https://doi.org/10.33472/AFJBS.6.6.2024.478-487

African Journal of Biological Sciences

Research Paper Open Access

Adaboost Fuzzy SVM Based Software Defect Prediction

[1] Mrs. A. Priyadarshini, [2] Dr.V.Krishnapriya

[1] Research Scholar, Sri Ramakrishna College Arts &Science and Assistant Professor , PSGR Krishammal College for women, Coimbatore.

[2] Associate Professor & Head Department of Computer Science with Cognitive System, Sri Ramakrishna College Arts & Science, Coimbatore.

[1] ndpriya7@gmail.com

Article History
Volume 6,Issue 6, 2024
Received:07 Mar 2024
Accepted : 31 Mar 2024
doi: 10.33472/AFJBS.6.6.2024.478-487

Abstract

Software Defect Prediction (SDP) is a critical task in the software development process that forecasts which modules are more prone to errors and faults before the testing phase begins. This paper proposes a SMOTE algorithm to handle class balanced dataset issue. Adaboost Priority based Fuzzy SVM classification technique is proposed for classifying the balanced data sets. This proposed method reduces error rate compared with other existing machine learning and Priority based Fuzzy SVM methods. Experimental results showed that proposed scheme yields better accuracy than the existing techniques.

Keywords: Software Defect Prediction, Balanced Data set, Adaboost Classification

I. INTRODUCTION

Software fault prediction is the process of estimating errors in a software product during and after development, using previously defined metrics or historical defect data gathered from previous similar projects. Early in the development process, or even before starting a project, the ability to estimate software faults can be invaluable in minimising software development time and effort, where accurate SFP can reduce the efforts required to detect software errors throughout the software life cycle and minimise the number of modules developed in each activity.Defect prediction is a method of developing models that are used early in the process to detect defective systems such as units or classes. This can be accomplished by categorising the modules as defect-prone or not. To identify the classification module, various methods are used, the most common of which are support vector classifiers (SVC), random forests, and naive Bayes, decision trees (DT), and neural networks (NN). The detected defect prone modules are given high priority during the progress testing phases, while the non-defect prone

modules are examined as time and budget allow. The classification feature, known as the relationship between the attributes and the training dataset class label, is established on the classifier method and examined using formulae for target categorization. These rules will also be required in the future to define dataset class labels. Thus, the unknown datasets can be categorized using the classification patterns and a classifier.

As a result of massive deployment of software, defining software defects, finding the defect, and identifying it is a repetitive task for researchers. The primary goal of categorising the software dataset as a model for bug prediction into defective and non-defective datasets is to reduce the number of bugs in the dataset. According to this method, the input software dataset is given to the classifier where the user knows the actual class values.

The following steps is the way the paper is structured. In Section II, a detailed description of existing techniques in the field of Software Defect Prediction is furnished. The proposed framework for software prediction with various datasets is discussed in Section III. Section IV presents the experimental results. Finally, section V brings the paper to a conclusion.

II. RELATED WORK

This section provides a brief overview of existing techniques in the field of SDP. Several researchers have used ML techniques for SDP during the early stages of software development. To solve the Software Fault Prediction (SFP) problem, Kassaymeh S et al [1] proposed the Salp Swarm Algorithm (SSA) combined with a Back Propagation Neural Network (BPNN). The SFP problem is a well-known software engineering problem that is concerned with anticipating software defects that are likely to appear during or after a software project. In order to evaluate their method, they used six performance measures (AUC, confusion matrix, sensitivity, specificity, accuracy, and ER). One limitation identified in their research is the high computational cost of most data sets. Therefore, a new strategy to optimize the proposed algorithm in terms of computational cost can be developed in future.

Mangla M et al [2] used a sequential ensemble model to formalise the SFP method. The proposed model is tested using eight datasets from well-known repositories. The performance of their sequential ensemble model is evaluated using various error metrics, including average absolute error, average relative error, and prediction. Another error metric, root mean squared error (RMSE), is not used for performance analysis because it assigns larger weights to larger errors because it squares the errors before averaging out, and thus it is more suitable for applications focusing on large errors. As a result, the RMSE should be more useful when large errors are especially undesirable. The results of their model were encouraging, and they supported the use of ensemble modelling for SFP.

DePaaS—Defect Prediction as a Service—a cloud-based, multi-model SDP framework was proposed by Pandit M et al [3]. It is designed to be a global, unified platform that serves both researchers who create SDP models and software industry practitioners who use the defect prediction services provided by these SDP models. The author described the DePaaS usage context, five types of users, and five initial use cases, as well as a layered, modular architecture. It defined the structure and behaviour of architectural elements. It defined the structure and behaviour of architectural elements.

