Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics e-ISSN: 2656-8624
Homepage: https://ijeeemi.org/; Vol. 8, No. 3, pp. 351-364, August 2026

RESEARCH PAPER OPEN ACCES

TabKANet for Software Defect Prediction: An Oversampling and Feature-Selection Ablation Study

Muhammad Faza Azhiman Saputra, Setyo Wahyu Saputro, Mohammad Reza Faisal, Radityo Adi Nugroho, and Andi Farmadi

Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia

Abstract

Software defect prediction (SDP) concentrates limited testing effort on the modules most likely to fail. However, real-world metric data are tabular, noisy, and severely class-imbalanced, which degrades conventional learners. TabKANet, which couples a Kolmogorov-Arnold Network embedding with a Transformer backbone, has shown promise on general tabular data but has not been examined for SDP. In addition, its common preprocessing steps effect remains unquantified. This study adapts TabKANet to the all-numerical, highly imbalanced SDP setting and empirically evaluates it against established baselines, using a structured ablation in order to isolate the contribution of oversampling and feature selection rather than to propose a new architecture. Twelve NASA Metrics Data Program datasets were processed with duplicate removal, MinMax normalization, effective class weighting, and stratified five-fold cross-validation; while SMOTE and Recursive Feature Elimination (RFE) were applied inside the training folds. Four TabKANet variants were compared with Multi-Layer Perceptron (MLP), standalone KAN, and TabNet, and differences were assessed with the Wilcoxon signed-rank test (α = 0.05). The base variant reached a mean AUC of 0.7589, statistically equivalent to MLP and KAN and significantly higher than TabNet. Within the ablation, SMOTE consistently lowered AUC and RFE was neutral, though neither effect was statistically significant; the model still performed best without additional resampling. The main contribution is therefore empirical: for imbalanced SDP, effective class weighting alone is sufficient and SMOTE is counter-productive for TabKANet.

Paper History
Received June 12, 2026
Revised July 25, 2026
Accepted August 2, 2026
Published August 16, 2026

Keywords
Software Defect Prediction;
TabKANet;
Kolmogorov-Arnold Network;
Class Imbalance;
Ablation Study;

Author Email
2211016210001@mhs.ulm.ac.id
setyo.saputro@ulm.ac.id
reza.faisal@ulm.ac.id
radityo.adi@ulm.ac.id
andifarmadi@ulm.ac.id

I. Introduction

Software quality remains a central concern in modern software engineering, especially for large, complex systems where every module testing is exhaustively impractical in time and cost. Software defect prediction (SDP) addresses this condition by learning from historical defect data so that testing effort can focus on the modules most likely to fail, improving reliability while reducing inspection cost [1], [2]. Building accurate SDP models on real software metrics is difficult; however, the data are tabular and noisy with redundant, overlapping features, and defective modules form only a small fraction of the samples, producing severe class imbalance that biases learning toward the majority class [3], [4]. Conventional learners and shallow networks therefore tend to overfit or favour the non-defective class unless careful manual feature engineering is applied [5], [6].

Several deep architectures target tabular data. TabNet uses sequential attention to select features at each decision step [7] and has been benchmarked for SDP on JM1, reaching AUC ≈ 0.90 with NearMiss undersampling, though only on a single dataset [8]. The Kolmogorov–Arnold Network (KAN) replaces the linear weights of a Multi-Layer Perceptron (MLP) with learnable B-spline activations and can rival or exceed MLP in accuracy and interpretability [9], [10]; Türk and Coşkunçay reported consistent gains of KAN over MLP on eleven NASA datasets, but used a standalone KAN without a Transformer to model global feature interactions [11]. Building on these ideas, Gao et al. introduced TabKANet, which embeds numerical features through a KAN module [12] inside a Transformer backbone and matched gradient-boosted trees on general tabular benchmarks [13]. Other recent SDP work combines resampling and feature selection with ensemble or deep models to counter imbalance and redundancy [14], [15], [16], [17].

In this case, two gaps remain. First, TabKANet has not been applied to SDP, where the NASA Metrics Data Program (MDP) datasets are entirely numerical and extremely imbalanced, a setting that removes TabKANet's categorical-embedding advantage and stresses its behaviour under scarcity and skew. Second, although SMOTE [18] and Recursive Feature Elimination (RFE) [14], [19] each help SDP in isolation, no study has

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

measured their contribution to a KAN-and-Transformer model under a class-weighted loss. The open question is whether these classical data- and feature-level remedies still help a model whose attention mechanismis able to, in principle, down-weight uninformative features on its own [20], or whether a weighted loss alone suffices when synthetic minority samples may merely amplify noise on small datasets [3], [21].

This study addresses both gaps empirically rather than architecturally. Its contributions are: (1) the first systematic evaluation of TabKANet for SDP, adapting it to twelve all-numerical, highly imbalanced NASA MDP datasets by removing the categorical-embedding module and applying effective class weighting; (2) a controlled four-variant ablation that quantifies, with non-parametric testing, how SMOTE and RFE affect the model while architecture, validation protocol, and class weighting are held fixed, isolating each component's marginal effect; (3) a benchmark against MLP, standalone KAN, and TabNet under an identical protocol, showing TabKANet to be statistically equivalent to the strongest baselines and significantly better than TabNet; and (4) practical configuration guidance, effective class weighting alone is sufficient, and adding SMOTE is counter-productive for this model class.

The remaining of the paper is organized as follows. Section 2 describes the datasets, preprocessing, model architectures, ablation design, and statistical analysis; Section 3 reports the comparative results, additional metrics, statistical tests, and ablation; Section 4 discusses and interprets the findings and their limitations; and Section 5 concludes with the main findings and future directions.

II. Materials And Method

This study follows a structured workflow, as illustrated in Fig. 1, that adapts TabKANet to software defect prediction [1] and evaluates it against three baselines through a controlled ablation. The process begins with collecting twelve NASA MDP datasets, followed by preprocessing, stratified five-fold cross-validation, fold-internal imbalance and feature handling, model training, evaluation, and statistical testing. Each stage is detailed in the following subsections.

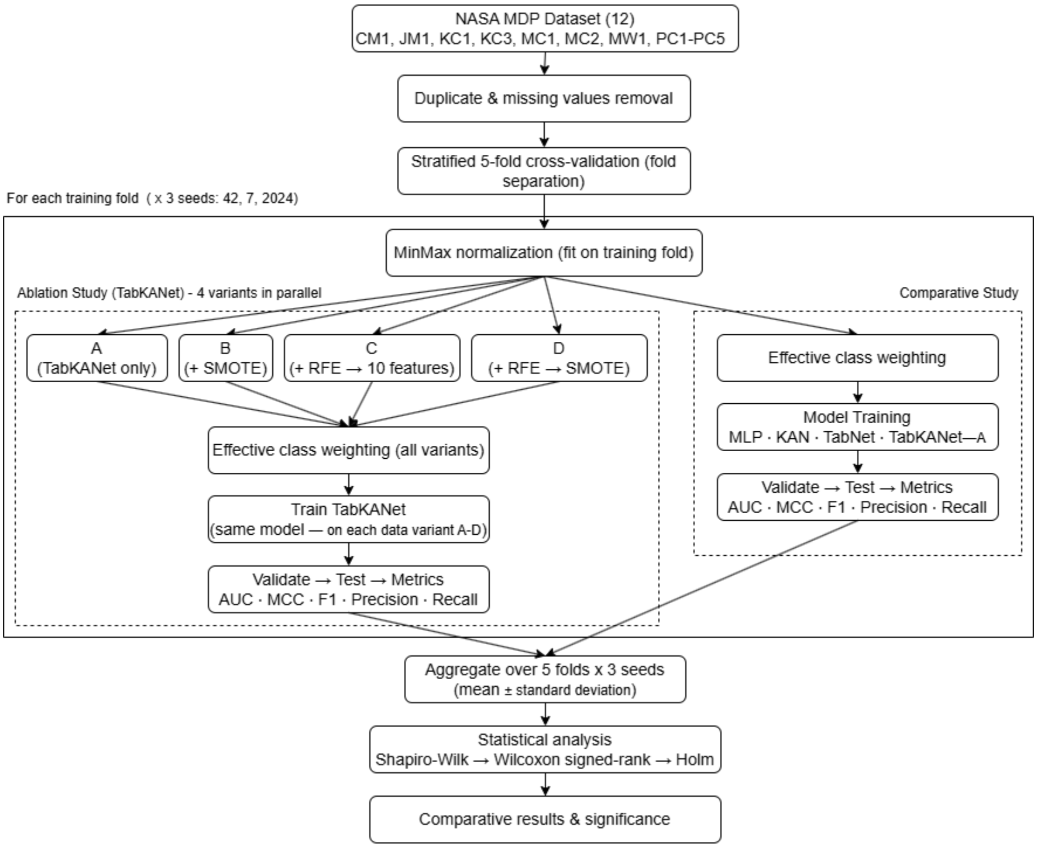

Fig. 1. Workflow: comparative study varies model; ablation study varies preprocessing pipeline

