DOI: http://dx.doi.org/10.26483/ijarcs.v13i6.6918

Volume 13, No. 6, November-December 2022

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

SOFTWARE QUALITY PREDICTION USING MACHINE LEARNING TECHNIQUES AND SOURCE CODE METRICS: A REVIEW

Santosh Saklani
Department of Computer Science
Himachal Pradesh University
Shimla, India

Dr. Anshul Kalia
Department of Computer Science
Himachal Pradesh University
Shimla, India

Dr. Sumesh Sood
Department of Computer Science
Himachal Pradesh University
Shimla, India

Kritika Kumari
Department of Computer Science
Himachal Pradesh University
Shimla, India

Abstract: Software quality prediction is the Machine Learning (ML) based technique in which ML models are trained using historical data. Output from these quality models can be used by software experts in the early phase of software development for improving the quality of software by controlling the various quality attributes like maintainability, reliability, security issues of software etc. In this study a systematic review of studies from 2005 to 2021 is performed. Studies that use ML techniques and source code metrics for Software Quality Prediction (SQP) are included for review. Study assesses the commonly used machine learning techniques and source code metric for SQP. Commonly used datasets, feature selection techniques and commonly used performance measures in software quality prediction are also assessed. In this paper 53 primary studies are selected for systematic review. Results of this study prove that Bayesian Learning (BL), Regression, Ensemble Learning (EL), Decision Tree (DT) and Support Vector Machine (SVM) are most commonly ML techniques used for quality prediction which comprises 58%, 52%, 41%, 32%, and 32% of the overall studies respectively. It is also assessed that NASA, PROMISE, Apache, Mozilla Firefox and Eclipse are the most commonly used datasets for training and testing the SQP models. LOC, CC, CBO, RFC, WMC, LCOM, DIT and NOC are among the most commonly used source code metrics in SQP. Based on the results from the selected studies it is concluded that ML techniques and source code metrics have the ability to improve the overall quality of the software.

Keywords: machine learning, software quality prediction, software vulnerabilities, source code metrics

I. INTRODUCTION

Dependency on software is increasing rapidly in various fields. Therefore, delivering quality software to user is very important. Quality software must be bug or defect free. ISO/IEC 9126-1 defines a quality model that comprises six characteristics i.e., functionality, reliability, usability, efficiency, maintainability and portability with 27 sub characteristics of software product quality [1]. To measure these characteristics, software metrics are used. Software metric is a standard of degree to which system or process have some properties. Software metric provides quantitative values to the attributes involved in software development process [2]. Source code metrics are the metrics extracted from some source code, and their value help developer to measure the quality attribute. Predicting various properties and sub properties of software quality like defects, vulnerability, change-proneness, maintainability, testability and complexity are very important and essential activities in order to improve software quality and reduce maintenance effort. Assessing of important quality characteristics in the early phases of software development help in reducing effort, time and money [3].

In this study a systematic review on the use of ML techniques and source code metrics in software quality prediction (SQP) is done. Software quality is determined by a set of quality factors [4]. Here in this study SQP means predicting the quality attributes/factors which includes defects, vulnerabilities, change-proneness, maintainability, complexity and testability of software. For example, software defect prediction can be done by classifying the modulus/classes as fault prone. Software defects and source code metrics obtained from similar projects or previous release can be used to construct software defect prediction model. In the same way models for predicting software vulnerabilities, software change proneness, maintainability, software complexity, software testability may be developed by using historical data and source code metrics to improve the software overall quality.

The rest of the paper is organized as follows: Section 2 provides the research questions which are addressed in this review and inclusion & exclusion criteria applied for the selection process of the studies. Section 3 presents results of the study and discussion. Limitation of this work is provided in section 4. Section 5 provides conclusions obtained from the study and future directions.

II. METHOD

Design of systematic review carried out in this paper is inspired from [5]and 0.2[6]. The work in this study is divided into three steps: planning, conducting, and reporting the review. Planning is done in the first step which include identifying the need of systematic review, identifying the research questions, and review protocol. Second step is conducting the review which includes criteria for selecting the studies, quality assessment criteria, and data extraction and synthesis.

A. Planning the review

Planning is the first step to perform any important task. Steps under the planning phase are described as below.

1) Identification of need

Machine Learning (ML) techniques have been used for developing the prediction model in various fields and software engineering is no exception. Source code metrics are very important to measure the various process and attributes of software development process. Source code metrics using ML techniques have been proved important in predicting fault, vulnerabilities, maintainability change-proneness and complexity. It is very important to establish the state of art of ML techniques and source code metrics used for predicting the various quality attributes of software by gathering the finding from current research. By extracting, synthesis the data and finding from existing empirical studies the current trends of predicting various quality features/characteristics of software using ML techniques and source code metrics can be obtained.

2) Research questions

Research questions which are addressed in this study are presented in the table below.

Table I: Research Questions

ID	Research question	Motivation
RQ1	Which ML techniques have been used for training software quality prediction (SQP) models using source code metrics?	Identification of ML techniques commonly used for developing SQP model.
RQ2	Which source code metrics are commonly used for SQP using ML techniques?	Identification of source code metrics commonly used for SQP.
RQ3	What are the datasets used for SQP?	Identify the commonly used data set for SQP.
RQ4	Which techniques are commonly used for feature reduction?	identify the commonly used feature reduction techniques for SQP
RQ5	Which performance measures are used for SQP?	Assessment of performance measures used for measuring the performance of different ML models for SQP.
RQ6	Which Programming languages are currently used in developing SQP model?	Identify the programming languages commonly used for SQP.

3) Review protocol

In the review protocol section, the search design is described which includes search scope in terms of time period, electronic databases, and overall search strategy.

Time period: In this review empirical study has been selected from the years 2005 to 2021.

Electronic databases: The following five electronic databases have been selected as a primary source for this review: IEEE Xplore, ScienceDirect, ACM Digital Library, Wiley Online Library, Google scholar. These electronic databases provide a good source for journal/event papers.

Search strategy: Initial search is made to identify the primary studies in which source code metrics and ML techniques have been used for software quality prediction (SQP). After performing an initial search relevant studies are determined by following the inclusion and exclusion criteria described as below.

