The Impact of Software Fault Prediction in Real-World Application: An Automated Approach for Software Engineering

Conference Paper · January 2020

DOI: 10.1145/3379247.3379278

CITATIONS
17

READS
2,256

5 authors, including:

Md. Razu Ahmed
Softcell Solutions Ltd
13 PUBLICATIONS **128** CITATIONS
SEE PROFILE

Md. Asraf Ali
American International University - Bangladesh (AIUB)
57 PUBLICATIONS **460** CITATIONS
SEE PROFILE

Nasim Ahmed
Massey University
83 PUBLICATIONS **505** CITATIONS
SEE PROFILE

Md Fahad Bin Zamal
Daffodil International University
6 PUBLICATIONS **21** CITATIONS
SEE PROFILE

The Impact of Software Fault Prediction in Real-World Application: An Automated Approach for Software Engineering

Md. Razu Ahmed
Department of Software Engineering
Daffodil International University
Dhaka-1207, Bangladesh
Cell: +8801787844301
razu35-1072@diu.edu.bd

Md. Asraf Ali
Department of Software Engineering
Daffodil International University
Dhaka-1207, Bangladesh
asraf.swe@diu.edu.bd

Nasim Ahmed
School of Natural and Computational Sciences, Massey University
Albany 0632, Auckland, New Zealand
nasim751@yahoo.com

Md. Fahad Bin Zamal
Department of Software Engineering
Daffodil International University
Dhaka-1207, Bangladesh
fahad.swe@diu.edu.bd

F.M. Javed Mehedi Shamrat
Department of Software Engineering
Daffodil International University
Dhaka-1207, Bangladesh
javedmehedicom@gmail.com

ABSTRACT

Software fault prediction and proneness has long been considered as a critical issue for the tech industry and software professionals. In the traditional techniques, it requires previous experience of faults or a faulty module while detecting the software faults inside an application. An automated software fault recovery models enable the software to significantly predict and recover software faults using machine learning techniques. Such ability of the feature makes the software to run more effectively and reduce the faults, time and cost. In this paper, we proposed a software defect predictive development models using machine learning techniques that can enable the software to continue its projected task. Moreover, we used different prominent evaluation benchmark to evaluate the model's performance such as ten-fold cross-validation techniques, precision, recall, specificity, f 1 measure, and accuracy. This study reports a significant classification performance of 98-100% using SVM on three defect datasets in terms of f1 measure. However, software practitioners and researchers can attain independent understanding from this study while selecting automated task for their intended application.

CCS Concepts

Computing methodologies→ Machine learning algorithms

Keywords

Software engineering; Software fault; Machine learning; Defect prediction.

1. INTRODUCTION

The vast area of software development and different applications makes it challenging for software developers and also customers to observe, maintains and manage software applications. Moreover, the fourth industrial revolution employs artificial intelligence by software industry is one of the promising sectors of modern times that observes a constant transformation in its practices because of the automating large quantities of software technologies [1]. The size and complexity of current software is increasing day by day. As a result, software engineers are struggling continuously with faults from the beginning of the development phase.

The classification of the software faults is important in real-time, otherwise, the effort and cost of finding defects hiding in an application are also rising fast. This inspires the development of automated fault prediction models for software fault prediction that can forecast the software defects. If software defects are identified before the release of software that can help the developer to allocate and fix those defect modules easily.

Software fault prediction through machine learning techniques are being the most prominent use case among the researchers and the software community [2]. State of the art machine learning algorithms have been applied to find the defect modules from the software applications for research and make effective solutions for consumers [3]. In this study, we have used the six most popular of the machine learning classifiers which are suggested in the most recent systematic literature study [4]. All selected classification techniques are applied over different real application datasets related to the software fault prediction of the applications. However, we have to consider particular features in terms of their quality of data but that cannot be validated in terms of correctness. Therefore, the state of the art [5] machine learning approaches have been applied to the fault datasets to enhance the prediction by reducing unnecessary features through several feature selection techniques and imbalance to balancing data methods.

