Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 199–209

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

Ensemble Machine Learning Paradigms in Software Defect Prediction

Tarunim Sharma[a] , Aman Jatain[b] , Shalini Bhaskar[c] and Kavita Pabreja[d]

[a] *Research Scholar,* [a,b,c]*Amity School of Engineering and Technology, Gurgaon, Haryana, India*

[d] *Maharaja Surajmal Institute, Janakpuri, New Delhi, India*

Abstract

Predicting faults in software aims to detect defects before the testing phase, allowing for better resource allocation and high-quality software development, which is a requisite for any organization. Machine learning techniques aid in the resolution of such issues and a variety of predictive models are being developed to categorize the software into defective modules and the one which is non-defective ones. Though applying these advanced machine learning techniques results in better utilization of time and other resources, there is still poor prediction as reported in many studies. This is because of several challenges that block defective software data, including redundancy, correlation, feature irrelevance, missing samples, and an imbalanced distribution between the faulty and non-faulty classes. Ensemble Machine learning has been adopted by practitioners and researchers globally to deal with such problems, and it is proven to demonstrate some improvement in defect prediction performance. In this review paper, all ensemble-based machine learning techniques developed for software defect prediction from 2018 to 2021 have been critically analyzed. The nucleus of this paper is to get a deep insight into why the various hybrid models still suffer from poor performance on the available datasets. A detailed review with a focus on multiple perspectives viz. faulty and non-defective datasets, performance evaluation criteria, and machine learning techniques have revealed certain gaps that can be addressed by developing more robust hyperparameter optimization algorithms, feature engineering, developing stacking and averaging models.

Keywords: Software Bug Prediction; Ensemble Learning; Classification, Software Metrics; Machine Learning Models; Software Quality Assurance; Class Imbalance.

1. Introduction

Software engineering is an area of engineering that creates diverse software products that are trustworthy, productive, and efficient in doing what they are supposed to do. It is concerned with all areas of software development, starting from software specification to maintenance after the software is handed over to the user. This implies that the quality of the software is entirely dependent on the process adopted to develop software and how successfully testing is carried out. Unfortunately, errors can occur at any point in the process followed for developing the software. The ability of software

*Tarunim Sharma. Tel.:9650215421
E-mail address.: Tarunimsharma@gmail.com

to function without failure determines its reliability. Predicting software problems in the initial phases of developing software has thus become a major concern in the field of logistics software, as many firms have invested significant resources in correcting errors caused by software faults. According to The Standish Group's previous report, only 16.2 percent of IT projects were rated successful in delivering all the functionalities they promised. Most initiatives were over budgeted, did not meet deadlines, and did not comply with finalizing functionalities. Moreover, a whopping 31.1 percent of IT expenditures were categorized as failures which means they have to be discontinued. The figures are depressing and worrisome for any company thereby attempting to improve its operations through early fault detection [1]. According to many kinds of research, correcting faults or issues after delivery costs more than five times as much as detecting and fixing bugs during the testing phase. According to a report released in Times Now by the “Consortium for Information & Software Quality”, poor software quality seems to be costly for firms across all U.S. industries $2.08 trillion in 2020. (CISQ). The predicted expenses because of software bug detection in later stages are the result of failed IT and software projects [2].

Detecting bugs in the software is an important part of developing software since it predicts which modules are more likely to have contained defects. It seeks to lower the software development cost by concentrating testing efforts on the projected defective components. Software defect prediction research has flared up as a result of the constraint of resources and the demand for high-quality software so that quality can be carefully inspected and implemented before the software is released. The majority of current testing methodologies do not ensure maximum coverage of statements. As a result, there is a need to have efficient algorithms to detect software components that are said to be faulty ones always. For fastening the process of detection Software defect prediction is becoming popular as it is further helping in lowering the expense incurred in finding and analyzing defects. Preventing a problem is usually preferable to waiting for it to arise on its own. Defects or vulnerabilities that go undetected build up to an increased expense in the form of cost-to-fix or remediation. Static analysis using automated tools provides immediate insight into the source of the problem and makes it straightforward to tackle long-standing issues. More traditional ways for detecting defects were through the preparation of activities like reviews, walkthroughs, code inspections, and testing, but they can also be discovered by accident. Early detection of flaws saves money and, in the long run, proves to be the most accurate approach to designing secure and strong software. Artificial Intelligence-based techniques viz Machine Learning, blending various weak learners’-based algorithms, and Deep Learning help in changing the standard of finding bugs in the software, according to Alsaeedi & Khan [3]. They are commonly used to forecast flaws in freshly generated modules based on models learned from previous software failures and updated details. During the last two decades number of approaches based on different software, and metrics have also been explored and implemented [4]. According to studies, many researchers recommended Ensemble models for detecting defects in the software by using sampling, dimension reduction, and feature selection strategies during the data preparation step. Learning multiple classifier systems is another name for ensemble learning (Zhi-Hau, 2012). It is built on the concept of combining judgments from several weak learners to facilitate accurate and improved decisions. Individual model accuracy is lower than ensemble model accuracy, which combines many of them. Noise, variation, and bias are the most common issues in learning models. Ensemble techniques help to eliminate these error-causing factors, resulting in more accurate and stable machine learning (ML) algorithms with improved performance.

