ISSN 2278-3091
Volume 11, No.4, July – August 2022
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse011142022.pdf
https://doi.org/10.30534/ijatcse/2022/011142022

Prediction of software defects by knowledge graph and genetic algorithm

Iman Shafiei Alavijeh
Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran,
iman.shafiei@srbiau.ac.ir

Received Date : June 12, 2022 Accepted Date : July 15, 2022 Published Date : August 06, 2022

ABSTRACT

Software defect detection is one of the biggest software development challenges and accounts for the largest budget in the software development process. One of the effective activities for software development and increasing its reliability is to predict software defects before reaching the test stage, which helps to save time in the production, maintenance and cost process. This research aims to present a software defect prediction method based on knowledge graphs and automated machine learning. We use knowledge acquisition, knowledge fusion, knowledge storage and knowledge calculation and other knowledge map construction technology research, to realize the knowledge map recommends high-quality software defect prediction models as the hot-start input conditions for automatic search. The empirical study uses NASA's open-source dataset experimental objects and six performance evaluation indicators include Precision, Recall, PRC (Precision Recall Characteristic), ROC (Receiver Operating Characteristic), F-Measure, MCC (Matthews Correlation Coefficient). The experimental results show that the proposed model performs better than the traditional classic software defect prediction model recommended by the knowledge map in terms of different datasets and evaluation indicators.

Key words: Software Defect Prediction, Machine Learning, Knowledge Graphs

1. INTRODUCTION

Software testing is an important measure to ensure the quality of software products and improve the software's credibility [1]. At present, in third-party confirmation testing, defects in software function modules are mainly found through code review and dynamic testing [2]. However, this static and dynamic test method relies heavily on human abilities and experience and requires a lot of code review time and dynamic test coverage analysis[3]. Software defect prediction can provide a basis for judging the defect tendency of function modules in the test planning stage, which helps code review to allocate resources reasonably, improve testing efficiency, and improve software testing quality [4]. However, software defect prediction technology is in software engineering practice. The reason is that the software defect prediction effect is not only related to the distribution of defect datasets is also mainly limited by the software defect prediction model [5]. How to effectively improve the performance evaluation indicators of the software defect prediction model has become an urgent problem to be solved in the application of software defect prediction [6]. In recent years, software defect prediction has become a hot spot in intelligent software engineering [7]. Software defect prediction is mainly historical data to predict potential defects in software [8]. Academic research has achieved good results. Wahono elaborated the research results of software defect prediction from 2000 to 2013 from the three dimensions of research trends, datasets, methods and frameworks [5]. Hassan et al. Summarized the research results of software defect prediction from 2009 to 2018 include six categories (Bayesian algorithm, Decision Tree algorithm, Clustering algorithm, Artificial Neural Networks algorithm, Deep Learning algorithm, Ensemble Learning algorithm) for 30 software defect prediction models [9]. Chris et al. proposed the combination selection and hyperparameter optimization of the AutoWEKA classification algorithm, through 3 kinds of ensemble methods, 14 types of meta-methods, 30 basic classifiers and various hyperparameter settings to realizes the construction of network structure, adjustment of network structure, adjustment of hyperparameters, model the evaluation and other processes are all automated [10]. In the context of artificial intelligence and big data, knowledge graphs [11], automated machine learning[12] and deep learning technologies [13] are increasingly by academia and industry. The core of artificial intelligence is the design of algorithms and automated machine learning and deep learning technology lowers the threshold of artificial intelligence applications. It helps to complete the development and deployment of artificial intelligence projects [8]. With the low-threshold and automated features, automated machine learning is completely subvert the traditional testing methods in the next few years make artificial intelligence truly popular.

This paper proposes a software defect prediction model based on automated machine learning using knowledge diagrams. First, the knowledge diagram in the software defect prediction model described, and then a model based genetic algorithm is designed to predict software defects. Then, experimental research used to confirm the performance comparison test of the proposed software

defect prediction model and the traditional classical fault prediction model, followed by the performance comparison test of the proposed model and AutoWeka[1].

2. KNOWLEDGE GRAPH CONSTRUCTION ORIENTED TO THE FIELD OF SOFTWARE DEFECT PREDICTION

The construction of domain knowledge graph includes, knowledge acquisition, knowledge extraction, knowledge fusion, knowledge storage, knowledge calculation and reasoning. Software defect prediction model domain knowledge graph construction process shown in Figure 1.

