Information and Software Technology 159 (2023) 107192

Contents lists available at ScienceDirect

Information and Software Technology

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

Industrial applications of software defect prediction using machine learning: A business-driven systematic literature review

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

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

A R T I C L E   I N F O

Dataset link: https://doi.org/10.5281/zenodo.7476403

Keywords:
Software defect prediction
Machine learning
Systematic literature review
Effort and cost minimisation
Real-world
Industry

A B S T R A C T

Context: Machine learning software defect prediction is a promising field of software engineering, attracting a great deal of attention from the research community; however, its industry application tents to lag behind academic achievements.
Objective: This study is part of a larger project focused on improving the quality and minimising the cost of software testing of the 5G system at Nokia, and aims to evaluate the business applicability of machine learning software defect prediction and gather lessons learnt.
Methods: The systematic literature review was conducted on journal and conference papers published between 2015 and 2022 in popular online databases (ACM, IEEE, Springer, Scopus, Science Direct, and Google Scholar). A quasi-gold standard procedure was used to validate the search, and SEGRESS guidelines were used for transparency, reporting, and replicability.
Results: We have selected and analysed 32 publications out of 397 found by our automatic search (and seven by snowballing). We have identified highly relevant evidence of methods, features, frameworks, and datasets used. However, we found a minimal emphasis on practical lessons learnt and cost consciousness — both vital from a business perspective.
Conclusion: Even though the number of machine learning software defect prediction studies validated in the industry is increasing (and we were able to identify several excellent papers on studies performed in vivo), there is still not enough practical focus on the business aspects of the effort that would help bridge the gap between the needs of the industry and academic research.

1. Introduction

Machine learning software defect prediction (ML SDP) is understood as a set of algorithms that learn by analysing historical data to predict areas of the code that are at increased risk of containing defects. Prediction models are created based on a set of features (usually software metrics, but also change metrics or test repository data) in a supervised or unsupervised manner [1]. Furthermore, there are several possible applications of such models attempting to benefit not only from historical data within a singular software release but also from previous releases, projects, or even different companies. Although very attractive to software businesses and gaining much attention from the academic research community, the field has not yet reached its full potential in commercial applications. Due to the "no free lunch" (NFL) theorem [2,3] and many other factors such as high complexity, low understandability, and expensive maintenance, many companies have not yet introduced ML SDP into their daily operations.

Therefore, we decided to study published research on industry examples of the usage of ML SDP from the perspective of software engineers. The goal is to understand current achievements and synthesise the findings and lessons from such applications to aid similar efforts in the future. In particular, Nokia is an example of a company seeking to improve the quality and minimise the cost of software testing of the 5G technology that it is developing [4] and this study is a baseline for this endeavour.

This paper consists of five sections. Section 1 describes the introduction to the research area and highlights the main objectives and contributions. Section 2 defines the research methodology, the research questions, the selection criteria, and the relevance assessment framework. Section 3 consists of a result analysis, focused on the evaluation of business adoption. Section 4 presents the discussion of the results obtained and the identified threats to validity. Finally, Section 5 offers a summary and conclusion.

* Corresponding author.
E-mail address: lech.madeyski@pwr.edu.pl (L. Madeyski).

https://doi.org/10.1016/j.infsof.2023.107192
Received 10 July 2022; Received in revised form 10 January 2023; Accepted 2 March 2023
Available online 8 March 2023

1.1. Contributions

Our systematic literature review aims to identify, evaluate, and synthesise primary research results to create a precise summary of evidence-based practise in the industry adoption of ML SDP. The most important contributions of the study are highlighted below:

