INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Software Defect Prediction Using Fuzzy Entropy Based Short Term Memory

M Prashanthi[1]*, M Chandra Mohan[2], N Pothanna[3], G Ramesh Chandra[4]

1 Research Scholar Jntuh, CMR Engineering College, Hyderabad, Telangana, India
2 Department of CSE, Jntuh, Hyderabad, Telangana, India
3 Department of Mathematics, Valluripalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana, India
4 Department of CSE, Valluripalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, Telangana, India

OPEN ACCESS

Received: 08-08-2024

Accepted: 08-05-2025

Published: 30-05-2025

Editor: Special Issue Editor: Dr. N. Pothanna, Dr.T. Jayashree, Dr. V. Ganesh Kumar

Citation: Prashanthi M, Mohan MC, Pothanna N, Chandra GR (2025) Software Defect Prediction Using Fuzzy Entropy Based Short Term Memory. Indian Journal of Science and Technology 18(SP1): 1-8. https://doi.org/10.17485/IJST/v18si1.icamada1

* **Corresponding author**.

prashanthi.m@cmrec.ac.in

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objective: To detect software defects with minimum computational complexity, the Fuzzy Entropy Based Short Term Memory (FESTM) model is proposed. **Methods:** The FESTM is a combination of collaborative entropies of fuzzy entropy, condition entropy as well as clustering entropy, and long short-term memory with two parallel layers (BiLSTM) where multi-entropy deals in decision-making with uncertain or imprecise data and BiLSTM increases the performance by enabling additional training to the model due to the passing of input data twice to the LSTM Layers. **Findings:** The comparative analysis of the model with the existing model is accomplished based on performance metrics such as accuracy is 98.99%, F-measure is 99.42%, sensitivity is 98.49%, and specificity is 99.47%. **Novelty:** An appropriate outcome to detect the defects in software is obtained by the proposed model with low computational cost (mention quantitatively). The collaboration of condition entropy, fuzzy entropy, and clustering entropy for feature selection serves better features with several advantages such as removing duplication in features and selecting the features that are suitable for the prediction process.

Keywords: Software Defects Prediction; PROMISE Dataset; Tunable BiLSTM; Multi-Entropy-Based Feature Selection; Clustering Entropy

1 Introduction

Software development follows a structured process to ensure the production of high-quality and reliable applications. However, deviations in software requirements or inconsistencies in code often lead to software defects, which can result in failures, increased maintenance costs, and reduced user trust. Mitigate these issues, defect prediction models are widely used to analyse software code and identify potential vulnerabilities before deployment. Traditional Software Defect Prediction (SDP) techniques rely on static code analysis, unit testing, and defect-prone module identification. However, existing approaches face several challenges, including class imbalance, feature redundancy, and the inability to generalize across different datasets and programming environments.

Recent advancements in Machine Learning (ML) and Deep Learning (DL) have significantly improved software defect prediction. Various models, such as BERT-based semantic feature learning (SDP-BB), deep learning frameworks like LSTM and BiLSTM, and contrastive learning-based models, have been proposed to enhance defect detection accuracy. Despite these efforts, current methods suffer from limitations such as overfitting on large datasets, high computational costs, insufficient memory allocation, and difficulty in capturing contextual relationships within the code. Furthermore, many existing models do not effectively address feature selection, leading to redundant and irrelevant attributes affecting predictive performance.

A critical gap in the literature lies in the effective handling of feature selection and computational efficiency in software defect prediction. Most existing models either fail to eliminate redundant features or lack a comprehensive approach to selecting the most relevant features. Additionally, while some methods perform well on specific datasets, they struggle to generalize across different environments due to class imbalance and asymmetric data distribution.

To address these challenges, this research introduces a Feature Selection-based Software Defect Prediction Model (FESTM) that integrates multi-entropy-based feature selection. The proposed approach utilizes clustering entropy, conditional entropy, and fuzzy entropy to refine feature selection, thereby improving classification accuracy and reducing computational complexity. The identification of code changes and defects is critical because even a minor code issue can impact the entire software framework. Therefore, testers employ Software Change Impact Analysis, allowing efficient and low-cost testing by evaluating the effect of code changes on the system. The collection of existing defect data, combined with Machine Learning (ML) or Deep Learning (DL) techniques—such as regression or segmentation models enables the training of predictive models using historical software defect data. The key contributions of this study are as follows:

