Materials Today: Proceedings 62 (2022) 4714–4720

Contents lists available at ScienceDirect

Materials Today: Proceedings

journal homepage: www.elsevier.com/locate/matpr

Software quality prediction using machine learning

Feisal Alaswad *, E. Poovammal

Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, Chennai 603203, India

ARTICLE INFO

Article history:
Available online 19 March 2022

Keywords:
Software Engineering
Software Metrics
Deep learning
Transfer learning
Trend analysis

ABSTRACT

Background: The demand for software is increasing every day in various fields. Software developers put more effort to develop and test the quality of the software and verify its reliability before it is released. High-quality software modules were developed to allow others to reuse the components. **Purpose**: This paper provides information to researchers in the software quality prediction field based on machine learning algorithms. **Methodology**: Most of the machine learning techniques and the relevant software metrics used in many high-quality papers published between 2010 and the end of 2021 have been analysed. **Findings**: Machine learning techniques are the most suitable for predicting software quality. Most of the researchers' interest was to predict the reliability of software. The detailed analysis enables researchers to choose the best way to plan their research and to make a good contribution in the field of software quality prediction.

Selection and peer-review under responsibility of the scientific committee of the International Conference on Innovative Technology for Sustainable Development.

1. Introduction

In recent years, artificial intelligence has entered into various fields of life. In the field of manufacturing industries instead of human beings, many robots are engaged. These robots are equipped with brains which got trained with machine learning algorithms (ML). The relationship between artificial intelligence (AI) and software engineering (SE) is getting closer day by day, although the gap between the integration of these two fields is still large compared to the relationship between AI and other sciences. Machine learning algorithms have recently started playing a major role in the software industry as well as in the life of software developers. These Software developers have sought to develop ML based software that help them to complete their tasks efficiently [1]. There are models to partially generate the code or estimate the cost, effort, and quality of the required software. Many researches have proven the ability of machine learning techniques to provide information and take measures that will speed up the process of developing software and sustain its efficiency. Not only at the level of processes, but at the level of the project as a whole, including the calculation of efforts, resources and financial issues as well as the time required for release [2].

2. Literature survey and background

Software quality is the main criterion for its evaluation. It does not mean the quality of the services it provides, but rather its quality in providing these services. The quality verification process is considered as one of the most important processes before the software release [3]. Software quality can be defined as the performance of the software within a specific work environment, taking into account the effort, time and cost required to maintain it when a failure occurs. There are two types or perspectives of quality assessment. The first is internal quality and the second is an external quality. Internal quality represents the quality of the software product and includes the quality of work during the software development lifecycle (SDLC), starting from the requirements specification stage to the design stage. The design stage includes building correct UML diagrams that reflect all possible scenarios. Then in the implementation stage writing the quality code is essential so that the code should be correct, free of vulnerabilities, and avoid excessive consumption of resources. Finally, the maintenance stage of SDLC requires a periodic effort to keep the software performing its work under changing conditions at the same level of efficiency. About external quality, it begins after the deployment stage, so that the performance of the software is measured based on several metrics these determine the functional efficiency of its performance in providing its services, such as the number of errors

* Corresponding author.
E-mail address: feisal.alaswad@hotmail.com (F. Alaswad).

https://doi.org/10.1016/j.matpr.2022.03.165
2214-7853/
Selection and peer-review under responsibility of the scientific committee of the International Conference on Innovative Technology for Sustainable Development.

that occur and number of maintenance times required periodically [4].

3. Methodology

Software quality can be described from a practical perspective by a set of attributes. Each of these attributes determines one aspect of software quality. These attributes are reliability, functionality, efficiency, maintainability, portability, and usability. Fig. 1 shows software quality model according to ISO/IEC 9126 standard. These attributes can be measured based on three different types of metrics, process metrics, project metrics, and product metrics. These metrics represent the state of the software from its early stages till after its deployment for several years. In this work, the capabilities of machine learning algorithms to predict software quality will be discussed, as well as the metrics and techniques that can be relied upon. The papers studied were selected based on their close relevance to the topic and their recentness, as all the papers studied are between 2010 and 2021. This section presents first the different types of metrics used in measuring quality, which represent the inputs of machine learning models. Then the prediction mechanisms and techniques for each of the previously mentioned quality attributes are discussed.

3.1. Software metrics types

Software metrics are classified into three main types. Each of these metrics types takes into account specific conditions and environments during the software life cycle [5]. Table 1 shows the different types of metrics with their specific characteristics during (SDLC). Table 2 shows the available datasets for the software quality metrics. The nature and what each type of metric represent will be explained later.

3.1.1. Product metrics

Product metrics are classified into two main categories, dynamic and static product metrics. Static product metrics reflect the state of the product independently of the work environment. They represent the measurements taken during all stages of software development from a software point of view, which start from the stage of requirements specification until the stage of deployment. Metrics for specifications quality represent the quality of the stage of collecting requirements. These should be characterized by completeness, understandability, achievability, verifiability, understandability, and precision [6]. While metrics for design quality represent the quality of the design stage, including architectural metrics, component metrics, and user interface metrics. Finally, metrics for coding quality include code complexity, size and cohesion coupling, etc. The dynamic product metrics represent the state of the product in operation within a specific environment, including the ability to detect failures, the execution time required to perform the tasks, and the required resources.

