Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Hybrid Machine Learning Model for Software Defect Prediction

[1]Deepak Kumar Nishad, [2]Lakshmi Shanker Singh, [3]Neelam
[1]*Research Scholar, Department of M-tech Computer Science, Sir Chhotu Ram Institute of Engineering and Technology, Chaudhary Charan Singh University, Meerut, UP, India*
Email: deepaknishad833@gmail.com
[2]*Assistant Professor, Department of Information Technology, Sir Chhotu Ram Institute of Engineering and Technology, Chaudhary Charan Singh University, Meerut, UP, India*
Email: Shanker.laxmi@gmail.com
[3]*Assistant Professor, Department of Information Technology, Sir Chhotu Ram Institute of Engineering and Technology, Chaudhary Charan Singh University, Meerut, UP, India*
Email: Neelam.scriet@gmail.com

ARTICLE INFO

Received: 30 Dec 2024

Revised: 05 Feb 2025

Accepted: 25 Feb 2025

ABSTRACT

A portion of software created to fulfill a particular purpose is known as software application. At the same time, engineering is focused on creating goods with specific technical methods and principles. Software defects can be anticipated at several stages, including data input and pre- processing, attribute extrication, and classification. This research study implements multiple classifiers to forecast software defects. This work makes use of several classifiers namely RF (random forest), GNB, Bernoulli NB, and MLP to forecast software faults. The development of an ensemble classifier increases the software fault's reliability. Class balancing and the Principal Component Analysis (PCA) approaches have been combined in the ensemble classifier that is being presented. Python is used for implementing the architecture that has been introduced. Diverse measures are used to examine the findings with regard to universal metrics (i.e. accuracy, precision, and recall).

Keywords: Software Defects, machine learning, PCA, MLP

1. Introduction

A set of executable programming code, related libraries, and documentation is referred to as software. A software product is software designed to meet certain needs [1]. In the meanwhile, clear technical concepts and procedures link engineering to product development. Thus, the engineering field that focuses on developing software products through the application of accepted scientific concepts, methods, and procedures is known as software engineering. The outcome of software engineering is a capable and dependable software product. Software engineering, according to some experts, is the methodical design and creation of software products as well as the supervision of the software development process. Creating programs that satisfy predetermined requirements, are observably accurate, are delivered on schedule, and are under budget is one of the main purposes of Software development. All of the terms used in this description highlight the fact that software development involves more than just assembling a processor, a programming language, and a programmer; it also entails developing a methodology that can satisfy predetermined requirements while taking quality, time, and cost into account. Software engineers typically follow a systematic and structured approach to their work, which is seen to be the most efficient way to create high-quality software products [3].

In addition to being a natural feature of software products, software flaws are also a critical component of software quality. They are an unavoidable consequence of developing software. Furthermore, it takes a lot of effort and time to provide software quality assurance. There are several approaches to defining flaws, most of which are defined concerning of quality [2]. Defects, on the other hand, are usually defined as variations from expectations or specifications that may result in malfunctions. Techniques for predicting software flaws help focus quality assurance efforts on the areas of the code that are most vulnerable to errors.

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

These methods devote greater energy to tackling complicated problems. Finding areas of a software system that may have flaws is the method of DeP (Defect Prediction in Software). A fascinating area of study in software engineering is the prediction of software flaws, or defects. Quality assurance teams can effectively allocate available resources for software product testing and analysis by using lists of software components that are prone to defects, generated by defect prediction models. Black box and white box approaches are the two main types of methodologies used to forecast various kinds of errors. Black box defect prediction techniques forecast future values by using historical metric values. For instance, it is possible to forecast the existence of hidden flaws in the product by looking at the quantity of defects found during the creation of software or testing. Other software features are not used by these models to forecast [3].

