International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

SOFTWARE DEFECT PREDICTION USING DYNAMIC FILTERING WITH ADAPTIVE NOISE-ENTROPY ALGORITHM

Mr. P. Ramesh[1] , Dr. S. Prasath[2]

[1]Ph.D Research Scholar in Department of Computer Science,

VET Institute of Arts and Science (Co-education) College, Thindal, Erode, Tamil Nadu, India ORCID: 0009-0009-6224-2009

[2]Assistant Professor and Head, Department of Computer Science and Applications, VET Institute of Arts and Science (Co-education) College, Thindal, Erode,Tamil Nadu, India

S^2 ORCID: 0000-0003-2611-0948

Abstract

In software design, software defect prediction is a very important activity because this assists us in identifying at early stages, those modules that contain errors and thus improve the quality of software and lower maintenance costs. Combining Noise-Enhanced Dynamic Filtering (NEDF) as preprocessing and Adaptive Noise-Entropy Feature Selection (ANEFS) as feature selection, generates new approach delivered in this work. In order to enhance the data quality, NEDF technique removes data noise, outliers and software measure distortions using dynamic filtering, smoothing and transformation techniques. Through this approach, ANEFS facilitates effective reduction of features by adaptive balancing of relevance and redundancy of features and entropy-based feature evaluation and consequently removing the least informative features. These are objectively chosen traits directly responsible for improving the efficiency of software defect prediction by basing the models on the most discriminative trait, thus leading to correct identification of faulty modules not causing the confusion caused by irrelevant or redundant measures. By combining the two, the issue of the existing preprocessing and feature selection plans can be circumvented because it provides improved input data and more indicative features. This built-in structure maximizes the potential of software defect prediction consistency and predictability to a large degree, allowing for more realistic decisions about which modules are erroneous. Overall, this study shows that noise-resistant preprocessing and entropy-based adaptive feature selection is a promising avenue of research trying to improve the software defect prediction model.

Keywords: Adaptive Noise-Entropy Feature Selection (ANEFS), Dimensionality Reduction, Noise-Aware Preprocessing, Noise-Enhanced Dynamic Filtering (NEDF), Software Quality Assurance, Software Defect Prediction

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

I.Introduction

The purpose of this paper is to improve the software defect prediction with using advanced preprocessing and features selection method to improve the accuracy and classification. Other studies have employed enhanced cross-project prediction [1], combination of feature selection and balancing practices [2], comparative distance measures [3] and model averaging practices [4]. Optimization-based, deep learning model-based and ensemble machine learning model-based approaches to defect classification have been explored and refined using balancing techniques [6] and [7], respectively. The intelligent testing systems have also enabled flawed prediction [8]. More advanced semantic and sequence models such as BiLSTM and BERT have demonstrated improved performance [9], and nested-stacking feature selection as well as heterogeneous feature selection is under investigation with strong performance [10].

Application-specific and model-specific contributions are emphasized in empirical assessments of healthcare applications [11], SMOTE-based balancing experiments [12], and attention-based GRU-LSTM designs [13]. This field is also extended to cloud-based prediction with decision-level fusion [14], hybrid PC-SVM methods [15], and analyses of feature collection practices [16]. Predicting vulnerabilities using transformers [17], AI-enabled collaboration platforms [18], and smart test automation [19] have a high potential to be put into practice. Its applicability is further broadened by practical defect and vulnerability prediction tools [20], by source code analysis that is done automatically, [21], and through automatic vulnerability discovery via ML and data-mining [22]. The challenges that have been addressed through PCA-based detection [23] and adaptive interference cancellation [24] include security and IoT-related challenges. Classical methods like non-negative matrix factorization have been applied as well [25]. Combination of SMOTE, RFE, and random forest feature importance analysis have also shed light on successful prediction [26]. Fuzzy mutual information with ant colony optimization to optimize feature selection has been suggested [27], whereas one-way ANOVA F-test has been used in similar classification problems [28].

Contribution: The key contribution of this work is in used two new methods of software defect prediction a preprocessing method was named NEDF, and a feature selection method was named ANEFS. Preprocessing method works well in noise-reduction and data-quality improvement of the input data and feature selection method makes the advantages that most relevant and material features are preserved to predict defects. Collectively, these techniques improve the performance of classifications, which leads to improved accuracy and reliability over the current methods.

Organization: This paper is organized in the following way. Section 2 provides the background research and work in the field of software defect prediction. Section 3 explains the materials and methods, with proposed preprocessing and feature selection methods. The fourth section talks about the experiment findings and analyzes the results in detail. Section 5

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

closes the paper and identifies potential future directions. At last, references are given at the end.

II. Literature Review

Software defect prediction (SDP) has been a research topic of interest to improve software quality and decrease software maintenance cost. Abdu et al. (2025) suggested a model for predicting defects across software projects that uses reduction and hybridization of software metrics to improve generalization across projects. Also, Febrian et al. (2025) developed a hybrid feature selection and data balancing which led to considerable improvements in defect prediction performance, and Maulidha et al. (2025) developed a distance metrics in KNN with SMOTE for handling class imbalance on defect datasets. Li et al. (2025) investigated within-project and cross-project defect prediction using model averaging approaches, whereas Saraireh et al. (2025) built a model of an adaptive ensemble learning method with a binary white shark optimizer to improve defect classification.

Long et al. (2025) optimized fault prediction by combining MnasNet/LSTM using a flower algorithm. Khleel et al. (2025) examined ensemble-based learning using the NearMiss method for software buggy prediction. Pareek (2025) examined AI-induced software testing using a deep learning view. Along with the traditional models, there is interest in deep learning-based approaches. Uddin et al. (2022) used semantic features based on BiLSTM and BERT to perform more suitable defect prediction. Chen et al. (2022) applied a heterogeneous feature selection in the form of a nested-stacking model in a similar model, which provided a high classification performance. Furthermore, Khan et al. (2021) tested various machine learning solutions on healthcare big data and discussed the need to consider the appropriate model types based on domain-specific defect prediction tasks.

