Preprint of an article: Szymon Stradowski and Lech Madeyski (2023), "Machine learning in software defect prediction: A business-driven systematic mapping study", *Information and Software Technology*, Volume 155, Pages 107128. DOI: 10.1016/j.infsof.2022.107128
Preprint: https://madeyski.e-informatyka.pl/download/StradowskiMadeyski23bISTpreprint.pdf

Machine learning in software defect prediction: A business-driven systematic mapping study

Szymon Stradowski[a,b,1], Lech Madeyski[b,2,*]

[a] *Nokia, Szybowcowa 2, Wrocław, 54-206, Dolnoslaskie, Poland*
[b] *Wrocław University of Science and Technology, Wyb. Wyspianskiego 27, Wrocław, 50-370, Dolnoslaskie, Poland*

Abstract

Context: Machine learning is a valuable tool in software engineering allowing fair defect prediction capabilities at a relatively small expense. However, although the practical usage of machine learning in defect prediction has been studied over many years, there is not sufficient systematic effort to analyse its potential for business application.
Objective: The following systematic mapping study aims to analyse the current state-of-the-art in terms of machine learning software defect prediction modelling and to identify and classify the emerging new trends. Notably, the analysis is done from a business perspective, evaluating the opportunities to adopt the latest techniques and methods in commercial settings to improve software quality and lower the cost of development life cycle.
Method: We created a broad search universe to answer our research questions, performing an automated query through the Scopus database to identify relevant primary studies. Next, we evaluated all found studies using a classification scheme to map the extent of business adoption of machine learning software defect prediction based on the keywords used in the publications. Additionally, we use PRISMA 2020 guideline to validate reporting.
Results: After the application of the selection criteria, the remaining 742 primary studies included in Scopus until February 23, 2022 were mapped to classify and structure the research area. The results confirm that the usage of commercial datasets is significantly smaller than the established datasets from NASA and open-source projects. However, we have also found meaningful emerging trends considering business needs in analysed studies.
Conclusions: There is still a considerable amount of work to fully internalise business applicability in the field. Performed analysis has shown that purely academic considerations dominate in published research; however, there are also traces of in vivo results becoming more available. Notably, the created maps offer insight into future machine learning software defect prediction research opportunities.

Keywords: software defect prediction, machine learning, systematic mapping study, business applicability, effort and cost minimisation

1. Introduction

Already in 1998 Slaughter et al. [1] made a case for justification of investments in software quality improvements to be done with similar consideration as with any other new project. Despite this and many subsequent efforts, in 2020, the Cost of Poor Software Quality (CPSQ) in the US alone was estimated at $2.08 trillion [2]. Based on these data, the Consortium for Information & Software Quality recommends that software companies strongly emphasise defect prevention and addressing weaknesses by identifying, isolating and correcting problems as early as possible. Otherwise, businesses worldwide will continue to lose fortunes due to escaped software defects.

Improving the quality and minimising the cost of software development life-cycle has been the goal of software engineering (SE) practitioners and researchers for decades. A significant number of research papers have been written and published to discover new and more efficient ways to identify defect-prone areas. However, the challenge becomes increasingly complex, exacerbated by the constantly growing code-base (illustrative example provided by Google [3]).

With its commercial popularisation, machine learning (ML) has become the central area of study and is seen as having significant potential to impact defect-finding efficiency. One exceptionally promising concept is software defect prediction (SDP), using models that indicate the areas of the code where faults are most probable [4]. As a result, many ML techniques have been conceived, each with different effects based on the circumstance [5]. Notably, it is also claimed that no universal model could be applied to all datasets to develop effective solutions as in the "no free lunch" (NFL) theorems [6, 7]. Due to this and several other causes, like high complexity, low understandability, and high maintenance, business adoption of the proposed

*Corresponding author
[1] ORC ID: 0000-0002-3532-3876
[2] ORC ID: 0000-0003-3907-3357

solutions is comparatively lagging the academic research. We aim to verify this assertion by conducting a systematic mapping study evaluating the degree of business adoption of the up-and-coming machine learning trends.

Our systematic mapping study follows the approach for evidence-based software engineering research by Kitchenham et al. [8]. The first step is to define the research questions that allow investigation of the subject. The next step is to search and identify all relevant primary studies targeted for assessment. Finally, the primary studies undergo data extraction and synthesis to create the systematic map. As a result, this paper is organised into five main sections. Section 1 serves as the introduction to the researched topic highlighting the main goals and contributions. Section 2 describes the research methodology, in particular, the research questions and selection criteria. Section 3 consists of result analysis focusing on quantitative analysis, techniques, new trends, and evaluation business adoption attempts. Discussion of the obtained results and identified threats to validity are presented in Section 4, while Section 5 contains the summary and conclusions.

Notably, the following systematic mapping study is business-driven in its nature. Its purpose contributes to developing a solution that directly satisfies business requirements of the Nokia company. The business requirements we want to address are improving the quality and minimising the cost of software testing of 5G systems, as defined by Stradowski and Madeyski [9]. Therefore, this article is part of a more extensive series committed to the business endeavour of introducing ML SDP in the company.

1.1. Contributions

Our systematic mapping study is designed to provide a broad overview of our research area and indicate the quantity and quality of the existing evidence on business adoption of ML SDP. The most significant contributions of the study are similar to the ones given by Hall et al. [10], albeit specified in emphasising new research trends in commercial business settings:

- An overview of the most significant secondary studies in the area of machine learning and software defect prediction.
- A set of 742 primary studies explicitly on the machine learning software defect prediction.
- A subset of 32 primary studies explicitly including business adoption examples of machine learning software defect prediction.
- Generic overview of the current state-of-the-art machine learning software defect prediction provided by studies satisfying our assessment criteria, depicting the adoption of new trends in real-life business applications.
- Sufficient contextual and methodological detail to enable similar studies to be reliably evaluated and mapped by other researchers and practitioners interested in selecting an appropriate methodology for a specific context.
- A complete PRISMA 2020 checklist for transparency and reporting purposes, together with all work-products needed for replicability. An offering of suggestions to help identify practical opportunities from the ML SDP area to commercial applications.
- Importantly, this mapping study serves as a follow-up to the survey by Stradowski and Madeyski [9] that analyses the challenges to improve quality and minimise the cost of software testing of 5G systems at Nokia. Our mapping will support Nokia software engineering practitioners in researching challenges they face in specific commercial circumstances by providing a classification of relevant literature [11]. This business-driven effort will be focused on introducing an ML SDP solution to the system-level testing within the company, and the systematic mapping study will help understand the current state-of-the-art.

1.2. Related work

To the best of our knowledge, there were no comprehensive secondary studies published on business adoption of machine learning software defect prediction focusing on real-life commercial applications published in recent years. There are, however, several valuable systematic literature reviews (SLR) and systematic mapping studies (SMS) conducted in isolation from business considerations. They represent some of the best articles in the field, are widely cited, and provide insight into the methodology and techniques used in machine learning research. Moreover, highlighted articles uncover meaningful ways to assess and critique research inadequacies.

- Fenton and Neil [12] covered defect prediction studies up to 1999 and provided a critique of the field, emphasising good and bad practices in statistical data analysis. Even though their research did not follow a systematic review approach, it identified and documented several pitfalls to avoid for future researchers. The critique also sets a high standard for state-of-the-art software engineering modelling for predicting software defects in methodological and theoretical aspects.
- In 2009, Catal and Diri [13] conducted a systematic review of software defect prediction studies focusing on metrics, methods, and datasets. The review identified 74 software defect prediction papers in journals and conference proceedings published between 1990 and 2007. The main recommendations based on the conclusions were to conduct more studies on fault prediction models using class-level measurements, increase the usage of public datasets for fault prediction, and increase the usage of machine learning-based models overall.

