Scientific Reports

https://doi.org/10.1038/s41598-026-41981-7

Article in Press

A novel generative oversampling for software defect prediction

Received: 13 April 2025

Accepted: 24 February 2026

Published online: 03 April 2026

Cite this article as: Goyal S.R. A novel generative oversampling for software defect prediction. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-41981-7

Somya R. Goyal

A Novel Generative Oversampling for Software Defect Prediction

Somya R. Goyal

Manipal University Jaipur
Jaipur-303007, Rajasthan, INDIA
somyagoyal1988@gmail.com; somya.goyal@jaipur.manipal.edu

ABSTRACT

Software defect prediction is one of the highly active research areas as it allows to focus the testing efforts on the defective modules and reduce the cost of development. The imbalanced nature of defect data poses a threat to the performance of software defect predictors. This study proposes a novel Generative oversampling-based Software Defect Prediction naming GeNSDP. It oversamples defect data by generating synthetic minority instances utilizing lightweight generative model, then the oversampled data is used for defect prediction using deep network. NASA and PROMISE datasets are used for experimentation, for which GeNSDP achieves a remarkable average score of 99.1% for Area Under the Curve, and 0.92 for F-measure. The proposed model outperforms the traditional oversampling methods (including ROS, SMOTE, COSTE) by 30.1%, and selected baseline models by 14.1%. From the statistical evidence obtained by conducting the Anova Test with Bonferroni Post-hoc test at the confidence level of 95%, it is concluded that the proposed model is effective to handle class imbalance and achieve stable defect prediction.

Keywords- Software Defect Prediction (SDP); Class Imbalance; Sampling; Generative oversampling; deep networks.

Notations Used-

GeNSDP	Generative oversampling for Software Defect Prediction	EBSE	Evidence Based Software Engineering
SDP	Software Defect Prediction	$\lvert D_{clean} \rvert$	Data-points labelled as 'Clean'
GAN	Generative Adversarial Network	$\lvert D_{buggy} \rvert$	Data-points labelled as 'Buggy'
ML	Machine Learning	ROS	Random Over Sampling
CIL	Class Imbalance Learning	SMOTE	Synthetic Minority Oversampling TEchnique
DL	Deep Learning	COSTE	Complexity based OverSampling TEchnique
RQ	Research Question	ROC	Receiver Operating Characteristic
AUC	Area under the curve	SOTA	State of the Art

1. INTRODUCTION

Software defect prediction (SDP) has been an active research area for the past four decades. It permits to have a screening of defective modules before the commencement of the testing phase. Testing efforts and personnel can be made more focused on those identified modules. SDP concessions the development cost and time by appropriate management of testing resources. Since the 1990s, Machine Learning (ML) techniques have been uncovering distinguished applications in the field of SDP [1] [2]. The performance of ML models is negatively affected by the imbalanced datasets [3] [4].

Class imbalance is the condition which causes a large difference in number of samples belonging to 'fault-prone' class and 'fault-free' class in the defect dataset. It is found that 'fault-prone' class, which is of interest, is minority in comparison to 'fault-free' class [5]. This imbalanced nature of dataset impairs the accurate prediction of risky modules. Class Imbalance Learning (CIL) is essential to improve the performance of ML techniques and to achieve unbiased and accurate predictions of fault-prone modules [6] [19]. The situation of class-imbalance in defect datasets is shown in *Figure 1*. In this figure the imbalance between the red bubbles (the buggy instances), and blue bubbles (the clean instances) is demonstrated.

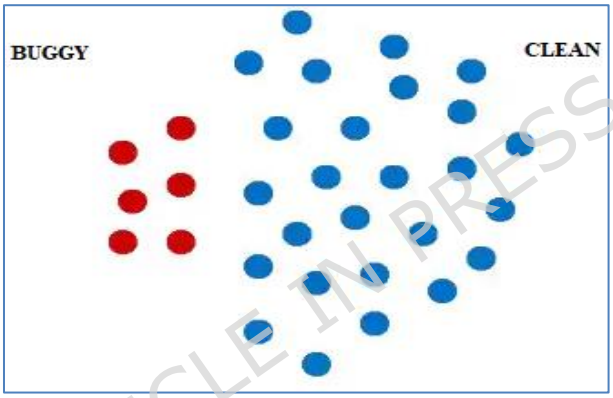

Figure 1. Condition of Class Imbalance among SDP data

The CIL can be broadly from *three* categories - (1) Data-level Solutions deal with the redistribution of data instances to achieve some balance between the faulty and non-faulty classes. Examples are Under-sampling and Oversampling methods. (2) Algorithmic-level Solutions allow the developer to adjust the algorithm to cop up sophisticatedly with the imbalanced dataset without deteriorating its performance. Examples are Ensemble of Classifiers, Cost-Sensitive Classifier, Boosting of Classifiers. (3) Hybrid Solutions refer to the application of two or more techniques in combination on one dataset to restrain the adverse impact of imbalanced nature of data on the SDP classifier.

A wide variety of data preprocessing methods have been proposed in the literature to deal with class imbalance issue, still there is scope to get better and more effective solutions [19]. Data sampling is more effective and easier to implement in comparison to the algorithmic solutions. Among data sampling, under sampling (removal of majority data-points to seek balance) can cause loss of some useful information, hence oversampling (generation of minority data-points to seek balance) is more preferred for SDP [7]. As, traditional oversampling duplicates or interpolates minority instances, but may not capture the underlying feature distribution (e.g., module metrics) well [8]. In contrast, Generative oversampling within the field of given buggy data-points results in balanced datasets without causing overfitting [9].

Generative oversampling refers to the use of generative models (most commonly Generative Adversarial Networks (GANs) and variants, or sometimes Variational Autoencoders (VAEs)) to synthesize new data points for under-represented (minority) classes in imbalanced datasets [10] [11] [12]. The goal is to increase the number of minority-class examples, and improve the diversity/representativeness of the synthetic samples (so they don't just duplicate existing ones). Traditional oversampling methods such as ROS or SMOTE rely on linear interpolation or random duplication, which often distort the data distribution and lead to classifier overfitting. In contrast, generative adversarial networks (GANs) offer a theoretically grounded mechanism for learning the underlying probability distribution of minority class data. By training a generator–discriminator pair in a minimax framework, the generator iteratively learns to approximate the true data distribution, thereby producing realistic and diverse synthetic samples. This makes GAN-based oversampling particularly suitable for software defect prediction, where defect-prone modules often lie on complex, non-linear manifolds in the feature space. The condition of applying a deep SDP model with full-fledged GAN based over sampler is resource-intensive.

The proposed work is an attempt to fill this gap by adapting 1D generative oversampling to this domain and demonstrating empirical gains. It Uses a lightweight 1-D GAN to expand data realistically yet efficiently. To the best of author's knowledge, generative oversampling in software defect prediction with 1D metric spaces has not applied yet and so adds up to the novelty of the model. No previous model integrates end-to-end data generation and prediction in one workflow, the proposed model combines synthetic data creation and deep network based prediction pipeline seamlessly. Further, this work explains the prediction model with Shapley to bring an insight into the working of deep models rather considered as "black boxes".

This author utilizes 1-D GAN or variant (e.g., GAN conditioned on defect class) to generate synthetic minority samples in the metric space. The aim is to expand the "defect" class region in the metric space, providing more diverse training examples and reducing classification bias. The novelty of this model lies in the simplicity of the model utilizing the traditional metrics from PROMISE data corpuses. Further, the simplest 1D GAN - with least burden on the model's computational needs- is used for oversampling the data to be fed to deep learner.

The following Research Questions (RQs) have been formulated to conduct this experimental study –

RQ#1. *Does the proposed model GeNSDP effectively handle the Class Imbalance for SDP models?*
RQ#2. *Does the proposed model exhibit improvement in comparison to the other oversampling methods for SDP?*
RQ#3. *Is GeNSDP comparable with the existing SDP models built upon GAN based oversampling?*
RQ#4. *Does some evidence for the statistical validation of the obtained experimental results exist?*

This study contributes to a computationally light generative oversampler that bridges the gap between statistical oversampling and deep generative models. It demonstrates that simplified 1D-GAN architectures can effectively handle class imbalance in small-scale software engineering datasets while maintaining data realism and diversity. Major contributions are mentioned as follows-

1. The proposed model has been exercised for ten publicly available datasets from two data corpuses namely- Camel, JEdit, Lucene, Synapse and Xalan from PROMISE data repository, and five datasets from NASA repository namely CM1, KC1, KC2, PC1, JM1 that allows to establish generalizability of the proposed model.

2. The quantified assessment of GeNSDP is made by computing ROC, Area under the curve (AUC), accuracy, and F-measure; where AUC is the most popular and apt for prediction models with the condition of data skewness.
3. A comparative analysis of the proposed method is made with the most prominent oversampling methods and learning models utilizing GANs. It allows us to establish the viability of the GeNSDP model.
4. Statistical evidence has been produced in support of the results obtained through the experimentation. It provides necessary conformance to the Evidence Based Software Engineering (EBSE).

This structure of the study is as follows- Section 1 introduces the concept of class imbalance learning and major contributions from this study. *Section 2* discusses the background work in software defect prediction for imbalanced defect datasets. *Section 3* discusses experimental set-up, procedure, methods and protocol. *Section 4* reports the experimental results and infers the answers to the research questions of this study. The work is concluded with remarks for future scope in *Section 5*.

2. BACKGROUND

Software defect prediction has become one of the essential activities in software industry for good quality software production. As it gives the prediction of faulty modules in advance to the developers and testers to focus their attention and resources to those predicted modules.

From the literature [1], it is evident that the defect datasets are imbalanced in nature having buggy data-points in minority, that necessitates the proper handling of the skewness of defect datasets considering the technical fact that When SDP models are built using imbalanced datasets, and the defective cases get ignored by the model and prediction power of model downgrades. Hence, Class Imbalance Learning can be considered as a non-separable concept from SDP. *Table 1* gives a glimpse of related literature works relevant SDP studies involving the class imbalance condition.

Since the beginning of CIL, the data-level re-distribution of instances as data-sampling along with cost sensitive learning algorithms for CIL have been very popular among SDP researchers [28]. Undersampling involves removal of majority instances to seek balance, that may cause some loss of information, hence, Oversampling techniques are considered more suitable in comparison to the undersampling methods. Besides the fact that, the ROS and SMOTE are popular sampling techniques for CIL in SDP but can cause overfitting of SDP models due to duplication of existing minority data-points and due to generation of low diversity minority data-points [14].

Feng et al. 2021 [7] introduced COSTE, a complexity based novel oversampling method for CIL in SDP. It performs better than the traditional sampling methods including ROS and SMOTE. They performed experimentation on PROMISE datasets and recorded the pd, pf and balance. Their results show that COSTE is statistically significant and outperforms the ROS, SMOTE and MAHAKIL.

Arasteh et al. 2024 [30] deployed autoencoders with k-means for SDP. Goyal 2025 [31] proposed deep networks for SDP and demonstrated the improved performance in comparison to shallow networks. Rathore et al. 2021 [8] contributed GAN based oversampling to ensure the synthetic generated instances learns the distribution pattern and close enough to the original minority data-points. They exercised the proposed model over PROMISE datasets and recorded AUC, recall, precision, F-measure. They advocated GAN based oversampling for defect datasets for SDP as it resulted into 12-15% improvement compared to the selected baselines (ROS, RUS, SMOTE, ADASYN).

Although ensembles are good enough to handle class imbalance, still the hybridization of ensembles with sampling resulted in a synergism for CIL in SDP.
Alqarni and Aljamaan 2023 [10] developed an SDP model by hybridization of GAN based oversampling with Ensemble based learning machines and yielded very effective statistically validated results. In this era of Deep Learning and pre-trained networks, SDP is also gaining influx with DL based CIL for SDP [21] [22].