3.1.2. Process metrics

These metrics represent the measurements taken during all stages of software development from a developer team point of view. These include efficiency, productivity, time-duration in locating and removing defects, cycle time, lead time, throughput and error rate, etc. These metrics assess the condition of each process and are greatly influenced by both other types. One of the most important examples is that the number of possible errors is affected by the quality of the code, which is a product metric, as well as affected by the programmer's experience or number of developers which is a project metric.

3.1.3. Project metrics

These metrics represent the measurements taken during all stages of software development from a project manager's point of view. These help the project manager to verify that the project is progressing according to the required timeline. Provides information that helps make better decisions when deviations from the established plan occur. Often these metrics are represented by effort, time, and cost, which can be changed several times within the project work plan as needed. Understanding them helps reduce cost, time, effort, and potential risks.

3.2. Software quality prediction models

Learning falls under the so-called science of artificial intelligence, which depends on various algorithms according to their function and method of work. Some of these algorithms are used to learn with supervision, with Semi-supervision, or unsupervision. In this section, all the distinguished technologies that are believed to have made a good contribution in this field will be reviewed.

3.2.1. Software quality prediction models-based machine learning

Machine learning algorithms are a group of algorithms that rely on learning based on prior knowledge to predict and present output values. Most of these algorithms depend on equations and statistical models to represent learning, and then through these models, it is possible to extract knowledge and predict new knowledge. What interests us in this research is to shed light on the mechanisms of software quality prediction using machine learning. Machine learning algorithms can predict software quality, as done in Research [7], where researchers evaluated the performance of eight machine learning techniques to predict software reliability and maintainability. Reliability is expressed as the number of defects in the software, while maintainability is seen as the number of changes required in the system, whether adding, deleting, or modifying. In the aforementioned research, models were evaluated random forest (RF), Naïve Bayes (NB), J48, Bayesian networks (BN), K-nearest neighborhood (KNN), PART, support vector machine (SVM), and artificial neural networks (ANN). The researchers in [7] relied on Object-Oriented Programming metrics (OO metrics) which are extracted from the PROMISE Dataset. PRO-

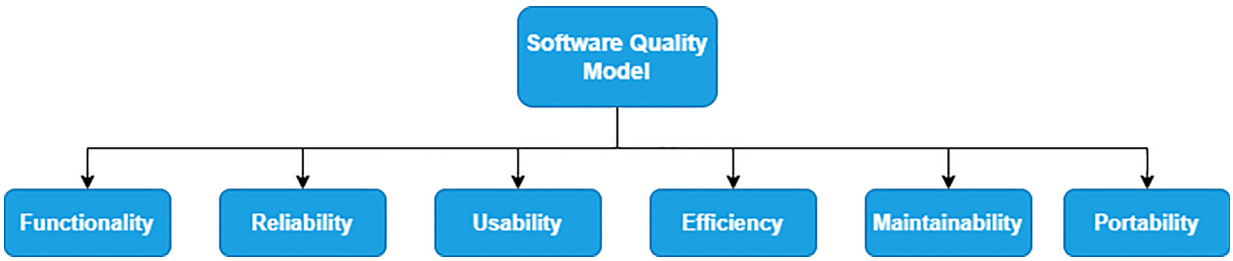

Fig. 1. ISO/IEC 9126 software quality model.

Table 1
Software attribute and metrics during SDLC.

	Product		Process		Project	
	Internal	External	Internal	External	Internal	External
Requirements	Size, reuse, complexity & redundancy	Understandability, achievability, completeness & functionality	Time & effort	Cost required	Communication level	Productivity
Specifications	Size, reuse, modularity, redundancy & precision	Understandability, achievability, completeness, independently & comprehensibility	Time, effort & number of requirements changes	Cost required	Experience	Productivity, usability
Design	Size, reuse, modularity, coupling, cohesion & simplicity	Comprehensibility, suitability, quality & functionality	Time, effort & number of specifications changes	Cost required	Experience	Usability & comprehensibility
Code	Size, reuse, modularity, coupling, cohesion, control flow & complexity	Reliability, usability, reusability, security, maintainability & compatibility	Time, effort & number of design changes	Cost required	Team developers experience & number of developers	Usability & reliability
Test	Size, coverage level & methods	Completeness, clearness & testability	Time, effort & number of code changes	Cost required	Number of testers	Reliability
Maintenance	Time to maintain	Compatibility, clearness & testability	Time to detect failure	Cost required	Technical expert & number of experts	Reliability

Table 2
Dataset available used for software quality predication.

