ISSN No:-2456-2165

A Comprehensive Analysis of Ensemble-based Fault Prediction Models Using Product, Process, and Object-Oriented Metrics in Software Engineering

Atul Pandey[1] , Srujana Maddula[2] , Gaddam Prathik Kumar[3] , Sarthak Kumar Shailendra[4] , Karan Mudaliar[5]

[1] Btech Graduate, (Electrical & Electronics Engineering), Vellore Institute of Technology, Vellore, Tamil Nadu, India
[2] Data Scientist, Target, Benguluru, Karnataka, India
[3] Btech Graduate, (Computer Science & Engineering), Sharda University, Uttar Pradesh, India
[4] Btech Graduate, Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
[5] Btech Graduate, (Computer Science & Engineering), Vellore Institute of Technology, Vellore, Tamil Nadu, India

Abstract:- In the expansive domain of software engineering, the persistent challenge of fault prediction has garnered scholarly interest in machine learning methodologies, aiming to refine decision-making and enhance software quality. This study pioneers advanced fault prediction models, intertwining product and process metrics through machine learning classifiers and ensemble design. The methodological framework involves metric identification, experimentation with machine learning classifiers, and evaluation, considering cost dynamics. Empirically, 42 diverse projects from PROMISE, BUG, and JIRA repositories are examined, revealing advanced models with ensemble methods manifesting an accuracy of (91.7%), showcasing heightened predictive capabilities and nuanced cost sensitivity. Non-parametric tests affirm statistical significance, portraying innovation beyond conventional paradigms. Conclusively, these advanced models navigate inter-project fault prediction with finesse, signifying a convergence of novelty and performance. Simultaneously, anticipating fault proneness in software components is a pivotal focus in software testing. Software coupling and complexity metrics are critical for evaluating software quality. Object-oriented metrics, including inheritance, polymorphism, and encapsulation, influence software quality and offer avenues for estimating fault proneness. This study contributes a comprehensive taxonomy to the discourse, offering a holistic perspective on the multifaceted landscape of object-oriented metrics in fault prediction within the broader context of advancing software quality.

***Keywords*:-** *Software Fault Prediction; Object-Oriented Testing; Object-Oriented Coupling; Machine Learning, Ensemble Design, Product, and Process Metrics.*

I. INTRODUCTION

Software fault prediction has been a focal point in the software engineering domain for over three decades, garnering escalating attention from researchers [1]. The term "fault" denotes an erroneous step, process, or data definition in a computer program, commonly referred to as a "BUG." Scholars have approached the software fault prediction (SFP) challenge from two perspectives. Firstly, novel methodologies or combinations of existing methods have been introduced by researchers to enhance fault prediction performance. Secondly, the exploration of new parameters to identify the most influential metrics for fault prediction has been undertaken. Despite numerous approaches proposed in the literature, the classification of software modules as faulty or non-faulty remains a largely unresolved issue [2]. To address this challenge, scholars have increasingly turned to sophisticated techniques, including machine learning, deep learning, and unsupervised methods, indicating a shift towards novel and more compelling directions in fault prediction [3]. Machine learning algorithms have witnessed a surge in popularity over the last decade and continue to be one of the preferred methods for defect prediction [4]. As noted by Lessmann et al. [5], "There is a need to develop more reliable research procedures before having confidence in the conclusion of comparative studies of software prediction models."

In this study, the aim is to evaluate the performance of various classifier models without bias towards any specific classifier. Additionally, the reported efficacy of ensemble techniques by previous researchers [6] for enhancing fault prediction accuracy is recognized. Furthermore, the investigation into the diversity of classifiers within ensemble models has been identified as crucial for improving the effectiveness of ensemble designs [7]. This motivation propels the exploration into the design of ensembles to enhance the predictive capability of classifiers. In the context of the second viewpoint, a substantial body of research has been dedicated to investigating the utilization of software metrics derived from code to discern the fault proneness of software components. While fault estimation models predominantly rely on product metrics in the existing literature [8], those constructed through a synergy of product and process metrics remain relatively scarce [9]. Although some scholars have underscored the importance of integrating both product and process metrics in their studies, the broader incorporation of such models has been limited. Madeyski and Jureczko [10], in their research, ascertained that process metrics contribute valuable information to fault proneness determination. The utilization of process metrics in

fault ascertainment demonstrates the potential for superior outcomes when compared to reliance solely on product metrics. The imperative for further investigations to substantiate and refine these advanced models was underscored by their findings.

