Hindawi
Scientific Programming
Volume 2023, Article ID 6221388, 18 pages
https://doi.org/10.1155/2023/6221388

Review Article

Predictive Analytics and Software Defect Severity: A Systematic Review and Future Directions

T. O. Olaleye,[1] **O. T. Arogundade**,[1,2] **Sanjay Misra**,[3] **A. Abayomi-Alli**,[1] **and Utku Kose**[4]

[1]*Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria*
[2]*Department of Mathematical Sciences, Anchor University Lagos, Lagos, Nigeria*
[3]*Department of Computer Science and Communication, Østfold University College, Halden, Norway*
[4]*Department of Computer Engineering, Suleyman Demirel Universitesi, Isparta, Turkey*

Correspondence should be addressed to O. T. Arogundade; arogundadeot@funaab.edu.ng and Sanjay Misra; ssopam@gmail.com

Received 23 July 2022; Revised 7 December 2022; Accepted 22 December 2022; Published 2 February 2023

Academic Editor: Cristian Mateos

Software testing identifies defects in software products with varying multiplying effects based on their severity levels and sequel to instant rectifications, hence the rate of a research study in the software engineering domain. In this paper, a systematic literature review (SLR) on machine learning-based software defect severity prediction was conducted in the last decade. The SLR was aimed at detecting germane areas central to efficient predictive analytics, which are seldom captured in existing software defect severity prediction reviews. The germane areas include the analysis of techniques or approaches which have a significant influence on the threats to the validity of proposed models, and the bias-variance tradeoff considerations techniques in data science-based approaches. A population, intervention, and outcome model is adopted for better search terms during the literature selection process, and subsequent quality assurance scrutiny yielded fifty-two primary studies. A subsequent thoroughbred systematic review was conducted on the final selected studies to answer eleven main research questions, which uncovers approaches that speak to the aforementioned germane areas of interest. The results indicate that while the machine learning approach is ubiquitous for predicting software defect severity, germane techniques central to better predictive analytics are infrequent in literature. This study is concluded by summarizing prominent study trends in a mind map to stimulate future research in the software engineering industry.

1. Introduction

Software defect prediction (SDP) suffixes to determine faulty parts of a software system proactively. It is one of the most time-consuming and important stages of the software development life cycle [1, 2], with different testing strategies including cloud-based options [3, 4]. Early prediction and identification of defective parts expectedly warrant immediate debugging [5], which is a function of the severity level of the identified defect(s). Moreover, software requirement gathering is a critical introductory phase of the software development process [6–8]; hence, high priority is given to software testing during development [9] with the aim of ensuring compliance with the requirement specifications. Software defect has been broadly described as either a fault, blunder, or bug in a software product, with a corresponding unexpected output and or an unpremeditated behavioral outcome [10] contrary to the quality intentions of software engineers and expectations of the end-users. Whether outsourced or developed in-house, critical success factors are essential for software development [3, 11–13] in order to avoid costly defects. However, advances in data science, automation, and information technology are rapidly enhancing the aim of improving assurances across the software development value chain [14], as better technological advancements are tailormade for massive data mining sequel to its predictive analytics. Nonetheless, a review of such problem-solving approaches is needed to guide practitioners and the research community for continuous improvement across the pipeline besides the dire need of an up-to-date understanding of trends, research direction, lapses, potentials, etc., in the deployment of data science, in particular, for software defect severity tasks.

While reiterating the applicability of data science in virtually all fields of human endeavor, Olaleye et al. assertions in ref. [15] underscore the global research community's interest in adopting predictive analytics for resolving industry problems just like for software defect predictive modeling [16, 17], as evident in literature across digital libraries consulted for the purpose of this work. Therefore, data science offers a reliable functionality for scientific studies which avail review studies a retinue of sources of investigations [18]. In the existing systematic literature reviews on software defect severity prediction, the bias-variance tradeoff consideration is seldom factored in research questions, which has dire consequences on the eventual performance metrics of the machine learning models. If unaddressed, it is a major threat to the internal validity of data science-based studies. Furthermore, only a few review works are obtainable focusing on text analytics techniques for software defect severity prediction. The approach of text analytics falls within the natural language processing (NLP) use case of artificial intelligence, which comes with diverse functionalities that would make a robust literature review with respect to encompassing research questions. For instance, the basic but all-important preprocessing cycle of NLP tasks, including tokenization, stemming, and other feature engineering techniques of feature selection, were not investigated in existing studies, including parameter optimization considerations. These further constitute internal threats to the validity of proposed frameworks. Investigation of future recommendations on future works is also left out in most of the existing reviews. Furthermore, the need to examine the level of multicollinearity inherent in training datasets, which is the hallmark of data science-based predictive analytics [19], is not investigated. In addition, the nature of datasets employed in literature is very important in mitigating threats to the external validity of proposed frameworks, a consideration hardly investigated by existing review studies.

This study is therefore motivated by the need to address the aforementioned gaps, which all constitute internal and external threats to the validity of machine learning-based studies.

The objective of this study is to address the lapses while carrying out a critical review of identified primary studies targeting techniques that speaks to the internal and external validity of their approaches. These expositions are exceptionally paramount for researchers with the intent of unraveling novel methodological concepts for future studies in the domain of software quality assurance using text analytics.

The main contribution of this study, therefore, is in the analysis of studies with respect to techniques that have a direct correlation with threats to the internal and external validity of their work, as well as their bias-variance tradeoff consideration. This partly determines the research questions formulated for this study and as well as unraveled the research agenda for future studies. As recommended by the authors of ref. [8], the guidelines for conducting a systematic literature review in software engineering in ref. [20] are employed in this study. In all, eleven research questions are formulated, as inspired by the objectives of the study, and are presented in Table 1. The mind map in Figure 1 aptly captures the focus of the paper and the internal validity considerations to be investigated in primary studies.

To fill the aforementioned gaps amongst many others, an SLR on the deployment of machine learning techniques for software defect prediction is hereby presented based on articles within the last decade (2011 and 2022). Other aims of the study include the following theories:

(i) Investigating limitations to inspire future research.
(ii) Unraveling success variables adopted in the preprocessing phase of text analytics (if any).
(iii) Contrasting current research trends with future works.
(iv) Bias-variance tradeoff consideration proposed in methodologies, etc. Besides a review of existing methodologies, this work equally aims to unravel and present the confines of the current approaches including data sampling consideration, training set, training strategy, and compliance with recommendations in the literature.

1.1. Research Questions and Study Mind Map. Research questions and objectives for this SLR are listed in Table 1. From studies, the data sampling state, the most prevalent public data, and the most popular learner algorithm in literature are discovered through research questions R-Q1, R-Q2, and R-Q3. The learning approach employed in literature including cross-validation or hold-out, popularity of parameter tuning, etc., is analyzed in R-Q4 to R-Q6. The significant area of feature engineering is covered in R-Q8 while R-Q9 answers the all-important question of the nature of software metrics deployed. To inspire future studies, threats to validity and proposed future works are determined in R-Q10 and R-Q11. Figure 1 shows the simple mind map of the SLR to identify software defect severity prediction models, approach, framework, dataset, and test natural language processing approach used in literature. The mind map will eventually be used to gauge observations from primary studies at the end of this review.

2. Literature Review

Systematic literature reviews (SLRs) on software defect predictions majorly concentrate on the choice of algorithms, methodologies, performance analysis, etc., without consideration for data-based threats to internal and external validities of primary studies. Hence, primary studies considered for mapping are chosen carefully to ensure research questions are efficiently addressed, which necessitated a clear-cut search criterion, and therefore, a limited number of primary studies is considered since this study is only concerned with literature that offers adequate answers to set-out research questions. Azeem et al. in ref. [5] focused on literature between 2000 and 2017 detecting code smell through machine learning

Table 1: Research questions guiding the SLR.

Research questions	Objectives
*R-Q1: which is the most widespread data sampling state?	To realize the sampling state of dataset mostly deployed so far
*R-Q2: which public data are often deployed	To identify public datasets popularly or frequently used in literature
*R-Q3: which machine learning approach is popular in literature?	To identify the type of machine learning variate mostly used
*R-Q4: does the choice of learner algorithm/ensemble impact the performance of defect severity prediction?	To realize the consensus learner algorithm recommended in the literature
*R-Q5: does training strategy impact prediction performance?	To study various fold validation option-choice
*R-Q6: is parameter tuning optimization popularly factored into predictive analytics?	To know the extent to which results in the literature are enhanced by tuning options
*R-Q7: which training tool is mostly adopted?	Way of identifying the utilitarian value of tools for ML
*R-Q8: what feature selection algorithm is mostly deployed?	To identify the most deployed dimensionality reduction technique in literature
*R-Q9: what is the course of action between "within" and "cross-project adoption"?	A way of understanding the road map of SDP as implemented in previous studies
*R-Q10: what are the prominent threats to the validity of proposed models	To identify from literature germane threats in literature to inspire future studies
*R-Q11: understanding the future direction of software defect prediction studies with respect to threats to validity reported	To do a one-to-one mapping of threats reported with future work directions