- Hall et al. [10] identified and analysed the models used to predict software defects in source code in 208 studies published between January 2000 and December 2010. The authors emphasised how model performance was affected by its development context, variables used, and the model's technique. Interestingly, research suggests that models that perform well tend to be built around large systems and that the methodology used to create models is influential to their predictive performance.
- Shepperd et al. [14] analysed the results of 42 papers (that collectively report 600 sets of empirical prediction results) reporting studies comparing ML algorithms used for software defect prediction. They found that the explanatory factor that accounted for the largest percentage of the differences among studies (as large as 31%) was the research group, while the main topic of research, i.e., the choice of the ML algorithm, accounted for only 1.3% of the variation among studies. They commented that "It matters more who does the work than what is done" and "Until this can be satisfactorily addressed, there seems little point in conducting further primary studies". Analysis by Shepperd et al. [14], among others, was used by Madeyski and Kitchenham to raise awareness of the problems caused by unreproducible research in software engineering, to discuss the concept of reproducible research (RR), advantages of and problems with RR, and the need for wider adoption of reproducible research in software engineering [15].
- Malhotra [16] conducted a systematic review of machine learning research in software defect prediction, which identified 64 primary studies and seven categories of machine learning techniques. In general, the ML models outperformed the linear regression (LR) benchmark models for SDP. The models based on the Random Forest, Naive Bayes, and Bayesian Networks techniques outperformed the other ML models in most studies. Notably, most studies were based on the NASA dataset or publicly available open-source software. Consequently, more industrial datasets should be used to demonstrate the practical effectiveness of the ML techniques.
- Durelli et al. [17] performed a comprehensive systematic mapping study of research published from 1980 to August 2017. The results outline the ML algorithms most commonly used to automate software testing activities and offer future researchers an account of the state-of-the-art ML applied to software testing. However, the authors also point out that only a few primary studies evaluated ML-based solutions in industrial settings or used industry-grade software and that further improvement is needed in this research area.
- Hosseini et al. [18] has identified 30 primary studies passing quality assessment on an auspicious from a business perspective concept of cross-project defect prediction (CPDP). The study shows that Naive Bayes and Linear Regression are the most widely used learning techniques in this context, with recall, precision, f-measure, and AUC being the most frequently used metrics. Importantly, although cross-project defect prediction generally shows worse performance than within-project defect prediction, it is still viable in situations where it can not be applied due to insufficient training data.
- Li et al. [19] conducted a systematic literature review identifying 49 studies containing 2456 individual experimental results, published between January 2000 and March 2018. The review focused on unsupervised learning techniques in software defect prediction and did not find evidence that unsupervised classifiers perform worse than their supervised counterparts. Therefore, they may be a more viable option in certain situations when labelled training data is scarce and general, and less human intervention is needed.
- The most recent SLR conducted by Wang et al. [20] aims to improve the applicability and generalizability of machine learning and deep learning studies. Authors analyze papers published between 2009 and 2020 and observe trends introduced into the field, focusing on complexity and resulting issues with reproducibility and replicability. Furthermore, the authors introduced categorization of the rationales behind selecting techniques into five areas to see how model performance, robustness, interpretability, complexity, and data simplicity affecting the resulting models.

Overall, there are numerous publications on using machine learning techniques in software engineering; however, a seemingly small fraction focuses on its commercial application. Notably, not all papers seem to adhere to the concept of reproducible research [15] to be available for reuse in other academic or business circumstances. Therefore, we believe that in future research, significant effort should be put into embedding business value creation and replicability principles to academic study. There is potential for improvement in this regard.

2. Review methodology

As mentioned, the following research adheres to the guidelines for secondary studies defined by Kitchenham et al. [8]. First, we clarify our research questions, then search and identify relevant publications. Next, we synthesise results from the selected studies. Finally, we explore and answer our research questions through quantitative analysis. We also followed the practice guidelines for systematic mapping studies by Petersen et al. [21, 11], as well as considered several relevant practices of systematic literature reviews [22]. The overview of the process

steps is visible in Figure 1. Furthermore, we used PRISMA 2020 [23, 24] and provided the details in a dedicated checklist available in Appendix B.

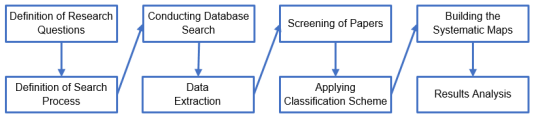

Figure 1: Used systematic mapping process.

2.1. Research questions

Goal Question Metric (GQM) by Basili et al. [25] helped us to establish a goal-oriented approach and build proper measures for our research purpose. The GQM goal was formulated as follows:

- *Analyse the* research literature on machine learning software defect prediction
- *for the purpose of* understanding
- *with respect to* business applicability
- *from the perspective of* a team (including both a software engineering practitioner and researcher) working for Nokia
- *in the context of* the research literature indexed by Scopus.

We have broken down our GQM goal into seven research questions (RQs). Each question reflects and evaluates the attractiveness of the analysed primary study from a practitioner's perspective, pursuing potential solutions to real-life industry challenges. Accordingly, we based our analysis on the notion of business applicability to limit the possible divergence between industry and academic priorities [26]. Therefore, by business applicability we mean the degree to which the proposed solution is suited to be used in a commercial environment. We believe a researched concept can be business applicable and have a chance to be implemented in a real-life industry environment if it addresses at least a subset of the rationales provided as motivation to our research questions. The list of RQs and their reasoning is available in Table 1.

Lastly, it is helpful when the study is easy to find using straightforward keywords and is available for access via popular databases. This requirement was not translated into a research question, but built into the assumption with which the search process and selection criteria were defined (see Section 2.2).

2.2. Search process

Our strategy for a business-driven mapping study was to include a wide selection of available literature, conducting an automated database search. Notably, as mapping studies can have less stringent search requirements Kitchenham et al. [8], it is not critical to include all possible sources when exploring general trends. Therefore, based on recommendations by Dieste et al. [27], we established

RQ1. How much research is being published in the field of ML SDP? **Business rationale:** To turn into standard practice, the area of machine learning software defect prediction should attract the attention of the research community, while gaining momentum should be visible. Answering RQ1, we are able to capture whether such attention and momentum exist to increase interest of researchers and practitioners to implement such solutions.
RQ2. What innovations are used to increase the effectiveness of ML SDP? **Business rationale:** The researched methods should be innovative and possess the potential to identify new opportunities and to expand the state-of-the-art. Innovations are means to build competitive advantage for commercial companies. Hence, mapping key innovations is crucial from the business perspective and is helpful for Nokia in its goal to increase usage of ML SDP in its operations.
RQ3. Which methods are most popular in ML SDP? **Business rationale:** The ML method needs to be considered highly effective in solving many SDP problems, to be considered in industrial practice. Research should identify the most popular and the most generally applicable methods in the current state-of-the-art. Consequently, such knowledge aids practitioners in further implementations in industry.
RQ4. What kind of datasets and projects are used for ML SDP? **Rationale:** It would be beneficial for the method to be proven to deliver satisfactory results on a broad scope of datasets, preferably also originating from real environments. Evaluating the extent to which the research is based on real-world business examples allows analysing in vivo verification of the studied solutions. For Nokia, understating the characteristics of researched datasets could enable cross-project defect prediction opportunities and accelerate future feasibility studies.
RQ5. Are human interactions successfully incorporated into ML SDP? **Business rationale:** Depending on the specifics of the problem they need to be overcome, practitioners in Nokia need to understand in great detail how the model works and be able to explain why an ML model arrived at a specific decision, or even may need to directly interact with the model using their expert knowledge as input. Therefore, we should identify the opportunities and the extent of leveraging human and machine intelligence as in eXplanable ML/AI or Human-in-the-loop (HITL).
RQ6. Are cost-conscious considerations included in ML SDP? **Business rationale:** A publication promoting an ML SDP solution should discuss cost considerations, as monetary estimates and operational savings are often integral to a project charter starting a new business initiative. Authors should be mindful of how much introducing the solution costs and estimating the potential gain that can be achieved, as value added is one of the most critical factors from the business applicability perspective. Therefore, it is necessary to assess to what extent researchers have considered the costs and benefits of the proposed software defect prediction solutions.
RQ7. Are reproducible research principles used for ML SDP? **Business rationale:** Documentation describing the method needs to provide reproducible evidence for its effectiveness. Such evidence is necessary for any form of a feasibility studies and technology readiness assessments to be initiated in commercial settings. Practitioners may need to evaluate if the researched software defect predictions can be consistently reproduced, showing the same results. Reproducibility issues may have serious consequences across disciplines, but also in software engineering [15].

Table 1: Research Questions.

a search universe based on the journals, conferences, and book chapters available in Scopus[3], as it covers the main venues such as IEEE, ACM, Springer, Elsevier, etc. Moreover, Scopus offers a functional advanced search engine allowing flexible queries and several export options, suiting the research aims, scale, and tooling needed in our R&D project conducted in Nokia.

We created a dedicated string used on the Scopus database to search through the publications' titles, abstracts, and keywords. Secondly, we did not constrain the year of publication to map the whole SE field up to the date the search was conducted. We choose keywords based on the defined GQM goal and adhere to the expectation that the research should be easy to find. Therefore, we decided on general conditions for the title or keywords or abstract to include the following words with respective boolean operators: "software" and "machine learning", and "defect" or "fault" or "bug", and "model" or "prediction" or "forecast". Next, we limited the field to "computer science" and document types to "article", "conference paper", and "book chapter". Also, we excluded papers that were not in English as 100% completeness was not critical when exploring trends. The final version of the used search string is as follows:

> TITLE-ABS-KEY (("software") AND ("machine learning") AND ("defect" OR "fault" OR "bug") AND ("model" OR "prediction" OR "forecast")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "ch")) AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (SUBJAREA , "COMP"))

We performed the final search on the 23rd of February 2022, finding 1222 papers. All of the results were saved in BibTeX and CSV formats for further analysis and are available in Appendix A.

