International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 4, 2025

Enhanced Software Defect Prediction Using a Stacking Ensemble of Decision Tree, Random Forest, and LightGBM

Sane Divya [1], Department of AIML, MJR College of Engineering and Technology, Piler, India

N. Surendra[2], Associate professor, Department of CSE, MJR College of Engineering and Technology, Piler,India

Abstract: *To facilitate the discovery of software defects, the latest study used a Stacking Classifier in conjunction with Decision Tree, Random Forest, and LightGBM models. Improved prediction accuracy and stability across a variety of software datasets are achieved by utilizing the complementing properties of many methods. Additionally, the system makes use of a Flask-based front end for safe user authentication and a straightforward user interface for real-time testing and prediction. Software quality assurance becomes much simpler with integration, allowing for faster, safer access, usage, and analysis of faults.*

Index Terms— *Software Defect Prediction, Ensemble Learning, Stacking Classifier, Random Forest, LightGBM, Decision Tree, Machine Learning, Flask Framework, Software Quality Assurance, NASA MDP Datasets..*

1. INTRODUCTION

Today's globally linked digital world relies on software to innovate, automate, and streamline everything from company operations to daily life. The software industry drives global growth by providing the infrastructure and communication tools that make it possible. Companies and people have depended on software more for work, communication, and commerce since the COVID-19 outbreak.

The Software Development Life Cycle (SDLC) has key steps that both the development and QA teams need to do. QA teams test the code that developers produce to detect and repair errors. This feedback loop between development and QA goes on until the product is dependable and of high quality.

Because of limits on time, money, and trained workers, it is still hard to build software that is free of bugs. These problems show that there is a growing need for smart and automated defect prediction systems that can find bugs in software early in the development process. Early defect prediction makes the best use of resources, lowers testing costs, and makes ensuring that software is delivered on time and to a high standard.

This study presents a sophisticated ensemble-based methodology for software fault prediction to tackle these challenges. The model makes predictions more accurate by using a voting ensemble mechanism to combine machine learning algorithms including Random Forest, SVM, Naïve Bayes, and MLP. An enhanced stacking-based methodology utilizing Decision Trees, Random Forest, and LightGBM enhances performance and reliability across diverse software datasets.

2. LITERATURE SURVEY

2.1 Semantic Feature Learning for Software Defect Prediction

✔ Proposes PM2-CNN, a Transformer-based software fault prediction model using multi-channel CNN and pretrained language models

✔ Integrates source code + external data (commit messages, comments) to enhance defect detection accuracy

✔ Captures both sequence correlation and contextual semantics from code repositories

✔ Outperforms baseline ML models on large public datasets, proving the benefit of non-code information

2.2 Deep Learning in Software Defect Prediction: Systematic Review & Meta-Analysis

✔ Conducts a comprehensive review of 63 studies comparing Deep Learning (DL) and Machine Learning (ML) models in SDP

✔ Identifies top DL models — CNN, DNN, LSTM, DBN, SDAE — trained on PROMISE & NASA datasets

✔ Meta-analysis shows DL outperforms ML in accuracy, recall, precision, F-measure, and AUC

✔ Sets a benchmark for future DL-based SDP research by outlining challenges and best practices

2.3 Cloud-Based Software Defect Prediction via Data & Decision-Level Fusion

✔ Introduces a two-step ML fusion system combining data-level learning and fuzzy logic decision fusion in the cloud
✔ Uses Naïve Bayes, ANN, and Decision Tree classifiers enhanced with eight fuzzy if-then rules for final prediction
✔ Achieves 91.05% accuracy using five NASA datasets (CM1, MW1, PC1, PC3, PC4)
✔ Outperforms standalone classifiers and ensemble models, improving defect identification reliability

2.4 Ensemble Learning for SDP using Adaptive Variable Sparrow Search Algorithm (AVSSA)

✔ Develops an ensemble Bagging-based SDP model using Extreme Learning Machine (ELM) optimized by AVSSA
✔ AVSSA enhances global optimization through adaptive hyperparameters and variable logarithmic spiral
✔ Addresses data imbalance and local optima problems common in conventional SDP approaches
✔ Outperforms four state-of-the-art algorithms on 15 public datasets, validated through Friedman & Holm tests