Dataset name	Dataset source	Dataset purpose	Number of records	Link
JM1	NASA	Defect prediction	10,885	http://promise.site.uottawa.ca/serepository/datasets/jm1.arff
KC!	NASA	Defect prediction	2109	http://promise.site.uottawa.ca/serepository/datasets/kc1.arff
Cocomo v1 & 81	Jairus Hihn, JPL, NASA, Manager SQIP	Cost estimation	125	http://promise.site.uottawa.ca/serepository/datasets/cocomonasa_v1.arff http://promise.site.uottawa.ca/serepository/datasets/cocomo81.arff
SF110 Corpus	SourceForge	Testability prediction	24,000	https://www.evosuite.org/experimental-data/sf110/
Antlr4	Oracle	Defect/Maintainability prediction	2506	https://www.kaggle.com/akshays12/antlr4
CM1	NASA	Defect prediction	498	https://data.nasa.gov/
Github	Github MS	Defect/Maintainability prediction	18 TB	https://ghtorrent.org/

MISE dataset includes several projects written in different programming languages. These metrics are related to the nature of the code such as length of code (LOC), cyclomatic complexity (CC), lack of cohesion (LOCM), weighted method per class (WMC), and the number of children (NOC). The results of the research showed that the best algorithm is RF, whether to predict reliability or maintainability. In [8], a comprehensive experiment was introduced by Medeiros et al showing the impact of selecting the most appropriate software metrics on accuracy in detecting vulnerable code. It also relied on object-oriented software metrics extracted from software projects written by C++ (Apache HTTPd, Firefox, Mozilla, Linux Kernel, Glibc, and Xen). In [8], researchers used statistical-based filtering to determine the most useful metrics for detecting vulnerable code. Correlation between metrics was calculated using Pearson and Spearman correlation coefficients to identify irrelevant metrics. In the end, the researchers applied all possible subsets of metrics to the classifiers such as RF, DT, SVM linear, SVM radial. Each of the SVM linear and DT classifiers provided the best results in most cases. Software quality prediction research is not limited to the use of pure models, but some research presented hybrid models that combine optimization algorithms and machine learning models. Bouktif et al presented in [9] a new approach based on integrating decision trees (DT) algorithm with genetic algorithm (GA). They relied on product and process metrics such as the effort and time required to build the code parts as well as OO metrics. The main objective of combining the GA with DT is to derive the model that presents the least number of errors for all dataset samples. In [10] Okutan et al used the software metrics found in PROMISE dataset. They also suggested two other metrics: the number of developers (NOD) and lack coupling quality (LOCQ). The result showed that the number of developers and the level of software quality has a positive correlation.

3.2.2. Software quality prediction models-based deep learning

Machine learning contains a subset of algorithms called deep learning algorithms, where learning algorithms are classified according to their working mechanism as well as their architecture. Deep learning algorithms are based mainly on ANN, which works similarly to the human brain. Where Learning can be supervised, unsupervised or semi-supervised. What distinguishes these algorithms is their ability to give high accuracy when the appropriate amount of data is available. Also, these algorithms do not need the stage of extracting features before training, as they can extract the distinctive features from the raw input samples. This is not considered important when dealing with software metrics, which are essentially extracted features by themselves. Therefore, the

Table 3
Summary of the papers for software quality prediction using machine learning models.