Inclusion criteria:

- Empirical studies using source code metrics and ML techniques for predicting software defects.
- Empirical studies using source code metrics and ML techniques for predicting software vulnerabilities.
- Empirical studies using source code metrics and ML techniques for predicting change-proneness in software.
- Empirical studies using source code metrics and ML techniques for predicting software maintainability, testing and complexity.
- Empirical studies from reputed publishers with good citations.

Exclusion criteria:

- Studies which do not include source code metrics and ML techniques.
- Studies using source code metrics and ML techniques in context other than predicting SQP (Which include prediction of software defects, vulnerabilities, change-proneness, maintainability, testing and complexity).
- Review studies.

After applying above inclusion and exclusion criteria 57 studies have been selected. The final selection is done by following the quality assessment criteria (QAC) described in the next section.

B. Conducting the review

This section defines quality assessment criteria, selection of primary studies, and data extraction and synthesis.

1) Quality assessment criteria

In this section quality questions are formed to check the quality and relevance of all the 57 primary studies (selected after following inclusion/exclusion criteria). Following questions are answered to measure the quality of selected studies:

- Q1. Are the aims or objectives of research clearly stated?
- Q2. Is the study peer reviewed?
- Q3. Are the independent and dependent variables are clearly defined?
- Q4. Are source code metrics used are clearly defined?
- Q5. Are ML techniques used in the study clearly defined?
- Q6. Are the results and findings obtained are clearly mentioned?
- Q7. Are the tools and datasets used clearly defined?

2) Selection of primary studies

In total 53 primary studies have been selected after following quality assessment criteria mentioned in the above section. Each study uses source code metrics and ML techniques for SQP.

Table II: Selected Primary Studies

Category	Study No.	Paper	Category	Study No.	Paper
Defect Prediction	S1	[7]	Defect Prediction	S28	[34]
	S2	[8]		S29	[35]
	S3	[9]		S30	[36]
	S4	[10]	Software Vulnerability Prediction	S31	[37]
	S5	[11]		S32	[38]
	S6	[12]		S33	[39]
	S7	[13]		S34	[40]
	S8	[14]		S35	[41]
	S9	[15]		S36	[42]
	S10	[16]		S37	[43]
	S11	[17]		S38	[44]
	S12	[18]		S39	[45]
	S13	[19]		S40	[46]
	S14	[20]		S41	[47]
	S15	[21]	Software Change Proneness	S42	[48]
	S16	[22]		S43	[49]
	S17	[23]		S44	[50]
	S18	[24]		S45	[51]
	S19	[25]		S46	[52]
	S20	[26]		S47	[53]
	S21	[27]		S48	[54]
	S22	[28]		S49	[55]
	S23	[29]		S50	[56]

	S24	[30]	Software Testing	S51	[57]
	S25	[31]	Software Maintainability	S52	[58]
	S26	[32]	Software Complexity	S53	[59]
	S27	[33]			

III. RESULT AND DISCUSSIONS

This section presents results obtained from the selected studies.

A. Description of primary studies

In this section we provide description of selected primary studies. This section includes publication source and publication year of all the primary studies selected for review.

1) Publication

The details of publisher for journals and events are shown in table III (a) and table III (b) respectively. The selected studies are published across 28 journals and 25 different events. Table III (c) presents the overall percentage of different publishers

Table III (a): Publisher-wise distribution of journals

Name of Publisher	Journal (s)
Elsevier	13
Springer	6
Wiley Online Library	2
ACM	2
IEEE	1
Korean Science	1
World Scientific	1
Blu Eye Intelligence Engineering & Science Publication	1
SAI Organization	1
Hindawi	1

Table III (b): Publisher-wise distribution of events

Name of Publisher	Event (s)
Elsevier	14
ACM	7
Springer	2
IET	1

Table III (c): Percentage distributions of studies

Name of Publisher	Percentage
IEEE	28
Elsevier	24
ACM	17
Springer	15
Wiley Online Library	4
Korean Science	2
World Scientific	2
Blu Eye Intelligence Engineering & Science Publication	2
IET	2
SAI Organization	2
Hindawi	2

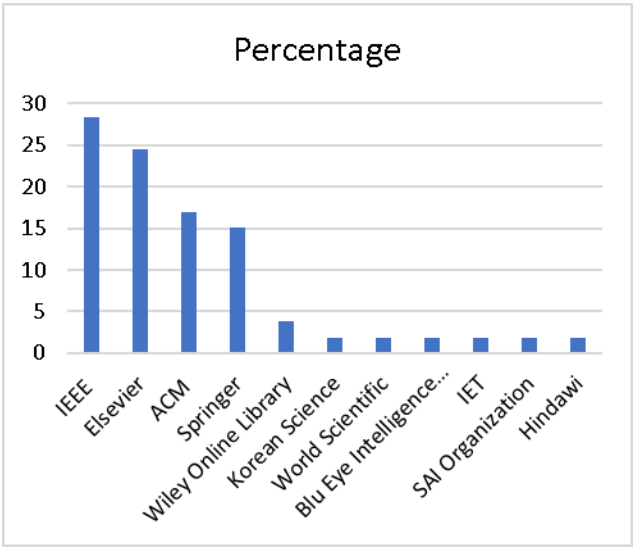


Figure 1: Bar chart for percentage distributions of studies

2) Publication year

Figure 2 presents the distributions of studies from 2005 to 2021. It shows that from the year 2017 the highest number of studies are included followed by 2021 and 2015. The number of studies in the years 2015, 2016, 2017, 2018,

2019, 2020 and 2021 are 6, 3, 7, 3, 3, 4 and 6 respectively accounting for 60% of the studies. This shows that preference is given to studies published in recent years Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