- A list of papers on machine learning software defect prediction in software engineering published from 01.01.2015 to 08.04.2022.
- A list of selected papers focused on the industry application of ML SDP solutions.
- A synthesis of the current state-of-the-art provided by studies satisfying our assessment criteria, describing the details of real-life business applications.
- Answers to our research questions defined in Section 1.2.
- Strategic, contextual, and methodological description to allow similar studies to be performed by other researchers in their own research or business context.
- A SEGRESS checklist for transparency and reporting purposes, as well as all artefacts enabling replicability.
- A set of guidelines and recommendations to identify further research opportunities in the area of ML SDP.
- Finally, this literature review serves as a continuation of the systematic mapping study conducted by Stradowski and Madeyski [5], following a survey study by the same authors [4]. Thus, it is part of the business-driven effort to map and review existing solutions to be able to develop a solution that directly satisfies the business requirements of Nokia. The overarching project aims to improve the quality and minimise the cost of 5G technology software testing at Nokia.

1.2. Research questions

We use Goal Question Metric (GQM) defined by Basili et al. [6] to establish a structured, goal-oriented approach and build a measurement framework for our research purposes. 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 prominent online databases.

We have broken down our goal into six research questions (RQs) that allow us to measure the outcome of our study. Our RQs were formulated as follows:

RQ1. What methods are used for ML SDP in commercial applications?
RQ2. Which features are used for ML SDP in commercial applications?
RQ3. What frameworks are used for ML SDP in commercial applications?
RQ4. What datasets are used for ML SDP in commercial applications?
RQ5. What cost considerations are used for ML SDP in commercial applications?
RQ6. What learnings come from the commercial applications of ML SDP?

1.3. Related work

There is a wide range of valuable secondary research on the topic of ML SDP: Catal and Diri [7], Hall et al. [8], Shepperd et al. [9], Malhotra [10], Wahono [11], Meiliana et al. [12], Durelli et al. [13], Son et al. [14], Pachouly et al. [15]. However, only a limited set of publications offers a broader discussion of the current state of its application in industry.

- Although not in the form of an SLR, Lanza et al. [16] offer a compelling critique of the approach to evaluating defect prediction as part of empirical software engineering research. Most importantly, the authors argue that fields such as defect prediction can be truly validated only if used in vivo, and much more effort should be put into employing proposed predictors to work on real-world problems.
- The work by Garousi and Felderer [17] threads on the topic of low industry and academia collaboration in software testing by analysing three industrial and two academic conferences (based on their representativeness and popularity). The study compares the focus of selected conferences by generating and differentiating word clouds based on presentation titles. The results show significant discrepancies in the interest of academia in exploring theoretical issues and seeking ways to improve the effectiveness of testing by practitioners. In addition, the authors offer insight into the reasons for low collaboration and provide suggestions on how to improve in the future.
- Bowes et al. [18] compare the performance of four ML classifiers to investigate individual predicted defects and analyse the level of prediction uncertainty. The publication contains a detailed description of the background and a review of the literature discussing the impact of the characteristics of datasets on predictive performance. Furthermore, it considers how the vast majority of research is based on NASA and PROMISE repositories, and how validation on in vivo datasets is essential.
- Li et al. [19] surveyed almost 70 representative software defect prediction papers published between January 2014 and April 2017 to review, analyse, and discuss the state-of-the-art. Among many other insights, the authors conclude that the lack of availability of proprietary and commercial data to validate proposed solutions leaves the question of in vivo applicability of models built using data from open source projects unanswered. Further in-depth investigation is advised.
- The key contribution of the study done by Sarker [20] is to provide the principles of different machine learning techniques together with their respective applications in various real-world scenarios. The examples given come from multiple domains such as cybersecurity, Internet of Things, Transportation, Healthcare, E-Commerce, and more; however, none come strictly from the software engineering field.
- Stradowski and Madeyski [5] performed a systematic mapping study of all publications in machine learning software defect prediction. They used keywords from 742 primary studies included in Scopus until February 2022 to confirm that the usage of commercial datasets is significantly smaller than the established NASA, PROMISE, and datasets based on open-source projects. However, the mapping study has also shown meaningful emerging trends in considering business needs in analysed studies such as “just-in-time”, “cost-effectiveness”, “software life cycle”, and “customer satisfaction”.