A. Dataset

This study used twelve publicly available defect datasets from the NASA MDP / PROMISE repository [22], [23]: CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, and PC5. Each dataset describes software modules through numerical static code metrics such as lines of code, McCabe cyclomatic complexity, and Halstead measures, with a binary label indicating whether a module is defective. Table 1 summarizes their characteristics after duplicate and missing-value removal. The datasets vary widely in size, from 124 instances (MC2) to 7720 (JM1), and in dimensionality, from 22 to 40 numerical features.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeemi.v8i3.351

Table 1. Characteristics of the twelve NASA MDP dataset.

Dataset	Instances	Features	Defective	Defect (%)
CM1	327	38	42	12.84
JM1	7720	22	1612	20.88
KC1	1162	22	294	25.30
KC3	194	40	36	18.56
MC1	1952	39	36	1.84
MC2	124	40	44	35.48
MW1	250	38	25	10.00
PC1	679	38	55	8.10
PC2	722	37	16	2.22
PC3	1053	38	130	12.35
PC4	1270	38	176	13.86
PC5	1694	39	458	27.04

All features are numerical, and the defect rate is highly imbalanced across datasets, ranging from only 1.84% on MC1 and 2.22% on PC2 to 35.48% on MC2, which makes the prediction task substantially harder on the most skewed sets. The NASA MDP datasets are also known to contain redundant and partly inconsistent instances [15]; duplicate rows were therefore removed before modelling to reduce this source of noise and to prevent train-test leakage.

B. Data Preprocessing and Validation

Duplicate rows, which are common in these datasets, were removed before any data splitting to prevent leakage between training and test partitions. All numerical features were transformed to the range [0, 1] using MinMax normalization, defined in Eq. (1) [24], which stabilizes and accelerates gradient convergence for deep architectures.

$$x'_j = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)} \quad (1)$$

where x_j is the raw value of feature j, $\min(x_j)$ and $\max(x_j)$ are computed on the training fold only, and $x'_j \in [0,1]$ is the normalized value. Model performance was estimated with stratified five-fold cross-validation (n_splits = 5, shuffle = True, random_state = 42), which keeps the defect-to-clean ratio of each fold identical to the full dataset [25]; this is essential under extreme imbalance, where validation that ignores class distribution yields biased estimates. To avoid optimistic bias, normalization, oversampling, and feature selection were all fitted only on the training portion of each fold. The overall preprocessing and partitioning procedure is given as pseudocode in Table 2.

C. Baseline Models

Three established architectures serve as baselines. MLP is a classic fully-connected network whose hidden units apply an affine transformation followed by a non-linear activation [26], [27]. KAN replaces the linear weights of an MLP with learnable activation functions placed on its edges. It is grounded in the Kolmogorov–Arnold representation theorem, which states that any multivariate continuous function can be written as a finite composition of univariate functions, as defined in Eq. (2) [9]:

$$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p} \left(x_p \right) \right) \quad (2)$$

where $\phi_{q,p}$ are inner univariate functions, Φ_q are outer univariate functions, and n is the input dimension. Each edge activation combines a third-order (k = 3) B-spline expansion, as defined in Eq. (3), with a SiLU base function, as defined in Eq. (4). Furthermore, the spline coefficients are learned during training, giving the network a flexible, theoretically grounded form [10], [28]:

$$\text{spline}(x) = \sum_i c_i \ B_i(x) \quad (3)$$

$$\phi(x) = w_b \ b(x) + w_s \ \text{spline}(x), \quad b(x) = \text{SiLU}(x) = \frac{x}{1+e^{-x}} \quad (4)$$

where $B_i(x)$ is B-spline basis functions, c_i is learnable coefficients, $b(x)$ is the SiLU base function, and w_b, w_s are learnable weights. TabNet applies sequential attention to select relevant features automatically at each decision step [7]. All three baselines were trained under the same class-weighting and validation protocol as the proposed model to ensure a fair comparison.

D. Proposed Method: TabKANet

TabKANet is the proposed model and the focus of this study. It couples a Kolmogorov-Arnold numerical embedding with a Transformer backbone [13]. Unlike linear-embedding tabular models that project each numerical feature with a single weight [12], TabKANet embeds every numerical feature independently using its own KAN block. Each feature is first standardized by batch normalization, as defined in Eq. (5), and then mapped to an expressive d_{model}-dimensional token by a feature-specific KAN, as defined in Eq. (6), where the KAN block follows the learnable B-spline formulation of Eqs. (3)-(4):

$$\hat{x}_j = \gamma_j \frac{x'_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} + \beta_j \quad (5)$$

$$\mathbf{e}_j = \text{KAN}_j(\hat{x}_j) \in \mathbb{R}^{d_{\text{model}}}, \ j = 1, \dots, m \quad (6)$$

where μ_j and σ_j^2 are the batch mean and variance, γ_j, β_j are learnable scale and shift, ϵ is a small constant, and m is the number of numerical features. The resulting feature tokens form the encoder input sequence, as defined in Eq. (7) [29]:

$$\mathbf{Z}^{(0)} = [\,\mathbf{e}_1;\ \mathbf{e}_2;\ \dots;\ \mathbf{e}_m\,] \in \mathbb{R}^{m \times d_{\text{model}}} \quad (7)$$

This sequence was processed by a pre-norm Transformer encoder whose multi-head self-attention captures global interactions among features. Each layer projects the tokens into queries, keys, and values, as defined in Eq. (8), and computes scaled dot-product attention, as defined in Eq. (9):

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}^Q,\ \mathbf{K} = \mathbf{Z}\mathbf{W}^K,\ \mathbf{V} = \mathbf{Z}\mathbf{W}^V \quad (8)$$

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

Table 2. Pseudocode of the complete TabKANet training, validation, and evaluation pipeline

Input: NASA MDP datasets {D_1, …, D_{12}}; folds F = 5; seeds S = {42, 7, 2024}; variant v ∈ {A, B, C, D}; RFE size k = 10; class-weight β = 0.9999

Output: Mean ± SD of AUC, MCC, F1, precision, recall per model; Wilcoxon / Holm significance

foreach dataset D **do**	
Remove duplicate and missing-value instances	→ *preprocessing*
foreach seed s ∈ S **do**	
Build stratified F-fold split of D	→ *fold separation*
for f ← 1 **to** F **do**	
(D_{tr}, D_{te}) ← fold f	
Fit MinMax scaler on D_{tr}; transform D_{tr}, D_{te}	→ *normalization*
if v ∈ {C, D} **then** D_{tr} ← RFE(D_{tr}, k)	→ *feature selection*
if v ∈ {B, D} **then** D_{tr} ← SMOTE(D_{tr})	→ *oversampling*
Compute effective class weights w_c (Eq. 19)	→ *class weighting*
Initialize model θ ∈ {MLP, KAN, TabNet, TabKANet}	
Hold out 20% of D_{tr} as D_{val}	→ *validation set*
repeat	
Update θ by minimizing weighted CE loss (Eq. 20) on D_{tr}	→ *training*
Evaluate loss on Dval	→ *validation*
until early stopping on Dval	
ŷ ← modelθ(Dte) at threshold 0.5	→ *testing*
Compute AUC, MCC, F1, precision, recall (Eqs. 21–25)	→ *metric calculation*
end for	
end foreach	
Aggregate metrics: mean ± SD over F × |S| runs	
end foreach	
Shapiro–Wilk test on paired differences	→ *statistical analysis*
Wilcoxon signed-rank test (Eq. 26) with Holm–Bonferroni correction (Eq. 27)	
return aggregated metrics and significance results	