In order to anticipate hidden software problems, white box defect prediction approaches combine the "uncovered defects" parameter with other software product attributes. Software defect classification techniques concentrate on classifying the flawed and non-defective components of a software product in addition to forecasting hidden flaws. These methods classify software artifacts by using different aspects of the software product. Software flaws are usually categorized at a lower degree of granularity, such at the file and class levels. Machine learning and logistic regression are two methods used for defect classification [4]. A type of statistics called logistic regression is used to categorize datasets with a number of independent variables in order to provide a result. Depending on the categorization output, which displays one of two possible outcomes, software modules are categorized as either defective or non-defective. Like previous regression algorithms, this technique classifies software parts using metric information. When the likely values of independent variables are 0 and 1, it can be employed in binary classification since it calculates the likelihood that an action will have one of two potential values [5]. This approach is helpful for analyzing data and figuring out how one or more independent variables relate to a dependent binary variable.

Data mining and statistically based algorithms are used by machine learning models to categorize flaws. Based on input prediction factors, these frameworks generate classification results for software parts, indicating whether they are problematic or not. Researchers are increasingly using machine learning methods to categorize software components. The initial step in creating a software defect prediction model is to extract data patterns from software databases, which often comprise issue tracking and version control systems. Source code and commit messages are stored in version control systems, whereas defect information is stored in issue tracking systems [6]. These patterns can represent a variety of granularities, such as a technique, class, source code file, package, or code change. Usually, each pattern is linked to a collection of anticipated defect attributes that are gathered from the software sources. The complexity of the software and its development process reflects in the metric values. Depending on whether or not they have flaws, patterns can be classified as either defective or non-defective. The acquired metrics and labels are then utilized to construct fault prediction models using a set of training patterns. Lastly, using the knowledge it has gained from the training data, the prediction model can determine if a new pattern is flawed or not.

Various popular machine learning models, including Naïve Bayes, SVM and KNN, are commonly used for software fault prediction. Naïve Bayes is well known for producing precise predictions with efficiency. It works especially well for classification issues with several independent variables. When compared to more sophisticated classifiers, this one frequently performs well and can handle big datasets [7]. The Naïve Bayes classifier is predicated on the idea that variables are independent of one another and is based on Bayes' theorem. The Bayes' theorem is a useful tool for computing conditional probabilities. The class with the highest posterior probability is the one predicted by the Naïve Bayes. KNN is a straightforward classifier that considers all possible outcomes. It uses a similarity approach to classify patterns. Pattern classification is determined by a majority vote from neighboring patterns. Each pattern is assigned a class based on the class that receives the most votes among its nearby patterns, typically determined using a distance metric like Euclidean distance. SVM is a supervised learning technique that can be applied to applications involving regression and classification. Its main goal is to find the optimal classification function that can differentiate between the members of two classes in the training data. It is a generalized linear classification technique. SVM aims to increase the geometric margin and reduce classification errors simultaneously. In order to make it simpler to discern between the two classes, it maps input data into a higher- dimensional space and then defines a

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

separating hyperplane. Two parallel hyperplanes are created to quantify the margin on both sides of the separating hyperplane, ensuring that it has the largest distance to the nearest data points from both classes. Maximizing the margin helps reduce generalization errors.

2. Literature Review

M. Nashaat, et al. (2025) introduced a novel architecture for predicting software problems that made use of transformer-based networks with attention mechanisms [8]. In order to create useful representations of software modules, the framework encoded input vectors. Programming languages were modeled using a bidirectional transformer encoder, and then the model was fine-tuned using labeled data to recognize flaws. Experiments on numerous software projects were conducted to evaluate the framework's performance and compare it to baseline methods. Furthermore, an ablation study and statistical hypothesis testing were carried out to evaluate the effects of many parameter selections. The actual results demonstrated that, in comparison to conventional methods, the suggested strategy improved the F1 score by up to 44.26% and enhanced classification accuracy by an average of 15.93%.