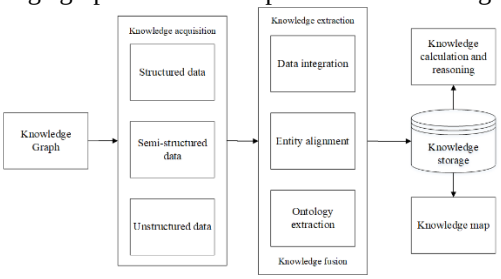

Figure 1: Knowledge graph construction process

Construct the data model of the knowledge map in the software defect prediction domain, define the structure of the entire knowledge map, and use a combination of top-down and bottom-up methods to construct the domain knowledge map of software defect prediction models. The classification of software defect prediction models includes Bayesian classifiers, neural network classifiers, functional classifiers, meta classifiers, lazy classifiers, rule classifiers, time series classifiers, tree structure classifiers, other classifiers, etc. Each software defect prediction model provides standardization input parameters, train and test data sets with various distribution laws, and build a knowledge map of software defect prediction models with multi-dimensional evaluation indicators.

The data acquired by knowledge include, structured data of software defect prediction model performance testing, semi-structured data of the extended list of software defect prediction models, and plain text unstructured data of various software defect prediction models in software engineering journals and papers. Among them, by writing automated scripts, the process of obtaining structured data for software defect prediction model performance testing is shown in Figure 2.

The knowledge fusion in the knowledge graph includes the fusion of the pattern layer and the the data layer, and knowledge extraction is in the form of RDF (resource description framework) [14]. The data structure of RDF mainly includes two forms, nodes and edges. Nodes represent entities and attributes, and edges represent relationships between entities or between entities and attributes. Import the RDF data constructed above into a graph database for storage. Graph database supports graph structure, entity and relationship representation, and query mechanism. Knowledge graph calculation includes graph mining calculation and rule-based reasoning. From Precision, Recall, PRC (Precision Recall Characteristic), ROC (Receiver Operating Characteristic), F-Measure, MCC (Matthews Correlation Coefficient) [15].

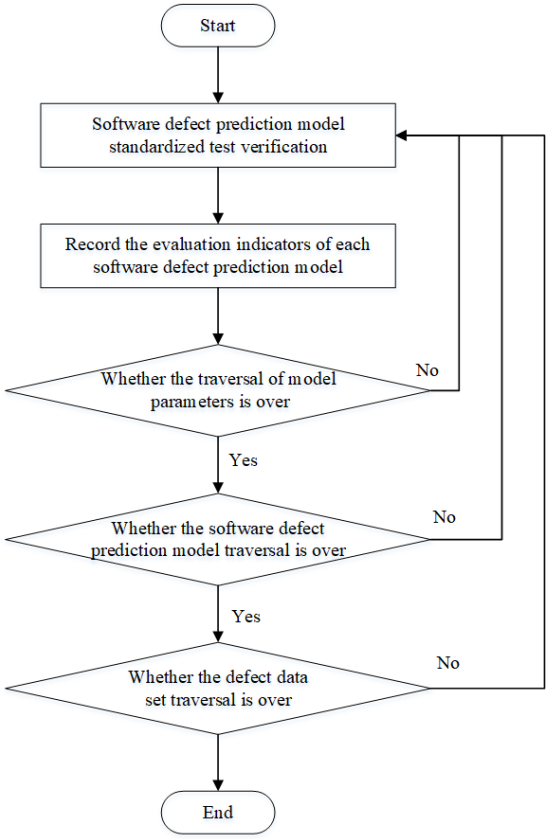

Figure 2: The process of structured data knowledge acquisition

3. DESIGN SOFTWARE DEFECT METHOD

3.1. Stacking integrated learning method

For the integrated learning of stacking structure, the integrated selected configuration includes the selection configuration of the base-level classifier and the meta-level classifier. The difficulties faced by the stacking structure are as follows: (1) Stacking configurations with high generalization accuracy are often field-related for different types of unbalanced or overlapping data sets, the best stacking configuration is different, so the same configuration is used on different data sets, the accuracy of the stacking classifier obtained may be different. (2) The generalization ability of the stacking configuration is determined by combining the base classifier and the meta classifier. These fixed-configuration methods all focus on the choice of the meta classifier, while ignoring the problem of how to choose the base classifier.

3.2. Genetic Algorithm

DEAP (distributed evolutionary algorithms in python) [16] is a novel genetic algorithm evolutionary computing framework for rapid prototyping and testing. It aims to make the algorithm clear and the data structure transparent.