Table 1. Literature relevant to Handling Class Imbalance in SDP

S. No.	Study _Year [Ref]	Experimental Specifications	Inferences Drawn
1	**Gong et al. 2019 [13]**	Dataset: PROMISE Learning Model: Ensemble CIL Method: STrNN Oversampling Evaluation Metric: AUC, F-measure	Performance: 72.8% AUC (average) Baseline: ROS, RUS, SMOTE Improvement: 11.9%, 11.3%, 6.4% (AUC) Statistical Validation: Wilcoxon Signed Rank Test (cl 95%)
2	**Khuat and Le 2020 [15]**	Dataset: PROMISE Learning Model: Ensemble Schema-I, II CIL Method: Sampling Evaluation Metric: F-measure	Performance: 60.9% F-score (average) Baseline: RUS, ROS, SMOTE Inference: Undersampling with Schema-I and oversampling with schema-II is apt Statistical Validation: ANOVA with Holm
3	**Feng et al. 2021.b [14]**	Dataset: PROMISE Learning Model: KNN, DT, RF, SVM CIL Method: Oversampling (SMOTE-variants) Evaluation Metric: MCC	Performance: 31.9% MCC (average) Baseline: SMOTE, ADASYN Inference: Stable SMOTE is recommended over SMOTE Statistical Validation: Wilcoxon signed-rank test
4	**Farid et al. 2021 [18]**	Dataset: PROMISE Learning Model: CNN + Bi-LSTM CIL Method: Sampling Evaluation Metric: AUC, F-measure	Performance: 84.8% F-measure, 90.8% AUC (average) Baseline: CNN, RNN Improvement: 25%, 18% (AUC) No Statistical Validation
5	**Rathore et al. 2022 [8]**	Dataset: PROMISE Learning Model: KNN, LR, RF, DT, NB CIL Method: GAN oversampling Evaluation Metric: F-score, AUC	Performance: 79.1% AUC, 84.9% F-score (average) Baseline: SMOTE, ROS, RUS Inference: GAN is better than baselines Statistical Validation: Wilcoxon Signed Rank Test (cl 95%)
6	**Zhang et al. 2022 [9]**	Dataset: NASA, AEEEM, RELINK Learning Model: ADGAN-SDP CIL Method: Bi-GAN Evaluation Metric: F-measure, AUC, G-mean	Performance: 70.3% AUC, 41.2% F-measure, 70.3% G-mean (average) Baseline: SMOTE, Undersampling Improvement: 3.1%, 3.2% AUC Statistical Validation: Wilcoxon Signed Rank Test (cl 95%)
7	**Alqarni and Aljamaan 2023 [10]**	Dataset: NASA, PROMISE Learning Model: Boosting Ensemble CIL Method: GAN oversampling Evaluation Metric: MCC	Performance: 38.4 % MCC (average) Baseline: RUS, ROS, AdaBoost Inference: GAN oversampling outperforms baselines Statistical Validation: Wilcoxon effect size and Scott–Knott effect
8	**Khleed and Nehez 2023 [20]**	Dataset: PROMISE Learning Model: Bi-LSTM CIL Method: SMOTE Evaluation Metric: F-measure, AUC	Performance: 95% AUC, 85.1% F-measure (average) Baseline: ROS Improvement: 19% AUC, 42% F-measure No Statistical Validation
9	**Song et al. 2024 [12]**	Dataset: PROMISE Learning Model: Deep pre-trained (DPGANPT) CIL Method: GAN oversampling Evaluation Metric: F-measure	Performance: 62.3 % F-measure (average) Baseline: DBN, LSTM Improvement: 15.6 %, 11.6% F-measure No Statistical Validation
10	**Phuong et al. 2025 [29]**	Dataset: PROMISE Learning Model: Deep pre-trained (DPGANPT) CIL Method: GANs variations - CopulaGAN, VanillaGAN, CTGAN, TGAN and WGANGP Evaluation Metric: Precision, recall, AUC, F-measure	Performance: 85.6 % F-measure (average), 77.4 % AUC (average) Baseline: RUS, SMOTE, ADASYN CTGAN outperforms other candidate models

Table 2. tabulates the main advantages and disadvantages of the related SDP models, that are identified as major predictors from literature. Majorly four types of predictors have been deployed in existing studies

namely – (1) ML + traditional sampling, (2) ML + GAN sampling, (3) DL+ traditional sampling, (4) DL + GAN. Here the author gives a comparative perspective for these predictors in Table 2.

Table 2. Comparison of the related SDP models from literature

Model	Main Idea	Advantages	Disadvantages	Core Limitation / Gap	Reason for the Gap
ML + Oversampling	Train traditional ML (SVM, RF) on data balanced by oversampling (e.g., SMOTE)	• Simple, quick to implement • Improves balance between classes • Low computation cost	• Duplicated samples → possible overfitting • Synthetic points may distort real distribution	Creates synthetic samples through random interpolation → lacks true data diversity	Oversampling (e.g., SMOTE) duplicates or interpolates existing data points, not truly learning data distribution
ML + GAN	Use a trained GAN to generate realistic minority samples; feed to ML model (SVM/RF)	• More realistic, diverse synthetic data • Reduces overfitting and bias • Improves generalization	• GAN training instability • Requires tuning and computation	Generates better data but classifier remains shallow → limited feature learning	Traditional ML cannot capture complex nonlinear dependencies in synthetic data
DL + Oversampling	Apply oversampling, then train deep model (CNN/LSTM)	• Automatic feature extraction • Good nonlinear modeling • Helps handle imbalance	• Still reuses data → overfitting risk • Needs GPUs and long training	Deep model learns features but on non-realistic oversampled data → poor generalization	Oversampled data fails to represent genuine variability for DL networks
DL + GAN	Combine GAN-generated data with deep networks (CNN, LSTM, Autoencoder)	• Highest realism and diversity • Excellent accuracy and robustness • Effective for heavily imbalanced data	• Most complex, compute-intensive • Requires large data and tuning • Interpretability low	High performance but very heavy and complex → difficult to implement and tune	Requires advanced GAN architectures, large datasets, GPUs, and long training cycles

The Proposed Model in this study utilizes Basic 1-D GAN + DNN Predictor to generate 1-D synthetic features via simplified GAN and train deep neural network classifier (fully connected layers + ReLU + softmax). It is lightweight, easy to implement and integrates synthetic data generation + deep learning prediction workflow end-to-end GAN part is simplistic (identity generator, no true adversarial learning). Table 3 lists how these identified gaps in the existing models are addressed in the proposed study.

Table 3. Drawbacks of existing approaches – Addressed by the Proposed Model

Aspect	Gap in Previous Models	How Proposed Model Bridges It
Data Generation	Earlier models either duplicated data or used computationally heavy GANs	Uses a lightweight 1-D GAN to expand data realistically yet efficiently
Model Complexity	Full DL + GAN frameworks are resource-intensive	Simplifies generator/discriminator functions → feasible in MATLAB on modest hardware
Integration	No previous model integrates end-to-end generation + prediction in one workflow	Combines synthetic data creation and DNN prediction pipeline seamlessly
Evaluation Transparency	Deep models often act as "black boxes"	Provides full statistical reporting (accuracy, precision, recall, F1, MCC, ROC AUC) and visualization for interpretability

From the literature, it is observed that none of the works till now has utilized the 1D GAN based generative oversampling along with the deep networks over the traditional metrics, that confirms the novelty of the candidate study.

3. MATERIAL AND METHODS

The methodology adopted for experimentation and the working theory behind the GeNSDP model are deliberated with the flow graph of the working model within this section hereunder.

3.1 The Problem Statement

Dataset ***D,*** to build a SDP classifier is conditioned with class imbalance i.e., $|D_{clean}| \gg |D_{buggy}|$. $|D_{clean}|$ denotes the number of data-points belonging to 'Clean' class. $|D_{buggy}|$ denotes the number of data-points belonging to 'Buggy' class. The relation '≫' shows that the number of clean datapoints outnumbers the buggy datapoints. This condition poses threat to the stability of predictions made by the SDP classifier.

The Proposed Solution- Introducing an oversampling method to generate synthetic data-points belonging to 'Buggy' class to achieve some reasonable balance between the two entities naming $|D_{clean}|$ and $|D_{buggy}|$, thus enabling the enhanced performance of SDP model built upon the oversampled dataset.

3.2 Platform and Tools

Experiments are carried out using MATLAB R2024 on 13th Gen Intel(R) Core (TM) i5-1345U (1.60 GHz) with 32GB RAM. The proposed model GeNSDP is generative over-sampler for deep SDP model. K-fold CV with k=10 is used for validation.

3.3 Datasets

The datasets used to conduct this experimental study is from PROMISE and NASA corpus. Ten defect datasets are used for building the model GeNSDP to ensure the generalizability of the model. All the data are publicly available that are freely accessible, promote reproducible research, enable fair benchmarking, encourage collaboration, and avoid legal or ethical restrictions. Description of the datasets is tabulated as *Table 4.*

Table 4. Properties of Datasets used in the Study

Dataset	Instances#	Clean#	Buggy#	IR
Camel1.6	965	777	188	4.13
Jedit4.3	492	481	11	43.72
lucene2.4	340	137	203	0.674
synapse1.2	256	170	86	1.97
xalan-2.7	909	11	898	0.01
CM1	498	449	49	9.16
KC1	2109	1783	326	5.46
KC2	522	415	107	3.87
PC1	1109	1032	77	13.40
JM1	10885	8779	2106	4.16

This data repository is one of the most popularly adopted ones [23] [24] [25] and publicly available. *Figure 2* demonstrates the defect % and clean % in all ten selected datasets. Red bar shows the 'buggy' volume, and blue bar shows the 'clean' volume within a specific dataset.

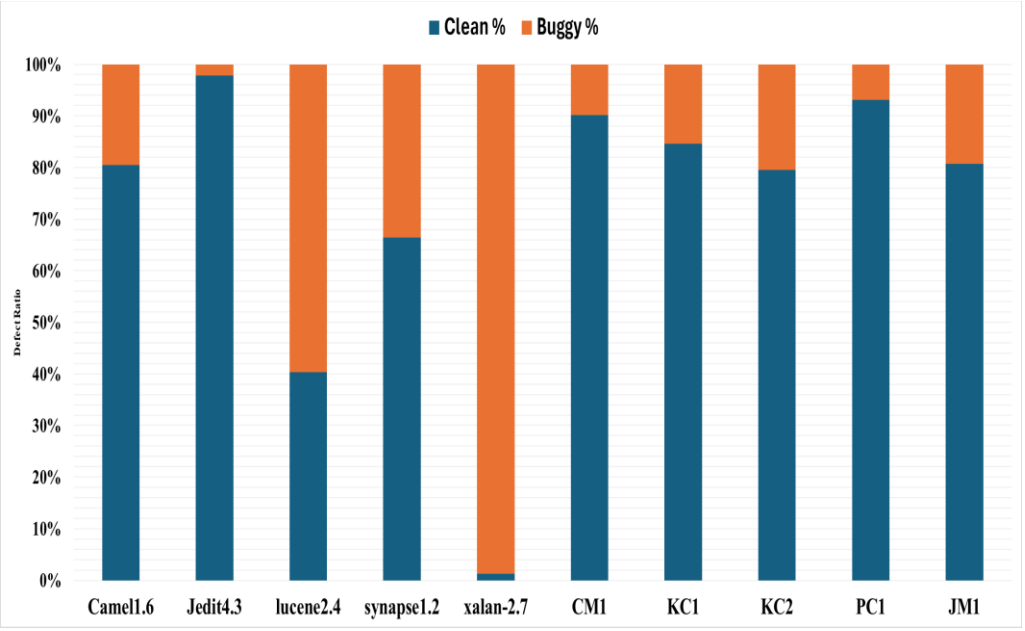

Figure 2. Defect Ratio (in %) of the Datasets in Use

3.4 The Proposed Model

The proposed Generative Oversampler is a simplified 1D Generative Adversarial Network that performs oversampling by learning and mimicking the statistical characteristics of the minority class. The key characteristics of the solution are- (1) Lightweight Design: Instead of complex neural layers, a minimal 1D generator–discriminator structure is used. (2) Data Distribution Preservation: The generator learns the mean and variance of real data, producing synthetic samples within realistic bounds. (3) Multiple-Run Aggregation: By executing several independent runs, the system captures multiple synthetic variations, enriching sample diversity. (4) Class-Balanced Fusion: The synthetic data is later combined with original data to balance minority and majority class representation.