Odejide BJ et al [4] used sampling methods on software defect datasets to alleviate the latent class imbalance problem by balancing the number of minority and majority class instances present, resulting in new defect datasets that did not have a class imbalance problem. On defect datasets from NASA and PROMISE repositories, three data oversampling methods (SMOTE, ADASYN, and ROS) and two data under sampling methods (RUS and NM) are used, while DT and RF classifiers are used on the original and newly developed software defect datasets. The author demonstrated that the data sampling methods investigated can overcome the class imbalance problem in SDP datasets. Furthermore, in the majority of cases, the data sampling methods improved the prediction performances of the tested prediction models. In terms of data sampling effectiveness, the examined oversampling approaches had a greater (positive) influence on the prediction models than their under sampling counterparts.

Balogun AO et al [5] addressed the SDP concepts and the class imbalance problem as described in order to develop a successful SDP model. On software defect datasets, data sampling methods are used to alleviate the latent class imbalance problem by levelling the number of minority and majority class instances observed, resulting in new defect datasets with no class imbalance problem. On defect datasets from the NASA repository, three data oversampling methods (SMOTE, ADASYN, and ROS) and two data under sampling methods (RUS and NM) are used, while ensemble (Bagging and Boosting) NB and DT classifiers are used on the original and newly developed software defect datasets.

For defect prediction, Mohammad UG et al [6] used machine learning techniques such as RF and SVM, as well as ensemble classifiers such as bagging, Adaboost, voting, and stacking. To evaluate the performance of an optimization model, key parameters such as precision, recall, and f1-measure are used. After addressing the imbalanced dataset issue, the proposed model improves the performance of all algorithms.

Wahono RS et al [7] presented a framework for comparing the performance of classification algorithms in the prediction of software defects. The framework is made up of nine NASA MDP datasets, ten classification algorithms, a ten fold cross validation model, and an AUC accuracy indicator. Friedman and Nemenyi are used to test the significance of model AUC differences. The experimental results show that the LR outperforms the others in the majority of NASA MDP datasets. NB, NN, SVM, and k* all perform well, with no statistically significant difference between them. Decision tree-based classifiers, as well as LDA and k-NN, tend to underperform.

Tsunoda M et al [8] discussed how feature reduction techniques can improve the predictive accuracy of software defect prediction models. They used the bandit algorithm (BA) to select a suitable feature reduction technique for defect prediction in this work. The Bandit Algorithm dynamically chooses the best technique from among candidates based on a comparison of test and prediction results on tested modules. As a result, it is expected that BA will prevent accuracy degradation. BANP had nearly the same or higher accuracy than existing approaches. That is, BANP could reduce the effort required to evaluate reduction techniques while avoiding the degradation of prediction accuracy. Their approaches suggested here can assist in selecting a suitable feature reduction technique that can improve the prediction model's overall accuracy.

Feng S et al [9] discovered in SDP that defect-containing datasets are normally imbalanced, a problem known as the class imbalance problem. Oversampling techniques are

commonly used to solve the problem. To address these issues, the Complexity-based Oversampling Technique (COSTE) was proposed as a novel oversampling technique. COSTE uses the complexity of instances rather than the distance between them to aid in the selection of those that will be used to generate synthetic instances, etc.

Tang S et al [10] proposed the transfer-learning algorithm TSboostDF for solving the CPDP problem. To overcome the shortcomings of the traditional CPDP algorithms, TSboostDF combines the BLS sampling method, which is based on sample weight, with the transfer-learning method. Their algorithm outperformed other transfer-learning-based CPDP algorithms in terms of performance. The effects of multi-source transfer learning on CPDP merit further research by integrating information from multiple source projects for knowledge transfer. This strategy will assist classifiers in improving their performance on CPDP problems.

Pandey SK et al [11] ran 864 experiments across three public datasets, analysing the noise endure for well-known SDP models. They had manually inserted noise ranging from 0% to 80%. They used four baseline SDP methods and trained them on noisy datasets. To avoid the problem of class imbalance, they used random sampling. The author proposed a method that can tolerate maximum noise while still outperforming baseline methods, and he compared the performance without using sampling methods. They discovered that the proposed approach outperforms baseline technologies with noisy instances and imbalanced data.

