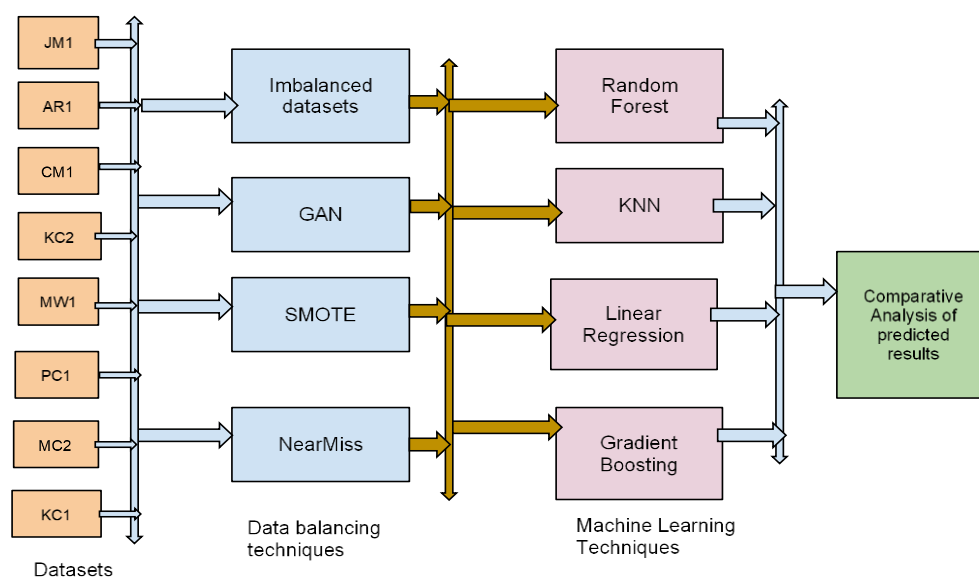


Figure 1. Logical steps of the proposed software fault prediction model

3.1. Data preprocessing

We begin by collecting software fault prediction datasets from reputable sources such as NASA's software engineering laboratory and the PROMISE software engineering repository. Data preprocessing is conducted to ensure data quality and consistency. This involves handling missing values, outliers, and inconsistencies in the datasets. We employ techniques such as mean imputation or deletion for missing values, outlier detection and removal using statistical methods or domain knowledge, and standardization or normalization to scale the features appropriately. Additionally, we perform exploratory data analysis (EDA) [18] to gain insights into the data distribution and characteristics, identifying potential patterns or trends that may aid in model development and interpretation. EDA also helps us understand the extent of class imbalance in the datasets, which is crucial for selecting appropriate data balancing techniques.

3.2. Data generation using generative adversarial network

Deep learning models such as GAN comprise two neural networks, the generator and the discriminator, that are engaged in a minimax game. The generator attempts to generate synthetic data that is almost like the real data, while the discriminator aims to identify the real and generated data. The generator aims to generate increasingly realistic samples by minimizing its own loss (G loss), which measures the discrepancy between the discriminator's predictions and a label indicating the generated data is real. Conversely, the discriminator seeks to correctly classify real and fake samples, thus minimizing its loss (D loss). G loss and D loss are fundamental components in training GANs, representing the objectives of the generator and discriminator, respectively, in achieving their competing goals. The generator loss is typically defined using binary cross-entropy, measuring the difference between the discriminator's predictions on

generated data and a matrix of ones (since the generator wants the discriminator to classify its outputs as real). Mathematically, it can be expressed as (1):

$$G_{loss} = -\frac{1}{N} \sum_{i=1}^N \log(D(G(z_i))) \quad (1)$$

Here, $G(z_i)$ is the output of the generator for the i -th input noise sample z_i and $D(G(z_i))$ is the discriminator's output when the probability of the generated sample $G(z_i)$ is real, and N is the batch size considered. The discriminator loss is computed using binary cross-entropy loss separately for real and fake data, and then averaged. Mathematically, it can be expressed as (2):

$$D_{loss} = -\frac{1}{2N} \sum_{i=1}^N \log \log(D(x_i)) + \log \log(1 - (D(G(z_i)))) \quad (2)$$

Here, x_i is the i th real data point and $D(x_i)$ the discriminator is output when x_i is real. $G(z_i)$ is the output of the generator for the i -th input noise sample z_i and $D(G(z_i))$. The discriminator's output when the probability of the generated sample $G(z_i)$ is real, and N is the batch size considered. Binary cross-entropy loss measures the difference between probability distributions, particularly useful for binary classification problems. It is defined as (3).

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N (1 - p(y_i)) \quad (3)$$

Where y_i is the true label (0 or 1) and (y_i) is the predicted probability of the positive class. The GAN objective function is essentially a minimax game between the generator and the discriminator. The generator tries to minimize the discriminator's ability to distinguish between real and fake data, while the discriminator tries to maximize it. Mathematically, it is given as (4).

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (4)$$

Here, $D(x)$ is the discriminator's output and $G(z_i)$ is the generator's output (fake data) and $p_{data}(x)$ and $p(z)$ are the data and noise distributions.

3.3. Data balancing using synthetic minority over-sampling technique

SMOTE is a widely-used data augmentation technique that produces unreal samples for the minority class by synthesizing between existing minority class samples. Specifically, SMOTE considers a minority class sample and its k nearest neighbors, then creates new synthetic samples along the line segments joining them. By effectively increasing the representation of the minority class, SMOTE helps to balance the class distribution in the datasets, thus mitigating the bias towards the majority class. This technique aims to improve the performance of ML models by providing them with more diverse and representative training data, thereby reducing the risk of model overfitting and improving predictive accuracy for the minority class.

3.4. Data balancing using NearMiss

Another data balancing technique we utilize is NearMiss, which is specifically designed to address class imbalance by under sampling the majority class. NearMiss selects a subset of majority class samples that are closest to the minority class samples in feature space, effectively reducing the imbalance ratio between the majority and minority classes. NearMiss offers three variants: NearMiss-1, NearMiss-2, and NearMiss-3, each employing different strategies to select the majority class samples for removal. NearMiss-1 considers samples from the majority class with the smallest average distance to the three nearest minority class samples. In NearMiss-2, samples from the majority class with the farthest average distance to the three nearest minority class samples are considered. NearMiss-3 on the other hand is a two-step process that first selects majority class samples using NearMiss-1 or NearMiss-2 and then further refines the selection based on the majority class samples that are misclassified by a k -nearest neighbor (KNN) classifier trained on the original dataset. By strategically removing majority class samples, NearMiss aims to enhance the balance between the classes and improve the performance of ML models on imbalanced datasets.