2.3. Inclusion and exclusion criteria

We applied inclusion and exclusion criteria presented in Table 2 based on Kitchenham et al. [8]. During the screening of papers, we found seven duplicates that were removed. Next, based on manual title and abstract filtering, we identified and excluded 473 (39%) studies as not fulfilling our criteria and not directly adhering to our research questions. As a result, out of all 1222 studies selected by the search method presented in Section 2.2, 742 (61%) papers remained for further analysis (see Figure 2). Considering the number of studies to be analysed, we decided against using further snowballing [28].

If not necessary, we would advise future researchers to avoid using "systematic literature review" and its paraphrases in the title or abstract if the paper is not, in fact, an SLR. Avoiding such confusion would help manual and automatic searches separate primary and secondary studies. Also, we would like to reiterate how critically important it is to provide complete yet concise, fully describing the goal and scope of investigation in the titles and abstracts in scientific research papers. Furthermore, it is beneficial to define the exact area of the research in the title, as "software defect prediction" is much more precise than "defect prediction", as the latter may refer to many fields like biology, geology, electronics, and many more.

[3]https://www.scopus.com

Inclusion criteria	Exclusion criteria
• The paper is an empirical study in software engineering (field of computer science). • The paper is a primary study. • The paper is focused on predicting defects in a software system using machine learning techniques. • The paper evaluates, analyses, or compares prediction methods and provides evidence for efficiency.	• The paper was not a peer-reviewed article, conference proceeding, or book chapter. • The publication's language was other than English. • The paper was made available in Scopus after the 23rd of February 2022. • The same or limited results were already published and included in another study.

Table 2: Inclusion and exclusion criteria.

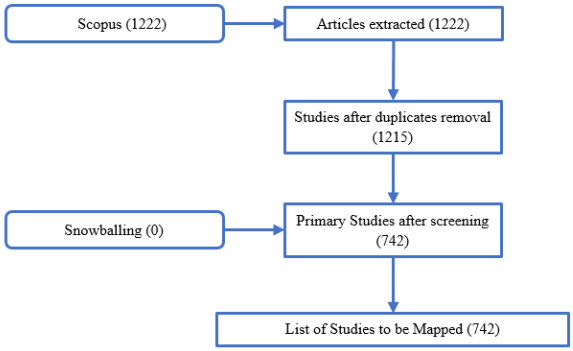

Figure 2: Study identification and selection.

2.4. Data extraction

Figure 2 illustrates the overview of the study selection process. As mentioned, in total, 1222 publications were found based on our defined search terms. After screening and selection criteria application, 742 primary studies remained for further assessment.

During the advanced search execution in Scopus, we automatically extracted the following data:

- Author, Title, Year, Source, Cited by, DOI, Link, Publisher, Document type, Source, Abstract, Author Keywords, Index Keywords

- During the screening we added an 'Excl.' field to mark if the paper was excluded.

The screened list of papers after was used to build our maps was saved in CSV and BibTex formats. Secondly, the ASH tool [29] (see Section 3) was used to download all studies available directly from the Scopus database automatically. Mentioned artefacts are available in Appendix A.

3. Results

Our goal was to understand the broad perspective on the published research in ML SDP. To obtain meaningful results, we carried out the Scopus database search described in Section 2. Next, we carried out systematic mapping and classification to answer our GQM-based research questions. As a result, we provide our insight from four distinct perspectives described below—general overview, keyword analysis, trend identification, and exemplary business applications.

- General overview and simple quantitative analysis [8] was performed to understand publication year distribution and the number of citations per year.
- Keyword (incl. both author and index keywords [30]) analysis and bibliometric maps were built using VOSviewer ver. 1.6.18, as this is a tool created specifically for larger number items [31].
- Mass publication download and was done with ASH tool [5] ver. 1.1.04 [29].
- Finally, full text reading of selected papers was done to confirm and verify industry application examples.

3.1. General overview

The first part of the mapping focused on understanding how many studies are being published each year and how intensively are they cited was done by simple graph visualisations.

RQ1. How much research is being published in the field of ML SDP?

Computer science was the 5th most popular subject area with 6,56 million papers, after medicine, engineering, physics and astronomy, biochemistry, genetics and molecular biology, respectively. The exact search string in computer science limiting the keywords to only "software" showed over 615 thousand records and limited to "software" AND "machine learning" 14.5 thousand. Therefore, our study touches approximately 5% of software-related machine learning publications in computer science and 0.1% of all software-related publications in computer science.

[4] https://www.vosviewer.com/
[5] https://github.com/LechMadeyski/AutomatedSearchHelper/wiki

Figure 3 shows a graph of the number of studies published each year and the number of citations. Interestingly, there are significant differences showing fluctuation of the intensity of citations, despite the amount of work published in ML SDP. There has been a substantial increase in published studies with a relatively stagnant citation count in recent years. For example, in 2008, only 12 published works were cited 1302 times, more than 130 titles published in 2019. However, more citations will scale with time, and the saturation of topics for research seems to be still far away. Moreover, the publication count is growing year on year until 2019. The number of publications flat-lines for the following periods, indicating a possible slow down of the research in the area or influencing factors hampering progress temporarily.

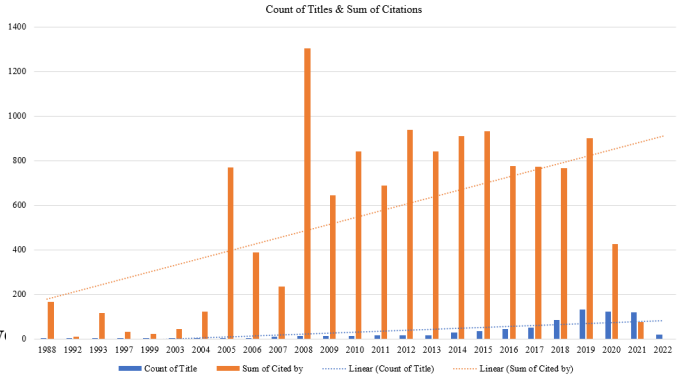

Figure 3: Number of publications and citations.

3.2. Keyword analysis

Next, we have built a map of co-authorship (see Figure 4), visualising all researchers with more than three publications on our list. Out of 1574 overall, 164 authors have been classified. A significant portion of the data points is strongly dispersed, with a high number (42) of different clusters with almost no connections. Nevertheless, a considerable chain connects more than half of the authors, indicating good synergy in the field and close cooperation within the research community.

Exactly 1181 authors have one publication (229 have two, 164 have three and more), showing that many researchers contribute to the body of knowledge in ML SDP. Moreover, there are also very prolific authors like Malhotra, with 36 works.

3.3. Keyword analysis part I

Figure 5 shows a graphical representation of keywords provided by the authors. There were 1398 keywords overall, and after excluding the ones occurring less than five times, 87 remained. After using the association strength for normalisation, 9 clusters were built using VOSviewer tool for bibliometric mapping by Eck and Waltman [31]. For analysis we used two item attributes: occurrence rate (the number of times the item appeared on the map), and

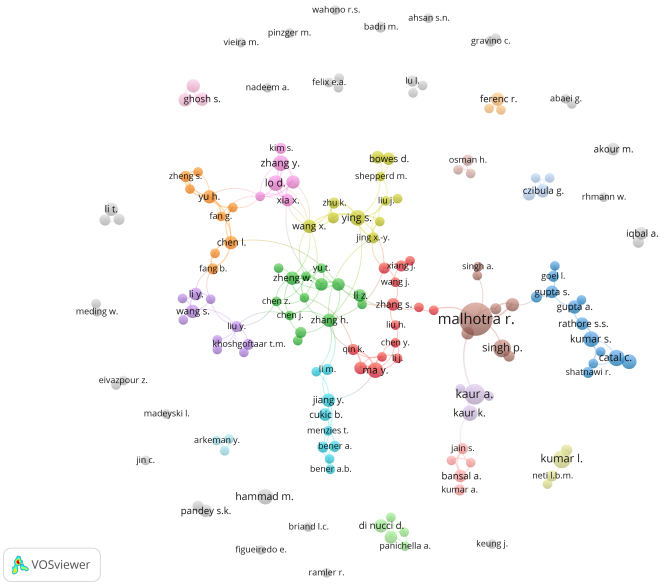

Figure 4: Map of co-authorship.

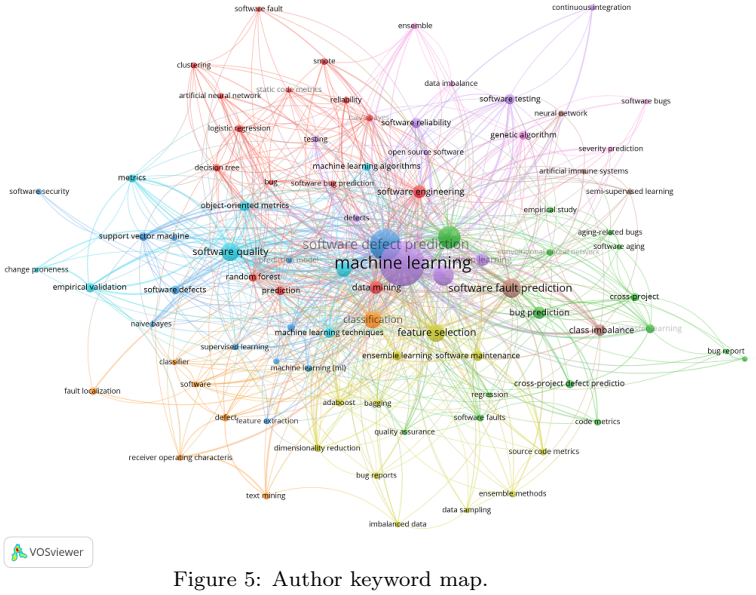

Figure 5: Author keyword map.

total link strength (total strength of the links of an item with connected items).

