2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) | 978-1-7281-9675-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICDABI51230.2020.9325677

Software Defects Prediction using Machine Learning Algorithms

Marwa Assim
Department of Computer Science
University of Bahrain
Sakheer, Bahrain
20190175@stu.uob.edu.bh

Qasem Obeidat
Department of Computer Science
University of Bahrain
Sakheer, Bahrain
qobeidat@uob.edu.bh

Mustafa Hammad
Department of Computer Science
University of Bahrain
Sakheer, Bahrain
mhammad@uob.edu.bh

***Abstract*— Software development and the maintenance life cycle are lengthy processes. However, the possibility of having defects in the software can be high. Software reliability and performance are essential measures of software success, which affects user satisfaction and software cost. Predicting software defects using machine learning (ML) algorithms is one approach in this direction. Implementing this approach in the earlier stages of the software development improves software performance quality and reduces software maintenance cost. Different models and techniques have been implemented in many studies to predict software defects. This investigation implements ML algorithms, such as artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P. A new software defect prediction model for software future defect prediction is proposed. The defect prediction is based on historical data. The results showed that a combination of ML algorithms could be used effectively to predict software defects. The SMOreg classifier scored the best performance results, where the ANN classifier scores the worst results.**

Keywords*— *software defect prediction, software defect, prediction model, machine learning (ML), artificial neural networks (ANN), SMOreg Classifier.

I. INTRODUCTION

In the last few years, software quality became one of the most critical aspects of the software. Software defect prediction is an essential part of the software development cycle, and it can increase software quality, performance and reduce maintenance costs. Defective software directly impacts software quality leading to higher software costs, delayed project plans, and higher maintenance costs. On the other hand, applying a software defect prediction model on the software before its deployment improves software performance and increases its reliability and user satisfaction. In this study, the software prediction performance was assessed using eight machine learning (ML) algorithms. The algorithms include artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P. Weka 3.8.3 tool has been used to run the algorithms using the k-fold cross-validation and percentage split techniques. Dataset used in this work was obtained from a publicly available repository from NASA Promise [1]. The data set was validated using two test modes: 10-fold cross-validation and percentage split with different training set percentages (60%, 70%, 80%, 90%). Using the WEKA tool, the two test modes applied the data classification on the eight ML algorithms.

In this investigation, different evaluation measures will be used to assess software defect prediction accuracy. According to the results of these measures, we can decide which ML algorithms we can use for software defect prediction. The evaluation metrics include correlation coefficient (R^2), mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE), and root-relative squared error (RRSE). The ML algorithm with the lowest error rates can be considered the most appropriate algorithm to apply in the prediction process.

The other sections of the paper are structured as follows: the literature review related to software prediction techniques is presented in Section II; background knowledge and description of the used Machine Learning algorithms are discussed in Section III; the dataset with evaluation methodology and experimental results are presented in Sections IV and V, respectively; the conclusion and future work are discussed in Section VI.

II. LITERATURE REVIEW

Many studies used machine learning (ML) algorithms to measure software defect prediction performance [2], [3], [4]. A recent study analyzed different ML algorithms' performance in software defect prediction. The ML algorithms used are NB, MLP, RBF, SVM, KNN, K*, OneR, PART, DTree, and RF [3]. The researchers of this study implemented the ML algorithms on 12 publicly available NASA datasets. This study evaluated the results using various measures, like accuracy, recall, precision, F-measure, ROC area, and MCC. Another study proposed a software defect prediction model based on a neural network using a levenberg-Marquardt (LM) algorithm [4]. In this study, the proposed model is compared with the polynomial neural network, and they found that the proposed model has higher accuracy results. M. Ahmad *et al.* reviewed different ML algorithms, lexicon-based techniques, and hybrid techniques, which combines ML algorithms and lexicon technique for automatic sentiment classification for user-generated data from various social media and blogging websites [5]. Arora *et al.* proposed a software defect prediction model. The proposed model is comparing two ML classifiers, which are ANN and SVM [6]. The study performed using seven datasets from the PROMISE repository and assessed the results on various measures using accuracy, recall, and specificity. The SVM is better than ANN when evaluating the recall measure results. Alsaeedi and Mustafa compared SVM, DTree, and RF classifiers on ten publicly available NASA

datasets. The results revealed that the RF algorithm outperforms other classifiers [7].