III. Materials and Methods

This research proposed a new software defect prediction framework in order to improve the preprocessing aspect of the data and features of the prediction process. The study proposed two main items which include Noise-Enhanced Dynamic Filtering (NEDF) to reduce noise and stabilize the data, and Adaptive Noise-Entropy Feature Selection (ANEFS) to select the most useful non-redundant features. Therefore, combining these two approaches leads to cleaner data and more stable defect prediction models.

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

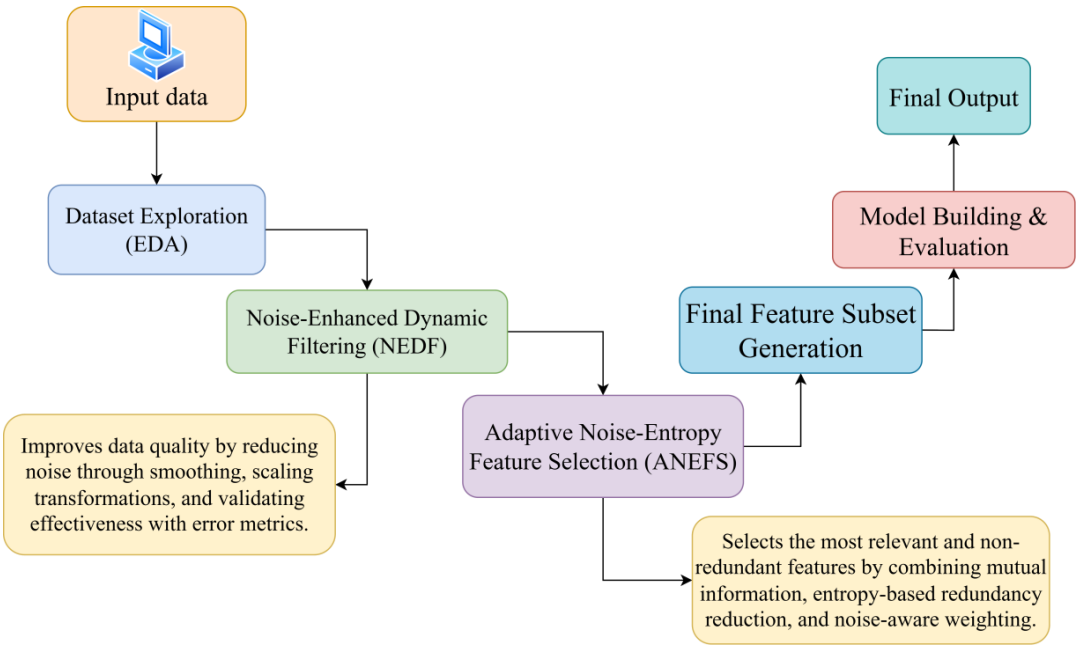


Figure 1.Overall Architecture

The figure 1, proposed architecture when commencing, starts with the exploration of the dataset, before going through the Noise-Enhanced Dynamic Filtering process (NEDF) to be able to improve the quality of the data using a reducing noise, smoothing, and scaling approach, then, passed through the Adaptive Noise-Entropy Feature Selection (ANEFS) algorithm selecting the most appropriate and non-redundant metrics with considerations for mutual information, redundancy reduction and weighting related to noise; and then the refined feature subset which is associated with model constructions and evaluation, yields more valid and trustworthy software defect projections.

3.1 Noise-Enhanced Dynamic Filtering (NEDF)

Noise-Enhanced Dynamic Filtering (NEDF) algorithm is proposed as a pre-processing technique to improve the quality of data through noise elimination and stabilization of feature values. The most important concept of NEDF is to identify and eliminate the impact of random fluctuation of the information, and keep bigger trends related than that of prediction. It begins by identifying noisy attributes where statistical variation is determined using quantile based intervals and attributes that behave in a highly irregular fashion. Rather than erasing these values or blowing them out to a different value, it applies adaptive smoothing to massage the original data back into gear with local averages. It is a way of retaining structural variations that are required, and removing random distortions. This way NEDF improves signal -to-noise ratio and pre-treats a more valid feature space in the future.

Besides the noise processing, NEDF also uses dynamic cluster-level filtering to address anomalies between a set of related records. The values in each feature are normalised to the centroid of a cluster, so it adds local fine-tuning that reduces the drastic outliers, but has no influence on the natural variations between instances. Then there are power transformation

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

and strong scaling, making distribution standard and decreasing the occurrence of outliers, such that there is no single attribute that defines learning process. NEDF minimizes redundancy, suppresses noise, and normalizes the scale; all of which are useful when it comes to creating a high-quality dataset. It is also adaptable such that it can be repeated and used across different kinds of data over time, which ultimately leads to improved predictability and accuracy of software defect.

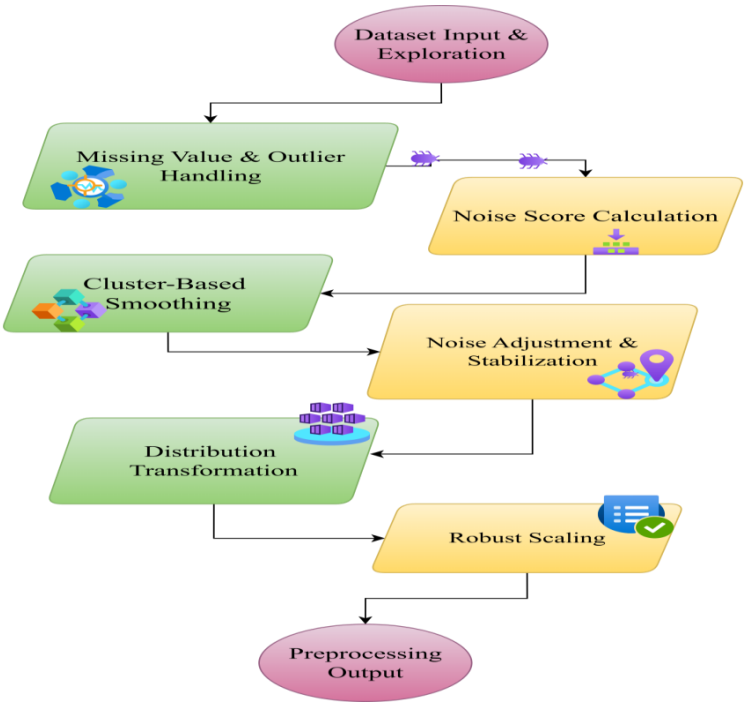


Figure 2.NEDF Workflow

The figure 2 describes the NEDF preprocessing process, where the input of the dataset is processed through exploration, after which missing values, outliers, and noise scoring are handled. Cluster-based smoothing, adjustment, and distribution transformation are then applied to reduce noise and the data is finally enhanced by robust scaling to create high-quality preprocessing output.