Thirdly, the overwhelming majority of keywords are generic (”test”, ”quality assurance”, ”forecasting”, etc.), and only a portion reflects any manifestation of the uniqueness of research, among other publications. Apart from a small number of data points such as ”neural network”, ”random forest”, or ”naive bayes”, showing more details about the conducted research. Thus, the map is very generic and allows one to observe only a narrow spectrum of meaningful detail. Created view does not allow further investigation in terms of our RQs. Furthermore, author keywords do not distinguish many alternative spelling or plurals, making it cumbersome to map automatically. In the light of accessability(see Section 2.1, we would advise highlighting used techniques and environment in author keywords to enable easier identification.

Interestingly, authors much more frequently use the phrase ”Software Defect Prediction” (157 occurrences, total link strength 246) than ”Software Fault Prediction” (59 occurrences, total link strength 96) and ”Software Bug Prediction” (only six occurrences). Therefore, we would advise all future authors to use the prevalent SDP nomenclature. Secondly, many keywords do not precisely determine the study area as keywords like ”defect modelling” or ”fault prediction” can apply to many fields of study apart from software engineering. Therefore, mindful and precise usage of keywords supports the accessibility of results and helps secondary research obtain more accurate results.

Next, we build a similar map based on index keywords[6] (available in Appendix A). They differ from the keywords provided by the authors in two significant ways. Firstly, following the description from Scopus, “These are keywords chosen by content suppliers and are standardised based on publicly available vocabularies. Unlike author keywords, the indexed keywords take into account synonyms, various spellings, and plurals”. Secondly, there is much more of them available for an average publication. Therefore, index keywords should provide more observation and conclusion opportunities during analysis.

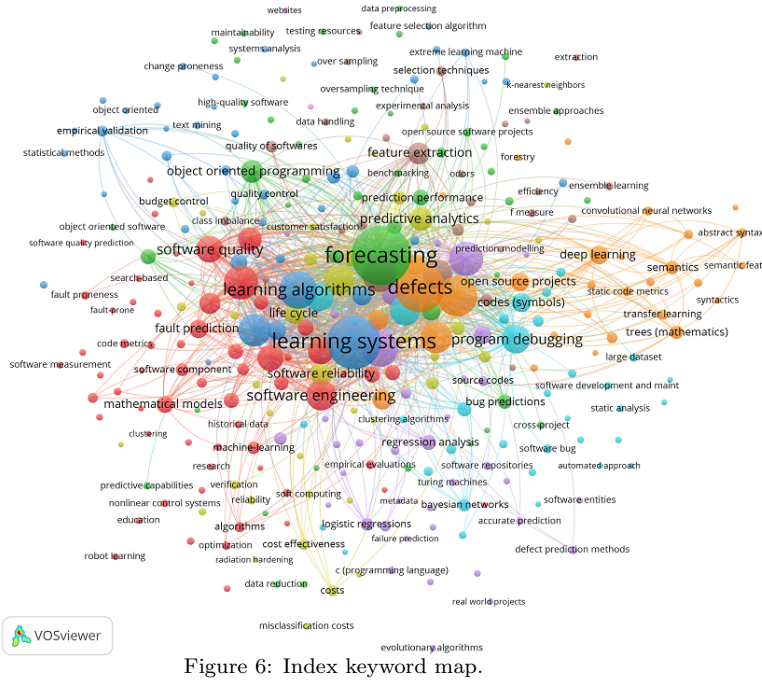

Figure 6: Index keyword map.

There were 2883 index keywords, and after excluding those occurring less than five times, 318 remained in our map Figure 6. Again, index maps are condensed and dominated by apparent terminology similar to author keywords. The representation of index keywords was predominantly

[6] https://service.elsevier.com/app/answers/detail/a_id/21730/supporthub/scopus/

generic, mainly providing the area or research without the essential details and not allowing to obtain many specific insights. Keywords like "research", "engineering", "test", "system" are a few examples that bring minimal value. Since we already limited the scope by creating a dedicated search string followed by manual selection, we decided to remove generic terms related to the area of our study as not containing any additional information to our research (exact list available in Appendix A). Filtering was done based on our expert knowledge and was not intended to be meticulously precise, enabling an approximation of the general trends as enough to answer our RQs.

After filtering, out of 2883 index keywords, 318 meet the threshold of occurring a minimum of five times. We manually eliminated 239 as generic, leaving 79 items on our map, see Figure 7 (to better visualise the clean information, we used a different than default layout of 'Attraction, from default 2 to 0' and 'Repulsion, from default 0 to -2').

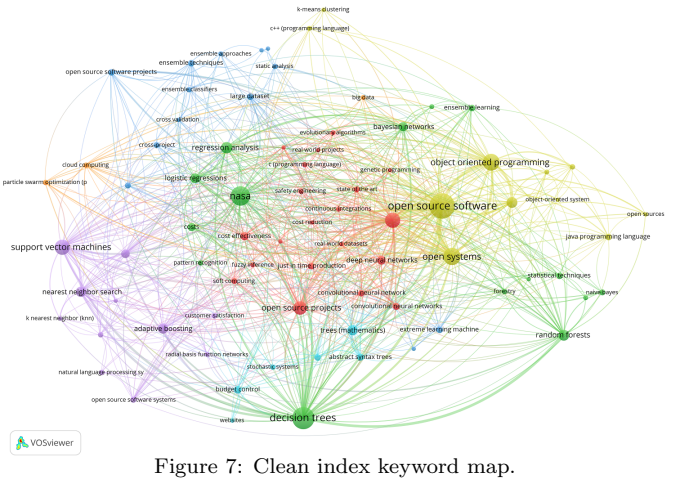

Figure 7: Clean index keyword map.

Below we provide analysing the occurrence rate and total link strength of several selected keywords that allow further insight into our RQ2-7, based on built maps (see Section 3.4).

RQ2. What innovations are used to increase the effectiveness of ML SDP?
We found that "state-of-the-art methods" occurred ten times with total link strength 26 and "state of the art" occurred nine times with total link strength 19. We also searched for the word "novel" but to no avail, despite frequently appearing in the titles and abstracts. Therefore, we conclude it is difficult to evaluate the number of innovations published in the field on keywords alone without considering the time of publishing (see Section 3.5). Wallwork [32] pointed out that millions of papers do not have adjectives, such as "innovative" or "novel", in their titles for a good reason. The problem with such adjectives is that they give no indication as to how something is novel (and if research is not "novel", then no one would want to read about it anyway). Our study indicates that the same applies to keywords, which poses a challenge if we want to map the innovations published in the field.

RQ3. Which methods are most popular in ML SDP?
After deselecting generic terms, techniques were the most popular keywords used, clearly revealing the relative popularity of each of them. The top five techniques are: "Decision trees" occurring 117 times, "support vector machines" occurring 57 times, "neural networks" occurring 54 times, "random forests" occurring 29 times, and "nearest neighbour search" occurring 26 times. Worth mentioning is also the emerging techniques that, if proven to be highly effective, will gain more popularity in the future, like "particle swarm optimisation", "multilayer neural networks", or "convolutional neural networks". There is also a wider variety of techniques that appear less than five times in the keywords, thus not present on our maps but potentially emerging in the following years.

RQ4. What kind of datasets and projects are used for ML SDP?
There was only minimal information on the datasets available in the used keywords. "Open source software" appeared 157 times, "open source projects" 46 and several other versions were also visible. Secondly, "NASA" appeared 89 times with a total link strength of 1113. Notably, "real-world projects" and "real-world data sets" both appeared five times, also "industry" and its abbreviations have 28 instances, but none of which satisfied our criterion of occurring at least five times. It shows that in vivo research is being published; however, rarely. We also observed "cross-project", as in cross-project defect predictions, to be relatively popular with nine occurrences and 19 link strength, indicating that this concept, attractive from the business perspective, is gaining the researcher's attention. Additionally, we found only minimal presence of the PROMISE repository in the keywords; however, it is frequently mentioned in the abstracts.

RQ5. Are human interactions successfully incorporated into ML SDP?
Human interaction and explainable AI were not highlighted on the keyword maps (further discussed in Section 3.4).