The preceding discussion demonstrated various methodologies used for defect prediction in the balanced dataset. Adaboost priority-based fuzzy SVM is used in our proposed work to predict defects in various data sets. The section that follows elaborates on the proposed methodology and the results obtained for performance measures.

III. METHODOLOGY

Software defect prediction (SDP) is a critical tool for assessing software quality and lowering development costs. Data collected during the software lifecycle can be used to forecast software defects. Many SDP models have been proposed recently; however, their performance was not always ideal. In this study, defected data sets are classified as defective or non-defective using Adaboost Priority based Fuzzy SVM.

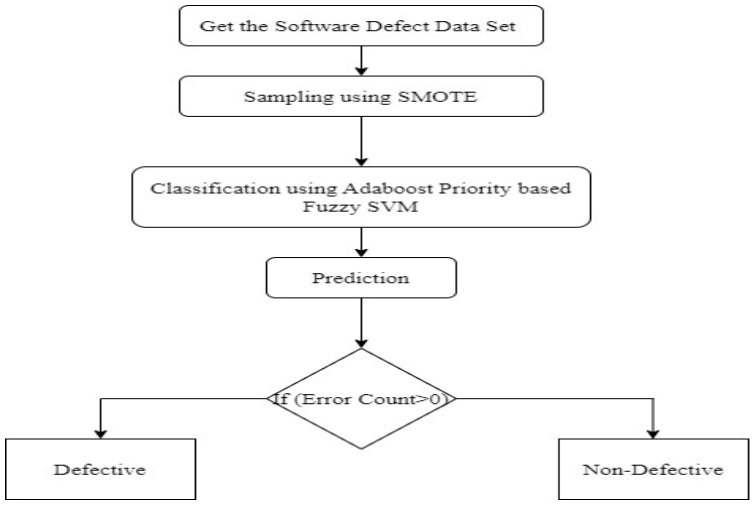

Fig 3.1 Flow Chart for Proposed Methodology

Fig 3.1 shows the flow of the proposed framework. It depicts that defected data set is taken as input. The data set having defections are predicted by Adaboost priority based fuzzy SVM technique. If the error count is greater than zero the data set is defective otherwise non defective.

A. Data Set Pre Processing

The acquired data sets were pre-processed to prepare them for subsequent machine learning approaches.

SMOTE (Synthetic Minority Over-sampling Technique) is a method of sampling the minority population by generating data points synthetically. The number of nearest neighbours chosen at random corresponds to the amount of sampling required. To begin, the difference between the sample under consideration and its corresponding nearest neighbours is calculated, multiplied by a random number between 0 and 1, and finally added to the original vector under consideration. It is important to note, however, that SMOTE cannot be applied to the entire data set and then split into testing and training sets.

If the data is first oversampled and then split into test and train sets, the results will be misleading because there is a high chance that the same data will be present in both sets. To avoid this, the data is first split into test and train sets, and the SMOTE is applied over the training data set for proper testing set validation.

B. Adaboost Priority Based Fuzzy SVM

Adaboost is a popular boosting algorithm that gradually increases the weights of the classifier's classification error weights. Create a new classifier in each iteration to overcome the failure of the old classifier, and then link the newly created classifier to the voting process. As a result of the Adaboost essence, the weak classifier is promoted to the strong classifier, which is an adaptive lifting technique. As a result, as the number of training data increases, so does the classification error rate. The Adaboost algorithm employs the following steps. Takes the training dataset and all training samples to learn the first weak learning classifier, and also provides the maximum number of iterations (M). The incorrect classification of sample and other data is combined to represent the new training dataset, while the sample weight is adjusted. Repeat these process M times. The new training data samples are generated for the next iteration learning classifier, which is based on a new weight, and finally, the strong classifier with improved classification effect is generated.

C. Performance Evaluation

One of the most important basic measures for evaluating the effectiveness of predictive models is classification accuracy, also known as the right classification rate. It is used to compute the proportion of correctly classified cases compared to total occurrences.