A cross-project defect arises when different projects working together, which causes a class imbalance problem in machine learning prediction. Many studies proposed different methods to overcome the cross-project defect prediction problem [7], [8], [9], [10], [11]. Goel *et al.* conducted an empirical analysis to evaluate whether the data sampling approach can solve the class imbalance issue and enhance the cross-project defect prediction (CPDP) model performance [12]. In this study, machine learning classifiers have been implemented using twelve publicly available object-oriented datasets. The results showed that the data sampling technique could be implemented to overcome the class imbalance problem on CPDP. Another study proposed an over-sampling transfer learning technique that uses the feature weighting naïve Bayes approach for CPDP [13]. Experiments were implemented on eleven public datasets. The results showed that this approach is beneficial for solving class imbalance in the CPDP model.

Most of the proposed software defect prediction models achieved a prediction performance rate of about 80% recall. As a result, Bowes *et al.* investigated four ML algorithms' performance levels, which are RF, naïve Bayes, RPart, and SVM, when predicting defects in NASA datasets [14]. The defect prediction results are presented in the confusion matrix and compared with each algorithm prediction uncertainty. Each classifier detects different sets of defects. The study concluded that a specific classifier could identify a subset of errors, and some algorithms can perform best in defect prediction if they are not based on majority voting.

According to many studies and surveys, software defect prediction models that are highly dependent on features of programs fail in detecting the semantics of programs [15], [16]. For instance, Maurice computed Halstead metrics using numbers of operators and operands [17]. Chidamber *et al.* calculated Chidamber and Kemerer (CK) metrics using the function and inheritance counts [18]. Thomas *et al.* used McCabe's metric by analyzing the software control flow graph to estimate its complexity [19]. As a result, the algorithm performance is not so high, even after adopting learning algorithms and filtering the data. Accordingly, a recent study for Phan *et al.* proposed a different software defect prediction approach to detect software defects by constructing control flow graphs extracted using a source code compilation [20]. Moreover, Phan applied multi-view convolutional neural networks with the multi-layer networks to the extracted control flow graphs to learn the semantic software features. The results showed that the performance of the defects prediction was higher than the previous techniques.

Rosli *et al.* proposed an application to detect software defects using genetic algorithms [21]. The app uses object-oriented metrics as input values to the algorithm and classifies the software modules as defective and non-defective. Cui *et al.* recent study proposed a novel software defect prediction model based on Genetic Algorithm and Back Propagation (GA-BP). The purpose of the model is to overcome the BP of neural networks, which can easily fall into local optimization when implementing software defect prediction [22]. GA is used to enhance the weights in BP neural networks. According to the results, this method is useful in software defect prediction.

Rahman proposed a software defect prediction using code profiles as a replacement for traditional measures [23]. This study trains the ML algorithm using vectors. The vectors are generated from the source codes' parse trees by analyzing language feature use. The parse trees are used as feature variables. Replacing the traditional metrics for predicting software defects with the code profiles are promising. However, the predictive performance can be considered as low in cross-version defect predictions for both metrics-based and code profile-based techniques.

In this investigation, a software defects prediction model will be proposed using eight ML algorithms. The algorithms include artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P. This study aims to assess the performance of the ML algorithms in predicting software defects. The performance of the used ML algorithms will be assessed using PROMISE Repository software engineering publicly available dataset, namely, Class-level data for KC1 (one of NASA products). WEKA tool will be used to run the eighth ML algorithms using two test modes: 10-fold cross-validation and percentage split. Different evaluation metrics will be used to measure the performance accuracy of the used ML algorithms. The evaluation metrics include: correlation coefficient (R^2), mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE), and root-relative squared error (RRSE).