The literature presents a range of models for software defect prediction, each attempting to address specific limitations of traditional approaches: Sahar et al.[1] proposed DP-CCL, a model that uses CodeBERT with contrastive learning but faced overfitting issues due to sample duplication. Tao et al.[2] developed TLSTM, a transfer learning-based method for cross-project defect prediction. Zhang et al.[3] introduced a Deep Q-learning approach for feature extraction and classification. The PROMISE dataset[4] serves as a benchmark dataset commonly used for training and evaluating software defect prediction models. Dhiyanesh B et al.[5] Uses Neuro-Fuzzy Logistic Regression with cloud computing to analyse healthcare data efficiently. Tang et al.[6] proposed an ensemble learning algorithm based on the Sparrow Search Algorithm to enhance model generalization. Despite these advancements, many models continue to struggle with redundant features, overfitting, computational inefficiencies, and poor generalization across different projects. The production of software follows a stepwise strategy to ensure the development of high-quality and reliable applications. Software Defect Prediction (SDP) plays a critical role in this process by identifying defective modules early in the development cycle, thereby preventing the occurrence of major software defects[7]. SDP facilitates the outflow of defect-related information by leveraging software module metadata during the design and development phases. Through the analysis of metadata, SDP identifies and categorizes defects such as the number of defects, defect credibility, and outcome segmentation, thus enabling efficient prediction and ranking of software module reliability[8]. Uddin et al.[9] introduced SDP-BB, a BiLSTM combined with a BERT-based semantic feature model, which improves contextual understanding but still suffers from overfitting on large datasets. Izadi et al.[10] combined RoBERTa embeddings with a Random Forest classifier to overcome contextual limitations, though their model lacked support across multiple programming environments. Revathi A et al.[11] Used MEL, BARK, and ERB scale features with vector quantization (VQ) for robust speaker recognition. Mehta S et al.[12] Used PLS-RFE for feature reduction and SMOTE to balance software defect data. Yang H et al.[13] Proposed a slow feature extraction method using myTICA and Gabor functions. Kernel PCA with ESMO to detect and diagnose faults in grid-connected doubly fed induction generators (DFIGs) for better performance[14]. Developed a hybrid software defect prediction model using LSTM and word embedding techniques[15]. Proposed SL Deep, a deep learning model that predicts software defects at the statement level using static code features[16]. Software defects are defined as divergences from the intended software statements or prerequisites, often leading to errors or undefined outcomes[17]. Prediction models assist in identifying defect-prone code, with training and testing processes relying on latent defects prior to software deployment.

To address these limitations, this study introduces the **Feature Selection-based Software Defect Prediction Model (FESTM)**. FESTM aims to deliver high defect prediction accuracy while reducing overfitting and improving efficiency by using optimized feature sets. It stands out as a robust and scalable solution capable of handling class imbalance, feature redundancy, and dataset-specific biases in software defect prediction.

FESTM incorporates a novel multi-entropy-based feature selection mechanism, integrating: Fuzzy Entropy, Conditional Entropy, Clustering Entropy. Each entropy measure serves a specific purpose, such as eliminating redundant features and selecting the most relevant ones for accurate prediction. This approach significantly enhances model generalizability and reduces computational costs. Key Contributions:

• **Multi-Entropy-Based Feature Selection**: Combines fuzzy entropy, conditional entropy, and clustering entropy to refine feature sets, remove duplication, and select features most suitable for defect prediction.

• **Software Defect Prediction Using FESTM**: Overcomes the shortcomings of previous methods and provides accurate defect prediction at a lower computational cost.

Thus, the FESTM model offers a promising solution to longstanding challenges in software defect prediction, ensuring more reliable, efficient, and scalable software development practices.

2 Methodology

To predict the software defects, FESTM is proposed in this research. The software-related data is fed into the defect prediction model for which the PROMISE dataset is employed for achieving accurate outcomes. The data from the dataset is fed into the multi-entropy-based feature selection process, where the modules are selected from the dataset without another interaction. The specific features are utilized to indicate, whether software is defective or not, which is identified continuously with minimum computational complexity. The feature vector is moved into the module for predicting the defects in software, where the proposed tunable DL classifier named FESTM is utilized to achieve efficient result.

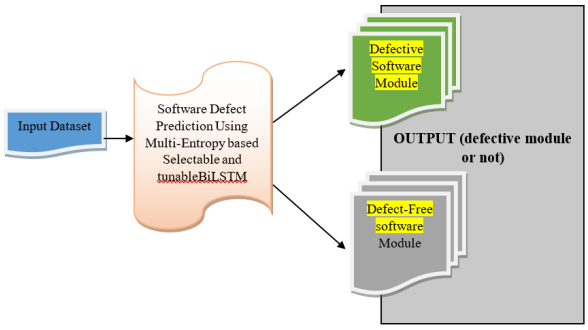

Fig 1. Block diagram of FESTM

2.1 Experimental setup:

PYTHON software is used as an implementation tool in Windows 11 with 32 GB RAM storage.