Precision is another metric that is calculated by dividing the number of instances correctly classified as faulty (TP) by the total number of instances classified as defective (TP + FP). Furthermore, recall quantifies the proportion of defective cases correctly classified (TP) to the total number of faulty instances (TP + FN). The F-score is a harmonic mean of accuracy

and recall that has been used in numerous studies. By balancing TPR and FPR, ROC-AUC calculates the area under the receiver operating characteristic (ROC) curve.

$$\text{Accuracy} = TP + TN / TP + TN + FP + FN \quad (1)$$

$$\text{Precision} = TP / TP + FP \quad (2)$$

$$\text{Recall} = TP / TP + FN \quad (3)$$

$$\text{F-Score} = 2 * \text{Precision} * \text{Recall} / \text{Precision} + \text{Recall} \quad (4)$$

G-measure is another measure used in software defect prediction. It is defined as a harmonic mean of recall and specificity. Probability of false alarm (PF) is the ratio of clean instances wrongly classified as defective (FP) among the total clean instances (FP + TN).

IV. RESULT AND DISCUSSION

The following table shows the accuracy for the machine learning algorithm used in the dataset.

TABLE I ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – CM1

Algorithm name	Accuracy
SVM	95
SMOTE SVM	96
SMOTE Fuzzy SVM	97
SMOTE Adoptive Fuzzy SVM	98.2

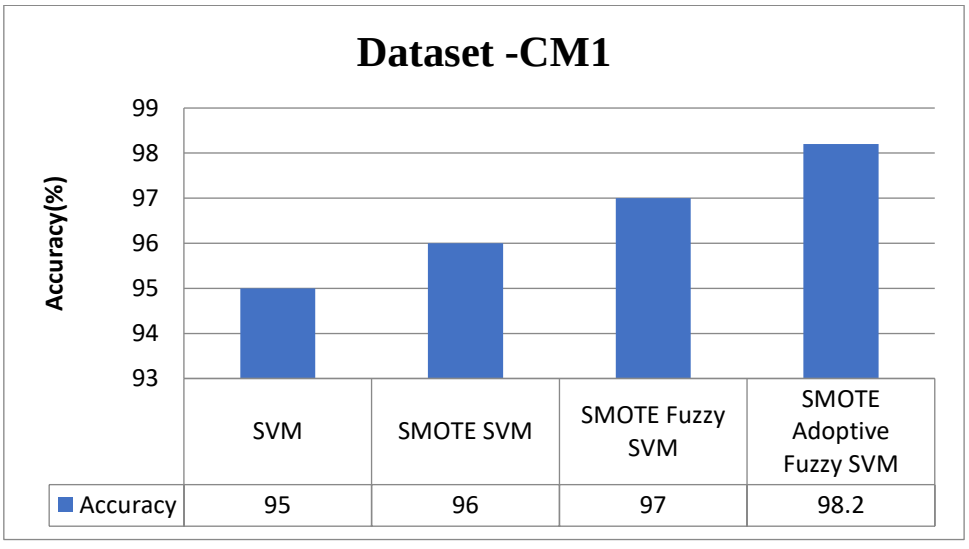

Fig 4.1 Accuracy for the used ML algorithms in the dataset – CM1

The following table shows the accuracy values obtained for various machine learning algorithms used and for the dataset KC1.

TABLE II ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – KC1

Algorithm name	Accuracy
SVM	91
SMOTE SVM	93.5
SMOTE Fuzzy SVM	95
SMOTE Adoptive Fuzzy SVM	97

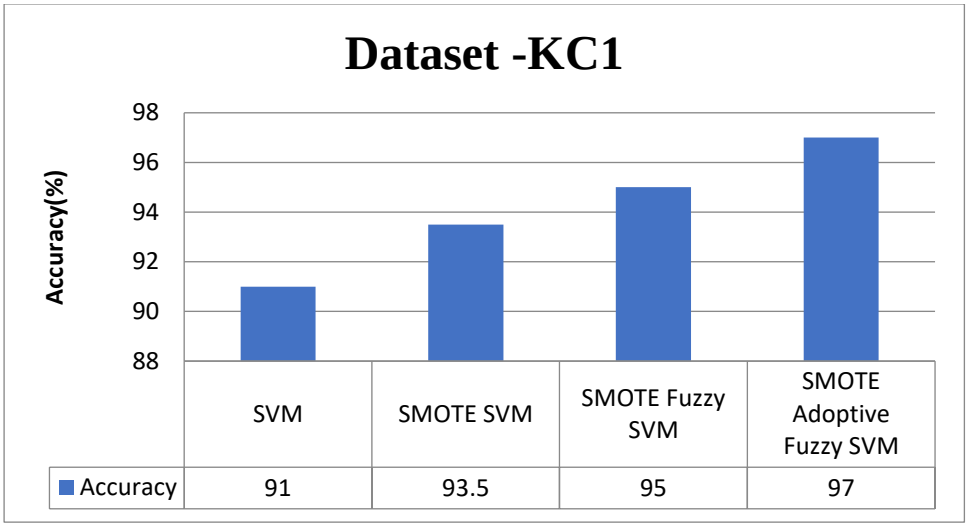

Fig 4.2 Accuracy for the used ML algorithms in the dataset – KC1

TABLE III ACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – KC2

Algorithm name	Accuracy
SVM	90
SMOTE SVM	92
SMOTE Fuzzy SVM	94.2
SMOTE Adoptive Fuzzy SVM	96

