UtilitasMathematica

ISSN 0315-3681 Volume 122, 2025

Ensemble learning and Class Imbalance mitigation technique in Software Fault prediction.

V. Bhargav Ram Charan[1]
[1]M. Tech Student
Department of
Computer Science and Engineering
School of Engineering
Anurag University
Hyderabad, Telangana, India
bhargavramcharanvinnakota@gmail.com

Meenakshi Simha[2]
[2]Assistant Professor
Department of
Computer Science and Engineering
School of Engineering
Anurag University
Hyderabad, Telangana, India
meenakshicse@anurag.edu.in

Dr. G. Vishnu Murthy[3]
[3]Professor and Dean
Department of
Computer Science and Engineering
School of Engineering
Anurag University
Hyderabad, Telangana, India
deancse@anurag.edu.in

***Abstract*—Predicting software fault is essential in enhancing the existing software reliability and quality, whereby, faulty modules that may arise are identified at early stages of software development. Since the software defect data has an inherent imbalance in their class labels, i.e., non-faulty cases far exceed the faulty ones, it is a task to effectively predict faulty cases. Trying to combat this problem, this study is combined with several different oversampling strategies (Synthetic Minority Oversampling Technique,SMOTE, BorderlineSMOTE, and Adaptive Synthetic Sampling,ADASYN) to produce synthetic minority samples more thoroughly. They can be used to balance the dataset, especially: borderline and hard-to-learn instances, which improves the model performance.**

We suggest using a resilient ensemble machine learning-based approach that allows integrating several machine learning models using a stacking method. Our architecture has base classifiers of Random Forest, Extra Trees, LightGBM and CatBoost. Output of such models is used to train a Multilayer Perceptron (MLP) neural network which acts as the meta-classifier. This stacking ensemble exploits the advantages of individual learners and learns complex interaction of features using deep learning. MLP is two-layered with 32 and 16 neurons, and activation ReLU and Adam optimizer were trained during a 300-epoch cycle.

We tested our model on benchmark datasets such as PC1, JM1 and KC2 to recenter its performance. Experimental results show clearly that the proposed NN Meta Stacking model is much more accurate compared to traditional classifiers to a significant extent; it was constructed when improved oversampling techniques were used. The PC1 dataset was the most accurate one after testing and it shows that the model is robust. Our results indicate that advanced oversampling has the effectiveness to increase fault prediction when used in conjunction with deep ensemble learning in software systems.

Keywords: Software Fault Prediction, SMOTE, BorderlineSMOTE, ADASYN, Stacking Ensemble, Neural Network, Multilayer Perceptron, Class Imbalance, Software Quality.

I. INTRODUCTION

In this hugely changing world of information and technology, software has found its use in nearly all fields, such as healthcare, transport, finance, education, and even in the military. Since the software applications are becoming larger and sophisticated, the difficulties in maintaining their quality and reliability are also increasing. Failure to detect and rectify software faults at the early stages could result to tremendous operation break downs, economic losses, and even compromise to human lives [1]. Therefore the issue of early fault detection in software development has become very crucial in ensuring reliability of software and maintaining the software at low costs.

Software fault prediction (SFP) means using the metrics on previous software and machine learning to infer that software module is likely to be at fault. Highly-accurate prediction allows developers and testers to concentrate their efforts on the modules that cause significant risks, which will increase the efficiency of testing and raise the quality of a product [2]. Even though machine learning and data mining are developed, high-accuracy SFP is a challenging task because of the evaluation difficulties of data quality and imbalance of classes in fault records.

DatasetsData points applied in fault prediction are normally lopsided, i.e., the count of faulty modules is so considerably less than that of functioning modules. It results in bias of conventional classifiers to the majority that causes recall and misclassification of the minority (faulty) class [5]. To address this problem the data-level techniques have been suggested such SMOTE (Synthetic Minority Over-sampling Technique) and BorderlineSMOTE, and ADASYN. SMOTE produces synthetic data of the minority class, interpolystically between neighboring instances [6]. BorderlineSMOTE concentrates on creating the samples that are close to the boundary between the classes, where misclassifications are much more probable [7]. ADASYN will provide the weights of harder-to-learn minority examples with more adaptive samples by generating many minority samples thus results in a more informative distribution [8].