2. Methods

The research described in this paper adheres to evidence-based software engineering principles and is inspired by the guidelines for systematic literature reviews by Kitchenham et al. [21,22]. The goal is to find as many primary studies relevant to our research questions as feasible, using an unbiased search strategy. First, we clarify our research questions, then search and identify relevant primary studies, synthesise results from the selected sources, and finally explore and answer the designed research questions. Furthermore, we used the SEGRESS guidelines [22] and provided the details in a dedicated checklist available in Appendix B.

Table 1
Search strings.

Database	Search string
ACM	[Abstract: "software"] AND [Abstract: "machine learning"] AND [[Abstract: "defect"] OR [Abstract: "fault"] OR [Abstract: "bug"]] AND [[Abstract: "model"] OR [Abstract: "prediction"] OR [Abstract: "forecast"]] AND [[Abstract: "industry"] OR [Abstract: "commercial"] OR [Abstract: "real-world"]] AND [Publication Date: (01/01/2015 TO *)]
Google Scholar	(("software") AND ("machine learning") AND ("defect" OR "fault" OR "bug") AND ("model" OR "prediction" OR "forecast") AND ("industry" OR "commercial" OR "real-world")) + manual date filter
IEEE	(("software") AND ("machine learning") AND ("defect" OR "fault" OR "bug") AND ("model" OR "prediction" OR "forecast") AND ("industry" OR "commercial" OR "real-world")) + manual date filtering
Science Direct	TITLE("machine learning") AND TITLE-ABS-KEY (((defect OR fault OR bug) AND (prediction OR model OR forecast) AND (industry OR commercial OR real-world))/due to limitations of advanced search + manual date filter
Scopus	TITLE-ABS-KEY(("software") AND ("machine learning") AND ("defect" OR "fault" OR "bug") AND ("model" OR "prediction" OR "forecast") AND ("industry" OR "commercial" OR "real-world")) AND (PUBYEAR > 2014)
Springer	(("software") AND ("machine learning") AND ("defect" OR "fault" OR "bug") AND ("model" OR "prediction" OR "forecast") AND ("industry" OR "commercial" OR "real-world")) + manual date filter

Table 2
Search results.

Database	# of papers	Search level
ACM	50 (most relevant)	Full text
Google Scholar	50 (most relevant)	Full text
IEEE	81	Title, abstract, keywords
Science Direct[a]	22	Title, abstract, keywords
Scopus	178	Title, abstract, keywords
Springer	16	Title

[a]Despite Science Direct has full text search capabilities, we decided to limit it to the title, abstract, and keywords. This way, we were able to build the exact search string as in other databases, as the full text search offers a much more limited string to be used.

Table 3
Search results — details.

Database	# of papers found	% of papers found	# of papers selected	% of papers selected
ACM	50	12.59%	23	20.35%
Google Scholar	50	12.59%	14	12.39%
IEEE	81	20.40%	27	23.89%
Science Direct	22	5.54%	2	1.77%
Scopus	178	44.84%	47	41.59%
Springer	16	4.03%	0	0.00%

Table 4
Inclusion and exclusion criteria.

Inclusion criteria	Exclusion criteria
The paper describes an empirical primary study in software engineering.	The paper was not a peer-reviewed article, conference proceeding, or a book chapter.
The paper is focused on predicting defects in a software system using machine learning techniques.	The paper's language was other than English.
The paper discusses using software defect prediction methods in an industrial setting on an industrial database.	The paper was published before 01.01.2015 or appeared in the searched databases after 08.04.2022.
	The same results were already published in another paper.