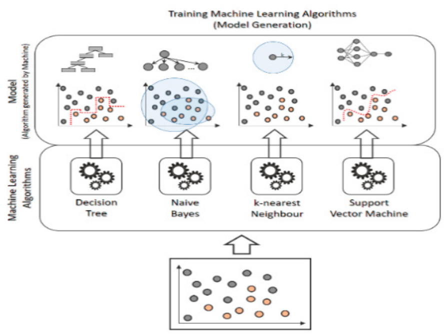
Training Machine Learning Algorithms
(Model Generation)
Model
(Algorithm generated by Machine)
Machine Learning Algorithms
Decision Tree
Naive Bayes
k-nearest Neighbour
Support Vector Machine

Fig 1. Ensemble Learning Algorithms.
Source: Data-Science-blog on Ensemble Learning [5]

The objective of this study is to critically analyze the application of the ensemble method of machine learning technique in the field of software defect prediction. Publications from till date 2018 have been examined from a multidimensional perspective, including the selection of a particular machine learning algorithm, the specific contribution of the authors, and the research gap that may lead to the future scope of the work that can be undertaken. The paper is organized as follows section 1 discusses types of machine learning techniques, and Section 2 emphasizes the adoption of machine learning ensemble techniques by various authors globally intending to obtain an accurate prediction of defects. In this section, the critical analysis of various research works has been illustrated and their characteristics along with advantages and disadvantages, Last Section focuses on the conclusion drawn from this study and throws light on a future direction as well.

2. Machine Learning Techniques

Machine Learning techniques have grown in popularity in recent years as a result of their remarkable attributes and technological developments. In today's increasingly competitive world, machine learning is enabling companies to accelerate their digital transformation and move into the age of automation. Algorithms implemented using machine learning are directed to make predictions or categorizations based on approaches using statistical techniques to uncover significant observations in regards to software as these deep insights will further affect the application-related number of decisions, hopefully impacting key growth key performance indicators. In terms of research and analysis, two main types of machine learning approaches were noticed that are being used while discovering bugs in the software during the study: Supervised and Unsupervised learning others can be Semi-supervised and Reinforcement Learning though they are rarely used in the area of software defect prediction. On the other hand, more, advanced technology Deep Learning is also gaining popularity in the detection field.

2.1 Supervised Learning

Regarding Software Bug prediction, Supervised Learning is regarded as the most prevalent method. The functionality is defined by its use of labeled datasets to train algorithms that reliably predict class [6]. There are two approaches to classification algorithms one divides the data with training, validation, and testing, other uses cross-validation techniques. Selective features of the supervised learning algorithm that are the backbone of the ensemble model are given below:

- The model's main purpose is to predict the labels, or values, of unseen occurrences based on their attributes where the label is the value of a data instance's selected target attribute, which is forecasted using another attribute's information.
- If the labels are classes, the Machine Learning task is a classification problem; if the labels are numbers in a continuous space, it is a regression problem.
- If a machine is to predict how much defective software will be created, the training data must comprise each software instance as a data instance, software features as attributes, and whether or not the software is defective as an output class. To identify whether or not the modules are defective, this supervised machine learning approach is applied.
- After obtaining the defect count data, a machine learning model may be developed that can be used to check new software before completing testing to determine whether faults are there or not at the early stages of development, hence ensuring the software's reliability and quality.
- The employment of data sampling techniques is a common strategy to deal with the problem of class imbalance which is mainly being accompanied while the software is detected for finding defects as the number of defective elements is few as compared to non-defective ones as it often hinders the accuracy of the prediction models.