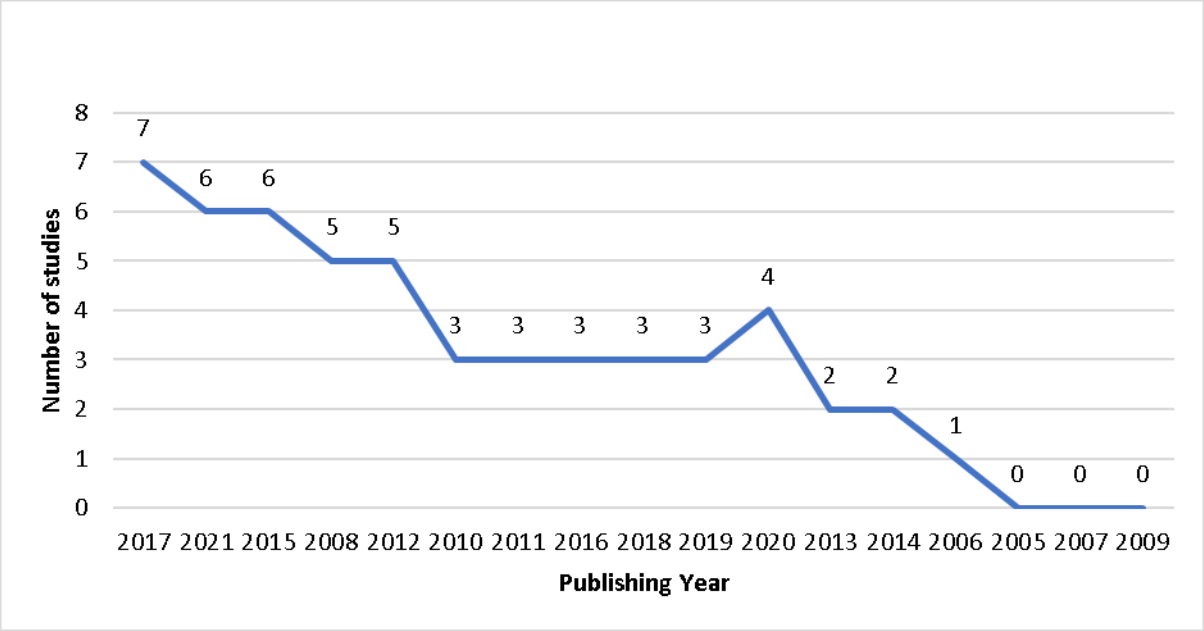


Figure 2: Year-wise distribution of studies

B. RQ1: Which ML techniques have been used for training software quality prediction (SQP) models using source code metrics?

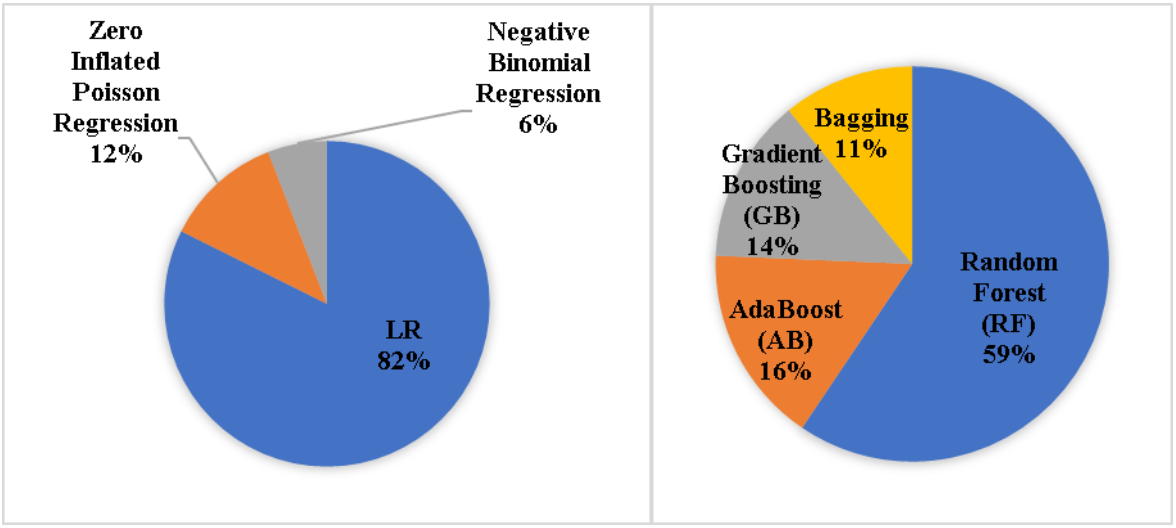
In this section details of ML techniques used in selected studies for SQP are presented. In this study the ML techniques used for SQP are classified in the following categories:

- Bayesian Learning (BL)
- Regressions
- Ensemble Learning (EL)
- Decision Trees (DT)
- Support Vector Machines (SVM)
- Neural Networks (NN)
- Clustering
- Rule Based Learning (RBL)

Table IV: Classification of ML techniques for SQP

Category	Method (s)	Number of Occurrence
Bayesian Learning (BL)	Naive Bayes (NB)	31
Regressions	Linear Regression (LR)	28
	Zero Inflated Poisson Regression	4
	Negative Binomial Regression	2
Ensemble Learning (EL)	Random Forest (RF)	22
	AdaBoost (AB)	6
	Gradient Boosting (GB)	5
	Bagging	4
Decision Trees (DT)	J48	10
	Quad Based Tree	3
	Alternating Decision Tree (ADT)	2
	Classification and Regression Tree (CART)	2
	C4.5	1
	Logistic Model Tree	1

	AdTree	1
Support Vector Machines (SVM)	Support Vector Machines (SVM)	17
Neural Networks (NN)	Artificial Neural Network (ANN)	9
	Multilayer Perceptron (MLP)	6
	Biological Neural Network (BNN)	4
	Radial Basis Function (RBF)	4
	Self-Organizing Maps (SOM)	1
	Recurrent Neural Network (RNN)	1
Clustering	K- mean	3
	Hierarchal Clustering (HC)	2
	Make Density Based Cluster (MDBC)	2
Rule Based Learning (RBL)	OneR	2
	Neighbor With Generalization (Nnge)	1
Miscellaneous	IBK	3
	K- Nearest Neighbor (KNN)	3
	DeepJIT	2
	Kstar	1
	CC2Vec	1
	EARL	1



(a) (b)