Radjenovic et al. [11], in a Systematic Literature Review (SLR), emphasized the necessity of identifying methodologies to measure and evaluate process-related information for fault proneness. Similarly, Wan et al. [12], in their study on perceptions, expectations, and challenges in defect prediction, concluded that software practitioners exhibit a preference for rational, interpretable, and actionable metrics in defect prediction. Additionally, the literature indicates not only the comparative superiority of process metrics over product metrics but also the proposition of alternative features based on developer-related factors, code smells, etc. [13]. This discernment necessitates further studies to meticulously examine the intricate association between metrics and fault proneness, thereby furnishing meaningful insights for informed decision-making. Consequently, the present study embarks on the development of advanced software fault prediction models that leverage a combination of metrics. Following the identification of a judicious set of product metrics, the research crafts advanced fault prediction models employing a systematic incorporation of process metrics, one at a time.

Motivated by the imperative to advance fault prediction models, this study establishes a comprehensive research framework characterized by meticulous pre-processing and feature extraction activities on datasets to identify pertinent metrics. Subsequently, diverse machine learning classifiers such as Naive Bayes (NB), Decision Tree (DT), Multilayer Perceptron (MLP), Random Tree (RT), and Support Vector Machine (SVM) are employed for training and testing experiments to evaluate the advanced models. The assessment involves a set of performance metrics encompassing accuracy, root mean square error (RMSE), F-score, and the area under the curve AUC(ROC).

Object-Oriented (OO) Metrics have been the subject of numerous proposals by researchers, resulting in metric suites designed for diverse perspectives within the context of Object-Oriented software. These suites find application in various contexts, serving as quality indicators, complexity measures, fault proneness predictors, and reliability measures. Table 1 below provides a comprehensive overview of the most frequently employed OO metrics documented in the literature.

Table 1. Object-oriented metrics

S.no.	Chidamber & Kemerer metrics (CK) [25]	Li and Henry Metrics [26]	MOOD Metrics [27]
1.	Weighted Methods Per Class (WMC)	N/A	Attribute Inheritance Factor (AIF)
2.	Depth of Inheritance tree (DIT)	Number of Methods NOM	Method Hiding Factor (MHF)
3.	Number of Children (NOC)	Message Passing Coupling (MPC)	Method Inheritance Factor (MIF)
4.	Coupling Between Objects (CBO)	Data Abstracting Coupling (DAC)	Attribute Hiding Factor (AHF)

Multiple classifiers are harmoniously combined to enhance overall performance, with a specific focus on improving fault-detection capabilities. Additionally, an examination of the cost sensitivity of the proposed ensemble-based classifier is undertaken. The outcomes of this analysis serve to validate the predictive efficacy of the proposed classifiers for the development of advanced fault prediction models.

The noteworthy contributions of this work can be delineated as follows:

- Establishment of a learning scheme comprising both base and ensemble learning classifiers.
- Construction and scrutiny of the predictive capability of advanced fault prediction models.
- Evaluation of the cost sensitivity of the proposed ensemble-based classifier through a comprehensive cost evaluation framework using Object oriented metrics and process metric.

II. RELATED WORK

Noteworthy contributions to the field of fault prediction have been documented through comprehensive surveys conducted by Catal and Diri [14], Li Zhiqiang et al. [1], Matloob et al. [7], and Radjenovic et al. [11]. These surveys encompass various aspects, including prediction models, modeling techniques, and the metrics employed. Radjenovic et al. [11] delineate that within the literature on fault prediction studies, process metrics constitute 24%, source code contributes 27%, and object-oriented metrics constitute 49% of the total. Prospective studies are urged to incorporate methodologies for measuring and evaluating process-related information for fault proneness in conjunction with product metrics.

In an empirical study conducted by Madeyski and Jureczko [9], utilizing both industrial and open-source software datasets, the significance of process metrics in enhancing results was notably observed. Emphasizing the need for replication using machine learning approaches, they underscore the uncertainty of features performing optimally in one method being equally effective in alternative approaches. Therefore, experimentation is warranted to explore the utility of both product and process-related metrics.

Khoshgoftaar et al. [15] extend the research landscape by constructing software quality models employing majority voting with multiple training datasets. This work presents an opportunity for further extension by incorporating data from diverse software project repositories. An analysis of the predictive capability of ensembles, in comparison to base classifiers, can offer insights into the efficacy of advanced fault prediction models.