III. RESEARCH METHODOLOGY

A. DATASET AND EVALUATION METHODOLOGY

In this investigation, a PROMISE repository software engineering publicly available dataset, namely, class-level data for KC1 (one of NASA products), was used. The dataset contains class-level and method-level attributes. Koru *et al.* converted 21 method-level attributes into 84 class-level attributes using maximum, minimum, average, and sum operations [24]. Besides the 84 attributes, the dataset consists of 10 class-level attributes to create 94 class-level attributes. The last numeric attribute represents the number of recorded class defects, and it is called NUMDEFECTS. It has been used as a response variable that will be used by ML algorithms in the prediction process. The dataset has 145 instances, which contains real measured data for one of NASA products for KC1. The dataset was used in several studies [25], [26], [27], [28], [29], [30]. Table I presents the first ten class-level attributes and the last NUMDEFECT attribute.

TABLE I. NUMERIC ATTRIBUTES DESCRIPTION OF KC1 DATASET

Item No.	Attribute	Description
1	PERCENT_PUB_DATA	The percentage of data that is public and protected data in a class.
2	ACCESS_TO_PUB_DATA	The number of times that a class's public and protected data is accessed.
3	COUPLING_BETWEEN_OBJECTS	The number of distinct non-inheritance-related classes on which a class depends.
4	DEPTH	The level for a class.
5	LACK_OF_COHESION_OF_METHODS	The percentage of the methods in the class.
6	NUM_OF_CHILDREN	The number of classes derived from a specified class.
7	DEP_ON_CHILD	Whether a class is dependent on a descendant.
8	FAN_IN	Count of calls by higher modules.
9	RESPONSE_FOR_CLASS	Count of methods implemented within a class.
10	WEIGHTED_METHODS_PER_CLASS	Count of methods implemented within a class.
11	NUMDEFECTS	No. of Defects

Fig. 1. illustrates the model and the research methodology used for software defect prediction. The KC1 dataset was used for the prediction process and analysis. The training and testing of the dataset were done using the 10-folds cross-validation. The results of the classifiers are then evaluated using different error measures and metrics [31].

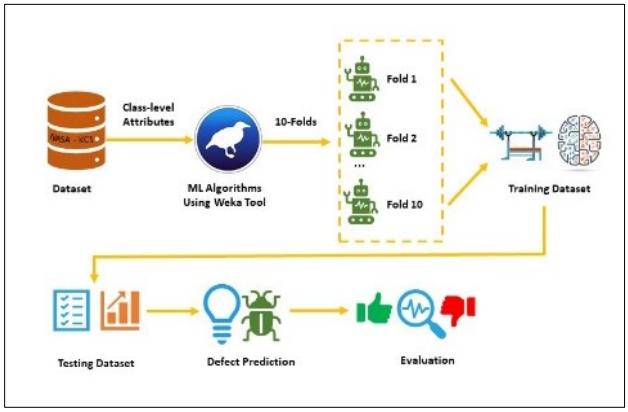

Fig. 1. Software Defect Prediction Model

The statistical metrics include correlation coefficient (R^2), mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE), and root relative squared error (RRSE). These metrics measure the error rate between the number of actual defects and the number of predicted defects in the dataset. Assuming E is the number of actual defects, $\tilde{E}$ is the number of predicted defects, $\bar{E}$ represents the mean of E, and n represents the number of instances. The following formulas represent the used performance metrics in the prediction process:

1. R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Ei - \tilde{E}i)^2}{\sum_{i=1}^{n}(Ei - \bar{E}i)^2} \quad (1)$$

2. MAE

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Ei - \tilde{E}i| \quad (2)$$

3. RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Ei - \tilde{E}i)^2} \quad (3)$$

4. RAE

$$RAE = \frac{\sum_{i=1}^{n}|Ai - \tilde{A}i|}{\sum_{i=1}^{n}|Ai - \bar{A}i|} \quad (4)$$

5. RRSE

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}(Ai - \tilde{A}i)^2}{\sum_{i=1}^{n}(Ai - \bar{A}i)^2}} \quad (5)$$

R^2 gives how much the actual and predicted values are related. These values range between −1 and 1, where 1 indicates correct positive linear relation, 0 means there is no relation, and −1 means correct negative linear relation. MAE is the average of the difference between the actual and the predicted values. RMSE is quite like MAE, but RMSE represents the square's average of the difference between the actual and the predicted values. As a result, the focus will be on more significant errors than on smaller errors [32]. RAE gives the average of the actual values, so the error is the total absolute error. RRSE is similar to the RAE because it is the average of the actual values, but the error is the total squared error [33].