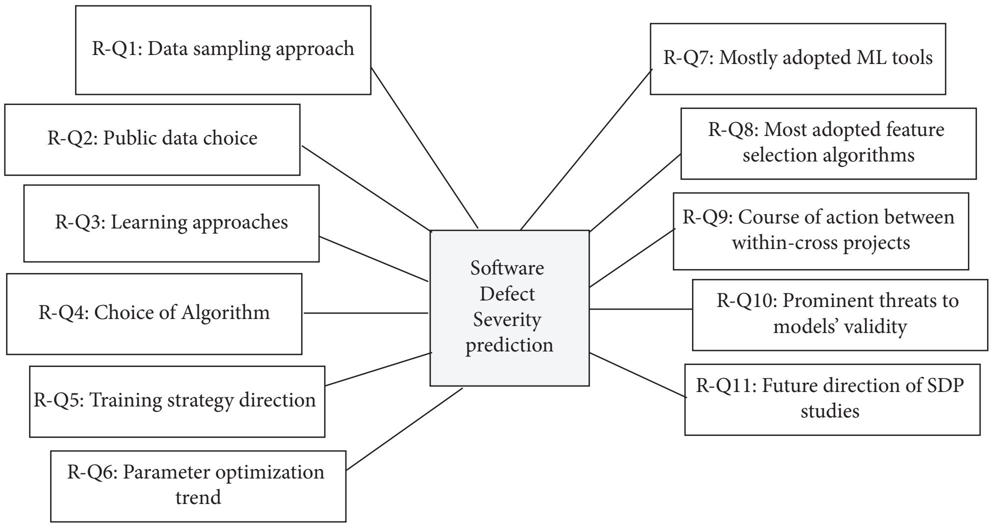
R-Q1: Data sampling approach
R-Q2: Public data choice
R-Q3: Learning approaches
R-Q4: Choice of Algorithm
R-Q5: Training strategy direction
R-Q6: Parameter optimization trend
Software Defect Severity prediction
R-Q7: Mostly adopted ML tools
R-Q8: Most adopted feature selection algorithms
R-Q9: Course of action between within-cross projects
R-Q10: Prominent threats to models' validity
R-Q11: Future direction of SDP studies

Figure 1: Internal validity-aware mind map of the review study.

techniques targeting setups of machine learning approaches, how evaluation strategies are conducted, and a meta-analysis of performance metrics from models under study. Their analysis revealed challenges that are not yet addressed by the research community. The work of Son et al. in ref. [21] focuses on research studies from 1995 to 2018 with a systematic mapping of literature in the software quality domain, adopting a multistage process. It focused on models that could classify software metrics from different projects to help organizations with little data, just as in ref. [10], aiming to identify research trends, methods, frameworks, and datasets in SDP deployed between 2000 and 2013.

While reviewing papers between 2004 and 2012, the authors in ref. [9] categorized primary studies into several parts. Those executed within the framework of classification, clustering, and ensemble with the profound investigation of software metrics deployed for software defect prediction in the works, including ISO standards, CMM, software testing, and unique software metrics. Quality improvements were central to the work of ref. [22] as it identifies contributory factors with consequent remedial courses to improve software productivity and quality. Models deployed in the literature were evaluated based on stated criteria with personal observations related to the models discussed.

In ref. [23], the authors reviewed studies between 1991 and 2013 and identified categories of machine learning models, as well as including studying performance accuracy analysis, reviewing statistical approaches while understudying the strength of machine learning models and similarly; the authors of ref. [24] reviewed data mining techniques deployed for software defect prediction works in literature under review. The authors thoroughly likewise reviewed datasets used, tools deployed for predictive analytics, performance measures used in literature, etc.

While working on an empirical study of literature between 1995 and 2018 for systematic mapping, the authors in ref. [25] reviewed 98 primary papers out of the initial 156 accessed from reputable digital libraries to address nine germane research questions centered on various aspects of defect prediction through predictive analytics. Unlike other SLRs, the authors factored in threat to validity items in their study. In ref. [26], only three research questions were investigated in 208 studies published between 2000 and 2010, trying to investigate how the performances of models are affected by the context in which the models are developed, the techniques upon which the models stand, and the independent variables deployed for the models.

A deliberate attempt to identify threats to the internal and external validity of existing studies is missing from the reviewed studies. Especially, studies do not make attempt to relate the aims of primary studies with their future recommendations. This would easily reveal lapses in the future direction. Notwithstanding the performance metrics of primary studies, identified threats to their internal and external validity would avail optimized conceptual frameworks in future studies. These lapses are to be investigated in this study for future SDSP studies.

2.1. Contributions. The relevance of this SLR is stated as follows:

(1) Identification of fifty-two primary studies on software defects predictive analytics within the last decade
(2) Analysis of feature engineering techniques, unsupervised, supervised, semisupervised and clustering learning approaches in the last decade to capture the past while inspiring the future of the domain use case
(3) This study underscores the direct correlation between threats to validity and future research recommendations as observed in the reviewed literature
(4) With this work, a review of data preprocessing techniques was sought-after, and other feature engineering techniques

3. Research Methodology

The research methodology adopted in this work is a systematic literature review (SLR) within the scope of software defect prediction. The study aims at addressing research questions set out earlier in this work. As common with other SLRs in the area of software engineering, guidelines stipulated in ref. [27] are adopted in this work as best practices while "snowballing" as defined by Wohlin [28] for a systematic inclusion of some references is likewise employed. Specifically, snowballing involves the study of the reference list of a particular paper or the mention of the paper to detect extra sources. The process followed is described in the subsections as follows.

3.1. Search Strategy. The search process consists of activities including selecting choice digital libraries, preferred search strings, initial search, retrieving the initial list, and refining the search string to get an initial list of primary studies from digital libraries. The following list of digital libraries was consulted as those reputable for software engineering resources [29] as recommended in several software engineering-based reviews:

(1) IEEE digital library *(@ieeexplore.ieee.org)*
(2) ACM digital library *(@dl.acm.org)*
(3) SpringerLink library *(@link.springer.com)*
(4) Google Scholar *(@scholar.google.com)*
(5) Elsevier *(@elsevier.com)*

To identify search items relevant to the course of intent and action:

(1) Search terms were influenced by research questions by identifying population, intervention, and outcome
(2) Synonyms were identified for major search terms for an inclusive result
(3) Keywords were verified and ascertained in all the listed literature
(4) Boolean operators, for allowed database, were used including OR and AND to either concatenate alternative spellings and synonyms or inclusion of major keywords, respectively
(5) The search string was summarized sometimes for a more compact and specific search outcome

For (1), population, intervention, and outcome were adopted so that better search terms could be obtained.

(i) Population: prediction of software defect severity
(ii) Intervention: machine learning techniques
(iii) Outcomes: severity level prediction

A sample research question with the detail mentioned above is given below:

*R-Q5: does the choice of learner algorithm/ensemble (INTERVENTION) impact performance (OUTCOME) of defect severity prediction? (POPULATION)

For (2 and 5), alternative synonyms and spellings as the case may be included:

(i) Software defect: ("software bug" OR "software smell" OR "software fault")
(ii) Machine learning: ("machine learning" OR "predictive analytics" OR "text analytics" OR "supervised learning" OR "unsupervised learning" OR "classification" OR "clustering")
(iii) Prediction: ("prediction" OR "detection" OR "identification")

For (3), keywords in the literature were conventional; hence, no suitable alternative was found.

For (4), Boolean operators were deployed and were acceptable to coin search queries.

3.2. Process of Selecting Primary Study. The selection process overview of Figure 2 shows paths followed for capturing the research articles concerning the study's defined objectives. Emerging search results are shown in the second column of Table 2 with their corresponding libraries with an initial 2641 papers regarding the query. Table 3 indicates the inclusion and exclusion criteria set out for this study. The title, abstract, and keywords of 2641 found articles were difficult to study, and articles that met such standards were removed, while the remaining were adopted for the next screening with new additions from snowballing (the process of identifying relevant literature from the reference list of a paper understudy) as depicted on Figure 2. Hence, 2.46% of the initial set comprising of 65 papers made the selection.

Specifically, the title is first considered: if it is out of scope (i.e., if it is not about the predictive analytics of the use case), the article is skipped; otherwise, if considered possibly useful, abstract, introduction, methodology, and conclusion suffixes to further ascertain appropriateness for the study. Thus, the eventual sixty-five papers passed all the steps. Once selection papers are established to be considered for the SLR, the quality assessment follows, primarily to confirm that all final papers had the requisite information to answer the research questions (see Table 1), culminating in the final selection analyzed for this study. The quality assessment test turned out positive for each of the selected fifty-one studies. Thus, the SLR study is based on fifty-two papers. The information extraction process and form are later presented in this section.

3.3. Inclusion and Exclusion Principles for Study Selection. The inclusion and exclusion criteria stipulated for primary study selection or rejection are as shown in Table 3. Table 4 shows the data extraction form designed to acquire valuable insights from the primary studies, which speaks to the internal validity observations inherent in their methodologies. The eventual list of primary studies, after the inclusion and exclusion criteria have been implemented as is presented in Table 5.

3.4. Quality Valuation of Papers. The selected primary studies are subjected to a quality assurance postfinal selection process. The following checklist was adopted for credibility checks on selected publications:

(i) Q1: are the learning approach, cross-validation, and class labeling presented?

(ii) Q2: is the choice of learner algorithms and ensemble stipulated?

(iii) Q3: is the choice or otherwise of (i) feature engineering and (ii) parameter optimization adopted stated with reasons?

(iv) Q4: is the choice of public data noted, sampling or resampling attempts, and metrics characterization stipulated?

(v) Q5: does the study shows its domain choice of software metrics between cross and within projects?

(vi) Q6: are there clearly stipulated aims, a threat to validity, and clearly defined future work suggestions?

Each question mentioned above attracts a "Yes," "Partly," or "No," while a study is considered as *partial* in situations where questions not adequately answered are not those addressed by Q3–5. These answers are graded as 1, 0.5, and 0 for "Yes," "Partially," and "No," respectively. For a study, its quality grade was computed as the sum of the graded answers with respect to the six questions. The quality level was regarded as good (with a grade ≥4), average (with 3≤ grade <4), and low (with a grade <3). Fifty-two studies made the good and average sets which forms the final selection.

As observed from Table 5, studies that made the final list with average and good qualities range from papers presented at conferences and international journals. The aims of the papers were clearly defined in the second column of the table and the year of publications show a period of eleven years, from 2011 to 2022 in the software defect severity prediction research domain.

3.5. Data Extraction. Data extraction commenced, answering stipulated research questions once the final papers selected for the review were ascertained. Precisely, the data extraction form earlier presented in Table 4 is used for grading.

4. Result Analysis

In this section, the results of the review are discoursed towards addressing the research questions.

4.1. Study Demographics. Table 5 shows the list of 52 kinds of literature studied for this SLR review with corresponding years of publication and the type of publication in a journal or conference proceedings. The works are clearly executed between 2011 and 2022, spanning a decade of software defect severity level prediction tasks using machine learning. As may be observed in Figure 3, 64% of the studies have been published in the last five years, indicating a rising profile of software defect severity prediction interest amongst researchers in the engineering genre, while over 58% of the publications are journal articles.