By adopting several techniques based on Machine Learning defects can be detected at the earliest saving further saving from penalties and resources can also be amicably allotted to the modules having defects. A summary of the working of the most commonly used supervised software defect prediction is presented in Table I. Bowes [7] recommended using classifier ensembles to accurately predict software faults. Several works in the field of predicting defects in the software. Frameworks encompassing feature selection methods, handling class imbalance techniques, and reducing dimensions along with ensemble learning are presented by various researchers but they are lacking in one or the other way. Many ensemble methods have also been adopted by researchers globally of which Random forest, boosting, and bagging are the most common as compared to Stacking, voting, and Extra Trees [8].

Table1. A detailed description of Supervised Defect Prediction Techniques.

Supervised Techniques	
SDP Techniques	Description of techniques
Random Forest	The popular algorithm is based on machine learning that can be employed for both classification and regression-based problems. The concept on which it has laid down its foundation is ensemble-based learning that combines various classifiers to solve the problem and further enhance the performance of models. It works by taking output based on majority voting that conveys that instead of relying on one decision tree speculations from several trees are being considered and analyzed. The more the number of trees being estimated better will be the accuracy of prediction and the overfitting problem is also being evaded.
Bagging	Commonly known as Bootstrap Aggregation, is an Ensemble approach. It selects a random sample of data from the entire set. As a result, each model is created using row sampling to replace the samples provided by the original data. Each model is now trained independently, and the results are generated. After merging the findings of all models, the outcome is based on majority voting. Aggregation is the process of integrating all of the findings and generating output based on majority voting [9]. Bagging aims to decrease variance not bias.
Boosting	An ensemble learning strategy for minimizing training errors by combining a group of weak learners into strong learners. This model is fitted with a sample of data chosen randomly which is further trained sequentially. Then the second phase involves developing another model in which compensation for the flaw's effort is being made by each model for the one before it. This approach is repeated until either the entire training data set is properly predicted or the maximum number of models has been added. In each cycle of redoing rules lacking in excellence are combined to form a single, unique and strong rule [10] that performs prediction aiming to lessen the issue of biasing not variance.
Stacking	Technique for examining several different models for the same problem. For that reason, several different learners can be generated and each one is used to create an intermediate prediction. Then a new model is added that uses the intermediate forecasts to learn the same aim. The name stems from the belief that the final model is stacked on top of the others. Thereby improving the overall performance and ending up with a model that's better than any intermediate model.
Voting	The technique where the classifier learns from several models and gives the result based on the assumption that the output class has the highest probability of being chosen as the desired class. Outcomes from each classifier are being collected and fed into the voting classifier and prediction is done based on aggregation and declaring the output class as the one having the highest majority of voting. Rather than building separate specialized models and determining their performance voting proposes a single model that learns from several models and for each output class, the output is predicted based on their combined majority voting.
Extra Trees	An "additional trees" classifier is a random forest version known as a "very randomized trees" classifier. In this, each step uses the entire sample, and decision limits are picked at random rather than the optimal option. In this forest, all decision-making trees are made using identical training samples. Reaching the test node, the tree is provided with a random sample having K number of features from which the most appropriate feature is chosen using the mathematical equation which further splits the data. So, these randomly selected samples of features act as a base for giving rise to decision trees that are de-correlated.

Table 2. Supervised Software Defect Prediction Techniques Pros and Cons.

Supervised Prediction Techniques	
Advantages	Disadvantages
Flexible with both classification and regression problems and reduces the problem of overfitting, improving the accuracy of detecting defects in software thereby.	As outputs of the number of trees are combined it requires lots of resources, processing power, and training time.
Normalizing of data is not required as it uses a rule-based approach.	It is also unintelligible, as it fails to determine the relevance of each variable in software.
It aids in the reduction of variance.	A high biasing problem can occur if it is not implemented properly and may also lead to an overfitting problem.
Training may be completed quickly on large datasets and provides more accuracy as compared to other methods.	As we are combining multiple models in this algorithm it seems to be computationally expensive and not feasible to implement sometimes.
It provides the benefit of allowing a group of weak learners to work together to outperform a single strong learner.	High sensitivity towards outliers affects the accuracy of prediction.
It may be used to generate fresh estimates and to show the best strategy to integrate expense estimates.	Expensive from a Time and Computation point of view.