2.5 3PcGE: 3-Parent Child Genetic Evolution for Feature Selection in SDP

✔ Proposes 3PcGE, an evolutionary feature selection strategy using a three-parent child genetic model
✔ Enhances prediction by selecting optimal feature subsets through multi-objective optimization
✔ Produces more stable and accurate SDP models than NSGA-II, improving software quality and reducing testing cost
✔ Demonstrates that 3-parent genetic evolution yields stronger, more diverse candidate solutions

3. METHODOLOGY

The suggested solution uses an ensemble-based approach and a structured, multi-step procedure to improve the accuracy of predicting software defects. First, datasets from the NASA MDP repository are loaded and preprocessed. This means getting rid of duplicates, cleaning up records that aren't useful, and using label encoding to turn categorical data into numbers. To fix the problem of class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) is used. After that, Particle Swarm Optimization (PSO) is used to choose the most important characteristics that make the model work better. After that, the cleaned-up data is separated into training and testing sets so that it may be fairly tested. An Adaptive Voting Classifier combines the predictions of many machine learning models, such as Random Forest, SVM, Naïve Bayes, and MLP, to make them more reliable. A Stacking Classifier that combines Decision Tree, Random Forest, and LightGBM generates a meta-model that makes predictions more accurate. Finally, a Flask-based interface lets users safely sign up, log in, enter data, and see real-time forecasts of defects. This makes sure that the system is easy to use and accessible.

A. Proposed Work:

An enhanced Stacking Classifier combining Decision Tree, Random Forest, and LightGBM models is implemented to improve the accuracy of software fault prediction in the suggested extended study. For consistent, high-quality predictions, this hybrid ensemble combines the strengths of three distinct learning algorithms: Decision Tree for interpretability, Random Forest for robustness, and LightGBM for gradient boosting efficiency. Users may safely register, log in, and conduct real-time defect predictions using an interactive and user-friendly platform made possible by the system's Flask-based web interface. For real-world use in SDLCs, this add-on increases model performance with multi-level learning while simultaneously making it more accessible, secure, and user-friendly.

B. System Architecture:

The system architecture shows how the proposed intelligent ensemble-based software fault prediction methodology will function. The first step is to import and preprocess the dataset, which includes cleaning the data, removing duplicates, and encoding the labels to make sure the input is of high quality. K-means clustering is used to group comparable instances of the cleaned data. Then, PSO (Particle Swarm Optimization) is used to improve the relevance of features by applying it to specific models like MLP, SVM, Naïve Bayes, and Random Forest. Then, an ensemble learning method is used to integrate these improved models to make predictions more accurate and stable. During the extension phase, a Stacking Classifier combines Decision Tree, Random Forest, and LightGBM to allow for multi-level learning that leads to better results. The last step is evaluation, when the model's ability to anticipate software bugs is tested by calculating accuracy and other performance indicators.