4.1.1. R-Q1: Data Sampling Approach in Literature. The outcome of this SLR concerning the R-Q1 is in no way different from trends observed from previous works where the majority of datasets deployed for training learner algorithms are grossly of imbalanced class distribution, which

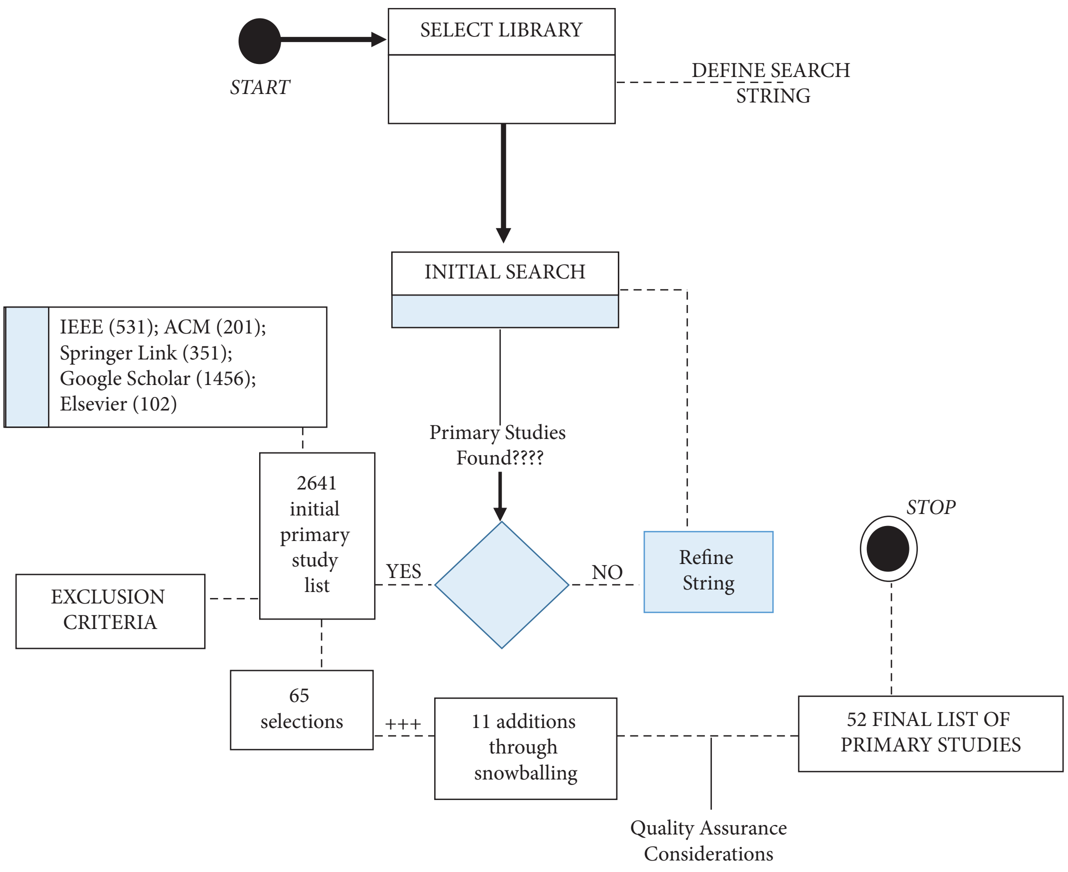
START
SELECT LIBRARY
DEFINE SEARCH STRING
INITIAL SEARCH
IEEE (531); ACM (201); Springer Link (351); Google Scholar (1456); Elsevier (102)
Primary Studies Found????
2641 initial primary study list
YES
NO
Refine String
STOP
EXCLUSION CRITERIA
65 selections
+++
11 additions through snowballing
52 FINAL LIST OF PRIMARY STUDIES
Quality Assurance Considerations

Figure 2: Interaction overview for primary studies selection.

Table 2: Source of primary data with search results.

Library name	Results found	Initial selection	Final selection after quality assurance
IEEE	531	17	14
ACM	201	9	6
SpringerLink	351	11	6
Google Scholar	1456	18	10
Elsevier	102	10	6
Total	2641	65	42
Snowballing	12	11	9
Final total	2653	76	52

Table 3: Inclusion and exclusion criteria.

Inclusion criteria	Articles adopting predictive analytics and text analytics, natural language processing, and written in English for software defect prediction Articles adopting an addendum method to its machine learning approach
Exclusion criteria	Studies not written in the English language Studies not relevant to machine learning or text analytics Studies that do not clearly define every item of corpus preprocessing and or text analytics Papers whose full text are not accessible

is a characteristic nature of software defect public data [67] and hence is highly biased. This is attributed to the fact that software metrics are more defect-free than defect prone; hence the defect instances are seldom as much as the instances without defect [48], which attests to why the dependent variables are favorably skewed towards nondefect instances, especially for binary classification since unsupervised learning is devoid of labeling. The result presented in Table 6 speaks volumes of the data sampling and resampling pattern in primary studies as only a few altered training dataset distributions by way of resampling to address the bias, which is essential for a better predictive

Table 4: Data extraction from.

S/N	Dimension	Attribute: description
1	The type of severity prediction class adopted	Probability, severity levels, or binary classification
2	Data sampling status and metrics characterization	The distinction between balanced or imbalanced data together with data attribute type either of nominal or numeric
3	Category of machine learning used and learner algorithm used for prediction	Choice of either supervised or unsupervised with learner algorithm used for prediction
4	Parameter tuning or optimization strategy adopted	Identification of performance improvement tuning adopted in literature
5	Threats to validity with respect to future work suggestion	Mapping of primary studies threats to proposed future work outlook in the software defect prediction industry
6	Threat to validity and future work recommendations	Juxtaposing vulnerability with future potentials
7	Use case application area	Identifying application area of either cross-project or within the project use case

Table 5: List of reviewed primary studies.

Id	Aim of research work	Ref.	Year	Type of publication
[L01]	Prediction of software defect vulnerability level with the textual corpus	[30]	2021	Conference
[L02]	Deploying ensemble learning for software defect prediction	[31]	2021	Journal
[L19]	Software defect predictions with techniques	[32]	2020	Conference
[L28]	To use text analytics on apache projects for bug severity prediction	[33]	2020	Journal
[L29]	Questioning and answering approach to bug prediction	[34]	2020	Journal
[L30]	To use the machine in the prediction of software defect	[35]	2020	Journal
[L31]	Weighted extreme learning machine approach to predict software defect	[36]	2020	Journal
[L36]	Heterogeneous ensemble classification approach to defect prediction	[37]	2020	Journal
[L12]	To use a feature selection strategy for improved defect prediction	[38]	2019	Journal
[L16]	Virtual classification approach to software defect prediction	[39]	2019	Conference
[L20]	Code smell prediction using an extreme learning machine	[40]	2019	Conference
[L27]	To use CNN and random forest for the classification of bug severity	[41]	2019	Journal
[L33]	Approaching software defect prediction through feature selection	[38]	2019	Journal
[L35]	Predicting software bugs in closed-source projects	[42]	2019	Journal
[L45]	Cluster and hybrid feature selection approach to defect prediction	[43]	2019	Conference
[L50]	Software defect prediction using an unsupervised approach	[44]	2019	Conference
[L11]	Using bug report classification for incorporate and textual field knowledge	[45]	2018	Conference
[L38]	Identifying impacts of imbalanced ensemble learning methods for cross projects	[46]	2018	Journal
[L43]	Semisupervised approach to defect prediction in cross-project and within-project	[47]	2018	Conference
[L46]	Heterogeneous piling and SMOTE approach to ensemble defect prediction	[48]	2018	Conference
[L51]	Deployment of the ensemble for class imbalance in defect prediction	[49]	2018	Journal
[L10]	Corpus vulnerability description approach towards defect prediction	[50]	2017	Conference
[L17]	Using feature-dependent Naïve Bayes approach	[51]	2017	Journal
[L26]	Adoption of predictive analytics for code smell prediction	[52]	2017	Journal
[L37]	Assessment of bug severity in cross-project while in training candidates	[53]	2017	Journal
[L42]	SMOTE and ensemble approach to defect severity prediction	[54]	2017	Conference
[L44]	Unsupervised machine learning approach to defect prediction	[55]	2017	Conference
[L04]	Diversity selection approach to defect prediction using ensemble	[56]	2016	Conference
[L09]	Using a semisupervised approach on imbalance set for defect prediction	[57]	2016	Conference
[L14]	Machine learning techniques for defect prediction in android software	[58]	2016	Journal
[L25]	Machine learning and text mining approach to bug severity classification	[59]	2016	Journal
[L48]	Predictive analytics on software project report	[60]	2016	Journal
[L49]	Diversity selection and ensemble approach to defect prediction	[61]	2016	Conference
[L08]	Using a dictionary of critical terms for predicting software bug severity	[62]	2015	Conference
[L03]	Using a dictionary of known terms for defect prediction	[63]	2014	Conference
[L05]	Mining software repository for defect prediction	[64]	2014	Book
[L07]	Neural network approach to predictive analytics	[65]	2014	Conference
[L15]	Improving VAB-SVM prediction for defect prediction in cross-project	[66]	2014	Conference
[L22]	Evaluating learners performance on imbalanced dataset for defect prediction	[67]	2014	Journal
[L23]	Defect prediction using machine learning techniques	[68]	2014	Journal
[L24]	Comparing statistical and machine learning methods for faulty modules	[69]	2014	Journal
[L47]	Data mining and multi-layer perceptron approach to defect prediction	[70]	2014	Journal

Table 5: Continued.