Many of the studies examined fault recovery methods into software with their focus on combining the automated recovery model inside the application [6][7][8]. The goal of this study is examined with six classifiers' performance and recommends an automated approach to solve software faults inside a software. We used three data sets (i.e. JM1, CM1, PC1) from PROMISE

ICCDE '20, January 4–6, 2020, Sanya, China

ACM ISBN 978-1-4503-7673-0/20/01…$15.00
https://doi.org/10.1145/3379247.3379278

Software Engineering Repository and 22 attributes such as McCabe, Halstead, and some other metrics including defect information [9].

We preprocessed our selected data for allocating correlated columns and we also examined data imbalance issues by using different computational techniques such as PCA, Resample and SMOTE [10]. Hence, this paper presents a comparative analysis of six machine learning techniques for the software fault prediction inside a software.

The remainder of the paper is organized as follows as the data sets, classification techniques, performance measure, and experimental Setup are presented in section 2. The analysis results are described in Section 4. And finally, conclusions and thoughts for future work are illustrated in Section 5.

2. RELATED WORK

Software fault localization and maintainability are defined as a software system or modules can be adapted to correct faults, improve performance or software and system testing, software development techniques or modify to a changed platform [11]. A software defect predictive model enables organizations to help to reduce the maintenance effort, time and cost overall on a software project [12] [13]. To ensure the quality of good software must be reliable, and it can occur a smaller number of failures during the software run time [14]. Hence, classification of defects on software modules has a large impact during software development process. But the real scenario would become hard, because when developer changes his program inside an application and it is related to other modules including failed to updated version of this application. Therefore, it is very possible case for the software become faulty and not stable [15].

The number of literatures in software fault proneness is increasing day by day for demand for automated services [16]. Ruchita Malhutra [17] presented a comparative study between ANN, SVM, DT, CCN, GMDH and GEP to predict software fault modules. The author have taken of 3 NASA project data from PROMISE Software Engineering Repository. Form the performance analysis of this study, DT has achieved the highest accuracy than other classifiers and the study used 21 CK metrics, McCabe metrics, and HalStead metrics. Zhou Xu et al. [18] introduced a defect prediction framework which is called KPWE. And it combines two approaches, i.e. Kernel Principal Component Analysis (KPCA) and Weighted Extreme Learning Machine (WELM). The contribution of this study is used to 44 software projects data and that indicates KPWE is an excellent technique to the baseline methods in utmost cases. Moeyersoms et al. [19] has presented a comparative study on RF, SVM, C4.5 (DT) and Regression Tree to predict the software defect modules. The experimental result shows that RF has the highest accuracy among classifiers. Qiao Yu et al. [20] developed a new feature subset selection and feature classifying techniques to investigate the effectiveness of feature selection for cross-project defect prediction (CPDP). Their experiment shows that the CPDP feature selection approaches can improve the performance of software defect analysis. Hotzkow et al. [21] described that the application can automatically learning user expectations from the semantic contexts over multiple applications.

Many researchers are enthusiastic about their works in the discipline of automated fault tolerance inside a software system [22]. Montani et al. [23] described Case-based reasoning (CBR) methods that provided abilities to the software for fault prediction in software systems appears to be very suitable for distributed software applications. A study [24] mentioned that the approaches in the article" A New Way to Find Bugs in Self-Driving AI Could Save Lives". Two studies stated that the machine learning-based approaches are better than statistical-based models for software fault prediction [25][26]. Therefore, it is very important to using open-source datasets because of their reliability, consistency, and verifiability [27] [28].

3. MATERIALS AND METHODS

3.1 Data Collection

In this experiment, we have used 3 open source publicly available data from PROMISE Software Engineering Database. These datasets Tim Menzies et al. have been used in their research paper [29]. In another study, Jureczko et al. [30] have been assembled a software fault prediction model to predict the software defects using machine learning algorithms. They have discussed in their paper about 8 projects (PROMISE Repository) data and by taking 19 CK metrics and McCabe metrics for constructed a predictive model. In our study, we have used 22 attributes for building our automated fault predict model. Table 1 shows 22 different attributes from software defect datasets including 21 independent metrics and one is outcome information. i.e. which is faulty and no-fault.