3.5. Machine learning algorithms

The implementation will utilize a variety of ML algorithms suitable for classification tasks, including but not limited to: random forests (RF) [23], KNN [24], logistic regression (LR) [25], [26], and gradient boosting (GB) [27] algorithms. In our analysis of various datasets across different data balancing techniques, we evaluated the performance of four commonly used machine-learning models: RF, LR, KNN, and GB. These models were chosen for their versatility and widespread applicability in classification tasks.

The first model employed was KNN, which is a non-parametric classification algorithm. Given a query point x , the algorithm finds the k nearest neighbors in the training set and assigns the majority class among these neighbors to x_q . Consider $N_k(x_q)$ to be the set of k nearest neighbors of x_q in the training set, then the predicted class label is for x_q is given as per (5):

$$\hat{y}_q = \arg \max_y \sum_{x_i \in N_k(x_q)} I(y_i = y) \quad (5)$$

Here $I(\cdot)$ is the indicator function and y_i is the class label of the i -th nearest neighbor and \hat{y}_q is the predicted class label for x_q . LR is a linear classification model that predicts the probability of a binary outcome. It models the probability ($y=1|x$) using the logistic function given as (6).

$$P \left(y = \frac{1}{x} \right) = \frac{1}{1 + e^{w^T x}} \quad (6)$$

Here, x is the input vector, w is the weight vector, and e is the base of the natural logarithm. RF classifier creates multiple decision trees during the training phase. Let us consider $T_i(x)$ to be the prediction of the i -th decision tree, then the predicted class label for input x is given as per (7). Here n indicates the number of decision trees.

$$\hat{y} = \text{mode} \{T_1(x), T_2(x), \dots, T_n(x)\} \quad (7)$$

GB is also an ensemble method like RF classifier. It builds a strong model by sequentially accumulating weak learners (typically decision trees) and also amending the errors made by previous learners. The prediction of the ensemble is a weighted sum of the predictions of all the weak learners. Consider $F_m(x)$ as the prediction of the m -th weak learner. The final prediction is computed as per (8).

$$\hat{y} = \sum_{m=1}^M \lambda_m F_m(x) \quad (8)$$

Here, M is the number of weak learners and λ_m are the corresponding weights.

3.6. Description of datasets

The software fault prediction datasets referenced, namely JM1, AR1, CM1, KC2, MW1, PC1, MC2, and KC1, are well-known datasets commonly used in research for evaluating ML models in the context of software defect prediction. These datasets contain a wide range of static code metrics and attributes associated with software modules, which serve as features for training predictive models. The JMI dataset is often used for evaluating ML models in software defect prediction tasks. It contains static code metrics and attributes extracted from Java projects, including measures related to code complexity, size, and object-oriented design properties.

Attributes may include lines of code, McCabe's cyclomatic complexity, Halstead's metrics, and various other software metrics [28], [29]. The research in [30]–[32] it demonstrates how these metrics play a role in software fault prediction. The dataset provides labeled instances indicating whether a software module contains defects or not. They consist of attributes like static code metrics and attributes collected from software repositories, focusing on code churn and change-related metrics.

Attributes may include lines added, lines deleted, number of code modifications, and other metrics related to code evolution and change patterns. Some attributes like combination of size, complexity, and design-related metrics, depending on the specific characteristics of the software projects from which the data was collected, are included. Overall, these datasets offer a rich source of information for training and evaluating machine-learning models for software fault prediction. By leveraging the diverse set of static code metrics and attributes provided in these datasets, researchers can develop robust predictive models capable of identifying potential defects in software modules, thereby aiding in the improvement of software quality and reliability.

4. RESULTS ANALYSIS AND DISCUSSION

Performance evaluation and testing are critical aspects of assessing the effectiveness and efficiency of the implemented software fault prediction system. We explain the evaluation process into two components: algorithm time complexity and testing methodologies. When evaluating the time complexity of different data balancing techniques and ML algorithms, it's essential to consider both the time required for training the models and the time needed for making predictions on new data.

GANs demonstrated a time complexity of approximately 25 minutes per dataset. GANs typically require more time for training compared to other balancing techniques due to their complex architecture and iterative training process. However, they can generate more accurate and precise synthetic data. Other balancing techniques (e.g., SMOTE, NearMiss) demonstrated a time complexity that was typically 1 to 2 seconds. Traditional data balancing techniques like SMOTE and NearMiss are computationally less expensive compared to GANs. They involve simpler algorithms and generate balanced datasets more quickly. When evaluating the time complexity of ML algorithms, it's crucial to consider both training time and prediction time. Decision trees have a time complexity as $O(n \cdot m \log m)$, where n is the number of samples, while m is the number of features. The time complexity of RF is $O(n \cdot m \log m \cdot k)$, where k is the number of trees existing in the forest.

SVM has a time complexity of $O(n^2 \cdot m)$, where n is the number of samples and m is the number of features. SVMs can be computationally expensive for large datasets hence, we did not employ it here. KNN has a time complexity of $O(n \cdot k \cdot m)$, where n is the number of samples, k is the number of neighbors, and m is the number of features. LR has a time complexity of $O(n \cdot m)$, where n is the number of samples and m is the number of features. K-Means has a time complexity of $O(n \cdot k \cdot t \cdot m)$, where n is the number of samples, k is the number of clusters, t is the number of iterations, and m is the number of features. Analyzing model performance on the KC2 dataset across different data balancing techniques provides valuable insights into their effectiveness. GAN balanced data consistently enhances accuracy across models compared to the unbalanced scenario, with RF leading in performance. This suggests the efficacy of GAN balancing in improving model accuracy without significant data loss. However, employing data balancing techniques such as SMOTE and NearMiss further improves accuracy, notably compared to the unbalanced data. SMOTE balanced data leads to significant accuracy improvements across models, with RF and KNN achieving the highest accuracy. NearMiss balanced data also enhances accuracy, but to a lesser extent compared to SMOTE balancing.

Figure 2 depicts the generator and discriminator loss of KC2, while Figure 3 depicts the comparison of accuracies for ML algorithms applied to KC2. In summary, while GAN balancing shows promise, SMOTE balancing emerges as the most effective technique for improving model performance on the KC2 dataset, followed by NearMiss balancing. These insights aid in selecting appropriate data balancing techniques for classification tasks on the KC2 dataset. In the analysis for the KC2 dataset, the generator loss reached its minimum at epoch 3452 as depicted in Figure 2, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained.