Y. Tang, et al. (2025) proposed MOSIG, an instance gravity-based oversampling technique [9]. The first step in this method was to measure the similarity between instances using a new metric called instance gravity. After that, feature models and instance groups were built. Within many instance groups, instances that satisfied particular criteria determined by instance gravity were found. By assigning weights to instances based on their gravity, a new technique for creating faulty instances was put forth. MOSIG considerably improved the prediction performance of the CART decision tree and Naive Bayes models on 21 publicly accessible software fault datasets, according to experimental results. In order to confirm that MOSIG was statistically significant, the experimental data were further validated using the Nemenyi post- hoc test and Friedman ranking.

Yan Zhou et al. (2019) presented KPCA-SVM to predict software faults [10]. The framework first reduces the dimensionality of software defect datasets and then employs the SVM algorithm to classify software defects, addressing the dimensionality issue with KPCA. It was evaluated using the extensive NASA MDP dataset and demonstrated effective handling of data redundancy issues, enhancing global attribute support and achieving high precision.

Y. Jiang, et al. (2024) suggested BiCC-BERT, a new bi-modal change pre-training model. BiCC- BERT learned bi-modal semantic representations by pre-training on a code change corpus [11]. RMI, a new pre-training objective that learned the semantic relationship between commit messages and code changes, was created in order to incorporate commit messages from the corpus. After BiCC-BERT was incorporated into JIT-DP, JIT-BiCC a novel method to defect prediction was proposed. More profound change semantics were captured by JIT-BiCC by using the bi- modal representations from BiCC-BERT. Eight cutting-edge JIT-DP techniques were used to compare the performance of JIT-BiCC, which was trained using 27,391 code changes. With a 10.8% improvement in F1-score, the results demonstrated that JIT-BiCC performed better than all baselines, demonstrating its efficacy in teaching JIT-DP bi-modal semantics.

X. Fan, et al. (2024) proposed S-DCCA, a novel CPDP technique based on SMOTE and DCCA [12]. The problem of non-linear correlations between the attributes of the source and target projects was resolved by using CCA. By using the MlpNet model to extract features from the dataset, S-DCCA expanded on CCA. Then, using the CCA loss function to maximize the associated feature subset, redundant features were removed. Using the SMOTE data sampling technique, cross-project defect prediction was accomplished. Evaluation measures were F1 scores and AUC. To validate the suggested approach, experiments were carried out on 27 projects from four publicly available datasets. The results showed that the suggested strategy had excellent performance characteristics, outperforming all baseline approaches by at least 1.2% in AUC and 5.5% in F1 score on average.

Rituraj Singh et al. (2020) explored TLCV for extracting attributes from software source code text to predict faults [13]. They used a pre-trained DL model to convert the code into vectors, followed by ML and DL techniques to analyze these vectors. Their approach demonstrated superior performance in predicting defects,achieving better weighted F1 scores compared to other methods.

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Aqsa Rahim et al. (2021) proposed a model to predict software defects, consisting of three phases: data preprocessing to eliminate noise and normalize data, correlation-based attribute extraction, and the use of NB and LR algorithms for classification [14]. The model achieved an accuracy of approximately 98.7% with NB and effectively reduced maintenance costs and code complexity while predicting software defects early.

3. Research Methodology

The presented approach is built on various classification algorithms, including RF, GNB, Bernoulli Naïve Bayes, and decision tree DT. An ensemble classifier, which integrates GNB, Bernoulli Naïve Bayes, RF, and MLP, is employed to predict software faults. PCA is used to extricate attributes within the ensemble classifiers with class balancing. Each classification method is detailed as follows: -

3.1 Multilayer Perceptron

Frank Rosenblatt, an esteemed American neurologist, proposed the perceptron in the process of learning, which was motivated by brain nerve cell adaptation principle, also referred to as cell assembly concept of synaptic plasticity. Synaptic plasticity refers to the link or synapse between cells in neuroscience. The initial framework incorporated a supervised learning mechanism that allowed artificial neurons to be adjusted to the right weights determined by training data. A binary perceptron, often referred to as a threshold function, converts an input vector $x \in R^d$ into an output (x). The various elements of the input vector are represented by

$x = [x_1, x_2 \ldots . x_d]^T$. Also, the weights associated with all input elements linked to the perceptron correspond to $w = [w_1, w_2 \ldots . w_d]^T$. Additionally, the perceptron incorporates a constant input with an associated weight b, commonly referred to as the bias. An MLP is illustrated in Figure 3.