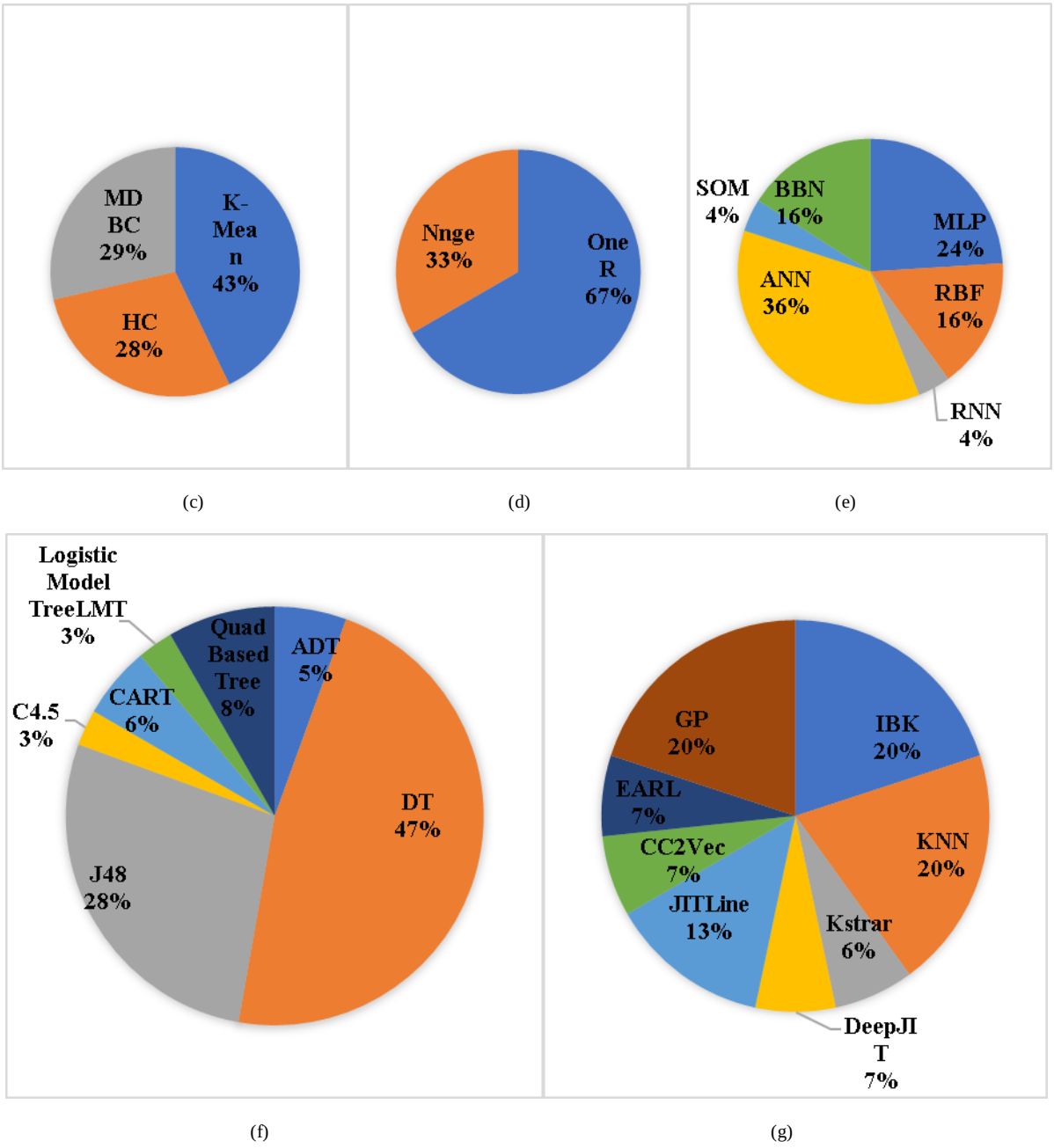


Figure 3: Distribution of sub categories in (a) Regressions (b) Ensemble Learning (c) Clustering (d) Rule Based Learning (e) Neural Network (f) DT (g) Miscellaneous

Table V presents both the number and percentage of the studies with respect to ML techniques. Table V shows that among the different categories of ML techniques most frequently used techniques are from the categories BL, Regression, EL, DT, and SVM which covers 58.49%, 52.8%, 41.5%, 32% and 32% of studies, respectively.

Table V: Distribution of studies across ML techniques based on classification

Machine Learning Method	Number of Studies	Percent
Bayesian Learning (BL)	31	58.49
Regression	28	52.8
Ensemble Learning (EL)	22	41.5
Decision Tree (DT)	17	32
Support Vector Machine (SVM)	17	32
Miscellaneous	15	28.03
Neural Network (NN)	10	18.86
Clustering	5	9.4
Rule Based Learning (RBL)	2	3.7

C. RQ2: Which source code metrics are commonly used for SQP using ML techniques?

Number of source code metrics are used for quantifying the characteristics of software. Table VI, VII, VIII and IX define various types of source code metrics used under different categories in selected primary studies. Table VI presents source code metrics used for defect prediction. Object oriented, complexity, size and Halstead's metrics are the commonly used metrics in this category

Table VI: Software metrics used in software defect prediction

Category	Metric Type	Studies Reference	List of metrics
Defect Prediction	1.Complexity metrics (15)	S2, S3, S6, S7, S9, S11, S13, S17, S19, S21, S23, S24, S25, S26, S27	CC, Essential Complexity, Max. CC, Avg. CC, Cyclomatic Density, Design Complexity, AMC, SDMC
	2. Halstead (9)	S2, S3, S7, S8, S9, S19, S23, S26, S27	N, V, D, I, E, B, L, T
	3. Size (16)	S1, S2, S3, S4, S5, S6, S7, S8, S9, S17, S19, S21, S22, S23, S25, S26	LOC, CLOC, BLOC, CCLOC, UNOD, UNOT, NOD, NOT, LOC_Blank, LOC_Comment, LOC_Execute, Branch_Count, Decision_Count
	4. Object Oriented (13)	S1, S4, S5, S6, S10, S11, S12, S13, S20, S23, S24, S25, S34	CBO, LCOM, LOC, MOA, NOM, RFC, CA, CE, DAM, SRFC, CAM, DIT, NOC, IC, CBM, MFA, WMC
	5. Others	S6, S28	Conditional Expression, ContinueStatement, DoStatement, FieldAccess, Javadoc, LabeledStatement, ParenthesizedExpression, PrefixExpression, QualifiedName, ReturnStatement, SuperMethodInvocation, SwitchStatement, ThisExpression, ThrowStatement, CommitLevel metrics

Table VII: Software metrics for software vulnerability prediction

Category	Metric Type	Studies Reference	List of metrics
Vulnerability Prediction	Complexity (7)	S32, S33, S34, S35, S36, S38, S39	Cyclomatic, CyclomaticModified, CyclomaticStrict, Essential Cyclomatic Complexity, Nesting Complexity