Chen et al. [16] investigated to determine whether distinct cross-project defect prediction methods yield consistent identifications of defective modules. The outcomes of this study suggest the potential for extension through the application of learning approaches founded on ensemble design, thereby further enhancing the performance of cross-project defect prediction. A related exploration by Zhang et al. [17] delved into the utilization of various algorithms integrating machine learning (ML) predictors for cross-project defect prediction. However, to comprehensively examine the predictive capabilities of advanced algorithms, additional experimentation is deemed necessary.

An analysis of the aforementioned studies underscores the pivotal role of pre-processing techniques in significantly influencing the performance of learning algorithms. However, a notable gap in the literature pertains to the scarcity of investigations on larger datasets, essential for the development of generalized models. Moreover, the prevalent issue of class imbalance demands attention to augment the efficacy of fault prediction [7]. The exploration of parameter combinations remains a relatively underexplored aspect in existing literature studies. Consequently, there is an opportunity to replicate this work by incorporating more datasets, with a dedicated focus on product and process software metrics, and experimenting with diverse scenarios or combinations of models, encompassing both simple and advanced models, to attain heightened reliability and robustness.

Table 2: Existing Review

Authors	Metrics employed	Outcomes and proposed benchmark solutions
Chen et al. [16]	Process and Product	The researchers in this study explored the alignment of distinct cross-project defect prediction methods in identifying common defective modules. The outcomes suggest the potential for extension through the implementation of learning approaches founded on ensemble design, to enhance the overall performance of cross-project defect prediction methodologies.
Khoshgoftaar et al. [15]	Product and Process	The authors constructed software quality models employing a majority voting approach with multiple training datasets. This work could be extended by incorporating data from diverse software project repositories. Such an extension would facilitate an in-depth analysis of the predictive capabilities of ensembles in comparison to base classifiers, particularly in the context of advanced models.
Erturk and Sezer [18]	CK Product metrics	In their study, the authors concluded that the Adaptive Neuro-Fuzzy Inference System (ANFIS) outperforms the Neural Network (NN) and Support Vector Machine (SVM) approaches in predicting faults. Future research endeavors may consider incorporating process metrics into the analysis or developing advanced defect prediction models to further enhance the predictive capabilities.
Li et al. [19]	Code metrics	The authors provided a summary of defect prediction studies with a focus on emerging topics, including machine learning-based algorithms, data manipulation techniques, and effort-aware prediction strategies. They emphasized the importance of addressing the class imbalance problem and the need for developing models in the field of defect prediction.

In software engineering, a well-established principle emphasizes that high-quality software should exhibit low coupling and high cohesiveness. Noteworthy contributions to the study of cohesion metrics for fault prediction include the work of Marcus, Poshyvanyk, and Ferenc [28], who introduced the Conceptual Cohesion of Classes (C3) as a novel measure based on the textual coherence of methods. Utilizing an information retrieval approach supported by Latent Semantic Indexing, the study performed experiments on three open-source subject programs. The findings advocate the integration of structural metrics and cohesion metrics for enhanced prediction accuracy.

Similarly, Zhou, Xu, and Leung [29] conducted empirical evaluations on the effectiveness of complexity metrics in predicting software faults, employing CK metrics and McCabe metrics. Using data from three versions of Eclipse IDE, the authors compared the performance of LR, Naive Bayes, AdTree, K Star, and Neural networks. Results indicated that several metrics exhibit a moderate ability to differentiate fault-prone and fault-non-prone classes, with lines of code and weighted method McCabe complexity identified as robust indicators of fault proneness. The study underscores the significance of not only metric selection but also the size of datasets and feature extraction techniques in fault prediction endeavors.

Recent trends in software fault prediction underscore the increasing popularity of machine learning algorithms. Catal and Diri [30] empirically examined the impact of metric sets, dataset size, and feature selection techniques on fault prediction models, employing random forest (RF) and AIRS algorithms. The study concluded that RF algorithms performed better for large datasets, while Naive Bayes algorithms demonstrated efficacy for smaller datasets. Additionally, Alan [31] employed an RF machine-learning algorithm for outlier detection, selecting six metrics from the CK suite. The study highlighted the promising nature of threshold-based outlier detection, advocating its application before the development of fault prediction models.