Id	Aim of research work	Ref.	Year	Type of publication
[L06]	Iterative and noniterative feature engineering approach for software defect prediction	[71]	2013	Journal
[L18]	Trying decision tree approach to software defect prediction	[72]	2013	Journal
[L39]	Deploying supervised and unsupervised learning approaches to defect prediction	[73]	2013	Journal
[L13]	Machine learning techniques adoption for bug severity prediction	[74]	2012	Conference
[L21]	Two-level data preprocessing approach to defect prediction	[75]	2012	Conference
[L32]	To identify the impacts of different classifiers to predict software defect	[76]	2012	Journal
[L34]	Ensemble learning approach to improve software defect prediction	[77]	2012	Journal
[L40]	Feature selection approach to defect prediction in software	[78]	2012	Conference
[L41]	An AHP-based evaluation method for ensemble defect prediction	[79]	2011	Journal
[L42]	Enhanced random forest (extRF) approach in IOT-based application processing environment using a business process management and improvement concept	[80]	2022	Journal

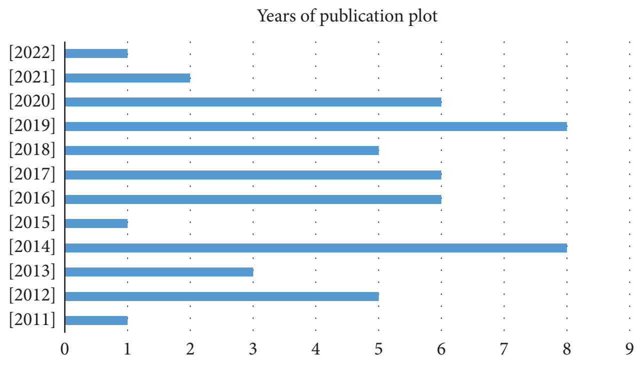

Figure 3: Distribution of primary across timelines.

Table 6: Data sampling approaches.

Data sampling	Frequency	Cited
Imbalanced	37	[L01], [L03], [L04], [L05], [L07], [L08], [L10], [L11], [L12], [L13], [L14],[L16],[L17],[L18], [L19], [L24], [L25], [L26], [L27], [L28], [L29], [L30], [L31], [L32], [L33], [L35], [L36], [L39], [L40], [L41], [L43], [L44], [L45], [L47], [L48], [L49], and [L50]
Resampled		
Resampling approach		
Oversampling	9	[L02], [L09], [L21], [L34], [L37], [L38], [L42], [L46], and [L51]
Undersampling	1	[L06]
Hybrid	2	[20] and [L22]
Others	2	[L15] and [L23]

accuracy [31]. Considering literature with resampling efforts, most of those adopting minority oversampling deployed the synthetic minority oversampling technique (SMOTE) for class distribution resampling amidst several other options. Ensemble approaches were likewise implemented in other literature, especially to limit the adverse effects of class imbalance. Few pieces of literature adopted the strategy of deploying a multivariate class labeling by further decomposing class labels into various severity levels to reduce bias.

(1) Summary for R-Q1. 72.5% of primary papers deployed an imbalanced training set for SDP while 17.6% adopted oversampling, with 1.96% adopting undersampling in their attempts to resample. Hybrid approaches and others which constitute 3.9% appease use other resampling methods.

4.1.2. R-Q2: Which Public Data Are Often Deployed? Public data are often used for predictive analytics in SDP literature [10]. As observed from primary studies for this SLR, both within-projects and cross-projects have adopted the same approaches with varying choices across publicly available sets. Targets of researchers in choice of dataset vary likewise but ultimately of high consideration is the nature of metrics concerning independent variables under consideration as noticed from this study. Concerning the graph plot presented in Figure 4, twenty-two different datasets are

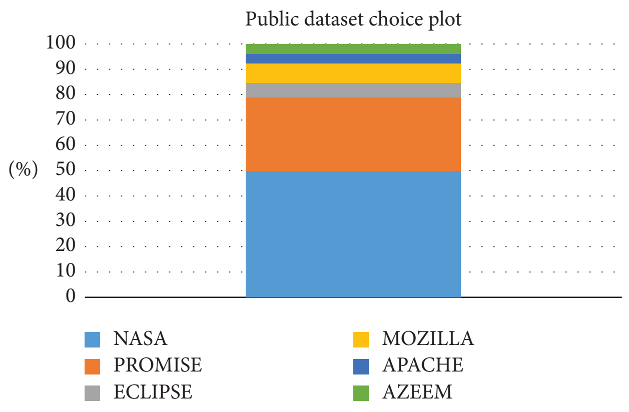

Figure 4: Utilitarian spread of public datasets.

under consideration, with each spreading across one, two, or more of the presented datasets.

(1) Summary for R-Q2. NASA dataset is the most adopted of all 22 used in literature with a 49% deployment rate while Promise follows with 29.41%. Eclipse, Mozilla, Apache, and Azeem likewise enjoyed considerable adoption in literature.

4.1.3. R-Q3: Most Adopted Machine Learning Variant. Machine learning variants adopted in the literature vary, and it is the exclusive preserve of a researcher to adopt whatever variant in the course of its SDP study. As observed in the sunburst plotted in Figure 5, the SDP research community highly deploys supervised learning with unsupervised learning coming second. [L09] and [L43] deployed a semisupervised approach in their study while [L34] adopted a dictionary learning approach. Adoption of both unsupervised and supervised learning variants is encapsulated in the work of [L36], where training data are clustered for a subsequent categorized supervised training while the work of [L44] seeks to compare the predictive accuracy of supervised and unsupervised variants of machine learning.

(1) Summary for R-Q3. Supervised learning is widely adopted despite reports of better performance by unsupervised learning. It is recommended that the unsupervised learning approach be given more attention in SDP to further improve the predictive accuracies of software defects.

4.1.4. R-Q4: Preferred Choice of Learner Algorithm and/or Ensemble. With several machine learning algorithms at the disposal of the software engineering research community, Figure 6 captures the most deployed in the primary studies across the various models proposed. While Naïve Bayes enjoyed the biggest patronage in the supervised learning subcategory, K-means was the most adopted in the unsupervised learning category for base learning and clustering, respectively. Bagging was mostly deployed across the literature in the ensemble set, as evident in Table 7 presented, followed closely by Stacking and Boosting with five (5) representations each.

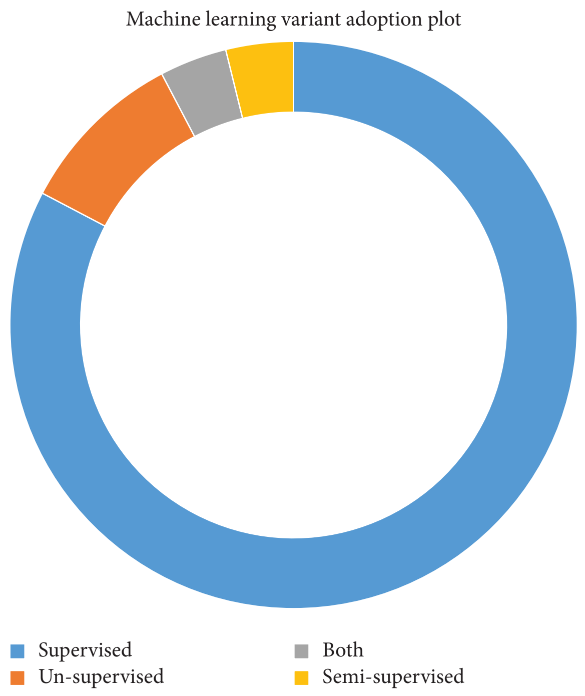

Figure 5: Choice of machine learning approach.

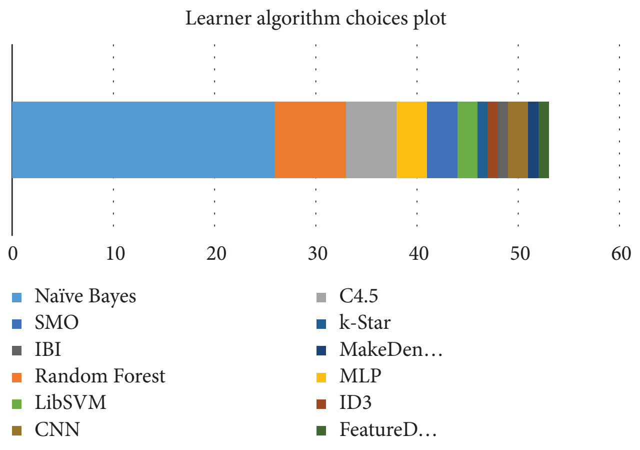

Figure 6: Learner algorithm distribution.

(1) Summary for R-Q4. Naïve Bayes is widely adopted in literature, accounting for over 57% of the utilitarian value, and it is the most recommended by literature in their conclusions. Bagging tops the ensemble category.

4.1.5. R-Q5: What Training Strategy is Mostly Deployed? Studies employed mostly 10-fold cross-validation with 5-fold cross-validation methodology coming next in various attempts to control the rate of variance in the estimation of predictive models. The stratified approach employed by some studies guarantees equal sampling of the minority and majority class as in the original data for each fold, avoiding folds founded only in the popular class. Table 8 shows the close range of the training strategy adopted in the literature.

Table 7: Distribution of ensemble methods in the literature.

S/N	Ensemble	Literature
1	Coding based	[L3]
2	Stacking	[L4], [L2], [L41], [L46], and [L49]
3	Vote	[L37]
4	XGBoost	[L2], [L30], and [L36]
5	Bagging	[L1], [L2], [L14], [L22], [L28], [L36], [L37], [L38], [L41], [L42], and [L46]
6	Boosting	[L15], [L27], [L28], [L36], [L38]
7	AdaBoost	[L41], [L42], and [L46]
8	RF ensemble	[L42]
9	Other	[L17] and [L51]

Table 8: Training strategy.

Fold cross-validation	[L04], [L06], [L12], [L13], [L14], [L21], [L28], [L22], [L24], [L23], [L30], [L36], [L39], and [L24] [L26], [L35], [L36], [L39], [L40], [L41], [L42], [L43], [L44], [L48], [L49], and [L51]
Hold-out	[L01], [L02], [L03], [L05], [L07], [L08], [L09], [L10], [L11], [L12], [L15], [L16], [L17], and [L18] [L19], [L20], and [L25] [L31], [L32], [L33], [L34], [L37], [L38], [L45], [L46], [L47], and [L50]

(1) Summary for R-Q5. An average of 47% of primary studies deployed fold cross-validation while 52.9% employed the hold-out strategy for their studies, showing an almost balanced representation in the population.