Besides the data preprocessing techniques there are also algorithm level techniques, e.g. ensemble learning, which has a lot of potential in improving the current performance of fault prediction. Ensembles learning techniques are used to assemble a group of base learners with the aim of producing an improved and reliable model of prediction [3]. Such methods as Bagging, Boosting and Stacking, employ the diversity of models to minimize generalization errors. One of them, stack-

ing in particular, has been found useful on more complicated classification problems. This requires training several base classifiers and feed their prediction to a meta-learner that gives the final output [4].

In this study, an improved stacking ensemble architecture is suggested and the meta-classifier is a neural network that is used in prediction of software faults. Base classifiers refer to Random Forest, Extra Trees, LightGBM, and CatBoost, having the best performance models and capabilities to work with various types of data [9], [13]. The meta-classifier is a Multilayer Perceptron (MLP) neural network, which is aimed at learning complicated dependencies among the predictions of the base models. Optimized datasets processed by combination of SMOTE, BorderlineSMOTE, and ADASYN are used as an input to train the model to learn in a balanced and representative way [10].

The proposed model is tested against challenge datasets like PC1, JM1, and KC2 having a different degree of complexity and class imbalance. In terms of accuracy and recall, our model of Meta Stacking NN outperforms other traditional classifiers when performing a comparative analysis against them, especially when the PC1 dataset is used [15]. The findings positively prove that the integration of oversampling and ensemble learning can solve the fundamental issues of fault prediction in software.

II. Literature Survey

Software fault prediction (SFP) includes an important subfield of software engineering that aims at early identification of the modules of a software system that are prone to have errors on the basis of certain fault characteristics. This active fault detection scheme contributes greatly to assuring the stability of a software, minimizes its maintaining expenditure and assist in streamlining resource use during testing periods [1]. During the last twenty years, a great variety of statistical, machine learning and in the recent past, deep learning algorithms have been studied to realize this objective. Initial implementations primarily employed conventional classifiers including Logistic Regression, Na" Actually, the initial implementations largely employed conventional classifiers, including the Logistic Regression, Naive bayes, k-Nearest Neighbor (KNN), and Support Vector Machines (SVM) classifiers, which depend upon object-based software metrics such as McCabe cyclomatic complexity, Halstead metrics, and Cheng and Kemerer (CK) metrics [2], [3].

Khoshgoftaar et al. stressed the effect of noisy and unbalanced dataset on the quality of modelling and presented preprocessing methods, such as feature selection and discretization to achieve better generalization [4]. Menzies et al. promoted minimal metric subsets which demonstrated that models that learned on smaller subset of relevant features tended to outperform ones that learned on the full, high-dimensional feature set [5]. Nevertheless, one of the most difficult problems of SFP is the problem of the class imbalance, which is usually manifested in the fact that the number of defective modules in a software project is much lower than modules that are not defective. This causes biased classifiers that tend to misclassify minority class samples.

To address this, a wide use of resampling methods e.g. SMOTE (Synthetic Minority Over-sampling Technique) has been undertaken. SMOTE draws artificial minority class elements by interpolating current ones and their k-nearest neighbors [6]. Although it works, SMOTE noise may be introduced unintentionally, which is highly probable when the generation of samples is conducted in sparse or overlapping areas. Extensions like BorderlineSMOTE and ADASYN were proposed to reduce such drawbacks. BorderlineSMOTE works by enhancing SMOTE by working more on that area and turning the side of samples that are likely to be misclassified lying towards the decision boundary [7]. As opposed to that, ADASYN (Adaptive Synthetic Sampling) determines the increase in synthetic data in areas with insufficient examples of the minority classes, which makes it more specific [8].

At the same pace with the development of resampling techniques, ensemble learning has transformed software defect prediction. Ensemble methods are a combination of predictions of various models, such that they deliver high accuracy and robustness. Bagging-based approaches such as Random Forest, combine the predictions of several decision trees which have been trained with bootstrapped data thereby having a high variance reduction [9]. Frame-work Boosting algorithms such as AdaBoost and XGBoost model the information step by step by treating more misclassified cases per construction [10]. These frame works, LightGBM and CatBoost, are modernized gradient boosting frameworks that enhance training speed and large scale data [11], [12]. Such approaches have always shown in the first line in terms of the predictive modeling accuracy contests and research experiments.

Stacking is one of the most potent ensemble methods that utilize the strengths of multiple base learners and passes their results to a meta-learner to make the final forecast. In contrast to bagging or boosting, unlike the latter, stacking permits the stacking of heterogeneous models, which increases the expressiveness and diversity of the resulting ensemble [13]. The stacking technique was also put in good practice by Zhang et al. who used a combination of Decision Trees, SVMs, NaTnex Bayes as the base learners and logistic Regression as a meta- learner; the stacking approach outperformed the results of the individual classifiers [14]. Recently, neural networks have also entered the scenery as meta-learners in stacking architectures because of their non-linear relationship coupling capability and their capacity to represent a complex interaction between predictions [15].