	CountLine (6)	S32, S33, S35, S36, S38, S41	AltCountLineComment, CountLineComment, CountInput, CountOutput, CountPath, CountLine, CountLineCode, CountLineInactive, CountLinePreprocessor, CountSemicolon, CountStmt, CountStmtExe, CountStmtDecl, CountStmtEmpty
	Coupling & cohesion (5)	S31, S32, S37, S38, S39	SumFanIn, SumFanOut, MaxFanIn, MaxFanOut, HK, WMC, DIT, NOC, CBC, RFC, CBO, LCOM
	Comments (4)	S31, S37, S38, S41	AltCountLineComment, CountLineComment, RatioCommentToCodE
	Halstead (1)	S31	Halstead Volume

Table VIII: metric classification for software change proneness prediction

Category	Metric Type	Studies Reference	List of metrics
Change Proneness	Object Oriented (9)	S42, S43, S44, S45, S46, S47, S48, S49, S50	WMC, DIT, CBO, RFC, Ca, NPM, MOA, CAM, IC, CBM, AMC only WMC, CBO, CA, NPM, CAM, IC, CBM, AMC, LCOM
	Size Metrics (4)	S45, S46, S47, S49	LOC

Table IX: metrics classification for software maintainability and complexity prediction

Categories	Metric Type	Studies Reference	List of metrics
software maintainability and complexity	Object Oriented	S52, S53	DIT, RFC, WMC, LOC, CBO, DIT, NOC, NOM, LCOM,

Table VII shows, that in the selected studies complexity, count, coupling & cohesion, and comment metrics are commonly used source code metrics for software vulnerability prediction. Table VIII shows the metrics used to predict software change proneness. It is clear from the table that object-oriented metrics and size metrics are the commonly used metric for predicting software change proneness. Metrics used for predicting software maintainability and software complexity are listed in table IX.

Table X: Popular metrics used in selected studies.

Metric	Total Occurrence
Cyclomatic Complexity (CC)	25
Line of Code (LOC)	21
Coupling Between Objects (CBO)	21
Response for a Class (RFC)	21
Weighted Methods per Class (WMC)	20
Lack of Cohesion in Methods (LCOM)	17
Depth of Inheritance Tree (DIT)	16
Number of Children (NOC)	16
Average Method Complexity (AMC)	14
Cohesion Among Methods (CAM)	13
Halstead Metrics (N, V, D, I, E, B, L, T)	10
IC	10

Table X presents popular metrics used in overall selected studies with their total occurrence

D. RQ3: What are the datasets used for SQP?

This section analyzes the datasets used for SQP. Table XI and figure 4 shows all the datasets used in different prediction models (for software defect, vulnerability, change-proneness, maintenance, testing and complexity prediction) used in this study. Major datasets used are as follow:

- NASA datasets: NASA's Datasets are freely available in NASA repository. 26% of our selected studies used NASA datasets. Table XI shows that they are the most common used datasets for SQP

- PROMISE datasets: these datasets are available in PROMISE repository.15% of our selected primary studies used PROMISE's datasets.
- Apache datasets: these datasets are used in 13% of the selected studies.
- Mozilla Firefox: used in 11% of the selected studies.
- Eclipse: used in 9% of the selected studies.
- Turkish and Stanford datasets: both are used in equal proportion that is 9% of each.
- Other open-source projects: 20% of the selected studies use datasets from other open-source projects.

Table XI: Classification of Datasets for SQP

Datasets Type	Number of studies	Percentage of Studies
NASA	14	26
PROMISE	8	15
Apache	7	13
Mozilla Firefox	6	11
Eclipse	5	9
Turkish Data Set	2	3
Stanford	2	3
Other open-source projects	11	20

Figure 4 present datasets for SQP used by the selected primary studies.

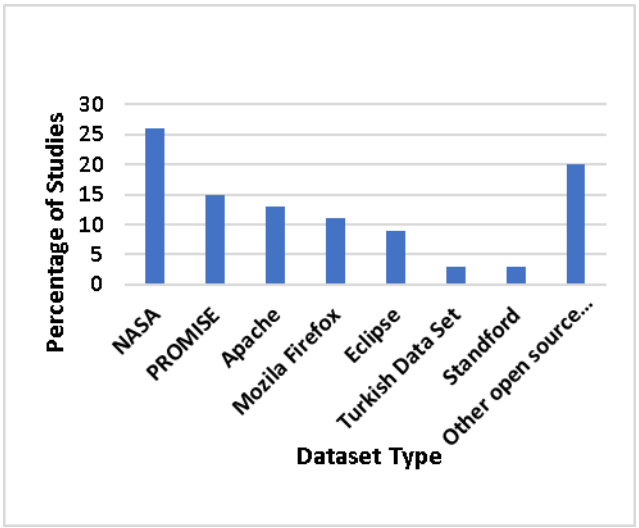


Figure 4: datasets used for SQP

1) Description of datasets used for Software Vulnerability Prediction (SVP)

Security is one of the important characteristics to measure the quality of software. Software vulnerabilities should be handled carefully to avoid non-compensable damage to the system. ML techniques are playing an important role in predicting software vulnerabilities. Description of datasets used for SVP in the selected primary studies (S31, S32, S33, S34, S35, S36, S37, S38, S39, S40, and S41) is given in table XII.

Mozilla: Vulnerabilities dataset of Mozilla Firefox is used in 54% of the selected studies.

Apache: Apache Tomcat and Apache CXF accounts for 18% of the selected studies for SVP. These datasets are freely available.

Ellipse, Stanford, and NIST: datasets from Ellipse, Stanford and NIST repository are used in equal proportion that is 9% of each for SVP.

Other open-source project: 27% of the selected studies for SVP used open-source project like github, glib, Linux kernel and other web application dataset.