ISSN No:-2456-2165

III. RESEARCH METHODOLOGY

In Phase I, the identification of a metrics suite is undertaken from metric datasets available in the PROMISE, BUG, and JIRA dataset repositories. Various pre-processing methods, including feature ranking methods, feature subset selection methods, and normalization, are employed to derive a reduced subset of features from the original dataset. This reduction is guided by a specific evaluation criterion, aiming to diminish feature space dimensionality, eliminate redundant and irrelevant information, and enhance data quality to improve the algorithm's performance. The experimental design incorporates N-fold cross-validation for training, testing, and replicating the experiment across diverse datasets. Phase II involves the evaluation of a simplified dataset under distinct scenarios: scenario-1 features a simple model based on product metrics; scenario-2 through scenario-5 explores advanced models incorporating additional process metrics (NR, NDC, NML, NDPV). These models are assessed using various base machine learning classifiers. Performance evaluation utilizes accuracy as key performance indices. To enhance the base machine learning classifier performance, classifier ensembles are designed through Bagging, AdaBoostM1 (a prominent boosting technique), and Voting algorithms. In Phase III, the focus shifts to examining the cost sensitivity of the proposed ensemble classifiers. This involves the development of a comprehensive cost analysis framework, facilitating a comparison between the best ensemble's cost and the best base classifier's cost through the determination of normalized fault removal cost.

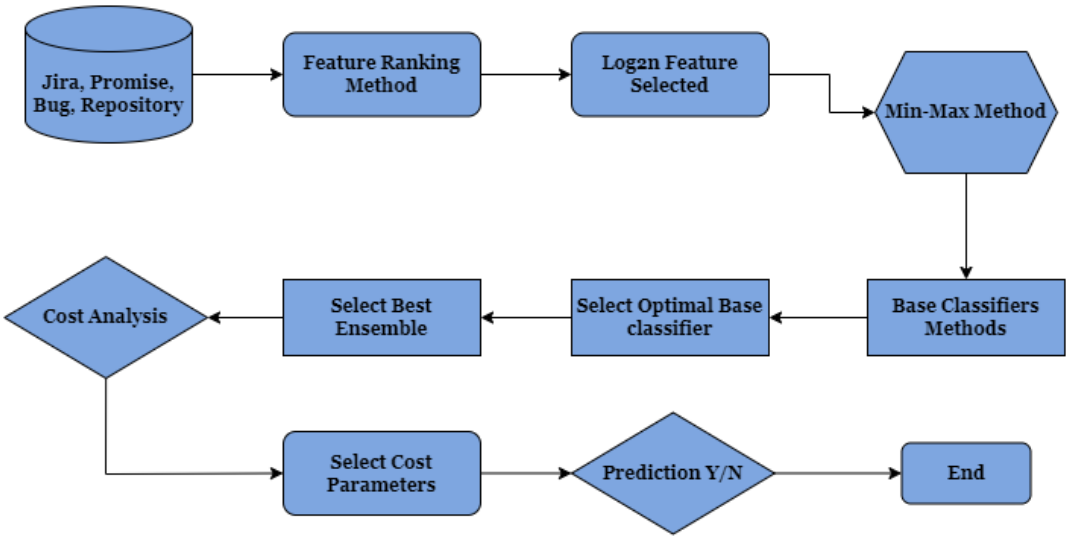

Fig 1. A framework of the Proposed ensemble model with cost analysis

In alignment with the insights derived from a comprehensive review of the existing literature and identifying potential research gaps, the following research questions have been formulated:

RQ1: How do the advanced defect prediction models, posited within the study, demonstrate performance variations across diverse machine learning classifiers?
RQ2: To what extent does the ensemble design contribute to enhancing classification performance compared to the individual machine-learning classifiers?
RQ3: Is there a discernible and statistically significant difference in performance among the base classifiers and ensemble classifiers?
RQ4: Within the context of a given software system, do the proposed ensembles exhibit a sensitivity to cost considerations?

The formulation of RQ1 and RQ2 is grounded in the intent to assess the efficacy of advanced models embodying distinct scenarios, characterized by a fusion of software product and process metrics. These models undergo training utilizing both base learning and ensemble-based classifiers, with their performances subjected to evaluation through metrics such as accuracy, RMSE, ROC(AUC), and F-score. The application of statistical tests is motivated by the aspiration to empirically substantiate the performance of predictors, thereby addressing RQ3. To address RQ4 and ascertain the cost-sensitivity of the proposed predictors, a comprehensive cost-based evaluation framework has been adopted.