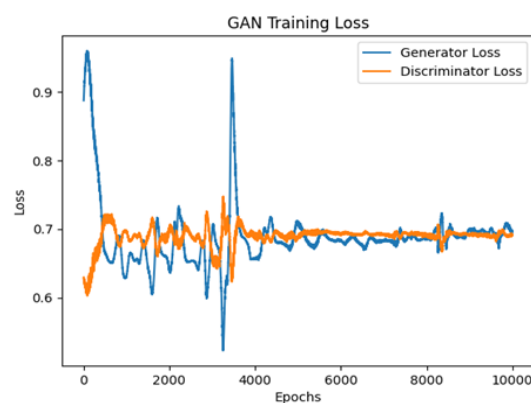


Figure 2. Generator and discriminator loss of KC2

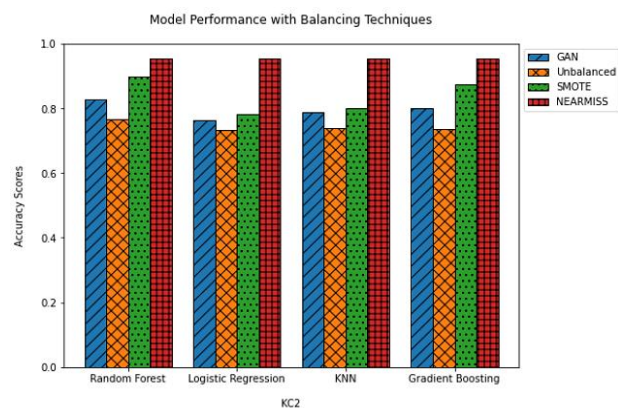


Figure 3. Accuracy comparison of KC2

Analyzing model performance on the JM1 dataset across various data balancing techniques reveals insightful trends. In the analysis for the JM1 dataset, as depicted in Figure 4, the generator loss reached its minimum at epoch 1132, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained. GAN-balanced data consistently improves accuracy compared to the unbalanced scenario across models, with RF showcasing particularly high accuracy. This underscores GAN balancing's effectiveness in enhancing model performance while maintaining data

integrity. However, employing techniques like SMOTE and NearMiss further enhances accuracy, notably compared to the unbalanced data. SMOTE balanced data notably improves accuracy across models, though LR exhibits lower accuracy with this technique, as can be inferred from Figure 5. NearMiss balanced data also enhances accuracy, albeit to a lesser extent compared to SMOTE balancing. In summary, while GAN balancing shows promise, SMOTE balancing emerges as the most effective technique for improving model performance on the JM1 dataset, followed closely by NearMiss balancing. These insights assist in selecting appropriate data balancing techniques for classification tasks on the JM1 dataset.

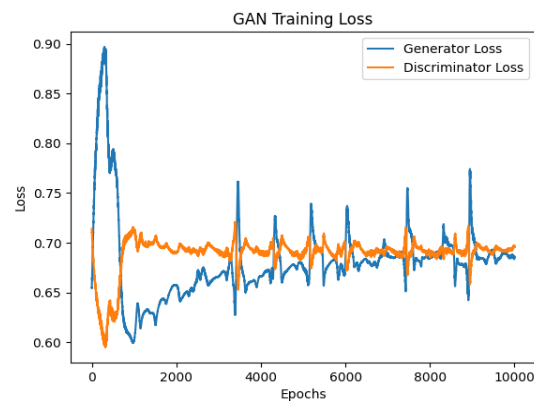


Figure 4. Generator and discriminator loss of JM1

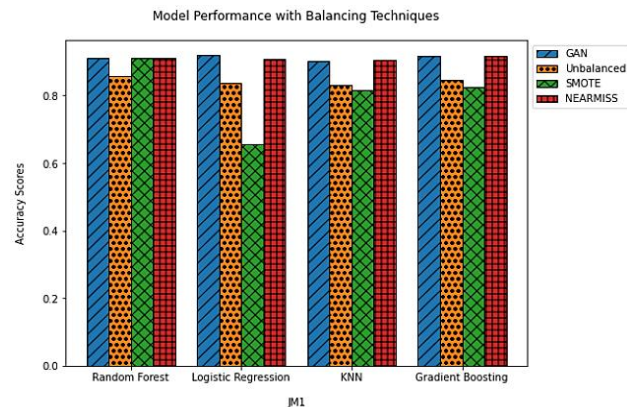


Figure 5. Accuracy comparison of JM1

In the analysis for the KC1 dataset, the generator loss reached its minimum at epoch 4128 as depicted in Figure 6. It indicates the optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained. The KC1 dataset analysis reveals that GAN balanced data consistently improves accuracy across models, indicating its effectiveness without data loss. Unbalanced data leads to lower accuracy, but SMOTE and NearMiss techniques notably enhance it. SMOTE balanced data varies in effectiveness among models, with RF and KNN benefiting the most as can be seen from Figure 7. NearMiss balanced data maintains high accuracy, though not significantly better than SMOTE balanced data. In summary, GAN balancing is robust for KC1, but the effectiveness of SMOTE and NearMiss techniques varies across models, aiding in selecting suitable balancing techniques for classification tasks on the KC1 dataset.

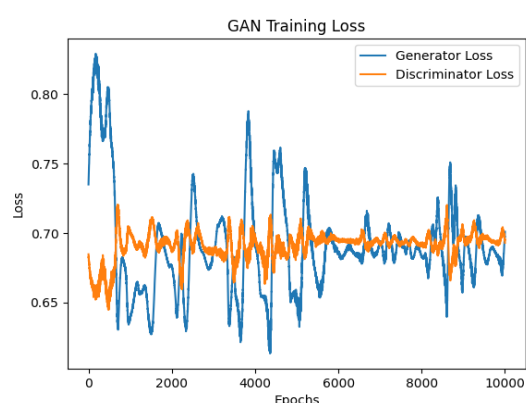


Figure 6. Generator and discriminator loss of KC1

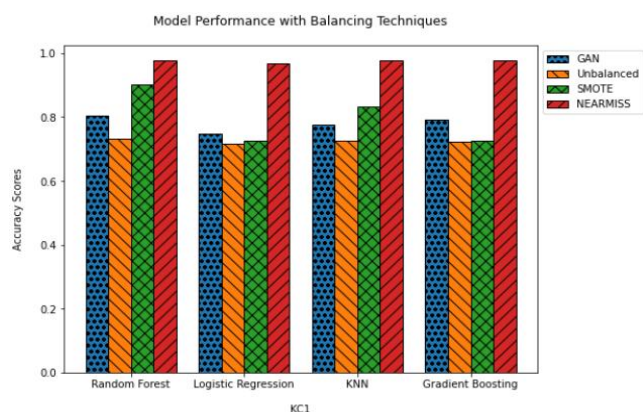


Figure 7. Accuracy comparison of KC1

Figure 8 depicts the generator and discriminator loss of PC1 dataset. In the analysis for the PC1 dataset, the generator loss reached its minimum at epoch 1767, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged,