Ref	Required prediction	Dataset used	language/ Metrics	Proposed approach / remarks
[7]	Reliability & Maintainability	PROMISE	C, C++, Java	Several models have been evaluated such as J48, RF, NB, Bayesian network, PART, KNN, SVM and ANN. RF model has provided the best result with AUC = 0.8 for defect and maintenance prediction.
[8]	Reliability	Apache HTTPd, Firefox, Mozilla, Linux Kernel, Glibc and Xen	C++	Studied the impact of selecting the most appropriate metrics on accuracy in detecting vulnerable code, SVM Radial, Extreme Boosting, Decision Tree, Random Forest, SVM Linear, and have evaluated. best accuracy gained from SVM and DT.
[9]	Reliability	Nasa Dataset	C, C++, Java	Genetic algorithm is developed to combine decision tree model in order to derive a composite" model that performs accurately.
[10]	Reliability	PROMISE	C, C++, Java	They suggested two new metrics: number of developers (NOD) and lack coupling quality (LOCQ). The result showed a positive correlation between the number of developers and the level of software quality.
[14]	Reliability	Private dataset	C++	To make a software quality estimation system The authors used case-based reasoning (CBR) based on Euclidean distance and Manhattan distance.
[15]	Reliability	PROMISE	C, C++, Java	The objective is to compare the performance of an ensemble model with baseline model. showed the effect of number of base predictors in the ensemble on the prediction performance of the algorithms.
[16]	Reliability	Nasa Dataset	C, C++, Java	In this work, customized WHICH framework with AUC out-performs all other methods (manualUP, manualDown, naive bayes, micro-20, Ripper)
[17]	Reliability	Apache POI	Java	Statistical approach vs six ML approaches have been evaluated to predict fault detection model. Random. RF and bagging out-performed the other models
[18]	Reliability	MIS (private dataset)	NA	Introduced new software prediction model using Bayesian model-based on finite Dirichlet mixture models.
[19]	Maintainability	UIMS, QUES	Classical Ada	Showed superiority of Probabilistic Neural Network (PNN) with Gaussian activation function, Group Method of Data Handling (GMDH) and Genetic Algorithms (GA) to predict maintainability.
[20]	Reusability	Private dataset	NA	Proposed a hybrid algorithm that combines the capabilities of K-means and decision trees to predict reusability.
[21]	Maintainability	Eclipse Mylyn Code	Java	A mini-study used the relationship between only four software criteria to predict maintainability using fuzzy inference
[22]	Reliability	Nasa Dataset	C,C++, Java	Bayesian Network approach used to predict software quality the results obtained from the proposed model 80%.
[23]	Reusability	Private dataset	NA	In this study the authors proposed SVM model to measure and estimate the reusability in the software using regularity metric, Halstead software, reuse frequency metric, coupling metric and Mccabe's cyclometric complexity.
[24]	Reliability	Nasa Dataset	C,C++, Java	The paper discussed the idea of do different classifiers find the same defects? The results prove that each classifier will detect a unique subset of defects. classifier ensembles with decision-making provide the best performance.
[25]	Maintainability	UIMS, QUES	Classical Ada	In this work, five learning models (RGF, MLR, GRNN, KNN and GBA) were built and evaluated based on features selection techniques (Pearson's Correlation, Backward Elimination and Lasso Regularization)
[26]	Reusability	Eclipse	Java	Weka software was used to make a software quality prediction based on class-level and method-level. The gained accuracy 98.64
[27]	Maintainability	UIMS, QUES	Classical Ada	Nineteen regression model evaluated to predict software maintainability. the result revealed that Quadratic SVM provided the best performance.
[28]	Maintainability	PROMISE	C,C++, Java	SVM with different kernels have been used for predicting maintainability. Several methods of selecting features are discussed in this research.
[29]	Maintainability	QUES	Classical Ada	Here individual machine learning models were compared with ensemble models to predict software maintainability. bagging ensemble based KNN was the best.
[30]	Reliability & Reusability	Private dataset	OO Metrics	Proposed KNN model for defect prediction, and reusability. accuracy of is 82%,93% for reusability is 82% and defect prediction respectively.

benefit of using neural networks in most quality prediction models is their ability to select the best features. Jha et al presented in [11] a proposal to use the long short-term memory (LSTM) network to predict software maintainability based on a dataset containing 299 software projects. The performance of the proposed model outperformed the performance of five other models are quantile regression forest, ridge regression, DT, SVM, and PCA. An advanced neural network-integrated with hybrid cuckoo search (HCS) algorithm has been developed by Sheoran et al in [12]. Where HCS is used to solve optimization problems. In this work [12], the test cases generated by the software were relied upon. The process of updating the weights in the advanced network depended on HCS. This method provided a highly effective prediction of software quality, both in terms of reliability and maintainability.

3.2.3. Software quality prediction models-based transfer learning

Pre-trained models with little training and customization can be used to predict new knowledge. These models, which are called transfer learning models, are characterized by their need for a small time to train on a relatively large amount of data compared to the large time required to train models built from scratch. In [13] researchers proposed a transfer learning model based on the Naïve Bayes algorithm to select the features. This model provided better accuracy and speed of execution compared to the CC and NN-filter methods.

4. Survey results and trends analysis

The results of the survey provided expected perceptions about the nature of the ongoing research in recent years. Table 3 provides a comprehensive overview of the data and techniques used in the studied research that rely on machine learning models. Table 4

Table 4
Summary of the papers for software quality prediction using deep learning models.

Ref	Required prediction	Dataset used	language/ Metrics	Proposed approach / remarks
[11]	Maintainability	299 open-source projects	OO Metric	Proposed LSTM neural network model was compared with five models which are DT, SVM, RR, QRF, PCA, and the proposed model performed better for predicting software maintainability.
[12]	Reliability & Maintainability	Test cases generated dataset	NA	An advanced neural network integrated with HCS algorithm has been developed. The process of updating the weights in the advanced network used HCS. This method provided a highly effective prediction of software quality, both in terms of reliability and maintainability.
[31]	Reliability	PROMISE , AR dataset, Nasa dataset	C, C++, Java	Proposed hybrid model combining salp swarm algorithm (SSA) and backpropagation neural network for software fault prediction modeling.
[32]	Reliability	PROMISE	C, C++, Java	New 36 conceptual coupling-based metrics suite was introduced. These new metrics have been tested on classifiers, including neural networks, and performed well.
[33]	Reliability	MIS & NASA	Java	Deep learning model was proposed to predict the number of defects. This model outperformed the SVR, FSVR and DTR models in terms of performance.
[34]	Maintainability	Abdera, Ivy & Rave	Java	The authors introduce an optimized extreme learning machine which is neural network with single hidden layer to predict software maintainability for three different datasets.
[35]	Reusability	COMETS	Java	In this study, code classes were categorized into three levels of reusability. SOM is based on a Competitive Learning rule used to predict reusability.
[36]	Maintainability	UIMS & QUES	Classical Ada	functional link artificial neural networks-based Hybrid models have been applied to predict maintainability. The results showed the importance of the process of features reduction, and the best results were by using hybrid network models with genetic algorithm.
[37]	Reusability	Haskell packages	Haskell	The Authors introduce MOGA-NN which is neural network trained by GA and RBF-NN with KNN to predict reuse estimate of three packages
[38]	Maintainability	UIMS & QUES	Classical Ada	Ten software standards are adopted to build a fuzzy logic neural network model. Features reduction techniques such as PCA and RSA were used to select the best Metrics a from the ten criteria.
[39]	Reusability	SourceForge	Java	They used three types of metrics (Cohesion, Complexity and coupling metrics). then these metrics after processed it used to train several ML model. the best model was AGA-ANN which based on Genetic algorithm and ANN.
[40]	Reliability	Nasa Dataset	C,C++, Java	For software defect prediction three cost-sensitive boosting algorithms are studied to boost neural networks.
[41]	Reliability	Javabean software	Java	Nelder–Mead approach with ant algorithm and ELM as hybrid model (ACO-NM) used to predict quality. Where ACO–NM used to update ELM weights.
[42]	Reliability	Nasa Dataset	C,C++, Java	The authors proposed text mining model to classify each defect report into risk levels that based on an existed classification done by RBF neural network