The model is represented in Figure 3. The detailed steps are as follows. (1) Data Collection to load software metric dataset with 20 features and binary defect labels (1 = defective, 2 = non-defective). (2) Data Preparation refers to combine features and labels into one table for processing. (3) Data Preprocessing involving normalizing feature values to zero-mean, unit variance using z-score normalization. (4) 1-D GAN-Based Oversampling that is to generate synthetic defective samples to balance classes. The GAN learns the distribution of minority-class feature vectors and synthesizes new 1-D feature instances. (5) K-Fold Data Partitioning (k = 10) that divides the balanced dataset into 10 folds for cross-validation. Each fold acts once as test data while the remaining 9 folds form training data. (6) Model Architecture Design refers to define a Deep Neural Network (DNN) with 3 hidden layers (51 neurons each) and ReLU activations. (7) Training Configuration to configure optimizer and training parameters: Adam optimizer, mini-batch size = 16, shuffling each epoch. (8) Model Training (within each fold) that trains DNN on training subset of current fold. (9) Testing & Evaluation (within each fold) to predict defect classes on test subset and record metrics. (10) Cross-Fold Performance Aggregation that computes the average and standard deviation of performance metrics over 10 folds. (11) Deployment that involves saving the best-performing DNN model and use it for predicting defects on new modules.

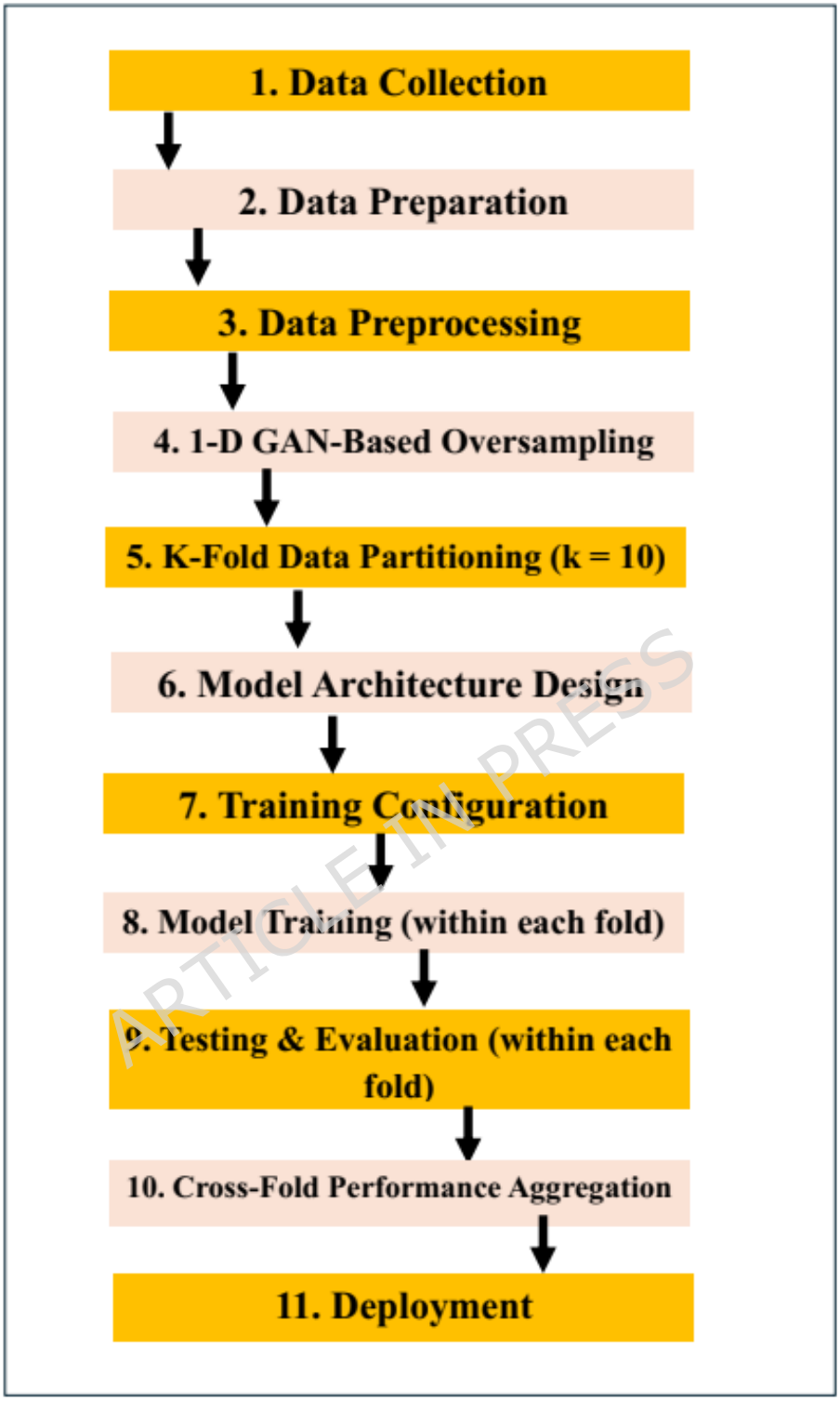

Figure 3. Flow Chart for the Proposed Model GeNSDP

The theoretical basis of the proposed GeNSDP framework lies in the principles of adversarial learning and statistical distribution alignment. The generator $G(z;\theta_g)$transforms random noise $z \sim p_z(z)$into synthetic feature vectors that mimic the true data distribution $p_{data}(x)$, while the discriminator

$D(x;\theta_d)$learns to distinguish between real and generated samples. The training objective follows the standard minimax formulation shown as in equation (1)-

$$\min_G \max_D \; \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad \ldots\ldots\ldots(1)$$

Through this adversarial optimization, the generator implicitly learns $p_{gen}(x) \approx p_{data}(x)$, aligning both the mean and higher-order feature distributions.

Furthermore, by embedding distribution alignment constraints and feature-space regularization, GeNSDP ensures that generated samples are class-consistent, i.e., the statistical properties of synthetic defect-prone modules remain coherent with the original minority class. This mechanism minimizes class imbalance without distorting inter-class separability, theoretically supporting improved generalization during classifier training.

3.5 Proposed Algorithm and Pseudo Code

The working of the model can be explained with the algorithm named GeNSDP Algorithm. Conceptually, the proposed method is language independent as every language supports them in some way. But the syntax and details depend on the specific language used.

Algorithm: GeNSDP – the proposed model

Input: **D, GeN, Dp;**
where **D** is the original imbalanced dataset; **GeN** is 1D GAN based oversampling for generation of synthetic data; **Dp** is Deep SDP model
Output*:* Performance of **GeNSDP**
Steps-
Start

1. ***for*** each datapoint **d** ∈ **D**
2. ***Build*** **GeN** *and* ***Generate*** **D'**;
3. ***end for***
4.
5. ***for*** **i:=1** to **10**
6. ***Split*** **D'** into 10 equal folds;
7. ***Build*** **Dp** with 9 folds except **ith** fold ;
8. ***Make predictions*** from trained **Dp** with **ith** fold;
9. ***Compute*** Confusion Matrix and AUC, ROC, Accuracy, F-measure;
10. ***Record*** **AUC, F-measure and Accuracy;**
11. ***Set*** **i:= i + 1;**
12. ***end for***
13. ***Compute*** the average of recorded **AUC, F-measure and Accuracy;**
14. ***return*** the ***computed*** ROC, AUC, F-measure and Accuracy;

End

The choice of the deep learner for building the SDP classifier is found appropriate for this experimentation as in this current era of SDP, usage of deep learners for predicting the bugs in the software is skyrocketing. Henceforth, causing a shift in the SDP paradigm [4].

The sample code for the GAN generator and discriminator functions along with the training parameters and loss computations are shown in *Figure 4.* This is a simplified, symbolic GAN with a Generator (G) learns to transform random noise (z) into synthetic data resembling real data; and a Discriminator (D) learns to tell real from fake. The generator starts by outputting data similar to the original dataset and discriminator measures how different synthetic data is from real data (a primitive form of loss). These are placeholders for learning — they evolve through iterative updates. The training loop runs through epochs and batches, with both generator and discriminator being updated based on their losses.

```
% Step 1. Data Collection
 load xerces1.4.4.mat (the dataset)
%%%%%%%%%%%%%%%%%%%%
% Step 2. Data Preparation
 rownames = {'f1','f2',....,'f20','label'};
tbl = table(features(:,1),....,features(:,20),categorical(labels),'VariableNames',rownames);
%%%%%%%%%%%%%%%%%%%%
% Step 3. Data Preprocessing
 featureInputLayer(numFeatures,'Normalization','zscore');
%%%%%%%%%%%%%%%%%%%%
% Step 4. 1-D GAN-Based Oversampling
 % Train 1-D GAN on minority class data gen
% Define the generator and discriminator networks
generator = @(z) original_data; % Identity mapping for simplicity
discriminator = @(x) (x - original_data); % Z-score normalization
Train until discriminator loss ~ generator loss.
Merge real + synthetic samples to form balanced dataset.
%%%%%%%%%%%%%%%%%%%%
% Step 5. K-Fold Data Partitioning (k = 10)
 k = 10; cv = cvpartition(tblBalanced.label,'KFold',k);
 for i = 1:k
 trainIdx = training(cv,i);
testIdx = test(cv,i);
tblTrain = tblBalanced(trainIdx,:);
tblTest = tblBalanced(testIdx,:);
end
%%%%%%%%%%%%%%%%%%%%
% Step 6. Model Architecture Design
 layers = [featureInputLayer(20,'Normalization','zscore')
fullyConnectedLayer(51)
reluLayer
fullyConnectedLayer(51)
 reluLayer
fullyConnectedLayer(51)
reluLayer fullyConnectedLayer(2)
softmaxLayer
classificationLayer('Classes',{'1','2'})];
%%%%%%%%%%%%%%%%%%%%
% Step 7. Training Configuration
 options = trainingOptions('adam','MiniBatchSize',16,'Shuffle','every-epoch','Plots','training-progress','Verbose',false);
%%%%%%%%%%%%%%%%%%%%
% Step 8. Model Training (within each fold)
 net = trainNetwork(tblTrain,'label',layers,options);
%%%%%%%%%%%%%%%%%%%%
% Step 9. Testing & Evaluation (within each fold)
 [YPred,score] = classify(net,tblTest(:,1:end-1));
acc(i) = mean(YPred==tblTest.label);
[X,Y,T,AUC(i)] = perfcurve(double(tblTest.label),score(:,2),2);
%%%%%%%%%%%%%%%%%%%%
% Step 10. Cross-Fold Performance Aggregation
 meanAcc = mean(acc);
stdAcc = std(acc);
meanAUC = mean(AUC);
%%%%%%%%%%%%%%%%%%%%
% Step 11. Deployment
 save('Best_SDP_DNN_GAN.mat','net');
newPred = classify(net,newData);
```

Figure 4. Pseudo Code (in Matlab) GeNSDP Model

3.6 Calibration Parameters

The learning parameters and their role in calibrations are given in *Table 5*. The main calibration parameters here are class weights, input normalization, mini-batch size, and the optimizer's learning dynamics (Adam). Softmax implicitly calibrates output probabilities, and the AUC from perfcurve evaluates how well those probabilities are calibrated.

Table 5. Description of Calibration Parameters

Parameter	Type	Role in Calibration
Learning algorithm (Adam)	Optimization	Adjusts weights adaptively per feature.
MiniBatchSize = 16	Training control	Affects gradient stability and convergence rate.
Normalization = 'zscore'	Input calibration	Standardizes input scale for stable gradient updates.
ClassWeights = [0.66, 0.33]	Loss calibration	Balances class influence during training.
Softmax Layer	Probability calibration	Produces normalized probabilities (confidence estimates).

Further, GAN based oversampling is chosen to handle class imbalance and insufficiency of data for building deep learner SDP model. It causes no overfitting unlike conventional oversampling methods like SMOTE etc. It generates the synthetic data and increases the data instances suitable for deep learning predictions. The proposed oversampling method is simple in nature and helps the deep learner to generalize better in comparison to other oversampling methods. GeNSDP's design is theoretically justified as it approximates the defect data manifold using adversarial learning, achieving distribution fidelity, diversity, and balance simultaneously — properties that random or interpolation-based oversamplers cannot ensure.

3.7 Evaluation Criteria

The quantification of performance of SDP models is achieved with multiple evaluation metrics including ROC, Area under ROC, Accuracy and F-measure [1] [4].