2.2 Dataset Description:

PROMISE Dataset: The two types of variables such as independent and dependent, where 2000 metrics are considered as independent and dependent are defect-proneness of the class utilized from the PROMISE dataset in the research.

2.3 Input:

The data collected from the PROMISE dataset [4] are considered as input to the software defect prediction model. The input I_l is.

$$S = (I_1, I_2, ...I_l, ...I_e\}$$

Where, the dataset is expressed as S comprising e, a number of software-related data.

2.4 Multi entropy-based feature selection:

Multi entropy-based feature selection is used to pick out the qualified features of data from the dataset. The process of combining three-entropy makes the feature selection process simpler, and efficient. Thus, the utilized entropies are listed and explained below,

2.4.1 . Conditional Entropy:

Conditional entropy originated to increase the segmentation performance of gene voicing datasets based on the fuzzy information granule

$L(S) = -\sum_{S=1}^{e} \frac{((I_l] \cap D|}{(S|} log \frac{((I_l] \cap D|}{((I_l]|}$

L(S)=$-\sum_{s=1}^{e} |[I_l] \cap D]|$(2)

where, D, is the decision attribute set that remains in the dataset, and thus the features of the input are selected using the conditional entropy, which is denoted as, $L(S)$.

2.4 .2. Fuzzy Entropy:

Fuzzy entropy is used to express the mathematical form of indistinctness in fuzzy sets and assist the feature selection process. The calculation of fuzzy entropy is estimated as,

$$F_1(S) = -\sum_{I=1}^{e} \eta_S(I_l) log \eta_S(I_l) + (1 - \eta_S(I_l) log(1 - \eta_S(I_l)))$$

Where $\eta_S(I_l)$is used to express the fuzzy values,F, a variable is used to express and define the maximum element in the fuzzy sets, which is the outcome of fuzzy entropy.

2.4 .3. Clustering Entropy:

The clustering entropy is an algorithm that is based on the elucidated distance that connects features in a cluster and expresses the cluster centres between the two feature vectors as an illustration.

$$CR(I_l) = \frac{V(I_l)}{\gamma_{I_l}}$$

where,$V(I_l)$,is used to denote the covariance, γ_{I_l} is used to express the variance of the coefficients. The above-mentioned conditional entropy, clustering entropy, and fuzzy entropy are combined for better feature selection through the average function.

$$ME = \frac{L(S) + F_1(S) + CR(I_l)}{3}$$

This fusion is known as the hybridization of multiple entropy.

2.5 Software Defect Detection using FESTM:

In this research, the FESTM model for predicting software defects by using FESTM model. The BiLSTM is an elongated model based on the LSTM with parallel layers, which is developed to overcome the existing LSTM method issues. The collaboration of two parallel layers of LSTM is the BiLSTM model, one LSTM layer is in a forward direction and another layer in a backward direction specifically each direction accesses separate input values. BiLSTM uses a dense layer for increasing the weight and decreasing the bias values, which convert the dimensional values of input data, the backward and forward propagation.

$$j_r = \partial\left(u_j * ME + p_j * k_{r-1} + bias_j\right)$$

$$q_r = \partial\left(u_q * ME + p_q * k_{r-1} + bias_q\right)$$

$$X_r = \partial\left(u_X * ME + p_X * k_{r-1} + bias_X\right)$$

$$_r = tanh\left(u_g * ME + p_g * k_{r-1} + bias_g\right)$$

$$g_r = X_r * g_{r-1} + j_r * _r$$

$$k_r = q_r * tanh(g_r)$$

where, u and p variable is used to express the weight metrics, $bias$the word denotes the bias vector, ∂ is used to express the LSTM network activation function, k_{r-1} is used to denote the past result values, A_r is the current input values, g_{r-1} it shows the quality information, the input gate.$_r$, is the expression of candidate value which is determined based on the A_r and k_{r-1} by using a $tanK$ function.$j_r q_r$is used to denote the output gate.g_r, is a cell memory,$k_r X_r$, is a forgotgate.

