American Academic Scientific Research Journal for Engineering, Technology, and Sciences
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402
http://asrjetsjournal.org/

Systematic Literature Review on the Machine Learning Approach in Software Engineering

Marcelo Cerqueira[a*], Paulo Silva[b], Sergio Fernandes[c]

[a,b,c]*Universidade Salvador (UNIFACS), Av. Tancredo Neves, 2131, Salvador-BA CEP: 41820-021, Brasil*

[a]*Email: marcelo.cerqueira@icloud.com* , [b]*Email: paulo.caetano@unifacs.br*, [c]*Email: sergiomfernandes63@gmail.com*

Abstract

The application of machine learning solutions in software engineering tools and processes can bring significant benefits to software engineering processes, as well as to processes results analysis. There are few primary, secondary, and tertiary studies referring to machine learning applications in software engineering. The apparent scarcity of this type of research makes it difficult to develop specific solutions for software engineering areas and processes. Thus, it is necessary to investigate and understand how the use of machine learning in software engineering is reported in the literature. This work aims to carry out a systematic literature review on the machine learning approach in software engineering. The search strategy resulted in 1725 articles, of which 54 articles were about empirical studies. The studies were grouped into four themes: the main machine learning algorithms and/or frameworks applied in software engineering; the software engineering activities in which these algorithms and/or machine learning frameworks are applied; the main types of application of these algorithms and/or frameworks; and the main results obtained with the application of these algorithms and/or frameworks. The results obtained indicate that the following algorithms are used: Support Vector Machine, Random Forest, Decision Tree and Naive Bayes and applied mainly in software testing and planning activities. Defect prediction and effort estimation are the main types of application of these algorithms and improvement in performance and accuracy of defect prediction and cost reduction are the main results obtained with the application of these algorithms in software engineering.

Keywords: Machine Learning; Software Engineering; Software Engineering Processes; Software Engineering Activities; Machine Learning Algorithms.

---

* Corresponding author.

1. Introduction

Machine learning (MA) is seen by some authors as a statistical technique overlap with the following areas: applied mathematics, information engineering and biological computing [1]. According to Clifton and his colleagues (2013), the Machine learning objective is to identify patterns in the data and then perform inferences using these identified patterns.

In recent years, many innovations have been created using machine learning: autonomous vehicles, voice recognition, data mining, biometrics, among other solutions. As a result, the demand for intelligent systems had a relevant growth in the market and in the scientific field [2]. These systems use implicit (not previously programmed) algorithms for pattern detection that involve various other disciplines, e.g. calculus, linear algebra, statistics, probability, information theory and neurobiology. These disciplines are used in intelligent algorithms build and are part of the machine learning area.

According to Mitchell (1997), machine learning occurs when the computer learns by improving the performance of a class of tasks, which are measured statistically. The process of learning a computer program involves three steps. The first step is task classes definition that will be learned. The second step is the measurements definition that will be performed to identify whether there was an improvement in the performance of each task by the computer. The third stage is the set of trainings to obtain learning and, consequently, improve performance. Thus, the implementation of machine learning involves a set of tasks, performance measurement of these tasks and a set of training to obtain experience in performing these tasks [2].

The software engineering use in the development of information systems has brought many benefits to companies [3]. However, only a few areas of software engineering have benefited from using machine learning approaches in their tools and processes [4,6].

In the academic setting, systematic reviews related to the ML application in software engineering have the main focus on testing and defect prediction activities, as well as there are few secondary studies characterizing the approaches of these applications in other areas of software engineering [7]. Therefore, it is necessary to investigate machine learning approaches in software engineering, in general aspects, to understand how the machine learning use in the software development process is reported in the literature.

This work has as general objective a systematic literature review on the machine learning approach in software engineering. To achieve the general objective of this work, it is necessary to reach the following specific objectives: collect, group, synthesize and analyze primary studies relevant to the characterization of ML application approaches in software engineering during the period from 2009 to 2021.

With the dissemination of the results obtained in this study, it will be possible to understand its importance for the area of software engineering, characterizing the state of art of the machine learning approach in software engineering and identifying research gaps on this topic. With the characterization of these research gaps, it will be possible to develop new research, experiments and more specific primary studies to fill the spaces identified in this work. But it will also be possible to develop more specific machine learning solutions in software

engineering. With the development of machine learning solutions in software engineering, we will be able to obtain the following benefits: cost reduction with smarter tools use, improvement in effort estimation of development teams, more efficient prediction of codes errors, reduction of costs in software testing processes and improvement in the application of design patterns with cognitive analysis tools.

This work is organized from this section. In section 2, Related Work, works that present a literature review on machine learning use in software engineering will be discussed. In section 3, the methodology used in this work will be presented. In section 4, the process of selecting studies and extracting data will be presented. In section 5, the results obtained will be presented. Finally, in section 6, the final considerations of this work are presented.

2. Related Work

MALHOTRA (2015) performed a systematic literature review (RSL) to identify machine learning application in predicting failures in the development process. This study was carried out with data extracted during the period from 1991 to 2013. The study evaluated the ability of machine learning techniques to predict software failures. Malhotra also compared the performance of the techniques found that used machine learning with other techniques using statistical inference. The result of the study was that machine learning use in software failures prediction contributes substantially to codes errors reduction. Furthermore, the study came to the conclusion that the application of machine learning to error prediction still needs further research [8].

WEN and his colleagues (2012) carried out a literature review to investigate the prediction of efforts required to develop a software, using machine learning techniques to improve the accuracy of the estimates. For this study, data were extracted from the period 1991 to 2010. The conclusion of this study was that the application of machine learning for effort estimation in software development is promising. However, this type of study for software engineering still has a limited number of primary studies [10].

ALSOLAI and ROPER (2019) developed a literature review to investigate maintainability prediction of systems built with the object-oriented paradigm, using machine learning techniques. In this review, 56 works from the period 1991 to 2018 were retrieved [12].

AFZAL and TORKAR (2011) conducted a literature review to investigate the use of genetic programming algorithms to predict code errors and to improve software quality. For this study, papers between 1995 and 2008 were considered. 23 relevant primary studies were found, which confirmed that the use of genetic programming algorithms can improve code errors prediction, as well as help to improve the quality of software [13]. From the related works discussed above, it can be stated that there is a lack of works that present systematic literature reviews to characterize the machine learning approach in software engineering. Thus, in this work, relevant primary studies identified in academic literature that identify the machine learning approach in software engineering will be extracted, categorized, and synthesized.

3. Methodology

A Systematic Review Protocol was elaborated according to the guidelines of [14]. In this protocol, the work

title, the research context (background), the reason for carrying out the studies, the main research question, the secondary questions, search process definition and the search string are evaluated. Furthermore, this protocol also includes the primary studies selection process, the inclusion and exclusion criteria, the primary studies process of assessing the quality, the data extraction process, the synthesis process, and study limitations. The protocol developed was also evaluated by two experts in the field of software engineering research (See Appendix B).

Four research questions were created, the last three being derived from the first. These research questions have the following objectives: to characterize the main machine learning algorithms and/or frameworks applied in software engineering; identify the main software engineering activities in which these algorithms and/or frameworks are applied; describe the main types of application of these algorithms and/or frameworks in software engineering and characterize the main results and benefits of applying these algorithms and/or frameworks within software engineering. The research questions are listed in Table 01 below.

Table 1: Research Questions.

Research Questions	Description
Research Question 01	What are the main machine learning algorithms and/or frameworks used in software engineering?
Research Question 02	What are the software engineering activities in which machine learning algorithms and/or frameworks are applied?
Research Question 03	What are the main application types of machine learning algorithms and/or frameworks in software engineering?
Research Questions 04	What are the main results achieved of applying machine learning algorithms and/or frameworks in software engineering?

Regarding the inclusion and exclusion criteria, those primary studies that presented machine learning approaches and applications in software engineering, including qualitative and quantitative primary studies, were considered eligible.

Papers in English and Portuguese published during the period 2009 to 2021 were considered. Primary papers that did not address the main objective of this work were excluded, as were papers that failed the quality assessment process (described in the next paragraphs), case studies and papers published outside the specified period.

Regarding the research repositories considered, the following research libraries were chosen:

- ACM Digital Library
- IEEE Xplore
- ScienceDirect – Elsevier
- SpringerLink
- Wiley Inter Science Journal Finder
- Google Scholar

These libraries were selected for being repositories of works from the academic computing community. Studies from manual searches carried out in academic journals were also considered, as well as studies extracted by the Snowballing technique [15].