III. Methodology

In this research project, a sophisticated fault prediction framework will be proposed that addresses the issues of imbalanced data and non-linear feature interactions by combining superior oversampling techniques to a well stabilized stacking ensembler.

Data Norming and Oversampling

TABLE I
COMPARISON OF METHODS AND DATASETS

Paper	Methods Used	Dataset	Performance	Limitations	Features Analyzed
[1]	Feature-dependent Naive Bayes variant	NASA MDP	High accuracy and robustness in fault detection	Sensitive to feature selection	Software metrics, code complexity
[2]	Comparative analysis of oversampling techniques (SMOTE, ADASYN, BorderlineSMOTE)	PROMISE	SMOTE-based methods improved minority class detection	Increased computational time	Code churn, complexity metrics
[3]	DeepStack: Deep learning stacking ensemble	NASA MDP	Outperformed traditional models in accuracy and F1-score	Requires large training data	Static and process metrics
[4]	Ensemble voting and stacking framework	Various open-source projects	Improved overall defect prediction rate	May suffer from model overfitting	Object-oriented metrics
[5]	Empirical study on preprocessing methods for imbalance	Multiple software repositories	SMOTE + feature selection yielded best results	Overfitting risk in noisy data	Process metrics, code metrics
[6]	Adaptive SMOTE-based ensemble learning	Proprietary datasets	Enhanced detection of rare faults	Increased training complexity	Software metrics, fault history
[7]	Borderline-SMOTE CNN hybrid model	PROMISE	Better minority class prediction near boundaries	Higher training time	Code metrics, change metrics
[8]	SMOTEBoost for defect prediction	NASA MDP	Balanced precision and recall	Potential synthetic sample noise	Code and process metrics

Software fault data in the real world is highly imbalanced in the sense that there are many more non-faulty modules than the faulty ones. Common classifiers have a tendency of being biased towards the majority category, and thus they are associated with low fault detection qualities. To counter this we apply a combination of oversampling elements including SMOTE, BorderlineSMOTE, and ADASYN to artificially seed samples of such minority classes to achieve better-balanced information and stronger model learning ability.

The basic idea of SMOTE is to form synthetic minority samples, interpolating among the known instances of minority classes and nearest neighbors. Mathematically, for a minority class sample x_i and one of its k nearest neighbors x_{nn}, a new synthetic sample x_{new} is generated as:

$$x_{new} = x_i + \lambda \times (x_{nn} - x_i), \quad \lambda \sim U(0, 1)$$

where λ is a random scalar sampled uniformly between 0 and 1.

BorderlineSMOTE is an improvement of this method because it is targeted to samples located close to the class boundary those that are more likely to be misclassified. In this way, the discriminative power of the classifier will be improved because it will learn more about the decision frontier since the synthetic samples will be generated in these critical boundary areas.

To further enhance sampling, ADASYN synthesizes additional data points in the areas that contain few minority samples, consequently, making it hard to learn. The targeted oversampling aids the model in better learning generalization in hard parts of the feature space.

The combination of these approaches augments the dataset with various and LM (Lowland) and HM (Highland) Scottish populations, which leads to the balanced and truly representative training set.

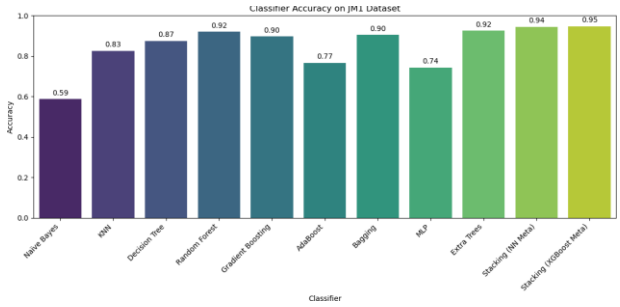

Fig. 1. Comparison of Algorithms Performance Across Fault Prediction Datasets.

A. Stacking Ensemble Model Architecture

Our model uses stacking ensemble technique in order to utilize the compensatory power of different classifiers. In stacking, learning goes on with a set of base learners whose outputs are then fused by a meta-classifier learned to use the outputs of the set of base learners in higher-order abstractions that lowers the bias and variance.