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V} \quad (9)$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$ are learnable projection matrices and d_k is the key dimension. Several attention heads were concatenated, as defined in Eq. (10), and followed by a position-wise feed-forward network, as defined in Eq. (11) [20], [29]:

$$\mathrm{MultiHead}(\mathbf{Z}) = [\mathrm{head}_1; \dots; \mathrm{head}_h]\, \mathbf{W}^O, \quad \mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \quad (10)$$

$$\mathrm{FFN}(\mathbf{z}) = \mathrm{GELU}(\mathbf{z}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \quad (11)$$

where h is the number of attention heads, $\mathbf{W}^O$ is the output projection, and $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ are learnable parameters. Each sublayer used a residual connection with pre-normalization, as defined in Eq. (12), and the final-layer tokens were mean-pooled into a single vector, as defined in Eq. (13):

$$\mathbf{z}' = \mathbf{z} + \mathrm{Sublayer}(\mathrm{LayerNorm}(\mathbf{z})) \quad (12)$$

$$\mathbf{h} = \frac{1}{m}\sum_{j=1}^{m} \mathbf{z}_j^{(L)} \quad (13)$$

where $\mathrm{Sublayer} \in \{\mathrm{MultiHead}, \mathrm{FFN}\}$ and L is the number of encoder layers. The pooled vector was passed to a GELU classification head, as defined in Eq. (14), and converted to a defect probability by a softmax, as defined in Eq. (15) [13]:

$$\mathbf{o} = \mathbf{W}_2\, \mathrm{GELU}(\mathbf{W}_1\, \mathrm{LayerNorm}(\mathbf{h}) + \mathbf{b}_1) + \mathbf{b}_2 \quad (14)$$

$$p(y = c \mid \mathbf{x}) = \frac{e^{o_c}}{\sum_{c'=1}^{2} e^{o_{c'}}} \quad (15)$$

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

Table 3. Principal experimental configuration of the baseline models and TabKANet

Component	Configuration
Framework and validation	PyTorch; seeds 42/7/2024; stratified 5-fold CV; inner 20% validation split for early stopping
Normalization	MinMax scaling to [0, 1], fitted on the training fold only
Optimizer / loss (neural models)	AdamW; class-weighted cross-entropy; early stopping (patience 15)
Batch size / epochs (by tier)	16 / 32 / 64 and 120 / 100 / 80 for the three size tiers
MLP (baseline)	2 hidden layers (hidden 32/64/128 by tier), ReLU, dropout 0.3 then 0.2
KAN (baseline)	efficient-kan; layers [in, hidden, 2]; B-spline grid size 3-5; AdamW lr 1e-3
TabNet (baseline)	n_d = n_a (16/32/64), n_steps = 3, entmax masks, AdamW lr 2e-2, weight decay 1e-4, StepLR (step 20, gamma 0.9)
TabKANet d_model / heads / layers	32-64 / 4-8 / 2-3 by tier
TabKANet feed-forward / dropout	64-128 / 0.2-0.1 by tier; lr 1e-3 (AdamW)
TabKANet KAN embedding	per-feature KAN [1, kan_hidden, d_model], grid size 3, with batch normalization; mean pooling head
SMOTE (variants B, D)	k = 5 neighbors (auto-reduced), training folds only, before scaling
RFE (variants C, D)	logistic-regression estimator (liblinear); 10 features retained

where $\mathbf{o} \in \mathbb{R}^2$ are the output logits and o_c is the logit for class $c \in \{0,1\}$. The token dimension d_{model}, the number of attention heads, and the number of encoder layers were set per dataset-size tier (32–64, 4–8, and 2–3, respectively, as listed in Table 3). Two adaptations were made for software defect prediction, where the categorical embedding branch of the original TabKANet was removed because all NASA MDP features are numerical and an effective class-weighting loss (Subsection E) was incorporated to counter the severe class imbalance. Together, these components allow TabKANet to learn non-linear per-feature representations and their global interactions within a single end-to-end network, which is the key distinction from the baselines. In this case, the MLP applied a single shared linear transformation to the raw feature vector. Furthermore, standalone KAN replaces those linear weights with learnable B-splines but still mixes features only through fully connected layers, whereas TabKANet gives every feature its own KAN embedding and then models pairwise feature interactions explicitly through Transformer self-attention.

E. Imbalance Handling and Ablation Design

For every variant, an effective class-weighting term was added to the loss to penalize errors on the minority defect class [3]. Rather than a plain inverse frequency, the weight was based on the effective number of samples, as defined in Eq. (16) [30], with a hyperparameter β = 0.9999; the majority-class weight was fixed to 1 and the minority weight scaled accordingly. These weights scale the per-sample cross-entropy, as defined in Eq. (17):

$$w_c = \frac{1-\beta}{1-\beta^{\,n_c}},\ \ \beta = 0.9999 \tag{16}$$

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{2} w_c\ y_{i,c}\ log p(y = c \mid \boldsymbol{x}_i) \tag{17}$$

where n_c is the number of training samples in class c, $\beta \in (0,1)$ controls the re-weighting strength, N is the number of samples, and $y_{i,c}$ is the one-hot label. To isolate the effect of each preprocessing component on TabKANet, four variants were defined: (A) base TabKANet with effective class weighting only; (B) A plus SMOTE; (C) A plus RFE; and (D) A plus both SMOTE and RFE. SMOTE generated synthetic minority samples by interpolating between each minority instance and one of its five nearest neighbours, as defined in Eq. (18) [18], [21], where δ is drawn uniformly from [0, 1]; it was applied only within the training folds before scaling:

$$x_{\text{new}} = x_i + \delta\left(x_i^{(k)} - x_i\right),\ \ \delta \sim U(0,1) \tag{18}$$

where x_i is a minority-class sample and $x_i^{(k)}$ is one of its k nearest minority neighbours. RFE used a logistic-regression estimator (liblinear solver) to iteratively eliminate features: it ranks features by the magnitude of the estimator coefficients, as defined in Eq. (19), and recursively discards the least important feature until the ten most informative remain, as defined in Eq. (20) [14], [19]:

$$r_j = |w_j|,\ j = 1, \dots, d \tag{19}$$

$$\mathcal{F}^{(t+1)} = \mathcal{F}^{(t)} \setminus \{j^*\},\ j^* = \arg\min_{j \in \mathcal{F}^{(t)}} r_j,\ \text{until } |\mathcal{F}| = k \tag{20}$$

where w_j is the coefficient of feature j, r_j its importance score, d the number of features, $\mathcal{F}^{(t)}$ the surviving feature set at iteration t, and $k = 10$ the number of retained features. RFE was likewise fitted only on training data. The number of retained features was fixed at ten across all datasets as a deliberate, controlled choice, so that the contribution of feature selection could be isolated within the ablation rather than confounded by a per-dataset search; ten corresponds to roughly a quarter to a half of

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

the 22-40 available software metrics. A single fixed count is admittedly a simplification for datasets of differing dimensionality, and tuning the number of retained features per dataset is left for future work.