The papers selection process was divided according to the phases specified in Figure 01.

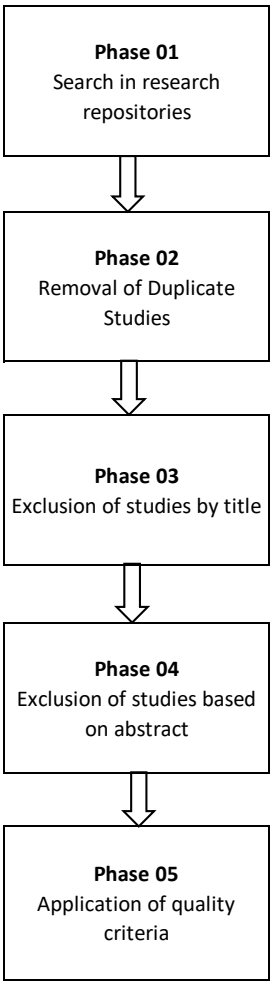

Figure 1: Systematic Review papers selection phases.

The search process in the repositories used the descriptors below to create the search string:

- (1) machine learning AND software engineering
- (2) artificial Intelligence AND software engineering
- (3) reinforcement learning AND software engineering
- (4) Deep learning AND software engineering
- (5) statistical learning AND software engineering
- (6) (machine learning OR learning) AND (test OR requirement OR process OR software OR software engineering OR development OR software fault OR software planning OR software design OR software quality).

After defining the descriptors above, these terms were combined using the Boolean operator "OR" to form a search string that, in turn, returned papers that at least included one of the terms specified in the search string. Below is the search string that was used:

1 OR 2 OR 3 OR 4 OR 5 OR 6

During the search process, abstracts, magazine editorials, interviews, letters, blog discussions, documents from workshops and posters were not considered. The retrieved papers were stored in Elsevier's Mendeley Reference Manager (https://www.mendeley.com). Mendeley was also used to remove retrieved papers more than once in the search process. After the analysis in Mendeley, the articles were imported into a spreadsheet in MS-Excel with the respective metadata (authors, citations, abstracts, publication data). The R language (https://www.r-project.org) was used in the process of grouping the results, in the use of statistical functions and in the generation of graphs.

In phase 02 of this literature review, the titles were analyzed to remove papers that are not related to the main theme of this work. In phase 03, an analysis was performed on the abstracts to select relevant papers. In phase 04, studies qualitative analysis was carried out using the previously defined quality criteria. The quality criteria considered in this work evaluated the research methods used in the studies, following the guidance of [14], [16]. In addition, the quality of the content presented in the papers was evaluated, as well as the studies relevance to the research. The quality assessment criteria are presented below:

- The evaluated study reports empirical work or only lessons learned, based on expert opinions.
- The answers to the research questions are clear in the paper.
- There is a clear description of the context in which the paper was applied or where the research was carried out.
- The research methods used in the paper are in accordance with academically applied methodologies.
- There is a clear description of the methods used in analyzing the results.
- The evaluated study provides value to the research.

The quality criteria answers, specified above, were mapped into numerical values to carry out results quantitative analysis. In the analysis process, as well as in the synthesis of results, the data that were stored in Excel were exported using the R language to perform the statistical analyses, using inferential and descriptive methods. Meta-analysis methods were also used as recommended by [14], [16]. The statistical functions used were mean, standard deviation, Kappa coefficient and correlation coefficient (when necessary). The analyzes were performed in the R Studio environment (R-Language Code Development Environment – Available: https://www.rstudio.com). The research questions were answered with the results obtained from the results synthesis process.

4. Process Papers Selection and Data Extraction

The article selection process was applied from the phases illustrated in Figure 1, presented in Section 3

(Methodology). Figure 2 illustrates the quantitative result obtained from the article selection process in each search repository.

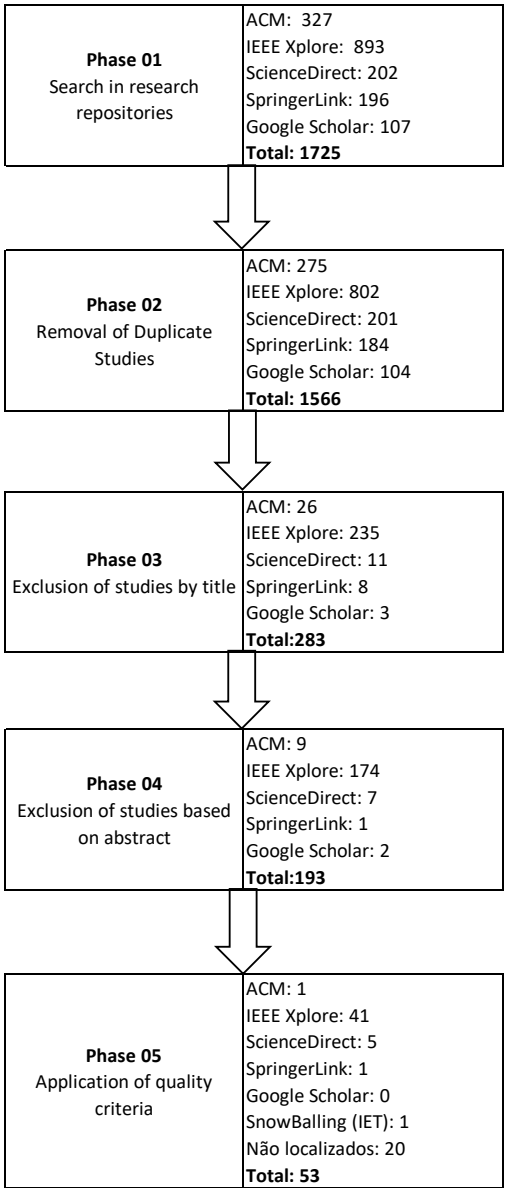

Figure 1: Paper selection phases.

5. Results

There has been an increase in publications in recent years, confirming the work of [17], who made similar analysis related to machine learning application in software testing. However, in 2020 and 2021, there was a significant drop compared to 2018 and 2019. This drop may be associated with the impacts of the Covid-19 Pandemic, which reduced the number of conferences due to sanitary restrictions. Figure 03 shows the number of publications per year.

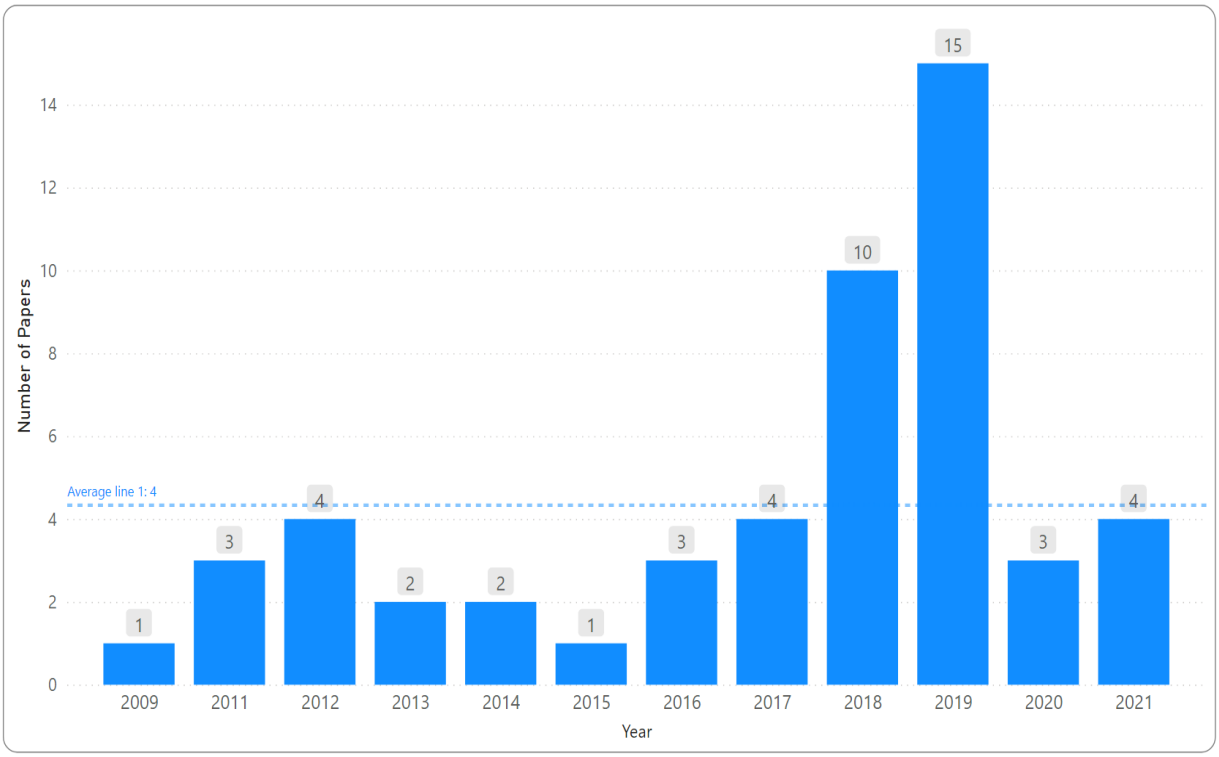

Figure 2: Papers distribution by publication year.