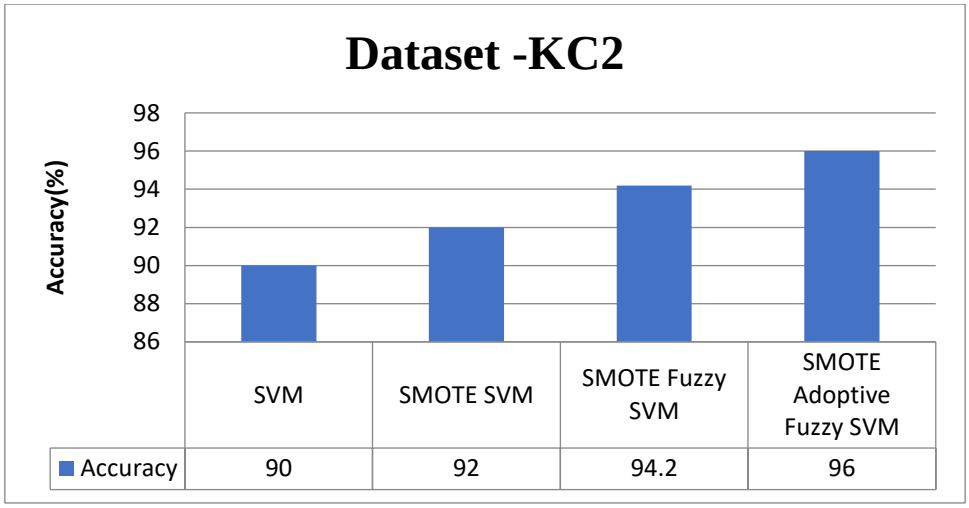

Fig 4.3 Accuracy for the used ML algorithms in the dataset – KC2

TABLE IVACCURACY FOR THE USED ML ALGORITHMS IN THE DATASET – PC1

Algorithm name	Accuracy
SVM	94.8
SMOTE SVM	95.6
SMOTE Fuzzy SVM	96.2
SMOTE Adoptive Fuzzy SVM	98

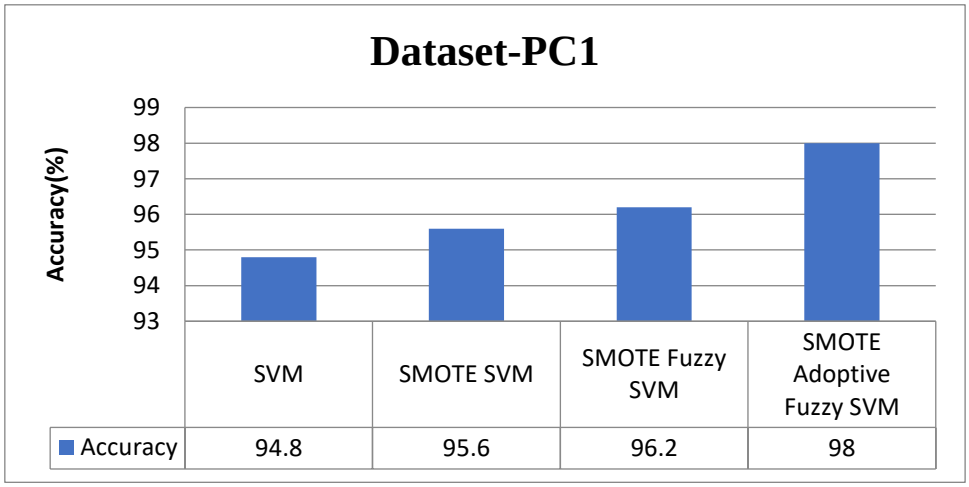

Fig 4.4 Accuracy for the used ML algorithms in the dataset – PC1

From the above graph, proposed SMOTE adaptive fuzzy SVM algorithm yields 98% accuracy for the dataset we used PC1.