B. USED MACHINE LEARNING ALGORITHMS

Machine learning (ML) uses multiple algorithms to analyze input data and predict output values. Machine learning algorithms learn and improve performance and help with automating the labeling process. Each ML algorithm is purposed to perform a specific task [34]. In this study, the performance of eight ML algorithms was measured to predict software defects. These ML algorithms are summarized as the following:

- *Artificial Neural Networks (ANNs):*

ANNs are complicated systems that simulate the function of the human nervous systems. They are made up of artificial neurons and can be used in ML implementations as a non-linear classifier by taking in multiple inputs and producing a single output. The input neurons are connected and work together in parallel by sending a signal to other neurons and using a non-linear function to calculates its output [35].

- *Random Forest (RF):*

RF algorithm is a supervised classification and regression learning algorithm. It produces a forest of decision trees and selects the best solution using committee voting rather than an individual tree decision [36].

- *Random Tree (RT):*

RT is a graphical representation of all possible solutions based on specific conditions. We call it a tree because basically, it starts with a root and has many branches that produce multiple sets of data to make a decision tree [37].

- *Decision Table (DT):*

DT is an organized model to form requirements with business rules. It is a classifier that can be used to model complicated logic. In a DT, conditions are labeled as true (T) or false (F). Each column in the table represents a business logic rule that describes the unique combination of instances.

- *Linear Regression (LR):*

LR is a supervised learning algorithm. It predicts the results by creating a linear regression relationship between an independent variable (x) to predict a dependent variable value (y) [38].

- *Gaussian Processes (GP):*

GP is a distribution of probabilities of possible results. It is a collection of random variables represented by time or space, where there is a multivariate distribution for every group of those random variables [39].

- *SMOreg:*

SMOreg implements a vector machine for regression and applies multiple algorithms for parameter learning. The algorithm replaces the missing values and transforming the nominal attributes into binary ones [40].

- *M5P:*

M5P is a construction of the M5 algorithm, which can produce trees of regression models. M5P combines both conventional decision tree and linear regression functions at the nodes.

IV. EXPERIMENTAL RESULTS

In this investigation, the aforementioned eight ML algorithms were used across the Weka 3.8.3 platform to predict software defects. Cross-validation (10-fold) and percentage split (60%, 70%, 80%, and 90%) testing modes were used in the tested dataset. Table II shows the statistical metrics to evaluate the classifiers' error rates in predicting the selected dataset's software defects.

As shown in Table II, the LR algorithm achieved the highest correlation coefficient (R^2) value in the KC1 dataset, followed by the SMOreg algorithm. On the other hand, the SMOreg algorithm achieved the lowest error rates for the remaining metrics. Therefore, the SMOreg classifier scored the best performance results among the eighth ML algorithms. In contrast, the ANN algorithm scored the worst performance results. Besides, the error rates for both metrics, RAE and RRSE, reached more than 100 %, where the algorithm was not able to predict the valid values because the values were constant for all the input data.

TABLE II. STATISTICAL PERFORMANCE RESULTS FOR THE ML ALGORITHMS USING 10 FOLDS CROSS-VALIDATION TESTING MODE

CLASSIFIER	R^2	MAE	RMSE	RAE	RRSE
ANN	0.1584	6.9869	14.7296	115.6751	134.899
Random Forest (RF)	0.4995	4.4208	9.3905	73.1899	86.0011
Random Tree (RT)	0.1317	5.5088	12.2059	91.2029	111.7856
Decision Table (DT)	0.2663	4.9232	11.2294	81.5082	102.8431
Linear regression (LR)	0.7594	6.5889	11.1759	109.0851	102.3526
Gaussian Processes (GP)	0.6794	4.6329	7.9437	76.7025	72.7514
SMOreg	0.7383	4.3159	7.4378	73.1085	68.1179
M5P	0.6989	4.3618	7.7834	72.2135	71.2835

The distribution graph of the actual and predicted values of the SMOreg classifier are shown in Fig. 2. The graph has two dimensions, where the dataset instance's number is represented on the x-axis. On the other hand, the actual and predicted defect values are represented on the y-axis. Fig. 1 also showed that the predicted defects' values are almost the same as the actual defects' values in most 145 instances. Therefore, the error rate of the performance metrics is decreased.