Based on the methodology used by Durelli (2019), descriptive statistics and frequency analysis functions were used to analyze the answers to the research questions. The frequency function is shown in the histograms of Figures 4, 5, 6, 7 and 8 and corresponds to the number of occurrences by the total of them, however, classified by the highest occurrence of the results (i,e, mode). To analyze research questions answers, the same approach applied in the works [18] and [19] was used, which consists of presenting the results indexed by research question. Below are the results obtained for each research question.

5.1. What are the main machine learning algorithms and/or frameworks used in software engineering?

This research question purpose is characterizing the main machine learning algorithms and/or frameworks applied in software engineering processes or activities.

As can be seen in Figure 4, the "Support Vector Machine" algorithm was the most applied in software engineering processes and activities (17.81% of the articles), followed by "Random Forest" (12.33%), "Decision Tree" (8.22%) and "Naive Bayes (6.85%).

The Weka Framework, developed in Java by the Waikato University in New Zealand, was identified in 5.48% of the analyzed articles, while the "K-Nearest NeighBor" and "Linear Regression" algorithms had the same results, i.e. 4.11%. As for the algorithms based on neural networks, the "Artificial Neural Network", "Feedforward Neural Network", "Multi-Layer Perceptron" and "Radial Basis Function Neural Network" were used in the same number of works (2.74%). The algorithms "Back Propagation", "Convolutional Neural Network", "Extreme Learning Machine", "Fourier Algorithm", "Fuzzy Learning", "Genetic Algorithm",

"GFsSDAEsTSE Algorithm", "Greedy Forward Selection", "K-Star Algorithm ", MATLAB ANN FrameWork, Daikon FrameWork, "Multi Objective Genetic Algorithm", "Probabilistic Reasoning Algorithm", "Recurrent Neural Network Algorithm", "Reinforcement Learning Algorithm", "Single-Layer Perceptron Algorithm", "Support Vector Regression" and " TF-IDF Weighting Algorithm" were identified in only 1 article, i.e. 1.37% of the total.

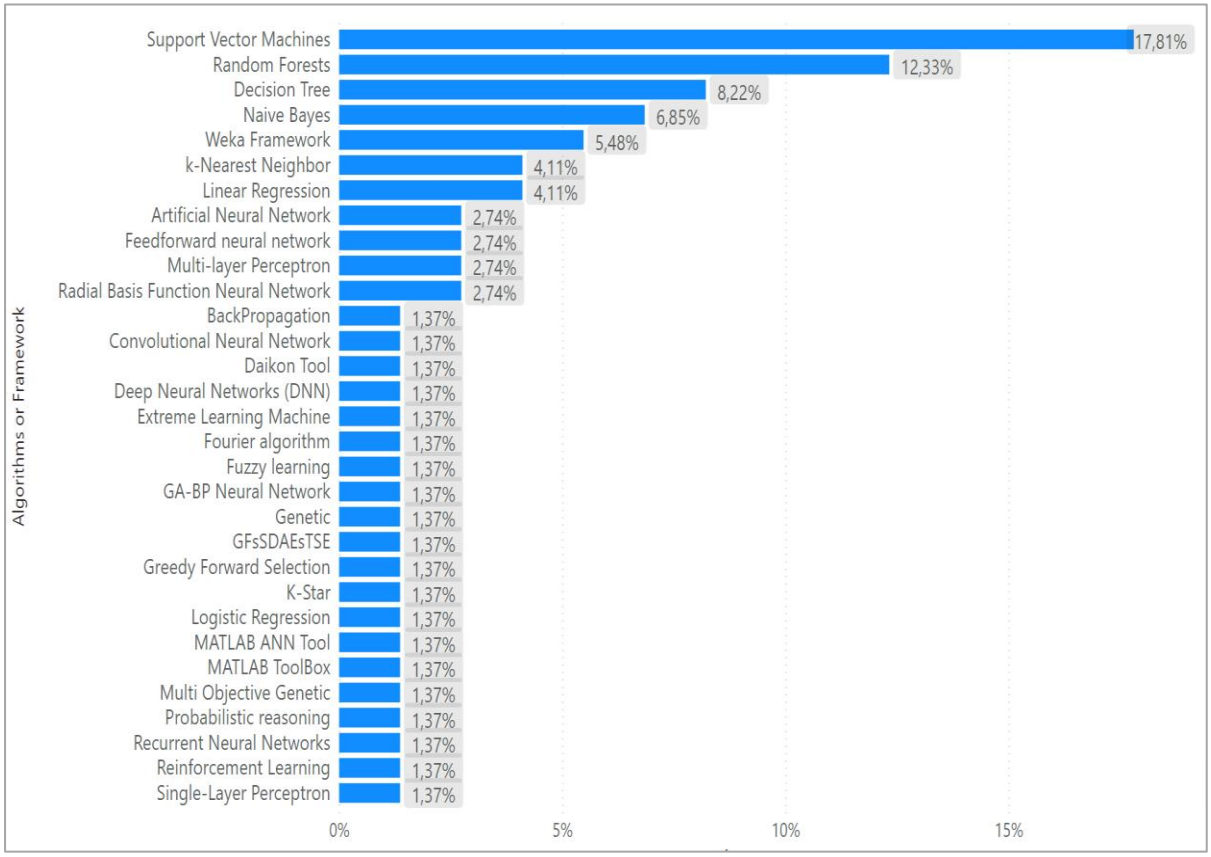

Figure 3: Papers distribution in relation to algorithms or frameworks**.**

Table 2 shows the distribution of papers by algorithms and Appendix A shows the list of papers found in the literature review. It is observed that in some works there was the occurrence of primary studies that evaluated two or more algorithms, therefore, the count of total occurrences of papers is higher than the 53 papers retrieved in the literature review. This fact is repeated in the other tables of this analysis.

The Support Vector Machines algorithm is used in classification or regression problems. Furthermore, it is the most available algorithm in "off-the-shelf", that is, it can be found in several frameworks on the market [20]. The Random Forest Algorithm is similar to Decision Tree, however with the more diverse K-tree to reduce the variance, it is also used in the classification or regression problems. The Decision Tree algorithm is most used in classifying instances from the root to the last layers of each node (a node specifies tests that are performed on attributes of the instance). The Naive Bayes algorithm is widely used in text classification and uses the probability theory of the Bayes Theorem [21]. The Weka Framework is a tool that contains a collection of machine learning algorithms (Decision Tree, K-Nearest Neighbors, Linear Regression and Naive Bayas) being

used mainly in data mining [22].

Table 2: Papers distribution by algorithms.

Algorithms	Primary Studies
Artificial Neural Network Algorithm	15, 16,
BackPropagation Algorithm	29,
Convolutional Neural Network	34,
Daikon Tool	36,
Decision Tree Algorithm	2, 14, 33, 42, 47, 50,
Deep Neural Network (DNN)	51,
Extreme Learning Machine	2,
Feedforward neural network	17, 37,
Fourier algorithm	30,
Fuzzy learning algorithms	24,
GA-BP Neural Network	52,
Genetic algorithm	31,
GFsSDAEsTSE Algorithm	45,
Greedy Forward Selection Algorithm	45,
k-Nearest Neighbor Algorithm	1, 13, 22,
K-Star Algorithm Algorithm	15,
Linear Regression Algorithm	7, 15, 28,
Logistic Regression	53,
MATLAB ANN Tool	53,
MATLAB ToolBox	52,
Multi Objective Genetic algorithm	40,
Multi-layer Perceptron Algorithm	9, 47,
Naive Bayes algorithm	5, 22, 28, 33, 48,
New Hybrid Nonlinear MD Model-DT Classification	18,
Probabilistic reasoning algorithm	19,
Radial Basis Function Neural Network	2, 4,
Random Forests Algorithm	6, 8, 11, 33, 35, 39, 45, 50,
Recurrent Neural Networks Algorithm	28, 32,
Reinforcement Learning Algorithm	10,
Single-Layer Perceptron Algorithms	23,
Support Vector Machines Algorithm	3, 9, 12, 15, 20, 21, 22, 25, 26, 38, 47,
Support Vetor Regression Algorithm	27,
TF-IDF weighting algorithm	43,
Weka Framework	12, 15, 39, 44,

The algorithms Support Vector Machine, Weka Framework, Random Forest, Naive Bayes and Decision Tree were the most reported in selected articles.