[1] https://github.com/automl/autoweka

It can be perfectly coordinated between parallel mechanisms.

3.3. Proposed model and working principle

The basic idea of proposed model for software defect prediction shown in Figure 3 and Figure 4.
(1) Obtain software defect prediction models (software defect prediction models derived from Weka 3.9.5[2]) performance test.
(2) With the ranking of each evaluation index recommended by the knowledge graph, it used as the hot-start input condition for the automatic search of the software defect prediction model.
(3) Use the DEAP genetic algorithm framework to build automated search optimization.
(4) Recursive layer by layer based on meta-stacking to find model nodes that can replaced by meta-stacking.
(5) According to different evaluation indicators, optimize other best stacking model structures.
The flow of proposed model software defect prediction shown in Figure 4.

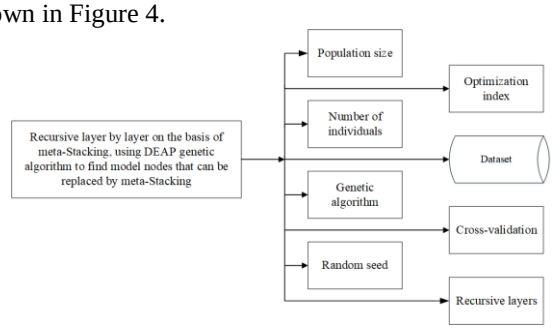

Figure 3: Proposed model base classifier selection configuration

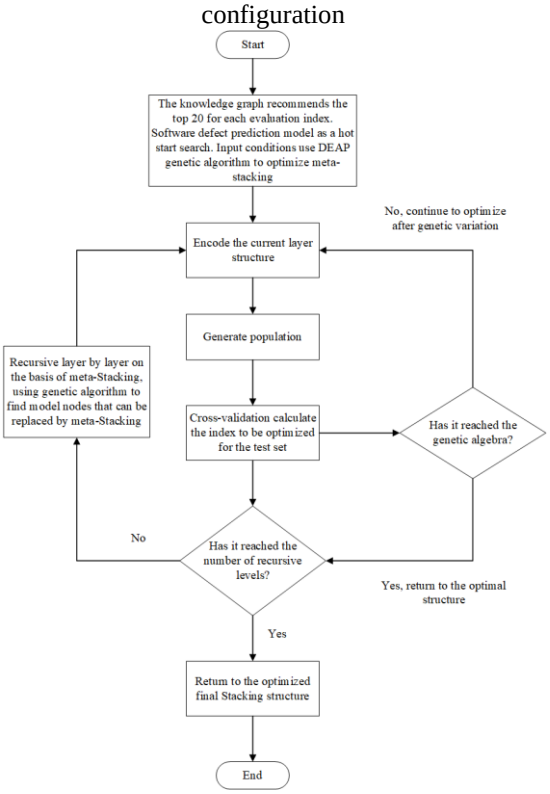

Figure 4: Proposed model software defect prediction method flow

[2] https://www.cs.waikato.ac.nz/~ml/weka/

4. EXPERIMENT AND EVALUATION

Under the conditions of various data sets with different sample sizes and defect rates, the software defect prediction model based on the knowledge graph-assisted automated machine learning proposed in this article and the traditional classic software defect prediction model used for comparative experimental testing. The experimental environment: Windows 10, Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz -2.90 GHz RAM 16 GB, Open JDK 1.8.0, Weka 3. 9. 5.

4.1- Software defect prediction prototype

Software defect prediction prototype configuration includes the selection of prediction types, the selection of data sets, the number of defect prediction models recommended by the knowledge map, the configuration of genetic algorithms (random seed, genetic algebra, population size, and mutation rate), model depth, evaluation indicators, etc.

4. 2 Experimental objects and evaluation indicators

NASA's 13 data sets [17] are all practical engineering projects of NASA, including satellite flight control software, simulator software, and ground station test software. The data sets cover C, C + +, Perl, Java. In the NASA dataset, the percentage distribution of the defect rate of function modules is 0.41% ~ 48. 80%. The classification label in the data set is whether the function module is defective. The attributes of the function module include the number of lines of code and McCabe. Measurement values, Halstead measurement values, etc. The NASA dataset is described in Table 1. Evaluation of 6 performance indicators of experimental objects including Precision, Recall, F-Measure, MCC, ROC, PRC [18].