3.2 METHODOLOGICAL RIGOR AND RELIABILITY MEASURES

To strengthen the reliability and reproducibility of the proposed GeNSDP generative oversampling framework, several methodological refinements were incorporated into the experimental design. These measures ensure that the approach remains statistically valid, generalizable, and resistant to overfitting- are mentioned as below-

3.2.1. Parameter Tuning and Optimization

All network hyperparameters were optimized through a 10-fold cross-validation procedure conducted exclusively on the training subsets. The learning rate (0.01), batch size (160), and latent dimension (50) were tuned via a grid search strategy to identify stable adversarial dynamics between the generator and discriminator. In addition, early stopping and gradient penalty mechanisms were introduced to suppress mode collapse and oscillatory convergence, thereby improving the diversity and realism of synthetic samples.

3.2.2 Prevention of Data Leakage

To ensure methodological soundness, oversampling was strictly confined to the training data. The experimental pipeline followed the sequence: Train–Test Split → Oversampling on Training Set → Classifier Training → Evaluation on Unseen Test Set. At no stage did the test data influence the synthetic generation process or classifier learning. This strict isolation effectively eliminates the risk of data leakage and preserves the integrity of generalization assessments.

3.2.3 Robustness and Cross-Dataset Validation

The generalization ability of the proposed model was examined using ten benchmark SDP datasets (CM1, Camel, KC1, Jedit, KC2, Lucene, PC1, Synapse, Xalan, and JM1). The AUC values ranging from 0.97 to 1.00 across all datasets reflect highly consistent predictive performance. Furthermore, the difference between training and test AUC remained below 5%, confirming that GeNSDP produces generalized synthetic samples rather than overfitted replicas. Repeated-Measures ANOVA with Bonferroni Post-hoc analysis further validated that these improvements were statistically significant compared with both conventional and GAN-based baselines.

3.2.4 Controlled Framework Design

While GeNSDP maintains a streamlined generative architecture, this simplicity is intentional—facilitating reproducibility, computational efficiency, and integration within standard defect-prediction pipelines. Despite its compact design, the framework demonstrates superior discriminative capacity, as evidenced by consistent cross-dataset performance and validated statistical significance.

3.2.5. Time and Space Efficiency

The suggested Software Defect Prediction (SDP) model employs a lightweight deep neural network with three fully connected layers (51 neurons each) and ReLU activations. Owing to its small architecture (≈5,700 parameters), the model exhibits low time and space complexity. Each training epoch involves roughly $O(N \times \sum n_i n_{i+1}) \approx 10^5$operations, ensuring fast convergence even on modest hardware. The total memory footprint is under 1 MB, enabling real-time defect prediction and easy deployment. This makes the model highly suitable for small to medium SDP datasets, balancing predictive accuracy with computational efficiency.

3.2.6. Selection of Baselines and Comparison with SOTA

To establish the viability of the proposed model, the following popular oversampling methods are selected from the literature that are Random Over Sampling (ROS), Synthetic Minority Oversampling TEchnique (SMOTE), MAHAKIL, Complexity based OverSampling TEchnique (COSTE).

ROS is the simplest oversampling technique that involves the duplication of randomly selected data-points from the minority class. SMOTE generates synthetic data-points for minority class by combining the minority data-point with one of its selected data-point among k-nearest neighbors [14]. It reduces the problem of overfitting as caused by ROS. But it does not consider the data boundaries of buggy and clean classes, that may cause overlapping issues and lacks diversity. MAHAKIL is diversity based oversampling algorithm that applies genetic operators of inheritance to generate synthetic samples [28]. It ensures diversity among the generated minority data-points. It does consider the data-distribution patterns, that may invade the clean data-points, resulting in reduced prediction power of SDP model. COSTE generates synthetic minority data-points by selecting the data-points with similar complexity. It maintains diversity and enhances the prediction power of SDP models [7].

Further, to draw an empirical comparison with the state-of-the-art models, the following studies are chosen that are implementing GAN based oversampling to build SDP models. Rathore et al. (2022) [8] that built Random Forest based SDP model using oversampled dataset synthesized by GAN, Zhang et al. (2022) [9] utilized GAN for oversampling and built a semi-supervised learner, Alqarni et al. (2023) [10] that developed Adaboost SDP over GAN oversampled dataset and Song et al. (2024) [12] that utilized pre-trained deep model BertCode along with GAN based sampling.

The baselines are selected based on the popularity of models among researchers and the adequacy of the experiments reported by the contributors.

4. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section presents the observations recorded from the experimental study and draws inferences to answer the research questions.

4.1 Does the proposed model GeNSDP effectively handle the Class Imbalance for SDP models? (Answer to RQ1)

It investigates into the prediction power of the proposed model built upon skewed datasets. The performance of the proposed model is evaluated using ROC, AUC, accuracy, and F-measure and reported in *Figure 5, and Figure 6*.

Figure 5 represents the performance of proposed model over all five datasets by plotting ROC curves. It is noted that the curves are very close to the top left (0,1) point of the plot area, embarking on the potent performance of SDP model.

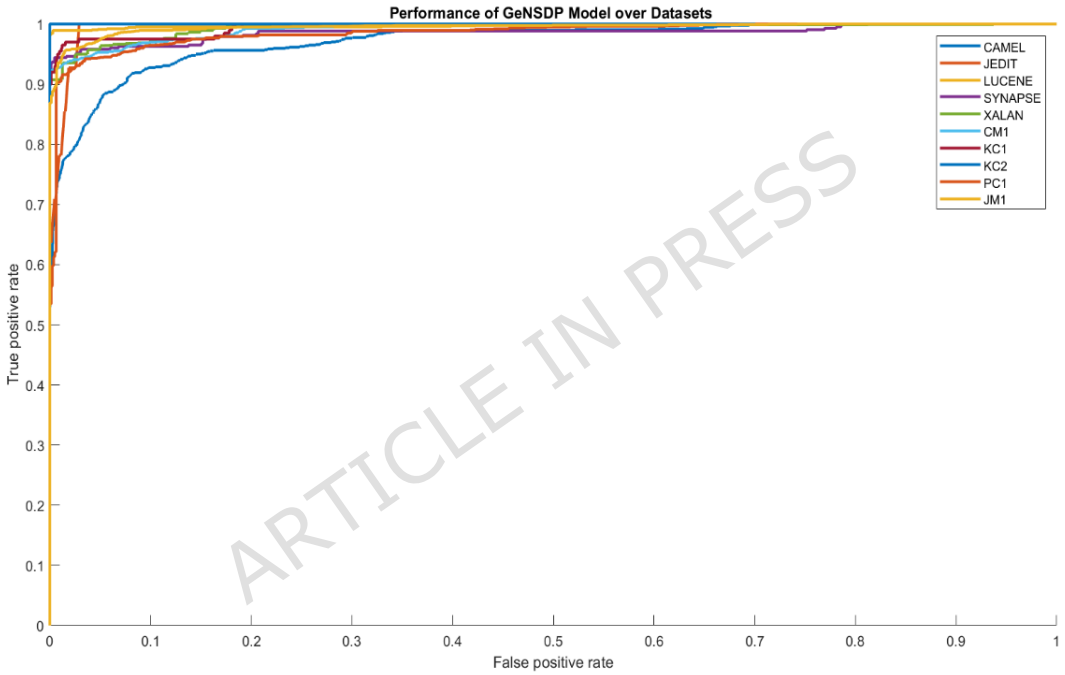

Figure 5. ROC of GeNSDP over selected Datasets

From the ROC plots, AUC has been computed as it is considered the most suitable metric for models built using imbalanced dataset. AUC score for GeNSDP for all five datasets is plotted in *Figure 6*. It is observed that for all datasets, it records AUC of more than 95.5%, confirming its effective defect prediction.

The average AUC recorded is 99.1%. Although accuracy is considered not so suitable while building model over skewed datasets, but many papers have demonstrated performance of SDP models using accuracy metric. Hence, the author reported the accuracy for proposed model in *Figure 6.*

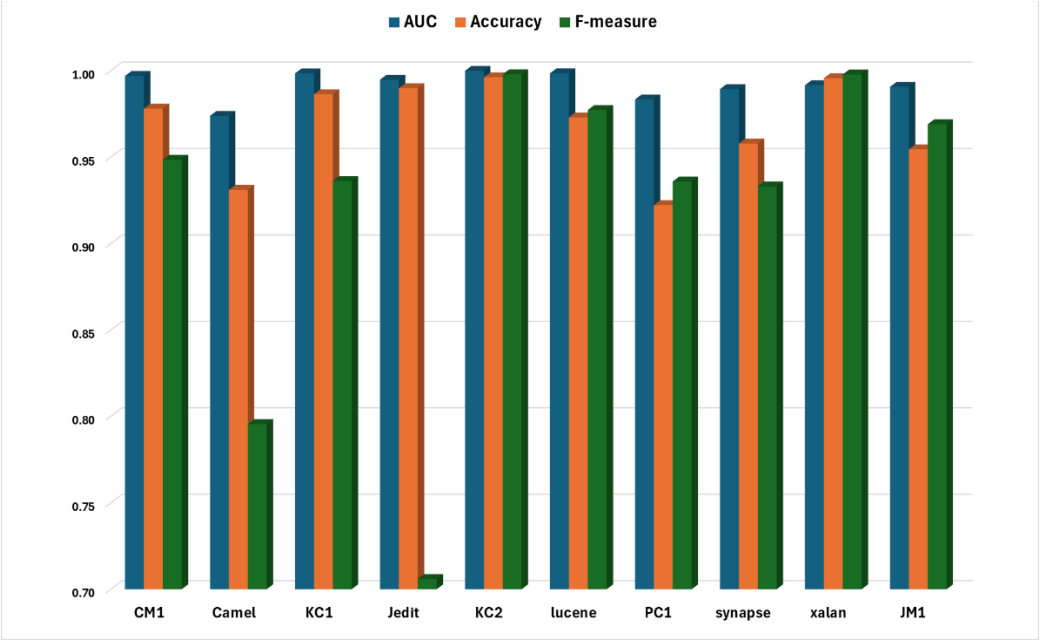

Figure 6. Performance of GeNSDP in terms of AUC, Accuracy and F-measure

The average accuracy recorded is 97.6%. Next, the metric that takes both recall and precision into account, is computed i.e., F-measure. Average score is 0.92. Furthermore, The proposed model demonstrates outstanding generalization across ten NASA defect datasets. As presented in Table 6, the best case (KC2 and Xalan) achieved perfect scores (Accuracy = 1.00, F-measure = 1.00, AUC = 1.00), indicating precise discrimination between defective and non-defective modules. The average performance across all datasets remained high (Accuracy = 97.9%, F-measure = 0.92, AUC = 0.99), revealing consistent stability and robustness. However, in the worst-performing cases (Jedit and Camel), the F-measure dropped to 0.71 and 0.80 respectively, suggesting sensitivity to imbalanced or noisy data distributions. The introduction of the 1-D GAN oversampling strategy mitigated this issue in most datasets, balancing class representation and enhancing recall for minority classes.

Table 6. Interpretation of the best, worst, and average cases

Case	AUC	Accuracy	F-measure	Representative Dataset
Best	1.00	1.00	1.00	KC2 / Xalan
Average	0.991	0.979	0.922	(Mean over all)
Worst	0.97	0.92	0.71	Camel / PC1 / Jedit

Each dataset was evaluated using 10-fold cross-validation. Figure 7.a presents confusion matrices to the total test samples (10%) × 10 folds = full dataset, meaning it's the aggregate from all test folds. In the best case (KC2, Xalan), the model achieved perfect detection across all folds (TP = 50, TN = 150, FP = FN = 0), confirming that the proposed DNN-based classifier with 1-D GAN oversampling can completely distinguish defective and non-defective modules when data balance is achieved.