Table 1. List of the metrics

No	Metrics name	Type
1	Line of code	McCabe
2	Cyclomatic complexity	McCabe
3	Essential complexity	McCabe
4	Design complexity	McCabe
5	Halstead operators and operands	Halstead
6	Halstead volume	Halstead
7	Halstead program length	Halstead
8	Halstead difficulty	Halstead
9	Halstead intelligence	Halstead
10	Halstead effort	Halstead
11	Halstead time estimator	Halstead
12	Halstead line count	Halstead
13	Halstead comments count	Halstead
14	Halstead blank line count	Halstead
15	IO code and comments	Miscellaneous
16	Unique operators	Miscellaneous
17	Unique operands	Miscellaneous
18	Total operators	Miscellaneous
19	Total operands	Miscellaneous
20	Branch count	Miscellaneous
21	b: numeric	Halstead
22	Defects	False or true

The present study used JM1, CM1, PC1 datasets which were implemented in C language. Table 2 depicted details about detail of all datasets with their features.

Table 2: Details about datasets

No	Dataset	Missing	Instance	Class distribution

		attribute		True	False
1	JM1	None	10885	8779 (80.65%)	2106 (19.35%)
2	CM1	None	498	49 (9.83%)	449 (90.16%)
3	PC1	None	1109	1032 (93.05%)	77 (6.94%)

3.2 Classification Techniques

Machine learning algorithm has been creating a significant role in software engineering fields. In recent years, machine learning techniques are one of the most operational techniques what are gained significantly high performance in real-world problems for the research and technical community. Harshita et al. [31] discussed in their review, there are common use of machine learning techniques for constructing software fault prediction models such as fuzzy logic-based software defect prediction, Naïve Bayes (NB), neural network (NN), random forest (RF), support vector machine (SVM), P-SVM, k-nearest neighbors (KNN), etc. Ruchita Malhotra [3] described in her systematic mapping study, the top five machine learning techniques were used to software defect analysis such as DT (46%), NB (74%), MLP in NN (85%), RF (59%), SVM (27.7%), etc.

In this study, 6 machine learning (ML) techniques have been considered to construct the defect model: Decision Tree (DT), k-nearest neighbors (KNN), Logistics Regression (LR), Naïve Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM).

3.3 Performance Measurement

Once the predictive model has been built, it can be applied to perform a test to predict the fault modules inside the software fault datasets. In this work, we examined the ML prediction models, utilizing six classification algorithms, based on different statistical techniques [32] such as confusion matrix (True Positive = TP, True Negative = TN, False Positive = FP, False Negative = FN), recall, precision, F1 measure, etc. Table 3 shows a quality measure of predictive model based on confusion matrix as below [33],

Table 3. Performance measurement criteria

Metrics	Mathematical formula
Accuracy	$\frac{(TP+TN)}{(TP+FP+TN+FN)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall = TPR	$\frac{TP}{(TP+FN)}$
F1 measure	$\frac{2*(Recall*Precision)}{(Recall+Precision)}$
Specificity = TNR	$\frac{TN}{(TN+FP)}$

3.4 Experimental Setup

This section represents the proposed software defect predictive development (SDPD) model and procedure of the experiment. The proposed SDPD model is developed during the initial stage of the software development life cycle. This model suggests that the integrated software development process and its uses of the implementation as an input when the requirement analysis given into the predictive model for software development. In SDPD development, the high-level diagram and detail level diagram are mandatory to build a scalable SDPD model. The main phase of the SDPD development is physical design and testing phase, analyzes defect modules to provide knowledge-based automated fault recovery of the software systems. And it is lead to predict defects and contain all the required data about faulty modules of the application. Figure 1. SDPD shown the main components of proposed SDPD approach.