Table XII: Datasets used in SVP

Dataset Type	Percentage of Studies
Mozilla	54
Apache	18
Ellipse	9
Stanford	9
NIST	9
Other Open-source Project	27

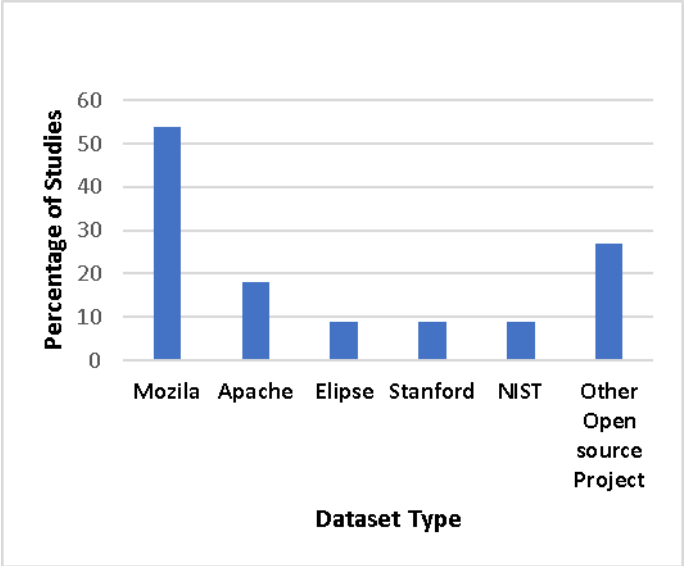


Figure 5: Datasets for SVP

E. RQ4: Which techniques are commonly used for feature reduction?

To reduce the dimensionality of features, number of feature reduction techniques have been used over the years. There are basically two types of feature reduction techniques: feature selection and feature extraction. Feature selection techniques help in selection of the most relevant feature in a dataset while on the other hand, in feature extraction technique a new feature is extracted by combining a set of relevant features. Only 36% of the selected primary studies clearly mentioned the use of feature reduction techniques. The most commonly used feature selection techniques in the selected studies are forward step method (used in S10, S11), correlation-based feature selection (CFS) (used in S9, S19), gain ratio (used in S27, S30), Pearson's correlation (used in S23, S29, S53) and information gain (used in S16, S49).

Some less commonly used feature selection techniques include Spearman's correlation (sued in S21), greedy algorithm (used in S29), chi square (used in S49), fisher's criterion (used in S23), Binary genetic algorithms (used in S13).

There are few studies which used principal component analysis for feature extraction (used by S9, S4).

F. RQ5: Which performance measures are used for SQP?

As shown in table XIII and figure 6 precision, accuracy, ROC, recall, f- measure and specificity are the most commonly used performance measures used in selected studies for SQP. F1 score, absolute error, sensitivity, FP rate, FN rate completeness, relative error, confusion metrics MAE are among the less used performance measure in the selected studies.

Table XIII: Percentage of occurrence of performance measures in selected studies for SQP

Performance Measure	Percentage of Occurrence
Precision	51
Accuracy	45
ROC	37
Recall	32
F-measure	24
Specificity	18
F1 Score	11
absolute error	11
Sensitivity	9
FP rate	7
FN rate	5
Completeness	5
relative error	3
confusion matrix	3
MAE	3

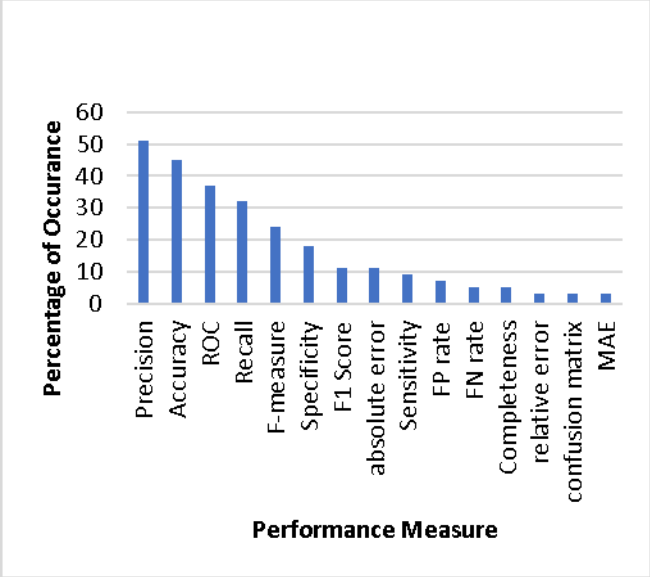


Figure 6: Performance measures used for SQP

G. RQ6: Which Programming languages are currently used in developing SQP models?

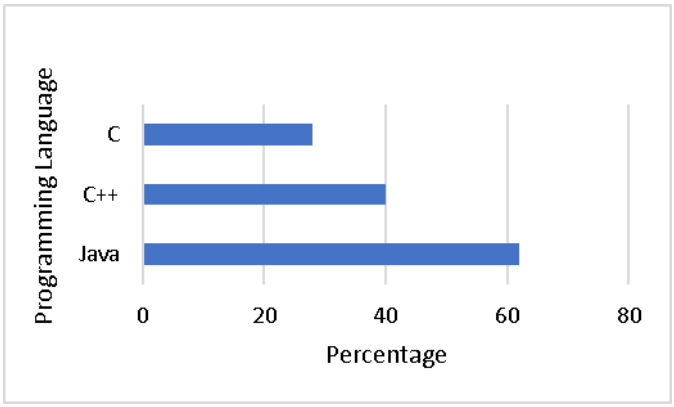


Figure 7: Programming language used for SQP.

Around 66% of the selected studies clearly specify the programming language of source code. Some of the studies uses only one language while some of the studies uses multi language like C/C++, Java/C++, C/C++/Java. In this 66% of the studies 62 % studies uses JAVA, 40% uses C++ and 28 % uses C as a source code language as shown in figure 7. Therefore, from the above facts and figure it is observed that most of the studies use object-oriented programming paradigm.

IV. LIMITATIONS

In this systematic review, a number of primary studies are evaluated to assess the source code metrics and ML techniques for SQP. A limitation in this review is that only studies which include source code metrics and ML techniques for predicting software quality are included. Important studies which include techniques other than source code metrics are excluded in this review. For example, there are other successful methods for predicting vulnerabilities in source code but not includes in this review. Though all the mentioned digital libraries have carefully searched, there still may be the possibility that a suitable study may be left out.

V. CONCLUSION AND FUTURE GUIDELINE

In this study a systematic review is performed in order to analyze and assess the ML techniques, source code metrics, datasets and performance measures used for SQP. In the first step 53 primary studies (2005-2021) are identified to meet research objectives. In the second step data is collected, analyzed and assessed to answer the research questions. In this study ML techniques and source code metrics used for SQP are assessed. Datasets and performance measures used in the selected studies for SQP are also analyzed and assessed.

Main finding obtained from the selected studies are:

- Most commonly used ML techniques for SQP are bayesian learning (BL), regression, ensemble learning (EL), decision tree (DT) and support vector machine (SVM).