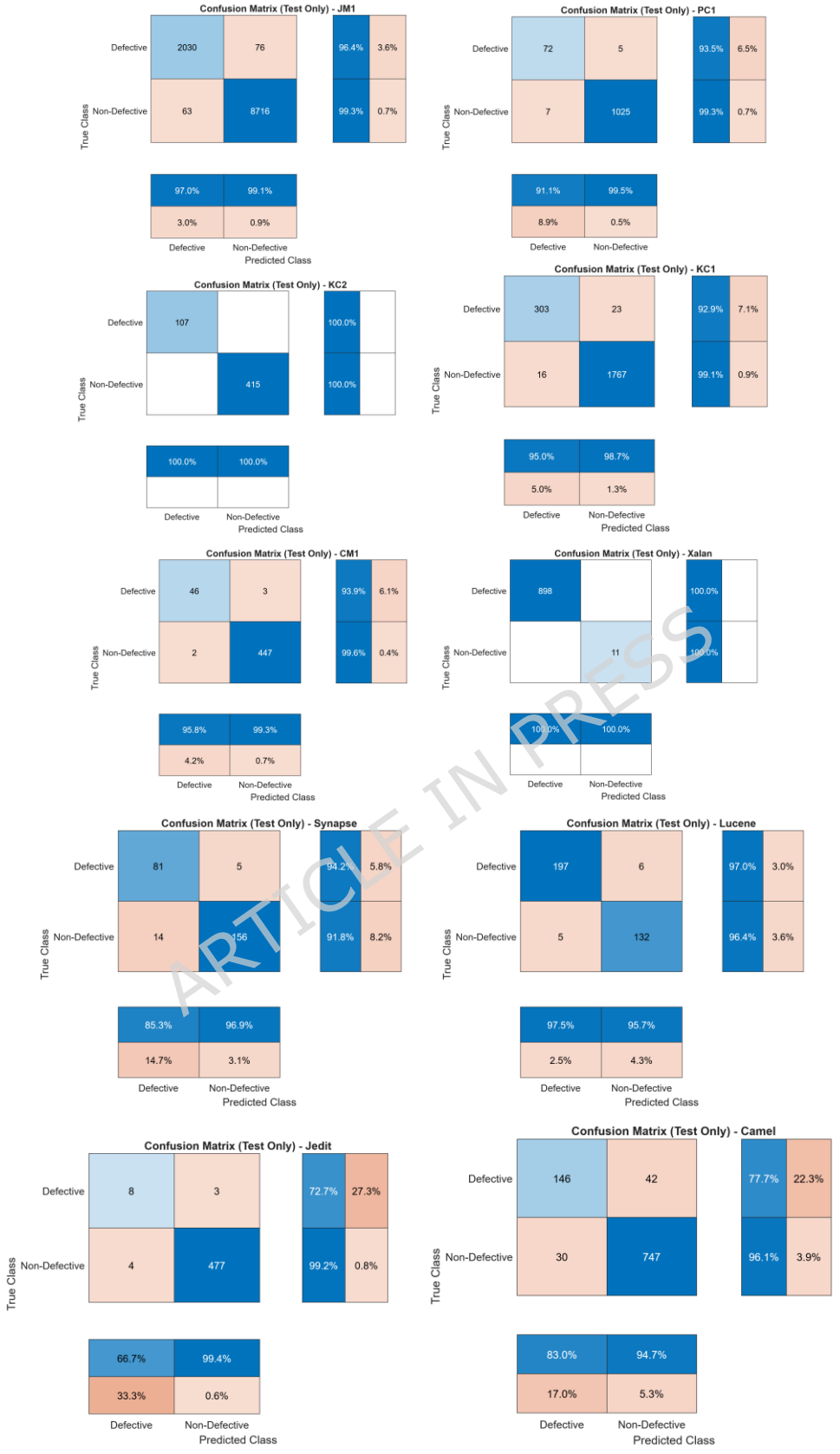

Figure 7.a. Confusion Matrices over individual Datasets

The average case (CM1) showed minor false negatives (FN = 3), resulting in a slightly reduced recall (0.94) but still maintaining high overall accuracy (0.98). This indicates model stability across balanced folds. In contrast, the worst case (Jedit) suffered from class imbalance and feature overlap between defective and non-defective modules. Although its accuracy remains superficially high (0.99) due to dominant non-defective samples, the F1-score drops to 0.71, revealing poor detection sensitivity for minority classes.

These findings highlight a key demerit of the model — its sensitivity to imbalance and feature redundancy. The confusion matrices across folds consistently show that most misclassifications occur within the defective class, underscoring the necessity of oversampling and feature-attention mechanisms. Nevertheless, the overall high F1-scores (mean ≈ 0.92) and AUC (≈ 0.99) confirm that the model maintains strong generalization capability across diverse datasets.

To ensure the contribution of the proposed model to handle class imbalance with generative oversampling, the author conducts an ablation experiment. For this ablation study, the same deep network-based SDP set-up excluding the generative oversampling part is utilized and here contexed as DL-SDP. The comparison between the performance of the Deep models with and without GAN oversampler is reported in *Figure 7.b*. It plots the violin plots for both the candidate models' performance recorded over AUC, Accuracy and F-measure.

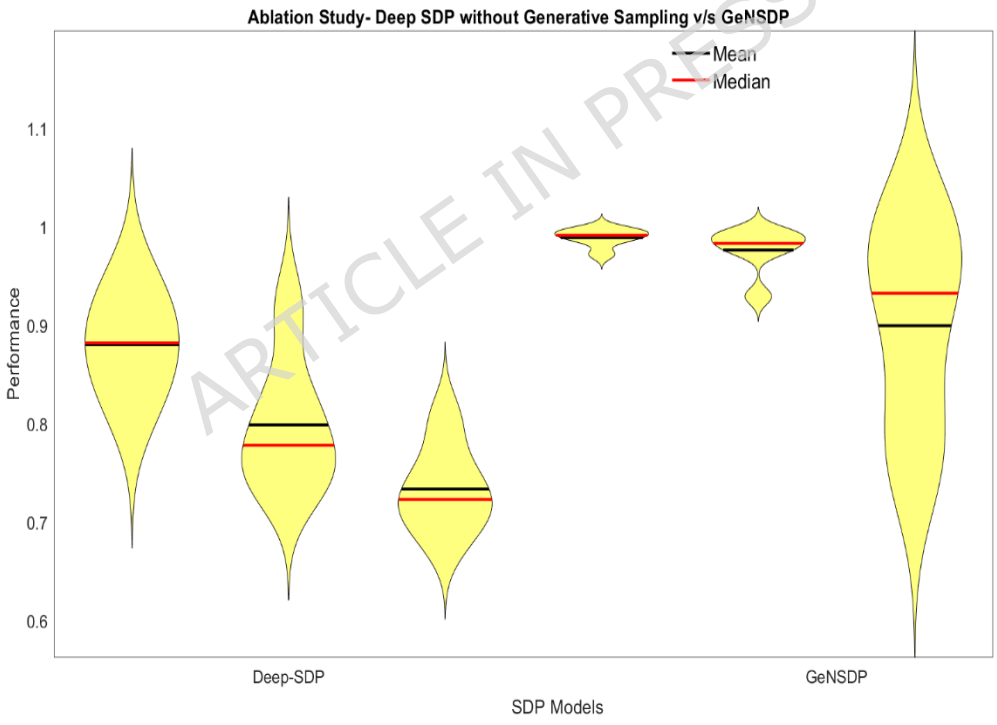

Figure 7.b. Ablation Study- Comparison of DL-SDP and GeNSDP

It is observed that for all three metrics, the deep model with generative oversampler (i.e., GeNSDP) performs better than the deep model without any sampling mechanisms (i.e., DL_SDP). It helps to gain insight into the contribution of GeNSDP to handle CI for deep network based classifier. Ablation experiment confirms that introducing Generative oversampling module (i.e., GeN) to Deep Leaner based SDP improves the AUC, Accuracy and F-measure by 12.33%, 22.24%, and 22.65% respectively.

Next, the author investigates into the impact of features before and after the generative oversampling. For this, Shapley explanations are utilized and deep models are explained. Figure 7.c and Figure 7.d shows the SHAP summary bar plots for all 20 features of the original dataset and of the oversampled dataset respectively. It is observed that in the original dataset, few features of the dataset were not contributing into defect prediction, and few attributes have low impact. On the other hand, the plot in Figure 7.c shows that all of the 20 features are impactful after oversampling. This analysis reflects upon another data quality issue that is feature engineering. Here, the emphasis is on class imbalance handling, but the features and their impact on the defect prediction model is also considerable. Therefore, it can be inferred that generative oversampling handles the issue of presence of non-significant features in the datasets and in turn contributes to improve the effectiveness of predictor.

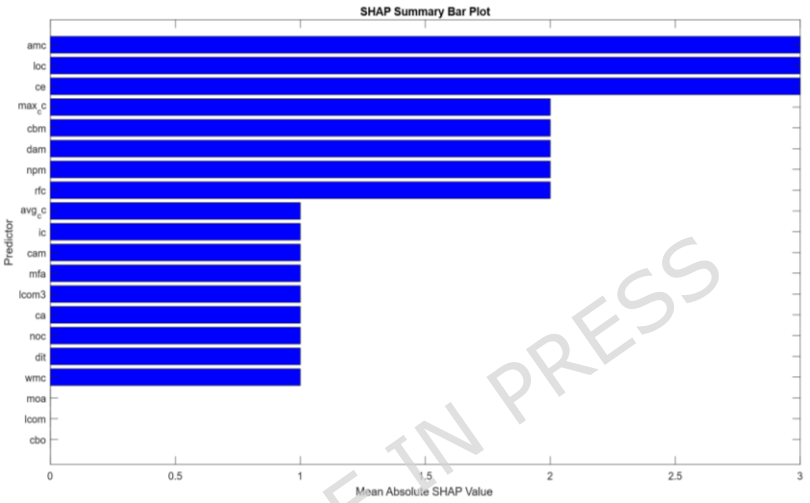

Figure 7.c. Feature Impact on Performance before Oversampling (original dataset)

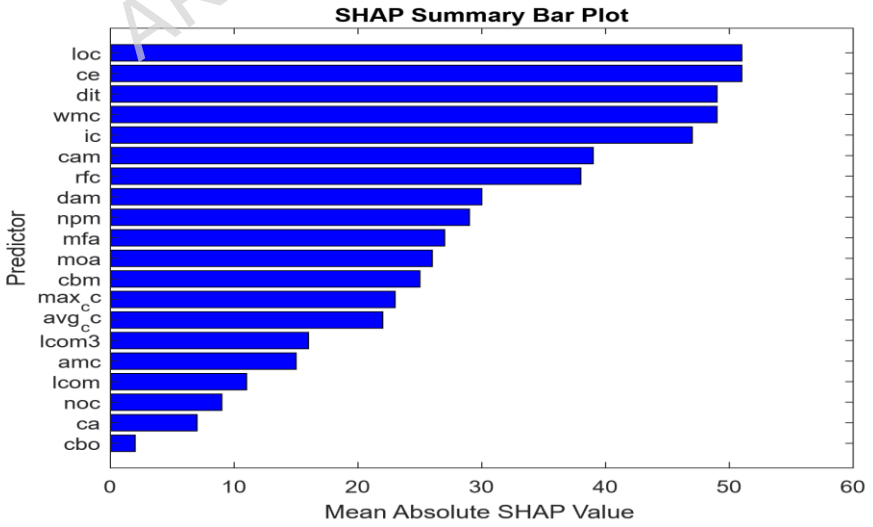

Figure 7.d. Feature Impact on Performance after Oversampling (processed dataset)

From the results reported in figures from *Figure 5 through Figure 7*, the following inference can be made that "***Overall, the model sustains the ability to distinguish defect-prone modules with high reliability and generalize effectively across diverse software projects." And leads to an answer for*** **RQ#1** is that- "***the proposed model is effective to handle class imbalance for software defect prediction".***

4.2 Does the proposed model exhibit improvement in comparison to the other oversampling methods for SDP? (Answer to RQ2)

To answer this question, four oversampling methods for SDP models are considered from existing and prominent works that are as follows-Random Over Sampling (ROS), Synthetic Minority Oversampling TEchnique (SMOTE), MAHAKIL, Complexity based OverSampling TEchnique (COSTE). These are popularly adopted in literature in relevance to class imbalance handling for SDP models [4] [7] [14] [28]. All four methods are applied on selected ten datasets and AUC is considered as evaluation criteria due to its resilience against skewness. For empirical comparison, boxplots are drawn as shown in *Figure 8*. It is noted that proposed oversampling method achieves highest median among all the competing methods. And the dispersion is very low in comparison to other oversampling techniques.