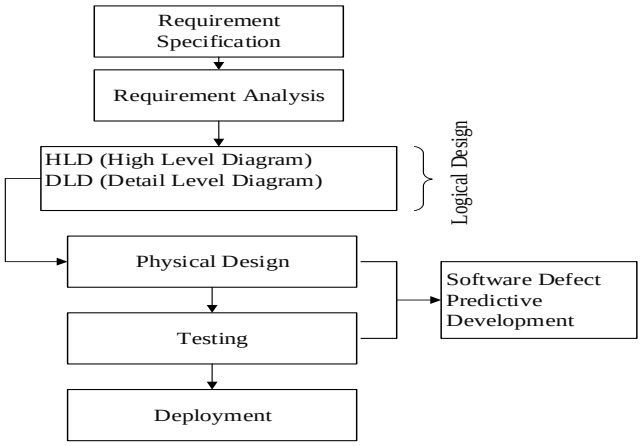

Figure 1. Proposed Software Defect Predictive Development Model

In this experiment, 3 defects datasets have been considered to build the software defect model. Then, we performed data preprocessing approach on defect datasets [34], such as covariance analysis to find high correlation inside data, to find the missing value, feature extraction, min-max normalization, resample, PCA, etc. Details workflow of our defect model for SDPD based application is presented in Figure 2.

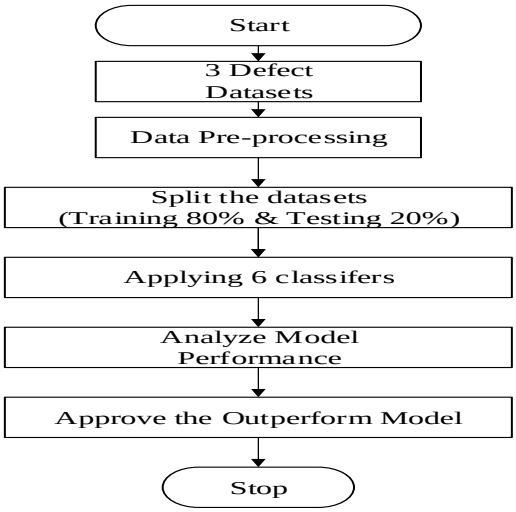

Figure 2. Workflow of the defect model

4. Results and Discussion

In this study, we focused on automated fault recovery inside software through a predictive model, besides we also observed

both defective and not faulty classes. Especially, defective modules are very crucial than not faulty modules. In our experiment, we used 10-fold cross-validation technique to evaluate the performance of six classification techniques. To determine the parameters for the software defect model, we used the different data preprocessing methods that have been increased the accuracy and consistency of the classification model. Table 4 shows the performance evaluation of six supervised classification techniques for software fault prediction. With respect to the precision: DT and SVM achieved the highest performance (i.e. 100%) on JM1 datasets; DT, NB, SVM, and RF achieved the best performance on CM1 datasets, (it's respectively 100%); DT, SVM, and RF obtained the highest performance (i.e. 97%) on PC1 datasets. Relatively, all of the classifiers have shown good performance in terms of precision. However, considering the recall of the analysis, SVM and RF achieved the highest performance on JM1 datasets; LR and NB attained the lowest performance on CM1 and PC1 datasets. Not that all of the classifiers achieved very similar scores in terms of recall. Another measure for classification is F1 measure. With respect to F1 measure: SVM achieved the highest value (i.e. 100%) on JM1 datasets and NB obtained the lowest score (i.e. 93%). By Looking CM1 datasets, we can monitor that the f1 scores are mostly similar (i.e. NB, DT, SVM, RF = 100% and KNN = 97%, LR = 95%). Moreover, RF achieved the best score (i.e. 99%) and KNN performed lowest (86%) on PC1 datasets. In addition, all of the classifiers have achieved utmost performance on JM1, CM1, and PC1 datasets, in terms of accuracy. This indicates that all of the classifiers are very effective in their classification performance to predict software defect modules.