2.2 Unsupervised Learning

It is a machine-based technique in which models are not supervised using a labeled training dataset but are trained using an unlabelled dataset. In this model, the hidden patterns are decreased using insights from the given data without any human intervention. Unsupervised learning cannot be directly applied to a regression or classification problem as output data is not present corresponding to input data. The strategy that helps in implementing unsupervised-based learning is clustering which further helps in categorizing the data objects into different classes or clusters. It is used to organize a collection of objects such that similar objects are grouped. Clustering analyses software quality by using either defective or non-defective software measurement datasets. In this scenario, data is divided into two clusters based on whether it is defective or not. The datasets are then clustered using appropriate techniques. Defective and non-defective modules are predicted using software defect prediction algorithms. We can also use clustering methods to assess software quality. One of the features of Clustering Algorithms is that they work well in presence of noise and need not specify clusters in advance.

Table 3. Unsupervised Software Defect Prediction Techniques.

Unsupervised Software Defect Prediction Techniques	
K-Partition Clustering	Based on their distance, an algorithm splits a given data set into a set of k groups, often known as clusters of spherical shape.
Hierarchical Clustering	Unlabelled data is organized into a hierarchy of clusters based on their similarity. Objects in clusters share similar characteristics.
Density-based clustering	Looks for core objects with dense surrounding areas that contain noise and outliers to find aspherical clusters.
Neural Network Clustering	Typical clustering method based on model thinking.

Fuzzy logic	Each data point can potentially belong to two or more clusters. The resulting partition is therefore said to be a fuzzy partition.
Expectation Maximization Clustering	Gets cluster membership probabilities based on probability distributions.
Spectral Clustering	Before clustering, use the eigenvalues of the similarity matrix to reduce dimensionality.
Optimization Clustering Optimization	Considering clustering as an optimization problem, Particle Swarm Optimization can be used to solve it.

Unsupervised Software Defect Prediction models were shown to be comparable to Supervised Defect Prediction models in terms of prediction in many pieces of research. This suggests that defect prediction models based on unsupervised models do not seem to be as problematic as previously thought and that they should be evaluated in most cases where labeled training data is insufficient. Finally, researchers have discovered that dataset properties have a significant effect on predicting performance regardless of whether unsupervised or supervised-based models are being used for finding defects in the software. As a result, it'll be worthwhile to dig deeper into which of these dataset properties are the most relevant. Similarly, investigating the learner-dataset interaction could be helpful, as it will always be the dilemma that which particular learning system will work with excellence, supervised or unsupervised [11].

3. Related Work

This section provides a summary of recent research on the subject of detecting defects in the software undertaken by various researchers. The study focuses on studies that developed a fault-free prediction model for predicting software flaws using an ensemble method, sampling, dimension reduction, and pre-processing techniques. The majority of studies are used to construct software prediction models, which allow software analysts to focus more on the code, provide an estimate of how much effort is necessary, and whether the module has problems, resulting in higher-quality software and better resource utilization as it was shown by Iqbal [18] and Li [19]. Researchers evaluated the ensemble learners' capacity for prediction using a variety of combinations of performance criteria. The distribution of research across performance indicators is depicted in Fig. 2.

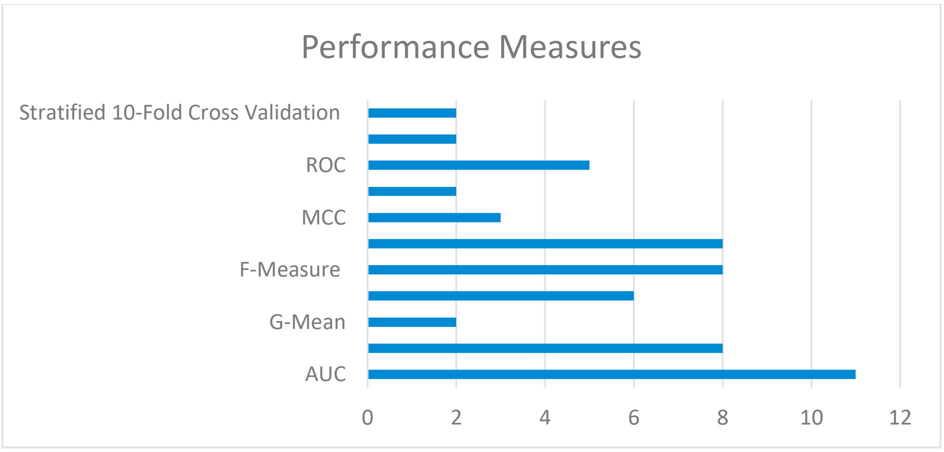

Fig. 2. Distribution of research across Performance Indicators.