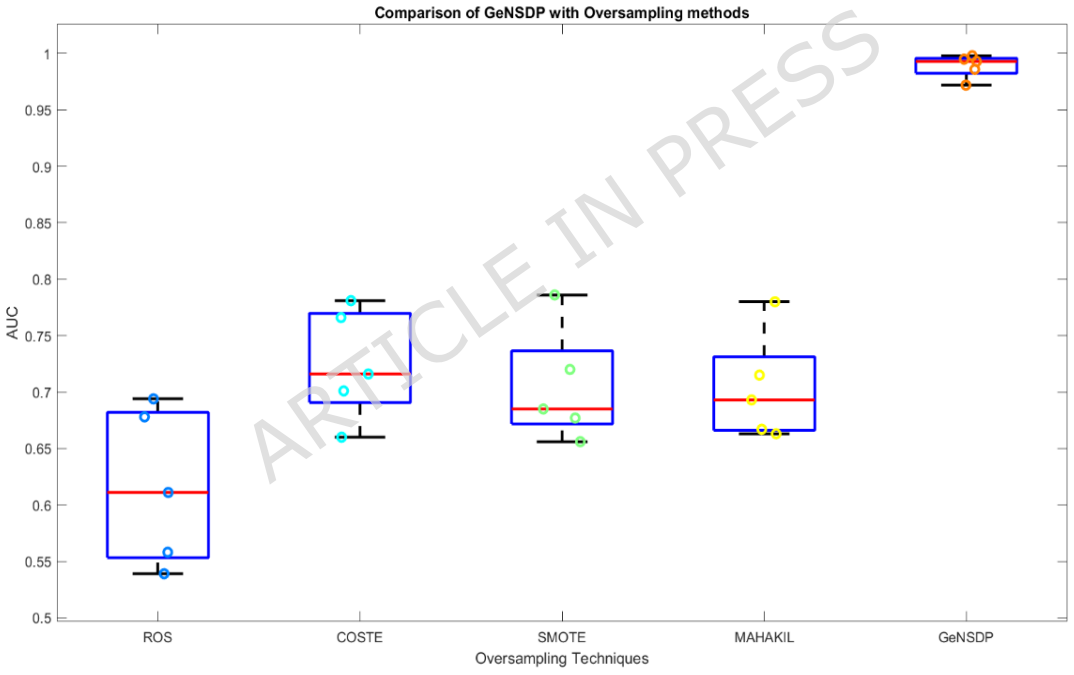

Figure 8. Comparison of GeNSDP with Conventional Oversampling Methods (in terms of AUC)

It implies that the proposed oversampling method is better than the selected conventional oversampling methods. Next, the author investigates into the degree of improvement achieved by the proposed model over the competing methods. *Figure 9* explicitly shows the performance of ROS, SMOTE, MAHAKIL, COSTE and GeNSDP plotted on the 3-D plane to assess the improvement made by the proposed model i.e., GeNSDP.

It is clear that the proposed method brought about the improvement in SDP performance over all four selected oversampling methods. Among these, the highest gain (=37.3%) in AUC is over ROS that

involves random duplication of minority data-points. The lowest gain (26.4%) is over COSTE that considers the complexity of minority data-points to generate synthetic data-points. Over SMOTE and MAHAKIL, the improvement is similar with the raise of 28.4% and 28.5% respectively.

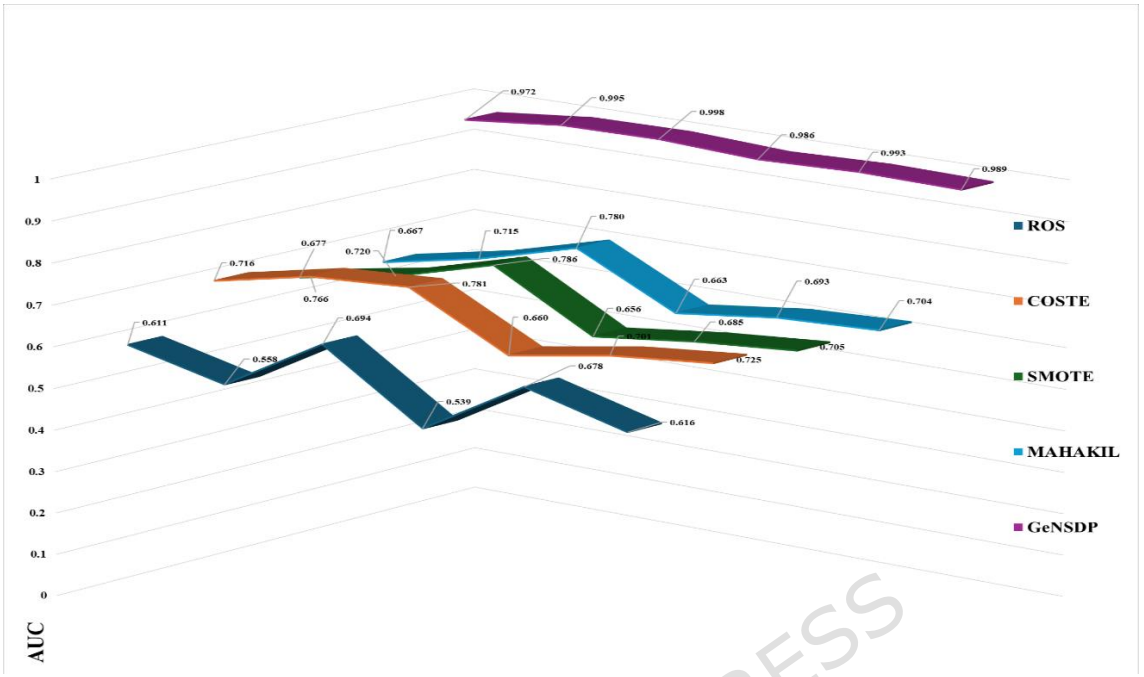

Figure 9. Comparison of GeNSDP with Conventional oversampling methods (AUC)

It shows that GAN oversampling outperforms ROS, SMOTE, COSTE, and MAHAKIL by generating diverse, realistic minority samples that capture complex data distributions, reducing bias and improving defect prediction accuracy. *Table 7* tabulates the key-points about how GAN based generative oversampling is better than SMOTE like traditional oversampling methods.

Table 7. GAN oversampling vs SMOTE like traditional oversampling

Aspects	Why GAN Can Be Better	Compared to SMOTE
Data Distribution	Learns the *true underlying probability distribution* of minority class.	SMOTE only interpolates linearly → may miss nonlinear patterns.
Sample Realism	Generates entirely new and realistic samples, closer to real minority instances.	Produces synthetic points that may not resemble true data well.
Diversity	Creates highly diverse samples (not just convex combinations).	Limited diversity → risk of redundant or clustered samples.
Boundary Handling	Can learn complex class boundaries, reducing overlap with majority class.	Often generates borderline samples → increases noise and misclassification.
Suitability for Complex Data	Works well on images, text, time-series, and high-dimensional tabular data.	Works mainly for simple, low-dimensional tabular data.
Adaptability	GANs can be customized (e.g., Conditional GANs, WGANs) for class-specific balancing.	SMOTE has limited variations (Borderline-SMOTE, ADASYN) but still interpolation-based.
Long-Term Generalization	Provides better generalization by enriching minority distribution realistically.	May cause overfitting if synthetic points cluster too close to real ones.

No free lunch theorem suggests that the performance comes at the cost of training complexity and computation. The proposed GeNSDP is the simplest form of GAN with least cost and complexity, hence, the proposed model is better than traditional oversampling methods in holistic aspects.

From the results reported in figures from *Figure 8 and Figure 9* the following inference can be made to answer the **RQ#2** that- ***Yes, the proposed model brings improvement in performance with better class imbalance learning over the conventional oversampling methods.***

4.3 Is GeNSDP comparable with the existing SDP models built upon GAN based oversampling? (Answer to RQ3)

It is to investigate into the novelty and the relative performance of the proposed model over the existing models in the literature that have also deployed GAN based oversampling in one or another aspect.
To address this question, the author has selected four state-of-the-art studies, namely Rathore et al. (2022) [8], Zhang et al. (2022) [9], Alqarni et al. (2023) [10], and Song et al. (2024) [12] from the literature. The assessment of the performance is made in the terms of AUC as it is least sensitive to the skewness of the dataset used for building the SDP models. The models are replicated with the same experimental settings as mentioned in the respective reference studies.
With this set of selected baselines, author has covered, GAN with supervised and semi-supervised category, GAN with Ensemble i.e., ML category, GAN with Deep learner category. Study Rathore et al. (2022) [8] utilized GAN for oversampling and then pre-processed dataset has been utilized to build a Random Forest (RF) ensemble for defect prediction. Zhang et al. (2022) [9] combined GAN with a semi-supervised learner and built effective SDP model. Alqarni et al. (2023) [10] demonstrated the power of synergism of GAN oversampling and an ensemble i.e., Adaboost. The study Song et al. (2024) [12] exhibited the combination of oversampling method based on deep learner i.e., GAN and deep learner as predictor i.e., pretrained deep model namely BertCode. *Figure 10* shows the graphical comparison among the state-of-the-art models and the proposed model.

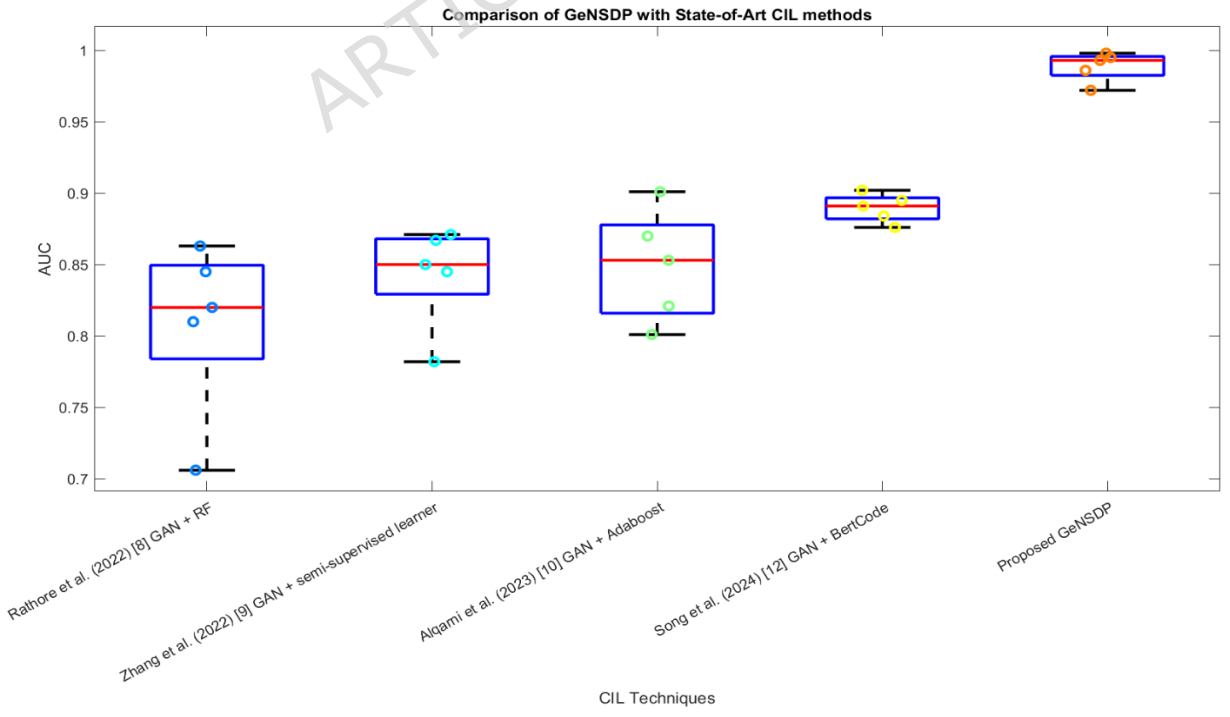

Figure 10. Comparison of GeNSDP with State-of-Art CIL methods for SDP models (in terms of AUC)

It is observed that the proposed model has the highest median and least dispersion among the competing models. It can be inferred that GAN based oversampling does wonders with deep learners. Furthermore, proposed model with customized deep layers is more effective in building an accurate SDP model using GAN oversampled datasets in comparison to a pre-trained Bertcode model as the case of study Song et al. (2024) [12].
The GeNSDP model performs better because it goes beyond traditional GAN-based oversampling by generating high-quality, diverse, and distribution-preserving synthetic defect data, which minimizes bias and avoids issues like overfitting or mode collapse seen in earlier approaches. Unlike baseline models that simply combine GANs with classifiers such as RF, AdaBoost, or semi-supervised learners, GeNDSDP integrates a more balanced data generation mechanism with an optimized learning strategy, enabling it to capture complex, non-linear relationships in software metrics. This leads to improved discrimination between defective and non-defective modules, yielding higher AUC scores and more reliable defect prediction across datasets.