For the experimental investigations, five distinct scenarios were devised by the outlined research questions. In Scenario 1, an assemblage of all product metrics was curated post-data processing and normalization, forming what is denoted as the "Simple model." The detailed selection of metrics is provided in Table 3. Subsequently, Scenario 2 introduced the "Advanced model-1," incorporating product metrics alongside a singular process metric (Product + NR). Similarly, Scenarios 3, 4, and 5 engendered the "Advanced model-2" (Product + NDC), "Advanced model-3" (Product + NML), and "Advanced model-4" (Product + NDPV), respectively. These designed models underwent testing across diverse project datasets from repositories such as PROMISE, Bug, and Jira, employing various classifiers, including DT, MLP, SVM, RT, NB, and classifier ensembles.

The performance assessment of the models—namely, "Simple model," "Advanced model-1," "Advanced model-2," "Advanced model-3," and "Advanced model-4"—comprised the utilization of accuracy. The metrics adopted in the base classifiers were obtained post-feature selection and ranking. N-fold cross-validation, with N set to 10 in this instance, facilitated the evaluation of base classifier performance, employing both training and testing phases. This methodology was consistently applied across various dataset versions for distinct base classifiers.

To address RQ2, which seeks to assess and compare the performance of diverse ensemble methods, relevant algorithm libraries were installed using the pip Python installer. Algorithms such as Bagging, AdaBoostM1, and Voting were deployed. Heterogeneous classifier ensembles adopted the majority voting method, while homogeneous ones employed both bagging and boosting methods. Boosting and bootstrap aggregating incorporated Decision Stump and REPTree as weak learners. AdaBoosting, involving repeated iterations with weight adjustments, and bootstrap aggregating, employing sampling with replacement, were integral components of this phase.

In light of RQ3, exploring potential statistically significant differences between base classifier and ensemble classifier performance, the authors employed Friedman's tests and Wilcoxon signed-rank tests. RQ4 delves into the cost sensitivity of the proposed ensembles, incorporating a normalized fault removal cost approach. To further scrutinize the cost sensitivity of the premier ensemble classifier, VOT-E2, concerning fault misclassification, a comparative analysis was conducted against the best-performing base classifier, MLP.

IV. RESULT AND DISCUSSION

Table 3. Summary of research questions

Research question	Discussion
Research Question 1 (RQ1): What is the performance of the advanced defect prediction models proposed in the study when subjected to various machine learning classifiers?	Each model underwent testing across diverse project datasets sourced from PROMISE, BUG, and JIRA repositories. Employing distinct classifiers such as DT, MLP, SVM, RT, and NB, the performance of the models was assessed.
Research Question 2 (RQ2): To what extent does ensemble design enhance classification performance compared to individual machine-learning classifiers?	In a comprehensive evaluation, ensemble methods demonstrated an overall median F-score ranging between 76.50% and 87.34%, and ROC (AUC) values between 77.09% and 84.05%. In contrast, base classifiers achieved an average F-score ranging between 73% (Simple model) and 83% (Advanced model-2) for the PROMISE dataset, and ROC (AUC) values between 60% (Advanced model-4) and 79% (Advanced model-2). This observation underscores the efficacy of ensemble design in leveraging the strengths of multiple predictors, contributing to the advancement of fault prediction methodologies.
RQ3: Whether there exists any statistically significant performance difference among the base classifiers and ensemble classifiers?	For pairwise comparisons, the Wilcoxon signed-rank test was employed. The outcomes from both Friedman's tests and Wilcoxon signed-rank tests provide statistical evidence supporting the existence of significant performance differences, particularly highlighting the unique standing of the ensemble method.