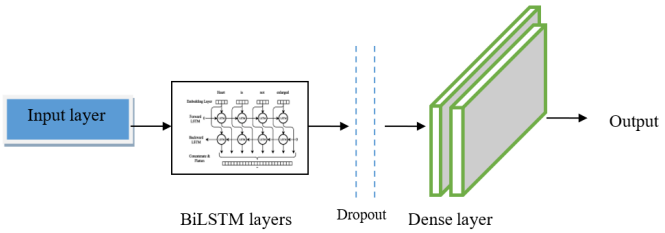

Fig 2. Layers

3 Result and Discussion

The presentation and comparative implementation of the FESTM model for predicting the defects in software are illustrated here.

3.1 Performance metrics

The accuracy is a metric that is used to measure the FESTM model efficiency. The FESTM model performs a true positive rate, which is calculated by the sensitivity metrics, the predicted defect quality is measured by the specificity metrics and the F-measure calculates the prediction performance made by the FESTM.

3.2 Performance Analysis

3.2.1 Performance Analysis with TP

Figure 3 explains the performance implementation of the FESTM model for software defect prediction with TP 90. The FESTM models attain 88.82% in epoch 10 by using accuracy, 92.91% in epoch 20, 93.04% in epoch 30, 96.48% in epoch 40, and 98.88% in epoch 50%. The sensitivity attains 87.69% in epoch 10, 91.78 in epoch 20, 91.90% in epoch 30, 95.35% in epoch 40, and 98.57% in epoch 50 by the FESTM model. The specificity of FEST attains 86.95% in epoch 10, 91.04% in epoch 20, 91.17% in epoch 30, 94.61% in epoch 40, and 99.98% in epoch 50. The F-measure attains 87.45% in epoch 10, 91.54% in epoch 20, 91.66% in epoch 30, 95.11% in epoch 40, and 99.40% in epoch 50 by the FESTM.

3.3 Comparative Methods

The existing models are utilized to compare with the performance of the FSTM model, the traditional models are linear regression model(LR),social spider-deep CNN (SS-deep CNN),Random Forest(RF), and social spider optimization-deep CNN (SSO-deep CNN) , SMO-deep LSTM, Greywolf Optimization deep CNN (GWO deep CNN),Support Vector Machine(SVM),Spider Hunt-Deep CNN, NN,Social spider-NN(S Spider-NN),Grey Wolf Optimization-Deep CNN(GWO-Deep CNN), BFA optimized Deep LSTM, deep CNN, Deep LSTM, Decision tree(DT)

3.3.1 Comparative Analysis with TP:

Figure 4 explains the comparative evolution+ of the FESTM model with the existing approach for TP 90. The accuracy of the FESTM attains 98.99% which is higher than 0.35% by BFA optimized deep LSTM. The sensitivity of the FESTMattains98.49%, which is higher than BFA-optimized deep LSTM by 0.35%. The specificity of the FESTM attains 99.77% which is higher than BFA optimized deep LSTM by 0.15%. The F-Measure attains 99.42% made by FESTM which is higher than BFA-optimized Deep LSTM by 0.50%. Therefore, the effective achievements of the proposed FESTM model were clearly shown by comparing the existing methods. This happened because of reducing the duplicate features by selecting the appropriate features for the prediction process using multi-entropy-based feature selection and also by limiting the existing drawbacks by the usage of BiLSTM.

Table 1 represents the comparative discussion of the FESTM for software defect prediction. The SVM did not have ahigh prediction potential and it required more time to evaluate, the decision tree method made the overfitting problem and did not have the potential to change the imbalanced data, the Social spider-NN method, as well as the spider hunt deep CNN, are used to detect software defects but the method requires more time for execution and expensive method where the proposed FESTM model provides the high potential to predict the defects by eliminating the unwanted features with low computational time. Neural Network and the deep LSTM method are used to predict the software defects but the method lacks in sample data