producing the highest quality synthetic data among the 10,000 epochs trained. Analyzing the PC1 dataset's data balancing techniques reveals distinct patterns in model performance. GAN balanced data consistently boosts accuracy across models, with RF achieving the highest accuracy. Conversely, unbalanced data leads to decreased accuracy, underlining the challenges posed by imbalanced class distributions. However, both SMOTE and NearMiss techniques notably enhance accuracy compared to the unbalanced scenario. SMOTE balanced data varies in effectiveness among models, particularly benefiting RF and GB. NearMiss balanced data maintains consistently high accuracy, achieving perfect scores for RF and GB. Figure 9 depicts the accuracy of PC1 datasets across all the models developed. However, LR and KNN show slightly lower accuracy compared to other techniques. In summary, GAN balancing emerges as a robust strategy for improving model performance on the PC1 dataset, while the effectiveness of SMOTE and NearMiss techniques varies across models. These insights offer guidance in selecting appropriate data balancing techniques for classification tasks on the PC1 dataset. In the analysis for the CM1 dataset, the generator loss reached its minimum at epoch 6189, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained.

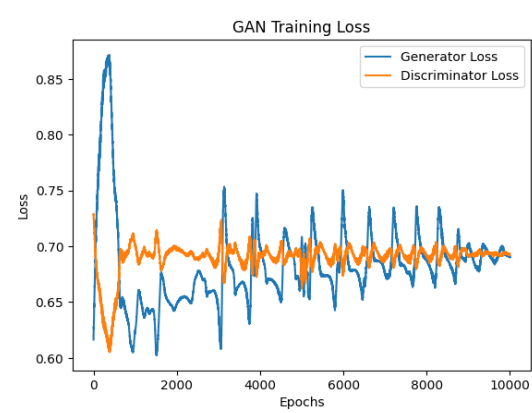


Figure 8. Generator and discriminator loss of PC1

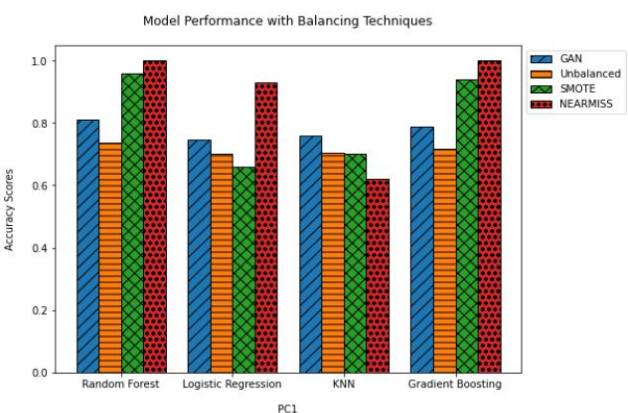


Figure 9. Accuracy comparison of PC1

Figure 10 demonstrates the generator and discriminator loss for CM1 dataset, while Figure 11 depicts the accuracy obtained across ML models developed in the study of CM1 dataset. Analyzing the CM1 dataset's data balancing techniques sheds light on model performance across various scenarios. GAN balanced data consistently enhances accuracy across models, with RF leading in performance. This underscores the effectiveness of GAN balancing in improving model accuracy without significant data loss. Conversely, unbalanced data leads to reduced accuracy, highlighting the challenges posed by imbalanced class distributions.

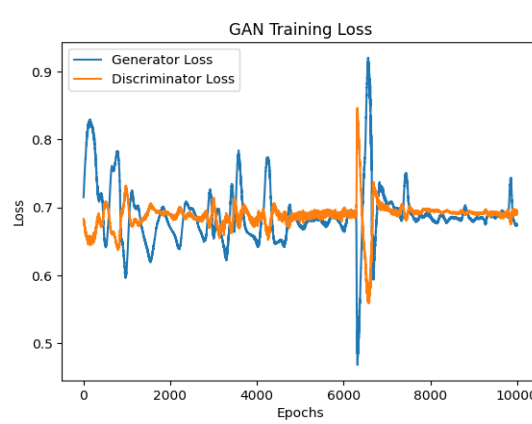


Figure 10. Generator and discriminator loss of CM1

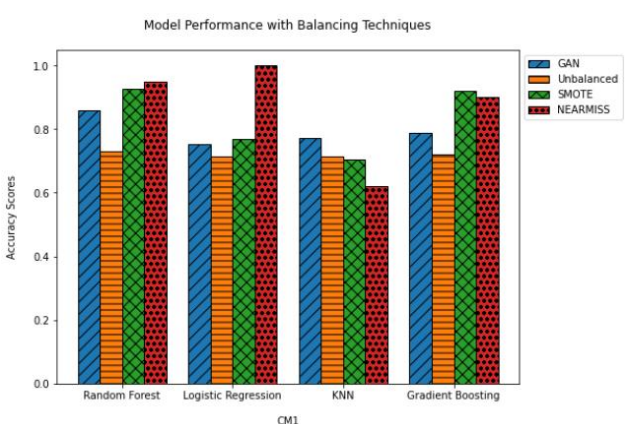


Figure 11. Accuracy comparison of CM1

SMOTE balanced data demonstrates varied effectiveness among models, particularly benefiting RF and GB. This technique substantially enhances accuracy across models compared to the unbalanced scenario. NearMiss balanced data maintains high accuracy levels, particularly achieving perfect scores for LR. However, KNN exhibits slightly lower accuracy compared to other techniques. In summary, GAN balancing emerges as a reliable strategy for enhancing model performance on the CM1 dataset, while SMOTE and NearMiss techniques offer additional improvements with varying effectiveness among models. These insights provide guidance for selecting appropriate data balancing techniques for classification tasks on the CM1 dataset.

Figure 12 depicts the generator and discriminator loss of MC2, while Figure 13 depicts the accuracy of MC2 for the ML models considered. In the analysis for the MC2 dataset, the generator loss reached its minimum at epoch 8998, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained. Comparing data balancing techniques on the MC2 dataset reveals distinct performance patterns. GAN balanced data generally enhances model accuracy by generating balanced datasets without significant information loss. Unbalanced data leads to lower accuracy as models struggle with minority class representation. SMOTE balanced data varies in effectiveness among models; LR notably struggles. NearMiss balanced data slightly decreases KNN's accuracy compared to other techniques. In summary, while GAN balancing consistently improves accuracy, the effectiveness of SMOTE and NearMiss techniques varies across models. These insights aid in selecting the most suitable balancing technique for classification tasks on the MC2 dataset, considering specific model requirements and dataset characteristics.