$$NoiseScore(f_i) = \frac{1}{B}\sum_{b=1}^{B} Var(f_i^b) \text{ ---------- (1)}$$

The formula 1 calculate the average noise level of feature (f_i). Here, B is the number of quantile bins, f_i^b represents the values of feature f_i in bin b, and (f_i^b) is the variance within that bin. A higher NoiseScore means the feature shows more inconsistency across its distribution.

$$f_i'(r) = 0.9.\, f_i(r) + 0.1.\, \mu_{C,i} \text{ ----------(2)}$$

The formula 2 represents cluster-based smoothing of feature f_i for record r. Here, $f_i(r)$ is the original value of the feature, and $\mu_{C,i}$ is the centroid (mean) of feature i within

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

cluster C. The weights (0.9 and 0.1) ensure that the smoothed value stays close to the original but is slightly adjusted toward the cluster mean to reduce noise.

$$\alpha_i = log\,(1 + \frac{NoiseScore(f_i)}{5}) + 1 \text{---------} (3)$$

The formula 3 represent the noise adjustment factor for feature f_i. Here, $NoiseScore(f_i)$ is the average variance-based noise of the feature, the division by 5 normalizes the score, and the logarithm ensures smoother scaling. Adding 1 prevents zero or negative values, making α_i stable adjustment factor for reducing noise influence.

$$f_i^{final} = min(max(f_i'', -12), 12) \text{----------} (4)$$

The formula 4 applies a clipping rule to the adjusted feature value f_i''. Here, $max(f_i'', -12)$ ensures that the feature value does not go below -12, and $min(.\,, 12)$ ensures it does not exceed 12 . This prevents extreme outliers from dominating the dataset while preserving meaningful variations.

$$f_i^{scaled} = \frac{f_i - Median(f_i)}{IQR(f_i)} \text{----------} (5)$$

The formula 5 represents Robust Scaling of feature f_i. Here, $Median(f_i)$ centers the feature around zero, while $IQR(f_i) = Q_3 - Q_1$ (interquartile range) scales it based on spread. This makes the feature less sensitive to outliers compared to standard scaling.

Algorithm 1: Noise-Enhanced Dynamic Filtering (NEDF)

Input: Dataset D with features F and label DEFECT_LABEL

Output: Preprocessed dataset X_final_scaled, original features X, labels y, error metrics

Begin

Load dataset D from "dataset.csv"

Exploratory Data Analysis.

Function NEDF_preprocessing(D, label_col):

X ← D without DEFECT_LABEL

y ← D[DEFECT_LABEL]

For each feature fi in X:

Replace missing values with median(fi)

Clip fi values within [0.5%, 99.5% quantiles]

For each feature fi in X:

Divide fi into 10 quantile bins

Compute variance in each bin

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

```
        noise_score[fi] ← mean variance
    Apply KMeans clustering (k=5) on X
    For each cluster C:
        For each feature fi:
            centroid ← mean(fi in cluster C)
            Smooth fi[r] ← 0.9*fi[r] + 0.1*centroid
    Apply Yeo-Johnson power transformation → X_transformed
    For each feature fi in X_transformed:
        noise_factor ← noise_score[fi] (if 0 then 1)
        adjusted_factor ← log1p(noise_factor/5) + 1
        fi ← fi / adjusted_factor
        Clip fi within [-12, 12]
    Apply RobustScaler → X_final_scaled
    Return X, X_final_scaled, y
End Function
Function evaluate_preprocessing_errors(original_X, processed_X):
    common_features ← intersection(original_X, processed_X)
    Compute:
        MSE  ← mean squared error
        RMSE ← sqrt(MSE)
        MAE  ← mean absolute error
        MedAE ← median absolute error
        SMAPE ← symmetric mean absolute percentage error
    Return {MSE, RMSE, MAE, MedAE, SMAPE}
End Function
Main:
    Errors ← evaluate_preprocessing_errors(X_original, X_preprocessed)
    Print "Error Evaluation After NEDF Preprocessing"
    For each metric in errors:
        Print metric and value
```

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

End

The NEDF pseudocode describes a preprocessing algorithm which eliminates noise, smooth's features with the help of clustering, and transforms features with the help of distribution stabilization. It modulates feature values according to noise scores, scales them strongly, and eventually judges the quality of preprocessing by error measures. This will give a cleaner and dependable data to predict defects.

3.2 Adaptive Noise-Entropy Feature Selection (ANEFS)

ANEFS algorithm is used to select the most informative and most discriminative features, as well as to reduce redundancy in a dataset. ANEFS is based on the idea of applying entropy-based relevance measures jointly with adaptive noise control to rank attributes that play an important role in prediction tasks. The algorithm starts with mutual information between each feature and the target variable that gives an idea of the relevance of a feature. Meanwhile, features are quantized into quantile-based bins to better reflect the distributional characteristics of features. Such a discretization allows the assessment of redundancy between features by a pair-wise entropy similarity measure. ANEFS balances feature relevance and redundancy reduction to ensure that the chosen features offer the most predictive information with no duplication of information.