F. Evaluation and Statistical Analysis

Since the NASA MDP datasets are highly imbalanced, no single scalar metric fully characterizes model performance. This study therefore reports five complementary metrics, AUC, MCC, F1-score, precision, and recall, following established evaluation practice in software defect prediction [31][32]. Precision and recall expose the two operationally important error types in testing, namely missed defects versus false alarms, and are defined in Eqs. (21)-(22); their harmonic mean, the F1-score, condenses this trade-off into a single value focused on the minority (defective) class, as defined in Eq. (23) [31], [33]:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (21)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (22)$$

$$F1 = \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \quad (23)$$

where TP, FP, and FN are the true-positive, false-positive, and false-negative counts. MCC used all four cells of the confusion matrix and is high only when predictions are good for both classes, which makes it the most reliable single-number summary under class imbalance, as defined in Eq. (24) [32], [34]; AUC is threshold-independent and measures the ability to rank defective modules above non-defective ones, which keeps it comparatively stable under imbalance, as defined in its Mann–Whitney form in Eq. (25) [31], [32]:

$$\text{MCC} = \frac{TP\cdot TN - FP\cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (24)$$

$$\text{AUC} = \frac{1}{n_+ n_-}\sum_{i=1}^{n_+}\sum_{j=1}^{n_-}\mathbf{1}\left(s_i^+ > s_j^-\right) \quad (25)$$

where TN is the true-negative count, n_+ and n_- are the numbers of positive and negative samples, s^+ and s^- their predicted scores, and $\mathbf{1}(\cdot)$ the indicator function. All threshold-dependent metrics (precision, recall, F1, and MCC) were computed at the default decision threshold of 0.5, taking for each module the class with the higher softmax probability (argmax); no threshold tuning was performed. MCC ranges from −1 to +1, where higher values indicate better agreement between predictions and ground truth.

For each dataset, the five-fold results were averaged to obtain a mean and standard deviation. After checking the distribution of the paired differences with the Shapiro–Wilk test, the Wilcoxon signed-rank test was applied as the primary non-parametric test, with its statistic defined in Eq. (26), complemented by a paired t-test, to assess whether AUC differences across the twelve datasets were statistically significant at a significance level of 0.05. Since three pairwise comparisons were performed in each analysis, p-values were additionally adjusted with the Holm–Bonferroni correction, as defined in Eq. (27) [31]:

$$W = \sum_{i=1}^{n}\text{sgn}\,(d_i)\,R_i,\;\; d_i = a_i - b_i \quad (26)$$

$$\text{reject } H_{(i)} \Leftrightarrow p_{(i)} \le \frac{\alpha}{m-i+1} \quad (27)$$

where d_i is the paired AUC difference between two models on dataset i, R_i is the rank of $|d_i|$, n is the number of datasets, $p_{(i)}$ is the i-th smallest p-value, m is the number of comparisons, and $\alpha = 0.05$.

G. Experimental Setup

All experiments were repeated over three random seeds (42, 7, and 2024), and we reported the mean and standard deviation across seeds; each run was implemented in PyTorch. Within each of the five stratified cross-validation folds, 20% of the training portion was held out as a validation set for early stopping. To keep the comparison fair across datasets of very different sizes, the main hyperparameters were adapted using three size tiers (fewer than 600, 600 to 2000, and 2000 or more samples), consistent with common practice for deep-learning-based defect models [5]. Table 3 lists the principal configuration of every model.

III. Result

A. Ablation Study

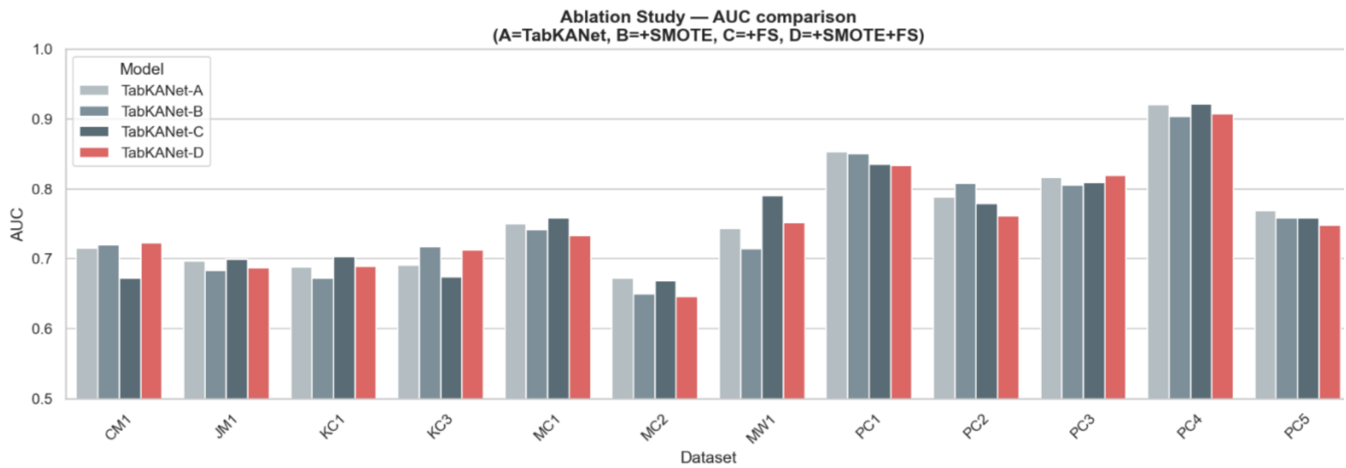

Fig. 2. Ablation study: AUC of TabKANet variants A (base), B (+SMOTE), C (+RFE), and D (+SMOTE+RFE) per dataset

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

The ablation study first established which TabKANet configuration is the best after the configuration was compared to the baselines. Fig. 2 shows the per-dataset AUC of the four TabKANet variants. The mean AUC was the highest for the base variant A (0.7589), followed by C with RFE (0.7558), B with SMOTE (0.7521), and D with both (0.7510). Table 4 reports the per-dataset AUC of the four variants, Table 5 the mean AUC, MCC, and F1 for each variant, and Table 6 the corresponding Wilcoxon tests against variant A.

Table 4. Per-dataset AUC (mean ± SD) of the TabKANet ablation variants A (base), B (+SMOTE), C (+RFE), and D (+SMOTE+RFE)

Dataset	A	B	C	D
CM1	0.7158 ± 0.0523	0.7205 ± 0.0411	0.6723 ± 0.0460	**0.7229 ± 0.0605**
JM1	0.6966 ± 0.0009	0.6836 ± 0.0053	**0.6994 ± 0.0009**	0.6875 ± 0.0025
KC1	0.6883 ± 0.0038	0.6725 ± 0.0149	**0.7030 ± 0.0052**	0.6892 ± 0.0065
KC3	0.6913 ± 0.0112	**0.7169 ± 0.0118**	0.6738 ± 0.0206	0.7124 ± 0.0043
MC1	0.7503 ± 0.0241	0.7413 ± 0.0300	**0.7587 ± 0.0062**	0.7336 ± 0.0206
MC2	**0.6722 ± 0.0058**	0.6496 ± 0.0352	0.6682 ± 0.0310	0.6458 ± 0.0501
MW1	0.7434 ± 0.0373	0.7141 ± 0.0578	**0.7905 ± 0.0398**	0.7517 ± 0.0213
PC1	**0.8533 ± 0.0131**	0.8505 ± 0.0171	0.8352 ± 0.0046	0.8330 ± 0.0163
PC2	0.7881 ± 0.0572	**0.8077 ± 0.0296**	0.7793 ± 0.0147	0.7615 ± 0.0437
PC3	0.8170 ± 0.0098	0.8056 ± 0.0123	0.8096 ± 0.0072	**0.8191 ± 0.0106**
PC4	0.9208 ± 0.0076	0.9042 ± 0.0033	**0.9216 ± 0.0039**	0.9072 ± 0.0070
PC5	**0.7692 ± 0.0031**	0.7589 ± 0.0148	0.7583 ± 0.0047	0.7485 ± 0.0079
Average	**0.7589**	0.7521	0.7558	0.7510

Table 5. Mean AUC, MCC, and F1 of the four TabKANet ablation variants across the twelve datasets

Variant	AUC	MCC	F1
A (base)	**0.7589**	**0.2747**	0.3787
B (+SMOTE)	0.7521	0.2731	**0.3814**
C (+RFE)	0.7558	0.2637	0.3727
D (+SMOTE+RFE)	0.7510	0.2685	0.3792

Table 6. Wilcoxon signed-rank test of AUC between the base TabKANet (A) and each ablation variant