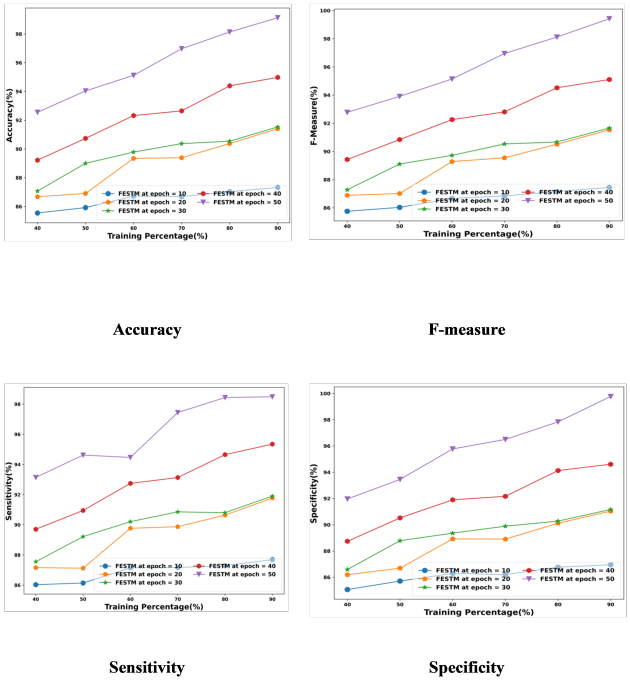

Fig 3. Performance Analysis with TP

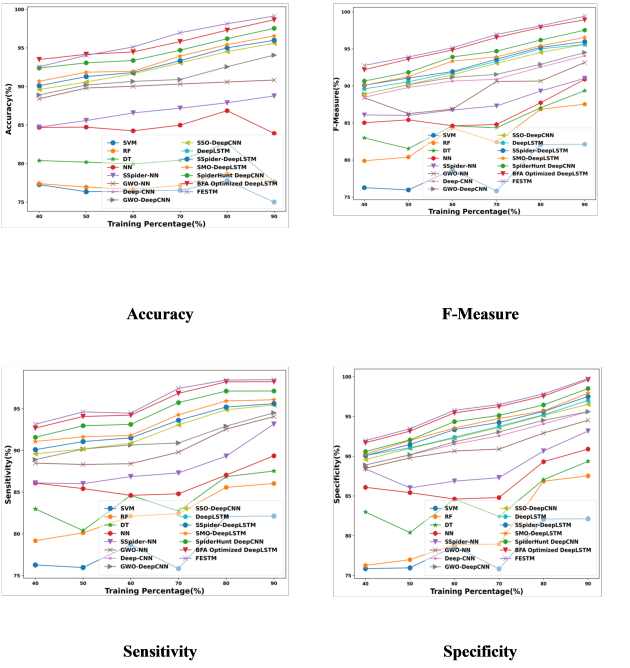

Fig 4. Comparative Analysis with TP

which was rectified using the proposed FESTM by providing a large amount of data. The RF and the GWO-deep CNN method did not have the ability to train and implement the model efficiently whereas the proposed FESTM model can illustrate and train the model efficiently due to the combination of BiLSTM. The achieved percentage of the GWO-NN method is extremely poor whereas the FESTM model has achieved a high percentage. The deep CNN method identified the defect in the software but did not have enough amount of memory to store data which was cleared by providing enough memory space using the FESTM model. The SSO-deep CNN method did not work well in complex situations, the BFA-optimized deep LSTM was very slow as as the implementation strategy of spider monkey optimization deep LSTM is not stable where the proposed FESTM model cleared these drawbacks by working with more stability along with high speed and provides comparatively high accuracy of 98.99%.

4 Conclusion

This research has proposed the FESTM model which offers the appropriate outcomes for defect prediction in software. It utilizes the collaboration of condition entropy, fuzzy entropy, and clustering entropy for accurate feature selection that serves appropriate features by reducing the unwanted features present in it. The feature selection process output is an input of the FESTM model for predicting the defects in software. The BiLSTM model is a combination of two LSTM parallel layers that contain data from both the past and present which produce a valid output by combining the LSTM layers from both forward and backward directions. The FESTM model is used to overcome the existing model issues. The FESTM model helps to predict software defects accurately. The FESTM model is compared with the classical models that express advanced performance and improvement of the FSTM model in terms of accuracy 98.99%, F-Measure 99.42%, specificity 99.77%, and sensitivity 98.49%. In future work, more algorithms will combine with this approach for better performance in software defect prediction.

Acknowledgements

This paper has been presented in "International Conference on Applied Mathematics and Advanced Data Analytics for Industry 5.0 (ICAMADA-2024)" held from 25th to 27th April 2024 at VNR VJIET, in collaboration with APTSMS and CCOE. The authors express their sincere gratitude to the guest editors for their tireless efforts and dedication by means of comments, editing and overseeing the manuscripts to bring in this final form.

The APC is deferred partially by Indian Society for Education and Environment.