Apart from relevancy and redundancy testing, ANEFS adds an adaptive filtering process to consider the sensitivity of the data to noise. Features that are volatile and whose features evolve with time are dampened by noise-sensitive weights so that variables that cause ambiguity or generate confusions in the model of prediction are unlikely to be admitted and stable ones are regulated at selection. The algorithm advances from the most significant feature to the target, choosing features only which positively contribute to the balance of predictability significance, independence, and noise resistance. Therefore, each next feature introduces new information but without repeating patterns elsewhere. ANEFS acquires immunity to overfitting in the selection procedure when features are subjected to random perturbation as well as resilience in practice when the data itself turns out to be noisy or wrong. The inference is that a very informative but minimal subset of features is derived, enhancing computational efficiency, classification accuracy, and the model's generalization. In software defect prediction, such discriminative subsets of features allow the predictive models to consider only the most discriminative software metrics, thus resulting in improved identification of faulty modules, fewer false alarms, and overall more stable defect prediction outcomes. Therefore, ANEFS is a strong and resistant feature selection technique that offers noise tolerance, entropy-based relevance, and minimum redundancy for the intention of eventually producing feature subsets that are brief yet highly effective for prediction modeling.

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

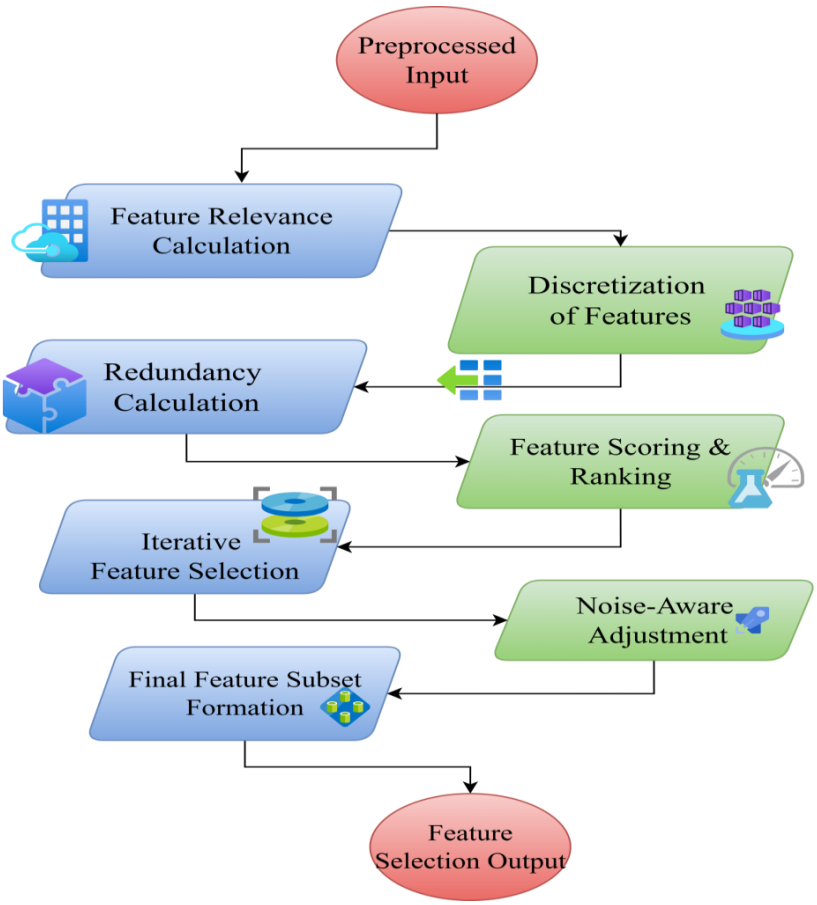


Figure 3: ANEFS Workflow

This figure 3 presents the ANEFS process of feature selection, starting with the input preprocessed data, and computing feature relevance and redundancy. The features are discretized, scored, and ranked with noise-sensitive adjustment then selectively chosen by iteratively eliminating redundant features until a final non-redundant and informative feature subset is obtained.

$Rel(f_i) = I(f_i; Y)$ --------- (6)

This formula 6 computes the mutual information (MI) between a feature f_i and the class label Y. It quantifies how much knowing the feature reduces uncertainty about the label higher MI means the feature is more relevant for prediction.

$d_i = QCut(f_i, q = n\_bins)$ ----------(7)

The formula 7, d_i represents the discretized version of the feature f_i, obtained by applying them $QCut$ function. The parameter $q = n\_bins$ ensures that f_i is divided into equal-frequency bins and each value of f_i are assigned a corresponding bin index in d_i .

$Red(f_i, f_j) = I(d_i; d_j)$----------(8)

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The formula 8 redundancy $Red(f_i, f_j)$ measures the dependency between two features f_i and f_j by calculating the mutual information $I(d_i;\ d_j)$. Here, $d_i$and d_j are the discretized versions of f_i and f_j , and this value indicates how much shared information exists between the two features, helping to detect overlapping or redundant attributes.

$$Score(f_i) = \begin{cases} \frac{Rel(f_i)}{1+\frac{1}{|s|}\sum_{f_j \in S} Rel(f_i, f_j)} \\ Rel(f_i) - \frac{1}{|s|}\sum_{f_j \in S} Rel(f_i, f_j) \end{cases} \text{---------- (9)}$$

The formula 9, feature score $Score(f_i)$evaluates the importance of feature f_i by balancing its relevance $Rel(f_i)$ with the target and its redundancy with already selected features S. The first form $\frac{Rel(f_i)}{1+\frac{1}{|s|}\sum_{f_j \in S} Rel(f_i, f_j)}$ represents the MIQ mode (relevance-to-redundancy ratio), while the second form $Rel(f_i) - \frac{1}{|s|}\sum_{f_j \in S} Rel(f_i, f_j)$ represents the MID mode.

$$Acc = \frac{|s|}{|F|} \times 100 \text{---------- (10)}$$

This formula 10 calculates the feature selection accuracy, where $|s|$ represents the number of selected features and $|F|$ represents the total number of features in the dataset. It shows the percentage of features retained after selection, providing insight into the compactness of the feature subset.