4.1.6. *R-Q6: Is Parameter Optimization Popularly Factored into SD Predictive Analytics?*

Parameter tuning has been highly recommended in the literature for any predictive analytics [10] study for improved accuracy in an attempt to optimize performance by changing parameter settings. [L22] created two versions of K-Nearest neighbor 2NN and 5NN by tuning values of k (2,5) and [L04] likewise tuned two parameters of K-NN learner including the k-value (denoting how many neighbors be taken into consideration) and the nearest neighbor searching algorithm. As evident in Table 9, few studies reported parameter optimization measures in their research.

(1) Summary for R-Q6. To the best of our knowledge, 11% of the study population (6 papers) only reported cases of parameter optimization to enhance prediction performance.

4.1.7. *R-Q7: Which Is the Most Adopted ML Tool?*

Some of the models presented in the literature are simulations of the SDP analytics, and other studies developed software codes to implement proposed models. In an attempt to fully grasp different approaches in primary studies, the inclusion of this research question becomes imperative. As observed from Figure 7, few studies explicitly mentioned specific tools employed to model their proposed architectures. However, the Waikato Environment for Knowledge Acquisition (WEKA) tool was majorly deployed in reported cases of primary studies, and Python with Rapid miner was also patronized for the modeling.

Table 9: Application of parameter tuning across primary studies.

S/N	Literature	Area of tuning application
1	[L12], [L19], and [L22]	K-NN
2	[L04]	K-NN/SVM
3	[L32]	N.B
4	[L36]	N
5	[L51]	R.F

(1) Summary for R-Q7. Integrated development environments are mostly preferred, especially for simulation purposes in analyzed primary studies.

4.1.8. *R-Q8: Which Feature Selection Option Is Widely Adopted for Dimensionality Reduction?*

Figure 8 shows the adoption rate and choice of feature engineering algorithms in primary studies with their respective rankers by way of reducing dimensionality in training sets. While machine learning thrives on huge datasets, feature engineering in predictive analytics optimizes performance [10] towards reducing redundancy, just as multicolinearity is topical for better performance in software defect predictive analytics. [L01] deployed Chi-Square and information gain algorithms for removing irrelevant features by finding dependence between two variables by ranking features and retaining top-ranking features for the SDP modeling.

(1) Summary for R-Q8. Feature engineering does not regularly feature in primary studies as only a few pieces of literature specifically aimed at deploying feature selection in their works.

4.1.9. *R-Q9: What Is the Course of Action between Inter- and Intraproject Research Direction?*

Figure 9 shows that within-project metrics are mostly deployed in primary studies with respect to the choice of data. At the same time, few studies

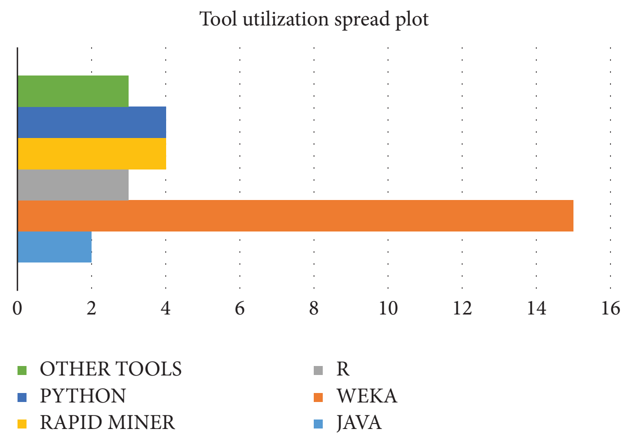

Figure 7: Utilitarian spread of machine learning and testing tools.

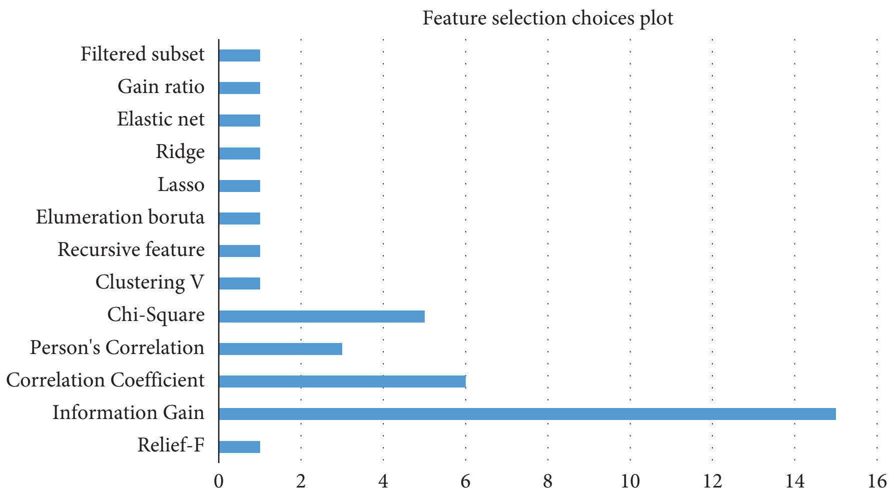

Figure 8: Feature engineering trend in studies.

reported inadequate defect metrics in within-project metrics which are used for both the training and testing phases of their work. [L37] and a few others employed a *cross*-project strategy between metrics from Eclipse and Mozilla as either is deployed for training and the other for testing for a better perspective on the model's predictive accuracy. On the contrary, [L38] adopted a cross-project to solve data imbalance, especially for assessing the efficiency of class disparity in ensemble learning.

(1) Summary for R-Q9. Defect prediction in cross-project needed further exploration in software defect severity prediction studies as few studies adopted the approach in their works.

4.1.10. R-Q10: Prominent Threats to Models' Validity Across Primary Study. Threats to validity (TTV) is germane in the field of software engineering for software defect severity prediction to ascertain levels of threats to internal and external validity in literature and as observed from Table 10, a sizeable number of studies that reported cases of TTV in their proposed models, which uncovers grey areas that may have impeded better performance in the models. Threats to external validity in an experimental software engineering study are circumstances that limit the generality of case study outcomes [71], while threats to internal validity are regarded as errors in empirical metrics and tool adoption [50]. The level of compliance with reporting TTV of inherent threats in studies is encouraging. However, studies like ref. [47] claim compliance with best practices while asserting the immunity of their study to threats. Table 10 clearly shows prominent areas of research studies where different categories of threats are reported.

4.1.11. R-Q11: Does TTV Inspire the Direction of Future Works in SDP. Mapping TTV with future work is an attempt to tag the consistency of current studies with proposed future studies as a continuous evaluation and improvement mechanism towards ensuring informed decisions. Future study direction in SDP could then be consistent with current realities. Table 11 shows the link between threats identified from primary studies and their respective future direction as

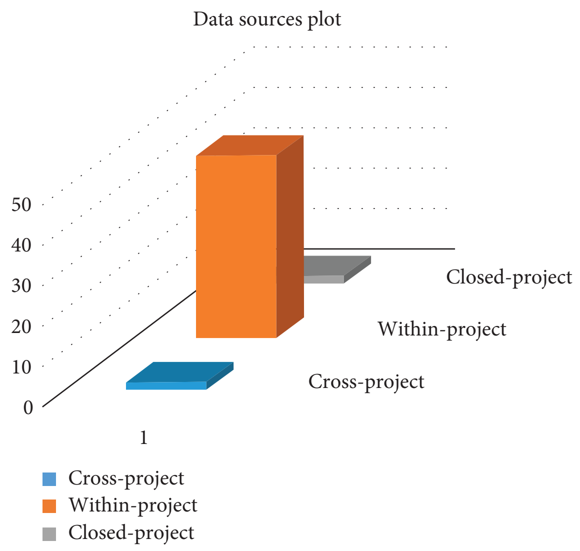

FIGURE 9: Within or cross-project study choices.

TABLE 10: Threats to validity subcategory spread.

S/N	Threat subcategory	References
1	Feature engineering-induced threats	[L04], [L06], and [L12]
2	Human error proneness	[L06], [L10], [L11], [L26], [L27], and [L29]
3	Data/software metrics related threats	[L04], [L06], [L12], [L21], [L26], [L27], [L33], [L37], [L38], [L44], and [L45]
4	Evaluation-related threats	[L15], [L21], [L43], [L38], and [L49]

TABLE 11: Mapping TTV with future work.

S/N	TTV subcategory	Reference	TTV	Future work
1	Feature engineering-induced threats	[L04]	Failure to apply full parameter search	Intend to apply full parameter search in the future
		[L40]	Feature selection does not consider semantics	Evaluation of other feature selection schemes in literature
		[L32]	Choice of feature selection algorithm	Intent to deploy other feature selection algorithms
		[L12]	Threats from feature selection	Checking for effects of imbalance & outliers on feature selection
2	Human error proneness	[L06], [L11], and [L26] [L27] and [L29]	Varying degrees and kinds of errors	No mention in future works
3	Data/software metrics related threats	[L12], [L21], and [L26] [L27], [L33], and [L37] [L38], [L44], and [L45] [L34]	Restriction of studies to particular software metrics and repositories	Studies intended to try proposed models on other data metric repositories as [L38] intend to address class imbalance in future and [L29] aiming to try commercial metrics repositories
4	Evaluation-related threats	[L15]	Standard deviation may not be an optimal choice for model evaluation	Other evaluation techniques to be considered except standard deviation
		[L21]	Bias in choice of performance parameters	Root square means method to be adopted in subsequently
		[L38]	AUC & MCC may not be appropriate measures	Trial of other performance evaluation measures

observed from the concluding section of each study. All twenty-two studies that reported threats to validity indeed expressed intent to include in their future studies grey areas enumerated in their TTVs.

(1) Summary for R-Q11. It is observed from the study that TTV indeed inspires future work direction in the SDP industry.