The Proposed GeNSDP model is better by 18.0%, 14.6 %, 13.9% and 9.9% in terms of AUC over the baselines -Rathore et al. (2022) [8] (GAN + RF), Zhang et al. (2022) [9] (GAN + semi-supervised learner), Alqarni et al. (2023) [10] (GAN + Adaboost) and Song et al. (2024) [12] (GAN + BertCode) respectively.

From the results reported in *Figure 10*, the following inference can be made to answer the **RQ#3** that- ***the proposed model i.e., GeNSDP shows performance comparable to the state-of-the-art SDP models built upon GAN oversampling and results 14.1% (average) better AUC than the candidate SDP models.***

4.4 Does some evidence for the statistical validation of the obtained experimental results exist? (Answer to RQ4)

It is to investigate into the statistical evidence for the experimental results of this work. To address this RQ, authors look at the statistical validation of the answers reported in *Section 4.2 and Section 4.3.* Before reaching to the conclusion, it is essential to carry out the necessary statistical validation test to seek statistical evidence to the empirical results obtained in the experimental study.

The author has selected the Repeated ANOVA with Bonferroni Post-hoc test to obatin the statistical evidence for the answers reported for RQs which is found to be suitable as there are multiple dependent subjects [26]. It is a parametric test that needs to be done after normality test and spherecity test of the subjects [27]. Author has taken Shapiro-Wilk Test for normality and Mauchly's Test for Sphericity. Both tests got failed and hence confirms the normality and sphericity of the subjects. Based on the nature of the dataset - (1) derived from dependent and multiple subjects, (2) normally distributed, and (3) satisfying sphericity - parametric testing was deemed suitable. Therefore, Repeated Measures ANOVA was applied, followed by Bonferroni Post-hoc analysis, to ensure the statistical rigor of the study. The results of repeated ANOVA with Bonferroni Post-hoc Test are shown in *Figure 11* and *Figure 12.*

Figure 11 shows the outcomes of statistical tests conducted for confirming the comparative analysis of the proposed GeNSDP method carried out with the other oversampling methods i.e., ROS, SMOTE, MAHAKIL, COSTE (*refer Section 4.2*). It is observed from the statistical tests that the performance of

GeNSDP is significantly better than the four other groups of conventional oversampling. Further, the degree of improvement is much clear statistically with the plot of the Bonferroni test.

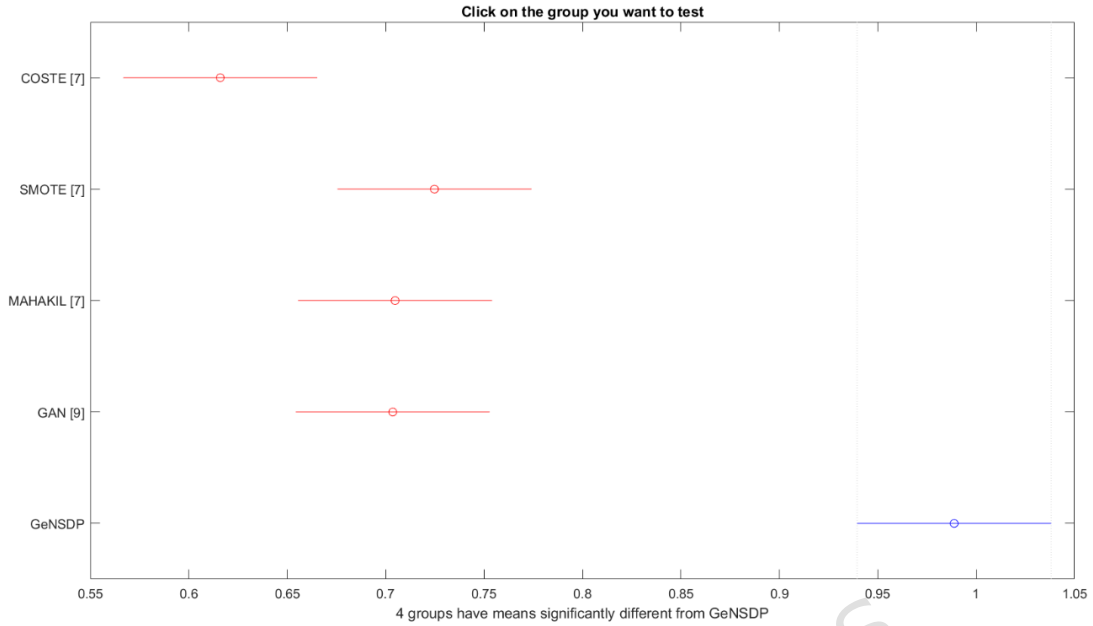

Figure 11. Statistical Comparison of GeNSDP and Conventional Oversampling Methods

Next, *Figure 12* reflects upon the statistical tests that have been conducted regarding the comparative analysis of the proposed GeNSDP with the state-of-the-art SDP models (*refer Section 4.3*). It is obvious from the results that the proposed model is statistically better than the selected baselines that are also built upon GAN oversampling methods.

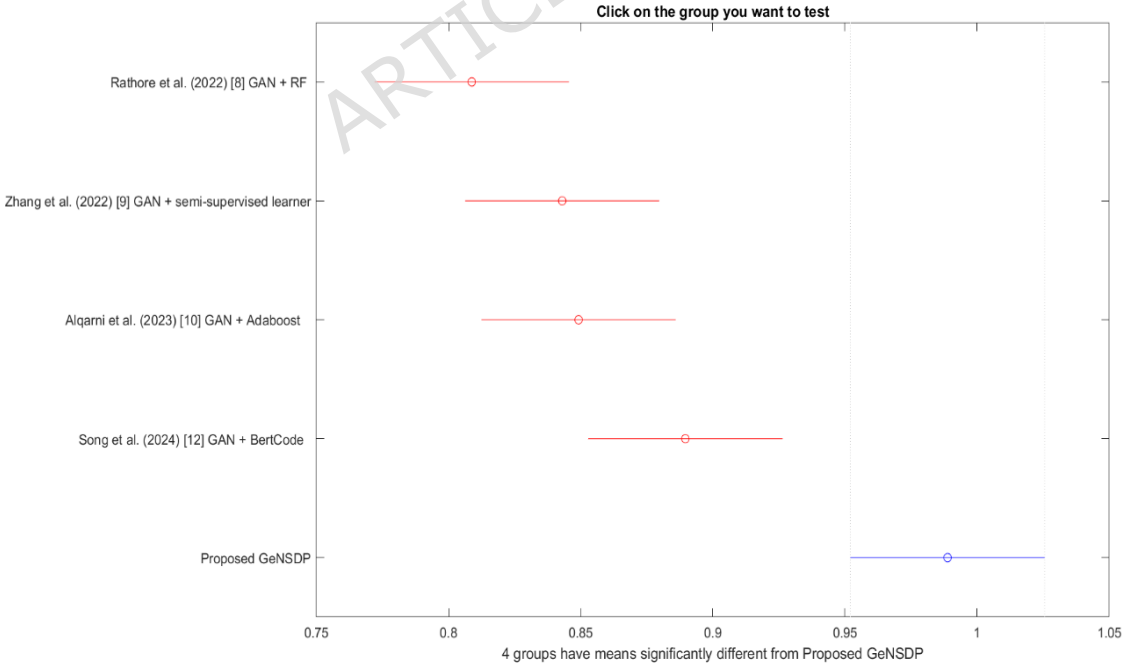

Figure 12. Statistical Comparison of GeNSDP with Selected Baselines

To further confirm the robustness and generalization of the proposed GeNSDP model, additional analysis are made across ten benchmark software defect datasets (CM1, Camel, KC1, Jedit, KC2, Lucene, PC1, Synapse, Xalan, JM1). The obtained AUC values range from 0.97 to 1.00, with a mean AUC of 0.991 and a standard deviation of 0.010. The near-perfect and low-variance results indicate that the proposed generative oversampler consistently enhances classifier discriminability across diverse datasets without exhibiting overfitting tendencies.

Such uniformity in AUC performance across projects with different codebases and metric distributions demonstrates the strong generalizability of the model. Therefore, in addition to achieving statistically significant improvements (as validated by Repeated Measures ANOVA and Bonferroni Post-hoc tests), the proposed GeNSDP model exhibits robust and transferable learning behavior suitable for broad defect prediction scenarios.

However, statistical significance alone does not guarantee robustness or generalizability. To address this, additional validation procedures are considered which are mentioned as below-

1. Cross-Validation Protocol: Each experiment was conducted using 10-fold cross-validation, ensuring that training and testing were performed on disjoint data partitions. This minimizes the likelihood of the model memorizing the training data, thus reducing overfitting risk.
2. Multi-Dataset Evaluation: The proposed model was tested across multiple benchmark datasets (e.g., Apache Xerces, Ant, Camel, Log4j), each with distinct distributions and imbalance ratios. The consistent performance trends across these diverse datasets demonstrate the generalization capability of GeNSDP beyond a single data domain.

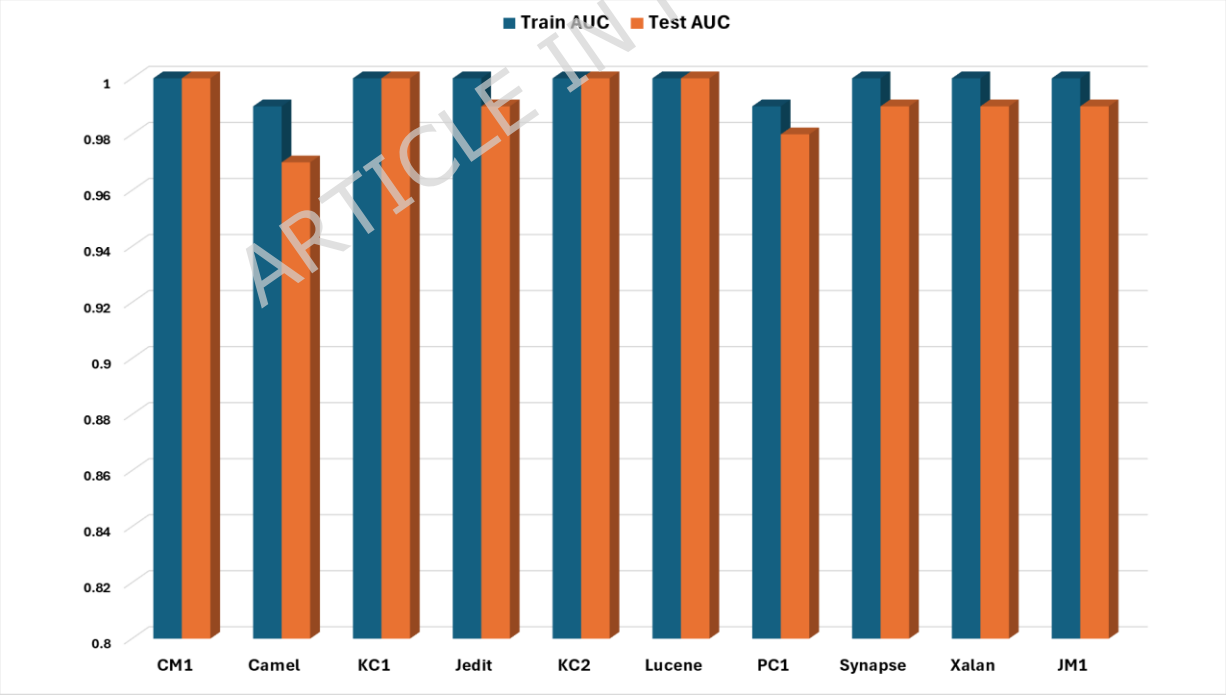

Figure 13. Overfitting Check via Train–Test Performance Gap

3. Variance and Stability Analysis: The standard deviation (σ) of performance metrics (Accuracy, F1, AUC) across datasets and cross-validation folds remained low (typically < 2–3%), indicating that the model's performance is stable and not highly sensitive to data variations.