Algorithm 2: Adaptive Noise-Entropy Feature Selection (ANEFS)

Input: Dataset D with features F and label DEFECT_LABEL

Output: Original features X, selected features X_selected, labels y, selected feature list

Begin

Load dataset D

Function ANEFS_feature_selection(D, label_col, fixed_features):

X ← D without DEFECT_LABEL

y ← D[DEFECT_LABEL]

Step 1: Feature relevance

For each feature fi in X:

Compute mutual information (MI) between fi and y

Store in rel_map[fi]

Step 2: Discretization

For each feature fi in X:

Discretize fi into equal-frequency bins

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

```
        Store discretized version
    # Step 3: Redundancy calculation
    For each pair of features (fi, fj):
        Compute MI between discretized fi and fj
        Store in redundancy matrix R[fi, fj]
    # Step 4: Feature selection
    total_features ← number of features in X
    target_count ← min(fixed_features, total_features)
    selected
    Choose first feature with highest relevance
    While |selected| < target_count:
        For each remaining feature f:
            redundancy ← average R[f, s] for s in selected
            relevance  ← rel_map[f]
            score ← relevance - redundancy
        Select feature with maximum score
        Add it to selected list
    # Step 5: Final dataset
    X_selected ← X with selected features only
    Return X, X_selected, y, selected
End Function
Main:
    (X, X_selected, y, feats) ← ANEFS_feature_selection(D, fixed_features=10)
    important_count ← |feats|
    original_count  ← total features in D
    accuracy ← (important_count / original_count) × 100
    Print original_count, important_count, accuracy, feats
    Plot variance of selected features
End
```

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

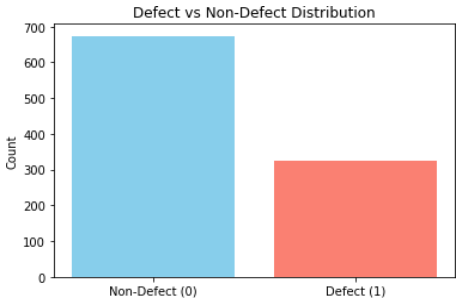
The ANEFS pseudocode describes the process of selecting features based on mutual information to form relevance and entropy-based redundancy reduction features. It begins by calculating the feature importance relative to a target, and then minimizes redundancy between features before finally sampling the most informative subset. This renders the chosen features discriminatory and non-redundant to promote prediction

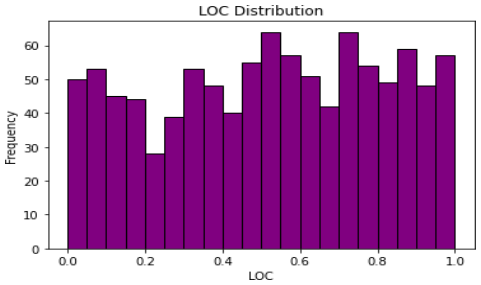
IV. Result and Discussion

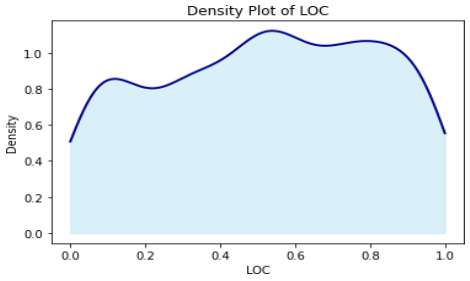
In this result and discussion section, the preprocessing and feature selection are addressed to complement software defect prediction. The Exploratory Data Analysis (EDA) plots were implemented in Python in Spyder environment to describe the information in the dataset, such as distribution of classes, variation of features and their correlation. The NEDF preprocessing method proposed was more effective than the current methods because it produced smaller error values thus giving better data quality. In the same way, the proposed ANEFS feature selection not only retained the most relevant features but also offered the highest accuracy, thus being more effective than traditional methods. On the whole, the findings provide evidence of the effectiveness of the suggested techniques in enhancing prediction.

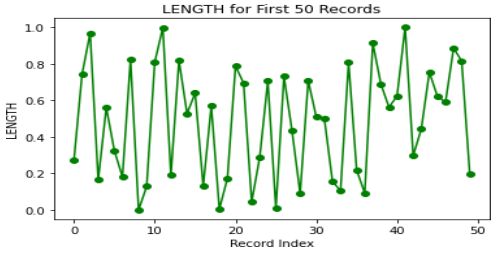
4.1 Preprocessing Result

Preprocessing has been introduced in this result, where Exploratory Data Analysis (EDA) plots have been utilized to explain clearly the dataset information in the form of class distribution, variations of features and correlations. Subsequently, the NEDF method is proposed and its effectiveness is illustrated with the help of comparative plots of the MSE, RMSE, MAE and MedAE, which indicate that NEDF is always superior to the other methods.









Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

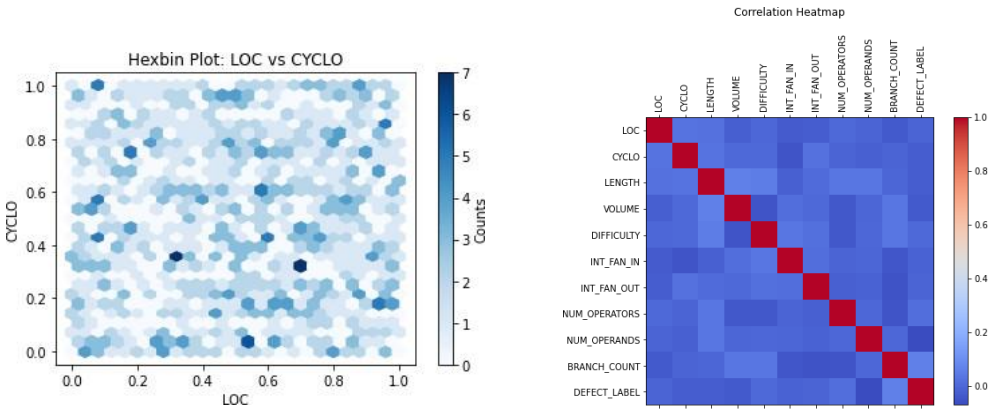


Figure 4: Exploratory Data Analysis Results

This figure 4 is the results of Exploratory Data Analysis (EDA) on the dataset. It contains allocation of classes, feature allocations like LOC (Lines of Code), CYCLO (Cyclomatic Complexity) and LENGTH, and correlation between features including LOC, CYCLO and general feature correlation. Such visualizations can be applied in understanding the ratio between defect and non-defect cases, variation in features and inter-feature relationships.