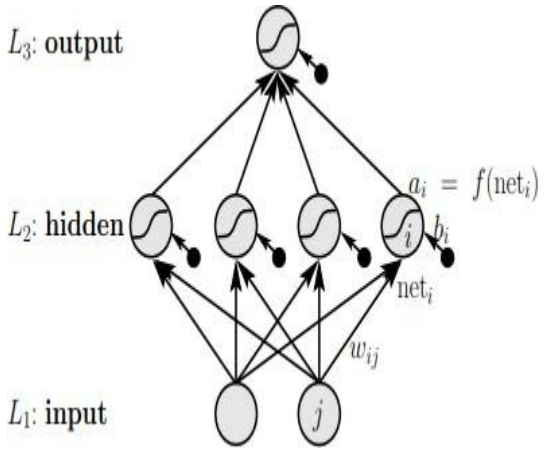

Figure 3 Multilayer Perceptron

The outcome of the perceptron is significantly affected by the magnitude of individual weights

w, and the mathematical formulation is given as follows:

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} + b \geq 0 \\ 0, & \text{else,} \end{cases} \tag{1}$$

The behaviour of a neuron is determined by its activation function, which in this case is a sign function. The sign function is mathematically represented as follows:

$$sign\ (z) = \begin{cases} +1, & \text{if } z \geq 0, \\ -1, & \text{else.} \end{cases} \tag{2}$$

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The perceptron's decision boundary is expressed as Equation (3). Since the decision-making function is linear with respect to the input x, it is referred to as a linear classifier.

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0 \quad (3)$$

A multilayer perceptron (MLP) network consists of multiple layers, each composed of several perceptrons. The hidden layers play a crucial role in extracting features from the input data. The dimensionality of the data can be adjusted either increased or reduced, based on the number of perceptrons in these layers, which are determined by the network's learning process from the given data.

3.2 Bernoulli Naive Bayes

Naive Bernoulli, based on Bayes' Theorem, is a probabilistic classification method that works well with binary or boolean input. It presupposes that each feature in the class is conditionally independent and has a Bernoulli distribution. The existence or lack of characteristics is used by the model to calculate the probability that the data belongs to each class. Bernoulli Naive Bayes, which is widely employed in text classification tasks such as spam detection, is efficient and successful on large datasets. Its performance is based on the assumption that the features are binary or can be transformed to binary.

$$P(x_i|y) = P(i|y)x_i + \big(1 - P(i|y)\big)\,(1 - x_i) \quad (4)$$

The multinomial NB's rule, which helps penalize the absence of attribute i for a class y sign, differs from this. The multinomial version, however, ignores a non-occurring characteristic. During the text classification process, this classification technique is developed and trained using word occurrence vectors. Some datasets, especially those with shorter documents, perform better on the BNB. If time permits, this is necessary for the models' analysis.

3.3. Gaussian Naive Bayes

A probabilistic classification method called Gaussian Naïve Bayes assumes that each class's characteristics have a Gaussian (normal) distribution. Each feature x_i is associated with a mean μ_i and a standard deviation σ_i unique to y. The Gaussian probability density function is used to calculate the likelihood of a feature x_i for a given class y.

$$P(x_i \vee y)$$

$$1 \qquad -(x_i - \mu_i)^2$$

under the Gaussian distribution. The model classifies the input data via choosing the class with the highest posterior probability. Gaussian Naive Bayes is most successful for continuous data and performs well when the characteristics within each class have a normal distribution, while its performance may suffer if the distribution assumption is compromised.