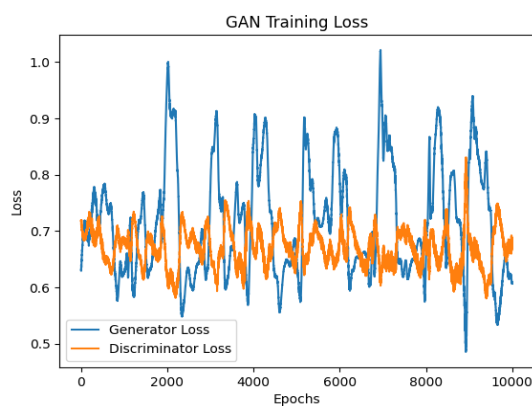


Figure 12. Generator and discriminator loss of MC2

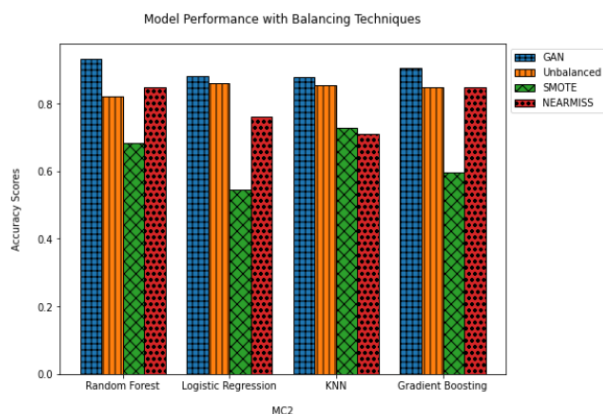


Figure 13. Accuracy comparison of MC2

Figure 14 depicts the generator and discriminator loss of MC2 while Figure 15 depicts the accuracy of MC2 for ML models considered. Analyzing model performance on the MW1 dataset across various data balancing techniques reveals notable trends. GAN balanced data consistently enhances accuracy compared to the unbalanced scenario, with RF achieving the highest accuracy among models. This underscores the effectiveness of GAN balancing in improving model performance without significant data loss. However, employing techniques like SMOTE and NearMiss further boosts accuracy notably compared to the unbalanced data. SMOTE balanced data notably improves accuracy across models, with LR achieving the highest accuracy. NearMiss balanced data also enhances accuracy but to a lesser extent compared to SMOTE balancing. In summary, while GAN balancing shows promise, SMOTE balancing emerges as the most effective technique for enhancing model performance on the MW1 dataset, followed by NearMiss balancing. These insights guide in selecting appropriate data balancing techniques for classification tasks on the MW1 dataset. In the analysis for the MW1 dataset, the generator loss reached its minimum at epoch 3649, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained.

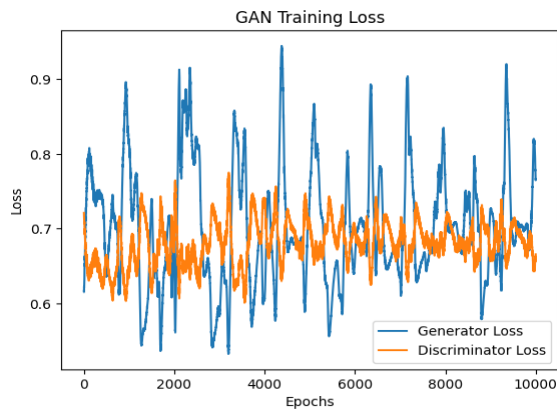


Figure 14. Generator and discriminator loss of MC2

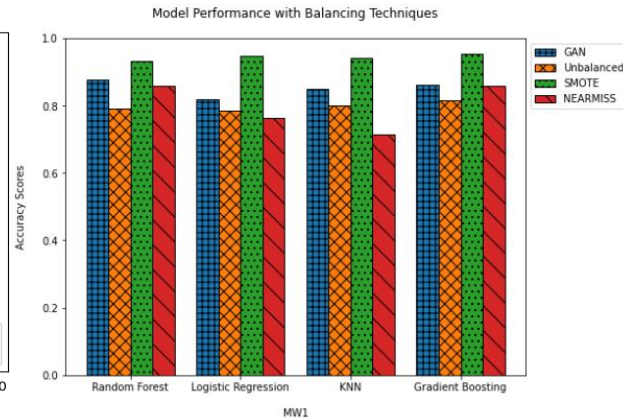


Figure 15. Accuracy comparison of MW1

Figure 16 depicts the generator and discriminator loss of AR1 while Figure 17 depicts the accuracy of AR1 for ML models considered. Examining model performance on the AR1 dataset across diverse data balancing strategies provides insightful observations. GAN balanced data generally enhances accuracy compared to the unbalanced setting, especially benefiting RF and KNN models. This underscores the effectiveness of GAN balancing in improving model performance while maintaining data integrity. Interestingly, the Unbalanced data scenario already yields high accuracy across all models, suggesting minimal impact on model performance in this specific scenario.