Table 1: Performance Comparison of Preprocessing Techniques

Method	MSE	RMSE	MAE	MedAE
NMF[25]	1.48	0.84	0.61	0.63
ICA[24]	4.08	2.02	1.60	1.35
PCA[23]	0.82	0.90	0.72	0.60
NEDF (proposed)	0.36	0.60	0.53	0.51

In this table 1, the various methods of preprocessing are compared in terms of error measures like MSE, RMSE, MAE and MedAE. The proposed NEDF has the smallest error values, meaning it better handles noise and has a higher quality of data. On the other hand, existing algorithms such as Independent Component Analysis (ICA), Principal Component Analysis (PCA) and Non-negative Matrix Factorization (NMF) have relatively high errors suggesting that NEDF is more useful to enhance the preprocessing.

Received: August 08, 2025

International Journal of Applied Mathematics

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

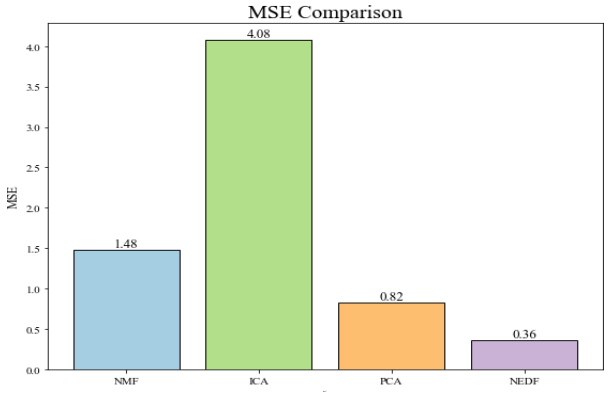


Figure 5 MSE Comparison

This figure 5 compares the values of MSE of four methods NMF, ICA, PCA and NEDF. Among them, the error of ICA is the highest and consequently, worse performance. Errors of PCA and NMF are moderate and the MSE of the proposed NEDF is the least. It proves that NEDF erases errors better, and improves data quality.

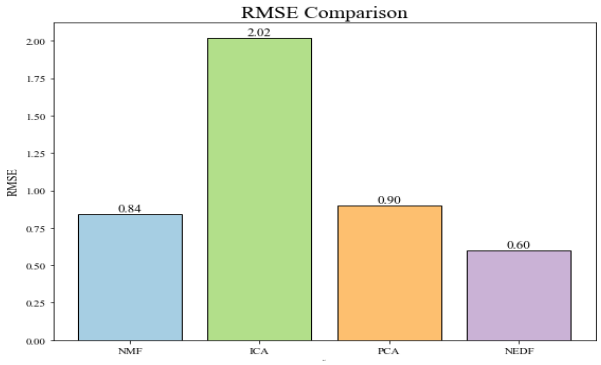


Figure 6: RMSE Comparison

The comparison of various methods in figure 6 is presented in this plot with RMSE. It is found that the best error value is poor in ICA and moderate in PCA and NMF. The proposed NEDF proves to be the most efficient at minimizing errors as it is more efficient than all methods that achieve minimum RMSE.

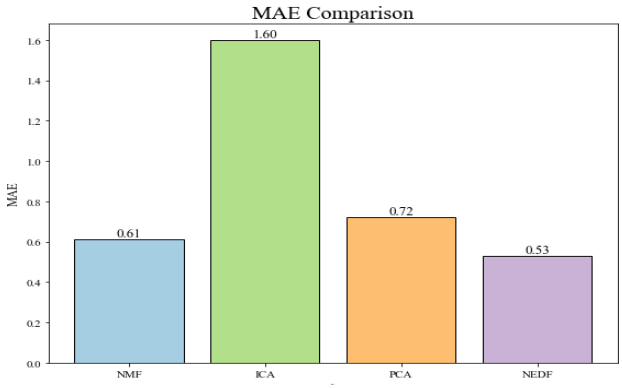


Figure 7: MAE Comparison

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 7 is a comparison of the different methods of MAE. The error of ICA is the largest, which reflects the low performance, and the error values of NMF and PCA are lower. The performance is also consistent with the proposed NEDF having the lowest MAE.

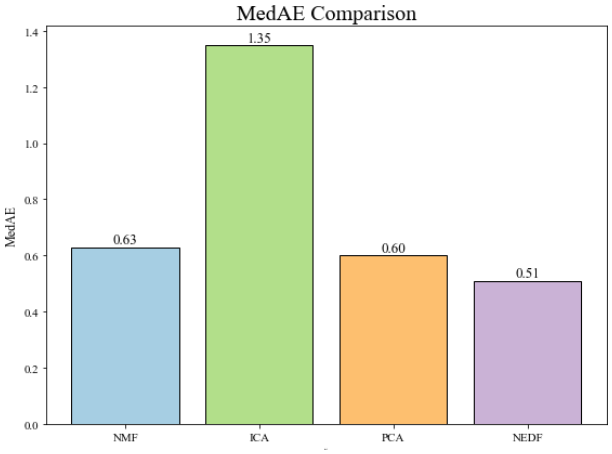


Figure 8: MedAE Comparison

This figures 8 comparing MedAE values of the different methods. ICA demonstrates the worst error, which means that the performance of this algorithm is less reliable, and NMF and PCA provide the average results. The proposed NEDF has the lowest MedAE that demonstrates that it is the strongest and can reduce the median prediction errors.

4.2 Feature Selection Result

In this section, the results of feature selection analysis with the various methods are provided. In comparison of accuracy and the number of selected features, the results demonstrate that the proposed ANEFS approach not only maintains the most important features but also attains the highest accuracy which is highly effective compared to the traditional method.