Comparison	W	p-value	Mean A	Mean variant	Conclusion
A vs B (SMOTE)	22	0.204	0.7589	0.7521	Not significant (Holm p=0.454)
A vs C (RFE)	28	0.424	0.7589	0.7558	Not significant (Holm p=0.454)
A vs D (SMOTE+RFE)	20	0.151	0.7589	0.7510	Not significant (Holm p=0.454)

Across three seeds, adding SMOTE consistently lowered AUC (A vs B, W = 22, p = 0.204), but this reduction was no longer statistically significant; RFE produced no significant change (A vs C, p = 0.424) and the combination remained statistically neutral (A vs D, p = 0.151). The single-seed significance initially observed for SMOTE did not survive averaging over multiple seeds, which highlights the importance of repeated runs on these small datasets. The per-dataset AUC in Table 4 confirms that no single variant dominated: RFE (variant C) gave the best AUC on several datasets (KC1, KC3, MC1) and the combined variant D on PC2 and PC3, but these gains did not translate into a significant overall advantage and were offset by losses elsewhere, most notably the large MW1

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

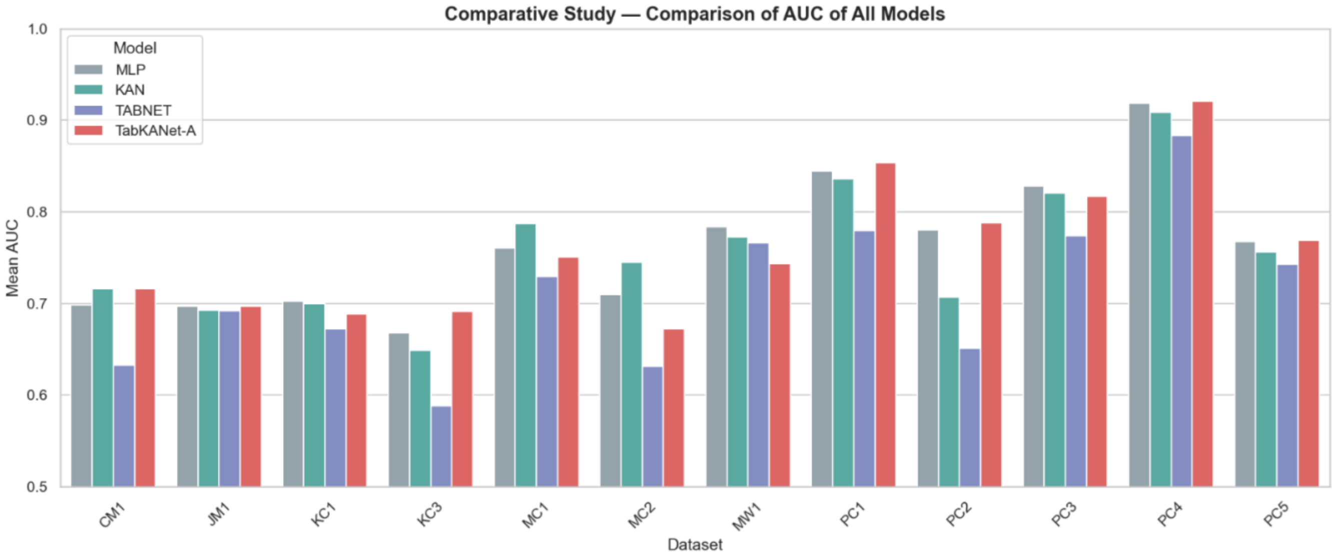

Fig. 3. **Comparative AUC of MLP, KAN, TabNet, and TabKANet-A across the twelve NASA MDP datasets**

drop when SMOTE was added (0.7434 to 0.7141). The base TabKANet with effective class weighting alone was therefore the best overall configuration: neither resampling nor feature selection improved it on average, and oversampling was counter-productive.

B. Comparative Performance

Having identified the base TabKANet (variant A) as the best configuration in the ablation above, we compared it with the three baselines. Variant A was chosen as the representative TabKANet for this comparison because it achieved the highest mean AUC among the four variants (0.7589, against 0.7558 for the next-best variant C) while being the simplest pipeline, using neither oversampling nor feature selection. The Wilcoxon tests moreover found no significant difference between A and any other variant

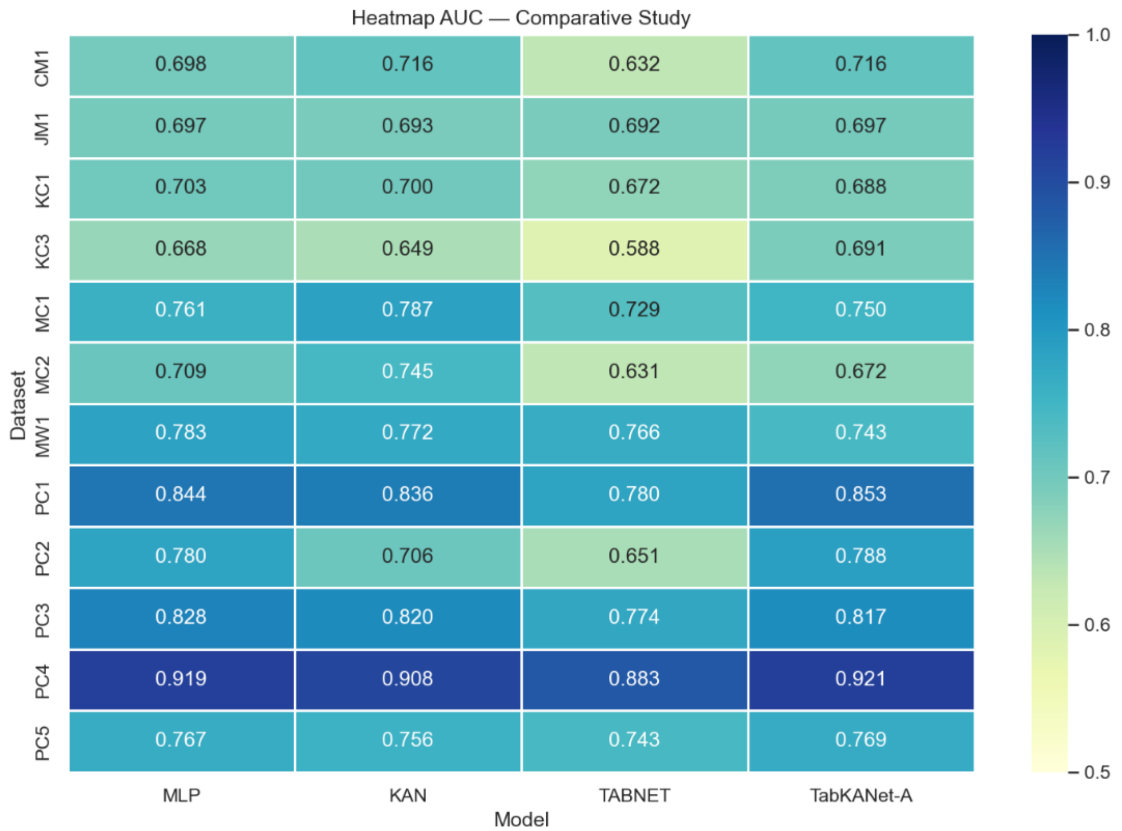

Fig. 4. **Heatmap of mean AUC per model and dataset for the comparative study**

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

(all p > 0.15), so the occasional per-dataset wins of variants C and D were not statistically reliable; by the principle of parsimony, the variant that matches the others without any additional preprocessing is the natural choice to represent the model against the baselines. Across the twelve datasets and three seeds, TabKANet-A reached a mean AUC of 0.7589, statistically on par with MLP (0.7631) and KAN (0.7574), thus clearly above TabNet (0.7118). Fig. 3 compares the AUC of all models per dataset and Fig. 4 summarizes the same values as a heatmap. TabKANet obtained the best or tied-best AUC on several datasets, notably PC4 (0.921), PC1 (0.853), and PC5 (0.769), and it was best or tied-best on seven of the twelve datasets (CM1, JM1, KC3, PC1, PC2, PC4, and PC5), while remaining competitive elsewhere. TabNet was consistently the weakest model on AUC, especially on the smaller datasets such as CM1, KC3, and PC2. The per-dataset AUC values are listed in Table S1 of the Supplementary Material.