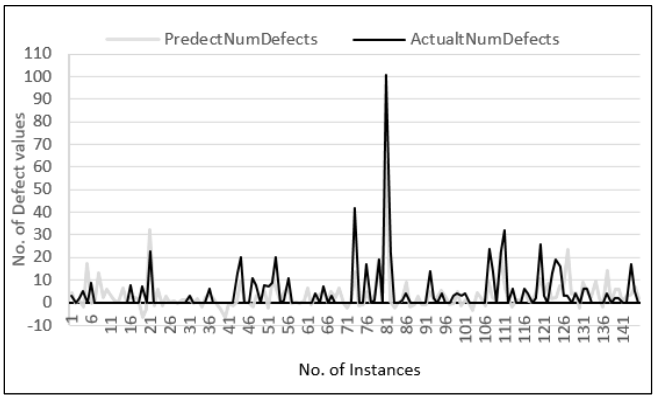

Fig. 2. The distribution of the predicted and actual defect values using the best classifier, SMOreg

Scored results of the eight ML algorithms using percentage split testing mode are shown in Table III. Different percentages for the training set and the testing set ranging from 60% to 90% were implemented. MAE statistical metric has been used as an evaluation metric for the assessment performance. We can conclude that as the percentage set is increased, the error rate will be decreased. The random forest (RF) algorithm achieved the lowest MAE error rate for both 60% and 70% training sets. On the other hand, the random tree (RT) algorithm achieved the lowest MAE error rate for both 80% and 90% training sets.

TABLE III. STATISTICAL PERFORMANCE RESULTS FOR ML ALGORITHMS USING PERCENTAGE SPLIT TESTING MODE

CLASSIFIER	60%	70%	80%	90%
ANN	6.3554	5.097	5.7436	0.8393
Random Forest (RF)	3.9995	3.604	1.9424	0.2115
Random Tree (RT)	4.3654	4.2973	1.3014	0.0303
Decision Table (DT)	6.2844	5.5632	2.5614	2.5614
Linear regression (LR)	5.9445	5.5295	4.2484	2.6998
Gaussian Processes (GP)	5.1449	5.0989	3.4081	1.9759
SMOreg	4.434	3.9551	2.7383	1.1206
M5P	4.0811	5.9991	3.0258	0.5304

V. CONCLUSION AND FUTURE WORK

Software evolution is essential, so the software is modified over time to adapt it to changing customer and market requirements. During software improvement, many software defects may arise. The software developer should update the test suites, such as adding new test cases to detect these new defects [41]. Software defect prediction is a necessary process that should be considered by the software developer before the software deployment. Software defect prediction can be implemented using ML-based prediction model. The model can use historical data for predicting future software defects [42]. This paper evaluated software defect prediction accuracy using different ML algorithms. Eight ML algorithms were used: *ANN*, *RF*, *RT*, *DT, LR, GP, SMOreg,* and *M5P*. The algorithms were applied on a publicly available dataset from the NASA promise repository, where different statistical measures were used to evaluate the error rates of the predicted values. The metrics are R^2, MAE, RMSE, RAE, and RRSE. According to the results, we can conclude that the ML algorithms are efficient techniques in predicting future software defects. First, a 10-fold cross-validation testing mode was used to measure the results. The results showed that the SMOreg algorithm, which implements a vector machine classification for regression and replacing the missing values, and normalizing all the attributes by default, has the best results among the other algorithms. In contrast, the RT algorithm achieved the lowest MAE error rate using percentage split testing mode for both (80% and 90%) training sets, respectively.

As future work, we may include the investigation of other ML classifiers and compare between them. Furthermore, we can use more datasets in the learning and prediction processes to increase prediction accuracy. We can also consider more practical software defect prediction scenarios, such as learning from software code with minimal defects.