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 4, 2025

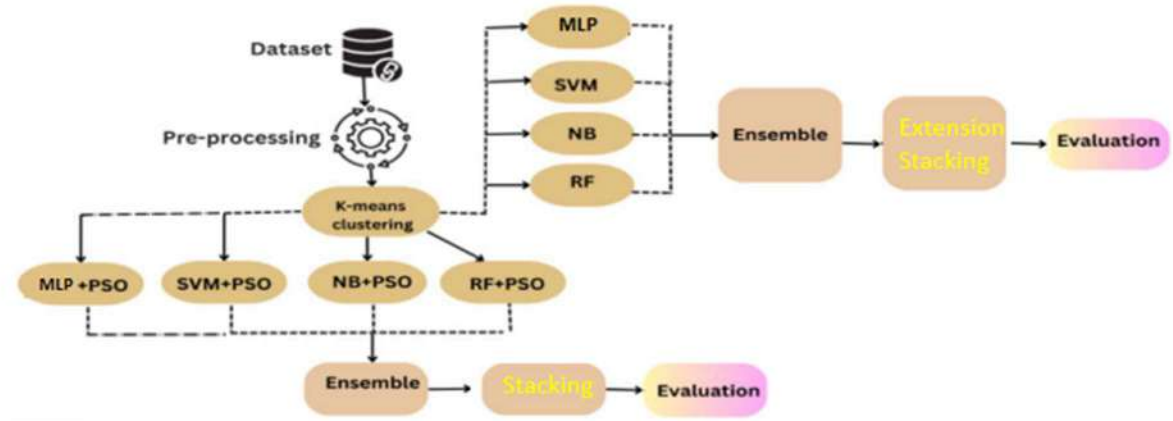


Fig proposed architecture

C. MODULES:

a) **Data Loading:**
- Imports the software defect datasets from the NASA MDP repository.
- Initializes data for further preprocessing and analysis.

b) **Data Preprocessing:**
- Removes duplicates, irrelevant, and incomplete records to ensure clean data.
- Converts categorical data into numerical form using label encoding for model compatibility.

c) **K-Means Clustering:**
- Groups the preprocessed data into clusters based on feature similarity.
- Helps the models learn distinct defect patterns more effectively.

d) **Feature Optimization using PSO:**
- Applies Particle Swarm Optimization to identify and retain the most relevant features.
- Reduces dimensionality, improving the model's accuracy and efficiency.

e) **Base Model Training (MLP, SVM, NB, RF):**
- Trains four supervised models independently using optimized features.
- Each model learns unique data patterns to improve prediction strength.

f) **Ensemble Learning:**
- Combines predictions from MLP, SVM, NB, and RF using an Adaptive Voting Classifier.
- Enhances overall model performance by merging individual model strengths.

g) **Extension – Stacking Classifier:**
- Integrates Decision Tree, Random Forest, and LightGBM into a meta-model for deeper learning.
- Further refines prediction accuracy through advanced ensemble stacking.

h) **Evaluation:**
- Measures performance using accuracy, precision, recall, and F1-score.
- Compares ensemble and stacking models to identify the most effective approach.

i) **User Authentication (Signup & Login):**
- Provides secure access through user registration and login.
- Ensures only authorized users can test and use the prediction system.

j) **User Input & Prediction:**
- Allows users to input new data for real-time defect prediction.
- Displays prediction results interactively through a Flask-based interface.

D. Algorithms:

a) **Support Vector Machine (SVM):**

Methods for supervised learning SVM finds the hyperplane in feature space that best separates the groups. The project uses it to sort out bad modules by making the gap between distinct classes as big as

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 4, 2025

possible. This makes the model more accurate and able to generalize.

b) Random Forest:

Random Forest uses ensemble learning to train several decision trees and calculate their mode class (classification) or mean prediction (regression). In the project, its resilience and accuracy in predicting faulty modules through aggregated decision-making increase model performance and decrease overfitting for various datasets.

c) Naive Bayes:

Naive Bayes is a probabilistic classifier that uses Bayes' theorem and assumes that features are independent. In the project, it is used to figure out how likely it is that faults will happen based on the feature values. This is a simple and effective way to work with massive datasets that have categorical properties.

d) MLP (Multi-Layer Perceptron):

MLP is an artificial neural network made up of many layers of neurons that can do complicated input-to-output mappings. The research uses it to find complex patterns and interactions in the data, which helps forecast which modules will be faulty since it can learn from both linear and non-linear correlations.

a) Adaptive Voting Classifier (RF + SVM + NB + MLP):

By using a voting system, the Adaptive Voting Classifier improves the accuracy of predictions made by Random Forest, SVM, Naive Bayes, and MLP. a study uses an ensemble approach to improve prediction accuracy for broken modules by combining the strengths of all the models.

b) Stacking Classifier (DT + RF with LightGBM):

To improve the accuracy of its predictions, the Stacking Classifier builds a meta-model that incorporates two gradient boosting frameworks, Decision Trees (DT), and Random Forest (RF). As part of the project, this classifier combines different models to improve the accuracy and resilience of fault prediction by taking use of their individual strengths.