The base learners (Level 1) consist of:

- **Random Forest (RF)** — a bagging-based ensemble that aggregates multiple decision trees trained on bootstrapped samples.
- **Extra Trees (ET)** — similar to RF but uses random splits for enhanced model diversity.
- **LightGBM** — a gradient boosting framework optimized for speed and efficiency with leaf-wise tree growth.
- **CatBoost** — gradient boosting that excels in handling categorical features and reduces overfitting.

The meta-classifier (Level 2) is a Multilayer Perceptron (MLP) neural network structured as follows:

- Two hidden layers with 32 and 16 neurons respectively.

- ReLU activation function, facilitating non-linear transformations.
- Adam optimizer, chosen for its efficient stochastic optimization.
- Trained over 300 epochs to ensure convergence.

If $h_i(x)$ denotes the prediction probability from the i-th base model for input x, then the input to the meta-classifier is the vector:

$$\mathbf{h}(x) = [h_1(x), h_2(x), \ldots, h_n(x)]$$

The meta-classifier then outputs the final prediction as:

$$\hat{y} = f_{\text{meta}}(\mathbf{h}(x))$$

where f_{meta} represents the learned neural network mapping.

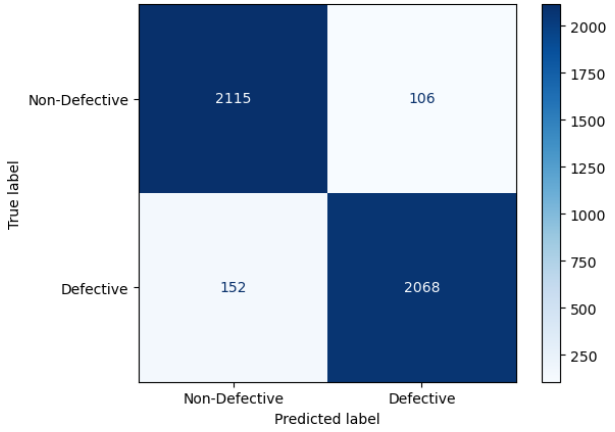

Fig. 2. Confusion Matrix for Fault Prediction Using the Stacking Ensemble.

B. Proposed System Overview

These components are incorporated within the proposed system as part of a smooth pipeline which began with data pre processing, continues with oversampling, model training, and concluded with the prediction.

This pipeline uses raw software metrics gathered in multiple projects. The imbalanced proportion of the faulty and non-faulty cases is adjusted with the help of hybrid oversampling (SMOTE + BorderlineSMOTE + ADASYN) methodology. Then one trains four different base learners on this balanced dataset to get complementary feature representations and predictions.

These predictions can then be regarded as input features of the MLP meta-classifier, which is to learn how to best combine the output of base models into higher accuracy and robustness. This technology can highly identify faults in highly difficult, noisy and high dimensional data.

The table and the corresponding figures above, Figure 1 illustrate that the accuracy of the stacking ensemble with neural meta-classifier was higher than the other results when the datasets were used. Figure 2 demonstrates that the model is able to build a proper classification between faulty and non-faulty modules with a small number of misclassification errors.

Overall, we have employed the state-of-the-art sampling methodology to overcome the class imbalance issue, a set of diverse base learners to capture the different data patterns, and a deep learning meta-learner to fine-tune the final predictions, which makes our software fault-prediction system the highly effective one.

Results and Discussion

The results of the use of different machine learning classifiers were determined on five datasets (CM1, KC1, KC2, PC1, and JM1). The Stacking (NN Meta) model was the only one that still maintained the highest accuracy out of all the models tested, demonstrating the power of having ensemble learning as well as deep learning applied techniques. The most relevant accuracy measures of the different datasets analyzed by each classifier are shown below:

TABLE II
Accuracy of Classifiers Across All Datasets

Classifier	CM1	KC1	KC2	PC1	JM1
Naïve Bayes	60.0%	66.5%	67.5%	60.8%	57.3%
KNN	78.9%	87.5%	85.5%	91.8%	82.6%
Decision Tree	88.9%	88.1%	86.1%	92.9%	86.1%
Random Forest	91.7%	91.3%	89.8%	97.1%	91.6%
Gradient Boost	93.9%	91.2%	89.2%	96.1%	88.5%
AdaBoost	87.8%	83.8%	84.3%	92.3%	73.6%
Bagging	92.8%	90.5%	89.8%	96.9%	90.4%
MLP	93.9%	86.2%	88.6%	96.6%	73.7%
Extra Trees	93.3%	91.5%	89.2%	97.3%	92.5%
Stacking (NN Meta)	**95.0%**	**93.8%**	**90.9%**	**97.6%**	**94.6%**

Advance oversampling was done to improve on the imbalance of the classes within the datasets. The main approach was SMOTE (Synthetic Minority Over-sampling Technique) that synthesizes the samples of the minority category through interpolating between other available ones. This is a method to avoid overfitting and increasing classification accuracy in the case of imbalance dataset. SMOTE is however sufficiently noisy on its own and does not pay attention to boundaries.