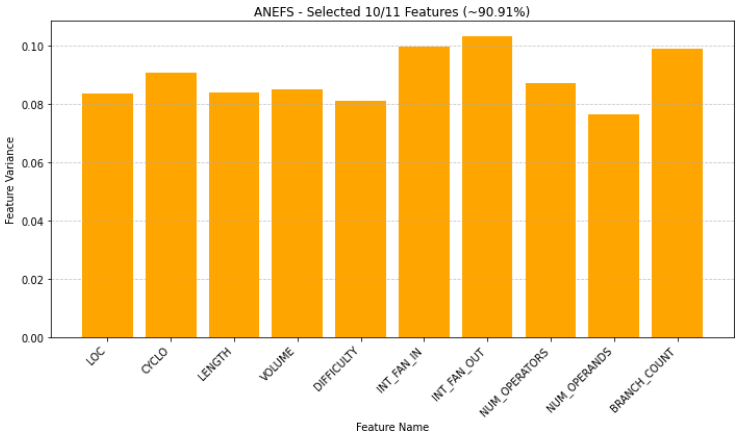


Figure 9: Important Features

This figure 9 indicates the significant characteristics as chosen by the ANEFS approach, and 10 of 11 characteristics. The selected features can be classified into features with higher

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

variance such as CYCLO, BRANCH_COUNT, INT_FAN_OUT, and all the features selected can be used to predict defects significantly.

Table 2: Performance Evaluation of Feature Selection Methods

Method	Accuracy	Total Features	Selected Features
RFE[26]	79.65	11	07
Mutual Information[27]	81.82	11	09
ANOVA F-test[28]	67.00	11	06
ANEFS (Proposed)	90.91	11	10

In this table 2, various methods of feature selection are compared according to their accuracy and the number of features selected. Though the current procedure is performing average, the suggested procedure is more effective as it picks the most useful features and attains the highest overall precision.

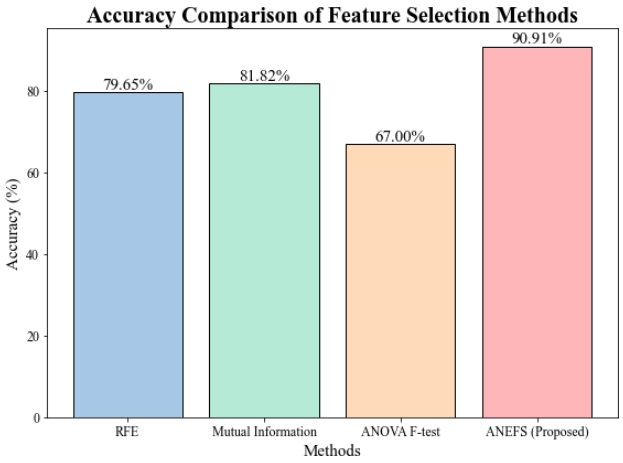


Figure 10: Accuracy Comparison

This figure 10 is a comparison with the process of various feature selection techniques. The one that comes out with the best accuracy among them is the proposed ANEFS and the

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

lowest accuracy is ANOVA F-test. The findings show that ANEFS is more successful in improving model performance than the other methods.

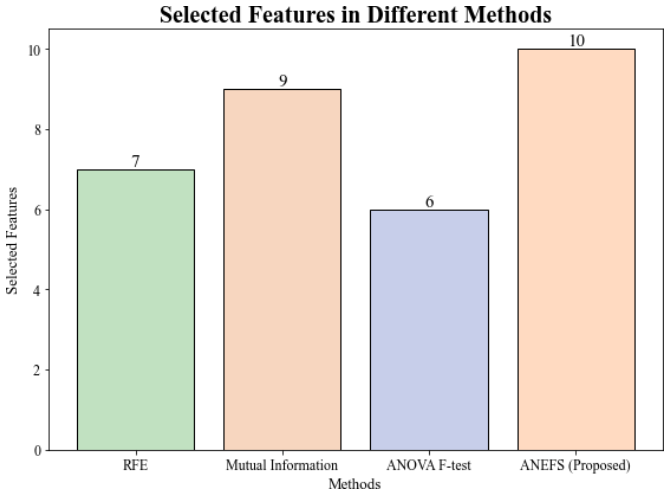


Figure 11: Selected Features

This figure 11 shows the number of the selected features with the various feature selection methods. The proposed ANEFS uses the maximum number of relevant features and ANOVA F-test uses the minimum number. It demonstrates that ANEFS can conserve significant features in order to learn a model better.

V. Conclusion

Software defect prediction is a significant activity in the software development lifecycle because it can detect erroneous modules at an early point within the product development process, minimize maintenance expense, and maximize reliability. Existing methodologies are often plagued by noisy data, decaying feature distributions, and overlapping features that result in deception of prediction models. In order to get rid of such issues, this approach is a two-stage process where Noise-Enhanced Dynamic Filtering (NEDF) is taken first as the pre-processing technique to get rid of missing values, outliers, and noise scores by cluster-based correction and then followed by noise stabilization, power transformation, and robust scaling in subsequent steps in order to have a cleaner and stable data. In the second most discriminative, most informative feature selection, Adaptive Noise-Entropy Feature Selection (ANEFS) is used to select most discriminative, most informative features that employ entropy-based relevance, redundancy reduction by mutual information and noise-weighted redundancy reduction in a way that consistent instead of redundant features are preserved. Features chosen by ANEFS are extremely significant in enhancing the effectiveness of software defect prediction since they enable the models to focus less on the worthless features, thus enhancing the quality of classification, precision, and recall and removing the confusing effect of worthless measures. Collectively, NEDF and ANEFS can provide good-quality feature space that can be utilized to improve the reliability, stability, and overall generalization of software defect prediction models for diverse software datasets, and as the

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

second step, the filtered features would be utilized to perform a classification, thereby transforming the improved preprocessing and feature selection into efficient and implementable software defect prediction.

Reference

[1]. Abdu, A., Zhai, Z., Abdo, H. A., Lee, S., Al-masni, M. A., Gu, Y. H., & Algabri, R. (2025). Cross-project software defect prediction based on the reduction and hybridization of software metrics. *Alexandria Engineering Journal*, *112*, 161-176.

[2]. Febrian, M. M., Saputro, S. W., Saragih, T. H., Abadi, F., & Herteno, R. (2025). Hybrid Feature Selection and Balancing Data Approach for Improved Software Defect Prediction. *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *7*(2), 232-244.