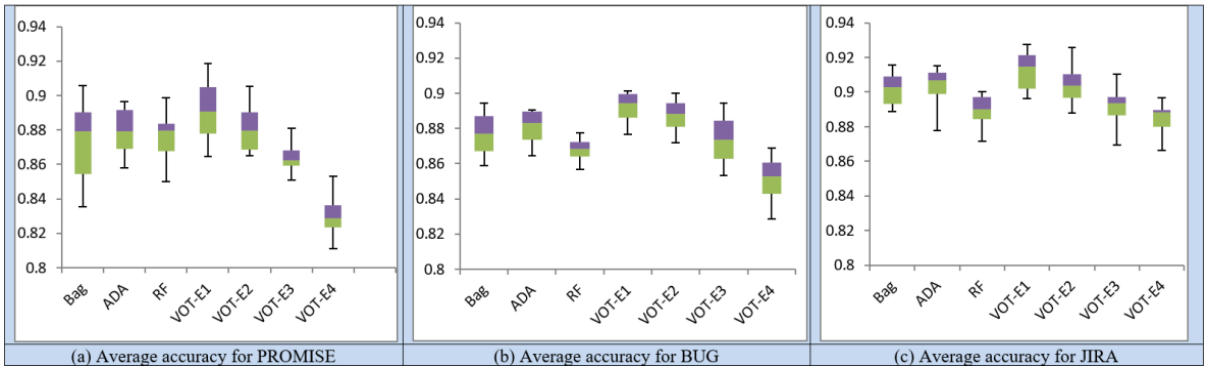

Fig 2. Box plots for Ensemble Results for average accuracy.

For the Promise dataset, the average accuracy for MLP in the simple model is 91.7%, advanced model-1 is 80%, advanced model-2 is 87%, advanced model-3 is 85%, and advanced model-4 is 79%. Notably, the bar graph in Figure 2 illustrates that the average accuracy for MLP is higher in advanced model-2 than in advanced model-3, advanced model-1, and the simple model. Similarly, the average accuracy for DT in the simple model is 74%, advanced

model-1 is 81%, advanced model-2 is 87%, advanced model-3 is 83%, and advanced model-4 is 77%. The corresponding bar graph (Figure 2a) indicates that the average accuracy for DT is higher in advanced model-2 compared to advanced model-3, advanced model-1, and the simple model.

The integration of machine learning techniques into software fault prediction has emerged as a recent and dynamic area of research. Researchers employ diverse machine learning methodologies across multiple dimensions within this domain, encompassing feature selection, classification, outlier detection, and model building. This study provides a comparative analysis of recent literature, shedding light on the various applications of machine learning in software fault prediction.

Researchers exhibit significant diversity in their approaches, encompassing variations like data, metrics under consideration, employed machine learning algorithms, and the tools utilized for experimentation. The following table (Table 4) presents a comprehensive overview of recent studies, offering a qualitative analysis of the machine learning algorithms utilized for fault prediction. This comparative analysis aims to provide insights into the nuanced differences and trends prevalent in the application of machine learning techniques in software fault prediction studies.

Table 4. Comparative analysis

Existing Research	Algorithm employed	Metrics evaluated	Programming Language
[32]	Logistic Regression	C-K	C++
[28]	Logistic Regression and PCA	Conceptual Cohesion of Classes	C++
[33]	Naive Bayes network, Random Forest	C-K	Java
[30]	Random Forest, J48	McCabe [36], Halstead [37]	C++
[34]	Decision tree and Neural network	C-K	Java

The presented table indicates the widespread popularity of logistic regression, random forest, and neural networks within the domain of fault prediction studies. Machine learning algorithms offer versatile applications for conducting diverse statistical and predictive analyses of Object-Oriented (OO) metrics. The WEKA platform serves as a comprehensive tool for executing and analyzing these algorithms. A succinct overview of the regression model and other pertinent techniques is provided below.

Logistic regression, a statistical classification technique rooted in maximum likelihood estimation, is applicable in two modes: univariate regression and multivariate regression. Univariate LR is employed for the isolated analysis of a single metric on fault proneness. In contrast, Multivariate LR proves useful when multiple metrics need assessment for their impact on fault proneness. LR is employed when there is one or more than one independent variable. The objective of LR is to construct the best-fitting model that elucidates the relationship between dependent and independent variables. The result of LR is expressed through a fitted logistic regression equation.

Learning can be categorized into supervised or unsupervised forms. In supervised learning, a dependent variable can be predicted from a given set of independent variables. A map function is generated using these variables to produce the desired outcome. Numerous research studies have leveraged various machine learning algorithms to predict the impact of Object-Oriented (OO) metrics on software fault proneness. For instance, in a study [37], a Decision tree was employed, and validation was conducted using the receiver operating characteristic (ROC) curve. The primary advantage of Decision trees lies in their ability to implicitly identify the most influential features from the dataset, and their performance is not influenced by the type of relationship between attributes. Machine learning algorithms like random forests are suitable for handling multiclass data, while Bayes networks rely on rules of probability for prediction. In certain studies [35-37], a set of learning algorithms, including Bayes networks, random forests, and NNge (nearest neighbor with generalization), were applied for a comparative analysis of prediction models. The Artificial Immune Recognition System (AIRS), inspired by the vertebrate immune system, is a machine-learning algorithm capable of working with both nominal and continuous data. The study that applied AIRS found its performance to be superior to J48. Principal Component Analysis (PCA) serves as a feature selection technique, emphasizing variation and producing strong patterns in the dataset. Some studies have utilized PCA for fault prediction and feature selection in the context of OO metrics. Additionally, Neuro-fuzzy and Latent Semantic Indexing are other competitive algorithms explored for prediction purposes.