In address these limitations, a combined method that involved SMOTE, BorderlineSMOTE and ADASYN was employed. BorderlineSMOTE aims at synthetically generating data around class boundaries to boost performance where it is critical the most. ADASYN, however, accounts to the complexity of samples and increases the creation of synthetical data in sparse or complex areas. The second strategy, when merged with the former, is more efficient and flexible in terms of handling the issues of class imbalance.

- **Advantages:**
  - More precise balancing of data, especially near decision boundaries.
  - Improved classification performance for challenging datasets.
  - Effective for high-dimensional datasets.
- **Limitations:**
  - Slight increase in computational cost.
  - Possibility of introducing synthetic noise in certain cases.

The techniques that were important in this study are ensemble learning techniques, especially stacking. Stacking involves using training of several base classifiers and inserting the outputs of the classifiers as features in training a meta-classifier. This strategy was applied with the help of a deep neural network (Multilayer Perceptron) as meta-classifier, which is called NN Meta. The architecture of the model contains Random Forest, Extra Trees, LightGBM and CatBoost as base learners and the neural network consisted of 2 hidden layers (32 and 16 neurons), actiation function ReLU and Adam optimizer. To achieve conformity, the model was trained during 300 epochs.

- **Benefits of Stacking with NN Meta:**
  - Combines strengths of diverse models to reduce bias and variance.
  - Learns complex feature interactions through the neural network.
  - Outperforms traditional ensemble techniques such as bagging and boosting.

Summary of Key Findings:

- The proposed Stacking (NN Meta) model showed the highest classification accuracy across all datasets.
- The PC1 dataset was the most accurately classified by all models, indicating better data quality or clearer patterns.
- JM1 and KC2 datasets posed greater challenges, with generally lower accuracies.
- The combined oversampling approach significantly improved model robustness.
- The NN Meta model outperformed even strong competitors like XGBoost-based stacking.

To sum it up, the current work concludes that in addition to the deep learning meta-classifier, stacking is a very efficient framework of software fault prediction when it is augmented with sophisticated oversampling strategies. These two allow the model to generalize significantly and be better at making forecasts.

Conclusion and Future Work

Conclusion

This study aimed at coming up with a powerful and precise model of software faults prediction using an ensemble learning approach and other improved oversampling techniques. To properly evaluate the performance of the model, the research was carried out on a variety of real-life datasets; CM1, KC1, KC2, PC1, and JM1. The main task was to enhance the level of fault detection accuracy as well as consider the natural problem of data imbalance in the program defect data.

The experimental findings revealed that the conventional classifiers such as Naive Bayes, Decision Tree and K-Nearest Neighbors showed the moderate success rate, yet their performances drastically varied according to the nature of data sets. On the contrary, ensemble models (Random Forest, Gradient Boosting and Bagging) demonstrated a higher level of accuracy and a more stable result. But the best results were achieved with Stacking model using a neural network meta-classifier (NN Meta), which performed best in all datasets except one.

The NN Meta model finds its great effectiveness in its capability to integrate some strengths of the given base learners (i.e., Random Forest, Extra Trees, LightGBM, and CatBoost) and the representational competency of a neural network. The model could represent complex relationship and non-linear interactions between the prediction of the base models because of use of multilayer perceptron as the meta-classifier. The incorporation of much improved oversampling methods that include SMOTE, BorderlineSMOTE, and ADASYN also increased the robustness of the model, in that, both the majority and minority classes were satisfactorily balanced during the learning process.

To conclude, this paper confirms that there is need to combine ensemble techniques with deep learning approaches and data balancing to enhance software fault prediction. In addition to having the highest level of accuracy, the proposed NN Meta model has a high level of generalization ability thus forming the best solution that would be used in real software engineering industry where faults need to be identified early.