Table 1: Software defects related to NASA data set

project name	Number of skin features	Total number of samples	Number of defective modules	Defect rate
CM1	41	505	48	9.50
JM1	22	10,878	2,102	19.32
KC1	22	2,107	325	15.42
KC3	41	458	43	9.39
KC4	41	125	61	48.80
MC1	40	9,466	68	0.72
MC2	41	161	52	32.30
MW1	41	403	31	7.69
PC1	41	1,107	76	6.87
PC2	41	5,589	23	0.41
PC3	41	1,563	160	10.24
PC4	41	1,458	178	12.21
PC5	40	17,186	516	3.00

4.3. Experimental Design

In the command line mode, through automated script testing, each test dataset is split into 66% as the training set and 34% as the test set. Each software defect prediction model is run ten times, and the average value taken as the model evaluation index.

4.3.1. Defect prediction model of knowledge graph

The category of software defect prediction models, 114 software defect prediction models (with default parameters) experimentally provide a knowledge map data visualization distribution map, divided according to the dataset, in the 13 datasets provided by NASA, include Precision, Recall, F-Measure , MCC, ROC, PRC.

4.3.2. Experiment design

Select four representative data sets CM1, KC4, PC1, PC4 from 13 data sets, and use the top 20 software defect prediction models as the base classifiers according to the evaluation indicators recommended by the knowledge graph: F-Measure, MCC, ROC, and PRC. The metaclass classifier, the experimental design is as follows:

(1) Comparative test between proposed model and classic software defect prediction models

Experiment 1: In the case of the PC1 data set ROC evaluation index, compare the proposed model with the top 20 traditional software defect prediction model tests recommended by the knowledge graph.

Experiment 2: In the case of the KC4 data set F-Measure evaluation index, compare the proposed model with the top 20 classic software defect prediction model tests recommended by the knowledge graph.

Experiment 3: In the case of the PC4 data set MCC evaluation index, compare the proposed model with the top 20 classic software defect prediction model tests recommended by the knowledge graph.

(2) Proposed model and AutoWeka automation model comparison test

Experiment 4: Under the conditions of NASA data set JM1, KC1, KC3, KC4, MC1, MC2, MW1, PC1, PC3, PC4, PC5, from the six dimensions of evaluation indicators Precision, Recall, F-Measure, MCC, ROC, and PRC, Compare proposed model and AutoWeka defect prediction model test verification.

4.4. Experimental results and analysis

(1) Test results analysis of proposed model and classic defect prediction model

Experiment 1: In the case of the PC1 data set, the ROC evaluation index of the proposed model is 0.887, which is the best performance, and the ROC comparison index shown in Figure 5.

Experiment 2: In the case of the KC4 data set, the F-Measure evaluation index of the proposed model is 0.832, which is the best performance, and the comparison index shown in Figure 6.

Experiment 3: In the case of the PC4 data set, the proposed model MCC evaluation index is 0.570, which is the best performance, and the MCC comparison index shown in Figure 7.

(2) Proposed model and AutoWeka automation model experiment results

Experiment 4: Using the default relevant parameters of the AutoWeka, under the conditions of the NASA data set JM1, KC1, KC3, KC4, MC1, MC2, MW1, PC1, PC3, PC4, and PC5, from the evaluation indicators Precision, Recall, Comparison of F-Measure, MCC, ROC, PRC in six dimensions, proposed model performance indicators surpass the AutoWeka defect prediction model. Among them, the performance evaluation indicators of the proposed model and AutoWeka shown in Figure 8.