- Most frequently used source code metrics in the selected studies are LOC, CC, CBO, RFC, WMC, LCOM, DIT and NOC.
- Most commonly datasets used in the selected studies are: NASA, PROMISE, Apache, Mozilla Firefox and Eclipse.
- Precision, accuracy, ROC, recall and f-measure are among the most commonly used performance measures in SQP.
- Java, C++ and C are the most frequently used programming language by researcher for SQP. In the selected studies only 66% of the studies clearly.
- mentioned about the programming languages in which 62% studies uses JAVA, 40% uses C++ and 28 % uses C as a source code language.

Guideline for researchers and software practitioners for carrying out research work in future on software quality prediction using ML techniques are as following:

(1) More studies should be carried out to predicting quality characteristics like functionality, reliability, usability, efficiency, maintainability and portability using ML techniques.

(2) Security of software is also one of the important sub characteristics of software product quality. More work needed to be done in predicting software vulnerabilities to ensure software security during software development.

VI. REFERENCES

[1] H.W. Jung, SG. Kim and C.S. Chung, "Measuring software product quality: A survey of ISO/IEC 9126," IEEE software, vol. 21, Oct. 2004, pp. 88-92.

[2] T. Honglei, S. Wei and Z. Yanan, "The research on software metrics and software complexity metrics," 2009 International Forum on Computer Science-Technology and Applications, IEEE, vol. 1, Dec. 2009 pp. 131-136.

[3] D. Azar, H. Harmanani and R. Korkmaz, "A hybrid heuristic approach to optimize rule-based software quality estimation models," Information and Software Technology, vol. 51, Sep. 2009, pp. 1365-76.

[4] M. Jørgensen, "Software quality measurement. Advances in engineering software," vol. 30, Dec. 1999, pp. 907-12.

[5] A.S. Nuñez-Varela, H.G. Pérez-Gonzalez, F.E. Martínez-Perez and C.Soubervielle-Montalvo, "Source code metrics: A systematic mapping study," Journal of Systems and Software, vol. 128, Jun. 2017, pp. 164-97.

[6] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Applied Soft Computing, vol. 27, Feb, pp. 504-18.

[7] B. Khan, R. Naseem, M.A. Shah, K. Wakil, A. Khan, M. I. Uddin and M. Mahmoud, "Software defect prediction for healthcare big data: an empirical evaluation of machine learning techniques," Journal of Healthcare Engineering,2021, Mar. 2021.

[8] M. Gayathri and A. Sudha, "Software defect prediction system using multilayer perceptron neural network with data mining," International Journal of Recent Technology and Engineering, vol. 3, May. 2014, pp. 54-59.

[9] S. Agarwal and D. Tomar, "Prediction of software defects using twin support vector machine," 2014 international conference on information systems and computer networks (ISCON), IEEE, Mar. 2014, pp. 128-132.

[10] R. Malhotra, "An empirical framework for defect prediction using machine learning techniques with Android software," Applied Soft Computing, vol. 49, Dec. 2016, pp. 1034-50.

[11] S.S. Rathore and S. Kumar, "A decision tree regression based approach for the number of software faults prediction," ACM SIGSOFT Software Engineering Notes, vol. 41, Feb. 2016, pp. 1-6.

[12] S. S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction," Soft Computing, vol. 21, Dec. 2017, pp. 7417-434.

[13] Y. Jiang, B. Cuki, T. Menzie and N. Bartlow, "Comparing design and code metrics for software quality prediction," Proceedings of the 4th international workshop on Predictor models in software engineering, vol. 12, May. 2008, pp. 11-18.

[14] I. Gondra, "Applying machine learning to software fault-proneness prediction," Journal of Systems and Software, vol. 81, Feb. 2008, pp. 186-95.

[15] V. U. Challagulla, F. B. Bastani, I. L. Yen and R. A. Paul, "Empirical assessment of machine learning based software defect prediction techniques,"International Journal on Artificial Intelligence Tools, vol. 17, Apr. 2008, pp. 389-400.

[16] Singh, Y., Kaur, A., & Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," Software quality journal, vol. 18, 2010, pp. 3-35.

[17] R. Malhotra and A. Jain, "Fault prediction using statistical and machine learning methods for improving software quality," Journal of Information Processing Systems, vol. 8, 2012, pp. 241-262.

[18] A. Janes, M. Scotto, W. Pedrycz, B. Russo, M.Stefanovic and G.Succi, "Identification of defect-prone classes in telecommunication software systems using design metrics," Information sciences, vol. 176, Dec. 2006, pp. 3711-34.

[19] H. Turabieh, M. Mafarja and X. Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction," Expert systems with applications, vol. 122, May. 2019, pp.27-42.

[20] S. S. Rathore and S. Kumar, "Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems," Knowledge-Based Systems, vol. 119, Mar. 2017, pp. 232-56.

[21] A. T. Haouari, L. Souici-Meslati, F. Atil and D. Meslati, "Empirical comparison and evaluation of Artificial Immune Systems in inter-release software fault prediction," Applied Soft Computing, vol. 96, Nov. 2020, pp. 106686. https://doi.org/10.1016/j.asoc.2020.106686

[22] H. Aljamaan and A. Alazba, "Software defect prediction using tree-based ensembles," Proceedings of the 16th ACM international conference on predictive models and data analytics in software engineering, Nov. 2020, pp. 1-10.

[23] P.S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Transactions on knowledge and data engineering, vol. 24, Jul. 2011, pp. 1146-150.

[24] A. Hammouri, M. Hammad, M. Alnabhan and F. Alsarayrah, "Software bug prediction using machine learning approach," International Journal of Advanced Computer Science and Applications, vol. 9, 2018.

[25] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," Journal of Systems and Software, vol. 81, May. 2008, pp. 649-60.

[26] P. Singh and S. Verma, "Empirical investigation of fault prediction capability of object oriented metrics of open source software," 2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE), IEEE, May. 2012, pp. 323-327.

[27] S.S. Rathore and A. Gupta, "Investigating object-oriented design metrics to predict fault-proneness of software modules," 2012 CSI Sixth International Conference on Software Engineering (CONSEG), IEEE, Sep. 2012, pp. 1-10.

[28] G. Abaei, A. Selamat and H. Fujita, "An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction," Knowledge-Based Systems, vol. 74, Jan. 2015 Jan, pp. 28-39.