[3]. Maulidha, K. R., Faisal, M. R., Saputro, S. W., Abadi, F., Nugrahadi, D. T., Adi, P. D. P., & Hariyady, H. (2025). Comparative analysis of distance metrics in KNN and smote algorithms for software defect prediction. *Telematika*, *18*(1), 1-12.

[4]. Li, T., Wang, Z., & Shi, P. (2025). Within-project and cross-project defect prediction based on model averaging. *Scientific Reports*, *15*(1), 6390.

[5]. Saraireh, J., Agoyi, M., & Kassaymeh, S. (2025). Adaptive ensemble learning model-based binary white shark optimizer for software defect classification. *International Journal of Computational Intelligence Systems*, *18*(1), 14.

[6]. Long, W., Qixin, Z., Zakharov, M. A., & Lee, S. (2025). Optimizing fault prediction in software based on MnasNet/LSTM optimized by an improved lotus flower algorithm. *Egyptian Informatics Journal*, *29*, 100623.

[7]. Khleel, N. A. A., Nehéz, K., Fadulalla, M., & Hisaen, A. (2025). Ensemble-Based Machine Learning Algorithms Combined with Near Miss Method for Software Bug Prediction. *International Journal of Networked and Distributed Computing*, *13*(1), 11.

[8]. Pareek, C. S. (2025). AI-Driven Software Testing: A Deep Learning Perspective. *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, *13*(1), 1-10.

[9]. Uddin, M. N., Li, B., Ali, Z., Kefalas, P., Khan, I., & Zada, I. (2022). Software defect prediction employing BiLSTM and BERT-based semantic feature. *Soft Computing*, *26*(16), 7877-7891.

[10]. Chen, L. Q., Wang, C., & Song, S. L. (2022). Software defect prediction based on nested-stacking and heterogeneous feature selection. *Complex & Intelligent Systems*, *8*(4), 3333-3348.

[11]. Khan, B., Naseem, R., Shah, M. A., Wakil, K., Khan, A., Uddin, M. I., & Mahmoud, M. (2021). Software defect prediction for healthcare big data: an empirical evaluation of machine learning techniques. *Journal of Healthcare Engineering*, *2021*(1), 8899263.

[12]. Feng, S., Keung, J., Yu, X., Xiao, Y., & Zhang, M. (2021). Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction. *Information and Software Technology*, *139*, 106662.

Received: August 08, 2025

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[13]. Munir, H. S., Ren, S., Mustafa, M., Siddique, C. N., & Qayyum, S. (2021). Attention based GRU-LSTM for software defect prediction. *Plos one*, *16*(3), e0247444.

[14]. Aftab, S., Abbas, S., Ghazal, T. M., Ahmad, M., Hamadi, H. A., Yeun, C. Y., & Khan, M. A. (2023). A cloud-based software defect prediction system using data and decision-level machine learning fusion. *Mathematics*, *11*(3), 632.

[15]. Mustaqeem, M., & Saqib, M. (2021). Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection. *Cluster Computing*, *24*(3), 2581-2595.

[16]. Herbold, S., Trautsch, A., Trautsch, F., & Ledel, B. (2022). Problems with SZZ and features: An empirical study of the state of practice of defect prediction data collection. *Empirical Software Engineering*, *27*(2), 42.

[17]. Fu, M., & Tantithamthavorn, C. (2022, May). Linevul: A transformer-based line-level vulnerability prediction. In *Proceedings of the 19th International Conference on Mining Software Repositories* (pp. 608-620).

[18]. Bakhsh, M. M., Joy, M. S. A., & Alam, G. T. (2024). Revolutionizing BA-QA Team Dynamics: AI-Driven Collaboration Platforms for Accelerated Software Quality in the US Market. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, *7*(01), 63-76.

[19]. Nama, P., Meka, N. H. S., & Pattanayak, N. S. (2021). Leveraging machine learning for intelligent test automation: Enhancing efficiency and accuracy in software testing. *International Journal of Science and Research Archive*, *3*(01), 152-162.

[20]. Fu, M., Tantithamthavorn, C., Le, T., Kume, Y., Nguyen, V., Phung, D., & Grundy, J. (2024). Aibughunter: A practical tool for predicting, classifying and repairing software vulnerabilities. *Empirical Software Engineering*, *29*(1), 4.

[21]. Laaber, C., Basmaci, M., & Salza, P. (2021). Predicting unstable software benchmarks using static source code features. *Empirical Software Engineering*, *26*(6), 114.

[22]. Shah, I. A., Rajper, S., & ZamanJhanjhi, N. (2021). Using ML and data-mining techniques in automatic vulnerability software discovery. *International Journal of Advanced Trends in Computer Science and Engineering*, *10*(3).

[23]. Dash, S. K., Dash, S., Mahapatra, S., Mohanty, S. N., Khan, M. I., Medani, M., ... & Gupta, M. (2024). Enhancing DDoS attack detection in IoT using PCA. *Egyptian Informatics Journal*, *25*, 100450.

[24]. Duan, H., Zhu, X., Jiang, Y., Wei, Z., & Sun, S. (2019). An adaptive self-interference cancelation/utilization and ICA-assisted semi-blind full-duplex relay system for LLHR IoT. *IEEE Internet of Things Journal*, *7*(3), 2263-2276.\

[25]. Chang, R., Mu, X., & Zhang, L. (2011). Software Defect Prediction Using Non-Negative Matrix Factorization. *J. Softw.*, *6*(11), 2114-2120.

[26]. Ghinaya, H., Herteno, R., Faisal, M. R., Farmadi, A., & Indriani, F. (2024). Analysis of Important Features in Software Defect Prediction Using Synthetic Minority Oversampling Techniques (SMOTE), Recursive Feature Elimination (RFE) and

Volume 38 No. 5s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Random Forest. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *6*(3), 276-288.

[27]. Manivasagam, G., & Gunasundari, R. (2018). An optimized feature selection using fuzzy mutual information based ant colony optimization for software defect prediction. *International Journal of Engineering & Technology*, *7*(1.1), 456-460.

[28]. Elssied, N. O. F., Ibrahim, O., & Osman, A. H. (2014). A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology*, *7*(3), 625-638.

Received: August 08, 2025