Software defect data is always present in large quantities containing many features in it causing redundancy, which adds to the complexity because of which the model's performance and classification effect are harmed. As a result, feature selection is a critical step that can be accomplished by utilizing several methods like filtering, embedding, and encapsulating. Suresh Kumar [23], Ali [24], and Balogun [25] employed characterized code feature, discriminating feature cost sensitive based, and selection approaches based on multi filtering of features for predicting defects in the software along with bootstrap and rank-based aggregation ensemble learning. Defect prediction datasets are highly skewed, with the number of samples for one class far exceeding that of another, implying that faulty classes are fewer in number than non-faulty ones. As a result, imbalanced learning algorithms have been introduced to improve the performance of most conventional classifiers, as demonstrated by Qui [12], Shamsul Huda [14], Khuat [16] [21], Chakraborty [20], and Malhotra [22] in their respective research by developing hybrid models or sampling-based classifier using number of ensemble techniques. According to the literature review, ensemble learning has better prediction power to detect software defects, as evidenced by the work of Balogun [13], Mousavi [15], and Mehta [26] who use ANP-based evaluation methods, over bagging, static and dynamic approaches, and heterogeneous ensemble classification based on segmented patterns methods for detection. The below table summarises the findings and conclusions reached after reviewing the literature. The author identified research gaps after analyzing the literature in many dimensions based on ensemble learning.

Table 4. Literature Survey

Author	Year	Dataset	Accuracy Evaluation Parameters	ML Algorithm	Gaps
Shaojian Qiu, Lu, Siyu Jiang, Yang Guo [12]	2018	AEEEM, NASA, PROMISE, RELINK, SOFTLAB	AUC, Mathews Correlation Coefficient, Probability of detection, Probability of False Alarm	Sampling, Cost-Sensitive algorithms, Ensemble, Imbalance Techniques, Random Forest, Naïve Bayes, Support Vector Machine, Logistic Regression	▪ To find a better solution for CIP in Cross Project Defect Prediction. ▪ Carrying out Cross Project Defect Prediction-related tasks on real-world commercial software.
Abdullateef O. Balogun, Amos O. Bajeh, Victor A. Orie and Ayisat W. Yusuf-Asaju [13]	2018	NASA MDP Repository, AR1, AR3, KC1, KC2, KC3, MC2, PC1, PC3, PC4, MC2, MW1	Bagged J48, Bagged KNN, Bagged MLP AUC metric, Voting	SMO, KNN, MLP, Decision Tree, Ensemble Methods	▪ For detecting several defects more unique performance measures should be considered while choosing the most appropriate classifier.
Shamsul Huda, Kevin Liu, Mohamed Abdelrazek, Amani Ibrahim Sultan Alyahya, Hmood Al-Dossari, And Shafiq Ahmad [14]	2018	PROMISE	RECALL, AUC	Ensemble of Oversampling techniques (Random Over Sampling, Majority Weighted Minority, Fuzzy Instance recovery using information Decomposition), Random Forest Base Classifier	• Adoption of significant software metrics is not being used while finding the defects in the software. • Method to be developed for handling variance problems present in the individual classifier.
Reza Mousavi, Mahdi Eftekhari, Farhad Rahdari [15]	2018	NASA	G-Mean, AUC	Over-Bagging, static and dynamic ensemble selection techniques are combined., 34 base classifiers, 6 combiners (VOTE, MAX, MIN, MEAN, MEDIAN, PRODUCT)	• Limited datasets are being tested
Thanh Tung Khuat, My Hanh Le [16]	2019	PROMISE (ANT 1.7, Camel 1.6, Ivy 2.0, Tomcat, Xalan 2.4,	Precision, Recall, F1-score measure	Support Vector Machine, Multilayer Perceptron, Bayesian Network, K-nearest Neighbor, Decision Tree J48	▪ Limited classifiers are used only under sampling strategies through several other sampling techniques are there.