2.1. Information sources and search strategy

We created our multi-term search strings improving the work done by Stradowski and Madeyski [5]. In particular, we added industry-related keyword requirements and devised a dedicated search string for each targeted database (see Table 1). Furthermore, we limited the date of publication to no older than 01.01.2015. Despite the possibility of excluding potentially relevant studies [5] due to strategic objectives, Nokia is primarily interested in recent studies and current trends. Based on Dieste et al. [23], we used the following databases: ACM Digital Library,[1] Google Scholar,[2] IEEE Xplore,[3] ScienceDirect,[4] Scopus,[5] and Springer Link.[6] We performed the final search in all databases on April 8th, 2022, finding 397 papers (see Tables 2 and 3). A CSV file with aggregated publications is available in Appendix A.

For each publication, the following data were automatically extracted and compiled into an aggregated CSV (see Appendix A) file: authors, title, year of publication, abstract, and database. Entries from databases that offer more sophisticated export solutions have more data available, such as DOI, keywords, or publisher.

[1] dl.acm.org
[2] scholar.google.com
[3] ieeexplore.ieee.org
[4] sciencedirect.com
[5] scopus.com
[6] link.springer.com

2.2. Eligibility criteria and selection process

The criteria for studies to be included in our SLR are based on the inclusion and exclusion criteria presented in Table 4 and based on the suggestions by Kitchenham et al. [21].

We decided to focus only on recent studies and limit the publication dates. ML SDP is a fast-evolving field, with the number of publications growing every year [5]. Secondly, there is evidence that emerging coding language constructs during the last years caused significant modification of projects in the used public datasets, as shown by Grodzicka et al. [24]. Lastly, Nokia is a cutting-edge technology company wanting to introduce the most innovative solutions, expand the state-of-the-art, and benefit form the newest trends in ML SDP.

We aimed to exclude all work that focuses only on open source and widely tested datasets such as NASA, PROMISE etc., and do not provide any validation in an industrial setting. After applying the inclusion and exclusion criteria to the titles and abstracts of 397 papers retrieved by our automated search in 6 databases, 113 primary studies remained

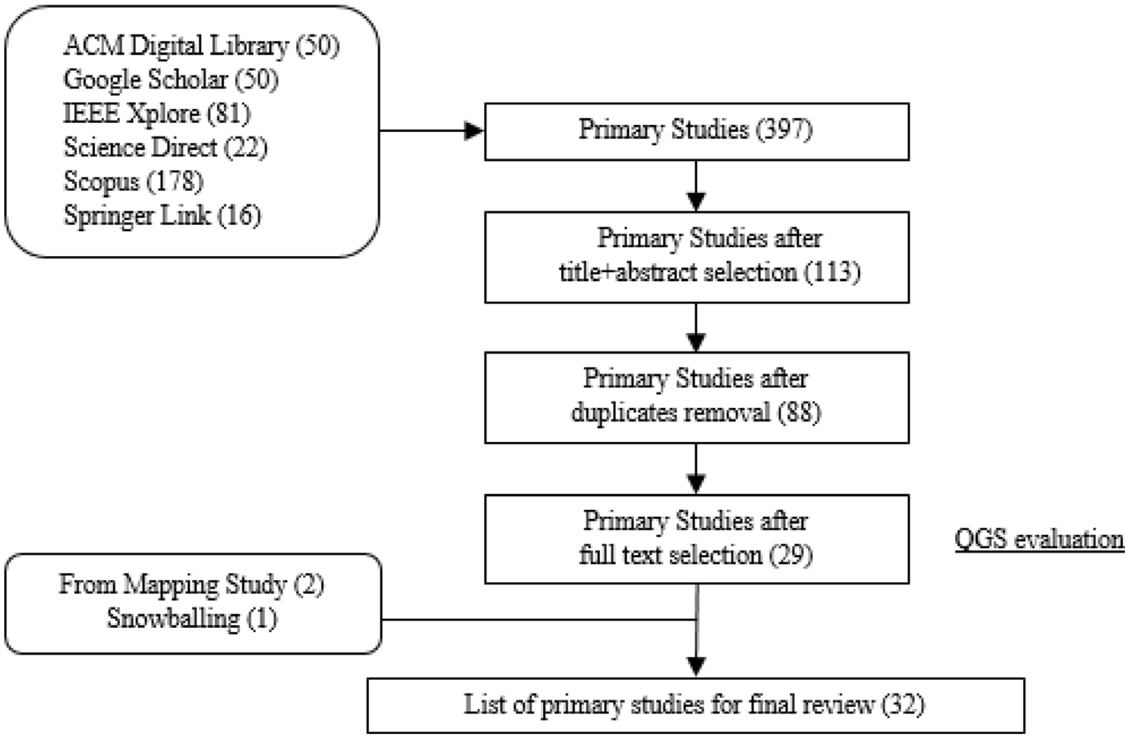
ACM Digital Library (50)
Google Scholar (50)
IEEE Xplore (81)
Science Direct (22)
Scopus (178)
Springer Link (16)
Primary Studies (397)
Primary Studies after title+abstract selection (113)
Primary Studies after duplicates removal (88)
Primary Studies after full text selection (29)
QGS evaluation
From Mapping Study (2)
Snowballing (1)
List of primary studies for final review (32)