5. Discussion and Implications

Further discussion of the main findings (which discusses contribution to knowledge) is presented in this section with respect to the research questions set out for this study.

5.1. R-Q1: Limited Awareness of Various Implications of the Imbalance Training Set. Consequent on the foregoing evaluation of the class sampling status of the dataset deployed in literature, it is evident that studies are fully not yet abreast of the varying implications of a bias training set for SDP studies, whereas some studies indeed noted the implications, thereby deploying the ensemble learner approach or other techniques, other studies failed to acknowledge the implications besides studies that employed resampling techniques as part of their methodologies. The first recommendation is for future studies to proactively incorporate a resampling component in the data preprocessing phase of software defect predictive analytics. To ease the choice of resampling approach, the subsequent research question shows the most adopted in the literature is oversampling. Furthermore, an exploratory data analysis is highly recommended to reveal actionable insights that could inspire future conceptual frameworks.

5.2. R-Q2: Need for Inclusiveness in Software Metrics Repository Choice. The majority of studies have concentrated on either Nasa or Promise repositories, with others trailing far behind in terms of application and analysis, whereas other studies trained on the two most prominent alongside other less popular repositories; there may be a need to spread studies across repositories for an inclusive and robust generalization of results during reportage which will situate conclusions reliably. That will also eliminate the possibility of a threat to the external validity of their experimental results.

5.3. R-Q3: Supervision of the Learner Algorithm Takes Center Stage in SDP Studies. A review in this work shows the towering adoption of the supervised learning approach in literature for SDP and the subtle adoption of semisupervised alongside unsupervised learning. The combination of both unsupervised and supervised approaches for the training set clustering and classification, respectively, is likewise gaining interest in the software engineering genre. However, it is pertinent to fully consider unsupervised learning as a way of software metrics clustering as a precursor to severity level prediction.

5.4. R-Q4: Towering Posture of Naïve Bayes. As witnessed in the study, the adoption of Naïve Bayes span through the eleven years of study despite the relative choice of software metrics adoption in literature. It is noteworthy that studies that experimented on more than two base learners likewise recommended the algorithm for its performance metrics in the study, hence the need to factor its relevance into subsequent studies while studying ways of its parameter optimization for better performance.

5.5. R-Q5: Widespread Adoption of n-Fold Cross-Validation. This study shows the widespread adoption of cross-validation predictive analytics for SDP alongside the traditional hold-out approach. Reduction of overfitting has been attributed to the trend, but either way, they are considered as efficient for SDP in primary studies.

5.6. R-Q6: Considerable Low Deployment of Parameter Optimization. Revelations show a low rate of parameter optimization in primary studies despite its efficiency in natural language processing with respect to the data preprocessing phase in software defect severity prediction. There is a dire need for its deployment for better performance metrics in subsequent studies as some prominent learner algorithms will be better efficient with parameter tuning.

5.7. R-Q7: Widespread Adoption of Simulation in SDP Modeling. As noted, integrated development environments for simulation is widely deployed in literature, with WEKA leading the pack of the most widely used environment for SDP predictive analytics.

5.8. R-Q8: Information Gain and Correlation Coefficient Algorithms for Feature Selection. An appreciable number of primary studies adopted feature engineering as part of their preprocessing techniques to reduce dimensionality, while information gain and correlation coefficient enjoyed the widespread application. Feature selection is recommended as an integral part of the preprocessing phase of software defect severity level prediction for enhanced performance metrics, as studies can observe.

5.9. R-Q9: Need for a Cross-Project Approach in Future Studies. Few literature gave consideration for a cross-project approach in their studies which is highly recommended to avail models of the opportunity of cross-fertilization of software metrics with respect to testing the efficiency of training through the deployment of the industry-based test set for prediction**.** Restriction of data metrics to within-project or closed-project metrics may constitute a threat to the external validity of proposed models, especially at test time with a cross-project test set.

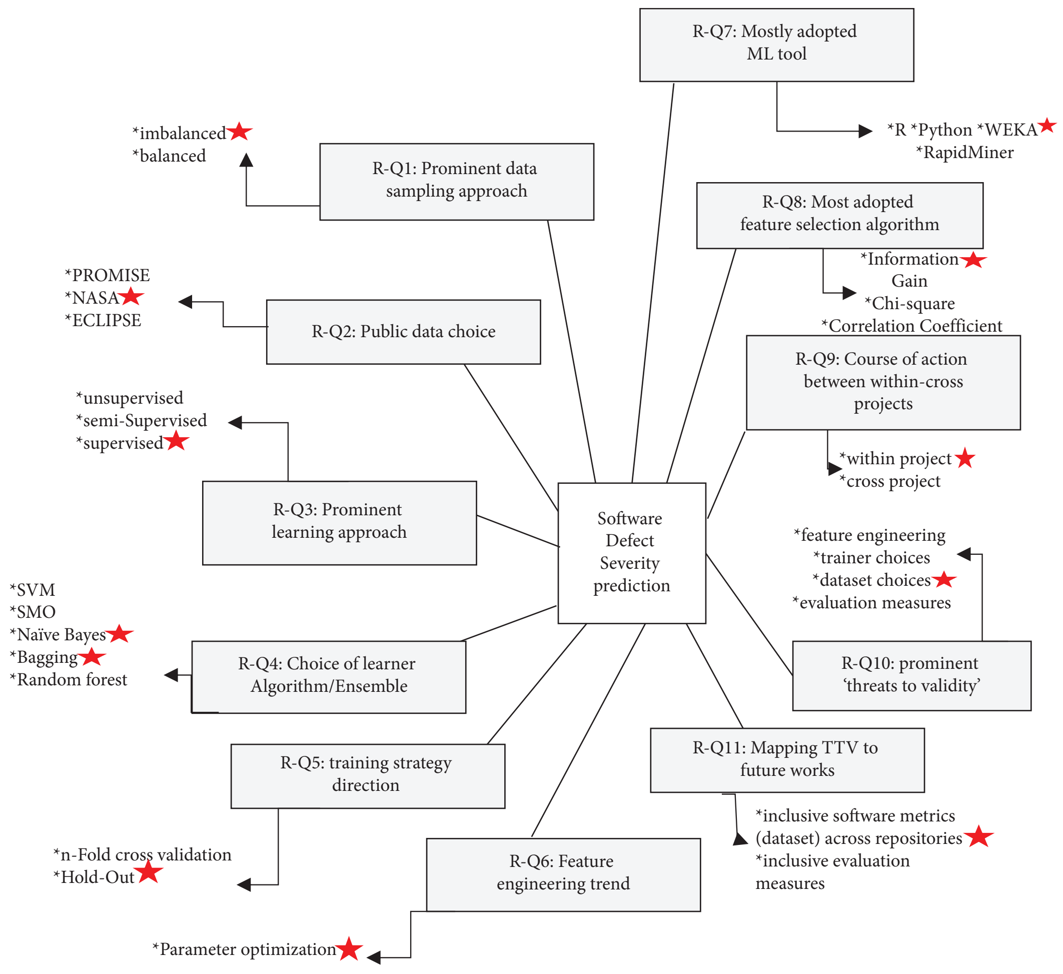
R-Q7: Mostly adopted ML tool
*R *Python *WEKA★ *RapidMiner
*imbalanced★ *balanced
R-Q1: Prominent data sampling approach
R-Q8: Most adopted feature selection algorithm
*Information★ Gain *Chi-square *Correlation Coefficient
*PROMISE *NASA★ *ECLIPSE
R-Q2: Public data choice
R-Q9: Course of action between within-cross projects
*unsupervised *semi-Supervised *supervised★
*within project★ *cross project
R-Q3: Prominent learning approach
Software Defect Severity prediction
*feature engineering *trainer choices *dataset choices★ *evaluation measures
*SVM *SMO *Naïve Bayes★ *Bagging★ *Random forest
R-Q4: Choice of learner Algorithm/Ensemble
R-Q10: prominent 'threats to validity'
R-Q5: training strategy direction
R-Q11: Mapping TTV to future works
*inclusive software metrics (dataset) across repositories★ *inclusive evaluation measures
*n-Fold cross validation *Hold-Out★
R-Q6: Feature engineering trend
*Parameter optimization★

Figure 10: Final mind map.

5.10. R-Q10: Software Metrics Posing Major Threats to SDP Models' Validity. As noted in studies with TTV included in their structure, dataset-related threats pose a major challenge to research works under reference, hence the need for future studies to undergo an in-depth analysis of software metrics available in repositories while ensuring inclusivity wider scope of databases for modeling of SDP studies.

5.11. R-Q11: Mapping of TTV with Future Works in Literature. The future direction of SDP studies is clear-cut for researchers in the industry as the consensus appears to be in the direction of ensuring the adoption of appropriate feature selection technique, ensuring data resampling, deployment of parameter optimization, concatenation of widespread software metrics of variant repositories to serve as the training set for building SDP models, etc.

Figure 10 shows a graphical representation of the final mind map of contributions in this study. With the initial mind map presented in Figure 1, it clearly clarifies the contributions of the primary studies with respect to the research questions presented in Table 1. With this, future studies on SDP could be guided by the various observations from this study.

5.12. Future Agenda. Consequently upon the foregoing, a descriptive statistical technique of exploratory data analysis (EDA) is highly recommended in future conceptual frameworks for data science-based Software Defect Severity Prediction. This will avail actionable insights from the dataset which should inspire conceptual frameworks. This will address various gaps identified from research questions R10, R9, R8, R6, R5, R2, and R1, which all constitute threats to the internal and external validities of primary studies. As observed from the final mind map of Figure 10, an EDA prior to predictive analytics would uncover the limitations of predictor attributes in an adopted dataset (R-Q10), which is caused by the prevalence of within-project choices (R-Q9). The threat to the external validity of studies is observed to be majorly caused by the adoption of a within-project dataset, whose results cannot be generalized when tested on a cross-project. If multicollinearity is discovered through a correlation test of EDA, this could influence the choice of feature selection (R-Q8) technique, depending on the degree of the level of identified correlation. The usual loss function problem without parameter tuning (R-Q6) will likewise be averted with an in-depth understanding of the training set at hand. While the cross-fold validation (R-Q5) helps reduce variance problems, an EDA would further help the tradeoff that must be considered between bias and variance. An EDA would clearly influence the choice of the tradeoff to be considered. Furthermore, the within or closed-project choices in literature indeed inspire the choice of the public repository (R-Q2) to adopt for historical data. The interquartile range (IQR) (through box plot), correlation coefficient (through heat map plot), and other multivariate plots of EDA would be needed to inspire required mitigating techniques to be adopted especially in respect to R-Q1 which reveals the adoption of highly imbalanced training sets across studies.