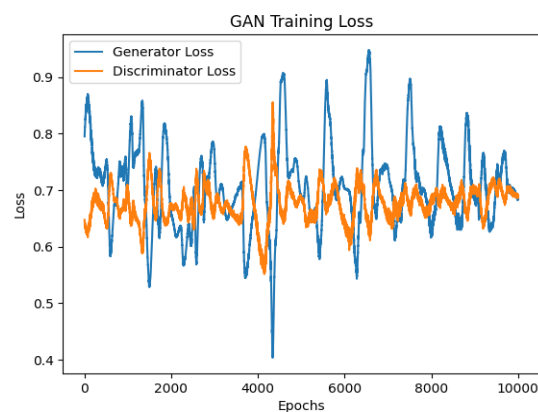


Figure 16. Generator and discriminator loss of AR1

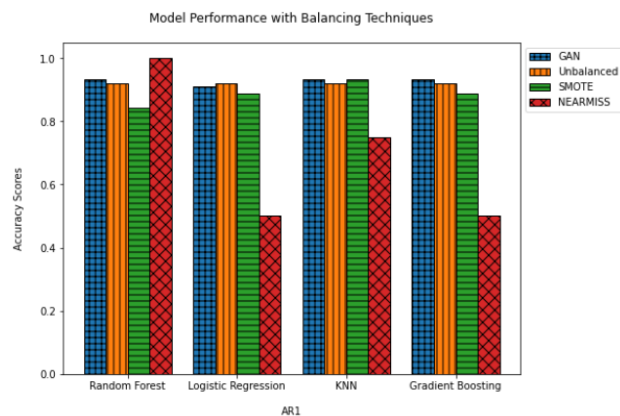


Figure 17. Accuracy comparison of AR1

However, the outcomes are more varied with SMOTE balanced data, with RF and KNN achieving higher accuracy, while LR and GB exhibit lower accuracy compared to other techniques. Moreover, NearMiss balanced data shows both positive and negative impacts, enhancing accuracy for some models while significantly reducing it for others. In summary, while GAN balancing consistently boosts model accuracy for the AR1 dataset, the effectiveness of SMOTE and NearMiss balancing techniques varies across models. These findings offer valuable insights for selecting suitable data balancing methods for classification tasks on the AR1 dataset. In the analysis for the AR1 dataset, the generator loss reached its minimum at epoch 4078, indicating optimal performance for synthetic data generation using the GAN model. Meanwhile, the discriminator loss also decreases, indicating improved discrimination between real and synthetic data. This milestone suggests that the generator has converged, producing the highest quality synthetic data among the 10,000 epochs trained. Table 1 presents a summary of the overall results obtained by developing 128 models (8 datasets*4 data balancing approaches*4 ML models).

The research findings on synthetic data generation and defect prediction in software engineering have several potential applications in real-world scenarios like software quality assurance, resource allocation, risk management, automated bug detection, software maintenance and evolution, continuous

integration/continuous deployment (CI/CD), security vulnerability identification, software project management, and decision support systems. Organizations can proactively identify and fix potential issues before they impact end-users, leading to higher software quality and increased customer satisfaction. Understanding where defects are likely to occur allows organizations to allocate resources more efficiently. They can focus testing efforts on critical areas of the codebase, reducing the time and cost associated with software testing. Predicting defects helps in assessing project risks more accurately. Synthetic data generation can be used to augment real-world data during code reviews. By simulating various scenarios and injecting synthetic defects, developers can gain deeper insights into the robustness and maintainability of their code. ML models trained on historical defect data can be integrated into automated testing tools to detect potential bugs automatically. This accelerates the bug detection process and reduces the manual effort required for testing. Organizations can focus their efforts on areas of the codebase that are most likely to require updates or improvements. Incorporating defect prediction models into CI/CD pipelines enables organizations to identify potential issues early in the development lifecycle. This promotes a culture of continuous improvement and ensures that software releases are more stable and reliable. Similar techniques can be applied to predict security vulnerabilities in software systems. By analyzing historical data on security incidents and code changes, organizations can proactively identify and address potential security threats. Predictive models can provide valuable insights into the overall health and progress of software projects. Project managers can use these insights to make data-driven decisions and optimize project planning and execution. Integrating defect prediction models into decision support systems enables stakeholders to make informed decisions regarding software development and maintenance activities. This promotes transparency and accountability throughout the software development lifecycle. Overall, the applications of research findings in synthetic data generation and defect prediction have the potential to significantly improve the efficiency, reliability, and security of software systems in various domains.

Table 1. Accuracy comparison for KC2, JM1, KC1, PC1, CM1, MC2, MW1, and AR1

Dataset	Model	GAN balanced data	Unbalanced data	SMOTE balanced data	NearMiss balanced data
KC2	RF	0.827	0.765	0.897	0.953
	LR	0.763	0.732	0.783	0.953
	KNN	0.788	0.739	0.801	0.953
	GB	0.8	0.735	0.873	0.953
JM1	RF	0.911	0.856	0.91	0.911
	LR	0.919	0.836	0.655	0.907
	KNN	0.903	0.83	0.816	0.906
	GB	0.918	0.847	0.826	0.916
KC1	RF	0.804	0.732	0.903	0.977
	LR	0.748	0.715	0.725	0.969
	KNN	0.777	0.726	0.834	0.977
	GB	0.793	0.722	0.725	0.977
PC1	RF	0.81	0.736	0.96	1
	LR	0.747	0.7	0.66	0.93
	KNN	0.759	0.703	0.7	0.62
	GB	0.789	0.718	0.94	1
CM1	RF	0.86	0.73	0.927	0.95
	LR	0.752	0.714	0.77	1
	KNN	0.772	0.715	0.704	0.62
	GB	0.789	0.721	0.92	0.9
MC2	RF	0.932	0.822	0.682	0.85
	LR	0.882	0.862	0.545	0.76
	KNN	0.878	0.856	0.727	0.71
	GB	0.906	0.85	0.596	0.85
MW1	RF	0.877	0.791	0.932885906	0.857
	LR	0.818	0.784	0.946308725	0.761
	KNN	0.85	0.801	0.939597315	0.714
	GB	0.863	0.814	0.953020134	0.857
AR1	RF	0.933	0.92	0.844	1
	LR	0.911	0.92	0.888	0.5
	KNN	0.9333	0.92	0.933	0.75
	GB	0.9333	0.92	0.888	0.5

5. CONCLUSION

One of the primary accomplishments was the successful implementation and training of the GAN model for synthetic data generation in MC2 dataset that enabled learning the underlying distribution of defect-related features. Training the GAN model for 10,000 epochs allowed sufficient time for the generator to converge to its optimal state, resulting in the selection of the best-performing model based on the lowest

generator loss. Various classifiers, such as LR, RF, and support vector machine, were employed to assess the performance of defect prediction models. Overall, the work represents a significant advancement in the field of defect prediction and synthetic data generation, with implications for improving the quality, reliability, and maintainability of software systems. By leveraging ML techniques and innovative approaches to data augmentation, the project has laid the foundation for future research and development efforts aimed at addressing the challenges associated with limited and imbalanced datasets in software engineering. Moving forward, the insights gained from this project can inform the design of more robust and accurate defect prediction models, ultimately contributing to the advancement of software quality assurance practices and the enhancement of software development processes. The GAN model demonstrated impressive results in generating realistic synthetic samples for defect prediction, there is room for improvement in terms of fine-tuning the model architecture, optimizing hyperparameters, and exploring alternative approaches to data augmentation. By leveraging advancements in deep learning and generative modeling, researchers can devise more sophisticated and efficient methods for synthesizing diverse and representative datasets, thereby improving the robustness and generalization capabilities of defect prediction models.

ACKNOWLEDGMENTS

The authors sincerely thank Mrs. Sonal Gadawe, Technical Staff at the Department of Computer Engineering and Technology, for her valuable support in data pre-processing. Consent has been obtained from the individual acknowledged for inclusion in this section.