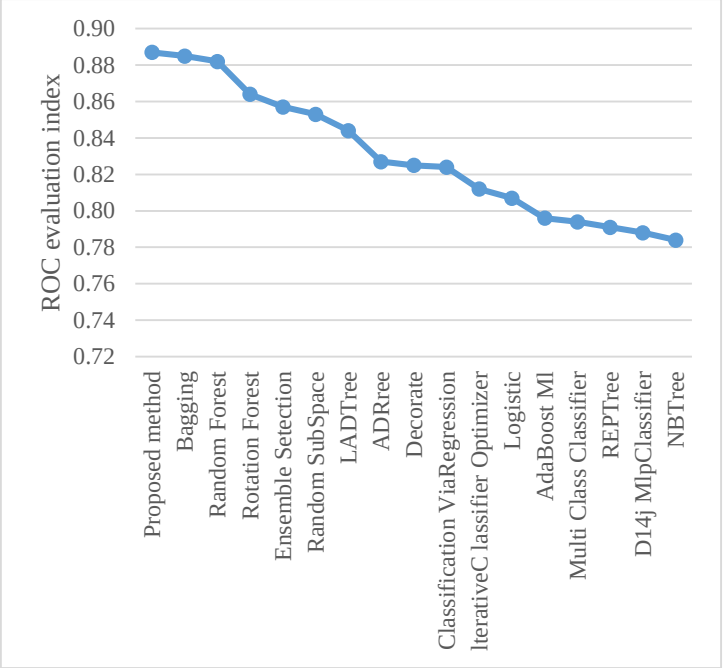

Figure 5: PC1 data set ROC evaluation

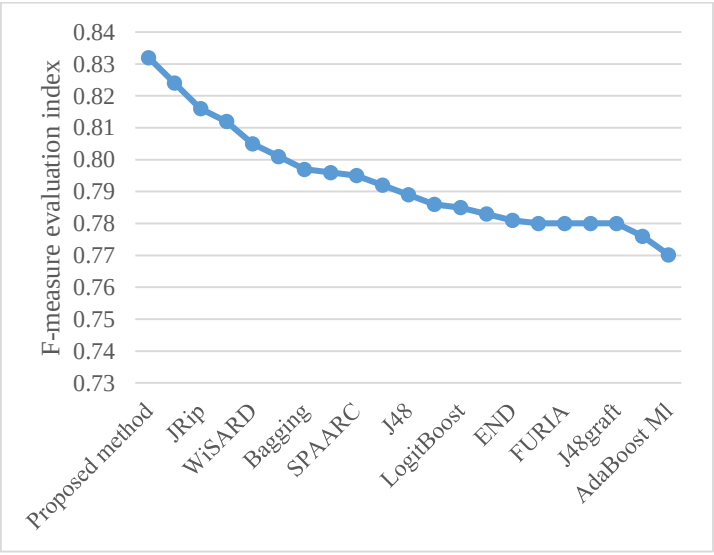

Figure 6: KC4 data set F-Measure evaluation

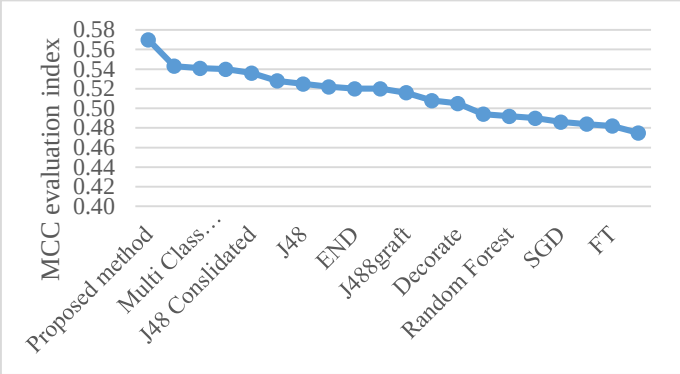

Figure 7: PC4 data set MCC evaluation

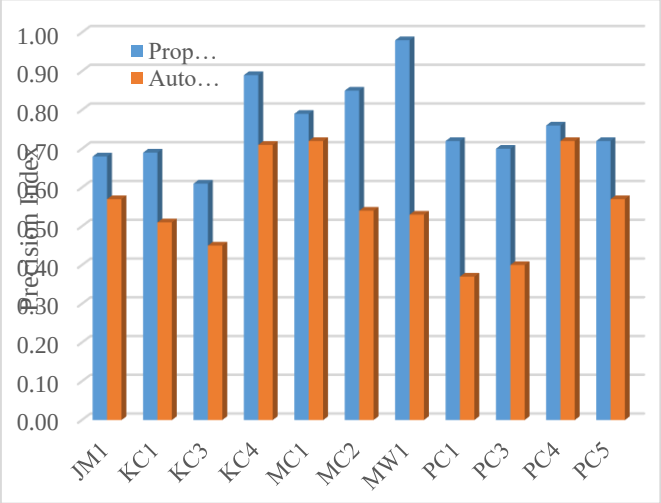

Figure 8: Proposed model and AutoWeka performance Precision

4.5. Experimental conclusion

According to the above experiments, the following conclusions can be obtained:

(1) Explore the construction of knowledge graphs in the field of software defect prediction models, obtain 1,339 software defect prediction model data samples, and the knowledge graph recommends the ranking of evaluation indicators. The traditional classic software defect prediction models used as the base classifier and metaclass classifier of the proposed model. The hot start input conditions of the automated search have achieved good results.