4. EXPERIMENTAL RESULTS

To determine how well the intelligent ensemble-based model could forecast software defects, it was tested on seven benchmark datasets taken from the NASA MDP repository: CM1, JM1, MC2, MW1, PC1, PC3, and PC4. The initial step involved training and optimizing four supervised learning algorithms using iterative parameter tuning: Random Forest, Support Vector Machine (SVM), Naïve Bayes, and Multi-Layer Perceptron (MLP). In order to increase consistency and decrease individual model bias, their forecasts were aggregated using a voting ensemble.

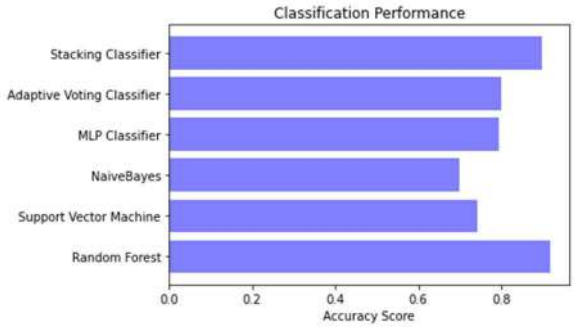
Accuracy rates of 86.5% for CM1, 66% for JM1, 73.5% for MC2, 80.1% for MW1, 87% for PC1, 87.5% for PC3, and 90.4% for PC4 were some of the noteworthy findings attained by the ensemble model across the datasets. We used an extension model that included Decision Tree, Random Forest, and LightGBM, a Stacking Classifier, to increase accuracy even more. With accuracies of 92.4% for CM1, 83.6% for JM1, 73.5% for MC2, 89.7% for MW1, 92.2% for PC1, 92.4% for PC3, and 93.5% for PC4, this hybrid technique obtained excellent results.

When compared to typical single-model approaches, the experimental findings show that integrating diverse classifiers using ensemble and stacking techniques greatly improves prediction accuracy, dependability, and resilience across different software fault datasets.

Accuracy: The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases. Mathematically, this can be stated as:

Accuracy = TP + TN TP + TN + FP + FN.

$$Accuracy = \frac{(TN + TP)}{T}$$



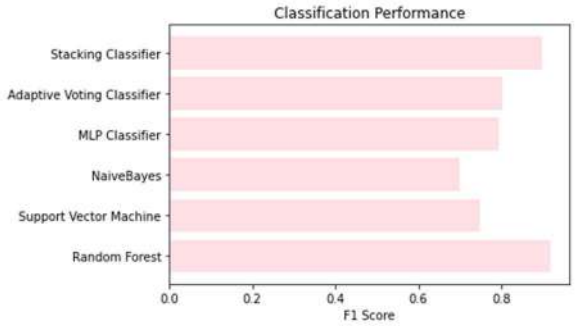
F1-Score: F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset.

$$F1 = 2 \cdot \frac{(Recall \cdot \Pr e\,cision)}{(Recall + \Pr e\,cision)}$$

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

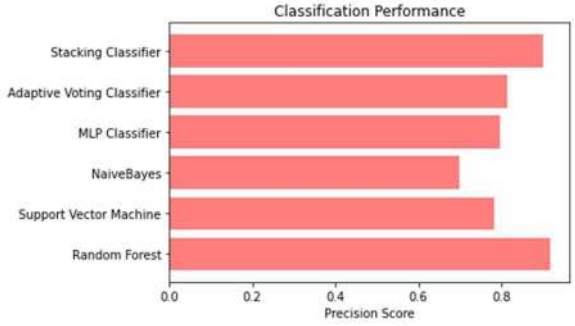
Volume 13, Issue 4, 2025



Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\Pr e\,cision = \frac{TP}{(TP + FP)}$$



Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

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

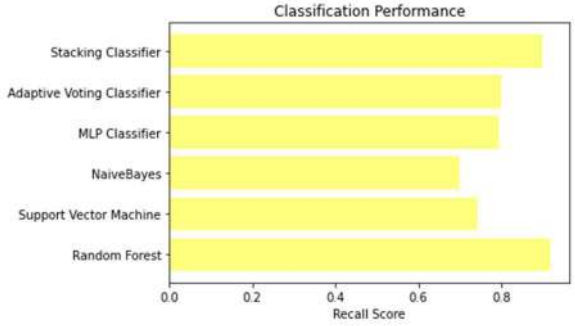


Fig 2. Results

ML Model	Accuracy	Precision	Recall	F1-Score	AUC Score	Specificity	Sensitivity
Random Forest	0.918	0.921	0.918	0.918	1.000	0.918	0.918
Support Vector Machine	0.713	0.719	0.713	0.714	0.833	0.714	0.713
Naive Bayes	0.719	0.765	0.719	0.726	0.783	0.721	0.719
MLP Classifier	0.860	0.867	0.860	0.860	0.961	0.859	0.860
Adaptive Voting Classifier	0.865	0.865	0.865	0.866	0.971	0.865	0.865
Stacking Classifier	**0.924**	**0.924**	**0.924**	**0.924**	**1.000**	**0.924**	**0.924**

5. CONCLUSION

The suggested intelligent ensemble-based software defect prediction model significantly improves software quality assurance by correctly finding modules that are likely to have defects before testing. The system makes strong and trustworthy predictions by combining many classifiers, such as Random Forest, SVM, Naïve Bayes, and MLP, into an Adaptive Voting Classifier. Adding a Stacking

Classifier that combines Decision Tree, Random Forest, and LightGBM makes accuracy and generalization much better over a wider range of datasets. The model's better performance compared to previous techniques is shown by experimental data. The addition of a Flask-based web interface also makes deployment easy, safe, and useful, which makes the system perfect for real-world software development settings.

6. FUTURE SCOPE

Adding deep learning architectures like CNNs or LSTMs to the suggested system might make it even better by helping it find complicated nonlinear patterns in software metrics. Future endeavors may concentrate on real-time defect prediction via continuous integration pipelines to facilitate agile development environments. Adding explainable AI (XAI) approaches and making the model work with massive industrial datasets helps make forecasts more trustworthy and clear. Also, making the system a cloud-based platform with API access would make it easier for more people to use, make it easier to scale, and make it easier to integrate into current software development workflows.

REFERENCES

[1] J. Liu, J. Ai, M. Lu, J. Wang, and H. Shi, ‘‘Semantic feature learning for software defect prediction from source code and external knowledge,’’ J. Syst. Softw., vol. 204, Oct. 2023, Art. no. 111753, doi: 10.1016/j.jss.2023.111753.

[2] Z. M. Zain, S. Sakri, and N. H. A. Ismail, ‘‘Application of deep learning in software defect prediction: Systematic literature review and metaanalysis,’’ Inf. Softw. Technol., vol. 158, Jun. 2023, Art. no. 107175, doi: 10.1016/j.infsof.2023.107175.

[3] S. Aftab, S. Abbas, T. M. Ghazal, M. Ahmad, H. A. Hamadi, C. Y. Yeun, and M. A. Khan, ‘‘A cloud-based software defect prediction system using data and decision-level machine learning fusion,’’ Mathematics, vol. 11, no. 3, p. 632, Jan. 2023, doi: 10.3390/math11030632.

[4] Y. Tang, Q. Dai, M. Yang, T. Du, and L. Chen, ‘‘Software defect prediction ensemble learning algorithm based on adaptive variable sparrow search algorithm,’’ Int. J. Mach. Learn. Cybern., vol. 14, no. 6, pp. 1967–1987, Jan. 2023, doi: 10.1007/s13042-022-01740-2.

[5] S. Goyal, ‘‘3PcGE: 3-parent child-based genetic evolution for software defect prediction,’’ Innov. Syst. Softw. Eng., vol. 19, no. 2, pp. 197–216, Jun. 2023, doi: 10.1007/s11334-021-00427-1.