C. Additional Metrics

Beyond AUC, the threshold-dependent metrics reveal how each model balances the detection of defective modules against false alarms. Table S2 of the Supplementary Material reports the per-dataset MCC, F1, precision, and recall for every model, and Table 7 summarizes their means. TabKANet-A obtained the best mean MCC (0.2747) and the second-best mean F1 (0.3787) and precision (0.3111), essentially tied with MLP (F1 0.3861, precision 0.3131) by a small margin and clearly exceeding TabNet (F1 0.2943, MCC 0.1353). Its MCC was the highest on the larger, better-conditioned datasets, where the defect signal is stronger.

TabNet showed a characteristic failure mode. It recorded the highest mean recall (0.7434) but the lowest mean precision (0.2290), indicating that it predicted the defect class far too liberally. This is the most visible on PC2, where TabNet's recall stayed high while its precision fell to about 0.04. By contrast, TabKANet, MLP, and KAN maintained a more balanced precision-recall trade-off, which is why their MCC and F1 values are consistently higher despite lower recall. On the most imbalanced datasets (PC2, MC1), all models struggled, and TabKANet collapsed to a zero F1 on PC2, mirroring the difficulty every model faced on that set.

Table S2 makes this trade-off explicit at the dataset level. TabNet recall exceeded 0.78 on CM1, JM1, PC3, and PC4 and reached 1.000 on PC2, yet its precision on those same datasets rarely rose above 0.20, confirming that its apparent sensitivity came at the cost of many false alarms. This weak precision may stem partly from the default 0.5 decision threshold and the hyperparameter configuration adopted here rather than from the TabNet architecture itself; a tuned threshold or dataset-specific hyperparameters could narrow the gap. TabKANet, in contrast, attained the highest precision on MW1 and a strong recall on PC1 (0.842), showing that it can detect defects without indiscriminately flagging clean modules. These per-dataset patterns explain why the aggregate AUC and MCC favor the balanced models even though TabNet leads on recall alone.

Table 7. Mean performance of each model across the twelve datasets for all evaluation metrics

Model	AUC	MCC	F1	Precision	Recall
MLP	**0.7631**	0.2746	**0.3861**	**0.3131**	0.5675
KAN	0.7574	0.2580	0.3705	0.2987	0.5567
TabNet	0.7118	0.1353	0.2943	0.2290	**0.7434**
TabKANet-A	0.7589	**0.2747**	0.3787	0.3111	0.5784

Table 8. Wilcoxon signed-rank test of AUC between TabKANet-A and each baseline across the twelve datasets

Comparison	W	p-value	Mean A	Mean other	Conclusion
TabKANet-A vs MLP	33	0.677	0.7589	0.7631	Not significant (Holm p=0.791)
TabKANet-A vs KAN	35	0.791	0.7589	0.7574	Not significant (Holm p=0.791)
TabKANet-A vs TabNet	4	0.003	0.7589	0.7118	Significant, A better (Holm p=0.010)

D. Statistical Significance

Table 8 summarizes the Wilcoxon signed-rank test on AUC. The differences between TabKANet-A and MLP (W = 33, p = 0.677) and between TabKANet-A and KAN (W = 35, p = 0.791) were not statistically significant, meaning TabKANet is statistically equivalent to these strong baselines. Against TabNet, the difference was significant (W = 4, p = 0.003) and remained significant after Holm-Bonferroni correction (adjusted p = 0.010), confirming that TabKANet performs better than TabNet on these datasets. The paired t-test, used as a complementary parametric check, led to the same conclusions, supporting the robustness of the ranking.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

Table 9. Comparison of TabKANet-A with representative prior studies on tabular and software defect prediction

Author (Year)	Method	Dataset	Result	Limitation
Turk & Coskuncay (2025) [11]	KAN vs. MLP	11 NASA MDP	KAN consistently outperformed MLP (~12% higher accuracy)	Standalone KAN; no Transformer for global feature interactions
Siswantoro & Yuhana (2023) [23]	SHL-MLP + SMOTE + PCA + random search	12 NASA MDP	Highest accuracy on MC1, PC1, PC3	Traditional ML; relies heavily on tuning; no self-attention
Asal & Yalciner (2026) [8]	TabNet, NODE, FT-Transformer + NearMiss	JM1 only	TabNet AUC = 0.90, accuracy 86%	Single dataset; aggressive undersampling inflates scores
Poeta et al. (2024) [10]	KAN benchmarking	UCI tabular (general)	KAN matched or exceeded MLP on accuracy and F1	Not defect prediction; general-purpose tabular data
Gao et al. (2024) [13]	TabKANet (KAN embedding + Transformer)	6 general tabular sets	Matched GBDT; outperformed neural baselines	Not defect prediction; mixed numerical + categorical features
Ghinaya et al. (2024) [14]	SMOTE + RFE + Random Forest	NASA MDP	SMOTE and RFE improved feature-importance analysis	Tree-based; SMOTE/RFE not tested on KAN + Transformer
This study (2026)	TabKANet + four-variant SMOTE/RFE ablation under class weighting	12 NASA MDP (all-numerical, highly imbalanced)	Statistically equivalent to MLP/KAN, significantly better than TabNet; class weighting alone sufficient, SMOTE counter-productive	Within-project only; fixed 0.5 threshold; RFE fixed at 10 features

IV. Discussion

A. Interpretation of Results

Several factors explain why TabKANet matches MLP and KAN yet clearly beats TabNet. First, dataset size: after duplicate removal the NASA MDP datasets are small, below the scale at which attention-based tabular models dominate, so the Transformer backbone's extra capacity cannot be fully exploited and a well-tuned MLP or standalone KAN matches it [13]. Second, the features are entirely numerical, which removes TabKANet's categorical-embedding advantage and reduces it to its KAN numerical embedding plus self-attention. Third, with defect rates as low as 1-2%, the effective class weighting applied to all models dominates achievable performance and compresses the gaps between the balanced models. Fourth, TabNet's sequential attention with sparse feature masking is data-hungry and unstable on small, noisy, imbalanced tables; it over-predicts the minority class (highest mean recall 0.7434 but lowest precision 0.2290), giving the lowest AUC, MCC, and F1. On JM1 it reached AUC = 0.90 only with NearMiss undersampling, versus 0.6919 here. Fifth, the fixed 0.5 threshold and tier-based hyper-parameters constrain every deep model, with TabNet the most affected.

A dataset-level view explains why the mean differences between TabKANet, MLP, and KAN are small. The models agree closely on the easier, larger datasets (PC1, PC3, PC4, PC5), where all reach high AUC, and they diverge mainly on the smallest and most imbalanced sets (CM1, KC3, MC2, PC2), where a handful of misclassified modules swings every metric and the high standard deviations across folds reflect this instability. TabKANet's advantage was clearest on MW1 and PC1, datasets with enough samples for the Transformer to exploit feature interactions, whereas on the very small MC2 it fell slightly behind KAN. This pattern is consistent with prior reports that KAN and attention-based models benefit from larger instance counts, and it indicates that TabKANet's headroom is the greatest on better-conditioned defect datasets. The ablation results further clarify how preprocessing interacts with the model. Adding SMOTE lowered mean AUC (variant B, 0.7521 versus 0.7589 for the base) and was most damaging on MW1, where AUC dropped from 0.7434 to 0.7141. Since the effective class weighting already directs the loss toward the minority class, the synthetic samples generated by SMOTE add little new signal and instead amplify noise on these small, sparsely populated datasets, so the two imbalance remedies partly cancel rather than reinforce one another. RFE was close to neutral (variant C, 0.7558); the self-attention backbone can already down-weight uninformative features, and the NASA metrics are strongly correlated, so removing eleven of the twenty-plus

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

features neither discards much unique information nor sharpens the representation. Combining both (variant D, 0.7510) gave no additional benefit, and none of the three contrasts against the base variant was statistically significant (all p > 0.15). The base configuration with class weighting alone was therefore the most effective and the simplest.