3.4 Random Forest

In contrast to previous classifiers, Random Forest, a variation of decision tree-based techniques, allows for the random growth of branches within a chosen subspace. A set of random base regression trees serves as the basis for the model's predictions. The method chooses a random subset of traits to split at each node, enabling the development of more branches. Because it combines many decision trees to increase prediction accuracy, Random Forest is notable as an ensemble learning technique. When the outcomes of several decision trees are combined to enhance overall performance, it can be seen as a type of bootstrapping.

The ith bootstrap denotes the specific sample that is selected, and the process first selects a

bootstrap sample $S^{(i)}$ from the sample space. Next, using a version of the normal decision tree algorithm, the algorithm learns a conventional decision tree. This alteration is implemented methodically as the tree develops. Specifically, at each node, instead of considering all possible feature splits, Random Forest randomly selects a subset of features $f \subseteq F$ is smaller than the entire set of features (f). The algorithm then splits based on the best feature in the subset. By narrowing down the feature set, the subset size

remains smaller, which reduces the

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$$= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x_i - \mu_i)^2}{2\sigma^2}\right) (5)$$

computational burden. This smaller subset helps minimize the complexity of feature selection, especially for datasets with many features and

Using Bayes' Theorem, the approach determines the posterior probability of each class based on the observable attributes. This is expressed as follows:

thus speeds up the learning process of the algorithm.

3.5 Principal Component Analysis

$$P(y) \prod_{i=1}^{d} P(x_i \vee y)$$

$$(y \vee x) = \frac{P(x)}{P(x)} (6)$$

Where P(y) is the class's prior probability, and

$(x_i \vee y)$ is the likelihood of the class's features

PCA finds the K major components by solving the characteristic equation of the correlation matrix of the observed variables and obtaining the associated eigenvectors and unit eigenvectors. The eigenvalues are then ranked

from largest to smallest, indicating the amount of variance in the observed data explained by each of the K principal components. These major component factors can be extracted using the following model:

$$F_i = T_{i1}X_2 + T_{i2}X_2 + T_{ik}(i = 1,2, \dots, m)(7)$$

In this case, F_i stands for the ith principal component factor, and T_{ij} represents the load of the i-th principal component factor on the j-th indicator.

In this case, k is the number of indicators, and m is the number of primary component elements. Several original indicators can be reduced to one or more composite indicators using the PCA technique. These reduced indicators capture most of the information from the original data, are uncorrelated with each other, and help eliminate redundancy. Additionally, the decrease in the number of indicators makes additional computations, study, and assessment more efficient.

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

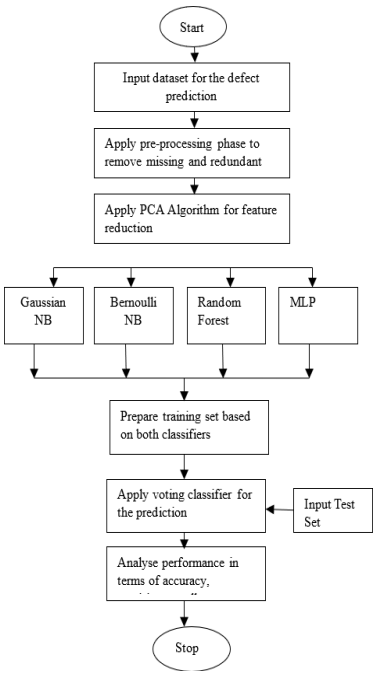

Figure 4: Proposed Methodology

4. Result and Discussion

This work analyzes and uses the PROMISE SE Library's "CM1/Software Defect Prediction" dataset. It has 22 attributes, three McCabe metrics, five distinct lines of code measurements, four basic Halstead measures, eight derived Halstead measures, one goal field, and a branch count. There are 498 records in this collection. Because it is independently accessible and comes from a reliable source, this sample data is chosen for the study.