Table 5
Summary of the papers for software quality prediction using transfer learning models.

Ref	Required prediction	Dataset used	language/ Metrics	Proposed approach / remarks
[13]	Reliability	PROMISE	C,aC++, Java	A transfer learning model based on the NB algorithm is proposed and the goal is to build a fast and efficient model for software quality prediction. This model provided better accuracy and speed of execution compared to the CC and NN-filter methods
[43]	Reliability	PROMISE	C,aC++, Java	Feature based transfer learning model proposed. A comparison was made between the accuracy of models NB, J48 and oneR, it was found that there was an improvement in performance when using the proposed model.
[44]	Reliability	PROMISE	C,aC++, Java	The main idea in this work is visualize the code as images and then the features are extracted using self-attention mechanism. Finally, these images were provided to the transfer learning model to detect the defect in the code
[45]	Maintainability	Linux, MySQL, HTTPD, and AXIS	C,aC++	Framework proposed used TrAdaBoost transfer learning model to predict cross-project bug. The results showed the proposed model improve the accuracy to predict cross-project bug type.
[46]	Reliability	Nasa Dataset	C,aC++, Java	They proposed a new model TCANN based on Transfer learning the aim is to solve three common problems during bug detection. The problems are noise data, transfer learning among crossproject and imbalance data.
[47]	Reliability	ReLink & AEEEM	C,aC++	They proposed an extended transfer component analysis model (TCA +) to gained high cross-project accuracy prediction. The main idea is selection the most appropriate normalization options for each cross-project pair.

Table 6
Summary of reviewed articles vs learning types used.

Models\Attribute	Reliability	Maintainability	Reusability
Machine learning models	[7,8,9,10,14,15,16,17,18,22,24,30]	[7,19,21,25,27,28,29]	[20,23,26,30]
Deep learning models	[12,31,32,33,40,41,42]	[11,12,34,36,38]	[35,37,39]
Transfer learning models	[13,43,44,45,46,47]	[45]	
Other models	[48,49,50]	[49]	

provides a comprehensive overview of the data and techniques used in the studied research that rely on deep learning models.

Table 5 shows the summary of the papers for software quality prediction using transfer learning models. Table 6 shows the summary of reviewed articles vs learning types used. Most of the research focused on predicting the defects in the software. Fig. 2 (a) shows the number of articles conducted to predict specific software attributes. Fig. 2 (b) shows the percentage of articles con-

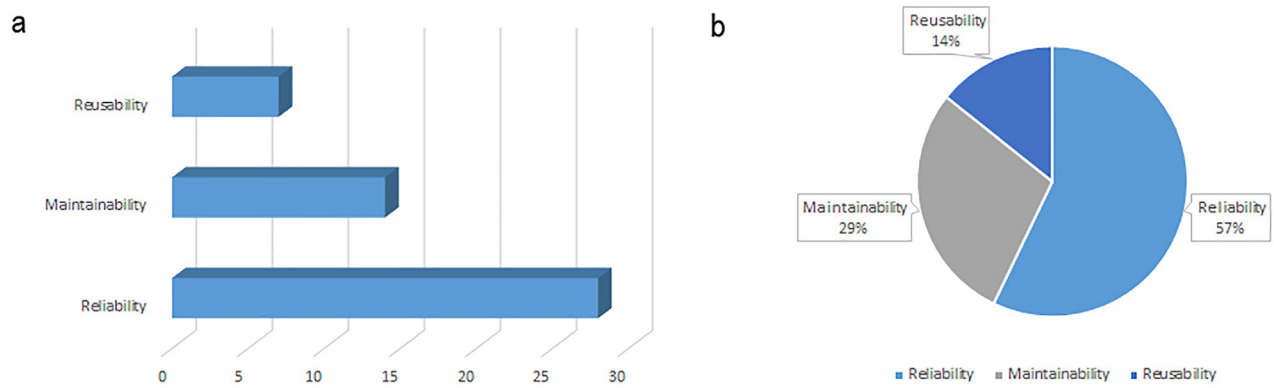

Fig. 2. (a) Number of articles intended to predict each attribute; (b) Percentage of articles intended to predict each software attributes.