RQ6. Are cost-conscious considerations included in ML SDP?
The cost-related context was relatively popular within the keywords used, appearing 44 times in the form of "cost", "cost effectiveness", "cost benefit analysis", and "cost reduction". This indicates a growing interest in including one of the most critical aspects of commercial operations in academic considerations. We do not expect that all proposed solutions will be

able to introduce a positive return on investment and provide exhaustive details on the cost of implementation. However, we believe that it is very beneficial to highlight these aspects as part of the discussion on research results, as cost efficiency is an essential enabler behind the search for more efficient test practices in software engineering companies [26].

RQ7. Are reproducible research principles used for ML SDP?
Reproducible research was not highlighted on the keyword maps (further discussed inSection 3.4).

3.4. Keyword analysis part II

As we failed in finding sufficient proof to address the following two of our RQs using created maps in Section 3.3, we decided to undertake the following actions with regard those RQs:

RQ5. Are human interactions successfully incorporated into ML SDP?
Our maps, based on the keyword inclusion criteria of occurring at least five times, did not provide evidence for incorporating human interaction in ML SDP research. We conducted a more exhaustive search through all available keywords and found no relevant instances of the word "human", and four instances on the word "explainable" but in various versions: two times "explainable models", one "prediction explanation", and one "explainable". Thus, we conclude that incorporating human interaction is still relatively uncommon and contains enormous potential in future research as increasingly important in the business context [33].

RQ7. Are reproducible research principles used for ML SDP?
Our keyword maps did not show any instances of reproducible research. We found one "replicated experiment" instance during an exhaustive search through all available keywords. As this is a well-established concept, we decided to extend the search to abstracts to confirm adhering to the reproducible research principles. Indeed we found several forms of reference in the abstracts; however, it was challenging to include all precisely. As the next step, a targeted deep text search could be performed among all selected studies (for example, using the ASH tool [29]) to measure the extent of concept applicability. However, we decided against it based on time and effort considerations, concluding that there is unquantified evidence in abstracts and text but not in the keywords. We conclude that despite a certain portion of publications allow reproduction, this is not verifiable or measurable using only keyword maps.

3.5. Keyword analysis part III

To gain an understanding of emerging trends, we created four smaller maps representing the periods from 1988-2010, 2011-2016, 2017-2019, and 2020-2022 to show how the used keywords were changing with time Figure 8. Such a split allowed a comparable number of keywords mapped in each period.

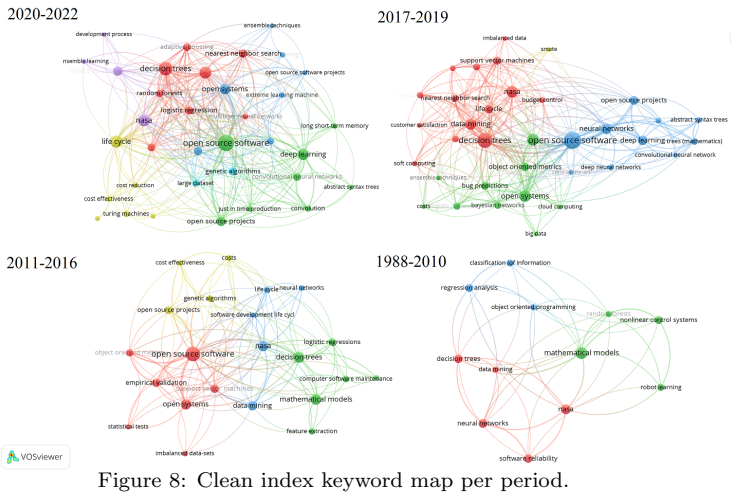

Figure 8: Clean index keyword map per period.

Timed graphs depict how the field of ML SDP has increased in complexity over the years, each period including more insight into what the research is focused on. Dividing the map into four periods revealed that the field is growing in complexity and adding considerable detail to the techniques used in each period. The most influential trends include:

- **1988-2010:** Early research is generic, showing only the main concepts with limited value to our RQs. Apart from "NASA", "object-orientated programming", "random forest", "decision trees", and "neural networks" not much details are available. Naturally, at the initial stage of exploration of ML SDP by researchers, the effort is focused on feasibility, exploring main possibilities, and has only a limited set of opportunities identified.

- **2011-2016:** In the following years, signs of deeper consideration for business settings as "development life-cycle" and "cost consideration" concepts raised in popularity. Also, bayesian networks and support vector machines appear as popular techniques. As in the previous period, "open source" and "NASA" prevail in datasets.

- **2017-2019:** The most crucial evolution in the next three years includes "state of the art" as the most frequently used techniques have been established. Secondly, "customer satisfaction", "budget control", and "cost" gain popularity as the field becomes more mature and business adoption becomes more feasible.

- **2020-2022:** During the last period, worth highlighting is the emergence of the "just in time" concept. Continued cost-consciousness has been displayed by "cost effectiveness" and "cost reduction keywords". Consistently, the "open source" and "NASA" datasets

dominate among published studies. Interestingly, enhancements to neural networks like deep, convolutional, and multi-layer appeared on the graph with relatively significant connection strength.

Interestingly, we observed a few keywords that have diminished in the research trends over the last few years. Most significantly, "robot learning" and "nonlinear control systems" appeared in the first period (1988-2010) but have not returned in the later ones. Also, the term "data mining", popular in the first three periods, does not appear in the latest research. This observation shows that some concepts, although attracting much attention during a particular time, either die out as not being sustainable or become an integral part of the research field and stop being highlighted by the authors.

4. Discussion of results

Research on improving the quality and minimising the cost of software development using machine learning defect prediction by performing a business-driven systematic mapping study has led us to several interesting conclusions. We aimed to find traces for detecting an increase in commercial implementation and results in the created publication database, and we did find evidence for a limited in vivo research being published as suggested by Lanza et al. [34]. The overwhelming majority was based on NASA and open source projects, as the need for more commercial benchmarks becomes critical [35]. Considering that the field is being heavily researched and the state-of-the-art is constantly expanding, we expect the in vivo research to gain popularity as more and more difficulties in commercial implementations are being solved. It is an indication that the gap between industry and academia can converge in the next years [26].

Also, we gained interesting insights similar to the systematic mapping study primary studies on using ML in software testing that was published from June 2017 to August 2018 by Durelli et al. [17]. Firstly, we could confirm the following finding "RQ4: What trends can be observed among research studies discussing the application of ML to support software testing activities? A trend we observed is that the oracle problem tends to be tackled by employing either ANN- or decision tree based approaches". The same techniques appear as most popular ones throughout our selected studies. Secondly, we concur with "RQ7: We found that the body of empirical research available at the intersection of ML and software testing leaves much to be desired, especially when compared with the level of understanding and body of evidence that has been achieved in other fields". However, ML SDP has a high potential to evolve significantly considering the rise of published works in recent years. Lastly, Durelli et al. [17] conclude that their results seem to suggest that there is no research group especially dealing with ML and software testing (RQ8 [17]). We found such a well-connected group while mapping out a broader scope of studies, see Figure 5.

4.1. Industry publications

An overwhelming amount of empirical case studies uses NASA and PROMISE datasets or is based on available open-source projects. Consequently, based on the cleaned index keyword map, we selected a subset of publications that were validated in an industrial context (listed in 5). We performed a full text screening of the selected papers to confirm industry validation.

Next, we mapped found industrial publications in Figure 9. As there are only 32 studies including 324 index keywords, only 14 meet the threshold of occurring five times. Therefore, we loosened the occurrence restriction to two, resulting in 82 items remaining. After filtering out generic words, 34 items were mapped.

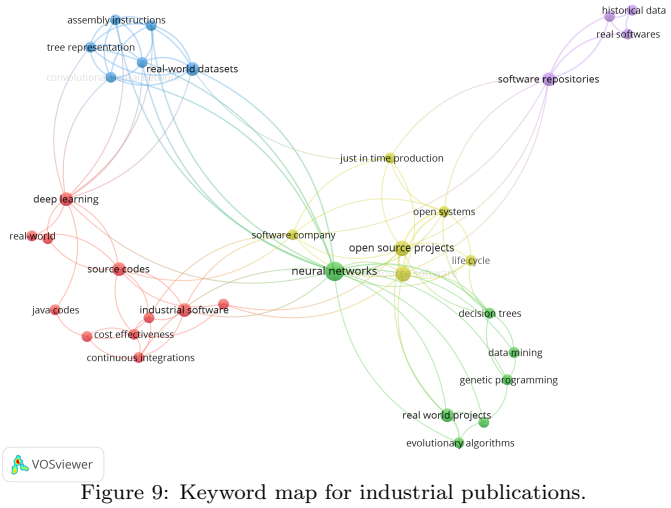

Figure 9: Keyword map for industrial publications.

Significantly, a considerable portion of commercial research is done in the telecommunication industry. There are also instances of software, finance, and transport companies that were unnamed, alongside huge trademarks such as IBM, Cisco, Microsoft, Ericsson, or Volvo. Also, there are interesting cases of utilising open source projects for training and then verification on industry data.