Table 4. Classification performance of ml techniques

Algorithms	Performance Measurement	Datasets		
		JM1	CM1	PC1
DT	Precision	1.0	1.0	.97
	Recall	0.99	1.0	1.0
	Accuracy	0.99	1.0	.99
	F1	0.99	1.0	.98
NB	Precision	0.94	1.0	.91
	Recall	0.93	1.0	.85
	Accuracy	0.98	1.0	.98
	F1	0.93	1.0	.87
SVM	Precision	1.0	1.0	.97
	Recall	1.0	1.0	1.0
	Accuracy	.99	1.0	.99
	F1	1.0	1.0	.98
RF	Precision	.99	1.0	.97
	Recall	1.0	1.0	.97
	Accuracy	0.99	1.0	.99
	F1	0.99	1.0	.97
KNN	Precision	0.95	.95	.86
	Recall	0.97	1.0	.88
	Accuracy	0.99	.99	.95
	F1	0.95	.97	.86
LR	Precision	0.94	.95	.94
	Recall	0.96	.95	.92
	Accuracy	0.98	.98	.97
	F1	0.94	.95	.92

Figure 3. Shows the negative predictive value (NPV) and false positive rate (FPR) based on three software defect datasets.

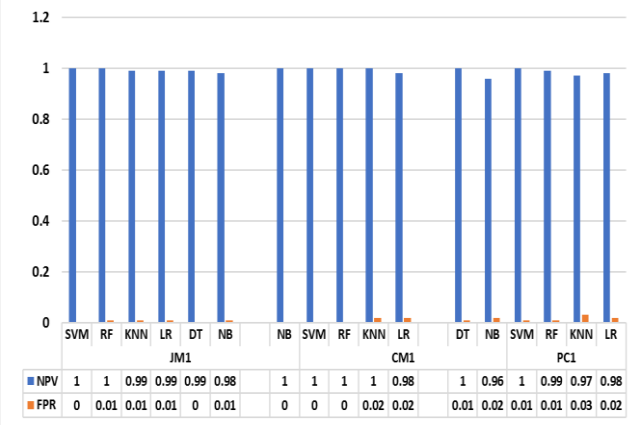

Figure 3. NPV and FPR for six ML classifiers

5. CONCLUSION

In this paper, we proposed an automated software engineering approaches for defect prediction model development (SDPD) on software development life cycle. After that, the main objective of our study was to evaluate the abilities of six supervised based the machine learning classifications techniques to predict the software defect modules using 3 NASA datasets. The results (i.e. accuracy: 98-100%) of the experiment with different attributes showed the capability and efficiency of the SDPD model to identify the fault and improve software quality. In addition, this SDPD model can be able to early detection of software faults by collecting real-time software development data from the target applications. The proposed approach can be used for software fault recovery inside a system and enhanced by applying machine learning techniques to construct SDPD more effective in software fault retrieval.

For future work, we will implement more classification algorithms, such as hybrid or ensemble model to verify the performance of the software fault prediction.

6. ACKNOWLEDGMENTS

The authors are grateful to all the researchers in this research study.

7. REFERENCES

[1] H. B. Bolat, G. T. Temur, and IGI Global, *Agile approaches for successfully managing and executing projects in the fourth industrial revolution*.

[2] G. P. Bhandari and R. Gupta, "Machine learning based software fault prediction utilizing source code metrics," in

2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018, pp. 40–45.

[3] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Appl. Soft Comput., vol. 27, pp. 504–518, Feb. 2015.

[4] L. Son et al., "Empirical Study of Software Defect Prediction: A Systematic Mapping," Symmetry (Basel)., vol. 11, no. 2, p. 212, Feb. 2019.

[5] C. W. Yohannese and T. Li, "A Combined-Learning Based Framework for Improved Software Fault Prediction," *Int. J. Comput. Intell. Syst.*, vol. 10, no. 1, p. 647, Dec. 2017.

[6] A. Hudaib *et al.*, "ADTEM-Architecture Design Testability Evaluation Model to Assess Software Architecture Based on Testability Metrics," *J. Softw. Eng. Appl.*, vol. 08, no. 04, pp. 201–210, Apr. 2015.

[7] S. Elmidaoui, L. Cheikhi, and A. Idri, "Towards a Taxonomy of Software Maintainability Predictors," Springer, Cham, 2019, pp. 823–832.

[8] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 367–377.

[9] "PROMISE DATASETS PAGE." [Online]. Available: http://promise.site.uottawa.ca/SERepository/datasets-page.html. [Accessed: 01-Jul-2019].

[10] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," *Neurocomputing*, vol. 343, pp. 120–140, May 2019.