5.2. What are the software engineering activities in which machine learning algorithms and/or frameworks are applied?

The Research Question 02 goal is to identify in which software engineering activities the algorithms identified in Section 5.1 are applied. Figure 2 illustrates that 56.52% of papers reported using machine learning algorithms in software testing activities. It is possible to identify that 26.09% of reported papers their use in software planning activities and 7.25% in software project activities. The use of machine learning algorithms in software requirements activities was reported by 4.35%. Only 2.90% indicated the use of this category of algorithms in software development activities and 1.45% in all software quality processes.

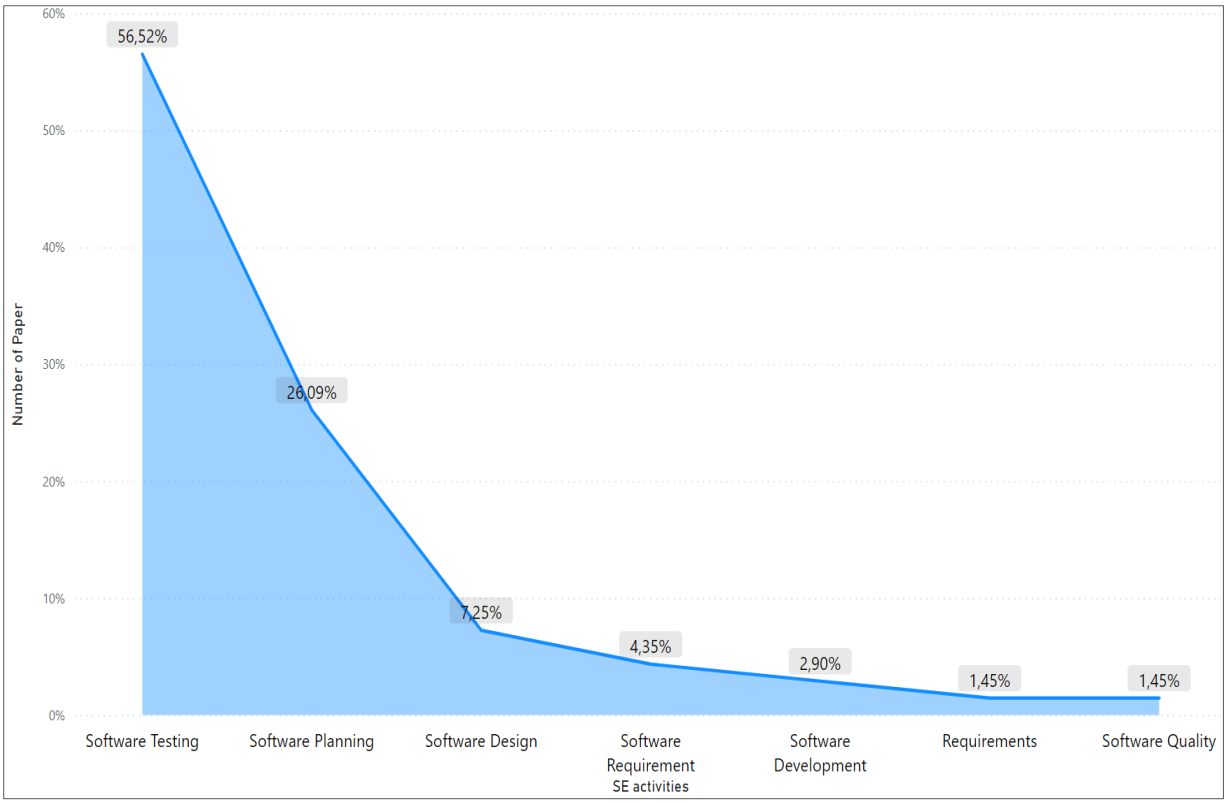

Figure 4: Papers distribution by software development process activities

In Table 3, the distributions of activities and/or software engineering processes by algorithm are presented. It can be identified that the Decision Tree algorithm participates in almost all software engineering activities. It is inferred that the reasons for this are ease of interpretation and adaptability of the Decision Tree algorithm in relation to other algorithms. Another algorithm that played a large part in all software engineering activities was Naive Bayes algorithm. The justification for this is related to the speed and efficiency of this algorithm in relation to others. In Table 4, references related to the application of machine learning algorithms in software engineering activities can be found. As can be seen, software testing and software planning activities had the greatest applications of machine learning algorithms.

Table 3: Algorithm Distribution by Software Engineering Activities.

SE Activities	Algorithms
Software Design	Artificial Neural Network Algorithm
	Decision Tree Algorithm
	Multi-layer Perceptron Algorithm
	Support Vector Machines Algorithm
Software Development	Feedforward neural network
	Support Vector Machines Algorithm
Software Planning	Artificial Neural Network Algorithm
	Decision Tree Algorithm
	Extreme Learning Machine
	Feedforward neural network
	Fuzzy learning algorithms
	K-Star Algorithm Algorithm
	Linear Regression Algorithm
	Naive Bayes algorithm
	Radial Basis Function Neural Network
	Random Forests Algorithm
	Support Vector Machines Algorithm
	TF-IDF weighting algorithm
	Weka Framework
Software Quality	Decision Tree Algorithm
Software Requirement	Naive Bayes algorithm
	Reinforcement Learning Algorithm
	Support Vector Machines Algorithm
Software Testing	BackPropagation Algorithm
	Convolutional Neural Network
	Daikon Tool
	Decision Tree Algorithm
	Fourier algorithm
	Genetic algorithm
	GFsSDAEsTSE Algorithm
	Greedy Forward Selection Algorithm
	k-Nearest Neighbor Algorithm
	Linear Regression Algorithm
	MATLAB ANN Tool
	Multi Objective Genetic algorithm
	Multi-layer Perceptron Algorithm
	Naive Bayes algorithm
	New Hybrid Nonlinear MD Model-DT Classification
	Probabilistic reasoning algorithm
	Radial Basis Function Neural Network
	Random Forests Algorithm
	Recurrent Neural Networks Algorithm
	Single-Layer Perceptron Algorithms
	Support Vector Machines Algorithm
	Support Vetor Regression Algorithm
	Weka Framework

Table 4: Papers distribution by Software Activities.

SE Activities	Software Design	Software Development	Software Planning	Software Quality	Software Requirement	Software Testing
Primary Studies	16	17	2	14	5	1
	21	26	15		10	3
	47		24		20	4
			25		51	6
			28			7
			37			8
			38			9
			43			11
			44			12
			53			13
						18
						19
						22
						23
						27
						29
						30
						31
						32
						33
						34
						35
						36
						39
						40
						42
						45
						48
						49

In Table 4 we can identify that software development, design, quality and software requirements activities had few applications of machine learning algorithms, despite being important in the area of software engineering. The software quality activity had only one paper published, development two papers, and software project only three papers published. Thus, we can conjecture that the focus of the application of machine learning algorithms is only on software testing and planning activities. The targeting of machine learning algorithms specifically in testing and planning activities may be related to the attempt to reduce human effort in the execution of these activities, due to the repetition of different types of tests and, regard to the application in planning activities, may be related to the attempt to reduce human effort in duration estimation, which also demand repetitive processes. Another factor that would justify the machine learning algorithms application in planning activities may be the need to increase the estimates accuracy. Although it also requires human effort, development activities and requirements analysis do not require repetitive processes, which can justify the low interest in

research on machine learning algorithms in these activities.

5.3. What are the main application types of machine learning algorithms and/or frameworks in software engineering?

This research question objective is to characterize the main types of machine learning algorithms applications in software engineering. Table 5 shows the papers distribution by type of application.

Table 5: Papers distribution by application types.