Nevertheless, a limited amount of research is conducted in a real-world setting, and low cooperation with commercial partners shows how much effort is still needed to converge the focus of academics and practitioners [26]. The scarcity of industrially verified approaches may discourage many commercial projects (see Section 2.1) from potentially revolutionary new solutions. However, it should also be an encouragement for companies like Nokia Stradowski and Madeyski [9] and others to promote and publish their ML SDP efforts.

4.2. Further recommendations

There are a wide plethora of further insights that can be obtained from our mapping effort for anyone who is

looking to understand how much research is has been done on specific aspects of ML SDP:

- A comparison of dataset "NASA" with 89 occurrences and 1113 total link strength and "real-world data sets" with five occurrences and 58 total link strength (see Figure 10) depicts one of the most significant discrepancies to be studied further. Importantly, it shows how an overwhelmingly large number of studies use this dataset in comparison to other sources. In particular, we believe it depicts the demand for further commercial and business-originating studies to be researched and published.
- Comparison of techniques like "decision trees" with 117 occurrences and 1530 total link strength and "convolutional neural networks" with 12 occurrences and 156 total link strength (see Figure 11) highlights not only the perceived popularity of already well-established techniques but also the potential for further development of the emerging ones.
- Many other analyses and SLRs can be performed for interesting keyword areas as "just-in-time", "cross-project", "software life cycle", or "customer satisfaction".
- Building more detailed maps not liming the occurrence rate, possibly filtering only selected themes as techniques, supervised/unsupervised learning, particular datasets, metrics and more, could be performed to gain further insight into ML SDP field.
- Considering the increasing amount of work published in the area, the next period 2023-2025 should be analysed to understand the evolution direction that has taken place as in Section 3.5.
- Lastly, during our research, we observed a relatively small number of systematic mapping studies published in the field of computer science, whereas we believe they can lead to meaningful discoveries of areas with the most significant potential for further research, see Petersen et al. [11]. We hope such studies will increase in popularity over time, and more areas within the field can be mapped.

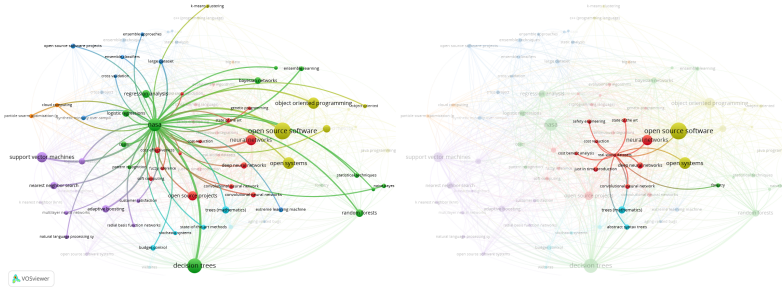

Figure 10: Comparison of "NASA" and "real-world projects" in index keywords.

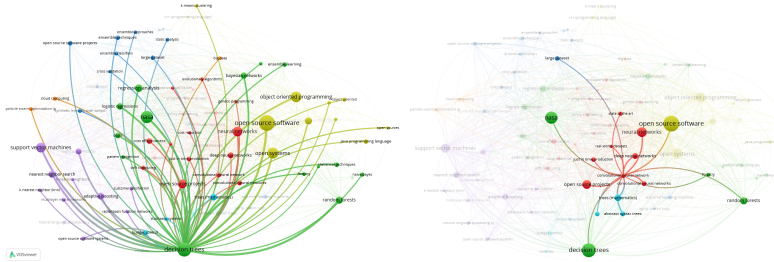

Figure 11: Comparison of "decision trees" and "convolutional neural networks" in index keywords.

We firmly believe that there is a significant potential for software practitioners to scale productivity, increase quality, and reduce costs by improving the usage of academic research in daily work [9, 36]. Therefore, systematic mapping studies such as this one, systematic literature reviews [8], and other methods like rapid reviews [37], should enable both researchers and practitioners to understand better what is available and where lay future opportunities.

In particular, for the challenges faced by Nokia on the system level testing of 5G technology described by Stradowski and Madeyski [9], our study shows a wide plethora of potential possibilities. Although limited industrial research is being published, many of the created techniques and methodologies can and should be validated in commercial settings — especially the recently emerging ML techniques and cost-conscious solutions of just-in-time defect prediction.

4.3. Threats to validity

Any systematic mapping study needs to provide constraints on the search process, and any deviations from the standard practice need to be discussed. To secure the validity of our research, we addressed the most common categories of threats from Zhou et al. [38].

- **Construct validity:** The most significant construct validity threat arises from the availability of commercial data to be published. It is critical to emphasise that we explored the business application of ML SDP only in officially academic journals. Most business-driven efforts in commercial software engineering are covered by data-protection laws and are not published in academia. Furthermore, we would like to point out that the incentive for software companies to make their internal research results available via academic journals is limited. The second construct validity threat reflects our assumption that a solution is business-applicable if it satisfies our criteria (described as Research Questions). There is a significant risk that we might have omitted aspects that may be critical in particular business circumstances, as we aimed for our criteria to be as general as possible.

- **External validity:** The systematic mapping study we performed was done from the business perspective and aimed to evaluate the extent of applicability of ML SDP research in commercial environments. We do not claim our conclusions can be generalised outside of the domain of our problem. Nevertheless, we have found articles showing similar findings regarding emerging ML trends in the studied discipline [34, 35].
- **Internal validity:** It is possible that some relevant papers were not included in our mapping. The first reason is using only the pre-selected electronic database (Scopus), which offers advanced search capabilities, but does not support full-text search. Secondly, some relevant research might have been missed due to search string imperfection. While some research papers may address the same concepts with different naming, we believe the terms are established well enough to minimise this risk. We contemplated further mitigation by snowballing to decrease missed research pool; however, we decided against it as the analysed publications' completeness was not our goal. The second major threat we identified was related to the screening process. The goal was to manually select all studies that coincidentally fit our search criteria but do not directly address our subject. The process was based on title and if that was not enough to decide on the abstract. Finally, more than 39% of papers were evaluated as not relevant. However, the process was considerably prone to mistakes. We used only those keywords that appeared more than five times in our maps to limit the impact of individual errors on the overall picture and derived conclusions.
- **Conclusion Validity:** We wanted to ensure that the data and analysis methods were transparent. The same results presented in the paper can be repeatedly achieved from the provided data and procedures. To enable such reproducibility, we included a detailed research procedure description in Section 2, gathered data snapshot and all additional maps in Appendix A, and a PRISMA 2000 checklist in Appendix B. The most significant conclusion validity threat is the manual evaluation and selection criteria application. To mitigate, we have performed a cross-check of the evaluations by two researchers. Still, due to the subjective nature of the process, another group of researchers might obtain a different subset of publications under study.

5. Conclusions

As software engineering businesses are especially prone to defect-caused monetary losses, it is critically important to deepen the knowledge base and create new defect prediction and modelling methods. To gain insight into improving the quality and minimising the cost of software development using machine learning software defect prediction, we have identified 742 relevant studies and used them to map out future opportunities.

The popularity of the keywords used to describe the content of the study translates not only to the amount of research of the subject but also to the accessibility of the state-of-the-art knowledge; therefore, precise keyword usage is critical to efficient business adoption. Also, we believe a broad picture overview of the field is fundamental to its understanding, allows meaningful conclusions, and uncovers areas for further research. Thus, good keyword maps can help practitioners and businesses in supporting decision-making within software engineering practice [17].

Lastly, our systematic mapping of all of the publications in the field has shown that the amount of research has grown in recent years, significantly adding new value to the researched concepts, techniques, and trends. However, despite the increasing popularity of in vivo research, the field is still dominated by NASA-based and open source projects, as commercial datasets available to the research community are scarce (e.g., a few commercial projects from Capgemini [39, 40]). Therefore, for Nokia [9], many opportunities lay in studying the feasibility of proposed solutions, careful development and deployment in Nokia, and by publishing research results based on their live system, using the novel, state-of-the-art solutions highlighted by our keyword mapping effort.

CRediT authorship contribution statement

Szymon Stradowski: Data curation, Methodology, Investigation, Writing – original draft, Writing - review & editing, Visualisation. **Lech Madeyski**: Conceptualisation, Funding acquisition, Methodology, Investigation, Writing – original draft, Writing - review & editing, Supervision.

Declaration of competing interest

This research was carried out in partnership with Nokia (with Szymon Stradowski as an employee).

Acknowledgement

This research was financed by the Polish Ministry of Education and Science 'Implementation Doctorate' program (ID: DWD/5/0178/2021).

References

[1] S. A. Slaughter, D. E. Harter, M. S. Krishnan, Evaluating the cost of software quality, Communications of the ACM 41 (1998) 67–73.

[2] H. Krasner, The Cost of Poor Software Quality in the US: A 2020 Report, Technical Report, Consortium for Information & Software Quality, 2020. URL: https://www.it-cisq.org/pdf/CPSQ-2020-report.pdf, accessed: 25.12.2021.