FUNDING INFORMATION

The authors declare that no funding was received for conducting this study.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Akshat Raj		✓	✓	✓	✓	✓		✓	✓					
Durva Mahadeo Chavan		✓	✓	✓	✓	✓		✓	✓					
Priyal Agarwal		✓	✓	✓	✓	✓		✓	✓					
Jestin Gigi		✓	✓	✓	✓	✓		✓	✓					
Madhuri Rao	✓			✓						✓		✓	✓	
Vinayak Musale		✓	✓	✓	✓		✓	✓		✓				
Akshita Chanchlani		✓	✓	✓	✓		✓	✓	✓					
Murtaza Shabbirbhai		✓	✓	✓	✓		✓	✓		✓	✓			
Dholkawala														
Kulamala Vinod Kumar	✓	✓	✓		✓	✓	✓		✓		✓	✓	✓	

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

All data and scripts used in this study are publicly available in the GitHub repository at <https://github.com/akshatraj36/GAN>.




REFERENCES

- [1] M. Mustaqeem, M. Alam, S. Mustajab, F. Alshanketi, S. Alam, and M. Shuaib, "Comprehensive bibliographic survey and forward-looking recommendations for software defect prediction: datasets, validation methodologies, prediction approaches, and tools," *IEEE Access*, vol. 13, pp. 866–903, 2025, doi: 10.1109/ACCESS.2024.3517419.
- [2] K. V. Kumar, P. Kumari, M. Rao, and D. P. Mohapatra, "Metaheuristic feature selection for software fault prediction," *Journal of Information and Optimization Sciences*, vol. 43, no. 5, pp. 1013–1020, 2022, doi: 10.1080/02522667.2022.2103301.
- [3] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "Machine learning based methods for software fault prediction: a survey," *Expert Systems with Applications*, vol. 172, 2021, doi: 10.1016/j.eswa.2021.114595.
- [4] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012, doi: 10.1109/TSE.2011.103.
- [5] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: a proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008, doi: 10.1109/TSE.2008.35.
- [6] Y. Zhao, Y. Hu, and J. Gong, "Research on international standardization of software quality and software testing," *IEEE/ACIS 21st International Conference on Computer and Information Science (ICIS)*, pp. 56–62, 2021, doi: 10.1109/ICISFall51598.2021.9627426.
- [7] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1806–1817, 2012, doi: 10.1109/TSMCC.2012.2226152.
- [8] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: datasets, data validation methods, approaches, and tools," *Engineering Applications of Artificial Intelligence*, vol. 111, 2022, doi: 10.1016/j.engappai.2022.104773.
- [9] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: interpretable representation learning by information maximizing generative adversarial nets," *30th Conference on Neural Information Processing Systems (NIPS 2016)*, 2016.
- [10] U. Aitimova et al., "Data generation using generative adversarial networks to increase data volume," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 2, pp. 2369–2376, 2024, doi: 10.11591/ijece.v14i2.pp2369-2376.
- [11] Y. Yuan and Y. Guo, "A review on generative adversarial networks," *5th International Conference on Information Science, Computer Technology and Transportation (ISCTT)*, pp. 392–401, 2020, doi: 10.1109/ISCTT51595.2020.00074.
- [12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.
- [13] N. Alamsyah, Budiman, T. P. Yoga, and R. Y. R. Alamsyah, "A stacking ensemble model with SMOTE for improved imbalanced classification on credit data," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 22, no. 3, pp. 657–664, 2024, doi: 10.12928/TELKOMNIKA.v22i3.25921.
- [14] N. M. Mqadi, N. Naicker, and T. Adeliyi, "Solving misclassification of the credit card imbalance problem using near miss," *Mathematical Problems in Engineering*, vol. 2021, no. 1, 2021, doi: 10.1155/2021/7194728.
- [15] J. S. Joy, F. F. Sakib, M. S. I. Juel, M. R. Prottasha, and R. Karim, "Performance comparison of ensemble and supervised models for software defect prediction," *International Conference on Robotics, Electrical and Signal Processing Techniques*, pp. 234–238, 2025, doi: 10.1109/ICREST63960.2025.10914464.
- [16] S. Pandey and K. Kumar, "Analysis of different sampling techniques for software fault prediction," *Security, Privacy and Data Analytics*, pp. 59–71, 2023, doi: 10.1007/978-981-99-3569-7_5.
- [17] A. Mehta, N. Kaur, and A. Kaur, "Addressing class imbalance in software fault prediction using BVPC-SENN: a hybrid ensemble approach," *International Journal of Performability Engineering*, vol. 21, no. 2, pp. 94–103, 2025, doi: 10.23940/ijpe.25.02.p4.94103.
- [18] J. Engelmann and S. Lessmann, "Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning," *Expert Systems with Applications*, vol. 174, 2021, doi: 10.1016/j.eswa.2021.114582.
- [19] A. Alqarni and H. Aljamaan, "Leveraging ensemble learning with generative adversarial networks for imbalanced software defects prediction," *Applied Sciences*, vol. 13, no. 24, 2023, doi: 10.3390/app132413319.
- [20] P. S. Kumar and D. R. Venkatesan, "Improving software defect prediction using generative adversarial networks," *International Journal of Science and Engineering Applications*, vol. 9, no. 9, pp. 117–120, Sep. 2020, doi: 10.7753/IJSEA0909.1001.
- [21] S. Feng, J. Keung, X. Yu, Y. Xiao, and M. Zhang, "Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction," *Information and Software Technology*, vol. 139, 2021, doi: 10.1016/j.infsof.2021.106662.
- [22] M. Prashanthi, G. Sumalatha, K. Mamatha, and K. Lavanya, "Software defect prediction survey introducing innovations with multiple techniques," *Cognitive Science and Technology*, pp. 783–793, 2023, doi: 10.1007/978-981-19-8086-2_75.
- [23] S. S. Rathore, "An exploratory analysis of regression methods for predicting faults in software systems," *Soft Computing*, vol. 25, no. 23, pp. 14841–14872, 2021, doi: 10.1007/s00500-021-06048-x.
- [24] K. V. Kumar, P. Kumari, A. Chatterjee, and D. P. Mohapatra, "Software fault prediction Using random forests," *Smart Innovation, Systems and Technologies*, vol. 194, pp. 95–103, 2021, doi: 10.1007/978-981-15-5971-6_10.
- [25] R. Goyal, P. Chandra, and Y. Singh, "Suitability of KNN regression in the development of interaction based software fault prediction models," *IERI Procedia*, vol. 6, pp. 15–21, 2014, doi: 10.1016/j.ieri.2014.03.004.
- [26] J. Goyal and R. R. Sinha, "Software defect-based prediction using logistic regression: review and challenges," *Second International Conference on Sustainable Technologies for Computational Intelligence*, pp. 233–248, 2022, doi: 10.1007/978-981-16-4641-6_20.
- [27] N. Saravanan, C. Dharanya, M. Dhina, R. E. Kumar, and S. L. Devi, "A novel approach to predict the defect density in software application using linear regression algorithm," *2024 International Conference on Science Technology Engineering and Management (ICSTEM)*, 2024, pp. 1–5, doi: 10.1109/ICSTEM61137.2024.10560850.
- [28] B. Gezici and A. K. Tarhan, "Explainable AI for software defect prediction with gradient boosting classifier," *7th International Conference on Computer Science and Engineering (UBMK)*, 2022, pp. 49–54, doi: 10.1109/UBMK55850.2022.9919490.
- [29] V. Côté, P. Bourque, S. Oligny, and N. Rivard, "Software metrics: an overview of recent results," *The Journal of Systems and Software*, vol. 8, no. 2, pp. 121–131, 1988, doi: 10.1016/0164-1212(88)90005-2.
- [30] H. Tu, W. Sun, and Y. Zhang, "The research on software metrics and software complexity metrics," *IFCSTA 2009 Proceedings - 2009 International Forum on Computer Science-Technology and Applications*, vol. 1, pp. 131–136, 2009, doi: 10.1109/IFCSTA.2009.39.
- [31] M. Alam and M. Mustaqeem, "Reinforcing defect prediction: a reinforcement learning approach to mitigate class imbalance in software defect prediction," *Iran Journal of Computer Science*, vol. 8, no. 1, pp. 151–162, 2025, doi: 10.1007/s42044-024-00214-8.