[11] *610.12-1990 IEEE Standard Glossary of Software Engineering Terminology*. .

[12] P. Oman and J. Hagemeister, "Construction and testing of polynomials predicting software maintainability," *J. Syst. Softw.*, vol. 24, no. 3, pp. 251–266, Mar. 1994.

[13] T. Anderson, P. A. Barrett, D. N. Halliwell, and M. R. Moulding, "Software Fault Tolerance: An Evaluation," *IEEE Trans. Softw. Eng.*, vol. SE-11, no. 12, pp. 1502–1510, Dec. 1985.

[14] I. U. Nisa and S. N. Ahsan, "Fault prediction model for software using soft computing techniques," in *2015 International Conference on Open Source Systems & Technologies (ICOSST)*, 2015, pp. 78–83.

[15] S. N. Ahsan and F. Wotawa, "Fault Prediction Capability of Program File's Logical-Coupling Metrics," in *2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement*, 2011, pp. 257–262.

[16] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, Aug. 2013.

[17] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," *Appl. Soft Comput.*, vol. 21, pp. 286–297, Aug. 2014.

[18] Z. Xu *et al.*, "Software defect prediction based on kernel PCA and weighted extreme learning machine," *Inf. Softw. Technol.*, vol. 106, pp. 182–200, Feb. 2019.

[19] J. Moeyersoms, E. Junqué de Fortuny, K. Dejaeger, B. Baesens, and D. Martens, "Comprehensible software fault and effort prediction: A data mining approach," *J. Syst. Softw.*, vol. 100, pp. 80–90, Feb. 2015.

[20] Q. Yu, J. Qian, S. Jiang, Z. Wu, and G. Zhang, "An Empirical Study on the Effectiveness of Feature Selection for Cross-Project Defect Prediction," *IEEE Access*, vol. 7, pp. 35710–35718, 2019.

[21] J. Hotzkow and Jenny, "Automatically inferring and enforcing user expectations," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2017*, 2017, pp. 420–423.

[22] A. A. Hudaib, H. N. Fakhouri, A. A. Hudaib, and H. N. Fakhouri, "An Automated Approach for Software Fault Detection and Recovery," *Commun. Netw.*, vol. 08, no. 03, pp. 158–169, Jul. 2016.

[23] S. Montani and C. Anglano, "Achieving self-healing in service delivery software systems by means of case-based reasoning," *Appl. Intell.*, vol. 28, no. 2, pp. 139–152, Apr. 2008.

[24] "A New Way to Find Bugs in Self-Driving AI Could Save Lives - IEEE Spectrum." [Online]. Available: https://spectrum.ieee.org/tech-talk/robotics/artificial-intelligence/better-bug-hunts-in-selfdriving-car-ai-could-save-lives. [Accessed: 02-Jul-2019].

[25] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, May 2009.

[26] V. U. B. CHALLAGULLA, F. B. BASTANI, I.-L. YEN, and R. A. PAUL, "EMPIRICAL ASSESSMENT OF MACHINE LEARNING BASED SOFTWARE DEFECT PREDICTION TECHNIQUES," *Int. J. Artif. Intell. Tools*, vol. 17, no. 02, pp. 389–400, Apr. 2008.

[27] E. Shihab *et al.*, "Studying re-opened bugs in open source software," *Empir. Softw. Eng.*, vol. 18, no. 5, pp. 1005–1042, Oct. 2013.

[28] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data Quality: Some Comments on the NASA Software Defect Datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013.

[29] T. Menzies, J. Distefano, A. O. S, and R. M. Chapman, "Assessing Predictors of Software Defects."

[30] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in Proceedings of the 6th International Conference on Predictive Models in Software Engineering - PROMISE '10, 2010, p. 1.

[31] H. Tanwar and M. Kakkar, "A Review of Software Defect Prediction Models," Springer, Singapore, 2019, pp. 89–97.

[32] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," Inf. Softw. Technol., vol. 49, no. 5, pp. 483–492, May 2007.

[33] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Addison Wesley, 2005.

[34] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, vol. 72. Cham: Springer International Publishing, 2015.