[3] R. Potvin, J. Levenberg, Why google stores billions of lines of code in a single repository, Commun. ACM 59 (2016) 78–87. doi:10.1145/2854146.

[4] T. M. Khoshgoftaar, N. Seliya, Fault prediction modeling for software quality estimation: Comparing commonly used techniques, Empirical Software Engineering 8 (2003) 255–283. Copyright - Kluwer Academic Publishers 2003; Last updated - 2012-02-08.

[5] K. Meinke, A. Bennaceur, Machine Learning for Software Engineering Models, Methods, and Applications, 2017. doi:10.1145/3183440.3183461.

[6] D. H. Wolpert, The Lack of A Priori Distinctions Between Learning Algorithms, Neural Computation 8 (1996) 1341–1390. doi:10.1162/neco.1996.8.7.1341.

[7] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82. doi:10.1109/4235.585893.

[8] B. Kitchenham, D. Budgen, P. Brereton, Evidence-Based Software Engineering and Systematic Reviews, CRC Press, 2016.

[9] S. Stradowski, L. Madeyski, Exploring the challenges in software testing of the 5g system at nokia: A survey, Information and Software Technology 153 (2023) 107067. doi:https://doi.org/10.1016/j.infsof.2022.107067.

[10] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Transactions on Software Engineering 38 (2012) 1276–1304. doi:10.1109/TSE.2011.103.

[11] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and Software Technology 64 (2015) 1–18. doi:https://doi.org/10.1016/j.infsof.2015.03.007.

[12] N. Fenton, M. Neil, A critique of software defect prediction models, IEEE Transactions on Software Engineering 25 (1999) 675 – 689. doi:10.1109/32.815326.

[13] C. Catal, B. Diri, A systematic review of software fault prediction studies, Expert Systems with Applications 36 (2009) 7346–7354. doi:https://doi.org/10.1016/j.eswa.2008.10.027.

[14] M. Shepperd, D. Bowes, T. Hall, Researcher Bias: The Use of Machine Learning in Software Defect Prediction, IEEE Transactions in Software Engineering 40 (2014) 603–616.

[15] L. Madeyski, B. Kitchenham, Would wider adoption of reproducible research be beneficial for empirical software engineering research?, Journal of Intelligent & Fuzzy Systems 32 (2017) 1509–1521. URL: http://madeyski.e-informatyka.pl/download/MadeyskiKitchenham17JIFS.pdf. doi:10.3233/JIFS-169146.

[16] R. Malhotra, A systematic review of machine learning techniques for software fault prediction, Applied Soft Computing 27 (2015) 504–518. doi:https://doi.org/10.1016/j.asoc.2014.11.023.

[17] V. H. S. Durelli, R. S. Durelli, S. S. Borges, A. T. Endo, M. M. Eler, D. R. C. Dias, M. P. Guimarães, Machine learning applied to software testing: A systematic mapping study, IEEE Transactions on Reliability 68 (2019) 1189–1212. doi:10.1109/TR.2019.2892517.

[18] S. Hosseini, B. Turhan, D. Gunarathna, A systematic literature review and meta-analysis on cross project defect prediction, IEEE Transactions on Software Engineering 45 (2019) 111–147. doi:10.1109/TSE.2017.2770124.

[19] N. Li, M. Shepperd, Y. Guo, A systematic review of unsupervised learning techniques for software defect prediction, Information and Software Technology 122 (2020) 106287. doi:10.1016/j.infsof.2020.106287.

[20] S. Wang, L. Huang, A. Gao, J. Ge, T. Zhang, H. Feng, I. Satyarth, M. Li, H. Zhang, V. Ng, Machine/deep learning for software engineering: A systematic literature review, IEEE Transactions on Software Engineering (2022) 1–1. doi:10.1109/TSE.2022.3173346.

[21] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, BCS Learning & Development Ltd., Swindon, GBR, 2008, p. 68–77.

[22] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007. URL: https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf.

[23] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, D. Moher, The prisma 2020 statement: an updated guideline for reporting systematic reviews, BMJ 372 (2021). doi:10.1136/bmj.n71.

[24] M. J. Page, D. Moher, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, J. E. McKenzie, Prisma 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews, BMJ 372 (2021). doi:10.1136/bmj.n160.

[25] V. R. Basili, G. Caldiera, H. D. Rombach, The Goal Question Metric Approach, 1994.

[26] V. Garousi, M. Felderer, Worlds Apart - Industrial and Academic Focus Areas in Software Testing, IEEE Software 34 (2017) 38–45.

[27] O. Dieste, A. Griman, N. Juristo, Developing search strategies for detecting relevant experiments, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), volume 14, 2009, p. 513–539. doi:10.1007/s10664-008-9091-7.

[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, EASE '14, Association for Computing Machinery, New York, NY, USA, 2014. doi:10.1145/2601248.2601268.

[29] M. Sośnicki, L. Madeyski, ASH: A New Tool for Automated and Full-Text Search in Systematic Literature Reviews, in: M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, J. J. Dongarra, P. M. Sloot (Eds.), Computational Science – ICCS 2021, Springer International Publishing, Cham, 2021, pp. 362–369. doi:10.1007/978-3-030-77967-2_30.

[30] Elsevier B.V. , Scopus, 2022. URL: https://service.elsevier.com/app/answers/detail/a_id/21730/supporthub/scopus/, accessed: 28.02.2022.

[31] N. J. Eck, L. Waltman, Software survey: Vosviewer, a computer program for bibliometric mapping, Scientometrics 84 (2010) 523–538. doi:10.1007/s11192-009-0146-3.

[32] A. Wallwork, English for Writing Research Papers, Springer, 2011.

[33] A. Adadi, M. Berrada, Peeking inside the black-box: A survey on explainable artificial intelligence (xai), IEEE Access 6 (2018) 52138–52160. doi:10.1109/ACCESS.2018.2870052.

[34] M. Lanza, A. Mocci, L. Ponzanelli, The tragedy of defect prediction, prince of empirical software engineering research, IEEE Software 33 (2016) 102–105. doi:10.1109/MS.2016.156.

[35] J. M. Zhang, M. Harman, L. Ma, Y. Liu, Machine learning testing: Survey, landscapes and horizons, IEEE Transactions on Software Engineering 48 (2022) 1–36.

[36] D. Lo, N. Nagappan, T. Zimmermann, How practitioners perceive the relevance of software engineering research, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, ACM, New York, NY,

USA, 2015, pp. 415–425. doi:10.1145/2786805.2786809.
[37] B. Cartaxo, G. Pinto, S. Soares, The role of rapid reviews in supporting decision-making in software engineering practice, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, EASE'18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 24–34.
[38] X. Zhou, Y. Jin, H. Zhang, S. Li, X. Huang, A map of threats to validity of systematic literature reviews in software engineering, in: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), 2016, pp. 153–160. doi:10.1109/APSEC.2016.031.
[39] M. Jureczko, L. Madeyski, Towards identifying software project clusters with regard to defect prediction, in: PROMISE'2010: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, ACM, 2010, pp. 9:1–9:10. doi:10.1145/1868328.1868342.
[40] L. Madeyski, M. Jureczko, Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study, Software Quality Journal 23 (2015) 393–422. doi:10.1007/s11219-014-9241-7.