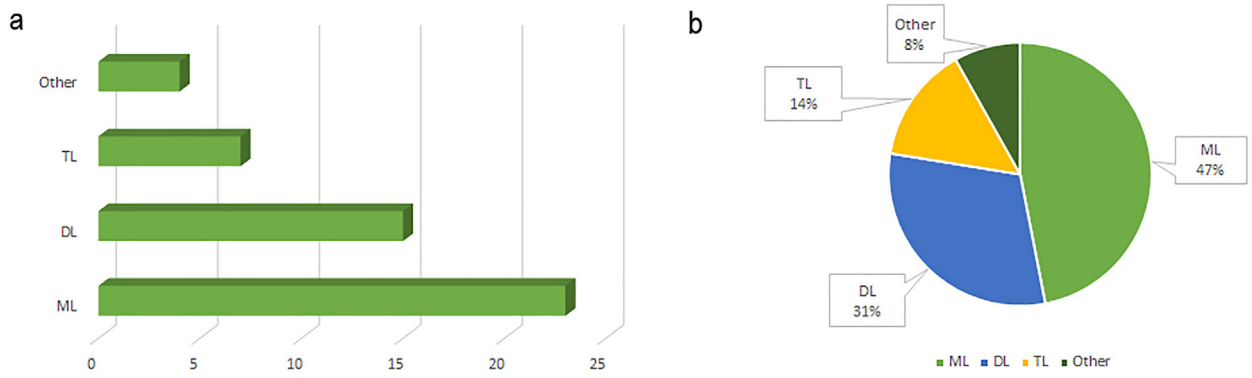

Fig. 3. (a) Number of articles used each technique; (b) Percentage of articles used each technique.

ducted to predict each software attribute. On the other hand, Fig. 3 (a) shows the number of articles conducted using different learning techniques. Fig. 3 (b) shows the percentage of articles conducted using the different learning techniques.

5. Conclusion and future works

The tasks of verifying and predicting the quality of the software need expert developers. Quality assurance can only be done after obtaining experience that helps in predicting possible error spots or defects. We find that artificial intelligence algorithms in general and machine learning, in particular, played their required role after quickly gaining experience through training on previously defective software. There is still a need to develop more comprehensive models to verify all quality specifications of the software. This work made a contribution that helps researchers in the future to see and choose the right way to determine their priorities and activities when working in the field of automated software quality prediction. In future works, a comparison can be made between the capabilities of each of the machine learning techniques. This requires finding a unified approach for making a fair and accurate comparison, as data type, software metrics, selected features, evaluation criteria, and techniques are different between research.

CRediT authorship contribution statement

Feisal Alaswad: Conceptualization, Methodology, Writing – review & editing, Validation. **E. Poovammal:** Supervision, Conceptualization, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Z. Wan, X. Xia, D. Lo, G.C. Murphy, How does Machine Learning Change Software Development Practices?, IEEE Trans Software Eng. (2019) 1, https://doi.org/10.1109/TSE.2019.2937083.
[2] Y. Mahmood, N. Kama, A. Azmi, A.S. Khan, M. Ali, Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation, Softw. Pract. Experience (2021), https://doi.org/10.1002/spe.3009.
[3] Q. Wang, X. Ji, Importance decision analysis on software testing design, J. Comput. Appl. 31 (2012) 1495–1497, https://doi.org/10.3724/sp.j.1087.2011.01495.
[4] Cowlessur S, Pattnaik S, Pattanayak B. A review of machine learning techniques for software quality prediction, Advanced Computing and Intelligent Engineering, Springer, Singapore 2020, p. 537–49. 10.1007/978-981-15-1483-8_45.
[5] [5] Eisty, Nasir U, Thiruvathukal GK, Carver JC. A survey of software metric use in research software development, IEEE 14th International Conference on e-Science 2018, p. 212–22. 10.1109/eScience.2018.00036.
[6] N. Fenton, J. Bieman, Software metrics: A rigorous and practical approach, third edition, 3rd ed., CRC Press, Inc., 2014. https://www.routledge.com/Software-Metrics-A-Rigorous-and-Practical-Approach-Third-Edition/Fenton-Bieman/p/book/9780367659028.
[7] Reddivari S, Raman J. Software quality prediction: An investigation based on machine learning, IEEE 20th International Conference on Information Reuse and Integration for Data Science 2019, p. 115–22. 10.1109/IRI.2019.00030.
[8] N. Medeiros, N. Ivaki, P. Costa, M. Vieira, Vulnerable Code Detection Using Software Metrics and Machine Learning, IEEE Access 8 (2020) 219174–219198, https://doi.org/10.1109/access.2020.3041181.
[9] S. Bouktif, F. Ahmed, I. Khalil, G. Antoniol, A novel composite model approach to improve software quality prediction, Inf. Softw. Technol. 52 (2010) 1298–1311, https://doi.org/10.1016/j.infsof.2010.07.003.