(2) The proposed model has better performance than the traditional classic software defect prediction model recommended by the knowledge map in terms of different datasets (PC1, KC4, PC4) and different evaluation indicators (F-Measure, MCC, ROC).

(3) Proposed model is used in NASA data set (JM1, KC1, KC3, KC4, MC1, MC2, MW1, PC1, PC3, PC4, PC5) and six dimensional evaluation indicators (Precision, Recall, F-Measure , MCC, ROC, PRC), the performance is better than AutoWeka's.

4. CONCLUSION

This paper uses the software defect prediction model as the research background, explores the construction and application of knowledge graphs, Stacking integrated learning, and proposes a software defect prediction model based on knowledge graph-assisted automated machine learning. The method empirical research uses NASA open source datasets experimental objects and six performance evaluation indicators. The experimental results show that the proposed model has better performance than the traditional classic software defect prediction model recommended by the knowledge map in terms of different data sets and different evaluation indicators. In comparison proposed model with the AutoWeka in the comparative test, the overall surpass has achieved good results.

Looking to the future, in the comparison and verification of data sets in other aerospace fields, software defect prediction based on automated deep learning and software defect prediction based on interpretable deep learning are future development trends.

REFERENCES

[1] V. Berg, J. Birkeland, A. Nguyen-Duc, I. O. Pappas, and L. Jaccheri, "Software startup engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 144, pp. 255-274, 2018.

[2] M. Varga and M. Kvassay, "Unit testing in data structures graphical learning environment," in *2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2019: IEEE, pp. 797-804.

[3] Z. Ullah, A. Lajis, M. Jamjoom, A. Altalhi, A. Al-Ghamdi, and F. Saleem, "The effect of automatic assessment on novice programming: Strengths and limitations of existing systems," *Computer Applications in Engineering Education*, vol. 26, no. 6, pp. 2328-2341, 2018.

[4] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology*, vol. 59, pp. 170-190, 2015.

[5] R. S. Wahono, "A systematic literature review of software defect prediction," *Journal of Software Engineering*, vol. 1, no. 1, pp. 1-16, 2015.

[6] X. Cai *et al.*, "An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 5, p. e5478, 2020.

[7] H. Tong, B. Liu, and S. Wang, "Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning," *Information and Software Technology*, vol. 96, pp. 94-111, 2018.

[8] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software defect prediction via convolutional neural network," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2017: IEEE, pp. 318-328.

[9] F. Hassan, S. Farhan, M. A. Fahiem, and H. Tauseef, "A Review on Machine Learning Techniques for Software Defect Prediction," *Technical Journal*, vol. 23, no. 02, pp. 63-71, 2018.

[10] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 847-855.

[11] J. Yan, C. Wang, W. Cheng, M. Gao, and A. Zhou, "A retrospective of knowledge graphs," *Frontiers of Computer Science*, vol. 12, no. 1, pp. 55-74, 2018.

[12] E. Barreiro, C. R. Munteanu, M. Cruz-Monteagudo, A. Pazos, and H. González-Díaz, "Net-net auto machine learning (automl) prediction of complex ecosystems," *Scientific reports*, vol. 8, no. 1, pp. 1-9, 2018.

[13] A. Hasanpour, P. Farzi, A. Tehrani, and R. Akbari, "Software defect prediction based on deep learning models: Performance study," *arXiv preprint arXiv:2004.02589*, 2020.

[14] V. Presutti, F. Draicchio, and A. Gangemi, "Knowledge extraction based on discourse representation theory and linguistic frames," in *International conference on knowledge engineering and knowledge management*, 2012: Springer, pp. 114-129.

[15] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques," *Expert Systems with Applications*, vol. 144, p. 113085, 2020.

[16] F.-M. De Rainville, F.-A. Fortin, M.-A. Gardner, M. Parizeau, and C. Gagné, "Deap: A python framework for evolutionary algorithms," in *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, 2012, pp. 85-92.

[17] H. Wei, C. Hu, S. Chen, Y. Xue, and Q. Zhang, "Establishing a software defect prediction model via effective dimension reduction," *Information Sciences*, vol. 477, pp. 399-409, 2019.

[18] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, "Machine learning based methods for software fault prediction: A survey," *Expert Systems with Applications*, vol. 172, p. 114595, 2021.