Systematic Mapping Study References - Industry

[SMS1] E. Arisholm, L. C. Briand, E. B. Johannessen, A systematic and comprehensive investigation of methods to build and evaluate fault prediction models, Journal of Systems and Software 83 (2010) 2–17. doi:https://doi.org/10.1016/j.jss.2009.06.055, sI: Top Scholars.
[SMS2] D. She, K. Pei, D. Epstein, J. Yang, B. Ray, Neuzz: Efficient fuzzing with neural program smoothing, 2019, pp. 803–817. doi:10.1109/SP.2019.00052.
[SMS3] J. Shirabad, T. Lethbridge, Mining the maintenance history of a legacy software system, 2003, pp. 95– 104. doi:10.1109/ICSM.2003.1235410.
[SMS4] A. Viet Phan, M. Le Nguyen, L. Thu Bui, Convolutional neural networks over control flow graphs for software defect prediction, in: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), 2017, pp. 45–52. doi:10.1109/ICTAI.2017.00019.
[SMS5] A. Phan, L. Nguyen, Convolutional neural networks on assembly code for predicting software defects, 2017, pp. 37–42. doi:10.1109/IESYS.2017.8233558.
[SMS6] S. Lee, C. Jung, S. Pande, Detecting memory leaks through introspective dynamic behavior modelling using machine learning (2014). doi:10.1145/2568225.2568307.
[SMS7] T. Yu, W. Wen, X. Han, J. Hayes, Conpredictor: Concurrency defect prediction in real-world applications, IEEE Transactions on Software Engineering PP (2018) 1–1. doi:10.1109/TSE.2018.2791521.
[SMS8] S. Tabassum, L. Minku, D. Feng, G. Cabral, L. Song, An investigation of cross-project learning in online just-in-time software defect prediction, 2020, pp. 554–565. doi:10.1145/3377811.3380403.
[SMS9] Y. Kim, S. Mun, S. Yoo, M. Kim, Precise learn-to-rank fault localization using dynamic and static features of target programs, ACM Transactions on Software Engineering and Methodology 28 (2019) 1–34. doi:10.1145/3345628.
[SMS10] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, W. Meding, The adoption of machine learning techniques for software defect prediction: An initial industrial validation, volume 466, 2014. doi:10.1007/978-3-319-11854-3_23.
[SMS11] D. She, R. Krishna, L. Yan, S. Jana, B. Ray, Mtfuzz: fuzzing with a multi-task neural network, Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (2020). doi:10.1145/3368089.3409723.
[SMS12] K. Blincoe, A. Dehghan, A.-D. Salaou, A. Neal, J. Linåker, D. Damian, High-level software requirements and iteration changes: a predictive model, Empirical Software Engineering 24 (2019). doi:10.1007/s10664-018-9656-z.
[SMS13] P. Koch, K. Schekotihin, D. Jannach, B. Hofer, F. Wotawa, T. Schmitz, Combining spreadsheet smells for improved fault prediction, in: Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 25–28. doi:10.1145/3183399.3183402.
[SMS14] Z. Zhu, Y. Li, H. Tong, Y. Wang, Cooba: Cross-project bug localization via adversarial transfer learning, in: IJCAI, 2020.
[SMS15] J. Kang, D. Ryu, J. Baik, Predicting just-in-time software defects to reduce post-release quality costs in the maritime industry, Software: Practice and Experience 51 (2021) 748–771. doi:https://doi.org/10.1002/spe.2927.
[SMS16] J. Briem, J. Smit, H. Sellik, P. Rapoport, G. Gousios, M. Aniche, Offside: Learning to identify mistakes in boundary conditions, 2020, pp. 203–208. doi:10.1145/3387940.3391464.
[SMS17] H. Wang, T. Khoshgoftaar, A study on software metric selection for software fault prediction, 2019, pp. 1045–1050. doi:10.1109/ICMLA.2019.00176.
[SMS18] M. Gokceoglu, H. Sozer, Automated defect prioritization based on defects resolved at various project periods, Journal of Systems and Software 179 (2021) 110993. doi:10.1016/j.jss.2021.110993.
[SMS19] H. Sellik, O. Paridon, G. Gousios, M. Aniche, Learning off-by-one mistakes: An empirical study, 2021.
[SMS20] E. Hershkovich, R. Stern, R. Abreu, A. Elmishali, Prioritized test generation guided by software fault prediction, 2021, pp. 218–225. doi:10.1109/ICSTW52544.2021.00045.
[SMS21] L. Gomes, R. Torres, M. Côrtes, On the prediction of long-lived bugs: An analysis and comparative study using floss projects, Information and Software Technology 132 (2020) 106508. doi:10.1016/j.infsof.2020.106508.
[SMS22] M. Kawalerowicz, L. Madeyski, Continuous Build Outcome Prediction: A Small-N Experiment in Settings of a Real Software Project, 2021, pp. 412–425. doi:10.1007/978-3-030-79463-7_35.
[SMS23] M. Kawalerowicz, L. Madeyski, Jaskier: A Supporting Software Tool for Continuous Build Outcome Prediction Practice, 2021, pp. 426–438. doi:10.1007/978-3-030-79463-7_36.
[SMS24] W. Afzal, Using faults-slip-through metric as a predictor of fault-proneness, Proceedings - Asia-Pacific Software Engineering Conference, APSEC (2010). doi:10.1109/APSEC.2010.54.
[SMS25] D. Bowes, S. Counsell, T. Hall, J. Petrić, T. Shippey, Getting defect prediction into industrial practice: the elff tool, 2017, pp. 44–47. doi:10.1109/ISSREW.2017.11.
[SMS26] R. Ramler, T. Natschläger, Applying heuristic approaches for predicting defect-prone software components, in: R. Moreno-Díaz, F. Pichler, A. Quesada-Arencibia (Eds.), Computer Aided Systems Theory – EUROCAST 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 384–391. doi:10.1007/978-3-642-27549-4_49.
[SMS27] R. Ferenc, Bug forecast: A method for automatic bug prediction, volume 117, 2010, pp. 283–295. doi:10.1007/978-3-642-17578-7_28.
[SMS28] D. Marijan, A. Gotlieb, A. Sapkota, Neural network classification for improving continuous regression testing, 2020, pp. 123–124. doi:10.1109/AITEST49225.2020.00025.
[SMS29] S. Pradhan, V. Nanniyur, P. Vissapragada, On the defect prediction for large scale software systems – from defect density to machine learning, 2020, pp. 374–381. doi:10.1109/QRS51102.2020.00056.
[SMS30] D. Elsner, F. Hauer, A. Pretschner, S. Reimer, Empirically Evaluating Readily Available Information for Regression Test Optimization in Continuous Integration, Association for Computing Machinery, New York, NY,

USA, 2021, p. 491–504. URL: https://doi.org/10.1145/3460319.3464834.

[SMS31] L. Zong, Classification based software defect prediction model for finance software system - an industry study, in: Proceedings of the 2019 3rd International Conference on Software and E-Business, ICSEB 2019, Association for Computing Machinery, New York, NY, USA, 2019, p. 60–65. doi:10.1145/3374549.3374553.

[SMS32] B. Agrawal, M. Mishra, Demo: Automatically retrainable self improving model for the automated classification of software incidents into multiple classes, in: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 2021, pp. 1110–1113. doi:10.1109/ICDCS51616.2021.00113.

.

Appendix A. Raw results

Original forms are available in Supplementary Material:
https://madeyski.e-informatyka.pl/download/MappingStudy22/ScopusExport.csv
https://madeyski.e-informatyka.pl/download/MappingStudy22/ScopusExport.bib
Created visualisations are available in Supplementary Material:
https://madeyski.e-informatyka.pl/download/MappingStudy22/MapData.zip
Details of the selected 742 publications exported from the ASH tool [29] are available in Supplementary Material:
https://madeyski.e-informatyka.pl/download/MappingStudy22/Publications.zip
The list of all keywords together with the occurrence rate and link strength are available in Supplementary Material:
https://madeyski.e-informatyka.pl/download/MappingStudy22/Words.csv

Original mapping artifacts are available in Supplementary Material:
https://doi.org/10.5281/zenodo.7375768

Appendix B. PRISMA 2020 checklist

PRISMA 2020 item checklist			
Topic	Item#	Details	Location
Title	1	Identify as a systematic review	Title, title page
Abstract	2	Do an abstract checklist	Abstract, page 1
Rationale	3	Provide rationale for the review in the context of existing knowledge	Section 1.1, page 3
Objectives	4	State the questions addressed by the review	Section 2.1, page 9
Eligibility criteria	5	Specify the inclusion and exclusion criteria	Section 2.3, page 11
Information sources	6	Specify all sources searched to identify studies	Section 2.2, page 10
Search strategy	7	Provide full search strategies for all databases	Section 2.2, page 10
Selection process	8	Describe methods used to decide whether a study met the inclusion criteria	Section 2.3, page 11
Data collection process	9	Specify the methods used to collect data from reports	Section 2.4, page 12
Data items	10	List and define all outcomes for which data were sought	Section 3, page 12
Study risk of bias assessment	11	Specify the methods used to assess risk of bias in the included studies	Section 4.3, page 26
Effect measures	12	Specify for each outcome the effect measures	Section 3, page 14
Synthesis methods	13	Describe the processes used to decide which studies were eligible for each synthesis	Section 3, page 14
Reporting bias assessment	14	Provide the methods used to assess risk of bias due to missing results in a synthesis	Section 4.3, page 26
Certainty assessment	15	Provide the methods used to assess certainty in the body of evidence	Section 4.3, page 26
Study selection	16	Describe the results of the search and selection process	Section 3, page 12
Study characteristics	17	Cite each included study and present its characteristics	Appendix A, page 38
Risk of bias in studies	18	Present assessments of risk of bias for each included study.	Section 4.3, page 26
Results of individual studies	19	For each study provide summary statistics and an effect estimate	Section 3, page 12
Results of syntheses	20	Present results of all statistical syntheses conducted.	Section 3, page 12
Reporting biases	21	Present assessments of risk of bias due to missing results	Section 4.3, page 26
Certainty of evidence	22	Present assessments of certainty in the body of evidence for each outcome	Section 4.3, page 26
Discussion	23	Provide a interpretation of the results in the context of other evidence.	Section 4, page 22
Registration and protocol	24	Provide registration information for the review,	Mapping not registered
Support	25	Describe sources of financial or non-financial support for the review,	Section 5, page 27
Competing interests	26	Declare any competing interests of review authors.	Section 5, page 27
Availability of data	27	Report which other deliverables can be publicly found and where	Appendix A, page 38

Table B.1: PRISMA 2020 checklist.