5.13. Threats to Validity. A prominent threat to the validity of this SLR is in the area of selecting primary studies, especially the quest of efficiently rating the previous studies to get the final fifty-one which constitutes the primary study. To mitigate the threat, the Herculean task of painstaking study of each and every 51 kinds of literature though time-consuming gave a robust and in-depth awareness of the subject of discussion and elements that make up each methodology. No step was speared in the process, just as the same exercise was repeated thrice to ascertain noted claims consecutively. In addition to the automated search of libraries, snowballing was likewise adopted, which is an offshoot of a thorough study of each paper to find other relevant papers. The exclusion and inclusion criteria earlier adopted for this SLR shaped the outcome in commendable ways as most papers adequately answered the research questions, just as the data extraction form plays a significant role in the whole process for quality assessment.

Other threats are the reporting styles of authors, which often encapsulate vital elements of their research away from their model designs. It takes various repetitive studies to identify some germane elements central to the quality assessment of their studies. While some of these elements are not included in their abstracts and keywords, glancing through their introduction and conclusion will yet not reveal their discovery until the entire study was studied prior to the full-scale understudying of the primary studies.

6. Conclusion

This study described an SLR on the deployment of machine learning techniques for software defect severity prediction in the software engineering genre. Eleven specific areas were targeted as an overview of how previous research conducted since eleven years up till 2022 fared with respect to (i) data sampling approach, (ii) choice of software repositories, (iii) most preferred machine learning variant, (iv) prominent learner algorithm and or ensemble, (v) mostly deployed training strategy, (vi) variant of parameter optimization adopted, (vii) most popular machine learning tool, (viii) prominent feature selection algorithm, (ix) choice between *within* and *cross*-project, (x) prominent threats to validity, and (xi) future direction of SDP with respect to threats aforementioned; which all speak to the bias-variance tradeoff considerations and likely threats to validity of proposed methodologies. The study was conducted on papers from 2011 to 2022 comprising of an initial 2653 study population, which eventually resulted in a 52-primary study subset after thorough analysis. The analysis conducted highlighted a handful of observations and limitations in primary studies which are essential to shape future studies in the SDP industry. The following are observed from primary studies: (i) less-deployment of unsupervised learning for SDP studies, (ii) less deployment of balanced data without resampling attempts by the majority of studies that used imbalance data for training, (iii) less-reportage of natural language processing techniques effected on software defect reports before classification, (iv) nonconsideration of multicolinearity problems, and (v) less consideration for cross-project approach which are all essential for a better software defect severity predictive analytics. Therefore, it is believed that this study will be a reference for future works and that the research community will find it as a signpost for better research quality in the software defect severity prediction case.

Data Availability

No data were used to support the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

APC/Open Access funding was provided by Østfold University College, Halden, Norway.

References

[1] M. Khari and N. Kumar, "Comparison of six prioritization techniques for software requirements," *Journal of Global Research in Computer Science*, vol. 4, no. 1, pp. 38–43, 2013.

[2] M. Arif, I. Naseem, M. Moinuddin, and U. M. Al-Saggaf, "Design of an intelligent q-LMS algorithm for tracking a non-stationary channel," *Arabian Journal for Science and Engineering*, vol. 43, no. 6, pp. 2793–2803, 2018.

[3] S. Ali and S. U. Khan, "Critical success factors for software outsourcing partnership (sop): a systematic literature review," in *Proceedings of the 2014 IEEE 9th International Conference on Global Software Engineering*, pp. 153–162, Wolfsburg, Germany, August 2014.

[4] S. Ali, S. Baseer, I. A. Abbasi, B. Alouffi, W. Alosaimi, and J. Huang, "Analyzing the interactions among factors affecting cloud adoption for software testing: a two-stage ISM-ANN approach," *Soft Computing*, vol. 26, no. 16, pp. 8047–8075, 2022.

[5] M. I. Azeem, F. Palomba, L. Shi, and Q. Wang, "Machine learning techniques for code smell detection: a systematic literature review and meta-analysis," *Information and Software Technology*, vol. 108, pp. 115–138, 2019.

[6] G. G. A. Tarhan, "On the use of ontologies in software process assessment: a systematic literature review," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, New York, NY, USA, July 2017.

[7] M. Yaseen, S. Ali, A. Mustapha, and N. Mazhar, "Success factors analysis for requirement elicitation in global software development paradigm: an empirical study," *Journal of Software: Evolution and Process*, vol. 34, no. 7, Article ID e2460, 2022.

[8] A. M. Mustapha, O. T. Arogundade, S. Misra, R. Damasevicius, and R. Maskeliunas, "A systematic literature review on compliance requirements management of business processes," *International Journal of System Assurance Engineering and Management*, vol. 11, no. 3, pp. 561–576, 2020.

[9] F. Akmel, E. Birihanu, and B. Siraj, "A literature review study of software defect prediction using machine learning techniques," *International Journal of Emerging Research in Management and Technology*, vol. 6, no. 6, pp. 300–9359, 2018.

[10] R. S. Wahono, N. Suryana, and S. Ahmad, "A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks," *Journal of Software*, vol. 9, no. 5, 2014.

[11] S. Ali, L. Hongqi, S. U. Khan, Y. Zhongguo, and Z. Liping, "Success factors for software outsourcing partnership management: an exploratory study using systematic literature review," *IEEE Access*, vol. 5, pp. 23589–23612, 2017.

[12] S. Ali, N. Ullah, M. F. Abrar, M. F. Majeed, M. A. Umar, and J. Huang, "Barriers to software outsourcing partnership formation: an exploratory analysis," *IEEE Access*, vol. 7, pp. 164556–164594, 2019.

[13] S. Ali, H. Li, S. U. Khan, M. F. Abrar, and Y. Zhao, "Practitioner's view of barriers to software outsourcing partnership formation: an empirical exploration," *Journal of Software: Evolution and Process*, vol. 32, no. 5, Article ID e2233, 2020.

[14] G. Köksal, I. Batmaz, and M. C. Testik, "A review of data mining applications for quality improvement in manufacturing industry," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13448–13467, 2011.

[15] T. Olaleye, T. Arogundade, and A. A. Abayomi-Alli, "An ensemble predictive analytics of covid-19 infodemic tweets using bag of words," in *Data Science Book for COVID-19*Elsevier, Amsterdam, Netherlands, 2021.

[16] S. Ali, M. Adeel, S. Johar, M. Zeeshan, S. Baseer, and A. Irshad, "Classification and prediction of software incidents using machine learning techniques," *Security and Communication Networks*, vol. 2021, Article ID 9609823, 2116 pages, 2021.

[17] M. kamal, S. Ali, A. Nasir, A. Samad, S. Basser, and A. Irshad, "An automated approach for the prediction of the severity level of bug reports using GPT-2," *Security and Communication Networks*, vol. 2022, Article ID 2892401, 11 pages, 2022.

[18] T. Olaleye, F. Ajayi, A. Aromolaran, I. Solanke, S. Akintunde, and A. JohnBosco, "Semantic relation evaluation of data science articles using network of mention," in *Proceedings of the 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON)*, pp. 1–9, IEEE, Lagos, Nigeria, April 2022.

[19] T. Olaleye, A. Abayomi-Alli, K. Adesemowo, O. T. Arogundade, S. Misra, and U. Kose, "SCLAVOEM: hyper parameter optimization approach to predictive modelling of COVID-19 infodemic tweets using smote and classifier vote ensemble," *Soft Computing*, vol. 15, pp. 1–20, 2022.

[20] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Information and Software Technology*, vol. 55, no. 12, pp. 2049–2075, 2013.

[21] L. H. Son, N. Pritam, M. Khari, R. Kumar, P. T. M. Phuong, and P. H. Thong, "Empirical study of software defect prediction: a systematic mapping," *Symmetry*, vol. 212, 2019.

[22] M. S. Rawat and S. K. Dubey, "Software defect prediction models for quality improvement: a literature study," *International Journal of Computer Science Issues*, vol. 9, no. 5, 2012.

[23] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Applied Soft Computing*, vol. 27, pp. 504–518, 2015.

[24] N. Kalaivani and D. R. Beena, "Overview of software defect prediction using machine learning algorithms," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 20, pp. 3863–3873, 2018.

[25] L. H. Son, N. Pritam, M. Khari, R. Kumar, P. T. M. Phuong, and P. H. Thong, "Empirical study of software defect prediction: a systematic mapping," *Symmetry*, vol. 11, no. 2, p. 212, 2019.

[26] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.

[27] Kitchenham, "Guidelines for performing systematic literature reviews," Keele University, Keele, UK, EBSE-2007-01, 2007.

[28] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, Trondheim, Norway, May, 2014, https://doi.org/10.1145/2601248.2601268.

[29] S. C. B. Kitchenham, "Guidelines for performing systematic literature reviews in software engineering," *School of Computer Science and Mathematics*, Keele University, Keele, UK, 2007.

[30] R. Malhotra, "Severity prediction of software vulnerabilities using textual data," in *International Conference on Recent Trends in Machine Learning, IoT, smart cities and applications*Springer, Singapore, 2021.

[31] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Computing & Applications*, vol. 33, no. 16, pp. 10551–10562, 2021.

[32] C. Prabha and D. N. shivakumar, "Software defect prediction using machine learning techniques," in *Proceedings of the 4th International Conference on Trends in Electronics and Informatics*, Coimbatore, India, September 2020.