[29] I. H. Laradji, M. Alshayeb, L. Ghouti, "Software defect prediction using ensemble learning on selected feature," Information and Software Technology, vol. 58, Feb. 2015, pp. 388-402.

[30] Y. Zhou, B. Xu and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," Journal of Systems and Software, vol. 83, Apr. 2010, pp. 660-674.

[31] P. He P, 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, Mar. 2015, pp. 170-90.

[32] A. Chug and S. Dhall, "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm," 2013.

[33] J. Li, P. He, J. Zhu and M.R. Lyu, "Software defect prediction via convolutional neural network," 2017 IEEE international conference on software quality, reliability and security (QRS), IEEE, Jul. 2017, pp. 318-328.

[34] C. Pornprasit C and C.K. Tantithamthavorn, "Jitline: A simpler, better, faster, finer-grained just-in-time defect prediction," 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, May. 2021, pp. 369-379.

[35] S. Mehta and K.S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," Neural Computing and Applications, vol. 33, Aug. 2021, pp. 10551-562.

[36] S.S. Rathore and S. Kumar, "An empirical study of ensemble techniques for software fault prediction," Applied Intelligence, vol. 51, Jun. 2021 pp. 3615-44.

[37] Y. Zhang, D. Lo,X. Xia,B. Xu B, J. Sun and S. Li, Combining software metrics and text features for vulnerable file prediction," 2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS), IEEE, Dec. 2015, pp. 40-49.

[38] I. Chowdhury and M. Zulkernine, "Can complexity, coupling, and cohesion metrics be used as early indicators of vulnerabilities?," Proceedings of the 2010 ACM Symposium on Applied Computing, Mar, 2010, pp. 1963-1969.

[39] H. Alves, B. Fonseca and N. Antunes, "Experimenting machine learning techniques to predict vulnerabilities," 2016 Seventh Latin-American Symposium on Dependable Computing (LADC), IEEE, Oct. 2016, pp. 151-156.

[40] Y. Shin and L. Williams, "An empirical model to predict security vulnerabilities using code complexity metrics," Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, Oct. 2008, pp. 315-317.

[41] K.Z. Sultana, V. Anu, T.Y. Chong, "Using software metrics for predicting vulnerable classes and methods in Java projects: A machine learning approach," Journal of Software: Evolution and Process, vo.l 33, Mar. 2021, e2303.

[42] A. Gupta, B. Suri, V. Kumar and P. Jain, "Extracting rules for vulnerabilities detection with static metrics using machine learning," International Journal of System Assurance Engineering and Management, vol. 12, Feb. 2021, pp. 65-76.

[43] Y. Shin and L. Williams, "An initial study on the use of execution complexity metrics as indicators of software vulnerabilities, Proceedings of the 7th International workshop on software engineering for secure systems, May. 2011, pp. 1-7.

[44] S. Moshtari, A. Sami and M. Azimi, "Using complexity metrics to improve software security," Computer Fraud & Security, vol. 5, May. 2013 May, pp. 8-17.

[45] I. Chowdhury and M. Zulkernine, "Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities," Journal of Systems Architecture, vol. 57, Mar. 2011, pp. 294-313.

[46] H. Perl, S. Dechand, M. Smith, D. Arp, F. Yamaguchi, K. Rieck, S. Fahl and Y. Acar, "Vccfinder: Finding potential vulnerabilities in open-source projects to assist code audits," Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Oct. 2015, pp. 426-437.

[47] J. Ren, Z. Zheng, Q. Liu, Z. Wei and H. Yan, "A buffer overflow prediction approach based on software metrics and machine learning," Security and Communication Networks, Mar. 2019.

[48] L. Kumar, S. K. Rath and A. Sureka, "Empirical analysis on effectiveness of source code metrics for predicting change-proneness," Proceedings of the 10th Innovations in Software Engineering Conference, Feb. 2017, pp. 4-14.

[49] L. Kumar, S. K. Rath and A. Sureka, "Using source code metrics to predict change-prone web services: A case-study on ebay services," 2017 IEEE workshop on machine learning techniques for software quality evaluation (MaLTeSQuE), IEEE, Feb. 2017, pp. 1-7.

[50] D. Romano and M. Pinzger, "Using source code metrics to predict change-prone java interfaces," 27th IEEE international conference on software maintenance (ICSM), IEEE, Sep. 2011, pp. 303-312.

[51] C. Liu, D. Yang, X. Xia, M. Yan M and X. Zhang, "Cross-project change-proneness prediction," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), IEEE, vol. 1, jul. 2018, pp. 64-73.

[52] L. Kumar, S. Lal, A. Goyal and N.B. Murthy, "Change-proneness of object-oriented software using combination of feature selection techniques and ensemble learning techniques," Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference), Feb. 2019, pp. 1-11.

[53] G. Catolino and F. Ferrucci, "Ensemble techniques for software change prediction: A preliminary investigation," In2018 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), IEEE, Mar. 2018, pp. 25-30.

[54] E. Giger, M. Pinzger and H.C. Gall, "Can we predict types of code changes? an empirical analysis," 2012 9th IEEE working conference on mining software repositories (MSR), IEEE, Jun. 2012, pp. 217-226.

[55] R. Abbas, F. A. Albalooshi and M. Hammad, "Software change proneness prediction using machine learning,"2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), IEEE, Dec. 2020, pp. 1-7.

[56] R. Malhotra and M. Khanna, "Investigation of relationship between object-oriented metrics and change proneness," International Journal of Machine Learning and Cybernetics, vol. 4, Aug. 2013, pp. 273-86.

[57] F. Toure, M. Badri and L. Lamontagne, "Investigating the Prioritization of Unit Testing Effort using Software Metrics," ENASE, Apr. 2017 Apr, pp. 69-80.

[58] L. Kumar, S.K. Rath and A. Sureka, "Using source code metrics and multivariate adaptive regression splines to predict maintainability of service oriented software," 2017 IEEE 18th international symposium on high assurance systems engineering (HASE), IEEE, Jan. 2017, pp. 88-95.

[59] S.R. Moshin, M. Rahman, H. Parvez, O. Badreddin and S. Al Mamun, "Performance analysis of machine learning approaches in software complexity prediction," Proceedings of International Conference on Trends in Computational and Cognitive Engineering, Springer, 2021 pp. 27-39.