		Synapse 1.2, Pol 2.0)			▪ Parameters optimization techniques were not there to further improve the accuracy.
N. Dhamayanthi (&) and B. Lavanya [17]	2019	NASA MDP program (PC2, PC3, CM1, KC3, MW1)	Stratified 10-Fold Cross Validation	For feature selection Principal component analysis is used and Random Forest, Adaboost, Bagging, and Regression Classification are the algorithms implemented	▪ Variance in-class problem was not handled ▪ On a small dataset, just a few machine learning techniques are tested. ▪ Noise filtering is also not taken care of.
Ahmed Iqbal, Shabib Aftab, Israr Ullah, Muhammad Salman Bashir, Muhammad Anwaar Saeed [18]	2019	NASA (datasets include: CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4 and PC5)	Precision, Recall, F-measure, Accuracy, MCC, and ROC	Boosting and Bagging (Random Forest) were employed as classifiers and Feature Selection (Chi-Square+ Filtering) are used.	▪ The class Imbalance problem is not being resolved ▪ Noise filtering and optimization are also not handled.
Ran Li, Lijuan Zhou, Shudong Zhang, Hui Liu, Xiangyang Huang, Zhong Sun [19]	2019	NASA MDP (CM1, JM1, KC1, KC2, PC1)	Precision, F-measure, Recall, AUC	Random Forest Algorithm in conjunction with oversampling SMOTE under-sampling Resample technique.	▪ The effect of only 5 ensemble algorithms is used so the next step in the research will be to increase the diversity of base classifiers to maximize the classification effects and performance of software defect prediction.
Tanujit Chakraborty and Ashish Kumar Chakraborty [20]	2020	Nasa SDP datasets from the PROMISE repository (10)	AUC, Recall, F-measure, Accuracy, Precision, Sensitivity	The Hellinger Distance is used as a criterion for splitting the tree which is based on Breiman's CART concept. As training methods, Random Forest and Naive Bayes with log filters were utilized.	▪ On the original dataset, data preprocessing techniques were not available. ▪ Cost-sensitive approaches were also not being considered ▪ An immediate extension suggested in this paper is to apply it to problems of imbalanced classification in the face of a conceptual shift, which is an important research field in industrial statistics.
Thanh Tung Khuat, My Hanh Le [21]	2020	NASA from Promise Repository	Accuracy, F-1 score, Recall, Precision	Using ensemble classifiers such as Logistic Regression, Support Vector Machine, Multilayer Perceptron, Nave Bayes, Bayesian networks, Decision Tree, and K-Nearest Neighbour along with sampling method	▪ Several other sampling methods are available that can be applied to binary classification models for improving accuracy. ▪ Dimension reduction and noise filtering too can be implemented further
Ruchika Malhotra, Juhi Jain [22]	2020	Synapse 1.0, Camel 1.4, and Ant 1.0 are three datasets chosen from open-source Apache that is written in Java.	Sensitivity, G-mean, Balance, AUC, Tenfold cross-validation, Friedman test G-Mean,	Boosting-based algorithms AdaBoost, AdaBoost.NC, AdaBoost.M1, AdaBoost.M2, RUSBoost, SMOTE, SMOTE Boost, DataBoost	▪ Problems like small disjuncts, cross-validation data distribution, and dataset shift are not handled which can further influence the performance of the classifier.

			Balance, and AUC		▪ By comparing various ensembles with different datasets, the results can be further generalized.
P. Suresh Kumar, H. S. Behera, Janmenjoy Nayak, Bighnaraj Naik [23]	2020	Jedit4.0, Ant1.7, and camel1.4 in the PROMISE repository.	Precision, AUC- ROC, F-Measure, TPR, TNR, FPR	Binary class Bagging with Decision tree as the base model	▪ For comparison, only a few machine learning methods are employed. ▪ The characteristics of product-based software are not examined. ▪ Finding the accuracy and number of faults is also missing.
Aftab Ali, Naveed Khan, Mamun Abu-Tair et.al. [24]	2021	PROMISE	AUC, Recall, Precision, F-measure, Mean Absolute Error, Root mean square error.	ANOVA F-Value, Metacost-based Domingo's Logistic Regression, Decision tree ensemble	▪ The paper focuses mainly on highly skewed data and prediction schemes that are cost-sensitive ignoring optimization techniques. ▪
Abdullateef O. Balogun, Shuib Basri [25]	2021	PROMISE, NASA, AEEEM, and Relink repositories	Accuracy, AUC, F-measure, Scott–Knott ESD statistical rank test.	Rank aggregation-based ensemble multi-filter feature selection (AREMFFS), Naïve Bayes, Decision Trees, Feature selection methods used are Chi-Square, Information Gain, Relief	▪ Computational time is not used for assessing the performance of the software bug detection model. ▪ The threshold value associated with the dataset being utilized is also ignored, which can have an impact on the filter feature selection efficiency.
Sweta Mehta, K. Sridhar Patnaik [26]	2021	NASA, PROMISE Repository Dataset	Accuracy, Precision, Recall	Stacking and XGBoost with a combination of Partial Least Square Regression and Recursive Feature Elimination, algorithms are further combined with cross-validation	▪ Implementation of Optimization techniques was missing which can further improve the accuracy of prediction. ▪ Finding the number of faults and reasons for their occurrence were also not present.
Abdullateef O. Balogun a, Kayode S. Adewole et.al. [27]	2021	UCI repositories (Phishing Websites)	AUC, Accuracy, False Positive Rate, MCC	Function Tree, Meta Learners Bagging Boosting, Rotation	▪ Class imbalance and High Dimensionality problems are not being handled which affects the accuracy of the model ▪ No feature selection and dimension reduction are applied at the data preprocessing stage.