[6] S. Mehta and K. S. Patnaik, ‘‘Stacking based ensemble learning for improved software defect prediction,’’ in Proc. 5th Int. Conf. Microelectron., Comput. Commun. Syst., vol. 748, 2021, pp. 167–178.

[7] M. Shafiq, F. H. Alghamedy, N. Jamal, T. Kamal, Y. I. Daradkeh, and M. Shabaz, ‘‘Retracted: Scientific programming using optimized machine learning techniques for software fault prediction to improve software quality,’’ IET Softw., vol. 17, no. 4, pp. 694–704, Jan. 2023, doi: 10.1049/sfw2.12091.

[8] Z. M. Zain, S. Sakri, and N. H. A. Ismail, ‘‘Application of deep learning in software defect prediction: Systematic literature review and meta-analysis,’’ Inf. Softw. Technol., vol. 158, Jun. 2023, Art. no. 107175, doi: 10.1016/j.infsof.2023.107175.

[9] M. Unterkalmsteiner et al., ‘‘Software startups—A research agenda,’’ 2023, arXiv:2308.12816.

[10] A. K. Gangwar and S. Kumar, ‘‘Concept drift in software defect prediction: A method for detecting and handling the drift,’’ ACM Trans. Internet Technol., vol. 23, no. 2, pp. 1–28, May 2023, doi: 10.1145/3589342.

[11] M. S. Alkhasawneh, ‘‘Software defect prediction through neural network and feature selections,’’ Appl. Comput. Intell. Soft Comput., vol. 2022, pp. 1–16, Sep. 2022, doi: 10.1155/2022/2581832.

[12] T. F. Husin and M. R. Pribadi, ‘‘Implementation of LSSVM in classification of software defect prediction data with feature selection,’’ in Proc. 9th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI), Jakarta, Indonesia, Oct. 2022, pp. 126–131, doi: 10.23919/EECSI56542.2022. 9946611.

[13] J. A. Richards, ‘‘Supervised classification techniques,’’ in Remote Sensing Digital Image Analysis. Cham, Switzerland: Springer, 2022, pp. 263–367.

[14] B. J. Odejide, A. O. Bajeh, A. O. Balogun, Z. O. Alanamu, K. S. Adewole, A. G. Akintola, and S. A. Salihu, ‘‘An empirical study on data sampling methods in addressing class imbalance problem in software defect prediction,’’ in Proc. Comput. Sci. Online Conf. Cham, Switzerland: Springer, Apr. 2022, pp. 594–610.

[15] X. Wu and J. Wang, ‘‘Application of bagging, boosting and stacking ensemble and EasyEnsemble methods for landslide susceptibility mapping in the three Gorges reservoir area of China,’’ Int. J. Environ. Res. Public Health, vol. 20, no. 6, p. 4977, Mar. 2023, doi: 10.3390/ijerph20064977.