4. Overfitting Check via Train–Test Performance Gap: The difference between the synthetic training accuracy and testing accuracy (evaluated on original datasets) remained within acceptable limits (< 5%), confirming that the generative oversampler produced generalized synthetic samples instead of overfitted duplicates. The recorded average train AUC is 0.998 and that for Test bed is 0.991. Hence, the Average Gap: 0.7% (refer Figure 13).

Thus, combining statistical evidence with these robustness evaluations (referring *Figure 11, Figure 12, and Figure 13)* the following inference can be made to answer the **RQ#4** that- ***the existence of statistical evidence in support of the results of this experimental study is confirmed. It*** can be confirmed that the proposed GeNSDP achieves both statistically significant and generalizable improvements over conventional and state-of-the-art oversampling methods.

Although, from Section 4.1 through Section 4.4, it is confirmed that the proposed model is effective to software defect prediction, yet it has few technical challenges that include-
(i) the risk of overfitting or mode collapse when training GANs on very few minority samples;
(ii) the need for optimal tuning of generator–discriminator parameters and oversampling ratios;
(iii) ensuring the statistical plausibility and validity of synthesized features; and
(iv) managing the additional computational overhead associated with adversarial training and robustness verification.

5. CONCLUSION AND FUTURE WORKS

This study contributed to a novel class imbalance learning method for software defect prediction- GeNSDP with 1D GAN based lightweight oversampling method that generates synthetic data-points to the original dataset by learning the data distribution properties in the original data. The way, it oversamples the dataset as pre-processing, it maintains diversity and avoids overfitting of SDP model. The preprocessed data is fed to the layers of a deep learner SDP model for training and predicting faulty modules.

5.1 MAJOR FINDINGS OF THE STUDY

The proposed model brings better results than the selected baselines. *The study has brought about the following findings-*

1. The proposed model has been exercised using multiple datasets from PROMISE and NASA repository that ensures the generalizability of the proposed model. The model shows the values of 99.1%, 97.6% and 0.92 for Area Under ROC Curve (AUC), Accuracy and F-measure respectively.
2. The comparison of the proposed model with the selected baselines including 4 oversampling techniques (ROS, COSTE, SMOTE, MAHAKIL) has been made. It results in an improvement of 37.3%, 26.4%, 28.4% and 28.5% in terms of AUC measure over ROS, COSTE, SMOTE, MAHAKIL based models respectively.
3. The proposed model outperformed the conventional oversampling methods with an increase of 30.1% in AUC values. And, with the than the baselines that are also built on GAN oversampling, it performs 14.1% better.
4. An Empirical comparison with state-of-the-art models reflects that the proposed model outperforms GAN + RF, GAN + semi-supervised learner, GAN + Adaboost, and GAN + BertCode by 18.0%, 14.6%, 13.9%, and 9.9% respectively in terms of AUC.
5. The results obtained through the experimentation are statistically validated by the conduction of Repeated ANOVA and Bonferroni Post-hoc Tests.

6. By integrating structured hyperparameter tuning, strict data isolation, multi-dataset robustness evaluation, and formal statistical validation, the proposed GeNSDP framework achieves both methodological rigor and empirical reliability. These enhancements collectively reinforce the novelty, robustness, and generalization capability of the generative oversampling approach

5.2 LIMITATIONS OF THE STUDY

Although this study presents an effective generative oversampling model (GeNSDP) to mitigate class imbalance in software defect prediction (SDP), certain limitations remain. First, preprocessing operations such as resampling can potentially introduce bias, and performing oversampling before the train–test split may lead to data leakage. In this study, strict experimental protocols were maintained to prevent such leakage; however, further verification through independent validation would strengthen reliability. Second, the proposed model assumes that available defect labels comprehensively represent real-world software quality, which may not always hold true due to latent or unreported defects. Third, while the experiments were conducted on well-known public benchmark datasets, generalization to large-scale industrial systems or projects from diverse domains may still be limited. Fourth, GAN part is simplistic as an identity generator is utilized which is not true adversarial learning). Fifth, it has limited scalability to multi-dimensional or noisy data. Sixth, the predictor depends on quality of synthetic input where it lacks advanced regularization and tuning.

5.3 FUTURE SCOPE OF THE WORK

Future research will focus on extending GeNSDP to handle multi-class imbalance scenarios, where class overlap in feature space poses additional complexity. The framework will be enhanced with automated hyperparameter tuning and robustness assessment modules to improve reproducibility and stability across datasets. Moreover, the approach will be validated on real-time industrial defect repositories to examine its scalability and domain adaptability. Such advancements aim to establish a more generalized SDP model capable of early defect prediction and cost-effective quality management in industrial software development.

DECLARATIONS

Funding – No Funding is availed for this project.

Data availability- The data utilized in this paper is available at:

https://github.com/feiwww/PROMISE-backup

https://github.com/ApoorvaKrisna/NASA-promise-dataset-repository

Further enquiries about data availability should be directed to the author (Somya R. Goyal, Manipal University Jaipur, at somyagoyal1988@gmail.com or somya.goyal@jaipur.manipal.edu).

Code availability- https://github.com/SRGoyal/GeNSDP.git

Conflict of interest - The Author has no conflicts of interest.

Ethical approval - This article does not contain any studies on human participants or animals.

REFERENCES

[1] Zhao, Y., Damevski, K. and Chen, H., 2023. A systematic survey of just-in-time software defect prediction. ACM Computing Surveys, 55(10), pp.1-35.

[2] Goyal, S., Software Measurements Using Machine Learning Techniques - A Review, Recent Advances in Computer Science and Communications 2023. Volume 16, Number 1, 2023, pp. 38-55(18), Bentham Science Publishers. https://doi.org/10.2174/2666255815666220407101922

[3] Goyal, S. (2023). Open Challenges in Software Measurements Using Machine Learning Techniques. In: Computational Intelligence Applications for Software Engineering Problems, pp. 19-31. Apple Academic Press. ISBN- 9781000575927, 1000575926.

[4] Bhandari, K., Kumar, K., & Sangal, A. L. (2023). Data quality issues in software fault prediction: a systematic literature review. Artificial Intelligence Review, 56(8), 7839–7908. https://doi.org/10.1007/S10462-022-10371-6

[5] Chen, L., Fang, B., Shang, Z. and Tang, Y., 2018. Tackling class overlap and imbalance problems in software defect prediction. Software Quality Journal, 26(1), pp.97-125. https://doi.org/10.1007/s11219-016-9342-6

[6] Goyal, S. R., Current Trends in Class Imbalance Learning for Software Defect Prediction, in *IEEE Access*, vol. 13, pp. 16896-16917, 2025, doi: 10.1109/ACCESS.2025.3532250.

[7] Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K. E., Kabir, M. A., & Zhang, M. (2021.a). COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction. Information and Software Technology, 129. p.106432 https://doi.org/10.1016/j.infsof.2020.106432

[8] Singh Rathore, S., Singh Chouhan, S., Kumar Jain, D., & Gopal Vachhani, A. (2022). Generative Oversampling Methods for Handling Imbalanced Data in Software Fault Prediction. IEEE TRANSACTIONS ON RELIABILITY, 71(2), 747. https://doi.org/10.1109/TR

[9] Zhang, S., Jiang, S. and Yan, Y., 2022. A software defect prediction approach based on bigan anomaly detection. Scientific Programming, 2022(1), p.5024399.

[10] Alqarni, A. and Aljamaan, H., 2023. Leveraging Ensemble Learning with Generative Adversarial Networks for Imbalanced Software Defects Prediction. Applied Sciences, 13(24), p.13319.

[11] Yedida, R., & Menzies, T. (2022). On the Value of Oversampling for Deep Learning in Software Defect Prediction. IEEE Transactions on Software Engineering, 48(8), 3103–3116. https://doi.org/10.1109/TSE.2021.3079841

[12] Song, W., Gan, L. and Bao, T., 2024. Software Defect Prediction via Generative Adversarial Networks and Pre-Trained Model. International Journal of Advanced Computer Science & Applications, 15(3). 10.14569/IJACSA.2024.01503119

[13] L. Gong, S. Jiang and L. Jiang. 2019. Tackling Class Imbalance Problem in Software Defect Prediction Through Cluster-Based Over-Sampling With Filtering," in IEEE Access, vol. 7, pp. 145725-145737, 2019, doi: 10.1109/ACCESS.2019.2945858.

[14] Feng, S., Keung, J., Yu, X., Xiao, Y. and Zhang, M., 2021.b. Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction. Information and Software Technology, 139, p.106662. https://doi.org/10.1016/j.infsof.2021.106662

[15] Khuat, T. T., & Le, M. H. (2020). Evaluation of Sampling-Based Ensembles of Classifiers on Imbalanced Data for Software Defect Prediction Problems. SN Computer Science, 1(2). https://doi.org/10.1007/s42979-020-0119-4

[16] Goyal, S. Predicting the Defects using Stacked Ensemble Learner with Filtered Dataset. Autom Softw Eng 28, 14 (2021). https://doi.org/10.1007/s10515-021-00285-y

[17] Wang, H., Zhuang, W. and Zhang, X., 2021. Software defect prediction based on gated hierarchical LSTMs. IEEE Transactions on Reliability, 70(2), pp.711-727.

[18] Farid, A.B., Fathy, E.M., Eldin, A.S. and Abd-Elmegid, L.A., 2021. Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM). PeerJ Computer Science, 7, p.e739.

[19] Goyal, S. R., A systematic review on AI based class imbalance handling in software defect prediction, Results in Engineering, Volume 27, 2025, 106578, ISSN 2590-1230, https://doi.org/10.1016/j.rineng.2025.106578.

[20] Khleel, N.A.A., Nehéz, K. 2023.b. Software defect prediction using a bidirectional LSTM network combined with oversampling techniques. Cluster Comput (2023). https://doi.org/10.1007/s10586-023-04170-z

[21] Qiao, L., Li, X., Umer, Q. and Guo, P., 2020. Deep learning based software defect prediction. Neurocomputing, 385, pp.100-110. https://doi.org/10.1016/j.neucom.2019.11.067

[22] Giray, G., Bennin, K.E., Köksal, Ö., Babur, Ö. and Tekinerdogan, B., 2023. On the use of deep learning in software defect prediction. Journal of Systems and Software, 195, p.111537. https://doi.org/10.1016/j.jss.2022.111537

[23] (PROMISE) https://github.com/feiwww/PROMISE-backup/tree/master/bug-data, (NASA) https://github.com/ApoorvaKrisna/NASA-promise-dataset-repository

[24] Aggarwal, Deepti (2021): Software Defect Prediction Dataset. figshare. Dataset. https://doi.org/10.6084/m9.figshare.13536506.v1

[25] Sayyad, S. & Menzies, T.,"The PROMISE Repository of Software Engineering Databases", Canada: University of Ottawa, 2005. http://promise.site.uottawa.ca/ SERepository.

[26] E.L. Lehmann and J.P.Romano, "Testing Statistical Hypothesis: Springer Texts in Statistics", Springer, New York, 2008.

[27] Ross, S., M., "Probability and Statistics for Engineers and Scientists", Third Edition, Elsevier Press, 2005, ISBN: 81-8147-730-8.

[28] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden and S. Mensah, MAHAKIL: Diversity Based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction, in IEEE Transactions on Software Engineering, vol. 44, no. 6, pp. 534-550, 1 June 2018, doi: 10.1109/TSE.2017.2731766.

[29] Thi Minh Phuong, H., Vu Thu Nguyet, P., Huu Nhat Minh, N. *et al.* A comparative study of handling imbalanced data using generative adversarial networks for machine learning based software fault prediction. *Appl Intell* **55**, 280 (2025). https://doi.org/10.1007/s10489-024-05930-z

[30] Arasteh, B., Golshan, S., Shami, S. et al. Sahand: A Software Fault-Prediction Method Using Autoencoder Neural Network and K-Means Algorithm. J Electron Test 40, 229–243 (2024). https://doi.org/10.1007/s10836-024-06116-8

[31] Goyal, Somya R., Effective software defect prediction with deep neural networks, Results in Engineering, Volume 29, 2026, 108378, ISSN 2590-1230, https://doi.org/10.1016/j.rineng.2025.108378.