V. CONCLUSION

Software defect prediction is a technique that uses data to create a prediction model that predicts future software faults. Several approaches have been proposed that make use of various datasets, metrics, and performance measures. This paper investigated the use of machine learning algorithms in the prediction of software bugs. The following machine learning techniques were used: NB, Random Forest, and Priority based Fuzzy SVM. Three real testing/debugging datasets are used in the evaluation process. The accuracy, precision, recall, F-measure, and RMSE measures are used to collect experimental results. The results show that ML techniques are effective methods for predicting future software bugs. The comparison results revealed that the proposed classifier outperformed the others.

References:

[1]. Kassaymeh S, Abdullah S, Al-Betar MA, Alweshah M. Salp swarm optimizer for modeling the software fault prediction problem. Journal of King Saud University-Computer and Information Sciences. 2022 Jun 1;34(6):3365-78.

[2]. Mangla M, Sharma N, Mohanty SN. A sequential ensemble model for software fault prediction. Innovations in Systems and Software Engineering. 2022 Jun;18(2):301-8.

[3]. Pandit M, Gupta D, Anand D, Goyal N, Aljahdali HM, Mansilla AO, Kadry S, Kumar A. Towards design and feasibility analysis of DePaaS: AI based global unified software defect prediction framework. Applied Sciences. 2022 Jan 4;12(1):493.

[4]. Odejide BJ, Bajeh AO, Balogun AO, Alanamu ZO, Adewole KS, Akintola AG, Salihu SA, Usman-Hamza FE, Mojeed HA. An Empirical Study on Data Sampling Methods in Addressing Class Imbalance Problem in Software Defect Prediction. InComputer Science On-line Conference 2022 (pp. 594-610). Springer, Cham.

[5]. Balogun AO, Odejide BJ, Bajeh AO, Alanamu ZO, Usman-Hamza FE, Adeleke HO, Mabayoje MA, Yusuff SR. Empirical Analysis of Data Sampling-Based Ensemble Methods in Software Defect Prediction. InInternational Conference on Computational Science and Its Applications 2022 (pp. 363-379). Springer, Cham.

[6]. Mohammad UG, Imtiaz S, Shakya M, Almadhor A, Anwar F. An Optimized Feature Selection Method Using Ensemble Classifiers in Software Defect Prediction for Healthcare Systems. Wireless Communications and Mobile Computing. 2022 Jun 27;2022.

[7]. Wahono RS, Herman NS, Ahmad S. A comparison framework of classification models for software defect prediction. Advanced Science Letters. 2014 Oct 1;20(10-11):1945-50.

[8]. Tsunoda M, Monden A, Toda K, Tahir A, Bennin KE, Nakasai K, Nagura M, Matsumoto K. Using Bandit Algorithms for Selecting Feature Reduction Techniques in Software Defect Prediction. In2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR) 2022 May 23 (pp. 670-681). IEEE.

[9]. Feng S, Keung J, Yu X, Xiao Y, Bennin KE, Kabir MA, Zhang M. COSTE: Complexity-based Over Sampling TEchnique to alleviate the class imbalance problem in software defect prediction. Information and Software Technology. 2021 Jan 1;129:106432.

[10]. Tang S, Huang S, Zheng C, Liu E, Zong C, Ding Y. A novel cross-project software defect prediction algorithm based on transfer learning. Tsinghua Science and Technology. 2021 Aug 17;27(1):41-57.

[11]. Pandey SK, Mishra RB, Tripathi AK. Machine learning based methods for software fault prediction: A survey. Expert Systems with Applications. 2021 Jun 15;172:114595.

[12]. Li, Z., Jing, X.Y. and Zhu, X., 2018. Progress on approaches to software defect prediction. *Iet Software*, *12*(3), pp.161-175.

[13]. Qiao, L., Li, X., Umer, Q. and Guo, P., 2020. Deep learning based software defect prediction. *Neurocomputing*, *385*, pp.100-110.

[14]. Cai, X., Niu, Y., Geng, S., Zhang, J., Cui, Z., Li, J. and Chen, J., 2020. An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. *Concurrency and Computation: Practice and Experience*, *32*(5), p.e5478.

[15]. Li, N., Shepperd, M. and Guo, Y., 2020. A systematic review of unsupervised learning techniques for software defect prediction. Information and Software Technology, 122, p.106287.

[16]. Esteves, G., Figueiredo, E., Veloso, A., Viggiato, M. and Ziviani, N., 2020. Understanding machine learning software defect predictions. *Automated Software Engineering*, *27*(3), pp.369-392.