V. CONCLUSION

This study introduces advanced models for software fault prediction, leveraging information related to both product and process metrics. The investigation involved forty-two open-source code projects extracted from Promise, Jira, and Bug repositories. Results indicate that the MLP-based base classifier exhibits superior performance, as reflected in high average accuracy (91.7%). Ensemble methods, incorporating bagging, boosting, and voting, further enhance classification performance, with VOT-E2 (DT + MLP + SVM) producing the best results. Statistical tests confirm significant performance differences between base classifiers and ensemble classifiers, validating the predictive capability of the proposed models. The study

emphasizes the potential utility of the combination models for developing advanced defect prediction models, providing valuable insights for software engineers in new projects. While the experiments utilized datasets from Promise, Jira, and Bug repositories, extending the investigation to more open-source and cross-project datasets would enhance the generalization of results. This research article also presents a comprehensive taxonomy of object-oriented (OO) metrics usage for fault proneness prediction, emphasizing their significance in determining software quality. Various machine learning algorithms have been applied in fault prediction, with opportunities for further exploration in sub-domains such as Support Vector Machine, Dimensionality Reduction, Gradient Boosting, and Deep Learning. While existing research primarily focuses on fault prediction, the application of OO metrics can extend to other testing phase activities, including test case selection, generation, prioritization, and clone detection. Future research directions may involve developing tools for extracting OO metrics from software, contributing to the efficiency of code analysis. The provided set of extensively used datasets and software tools, discussed in the paper, facilitates the evaluation of techniques/methodologies. The integration of predictive measures based on OO metrics into testing processes can optimize fault localization, refactoring, debugging, and test case minimization, potentially minimizing software maintenance costs. Ongoing research in object-oriented software testing aims to explore the impact of OO metrics on software maintenance for a more accurate examination of the problem.

REFERENCES

[1]. Z. Li, X.Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," Iet Software, Vol. 12, No. 3, 2018, pp. 161–175.
[2]. Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," IEEE Transactions on Software Engineering, Vol. 37, No. 3, 2010, pp. 356–370.
[3]. X. Yang, D. Lo, X. Xia, and J. Sun, "TLEL: A two-layer ensemble learning approach for just-in-time defect prediction," Information and Software Technology, Vol. 87, 2017, pp. 206–220.
[4]. L. Pascarella, F. Palomba, and A. Bacchelli, "Fine-grained just-in-time defect prediction," Journal of Systems and Software, Vol. 150, 2019, pp. 22–36.
[5]. S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," IEEE Transactions on Software Engineering, Vol. 34, No. 4, 2008, pp. 485–496.
[6]. S.S. Rathore and S. Kumar, "An empirical study of ensemble techniques for software fault prediction," Applied Intelligence, Vol. 51, No. 6, 2021, pp. 3615–3644.
[7]. F. Matloob, T.M. Ghazal, N. Taleb, S. Aftab, M. Ahmad et al., "Software defect prediction using ensemble learning: A systematic literature review," IEEE Access, 2021.
[8]. R. Jabangwe, J. Börstler, D. Šmite, and C. Wohlin, "Empirical evidence on the link between object-oriented measures and external quality attributes: A systematic literature review," Empirical Software Engineering, Vol. 20, No. 3, 2015, pp. 640–693.
[9]. I. Kiris, S. Kapan, A. Kılbas, N. Yılmaz, I. Altuntaş et al., "The protective effect of erythropoietin on renal injury induced by abdominal aortic-ischemia-reperfusion in rats," Journal of Surgical Research, Vol. 149, No. 2, 2008, pp. 206–213.
[10]. L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? An empirical study," Software Quality Journal, Vol. 23, No. 3, 2015, pp. 393–422.
[11]. D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," Information and software technology, Vol. 55, No. 8, 2013, pp. 1397–1418.
[12]. Y. Wu, Y. Yang, Y. Zhao, H. Lu, Y. Zhou et al., "The influence of developer quality on software fault-proneness prediction," in Eighth International Conference on Software Security and Reliability (SERE). IEEE, 2014, pp. 11–19.
[13]. C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu, "Don't touch my code! Examining the effects of ownership on software quality," in Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, 2011, pp. 4–14.
[14]. C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," Information Sciences, Vol. 179, No. 8, 2009, pp. 1040–1058.
[15]. T.M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in 22nd IEEE International conference on tools with artificial intelligence, Vol. 1. IEEE, 2010, pp. 137–144.
[16]. X. Chen, Y. Mu, Y. Qu, C. Ni, M. Liu et al., "Do different cross-project defect prediction methods identify the same defective modules?" Journal of Software: Evolution and Process, Vol. 32, No. 5, 2020, p. e2234.
[17]. Y. Zhang, D. Lo, X. Xia, and J. Sun, "Combined classifier for cross-project defect prediction: An extended empirical study," Frontiers of Computer Science, Vol. 12, No. 2, 2018, p. 280.
[18]. E. Erturk and E.A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert systems with applications, Vol. 42, No. 4, 2015, pp. 1872–1879.
[19]. Z. Li, X.Y. Jing, and X. Zhu, "Heterogeneous fault prediction with cost-sensitive domain adaptation," Software Testing, Verification, and Reliability, Vol. 28, No. 2, 2018, p. e1658.
[20]. T. Wang, W. Li, H. Shi, and Z. Liu, "Software defect prediction based on classifiers ensemble," Journal of Information and Computational Science, Vol. 8, No. 16, 2011, pp. 4241–4254.