4. Conclusion and Future Scope

This survey's objective is to learn about the most recent advances in artificial intelligence-based software fault prediction and digital technologies that utilize Ensemble Models of machine learning, especially the research on the global state during the period 2018-2021 focused upon. The authors have attempted to comprehend the trends, algorithms, and datasets utilized by researchers for the detection of software defects in various phases of software development to critically analyze the adapted approaches for the identification of gaps that may be taken up in the future for further enhancement and improvements. It can be summarised that current software defect prediction research focuses on the trends like estimation, association, classification, clustering, and dataset analysis. Both supervised and unsupervised methods are used for detecting defects in software. Although, academicians have provided a variety of Software defect prediction classifiers that use various Machine Learning methods yet the performance may require some improvement, and a fresh approach would be ideal. Researchers also suggested methods for combining different machine learning approaches, using bagging, boosting, and clustering algorithms, adding feature selection, and using parameter optimization to increase the accuracy of models while predicting software defects. But still, incomplete validation mechanisms might lead to unintentionally misleading results and overconfidence on the side of the researchers. The problem of class imbalance in datasets has still not been overcome by models suggested by various academicians. No study has employed the neighborhood-based under-sampling strategy to address the class imbalance problem for software defect prediction classifiers. Dealing with noisy data was the next issue that need to be resolved. In this case, there are irrelevant features in the dataset that are affecting the model's accuracy. Accuracy can improve if these redundant features are removed using various metaheuristic algorithms. To increase the effectiveness of the ensemble model, it is also important to explore the diversity of classifiers while developing them. The authors' objective is to create a hybrid machine learning model with improved performance metrics by resolving these problems.

References

[1] "The Standish Group Report" by open door technology is available from https://www.opendoorerp.com/the-standish-group-report-83-9-of-it-projects-partially-or-completely-fail.

[2] 8 Biggest IT disasters of 2021 by CIO available from https://www.cio.com/article/302010/8-biggest-it-disasters-of-2021.html.

[3] Abdullah Alsaeedi, Mohammad Zubair Khan (2019), "Software defect prediction using supervised machine learning and ensemble techniques: a comparative study", *Journal of Software Engineering and Applications*, 12(5), 85-100.

[4] Ning Li, Martin Shepperd, Yuchen Guo (2020), "A systematic review of unsupervised learning techniques for software defect prediction", Information and Software Technology, 122, 106287, 2020.

[5] Ensemble Learning by Data Science Blog downloaded from https://data-science-blog.com/blog/2017/12/03/ensemble-learning.

[6] Xiang Chen, Dun Zhang, Yingquan Zhao, Zhanqi Cui, Chao Ni (2019)," Software defect number prediction: Unsupervised vs supervised methods", Information and Software Technology, 106, 161-181.

[7] Wang Tao, Weihua Li, Haobin Shi, and Zun Liu (2011), "Software Defect Prediction Based on Classifiers Ensemble", *Journal of Information & Computational Science*, 8, 4241-4254.

[8] Abdullah Alsaeedi, Mohammad Zubair Khan (2019), "Software defect prediction using supervised machine learning and ensemble techniques: a comparative study. *Journal of Software Engineering and Applications*, *12*(5), 85-100.

[9] Machine Learning by Java Point downloaded from https://www.javatpoint.com/machine-learning-random-forest-algorithm.

[10] Boosting by IBM downloaded from https://www.ibm.com/cloud/learn/boosting.

[11] Zhou Xu, Li Li, Meng Yan, Jin Liu, Xiapu Luo, John Grundy, Yifeng Zhang, and Xiao Hong Zhang (2021), "A comprehensive comparative study of clustering-based unsupervised defect prediction models", *Journal of Systems and Software* 172: 110862.