B. Comparison with Prior Studies

To position these findings within the literature, we compared TabKANet-A against representative prior studies on tabular and defect-prediction data, summarized in Table 9. Turk and Coskuncay [11] reported that Kolmogorov-Arnold Networks are competitive with or superior to MLP on eleven NASA MDP datasets, which agrees with our finding that the KAN-based TabKANet matches the strong MLP and KAN baselines; their standalone KAN, however, omits a Transformer and therefore cannot model global feature interactions. Siswantoro and Yuhana [23] reached their best results on the same twelve datasets only after combining SMOTE, PCA, and hyper-parameter search, underlining how heavily classical pipelines depend on tuning, where as TabKANet-A attains a comparable mean AUC with class weighting alone. Asal and Yalciner [8] further reported an AUC of about 0.90 for TabNet, but only on the single JM1 dataset. After aggressive NearMiss undersampling and under our leakage-free protocol without undersampling, TabNet reached only about 0.69 on JM1, showing how strongly such design choices inflate single-dataset scores. Gao et al. [13] introduced TabKANet for general tabular benchmarks with mixed numerical and categorical features and matched gradient-boosted trees, but did not evaluate it on defect prediction or on all-numerical, highly imbalanced data. Ghinaya et al. [14] showed that SMOTE and RFE improve tree-based defect models, yet their contribution to a KAN-and-Transformer model under a class-weighted loss had not been quantified before this study. Taken together, reported mean AUC on NASA MDP tends to cluster around 0.75-0.80 once realistic, leakage-free validation was used, and TabKANet-A sits firmly within this competitive band while maintaining a balanced precision-recall profile rather than trading precision for recall as TabNet does. Against this backdrop, the present study is, to our knowledge, the first to apply TabKANet to software defect prediction, the first to adapt it to all-numerical and extremely imbalanced NASA MDP data, and the first to isolate the marginal effect of SMOTE and RFE on a KAN-and-Transformer model through a controlled four-variant ablation under a fixed class-weighted protocol.

C. Limitations

This study has several limitations. The NASA MDP datasets contain duplicate, inconsistent, and implausible records, so despite duplicate removal residual noise and labelling errors limit the achievable performance ceiling [22]. All experiments are within-project; the more realistic cross-project and cross-version settings were not evaluated. Hyper-parameters were fixed per size tier rather than tuned per dataset, which could change the ranking, especially for TabNet. RFE retained a fixed ten features for every dataset, although the optimal subset size likely varies. All threshold-dependent metrics use the default 0.5 threshold, which is suboptimal under extreme imbalance, as shown by the near-zero MCC of every model on MC1 and PC2 despite moderate-to-high AUC. On the smallest, most imbalanced datasets (CM1, MC2, PC2) the per-fold standard deviations are large, so per-dataset rankings there are unreliable.

D. Implications

The findings carry both practical and research implications. In practice, because no single metric is sufficient, practitioners should pair a ranking metric (AUC) for prioritizing modules with a threshold-calibrated metric (MCC or F1) tuned to the project's imbalance before allocating testing effort; relying on AUC alone can overstate usefulness on extremely imbalanced modules. Equally important, effective class weighting alone was sufficient for TabKANet and adding SMOTE was counter-productive, so teams applying KAN-and-Transformer models to defect data can favour a simpler, leakage-safe pipeline without synthetic oversampling, reducing computation and tuning effort.

For research, this study provides the first TabKANet benchmark on software defect prediction and shows that classical data- and feature-level remedies do not automatically transfer to attention-based tabular models on small, highly imbalanced tables. This motivates future work on threshold calibration, per-dataset feature-count selection, and evaluation in cross-project and cross-version settings.

V. Conclusion

This study set out to adapt TabKANet, a Kolmogorov–Arnold numerical embedding within a Transformer backbone to software defect prediction on twelve all-numerical, highly imbalanced NASA MDP datasets, and to quantify, through a controlled ablation, whether SMOTE and RFE improve it. Numerically, the base TabKANet reached a mean AUC of 0.7589, statistically equivalent to MLP (0.7631, p = 0.677) and KAN (0.7574, p = 0.791) and significantly higher than TabNet (0.7118, p = 0.003 after Holm correction); on threshold metrics it was comparable to MLP (MCC 0.2747 versus 0.2746; F1 0.3787 versus 0.3861) and clearly exceeded TabNet on every measure except recall. The practical conclusion on imbalance handling is that effective class weighting alone is sufficient for this data class: SMOTE consistently lowered AUC and RFE was neutral, and although neither effect was statistically significant, the simplest configuration retained the best mean performance and is therefore preferred. Future work includes cross-project and cross-version validation, per-dataset decision-threshold optimization to recover the precision–recall balance lost at the 0.5 threshold, per-dataset feature-subset and hyper-parameter tuning, and evaluation on larger and more diverse defect datasets, where the Transformer backbone of TabKANet may reveal a clearer advantage.

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

References

[1] S. R. Goyal, "Effective software defect prediction with deep neural networks," *Results in Engineering*, vol. 29, Mar. 2026, doi: 10.1016/j.rineng.2025.108378.

[2] A. B. Nassif *et al.*, "Software defect prediction using learning to rank approach," *Sci. Rep.*, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-45915-5.

[3] H. Shi, J. Ai, J. Liu, and J. Xu, "Improving Software Defect Prediction in Noisy Imbalanced Datasets," *Applied Sciences (Switzerland)*, vol. 13, no. 18, Sep. 2023, doi: 10.3390/app131810466.

[4] R. Malhotra, R. Kapoor, P. Saxena, and P. Sharma, "SAGA: A Hybrid Technique to handle Imbalance Data in Software Defect Prediction," in *ISCAIE 2021 - IEEE 11th Symposium on Computer Applications and Industrial Electronics*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 331–336. doi: 10.1109/ISCAIE51753.2021.9431842.

[5] A. Rawat, A. Mishra, H. Alsharif, R. M. Alharthi, and B. B. Gupta, "Fault classification in the architecture of virtual machine using deep learning," *Sci. Rep.*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-17109-8.

[6] T. Siddiqui and M. Mustaqeem, "Performance evaluation of software defect prediction with NASA dataset using machine learning techniques," *International Journal of Information Technology*, vol. 15, no. 8, pp. 4131–4139, 2023, doi: 10.1007/s41870-023-01528-9.

[7] S. Sercan¨, S. Arık, and T. Pfister, "TabNet: Attentive Interpretable Tabular Learning," 2021. [Online]. Available: www.aaai.org

[8] B. Asal and B. Yalciner, "Benchmarking TabNet, NODE, and FT-Transformer for Software Defect Prediction: An Empirical Comparison and Explainability Analysis," *IEEE Access*, pp. 1–1, Jan. 2026, doi: 10.1109/access.2026.3656247.

[9] Z. Liu *et al.*, "KAN: Kolmogorov-Arnold Networks," Feb. 2025, [Online]. Available: http://arxiv.org/abs/2404.19756

[10] E. Poeta, F. Giobergia, E. Pastor, T. Cerquitelli, and E. Baralis, "A Benchmarking Study of Kolmogorov-Arnold Networks on Tabular Data," Jun. 2024, doi: 10.1109/AICT61888.2024.10740444.

[11] S. Türk and A. Coşkunçay, "A comparative study of Kolmogorov-Arnold networks and multilayer perceptrons for software defect prediction with SHAP analysis," *Cluster Comput.*, vol. 28, Oct. 2025, doi: 10.1007/s10586-025-05691-5.

[12] W. Gao, Z. Gong, Z. Deng, and L. Ma, "Revisiting the numerical feature embeddings structure in neural network-based tabular modelling," *Knowl. Based. Syst.*, vol. 330, Nov. 2025, doi: 10.1016/j.knosys.2025.114697.