Types of Applications	Primary Studies
Anti-Patterns Detection	47
Software Quality Prediction	14
Software Project Quality Assessment	12
Bad Smell Prediction	39
Bad Smell Detection	48
Task reusability	17
Determining the Defect Severity Class	22
Testing Process Improvement	31
Improved Defect Prediction Processes	33, 34
Improved Web Application Testing Activities	1
Classification of Non-Functional Requirements	20
Software Failure Prediction	7, 45
Software Project Duration Prediction	37
Reduction of Software Project Duration	25
Requirements Elicitation	5, 31
Requirements Traceability	10
Component Reusability	16
Defect Detection	3, 42
Defect Prediction	4, 6, 8, 9, 11, 13, 18, 19, 23, 27, 29, 30, 32, 35, 36, 49
Estimation of Development Activities Effort	2, 15, 24, 43, 44
Software Maintainability Prediction	21
Estimate Prediction of Development Activities	28
Software Process Evaluation	26
Software Project Risk Prediction	38
Reduction of Test Cases in Web Applications	40

In Table 5, we can identify that the main types of machine learning algorithms applications in software engineering are software defect prediction and development activities effort estimation. These results confirm the results obtained in Research Question 02, which obtained the highest frequencies in software testing and planning activities, as defect prediction and effort estimation types are respectively related to software testing activities and software planning.

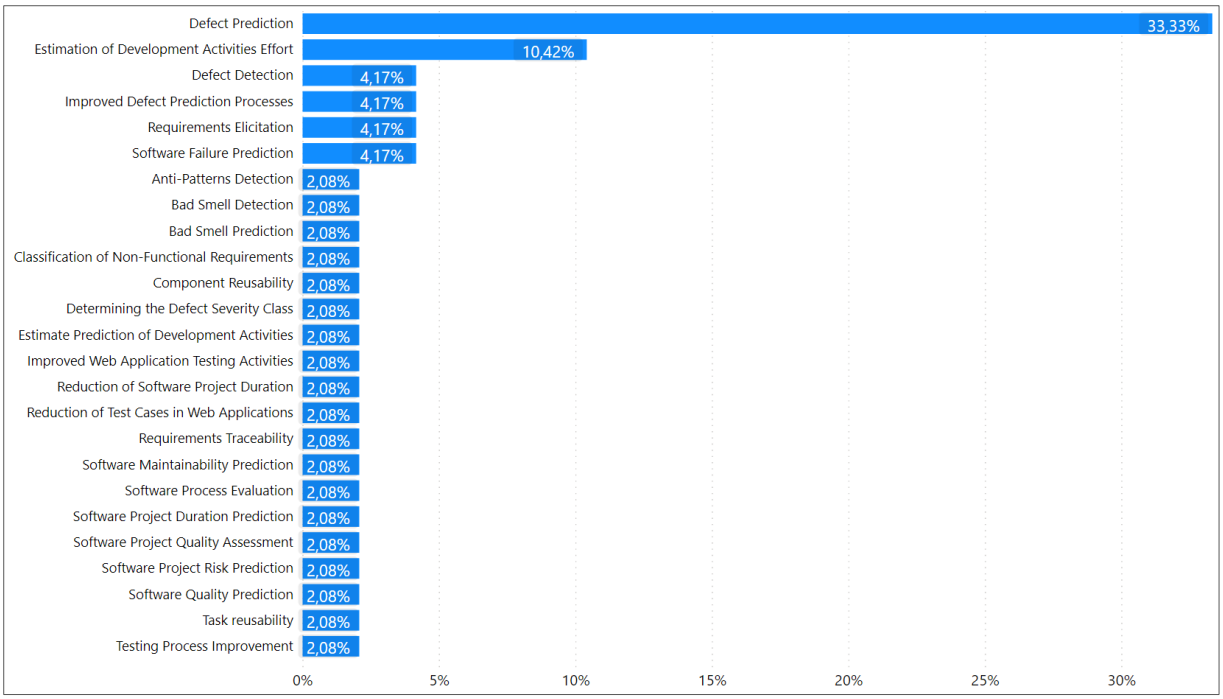

Figure 5: Papers distribution by application types.

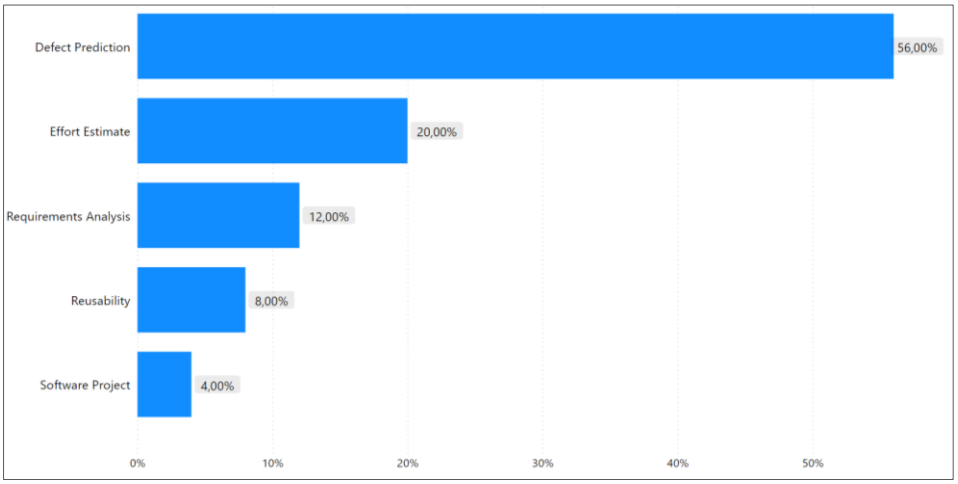

Figure 6: Grouped summary of main application types distributed by papers.

The Figure 7 is a grouped summary classification of Figure 6. It was necessary to show the results in a more concise visualization of main application types by distributed papers. For example, in Figure 6, Defect Prediction application type got a percentual of 33,33%. However, Bad Smell Prediction obtained 2,08%. Both are related to Defect Prediction. Then, in Figure 7, these results were grouped in five main categories (Defect Prediction, Effort Estimate, Requirements, Reusability and Software Project) that increased all percentual of these categories because of summarization of the numbers in five categories. As results of grouped summary, we can see in the Figure 7 that the application of machine learning algorithms in software engineering is more restricted to software defect prediction (56%) and effort estimation (20%).

5.4. What are the main results achieved by the application of algorithms and/or machine learning frameworks in software engineering?

This section has as main objective to identify the main results obtained with machine learning algorithms application. Table 6 shows the results obtained from the papers.

Table 6: Papers distribution by algorithm application results

Types of Applications	Primary Studies
Cost reduction	31, 40,
Improved Defect Prediction	3, 4, 14, 25, 26,
Improving Defect Prediction Accuracy	7, 12, 15, 21, 23, 28, 30, 45, 48,
Error Rate Reduction	28, 42,
Improved Performance in Defect Prediction	9, 13, 17, 22, 28, 29, 32, 34, 35, 37, 38,
Accuracy of Defects and Bad Smell Predictions	11, 39,
Reduction in Software Maintenance and Testing Costs	8,
Improved Estimates for Large Projects	2,
Improvement in Small and Medium Project Estimates	2, 44,
Improved Accuracy of Predictions	10, 18,
Reduction of Manually Elaboration of Test Cases	1,
Reduction of Task Duration	45,
Improved Anti-Pattern Detection	47,
Improvement in Software Quality	6, 33,
Improved Failure Prediction	36,
Reduced Cost and Time of Development Activities	16, 33, 43,
Improved Software Testing Activities	27,

In Table 6, we can identify that most papers reported positive results (e.g. cost reduction, improved accuracy in defect prediction, time reduction) in the application of machine learning algorithms in software engineering. The main results identified and that showed improvement in the software development process were improvement in defect prediction performance (11 articles), improvement in defect prediction accuracy (9 articles), improvement in defect prediction (5 articles) and reduction of the cost/time of development activities (3 articles), which corroborates the result identified in Section 5.3, that the main types of machine learning algorithms applications in engineering software are software defect prediction and development activities effort estimation. The results of accuracy in defect and bad-smell predictions, improvement in project estimates, and improvement in software quality were reported in only two papers. Some of the papers reported more than one result, being, therefore, replicated in more than one type of application in Table 6. In Figure 8, these results are presented in the form of a histogram.