[33] A. Kaur and S. G. Jindal, "Text analytics based severity prediction of software bugs for Apache projects," *Int J Syst Assur Eng Manag*, vol. 10, no. 4, pp. 765–782, 2019.

[34] Y. Tan, S. Xu, Z. Wang, T. Zhang, Z. Xu, and X. Luo, "Bug severity prediction using question-and-answer pairs from Stack Overflow," *Journal of Systems and Software*, vol. 165, Article ID 110567, 2020.

[35] M. A. Ihsan Aquil and W. H. W. Ishak, "Predicting software defects using machine learning techniques," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 6609–6616, 2020.

[36] J. Gai, S. Zheng, H. Yu, and H. Yang, "Software defect prediction based on weighted extreme learning machine," *Multiagent and Grid Systems*, vol. 16, no. 1, pp. 67–82, 2020.

[37] H. Alsawalqah, N. Hijazi, M. Eshtay et al., "Software defect prediction using heterogeneous ensemble classification based on segmented patterns," *Applied Sciences*, vol. 10, no. 5, p. 1745, 2020.

[38] A. O. Balogun, S. Basri, S. J. Abdulkadir, and A. S. Hashim, "Performance analysis of feature selection methods in software defect prediction: a search method approach," *Applied Sciences*, vol. 9, no. 13, p. 2764, 2019.

[39] S. Shaikh, L. Changan, M. R. Malik, and M. A. Khan, "Software defect-prone classification using machine learning: a virtual classification study between LibSVM & LibLinear," in *Proceedings of the 13th International Conference on Mathematics, Actuarial Science*, Computer Science and Statistics, Karachi, Pakistan, December 2019.

[40] H. Gupta, L. Kumar, and L. B. M. Neti, "An empirical framework for code smell prediction using extreme learning machine," in *Proceedings of the 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, Jaipur, India, March 2019.

[41] A. Kukkar, R. Mohana, A. Nayyar, J. Kim, and B.-G. Kang, "A novel deep-learning-based bug severity classification technique using convolutional neural networks and random forest with boosting," *Sensors*, vol. 19, no. 13, p. 2964, 2019.

[42] A. Baarah, A. Aloqaily, Z. Salah, M. Zamzeer, and M. Sallam, "Machine learning approaches for predicting the severity level of software bug reports in closed source projects," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, 2019.

[43] F. Wang, J. Ai, and Z. Zou, "Reliability and security," in *Proceedings of the IEEE 19th International Conference on Software Quality*, Sofia, Bulgaria, July 2019.

[44] D. A. Ha, T.-H. Chen, and S.-M. Yuan, "Unsupervised methods for software defect prediction," in *Proceedings of the 10th International Symposium on Information and Communication Technology*, Hanoi, Vietnam, December 2019.

[45] A. Kukkara and R. Mohana, "A supervised bug report classification with incorporate and textual field knowledge," in *Proceedings of the International Conference on Computational Intelligence and Data Science*, Gurugram, India, April 2018.

[46] S. Qiu, L. Lu, S. Jiang, and Y. Guo, "An investigation of imbalanced ensemble learning methods for crossproject defect prediction," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 12, Article ID 1959037, 2019.

[47] F. Wu, X.-Y. Jing, Y. Sun et al., "Cross-project and within-project semisupervised software defect prediction: a unified approach," *IEEE Transactions On Reliability*, vol. 67, no. 2, pp. 581–97, 2018.

[48] S. A. El-Shorbagy, W. M. El-Gammal, and W. M. Abdelmoez, "Using smote and heterogeneous stacking in ensemble learning for software defect prediction," in *Proceedings of the 7th International Conference on Software and Information Engineering*, Cairo, Egypt, December 2018.

[49] S. Huda, K. L. M. Abdelrazek, M. Abdelrazek, A. Ibrahim, H. A.-D. Sultan, and S. Ahmad, "An Ensemble Oversampling Model For Class An Oversampling Ensemble Model For Class," *IEEE Access*, vol. 6, pp. 24184–95, 2018.

[50] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*, Victoria, UK, September 2017.

[51] O. F. Arar and K. Ayan, "A feature dependent naive bayes approach and its application to the software defect prediction problem," *Applied Soft Computing*, vol. 59, pp. 197–209, 2017.

[52] F. A. Fontana and M. Zanoni, "Code smell severity classification using machine learning techniques," *Knowledge-Based Systems*, vol. 128, pp. 43–58, 2017.

[53] V. B. Singh, S. Misra, and M. Sharma, "Bug severity assessment in cross project context and identifying training candidates," *Journal of Information and Knowledge Management*, vol. 16, no. 1, Article ID 1750005, 2017.

[54] H. Alsawalqah, H. Faris, I. Aljarah, L. Alnemer, and N. Alhindawi, "Hybrid SMOTE-ensemble approach for software defect prediction advances in intelligent systems and computing," *Software Engineering Trends and Techniques in Intelligent Systems*, Vol. 575, Springer, Berlin, Germany, 2017.

[55] W. Fu and T. Menzies, "Revisiting unsupervised learning for defect prediction," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE, Singapore, August 2017.

[56] J. Petrie, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an ensemble for software defect prediction based on diversity selection," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM, Bari, Italy, September 2016.

[57] T. Choeikiwong and P. Vateekul, "Improve accuracy of defect severity categorization using semi-supervised approach on imbalanced data sets," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, China, October 2016.

[58] R. Malhotra, "An empirical framework for defect prediction using machine learning techniques with Android software," *Applied Soft Computing*, vol. 49, pp. 1034–1050, 2016.

[59] P. Kaur and C. Singh, "A systematic approach for bug severity classification using machine learning's text mining techniques," *International Journal of Computer Science and Mobile Computing*, vol. 5, no. 7, pp. 523–528, 2016.

[60] R. Jindal, R. Malhotra, and A. Jain, "Prediction of defect severity by mining software project reports," *International Journal of Systems Assurance Engineering and Management*, vol. 8, no. 2, 2016.

[61] J. Petrie, D. Bowes, T. Hall, B. Christianson, and N. Baddoo, "Building an ensemble for software defect prediction," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM, Ciudad Real, Spain, September 2016.

[62] G. Sharma, S. Sharma, and S. Gujral, "A novel way of assessing software bug severity using dictionary of critical terms," in *Proceedings of the 4th International Conference on Eco-Friendly Computing and Communication Systems, ICECCS*, Kurukshetra, India, December 2015.

[63] X. Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and j. Liu, "Dictionary learning based software defect prediction," in *Proceedings of the 36th international conference on software engineering*, Hyderabad, India, June, 2014.

[64] H. Wang, *Software Defects Classification Prediction Based on Mining Software Repository*, Uppsala universitet, Uppsala, Sweden, 2014.

[65] R. Jindal, R. Malhotra, and A. Jain, "Software defect prediction using neural networks," in *Proceedings of the 3rd International Conference on Reliability, Infocom Technologies and Optimization*, Noida, India, October 2014.

[66] O. C. Duksan Ryu, "Improving prediction robustness of VAB-SVM for cross-project defect prediction," in *Proceedings of the IEEE 17th International Conference on Computational Science and Engineering*, Lviv, Ukraine, November 2014.

[67] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Folleco, "An empirical study of the classification performance of learners on imbalanced and noisy software quality data," *Information Sciences*, vol. 259, pp. 571–595, 2014.

[68] J. Ren, K. Qin, Y. Ma, and G. Luo, “On software defect prediction using machine learning,” *Journal of Applied Mathematics*, vol. 2014, Article ID 785435, 8 pages, 2014.

[69] R. Malhotra, “Comparative analysis of statistical and machine learning methods,” *Applied Soft Computing Predicting Faulty Modules*, vol. 21, pp. 286–297, 2014.

[70] A. Sudha, “Software defect prediction system using,” *International Journal of Recent Technology and Engineering*, vol. 3, no. 2, pp. 2277–3878, 2014.

[71] T. M. Khoshgoftaar, K. Gao, A. Napolitano, and R. wald, “A comparative study of iterative and non-iterative feature selection techniques for software defect prediction,” *Information System Frontiers*, vol. 16, no. 5, 2013.

[72] M. S. Naidu and D. Geethanjali, “Classification of defects in software using decision tree algorithm,” *International Journal of Engineering Science and Technology*, vol. 5, no. 6, pp. 1332–1340, 2013.

[73] A. Chug and S. Dhall, *Software Defect Prediction Using Supervised Learning Algorithm and Unsupervised Learning Algorithm*, Institution of Engineering and Technology, London, UK, 2013.

[74] K. K. Chaturvedi and V. B. Singh, “Determining bug severity using machine learning techniques,” in *Proceedings of the 2012 CSI 6th international conference on software engineering (CONSEG)*, Indore, India, September 2019.

[75] R. Verma and A. Gupta, “Software defect prediction using two level data pre-processing,” in *Proceedings of the International Conference on Recent Advances in Computing and Software Systems*, Hyderabad, India, August 2012.

[76] D. Bowes, T. Hall, and J. Petri´c, “Software defect prediction: do different classifiers find the same defects?” *Software Quality Journal*, vol. 26, no. 5, 2012.

[77] Z. Sun, Q. Song, and X. Zhu, “Using coding-based ensemble learning to improve software defect prediction,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1806–1817, 2012.

[78] C. Z. Yang, C. Kao, and I.-X. Chen, “An empirical study on improving severity prediction of defect reports using feature selection,” in *Proceedings of the Asia-Pacific Software Engineering Conference*, Hong Kong, China, December 2012.

[79] Y. Peng, G. Kou, G. Wang, W. Wu, and Y. Shi, “Ensemble of software defect predictors: an ahp-based evaluation method,” *International Journal of Information Technology and Decision Making*, vol. 10, no. 1, pp. 187–206, 2011.

[80] F. H. Alshammari, “Software defect prediction and analysis using enhanced random forest (extrf) technique: a business process management and improvement concept in iot-based application processing environment,” *Mobile Information Systems*, vol. 2022, Article ID 2522202, 11 pages, 2022.