[10] A. Okutan, O.T. Yıldız, Software defect prediction using Bayesian networks, Empirical Softw. Eng. 19 (2012) 154–181, https://doi.org/10.1007/s10664-012-9218-8.
[11] S. Jha, R. Kumar, L. Hoang Son, M. Abdel-Basset, I. Priyadarshini, R. Sharma, et al., Deep Learning Approach for Software Maintainability Metrics Prediction, IEEE Access 7 (2019) 61840–61855, https://doi.org/10.1109/access.2019.2913349.
[12] K. Sheoran, P. Tomar, R. Mishra, Software Quality Prediction Model with the Aid of Advanced Neural Network with HCS, Procedia Comput. Sci. 92 (2016) 418–424, https://doi.org/10.1016/j.procs.2016.07.365.
[13] Y. Ma, G. Luo, X. Zeng, A. Chen, Transfer learning for cross-company software defect prediction, Inf. Softw. Technol. 54 (2012) 248–256, https://doi.org/10.1016/j.infsof.2011.09.007.
[14] E. Rashid, S. Patnaik, V. Bhattacherjee, Software Quality Estimation using Machine Learning: Case-based Reasoning Technique, Int. J. Comput. Appl. 58 (2012) 43–48, https://doi.org/10.5120/9354-3687.
[15] F. Yucalar, A. Ozcift, E. Borandag, D. Kilinc, Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability, Eng. Sci. Technol. Int. J. 23 (2020) 938–950, https://doi.org/10.1016/j.jestch.2019.10.005.
[16] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, A. Bener, Defect prediction from static code features: current results, limitations, new approaches, Automated Softw. Eng. 17 (2010) 375–407, https://doi.org/10.1007/s10515-010-0069-5.
[17] R. Malhotra, A. Jain, Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality, J. Inf. Process. Syst. 8 (2012) 241–262, https://doi.org/10.3745/jips.2012.8.2.241.
[18] N. Bouguila, J.H. Wang, H.A. Ben, A bayesian approach for software quality prediction, IEEE 2 (2008) 11–49, https://doi.org/10.1109/IS.2008.4670508.
[19] Malhotra R, Chug A. Software maintainability prediction using machine learning algorithms. Software Engineering: An International Journal (SeiJ) 2012;2.
[20] A. Shri, P.S. Sandhu, V. Gupta, S. Anand, Prediction of reusability of object-oriented software systems using clustering approach, Int. J. Comput. Inf. Eng. 43 (2010) 853–856, https://doi.org/10.5281/zenodo.1074887.
[21] Mochammad Abdul Azis, Imam Nawawi, A. Fauzi, Ahmad Hafidzul Ginabila, Hamid A. Kahfi, Maintainability prediction in eclipse mylyn software program code using mamdani's fuzzy inference system approach, Jurnal Mantik (2021), https://doi.org/10.35335/mantik.
[22] K. Jeet, N. Bhatia, R.S. Minhas, A bayesian network based approach for software defects prediction, ACM SIGSOFT Softw. Eng. Notes 36 (2011) 1, https://doi.org/10.1145/1988997.1989017.
[23] A. Kumar, Measuring Software reusability using SVM based classifier approach, Int. J. Inf. Technol. Knowl. Manage. 5 (2012) 205–209. http://www.csjournals.com/IJITKM/PDF%205-1/Article_41.pdf.
[24] D. Bowes, T. Hall, J. Petrić, Software defect prediction: do different classifiers find the same defects?, Software Qual J. 26 (2017) 525–552, https://doi.org/10.1007/s11219-016-9353-3.
[25] K. Lakra, A. Chug, Development of efficient and optimal models for software maintainability prediction using feature selection techniques, IEEE (2021) 798–803, https://doi.org/10.1109/INDIACom51348.2021.00143.
[26] Negi P, Umesh Kumar Tiwari. Machine learning algorithm for assessing reusability in component based software development. EasyChair Preprint; 2020.
[27] S. Gupta, A. Chug, Assessing Cross-Project Technique for Software Maintainability Prediction, Procedia Comput. Sci. 167 (2020) 656–665, https://doi.org/10.1016/j.procs.2020.03.332.
[28] L. Kumar, M. Kumar, S.Ku Rath, Maintainability prediction of web service using support vector machine with various kernel methods, Int. J. Syst. Assurance Eng. Manage. 8 (2016) 205–222, https://doi.org/10.1007/s13198-016-0415-5.
[29] H. Alsolai, M. Roper, D. Nassar, Predicting software maintainability in object-oriented systems using ensemble techniques, in: International Conference on Software Maintenance (ICSM), 2018, pp. 716–721, https://doi.org/10.1109/ICSME.2018.00088.
[30] H. Sinha, R.K. Behera, Supervised machine learning approach to predict qualitative software product, Evol. Intel. 14 (2020), https://doi.org/10.1007/s12065-020-00434-4.