Fig. 1. Primary studies identification and selection.

for further analysis. After removing 25 duplicates, 88 studies remained for full-text examination. Next, after reading the full text, another 59 articles were excluded, leaving 29 primary studies to be included in the final review (a complete list is available in the supplementary material in Appendix A).

Our agreement on the selection criteria applied to the initial list of 88 papers was good/substantial (see [21, Table 6.2]), with the Kappa coefficient [21] as below:

$\kappa = 0.73$

Also, during our search and in Stradowski and Madeyski [5] we have identified 13 near-miss studies that appeared to meet the inclusion criteria, but were excluded as not directly addressing the topic of ML SDP: [NM1,NM2,NM3,NM4,NM5,NM6,NM7,NM8,NM9,NM10, NM11,NM12,NM13]

From Stradowski and Madeyski [5] we added two publications ([SLR5,SLR6]) that were not found by our automatic search. Next, we performed backward snowballing [25] through all references form the original 29 selected papers and in the work by Bowes et al. [SLR7] we found another relevant publication: [SLR8]. As a result, we received a hybrid solution in which string-based database searches and snowballing complemented each other and identified a broad spectrum of relevant primary literature. Fig. 1 illustrates the overview of the study selection process.

2.3. Outcome data evaluation

We established our "quasi-gold standard" (QGS) (see also [21,26,27]) baseline in the systematic mapping study by Stradowski and Madeyski [5]. 14 of the selected QGS publications also matched the selection criteria for this study (provided in Appendix A), and our automated search has found 12 of these publications. We used two criteria to assess the automated search — sensitivity and precision, which were calculated as follows:

$$Sensitivity = \frac{R_{found}}{R_{total}} \times 100\% \qquad Precision = \frac{R_{found}}{N_{total}} \times 100\%$$

Where R_{found} is the number of relevant studies retrieved, R_{total} is the total number of relevant studies, and N_{total} is the total number of studies retrieved. The results of our search are as follows:

$$Sensitivity = \frac{12}{14} \times 100\% = 85.71\% \qquad Precision = \frac{12}{397} \times 100\% = 3.02\%$$

We satisfy the threshold for sensitivity recommended by Zhang et al. [26], as 85.71% is above the 70%–80%. Hence, we accepted the search strategy. That said, we do not aim to confer conclusions for the whole population of all published research in the field, but rather to synthesise and learn from the sources we were able to gather.