- [32] G. Hao *et al.*, “Complementarity in software code complexity metrics,” *Journal of Systems and Software*, vol. 232, 2026, doi: 10.1016/j.jss.2025.112679.

BIOGRAPHIES OF AUTHORS






Akshat Raj    is currently pursuing his Bachelor of Technology undergraduate course from the Dr. Vishwanath Karad MIT World Peace University, located in the city of Pune in Maharashtra, India. His research interests include artificial intelligence, generative AI, and prompt engineering. He can be contacted at email: rajakshat130@gmail.com.






Durva Mahadeo Chavan    is currently pursuing her Bachelor of Technology undergraduate course from the Dr. Vishwanath Karad MIT World Peace University located in the city of Pune in Maharashtra, India. Her research interests include artificial intelligence, machine learning, and natural language processing. She can be contacted at email: durvac102@gmail.com.






Priyal Agarwal    is currently pursuing her Bachelor of Technology undergraduate course from the Dr. Vishwanath Karad MIT World Peace University located in the city of Pune in Maharashtra, India. Her research interests include machine learning and deep learning. She can be contacted at email: 1032201406@mitwpu.edu.in.







Jestin Gigi    is currently pursuing his Bachelor of Technology undergraduate course from the Dr. Vishwanath Karad MIT World Peace University located in the city of Pune in Maharashtra, India. His research interests include artificial intelligence, generative AI, and IoT. He can be contacted at email: jestingigi@gmail.com.







Madhuri Rao    is currently working as a Senior Assistant Professor at Dr. Vishwanath Karad MIT World Peace University. She acquired her Doctoral Degree from the Biju Patnaik University of Technology, Rourkela, and received the best research scholar award in the year 2019 under TEQIP-III, MHRD, Gov. of India. She obtained her Bachelor of Technology Degree from Biju Patnaik University of Technology and Master of Technology from Bharath University in the years 2005 and 2008, subsequently. She has published her research work as journal papers, book chapters, and conference papers. She is interested in distributed systems and machine learning. She can be contacted at email: madhurirao.iter.soa@gmail.com.







Dr. Vinayak Musale     is a distinguished academic and researcher with over 15 years of experience in the field of Computer Science and Engineering. He is currently serving as an Assistant Professor in the Department of Computer Engineering and Technology at Dr. Vishwanath Karad MIT World Peace University, Pune. He holds a Ph.D. in Computer Science and Engineering from SGB Amravati University, and he has an extensive teaching portfolio that covers a wide range of subjects, including programming languages, operating systems, advanced web technologies, cybersecurity, digital forensics, and more. His research interests are deeply rooted in cybersecurity, wireless sensor networks, IoT, machine learning, advanced computing, and blockchain techniques. He has published over 45 research papers in various national and international conferences and journals, many of which are indexed in Scopus, Web of Science, and IEEE Xplore. He has also contributed as a research paper reviewer for several conferences and journals. He also contributed as a research paper reviewer for various journals and national and international conferences. He can be contacted at email: vinayak.musale@gmail.com.







Akshita Chanchlani     holds a Ph.D. in Computer Science and Engineering from Sant Gadge Baba Amravati University, Amravati, India. She contributes to the educational team at MITWPU Pune. With over 15 years of experience, she has a robust professional background spanning both academia and the technical industry. Her roles have included associate head technical trainer, corporate trainer, and assistant professor, accumulating extensive experience in teaching and delivering industrial and technical corporate training. She possesses a proven track record of equipping students to produce high-quality work across a diverse range of technical training areas, including C and C++ programming, Core Java, Python programming, machine learning, deep learning, artificial intelligence, and big data technologies. She has over 25 research paper publications in reputable journals and numerous book chapters. She can be contacted at email: akshita.chanchlani@mitwpu.edu.in.



Dr. Murtaza Shabbirbhai Dholkawala     holds Ph.D. in Computer Engineering with 16 years of rich experience, mostly in Technical Industry and some in academia. He started with junior developer and went till senior technical manager with a strong background in react JS, low-code platforms, and Python development. He has successfully contributed to a variety of complex projects across multiple domains, collaborating with prominent organizations such as Cummins India, IDFC First. His work spans full-stack development, application design, and process automation, leveraging both traditional coding and low-code solutions to deliver efficient and scalable outcomes. He brings a research-driven approach to problem-solving and software innovation. He can be contacted at email: imgeminite@gmail.com.



Kulamala Vinod Kumar     is currently working as an Assistant Professor at MIT World Peace University in Pune, Maharashtra, India, and has more than a decade of teaching and research experience. He is also pursuing his Doctoral studies from National Institute of Technology, Rourkela. He acquired his Bachelor of Technology Degree from Osmania University, Hyderabad, India, and his Master of Technology degree from Hyderabad Central University, Hyderabad, India, in the year 2005 and 2008, subsequently. He has authored many research papers in reputed journals, book chapters, and conference papers. His research interests include software engineering, software fault prediction, and machine learning. He can be contacted at email: kvinod2208@gmail.com.