[31] S. Kassaymeh, S. Abdullah, M.A. Al-Betar, M. Alweshah, Salp swarm optimizer for modeling the software fault prediction problem, J. King Saud Univ. Comput. Inf. Sci. (2021), https://doi.org/10.1016/j.jksuci.2021.01.015.
[32] D.-L. Miholca, G. Czibula, V. Tomescu, COMET: A conceptual coupling based metrics suite for software defect prediction, Procedia Comput. Sci. 176 (2020) 31–40, https://doi.org/10.1016/j.procs.2020.08.004.
[33] L. Qiao, X. Li, Q. Umer, P. Guo, Deep learning based software defect prediction, Neurocomputing 385 (2020) 100–110, https://doi.org/10.1016/j.neucom.2019.11.067.
[34] S. Gupta, A. Chug, An optimized extreme learning machine algorithm for improving software maintainability prediction, International Conference on Confluence The Next Generation Information Technology Summit (2021) 829–836, https://doi.org/10.1109/Confluence51648.2021.9377196.
[35] A. Hudaib, A. Huneiti, I. Othman, Software Reusability Classification and Predication Using Self-Organizing Map (SOM), Communications and Network 08 (2016) 179–192, https://doi.org/10.4236/cn.2016.83018.
[36] L. Kumar, S.Ku Rath, Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software, J. Syst. Softw. 121 (2016) 170–190, https://doi.org/10.1016/j.jss.2016.01.003.
[37] D. Manjhi, A. Chaturvedi, Reuse estimate and interval prediction using MOGA-NN and RBF-NN in the functional paradigm, Sci. Comput. Program. 208 (2021), https://doi.org/10.1016/j.scico.2021.102643.
[38] L. Kumar, S.K. Rath, Software maintainability prediction using hybrid neural network and fuzzy logic approach with parallel computing concept, Int. J. Syst. Assurance Eng. Manage. 8 (2017) 1487–1502, https://doi.org/10.1007/s13198-017-0618-4.
[39] N. Padhy, R.P. Singh, S.C. Satapathy, Cost-effective and fault-resilient reusability prediction model by using adaptive genetic algorithm based neural network for web-of-service applications, Cluster Comput. 22 (2018) 14559–14581, https://doi.org/10.1007/s10586-018-2359-9.
[40] J. Zheng, Cost-sensitive boosting neural networks for software defect prediction, Expert Syst. Appl. 37 (2010) 4537–4543, https://doi.org/10.1016/j.eswa.2010.12.056.
[41] K. Sheoran, P. Tomar, R. Mishra, A novel quality prediction model for component based software system using ACO–NM optimized extreme learning machine, Cogn. Neurodyn. 14 (2020), https://doi.org/10.1007/s11571-020-09585-7.
[42] [42] Jindal R, Malhotra R, Jain A. Software defect prediction using neural networks, Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization 2014, p. 1–6. 10.1109/ICRITO.2014.7014673 .
[43] Qing H, Biwen L, Beijun S, Xia Y. Cross-project software defect prediction using feature-based transfer learning, International Conference Proceeding Series, Wuhan, China: Association for Computing Machinery; 2015, p. 74-82. 10.1145/2875913.2875944.
[44] J. Chen, K. Hu, Y. Yu, Z. Chen, Q. Xuan, Y. Liu, et al., Software visualization and deep transfer learning for effective software defect prediction, International Conference on Software Engineering (2020) 578–589, https://doi.org/10.1145/3377811.3380389.
[45] X. Du, Z. Zhou, B. Yin, G. Xiao, Cross-project bug type prediction based on transfer learning, Software Qual. J. 28 (2019) 39–57, https://doi.org/10.1007/s11219-019-09467-0.
[46] Cao Q, Sun Q, Cao Q, Tan H. Software defect prediction via transfer learning based neural network, 2015, p. 1–10. 10.1109/ICRSE.2015.7366475.
[47] [47] Nam J, Pan SJ, Kim S. Transfer defect learning. International Conference on Software Engineering, 2013. First International Conference on Reliability Systems Engineering p. 382–91. 10.1109/ICSE.2013.6606584.
[48] A. Amin, L. Grunske, A. Colman, An approach to software reliability prediction based on time series modeling, J. Syst. Softw. 86 (2013) 1923–1932, https://doi.org/10.1016/j.jss.2013.03.045.
[49] B. Singh, Kannojia S. Prasad, A Model for Software Product Quality Prediction, J. Softw. Eng. Appl. 05 (2012) 395–401, https://doi.org/10.4236/jsea.2012.56046.
[50] Y. Shi, M. Li, S. Arndt, C. Smidts, Metric-based software reliability prediction approach and its application, Empirical Softw. Eng. 22 (2016) 1579–1633, https://doi.org/10.1007/s10664-016-9425-9.