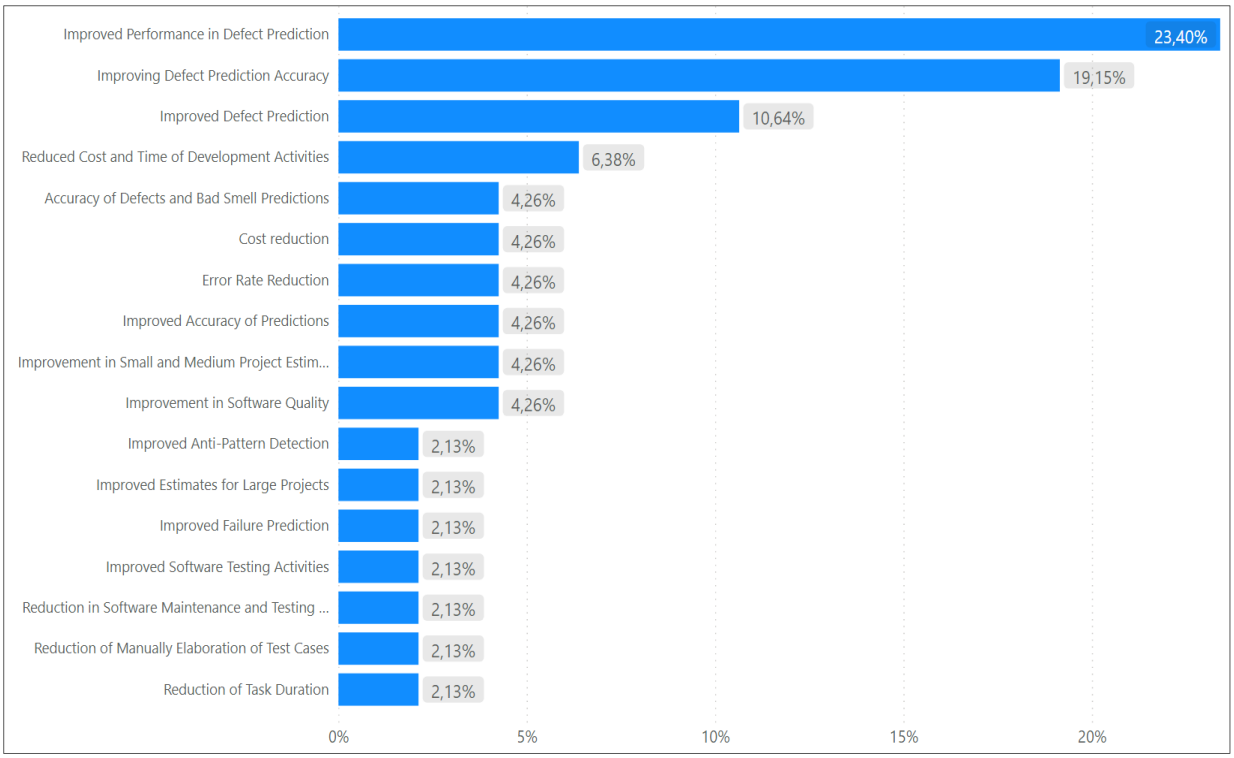

Figure 7: Papers distribution by results achieved.

5.5. Results discussion

This work characterized, in Sections 5.1, 5.2, 5.3 and 5.4, the main approaches of machine learning algorithms in software engineering. From the four research questions, the following questions were answered:

- What algorithms are applied?
- Where are they applied?
- How are they applied?
- What are the results of these applications?

Below are the answers to the research questions:

RQ-01

What are the main machine learning algorithms and/or frameworks used in software engineering?

Answer: The main machine learning algorithms/framework applied in software engineering are: Support Vector Machine, Random Forests, Decision Tree, Naive Bayes, Weka Framework, K-Nearest Neighbor, Linear Regression, Artificial Neural Network, Feedforward, Multi- layer Perceptron and Radial Basis Function.

RQ-02

What are the software engineering activities in which machine learning algorithms and/or frameworks are applied?

Answer: The main activities are Software Testing, Software Planning and Software Design.

RQ-03

What are the main application types of machine learning algorithms and/or frameworks in software engineering?

Answer: The main applications of machine learning algorithms in software engineering are in defect prediction, in software activity effort estimation, in defect detection, in the improvement of defect prediction processes and in software failure prediction.

QP-04

What are the main results achieved of applying machine learning algorithms and/or frameworks in software engineering?

Answer: The main results machine learning algorithms application in software engineering are: improved performance in defect prediction, improved accuracy in defect prediction, improved defect prediction and reduced activities cost and duration.

The results obtained differ from the results of the work [17], which identified only the Artificial Neural Network, Bayesian Algorithm, Clustering Algorithm, Decision Tree Algorithm, Ensemble Algorithm, Instance Based Algorithm, Learning Finite Automata and Regression Algorithm algorithms. However, in this work the following algorithms were identified that were not reported in [17]: Support Vector Machines, Random Forests, Naive Bayes, K-Nearest Neighbor, FeedForward Neural Network, Back Propagation, Fourier Algorithms, Greedy Forward Selection and Probabilistic Reasoning. However, this difference in the number of identified algorithms could be related to the type of methodology applied in both works and the number of digital libraries examined. In work [17], only four digital libraries were selected (IEEE Digital Library, ACM Digital Library, SpringerLink, and ScienceDirect). Whereas in this work, we selected five digital libraries (ACM, IEEE, ScienceDirect, SpringerLink, and Google Scholar).

The results achieved confirmed those obtained in the work [8], which is related to code errors reduction through software failure prediction, using machine learning algorithms. It was also confirmed the results obtained in the research [9], which is related to improve the accuracy of effort estimates through machine learning.

No negative results were reported in the analyzed papers. However, it is recommended as future research to verify the existence of bias in the primary studies analyzed. It is recommended to direct new primary studies in

the research gaps identified in this work, i.e. requirements analysis, coding and software design activities which, according to Figures 4, 5, 6 and 7, no significant publications amount was found. Currently, publications academic focus, involving machine learning algorithms application in software engineering, is related to defect prediction and effort estimation activities. However, other important areas in software engineering, e.g. requirements analysis and software design, remain without the benefits of applying these algorithms.

5.6. Threat to validity

Different factors can influence the results of this systematic review, for example, the research repositories selected in the study, the search string created for the study, and the study review process. To mitigate these influences, the recommendations specified in Section 3, Methodology, were followed. In addition, the search string creation process was elaborated according to the recommendations of [23].

Another threat to validity was having used as the search string term only “Software Engineering”, not including terms specific to the phases of the software process, such as, “Requirements Engineering”, “Design”, “Testing”, among other similar terms. This may have excluded articles that deal more specifically with machine learning application in software development phases from the results.

6. Final Considerations

The main objective of this work was to carry out a systematic literature review to characterize the machine learning approach in software engineering. For this, systematic review methods recommended by the software engineering literature were used.

53 articles were analyzed that provided answers to the research questions defined in Section 3. The algorithms Support Vector Machine, Random Forests, Decision Tree and Naive Bayes were the most applied algorithms in software engineering. Software Testing and Software Planning activities were the main activities benefited by machine learning algorithms. Defect prediction and software activity effort estimation were the main applications of machine learning algorithms in software engineering and the main results of this application were: improved performance in defect prediction, improved accuracy in defect prediction, improved defect prediction and reduced cost and duration of activities.

It was possible to identify that the focus of academic research on the application of machine learning in software engineering is directed to testing and planning activities. Thus, design, requirements analysis and coding/development activities had few studies identified. Therefore, future work can be developed involving the application of machine learning in these activities, which may bring technological improvements and advances to software projects.

References

[1] D. A. Clifton, J. Gibbons, J. Davies, and L. Tarassenko, "Machine learning and software engineering in health informatics," 2012 1st Int. Work. Realiz. AI Synerg. Softw. Eng. RAISE 2012 - Proc., pp. 37–41, 2012, doi: 10.1109/RAISE.2012.6227968.

[2] T. M. Mitchell, Machine learning, 1TH Editio. Pittsburgh, PA 15213, EUA: McGraw-Hill Science/Engineering/Math; 1997.

[3] I. Sommerville, Software Engineering Tenth Edition. 2016.

[4] J. Shivhare and S. K. Rath, "Software effort estimation using machine learning techniques," 2014, doi: 10.1145/2590748.2590767.

[5] M. Perkusich et al., "Intelligent software engineering in the context of agile software development: A systematic literature review," Inf. Softw. Technol., vol. 119, p. 106241, 2020, doi: https://doi.org/10.1016/j.infsof.2019.106241.

[6] H. K. Dam, "Artificial intelligence for software engineering," XRDS Crossroads, ACM Mag. Students, vol. 25, no. 3, pp. 34–37, 2019, doi: 10.1145/3313117.