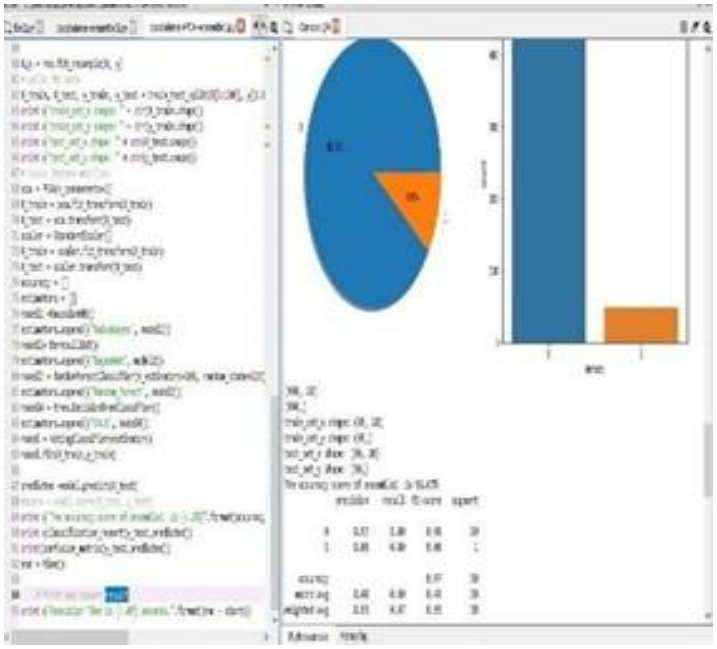

Figure 5: Integration of Ensemble 1 and PCA

To maintain balance between classes, the above figure shows the use of the PCA with ensemble approach.

The accuracy, precision, and recall of each model's output are measured. The outcomes of every categorization algorithm are shown in table 1.

Journal of Information Systems Engineering and Management
2025, 10(58s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Table 1. Outcome Evaluation

Model Name	Accuracy %	Precision %	Recall %
Bernoulli NB	74	15.15	31.25
C4.5	84.67	23.08	18.75
Gaussian NB	80.67	11.76	12.50
MLP Classifier	82.67	18.75	18.75
SVC(kernel=-linear)	88.67	33.33	6.25
Random Forest	87.33	20	6.25
Proposed Model	96.67	93	97

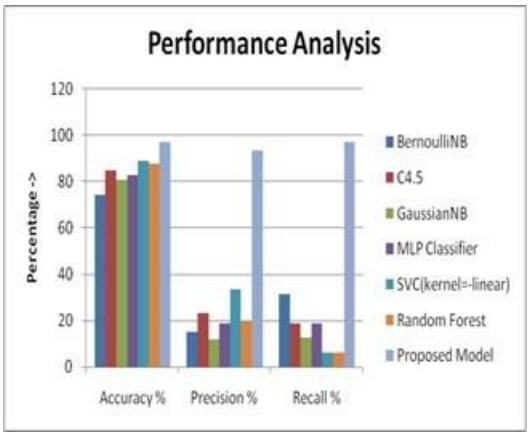

Figure 6 Performance of Models

To anticipate the software defect, a variety of classification methods are used, including BNB, GNB, RF, DT, MLP, and SVM. Figure 6

illustrates how a single classification method, without the use of a feature extraction tool like PCA or class balancing, predicts the software error. The proposed technique incorporates BNB, GNB, RF, and MLP. While preserving class balance, the attributes are extracted using PCA. The proposed method outperforms the current independent classification models in software fault prediction.

Conclusion

Software defects and inherent features of software products are regarded as important aspects of software quality. Defects in software are an unavoidable result of developing software. In addition, there is no assurance of software quality, and it takes a long time. There are vaious methods to characterize the flaws in terms of quality. However, the term "defects" refers to the variations from expectations or requirements that result in malfunctions. This study uses several distinct models, such as GNB, BNB, RF, C4.5, SVM, and MLP, to forecast software errors. The goal of ensemble strategy that integrates GNB, BNB, RF, and MLP is to anticipate software defects. In order to anticipate the software flaw, the PCA is used in conjunction with ensemble 1 and class balance. When compared to other models, the accuracy of the anticipated approach has reached 96.67%.