References

1) Sahar S, Younas M, Khan MM, Sarwar MU. DP-CCL: A Supervised Contrastive Learning Approach Using CodeBERT Model in Software Defect Prediction. *IEEE Access*. 2024;12:22582–22594. Available from: https://dx.doi.org/10.1109/access.2024.3362896.
2) Tao H, Fu L, Cao Q, Niu X, Chen H, Shang S, et al. Cross-Project Defect Prediction Using Transfer Learning with Long Short-Term Memory Networks. *IET Software*. 2024;(1). Available from: https://dx.doi.org/10.1049/2024/5550801.
3) Zhang Q, Zhang J, Feng T, Xue J, Zhu X, Zhu N, et al. Software Defect Prediction Using Deep Q-Learning Network-Based Feature Extraction. *IET Software*. 2024;p. 1–34. Available from: https://dx.doi.org/10.1049/2024/3946655.
4) Software Defect Prediction Dataset. 2021. Available from: https://figshare.com/articles/dataset/Software_Defect_Prediction_Dataset/13536506?file=25981097.
5) Dhiyanesh B, Rameshkumar M, Karthick K, Radha R. Cloud computing and machine learning for analysis of health care data based on neuro fuzzy logistic regression. *Journal of Intelligent & Fuzzy Systems*. 2023;44(6):9955–9964. Available from: https://dx.doi.org/10.3233/jifs-223280.
6) Tang Y, Dai Q, Yang M, Du T, Chen L. Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm. *International Journal of Machine Learning and Cybernetics*. 2023;14(6):1967–1987. Available from: https://dx.doi.org/10.1007/s13042-022-01740-2.
7) Batool I, Khan TA. Software fault prediction using deep learning techniques. *Software Quality Journal*. 2023;31(4):1241–1280. Available from: https://dx.doi.org/10.1007/s11219-023-09642-4.
8) Kowdiki M, Khaparde A. Adaptive hough transform with optimized deep learning followed by dynamic time warping for hand gesture recognition. *Multimedia Tools and Applications*. 2022;81(2):2095–2126. Available from: https://dx.doi.org/10.1007/s11042-021-11469-9.
9) Uddin MN, Li B, Ali Z, Kefalas P, Khan I, Zada I. Software defect prediction employing BiLSTM and BERT-based semantic feature. *Soft Computing*. 2022;26(16):7877–7891. Available from: https://dx.doi.org/10.1007/s00500-022-06830-5.
10) Izadi M, Akbari K, Heydarnoori A. Predicting the objective and priority of issue reports in software repositories. *Empirical Software Engineering*. 2022;27(50). Available from: https://dx.doi.org/10.1007/s10664-021-10085-3.
11) Revathi A, Nagakrishnan R, Sasikaladevi N. Robust HI and dysarthric speaker recognition – perceptual features and models. *Multimedia Tools and Applications*. 2022;81:8215–8233. Available from: https://dx.doi.org/10.1007/s11042-022-12184-9.
12) Mehta S, Patnaik KS. Improved prediction of software defects using ensemble machine learning techniques. *Neural Computing and Applications*. 2021;33(16):10551–10562. Available from: https://dx.doi.org/10.1007/s00521-021-05811-3.
13) Yang H, Zhao Y, Su G, Liu X, Jin S, Fan H, et al. Slow Feature Extraction Algorithm Based on Visual Selection Consistency Continuity and Its Application. *Traitement du Signal*. 2021;38(6):1599–1611. Available from: https://dx.doi.org/10.18280/ts.380604.
14) Stallon SRD, Rajkumar MN. Improving the performance of grid-connected doubly fed induction generator by fault identification and diagnosis: A kernel <scp>PCA-ESMO</scp> technique. *International Transactions on Electrical Energy Systems*. 2021;31(4). Available from: https://dx.doi.org/10.1002/2050-

7038.12844.
15) Bahaweres RB, Jumral D, Hermadi I, Suroso AI, Arkeman Y. Hybrid Software Defect Prediction Based on LSTM (Long Short Term Memory) and Word Embedding. *2021 2nd International Conference On Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*. 2021;p. 70–75. Available from: https://ieeexplore.ieee.org/document/9617182.
16) Majd A, Vahidi-Asl M, Khalilian A, Poorsarvi-Tehrani P, Haghighi H. SLDeep: Statement-level software defect prediction using deep-learning model on static code features. *Expert Systems with Applications*. 2020;147. Available from: https://dx.doi.org/10.1016/j.eswa.2019.113156.
17) Qiao L, Li X, Umer Q, Guo P. Deep learning based software defect prediction. *Neurocomputing*. 2020;385:100–110. Available from: https://dx.doi.org/10.1016/j.neucom.2019.11.067.