REFERENCES

[1] Promise.site.uottawa.ca. 2005. [online] Available: http://promise.site.uottawa.ca/SERepository/datasets/kc1-class-level-numericdefect.arff [Accessed 16 March 2020].

[2] Alenezi, M., Banitaan, S., and Obeidat, Q.. "Fault-proneness of open source systems: An empirical analysis." *Synapse* 1 (2014): 256.

[3] Iqbal, Ahmed, et al. "Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets." *Int. J. Adv. Comput. Sci. Appl* 10.5, 2019.

[4] Singh, Malkit, and Dalwinder Singh Salaria. "Software defect prediction tool based on neural network." *International Journal of Computer Applications,* 70(22), 2013.

[5] M. Ahmad, S. Aftab, and S. S. Muhammad, ―Machine Learning Techniques for Sentiment Analysis: A Review, *Int. J. Multidiscip. Sci. Eng.*, vol. 8 (3). 3, p. 27, 2017.

[6] I. A. and A. Saha, ―Software Defect Prediction: A Comparison Between Artificial Neural Network and Support Vector Machine,‖ Adv. Comput. Commun. Technol., pp. 51–61, 2017.

[7] Alsaeedi, Abdullah, and Mohammad Zubair Khan. "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study." *Journal of Software Engineering and Applications* 12.5, 85-100, 2019.

[8] Wang, Shuo, and Xin Yao. "Using class imbalance learning for software defect prediction." *IEEE Transactions on Reliability* 62.2, 434-443, 2013.

[9] Tomar, Divya, and Sonali Agarwal. "Prediction of defective software modules using class imbalance learning." *Applied Computational Intelligence and Soft Computing* 2016, 2016.

[10] Song, Qinbao, Yuchen Guo, and Martin Shepperd. "A comprehensive investigation of the role of imbalanced learning for software defect prediction." *IEEE Transactions on Software Engineering* 45.12, 1253-1269, 2018.

[11] Goel, Lipika, et al. "Cross-project defect prediction using data sampling for class imbalance learning: an empirical study." *International Journal of Parallel, Emergent and Distributed Systems*, 1-14, 2019.

[12] Goel, Lipika, and Sonam Gupta. "Cross Projects Defect Prediction Modeling." *Data Visualization and Knowledge Engineering*. Springer, Cham, 1-21, 2020.

[13] Tong, Haonan, et al. "Transfer-Learning Oriented Class Imbalance Learning for Cross-Project Defect Prediction." *arXiv preprint arXiv:1901.08429*, 2019.

[14] Bowes, David, Tracy Hall, and Jean Petrić. "Software defect prediction: do different classifiers find the same defects?." *Software Quality Journal* 26.2, 525-552, 2018.

[15] Jones, Capers. "Strengths and weaknesses of software metrics." *American Programmer* 10 (1997): 44-49.

[16] Menzies, Tim, Jeremy Greenwald, and Art Frank. "Data mining static code attributes to learn defect predictors." *IEEE transactions on software engineering* 33.1, 2-13, 2006.

[17] Halstead, Maurice Howard. *Elements of software science*. Vol. 7. New York: Elsevier, 1977.

[18] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design." *IEEE Transactions on software engineering* 20.6, 476-493, 1994.

[19] McCabe, Thomas J. "A complexity measure." *IEEE Transactions on software Engineering* 4, 308-320, 1976.

[20] Phan, Anh Viet, Minh Le Nguyen, and Lam Thu Bui. "Convolutional neural networks over control flow graphs for software defect prediction." *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2017.

[21] M. M. Rosli, N. H. I. Teo, N. S. M. Yusop and N. S. Moham, "The Design of a Software Fault Prone Application Using Evolutionary Algorithm," *IEEE Conference on Open Systems*, 2011.

[22] Cui, Mengtian, Yameng Huang, and Jing Luo. "Software Defect Prediction Model Based on GA-BP Algorithm." *International Symposium on Cyberspace Safety and Security*. Springer, Cham, 2019.