Future Work

Although the results achieved in this study are promising, there remain several areas for future exploration and enhancement. Further research can build upon the current work in the following ways:

- **Incorporation of Additional Meta-Learners:** Future models could explore different architectures for the meta-classifier, such as Gradient Boosting Machines, Long Short-Term Memory networks (LSTM), or transformer-based models, which may offer even greater accuracy and interpretability.
- **Real-Time Fault Prediction:** The current evaluation is limited to within-project scenarios. Future work could expand the model's applicability to fault prediction and real-time fault detection systems, enhancing its utility in dynamic software environments.
- **Scalability and Performance Optimization:** As the size and complexity of software projects grow, optimizing the training time and computational efficiency of the model becomes increasingly important. Implementing parallel processing, model pruning, or lighter neural architectures could improve scalability.
- **Feature Engineering and Dimensionality Reduction:** The effectiveness of the model can be further enhanced by applying sophisticated feature selection, dimensionality reduction techniques like PCA or t-SNE, and including semantic or contextual features such as developer behavior, code complexity, and versioning information.
- **Integration with CI/CD Pipelines:** The potential future directions might be targeted at the integration of fault prediction model into continuous integration and deployment (CI/CD) pipelines. This would allow the developers

to know of the issues in real-time and thus facilitate proactive software quality assurance.

- **Explainability and Interpretability:** Although neural networks are strong, they tend to be black boxes. The introduction of explainable AI (XAI) methods might assist the developers to comprehend why some of the modules are marked as defective, and it, therefore, boosts the level of trust in the system.
- **Testing on Diverse Datasets:** To assure this, the model has to be tested on larger scopes of data, those that belong to other domains, such as industrial projects and open-source offerings, which could be based on various fault properties.

The suggested stacking ensemble incorporating a neural network meta-classifier has redefined the bar of software defect prediction. This solution can turn into a central part of smart software quality assurance systems with additional progress in model architecture, integration, and deployment.

REFERENCES

[1] Arar, O¨ . F., & Ayan, K. (2021). A feature dependent Naive Bayes variant and its application to software fault prediction. *Applied Soft Computing*, 108, 107432.

[2] Sharma, S., & Jain, S. (2023). A comparative study of oversampling techniques for software defect prediction. *Software Quality Journal*, 31(1), 59–84.

[3] Zou, Y., Chen, Y., & Liu, J. (2022). DeepStack: Stacked generalization based deep learning ensemble for software defect prediction. *Journal of Systems and Software*, 183, 111121.

[4] Li, X., & Xie, X. (2020). An ensemble framework for software defect prediction based on voting and stacking. *IEEE Access*, 8, 63297–63311.

[5] Wang, J., & Zhang, H. (2021). Learning from class-imbalanced software defect data: An empirical study of data preprocessing methods. *Information and Software Technology*, 131, 106480.

[6] Khan, S., & Ahmad, I. (2023). Adaptive SMOTE-based ensemble learning model for fault detection in software systems. *Procedia Computer Science*, 219, 184–191.

[7] Tran, H., Nguyen, H. D., & Pham, H. (2022). Borderline-SMOTE CNN: A hybrid deep learning model for imbalanced software defect prediction. *Expert Systems with Applications*, 195, 116561.

[8] Silva, M. R., & dos Santos, R. (2021). SMOTEBoost for software defect prediction with class imbalance. *Information Sciences*, 548, 100–116.

[9] Awasthi, R., & Bhadoria, R. S. (2022). Comparative analysis of stacking and boosting classifiers in software fault prediction. *Journal of King Saud University - Computer and Information Sciences*, 34(5), 1913–1923.

[10] He, P., & Zhang, Y. (2023). Investigating ensemble learning with neural meta-learners for defect prediction. *IEEE Transactions on Software Engineering*, Early Access.

[11] Zhang, T., Lin, Y., & Wu, J. (2021). Performance optimization of deep ensemble networks for fault-prone module identification. *Neural Computing and Applications*, 33, 11589–11604.

[12] Jiang, Y., & Chen, L. (2020). Multi-level ensemble learning for high-dimensional software metrics. *Knowledge-Based Systems*, 194, 105590.

[13] Verma, S., & Pandey, P. (2023). Software defect prediction using LightGBM with synthetic oversampling. *International Journal of Information Technology*, 15(1), 219–226.

[14] Kaur, H., & Malhotra, R. (2021). Improving software fault prediction using neural network-based meta learners. *International Journal of Advanced Computer Science and Applications*, 12(4), 102–110.

[15] Guo, Y., & Wang, Q. (2022). Software defect prediction using hybrid ensemble with ADASYN and deep learning. *Computers in Industry*, 139, 103642.