[7] R. Braga, P. S. Neto, R. Rabêlo, J. Santiago, and M. Souza, "A machine learning approach to generate test oracles," in ACM International Conference Proceeding Series, 2018, pp. 142–151, doi: 10.1145/3266237.3266273.

[8] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," Appl. Soft Comput., vol. 27, pp. 504–518, 2015, doi: https://doi.org/10.1016/j.asoc.2014.11.023.

[9] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," Inf. Softw. Technol., vol. 54, no. 1, pp. 41–59, 2012, doi: https://doi.org/10.1016/j.infsof.2011.09.002.

[10] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," Inf. Softw. Technol., vol. 54, pp. 41–59, 2012, doi: 10.1016/j.infsof.2011.09.002.

[11] H. Alsolai and M. Roper, "A systematic literature review of machine learning techniques for software maintainability prediction," Inf. Softw. Technol., vol. 119, p. 106214, 2020, doi: https://doi.org/10.1016/j.infsof.2019.106214.

[12] H. Alsolai and M. Roper, "A systematic literature review of machine learning techniques for software maintainability prediction," Inf. Softw. Technol., vol. 119, p. 106214, Mar. 2020, doi: 10.1016/j.infsof.2019.106214.

[13] W. Afzal and R. Torkar, "On the application of genetic programming for software engineering predictive modeling: A systematic review," Expert Syst. Appl., vol. 38, no. 9, pp. 11984–11997, 2011, doi: https://doi.org/10.1016/j.eswa.2011.03.041.

[14] B. Kitchenham, D. Budgen, and O. P. Brereton, Evidence-Based Software Engineering and Systematic Reviews, 1th Editio. United States: CRC Press, 2015.

[15] M. Anjum and D. Budgen, "A mapping study of the definitions for service oriented architecture," no. May 2012, pp. 57–61, 2012, doi: 10.1049/ic.2012.0008.

[16] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," Inf. Softw. Technol., vol. 50, no. 9–10, pp. 833–859, 2008, doi: 10.1016/j.infsof.2008.01.006.

[17] V. H. S. Durelli et al., "Machine Learning Applied to Software Testing: A Systematic Mapping Study," IEEE Trans. Reliab., vol. 68, no. 3, pp. 1189–1212, 2019, doi: 10.1109/TR.2019.2892517.

[18] G. G. M. Dalveren and D. Mishra, "Software engineering in medical informatics: A systematic literature review," in ACM International Conference Proceeding Series, Aug. 2019, pp. 112–117, doi: 10.1145/3357419.3357444.

[19] F. Elberzhager, A. Rosbach, and R. Eschbach, "Reducing Test Effort : A Systematic Mapping Study on Existing Approaches," Inf. Softw. Technol., vol. 54, no. October, pp. 1092–1106, 2012, doi: 10.1016/j.infsof.2012.04.007.

[20] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Third Edit., vol. 1, no. 70. New Jersey: Pearson Prentice Hall, 2013.

[21] T. Mitchell, "Chapther 06," Mach. Learn., pp. 125–174, 1997, doi: 10.1007/s10994-009-5101-2.

[22] I. H. Witten, E. Frank, and M. a. Hall, "Data Mining: Practical Machine Learning Tools and Techniques, Third Edition," Ann. Phys. (N. Y)., vol. 54, no. 2, p. 664, 2011, doi: 10.1002/1521-3773(20010316)40:6<9823::AID-ANIE9823>3.3.CO;2-C.

[23] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3," Engineering, vol. 45, no. 4ve, p. 1051, 2007, doi: 10.1145/1134285.1134500.

APPENDIX A. Studies Included in the Systematic Review

[1] A. Stocco, "How artificial intelligence can improve web development and testing," Apr. 2019, doi: 10.1145/3328433.3328447.

[2] M. T. Rahman and M. M. Islam, "A Comparison of Machine Learning Algorithms to Estimate Effort in

Varying Sized Software," in 2019 IEEE Region 10 Symposium (TENSYMP), 2019, pp. 137–142, doi: 10.1109/TENSYMP46218.2019.8971150.

[3] K. Chandra, G. Kapoor, R. Kohli, and A. Gupta, "Improving software quality using machine learning," in 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), 2016, pp. 115–118, doi: 10.1109/ICICCS.2016.7542340.

[4] R. Jindal, R. Malhotra, and A. Jain, "Software defect prediction using neural networks," in Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, 2014, pp. 1–6, doi: 10.1109/ICRITO.2014.7014673.

[5] T. Iqbal, P. Elahidoost, and L. Lúcio, "A Bird's Eye View on Requirements Engineering and Machine Learning," in 2018 25th Asia-Pacific Software Engineering Conference (APSEC), 2018, pp. 11–20, doi: 10.1109/APSEC.2018.00015.

[6] M. B. R. Pandit and N. Varma, "A Deep Introduction to AI Based Software Defect Prediction (SDP) and its Current Challenges," in TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 284–290, doi: 10.1109/TENCON.2019.8929661.

[7] Y. A. Alshehri, K. Goseva-Popstojanova, D. G. Dzielski, and T. Devine, "Applying Machine Learning to Predict Software Fault Proneness Using Change Metrics, Static Code Metrics, and a Combination of Them," in SoutheastCon 2018, 2018, pp. 1–7, doi: 10.1109/SECON.2018.8478911.

[8] P. D. Singh and A. Chug, "Software defect prediction analysis using machine learning algorithms," in 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017, pp. 775–781, doi: 10.1109/CONFLUENCE.2017.7943255.

[9] D. Mao, L. Chen, and L. Zhang, "An Extensive Study on Cross-Project Predictive Mutation Testing," in 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), 2019, pp. 160–171, doi: 10.1109/ICST.2019.00025.

[10] H. Sultanov and J. H. Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing," in 2013 21st IEEE International Requirements Engineering Conference (RE), 2013, pp. 52–61, doi: 10.1109/RE.2013.6636705.

[11] E. Ronchieri, M. Canaparo, A. Costantini, and D. C. Duma, "Data mining techniques for software quality prediction: a comparative study," in 2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC), 2018, pp. 1–2, doi: 10.1109/NSSMIC.2018.8824313.

[12] H. Lounis, T. F. Gayed, and M. Boukadoum, "Machine-Learning Models for Software Quality: A Compromise between Performance and Intelligibility," in 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, 2011, pp. 919–921, doi: 10.1109/ICTAI.2011.155.

[13] S. Agarwal, S. Gupta, R. Aggarwal, S. Maheshwari, L. Goel, and S. Gupta, "Substantiation of Software Defect Prediction using Statistical Learning: An Empirical Study," in 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1–6, doi: 10.1109/IoT-SIU.2019.8777507.

[14] R. Rana and M. Staron, "Machine learning approach for quality assessment and prediction in large software organizations," in 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2015, pp. 1098–1101, doi: 10.1109/ICSESS.2015.7339243.

[15] M. Hammad and A. Alqaddoumi, "Features-Level Software Effort Estimation Using Machine Learning Algorithms," in 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 2018, pp. 1–3, doi: 10.1109/3ICT.2018.8855752.

[16] D. P. Wangoo, "Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design," in 2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018, pp. 1–4, doi: 10.1109/CCAA.2018.8777584.

[17] N. Nascimento, P. Alencar, C. Lucena, and D. Cowan, "Toward Human-in-the-Loop Collaboration Between Software Engineers and Machine Learning Algorithms," in 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 3534–3540, doi: 10.1109/BigData.2018.8622107.

[18] S. Ghosh, A. Rana, and V. Kansal, "A Hybrid Nonlinear Manifold Detection Approach for Software Defect Prediction," in 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018, pp. 453–459, doi: 10.1109/ICRITO.2018.8748788.

[19] S. Roychowdhury and S. Khurshid, "Localization of faults in software programs using Bernoulli divergences," in 2012 International Symposium on Information Theory and its Applications, 2012, pp. 586–590.

[20] M. A. Haque, M. A. Rahman, and M. S. Siddik, "Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study," in 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019, pp. 1–5, doi: 10.1109/ICASERT.2019.8934499.

[21] S. Jha et al., "Deep Learning Approach for Software Maintainability Metrics Prediction," IEEE Access, vol. 7, pp. 61840–61855, 2019, doi: 10.1109/ACCESS.2019.2913349.

[22] K. K. Chaturvedi and V. B. Singh, "Determining Bug severity using machine learning techniques," in 2012 CSI Sixth International Conference on Software Engineering (CONSEG), 2012, pp. 1–6, doi: 10.1109/CONSEG.2012.6349519.