[21]. K. Bańczyk, O. Kempa, T. Lasota, and B. Trawiński, "Empirical comparison of bagging ensembles created using weak learners for a regression problem," in Asian Conference on Intelligent Information and Database Systems. Springer, 2011, pp. 312–322.
[22]. G. Catolino and F. Ferrucci, "An extensive evaluation of ensemble techniques for software change prediction," Journal of Software: Evolution and Process, Vol. 31, No. 9, 2019, p. e2156.
[23]. L. Reyzin and R.E. Schapire, "How boosting the margin can also boost classifier complexity," in Proceedings of the 23rd International Conference on Machine Learning, 2006, pp. 753–760.
[24]. J. Petrić, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an ensemble for software defect prediction based on diversity selection," in Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2016, pp. 1–10.
[25]. Chidamber SR, Kemerer CF. Towards a metrics suite for object-oriented design. ACM; 1991 Nov 1.
[26]. Li W, Henry S. Object-oriented metrics that predict maintainability. Journal of systems and software. 1993 Nov 1;23(2):111-22.
[27]. Abreu FB, Carapuça R. Object-oriented software engineering: Measuring and controlling the development process. In Proceedings of the 4th International Conference on software quality 1994 Oct 3 (Vol. 186, pp. 1-8).
[28]. Marcus A, Poshyvanyk D, Ferenc R. Using the conceptual cohesion of classes for fault prediction in object-oriented systems. IEEE Transactions on Software Engineering. 2008 Mar;34(2):287-300.
[29]. Zhou Y, Xu B, Leung H. On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. Journal of Systems and Software. 2010 Apr 1;83(4):660-74.
[30]. Catal C, Diri B. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Information Sciences. 2009 Mar 29;179(8):1040-58.
[31]. Alan O, Catal PD. An outlier detection algorithm based on object-oriented metrics thresholds. InComputer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on 2009 Sep 14 (pp. 567-570). IEEE.
[32]. Basili VR, Briand LC, Melo WL. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering. 1996 Oct;22(10):751-61.
[33]. Catal C, Diri B. Software fault prediction with object-oriented metrics based artificial immune recognition system. In International Conference on Product Focused Software Process Improvement 2007 Jul 2 (pp. 300-314). Springer, Berlin, Heidelberg.
[34]. Gyimothy T, Ferenc R, Siket I. Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Transactions on Software Engineering. 2005 Oct;31(10):897-910.
[35]. Thwin MM, Quah TS. Application of neural networks for software quality prediction using object-oriented metrics. Journal of systems and software. 2005 May 1;76(2):147-56.
[36]. Zhou Y, Leung H. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. IEEE Transactions on Software Engineering. 2006 Oct;32(10):771-89.
[37]. Singh Y, Kaur A, Malhotra R. Empirical validation of object-oriented metrics for predicting fault proneness models. Software quality journal. 2010 Mar 1;18(1):3.