[13] W. Gao, Z. Gong, Z. Deng, F. Rong, C. Chen, and L. Ma, "TabKANet: Tabular Data Modeling with Kolmogorov-Arnold Network and Transformer," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2409.08806

[14] H. Ghinaya, R. Herteno, M. R. Faisal, A. Farmadi, and F. Indriani, "Analysis of Important Features in Software Defect Prediction using Synthetic Minority Oversampling Techniques (SMOTE), Recursive Feature Elimination (RFE) and Random Forest," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 6, no. 3, pp. 276–288, Jul. 2024, doi: 10.35882/jeeemi.v6i3.453.

[15] Y. Zhang and N. Liu, "Investigation and Research on Several Key Issues of Software Defect Prediction," *IET Software*, vol. 2025, no. 1, p. 6615496, Jan. 2025, doi: https://doi.org/10.1049/sfw2/6615496.

[16] E. A. Kusnanti, L. D. F. Vantie, and U. L. Yuhana, "SOFTWARE DEFECT PREDICTION USING PCA BASED RECURRENT NEURAL NETWORK," *JUTI: Jurnal Ilmiah Teknologi Informasi*, pp. 23–31, Jan. 2024, doi: 10.12962/j24068535.v22i1.a1199.

[17] Q. Zhang *et al.*, "Software Defect Prediction Using Deep Q-Learning Network-Based Feature Extraction," *IET Software*, vol. 2024, no. 1, 2024, doi: 10.1049/2024/3946655.

[18] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," 2002.

[19] H. Hartati, R. Herteno, M. R. Faisal, F. Indriani, and F. Abadi, "Recursive Feature Elimination Optimization Using Shapley Additive Explanations in Software Defect Prediction with LightGBM Classification," *JURNAL INFOTEL*, vol. 17, no. 1, pp. 1–16, Feb. 2025, doi: 10.20895/infotel.v17i1.1159.

[20] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting Deep Learning Models for Tabular Data." [Online]. Available: https://github.com/yandex-research/rtdl.

[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," *Inf. Softw. Technol.*, vol. 139, Nov. 2021, doi: 10.1016/j.infsof.2021.106662.

[22] M. Fikri *et al.*, "MULTI-CRITERIA DECISION MAKING DALAM SELEKSI FITUR ENSEMBLE UNTUK PREDIKSI CACAT PERANGKAT LUNAK MULTI-CRITERIA DECISION MAKING IN ENSEMBLE FEATURE SELECTION FOR

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

SOFTWARE DEFECT PREDICTION," vol. 12, no. 6, pp. 1384–1394, 2025.

[23] M. Z. F. N. Siswantoro and U. L. Yuhana, "Software Defect Prediction Based on Optimized Machine Learning Models: A Comparative Study," *Teknika*, vol. 12, no. 2, pp. 166–172, Jun. 2023, doi: 10.34148/teknika.v12i2.634.

[24] J. M. H. Pinheiro *et al.*, "The Impact of Feature Scaling in Machine Learning: Effects on Regression and Classification Tasks," *IEEE Access*, vol. 13, pp. 199903–199931, 2025, doi: 10.1109/ACCESS.2025.3635541.

[25] S. Prusty, S. Patnaik, and S. K. Dash, "SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer," *Frontiers in Nanotechnology*, vol. 4, Aug. 2022, doi: 10.3389/fnano.2022.972421.

[26] S. Juneja *et al.*, "Machine learning-based defect prediction model using multilayer perceptron algorithm for escalating the reliability of the software," *Journal of Supercomputing*, vol. 80, no. 7, pp. 10122–10147, May 2024, doi: 10.1007/s11227-023-05836-6.

[27] S. H. Gulo and A. H. Lubis, "Penerapan Multi-Layer Perceptron untuk Mengklasifikasi Penduduk Kurang Mampu," 2024.

[28] C. Dong, L. Zheng, and W. Chen, "Kolmogorov-Arnold Networks (KAN) for Time Series Classification and Robust Analysis," Sep. 2024, [Online]. Available: http://arxiv.org/abs/2408.07314

[29] A. Vaswani *et al.*, "Attention Is All You Need."

[30] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-Balanced Loss Based on Effective Number of Samples," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.05555

[31] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-56706-x.

[32] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," *Journal of Systems and Software*, vol. 195, Jan. 2023, doi: 10.1016/j.jss.2022.111537.

[33] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS One*, vol. 10, no. 3, Mar. 2015, doi: 10.1371/journal.pone.0118432.

[34] K. M. Sujon, R. Hassan, K. Choi, and M. A. Samad, "Accuracy, precision, recall, f1-score, or MCC? empirical evidence from advanced statistics, ML, and XAI for evaluating business predictive models," *J. Big Data*, vol. 12, no. 1, Dec. 2025, doi: 10.1186/s40537-025-01313-4.

Author Biography

Muhammad Faza Azhiman Saputra is an undergraduate student in the Computer Science Study Program, at the Faculty of Mathematics and Natural Sciences, Universitas Lambung Mangkurat (ULM), South Kalimantan, Indonesia. He began his studies in 2022. His research interests include software defect prediction, deep learning, and artificial intelligence. He is particularly focused on applying deep learning methods for improving software performance. His current work involves applying the TabKANet architecture for software defect prediction. Email: 2211016210001@mhs.ulm.ac.id

Setyo Wahyu Saputro obtained his bachelor's degree in Computer Science from Lambung Mangkurat University in 2011 and completed his Master's degree in Informatics at STMIK Amikom University in 2016. He is currently a lecturer in the Computer Science program at the Faculty of Mathematics and Natural Sciences, Lambung Mangkurat University, Banjarbaru. Since 2017, he has been actively engaged as an information technology practitioner and consultant, serving as a project manager and system analyst for various projects in both government and private sectors in South Kalimantan. His research interests include software engineering, human computer interaction, and artificial intelligence applications. He can be contacted via email at setyo.saputro@ulm.ac.id

Mohammad Reza Faisal was born in Banjarmasin. After graduating from high school, he pursued his undergraduate studies in the Department of Informatics at Universitas Pasundan in 1995 and later majored in Physics at Institut Teknologi Bandung (ITB) in 1997. After completing his bachelor's degree, he gained professional experience as an information technology and software development trainer.Since 2008, he has been a lecturer in Computer Science at Universitas Lambung Mangkurat. He completed his master's degree in Informatics at Institut Teknologi Bandung in 2010. In 2015, he earned a doctoral degree in Bioinformatics from Kanazawa University, Japan. He continues to serve as a lecturer in Computer Science at Universitas Lambung Mangkurat. His research interests include Data Science, Software Engineering, and Bioinformatics.

Email: reza.faisal@ulm.ac.id

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351

Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics e-ISSN: 2656-8624
Homepage: https://ijeeemi.org/; Vol. 8, No. 3, pp. 351-364, August 2026

Radityo Adi Nugroho obtained his bachelor's degree in Informatics from the Islamic University of Indonesiain 2004 and completed his Master of Computer Science at Gadjah Mada University in 2007. He currently holds the position of Associate Professor in the Computer Science program at Lambung Mangkurat University. In addition to his academic role, he has extensive leadership experience, having served as the Head of the Information and Communication Technology Center at Lambung Mangkurat University from 2014 to 2023. His research interests include software defect prediction and computer vision with a focus on improving system reliability and optimizing visual data processing across various applications. He can be contacted via email at radityo.adi@ulm.ac.id

Andi Farmadi is a senior lecturer in the Computer Science program at Lambung Mangkurat University. He has been teaching since 2008 and has served as the Head of the Data Science Lab since 2018. He completed his undergraduate studies at Hasanuddin University and his graduate studies at Bandung Institute of Technology. His research area focuses on Data Science. One of his research projects, along with other researchers, published in the International Conference of Computer and Informatics Engineering (IC2IE), is titled "Hyperparameter tuning using GridSearchCV on the comparison of the activation function of the ELM method to the classification of pneumonia in toddlers," and this research was published in 2021.

Email: andifarmadi@ulm.ac.id

Corresponding author: Setyo Wahyu Saputro, setyo.saputro@ulm.ac.id, Department of Computer Science, Faculty of Mathematics and Natural Science, Lambung Mangkurat University, Banjarbaru, Indonesia.
DOI: https://doi.org/10.35882/ijeeemi.v8i3.351