[23] Rahman, Ashiqur. Software Defect Prediction Using Rich Contextualized Language Use Vectors. Diss. 2019.

[24] A. Gunes Koru and Hongfang Liu, "An Investigation of the Effect of Module Size on Defect Prediction Using Static Measures," *PROMISE-Predictive Models in Software Engineering Workshop*, *ICSE* 2005, Saint Louis, Missouri, US, May 15th 2005.

[25] Shanthini, A. "Effect of ensemble methods for software fault prediction at various metrics level," 2014.

[26] Shanthini, A., and R. M. Chandrasekaran. "Analyzing the effect of bagged ensemble approach for software fault prediction in class level and package level metrics." *International Conference on Information Communication and Embedded Systems (ICICES2014)*. IEEE, 2014.

[27] Catal, Cagatay, Banu Diri, and Bulent Ozumut. "An artificial immune system approach for fault prediction in object-oriented software." *2nd International Conference on Dependability of Computer Systems* (DepCoS-RELCOMEX'07). IEEE, 2007.

[28] Gayatri, N., et al. "Feature selection using decision tree induction in class level metrics dataset for software defect predictions." *Proceedings of the world congress on engineering and computer science* vol. 1. 2010.

[29] Wu, Fang Jun. "Empirical tests of scale-free geometry in NASA data."*Advanced Materials Research*, vol. 301. Trans Tech Publications Ltd, 2011.

[30] Singh, Yogesh, Arvinder Kaur, and Ruchika Malhotra. "Software fault proneness prediction using support vector machines." *Proceedings of the world congress on engineering*, vol. 1. 2009.

[31] Botchkarev, Alexei, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology." *arXiv preprint arXiv*:1809.03006, 2018.

[32] "Metrics to Evaluate your Machine Learning Algorithm", Medium, 2020. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234. [Accessed: 13- Apr- 2020].

[33] "Relative Absolute Error", Gepsoft.com, 2020. [Online]. Available: https://www.gepsoft.com/GeneXproTools/AnalysesAndComputations/MeasuresOfFit/RelativeAbsoluteError.htm. [Accessed: 13- Apr- 2020].

[34] Dey, Ayon. "Machine learning algorithms: a review." *International Journal of Computer Science and Information Technologies* 7, no. 3, 1174-1179, 2016.

[35] Niculescu, S. P., "Artificial neural networks and genetic algorithms in QSAR". *Journal of molecular structure: THEOCHEM*, *622*(1-2), 71-83,2003.

[36] A. Kaur and R. Malhotra, "Application of Random Forest in Predicting Fault-Prone Classes," *2008 International Conference on Advanced Computer Theory and Engineering*, Phuket, pp. 37-43, 2008.

[37] Kalmegh, S., "Analysis of weka data mining algorithm reptree, simple cart and randomtree for classification of indian news", *International Journal of Innovative Science, Engineering & Technology*, *2*(2), 438-446, 2015.

[38] J. Brownlee, "Linear Regression for Machine Learning", Machine Learning Mastery, 2020. [Online]. Available: https://machinelearningmastery.com/linear-regression-for-machine-learning/. [Accessed: 13- Apr- 2020].

[39] O. Knagg, "An intuitive guide to Gaussian processes", Medium, 2020. [Online]. Available: https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d. [Accessed: 13- Apr- 2020].

[40] "SMOreg (Documentation for extended WEKA including Ensembles of Hierarchically Nested Dichotomies)", Dbs.ifi.lmu.de, 2020. [Online]. Available: https://www.dbs.ifi.lmu.de/~zimek/diplomathesis/implementations/EHNDs/doc/weka/classifiers/functions/SMOreg.html. [Accessed: 13- Apr- 2020].

[41] Alsolami, N., Obeidat, Q., and Alenezi, M.. "Empirical Analysis of Object-Oriented Software Test Suite Evolution." *International Journal of Advanced Computer Science and Applications* 10.11 (2019).

[42] Abdulshaheed, M., Hammad, M., Alqaddoumi, A., and Obeidat, Q., "Mining historical software testing outcomes to predict future results". *Compusoft*, 8(12), pp.3525-3529, 2019.