[23] L. Kumar and A. Sureka, "Feature selection techniques to counter class imbalance problem for aging related bug prediction," 2018, doi: 10.1145/3172871.3172872.

[24] G. Gabrani and N. Saini, "Effort estimation models using evolutionary learning algorithms for software development," in 2016 Symposium on Colossal Data Analysis and Networking (CDAN), 2016, pp. 1–6, doi: 10.1109/CDAN.2016.7570916.

[25] C. Lopez-Martin, S. Banitaan, A. Garcia-Floriano, and C. Yanez-Marquez, "Support Vector Regression for Predicting the Enhancement Duration of Software Projects," in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 562–567, doi: 10.1109/ICMLA.2017.0-101.

[26] N. Chen, S. C. H. Hoi, and X. Xiao, "Software process evaluation: A machine learning approach," in 2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings, 2011, pp. 333–342, doi: 10.1109/ASE.2011.6100070.

[27] M. F. Sohan, M. A. Kabir, M. I. Jabiullah, and S. S. M. M. Rahman, "Revisiting the Class Imbalance Issue in Software Defect Prediction," in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1–6, doi: 10.1109/ECACE.2019.8679382.

[28] A. BaniMustafa, "Predicting Software Effort Estimation Using Machine Learning Techniques," in 2018 8th International Conference on Computer Science and Information Technology (CSIT), 2018, pp. 249–256, doi: 10.1109/CSIT.2018.8486222.

[29] T. Sethi and Gagandeep, "Improved approach for software defect prediction using artificial neural networks," in 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2016, pp. 480–485, doi: 10.1109/ICRITO.2016.7785003.

[30] K. Yang, H. Yu, G. Fan, X. Yang, S. Zheng, and C. Leng, "Software Defect Prediction Based on Fourier Learning," in 2018 IEEE International Conference on Progress in Informatics and Computing (PIC), 2018, pp. 388–392, doi: 10.1109/PIC.2018.8706304.

[31] C. Tao, J. Gao, and T. Wang, "Testing and Quality Validation for AI Software–Perspectives, Issues, and Practices," IEEE Access, vol. 7, pp. 120164–120175, 2019, doi: 10.1109/ACCESS.2019.2937107.

[32] J. Deng, L. Lu, and S. Qiu, "Software defect prediction via LSTM," IET Softw., vol. 14, no. 4, pp. 443–450, 2020, doi: 10.1049/iet-sen.2019.0149.

[33] S. D. Immaculate, M. F. Begam, and M. Floramary, "Software Bug Prediction Using Supervised Machine Learning Algorithms," in 2019 International Conference on Data Science and Communication (IconDSC), 2019, pp. 1–7, doi: 10.1109/IconDSC.2019.8816965.

[34] T. Hoang, H. K. Dam, Y. Kamei, D. Lo, and N. Ubayashi, "DeepJIT: An End-to-End Deep Learning Framework for Just-in-Time Defect Prediction," in 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), 2019, pp. 34–45, doi: 10.1109/MSR.2019.00016.

[35] Y. N. Soe, P. I. Santosa, and R. Hartanto, "Software Defect Prediction Using Random Forest Algorithm," in 2018 12th Southeast Asian Technical University Consortium (SEATUC), 2018, vol. 1, pp. 1–5, doi: 10.1109/SEATUC.2018.8788881.

[36] R. Almaghairbe and M. Roper, "An Empirical Comparison of Two Different Strategies to Automated Fault Detection: Machine Learning Versus Dynamic Analysis," in 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2019, pp. 378–385, doi: 10.1109/ISSREW.2019.00099.

[37] C. López-Martín, A. Chavoya, and M. E. Meda-Campaña, "Use of a Feedforward Neural Network for Predicting the Development Duration of Software Projects," in 2013 12th International Conference on Machine Learning and Applications, 2013, vol. 2, pp. 156–159, doi: 10.1109/ICMLA.2013.182.

[38] Y. Hu, X. Zhang, X. Sun, M. Liu, and J. Du, "An Intelligent Model for Software Project Risk Prediction," in 2009 International Conference on Information Management, Innovation Management and Industrial Engineering, 2009, vol. 1, pp. 629–632, doi: 10.1109/ICIII.2009.157.

[39] N. Maneerat and P. Muenchaisri, "Bad-smell prediction from software design model using machine learning techniques," in 2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), 2011, pp. 331–336, doi: 10.1109/JCSSE.2011.5930143.

[40] J. Gao, C. Tao, D. Jie, and S. Lu, "Invited Paper: What is AI Software Testing? and Why," in 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), 2019, pp. 27–2709, doi: 10.1109/SOSE.2019.00015.

[41] D. A. Clifton, J. Gibbons, J. Davies, and L. Tarassenko, "Machine learning and software engineering in health informatics," 2012 1st Int. Work. Realiz. AI Synerg. Softw. Eng. RAISE 2012 - Proc., pp. 37–41, 2012, doi: 10.1109/RAISE.2012.6227968.

[42] R. G. Ramani, S. V Kumar, and S. G. Jacob, "Predicting fault-prone software modules using feature selection and classification through data mining algorithms," in 2012 IEEE International Conference on Computational Intelligence and Computing Research, 2012, pp. 1–4, doi: 10.1109/ICCIC.2012.6510294.

[43] V. Ionescu, "An approach to software development effort estimation using machine learning," in 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2017, pp. 197–203, doi: 10.1109/ICCP.2017.8117004.

[44] Z. Polkowski, J. Vora, S. Tanwar, S. Tyagi, P. K. Singh, and Y. Singh, "Machine Learning-based Software Effort Estimation: An Analysis," in 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2019, pp. 1–6, doi: 10.1109/ECAI46879.2019.9042031.

[45] H. D. Tran, L. T. M. Hanh, and N. T. Binh, "Combining feature selection, feature learning and ensemble learning for software fault prediction," in 2019 11th International Conference on Knowledge and Systems Engineering (KSE), 2019, pp. 1–8, doi: 10.1109/KSE.2019.8919292.

[46] D. Amaratunga, J. Cabrera, D. Sargsyan, J. B. Kostis, S. Zinonos, and W. J. Kostis, "Uses and opportunities for machine learning in hypertension research," Int. J. Cardiol. Hypertens., vol. 5, Jun. 2020, doi: 10.1016/j.ijchy.2020.100027.

[47] A. Barbez, F. Khomh, and Y. G. Guéhéneuc, "A machine-learning based ensemble method for anti-patterns detection," J. Syst. Softw., vol. 161, Mar. 2020, doi: 10.1016/j.jss.2019.110486.

[48] F. Pecorelli, D. Di Nucci, C. De Roover, and A. De Lucia, "A large empirical assessment of the role of data balancing in machine-learning-based code smell detection," J. Syst. Softw., vol. 169, Nov. 2020, doi: 10.1016/j.jss.2020.110693.

[49] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and W. Meding, "The Adoption of Machine Learning Techniques for Software Defect Prediction: An Initial Industrial Validation," in Communications in Computer and Information Science, 2014, vol. 466 CCIS, pp. 270–285, doi: 10.1007/978-3-319-11854-3_23.

[50] M. Staron et al., "Robust Machine Learning in Critical Care — Software Engineering and Medical Perspectives," in 2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN), 2021, pp. 62–69, doi: 10.1109/WAIN52551.2021.00016.

[51] S. Chakraborty, L. Deng, and J. Dehlinger, "Towards Authentic Undergraduate Research Experiences in Software Engineering and Machine Learning," in Proceedings of the 3rd International Workshop on Education through Advanced Software Engineering and Artificial Intelligence, 2021, pp. 54–57, doi: 10.1145/3472673.3473966.

[52] W. Gong, G. Hu, J. Zou, W. Gong, G. Huang, and Z. Zhang, "Research on the application of artificial intelligence-based methods in civil engineering monitoring," in 2021 2nd International Conference on Artificial Intelligence and Education (ICAIE), 2021, pp. 200–203, doi: 10.1109/ICAIE53562.2021.00049.

[53] J. Alharbi and S. Bhattacharyya, "Machine Learning with System/Software Engineering in Selection and Integration of Intelligent Algorithms," in 2021 IEEE International Systems Conference (SysCon), 2021, pp. 1–7, doi: 10.1109/SysCon48628.2021.9447111.

APPENDIX B. Systematic Review Protocol

The Protocol used in this study is available on the link (URL):

https://1drv.ms/u/s!AkbPoDpRXQFpkPoNSxSMSzHbMjHF2Q?e=5FDmkS