References

[1] P Lakshmi, T. LathaMaheswari, “An effective rank approach to software defect prediction using software metrics”, 2016, 10th International Conference on Intelligent Systems and Control (ISCO)
[2] Prianka Mandal, Amit Seal Ami, “Selecting best attributes for software defect prediction”, 2015, IEEE

International WIE Conference on Electrical and Computer Engineering (WIECON- ECE)
[3] Misha Kakkar, Sarika Jain, Abhay Bansal, P.S. Grover, "Evaluating Missing Values for Software Defect Prediction", 2019, International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)
[4] A Shanthini, R M Chandrasekaran, "Analyzing the effect of bagged ensemble approach for software fault prediction in class level and package level metrics", 2014, International Conference on Information Communication and Embedded Systems (ICICES2014)
[5] Ling-Feng Zhang, Zhao-Wei Shang, "Classifying feature description for software defect prediction", 2011, International Conference on Wavelet Analysis and Pattern Recognition
[6] Shiwang Agarwal, Sajal Gupta, Rishabh Aggarwal, Shashank Maheshwari, Lipika Goel, Sonam Gupta, "Substantiation of Software Defect Prediction using Statistical Learning: An Empirical Study", 2019, 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)
[7] Jun Wang, Beijun Shen, Yuting Chen, "Compressed C4.5 Models for Software Defect Prediction", 2012, 12th International Conference on Quality Software
[8] M. Nashaat and J. Miller, "Refining software defect prediction through attentive neural models for code understanding," Journal of Systems and Software, vol. 220, p. 112266, Feb. 2025, doi: https://doi.org/10.1016/j.jss.2024.112266.
[9] Y. Tang, Y. Zhou, C. Yang, Y. Du, and M. Yang, "Instance gravity oversampling method for software defect prediction," Information and Software Technology, vol. 179, p. 107657, Mar. 2025, doi:
[10] https://doi.org/10.1016/j.infsof.2024.107657.
[11] Yan Zhou, Chun Shan, Shiyou Sun, Shengjun Wei, Sicong Zhang, "Software Defect Prediction Model Based On KPCA-SVM", 2019, IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/I OP/SCI)
[12] Y. Jiang, B. Shen, and X. Gu, "Just-in-time software defect prediction via bi-modal change representation learning," Journal of Systems and Software, vol. 219, pp. 112253–112253, Oct. 2024, doi: https://doi.org/10.1016/j.jss.2024.112253.
[13] X. Fan, S. Zhang, K. Wu, W. Zheng, and Y. Ge, "Cross-Project Software Defect Prediction Based on SMOTE and Deep Canonical Correlation Analysis," Computers, Materials & Continua, vol. 78, no. 2, pp. 1687–1711, 2024, doi: https://doi.org/10.32604/cmc.2023.046187.
[14] Rituraj Singh, Jasmeet Singh, Mehrab Singh Gill, Ruchika Malhotra, Garima, "Transfer Learning Code Vectorizer based Machine Learning Models for Software Defect Prediction", 2020, International Conference on Computational Performance Evaluation (ComPE)
[15] Aqsa Rahim, Zara Hayat, Muhammad Abbas, Amna Rahim, Muhammad Abdul Rahim, "Software Defect Prediction with Naïve Bayes Classifier", 2021, International Bhurban Conference on Applied Sciences and Technologies (IBCAST)
[16] Kumar, R., Singhal, N., & Chhabra, A. (2025). Revolutionizing Business Management Strategies for Enhanced Output Through the Integration of Deep Learning and Cloud Computing. Journal of Information Systems Engineering and Management, 10(58s).
[17] Kumar, R., Singhal, N., & Chhabra, A. (2025). Hybrid Optimization algorithm with the combination of PSO and genetic algorithm for task scheduling in cloud computing. E-Learning and Digital Media, 0(0)