[16] F. Jiang, X. Yu, D. Gong, and J. Du, ‘‘A random approximate reduct based ensemble learning approach and its application in software defect prediction,’’ Inf. Sci., vol. 609, pp. 1147–1168, Sep. 2022, doi: 10.1016/j.ins.2022.07.130.
[17] H. Chen, X.-Y. Jing, Y. Zhou, B. Li, and B. Xu, ‘‘Aligned metric representation based balanced multiset ensemble learning for heterogeneous defect prediction,’’ Inf. Softw. Technol., vol. 147, Jul. 2022, Art. no. 106892, doi: 10.1016/j.infsof.2022.106892.
[18] A. O. Balogun, A. O. Bajeh, V. A. Orie, and A. W. Yusuf-Asaju, ‘‘Software defect prediction using ensemble learning: An ANP based evaluation method,’’ FUOYE J. Eng. Technol., vol. 3, no. 2, pp. 50–55, Sep. 2018, doi: 10.46792/fuoyejet.v3i2.200.
[19] A. O. Balogun, F. B. Lafenwa-Balogun, H. A. Mojeed, V. E. Adeyemo, O. N. Akande, A. G. Akintola, A. O. Bajeh, and F. E. Usman-Hamza, ‘‘SMOTE-based homogeneous ensemble methods for software defect prediction,’’ in Computational Science and Its Applications—ICCSA 2020, vol. 12254, O. Gervasi, B. Murgante, S. Misra, C. Garau, I. B. D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino, C. M. Torre, and Y. Karaca, Eds. Cham, Switzerland: Springer, 2020, pp. 615–631.
[20] R. J. Jacob, R. J. Kamat, N. M. Sahithya, S. S. John, and S. P. Shankar, ‘‘Voting based ensemble classification for software defect prediction,’’ in Proc. IEEE Mysore Sub Sect. Int. Conf. (MysuruCon), Hassan, India, Oct. 2021, pp. 358–365, doi: 10.1109/MysuruCon52639.2021. 9641713.
[21] A. Alsaeedi and M. Z. Khan, ‘‘Software defect prediction using supervised machine learning and ensemble techniques: A comparative study,’’ J. Softw. Eng. Appl., vol. 12, no. 5, pp. 85–100, 2019, doi: 10.4236/jsea.2019.125007.
[22] A. Iqbal and S. Aftab, ‘‘A classification framework for software defect prediction using multi-filter feature selection technique and MLP,’’ Int. J. Mod. Educ. Comput. Sci., vol. 12, no. 1, pp. 18–25, Feb. 2020, doi: 10.5815/ijmecs.2020.01.03.
[23] M. Cetiner and O. K. Sahingoz, ‘‘A comparative analysis for machine learning based software defect prediction systems,’’ in Proc. 11th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT), Kharagpur, India, Jul. 2020, pp. 1–7, doi: 10.1109/ICCCNT49239.2020.9225352.
[24] K. Wang, L. Liu, C. Yuan, and Z. Wang, ‘‘Software defect prediction model based on LASSO–SVM,’’ Neural Comput. Appl., vol. 33, no. 14, pp. 8249–8259, Jul. 2021, doi: 10.1007/s00521-020-04960-1.
[25] M. S. Daoud, S. Aftab, M. Ahmad, M. A. Khan, A. Iqbal, S. Abbas, M. Iqbal, and B. Ihnaini, ‘‘Machine learning empowered software defect prediction system,’’ Intell. Autom. Soft Comput., vol. 31, no. 2, pp. 1287–1300, 2022, doi: 10.32604/iasc.2022.020362.
[26] Y. N. Soe, P. I. Santosa, and R. Hartanto, ‘‘Software defect prediction using random forest algorithm,’’ in Proc. 12th South East Asian Technical Univ. Consortium, Yogyakarta, Indonesia, Mar. 2018, pp. 1–5, doi: 10.1109/SEATUC.2018.8788881.
[27] F. H. Alshammari, ‘‘Software defect prediction and analysis using enhanced random forest (extRF) technique: A business process management and improvement concept in IoT-based application processing environment,’’ Mobile Inf. Syst., vol. 2022, pp. 1–11, Sep. 2022, doi: 10.1155/2022/2522202.
[28] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, and A. Husen, ‘‘Performance analysis of machine learning techniques on software defect prediction using NASA datasets,’’ Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 5, 2019, doi: 10.14569/IJACSA.2019.0100538.
[29] H. Alsghaier and M. Akour, ‘‘Software fault prediction using particle swarm algorithm with genetic algorithm and support vector machine classifier,’’ Softw., Pract. Exper., vol. 50, no. 4, pp. 407–427, Apr. 2020, doi: 10.1002/spe.2784.
[30] S. K. Rath, M. Sahu, S. P. Das, S. K. Bisoy, and M. Sain, ‘‘A comparative analysis of SVM and ELM classification on software reliability prediction model,’’ Electronics, vol. 11, no. 17, p. 2707, Aug. 2022, doi: 10.3390/electronics11172707.