[12] Shaojian Qiu, Lu Lu, Siyu Jiang, Yang Guo (2019)," An investigation of imbalanced ensemble learning methods for cross-project defect prediction", *International Journal of Pattern Recognition and Artificial Intelligence*, *33*(12), 1959037.

[13] Balogun Abdullateef Oluwagbemiga, Amos Orenyi Bajeh, Victor A. Orie and Ayisat W. Yusuf-Asaju (2018), "Software defect prediction using ensemble learning: an ANP based evaluation method", FUOYE J. Eng. Technol 3, no. 2: 50-55.

[14] Shamsul Huda, Kevin Liu, Mohamed Abdelrazek, Amani Ibrahim, Sultan Alyahya, Hmood Z. Al-Dossari, Shafiq Ahmad (2018)," An ensemble oversampling model for class imbalance problem in software defect prediction", *IEEE Access*, *6*, 24184-24195.

[15] Reza Mousavi, Mahdi Eftekhari, Farhad Rahdari (2018), "Omni-ensemble learning (OEL): utilizing over-bagging, static and dynamic ensemble selection approaches for software defect prediction", *International Journal on Artificial Intelligence Tools*, *27*(06), 1850024.

[16] Thanh Tung Khuat, My Hanh Le (2019), "Ensemble learning for software fault prediction problem with imbalanced data", *International Journal of Electrical and Computer Engineering, 9*(4), 3241.

[17] N. Dhamayanthi, B. Lavanya (2019), "Software defect prediction using principal component analysis and naïve Bayes algorithm", In *Proceedings of International Conference on Computational Intelligence and Data Engineering* (pp. 241-248), Springer, Singapore.

[18] Ahmed Iqbal, Shabib Aftab, Israr Ullah, Muhammad Salman Bashir, Muhammad Anwaar Saeed (2019), "A feature selection-based ensemble classification framework for software defect prediction", *International Journal of Modern Education and Computer Science*, 11(9), 54.

[19] Ran Li, Lijuan Zhou, Shudong Zhang, Hui Liu, Xianyang Huang, Zhong Sun (2019)," Software defect prediction based on ensemble learning", In *Proceedings of the 2019 2nd International conference on data science and information technology* (pp. 1-6).

[20] Tanujit Chakraborty, Ashish Kumar Chakraborty (2020)," Hellinger net: A hybrid imbalance learning model to improve software defect prediction", IEEE Transactions on Reliability, 70(2), 481-494.

[21] Thanh Tung Khuat, My Hanh Le (2020)," Evaluation of sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems", SN Computer Science, 1(2), 1-16.
[22] Ruchika Malhotra, Juhi Jain (2020)," Handling imbalanced data using ensemble learning in software defect prediction", In 2020 *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 300-304), IEEE.
[23] P. Suresh Kumar, H.S. Behera, Janmenjoy Nayak, Bighnaraj Naik (2021), "Bootstrap aggregation ensemble learning-based reliable approach for software defect prediction by using characterized code feature", Innovations in Systems and Software Engineering, 17(4), 355-379.
[24] Aftab Ali, Naveed Khan, Mamun Abu-Tair, Joost Noppen, Sally McClean, Ian McChesney (2021)," Discriminating features-based cost-sensitive approach for software defect prediction", Automated Software Engineering, 28(2), 1-18.
[25] Abdullateef O. Balogun, Shuib Basri, Luiz Fernando Capretz, Saipunidzam Mahamad, Abdullahi A. Imam, Malek A. Almomani, Ganesh Kumar(2021)," An adaptive rank aggregation-based ensemble multi-filter feature selection method in software defect prediction", Entropy, 23(10), 1274.
[26] Sweta Mehta, K. Sridhar Patnaik (2021)," Improved prediction of software defects using ensemble machine learning techniques", Neural Computing and Applications, 33(16), 10551-10562.
[27] Abdullateef O. Balogun, Kayode S. Adewole, Muiz O. Raheem, Oluwatobi N. Akande, Fatima E. Usman-Hamza, Madinat A. Mabayoje, Abimbola G. Akintola, Ayisat W. Asaju Gbolagade, Muhammed K. Jimoh, Adeyemo, Rasheed G. Jimoh, Victor E. Adeyemo (2021)," Improving the phishing website detection using empirical analysis of Function Tree and its variants", Heliyon, 7(7), e07437.