2.4. Relevance assessment

Next, a relevance assessment was performed to determine the strength of the evidence and recommendations generated by our systematic review [28]. For a business-driven effort intending to support industrial adoption of ML SDP, it is important to evaluate the quality, quantity, and certainty of evidence in the field, as well as how applicable it is to the business context. Therefore, we scored all 32 selected studies in relation to all our RQs [21] from the perspective of Nokia. The questionnaire we used aims to evaluate relevance and significance and assign a subjective score. The grades that can be given to a study in each category are as follows:

- 1, if the study provided ample and highly relevant evidence,
- 0.5, if the study provided adequate and moderately relevant evidence,
- 0, if the study provided little information or not relevant evidence.

Secondly, the criteria used for evaluation were designed to reflect on business applicability and are as follows:

- For '**RQ1.** What methods are used for ML SDP in commercial applications?' we evaluated the number of tested methods, as well as novelty and effectiveness.

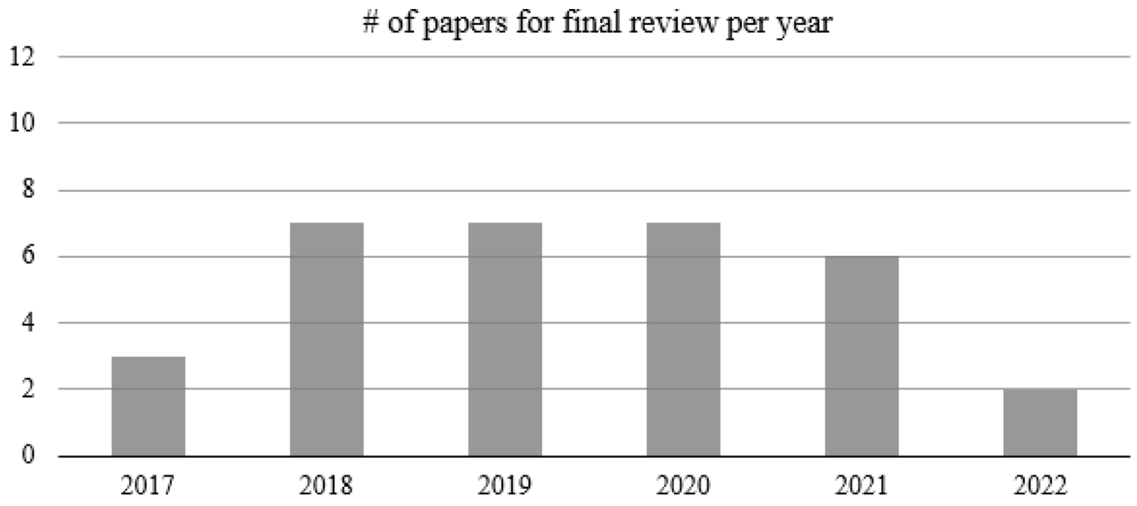

Fig. 2. Number of selected industrial ML SDP publications per year.

- For '**RQ2.** Which features are used for ML SDP in commercial applications?' we evaluated how many different features are taken into consideration to build the models.
- For '**RQ3.** What frameworks are used for ML SDP in commercial applications?' we evaluated the completeness of the description of how the tooling works and how it was used in the real-world setting.
- For '**RQ4.** What datasets are used for ML SDP in commercial applications?' we evaluated the approach to in vivo validation from the perspective of how much original insight it brings.
- For '**RQ5.** What cost considerations are used for ML SDP in commercial applications?' we evaluated any evidence of cost and benefit evaluation of the introduced solution.
- For '**RQ6.** What learnings come from the commercial applications of ML SDP?' we evaluated the emphasis of sharing any guidelines and good practises that were highlighted during the study for future researchers and practitioners to learn from.

Two independent researchers evaluated the search papers based on relevance assessment questions to mitigate personal bias, and a sum of grades was calculated. It is important to note that we included all selected primary studies in our SLR independent of the results in each category. Industry research is scarce enough, and we did not wish to limit it beyond our selection criteria and decided that all publications are valuable in this regard (detailed results are available in Section 3). However, we offer our risk of bias and certainty assessment in the next section to highlight the risks that we have observed.

2.5. Study risk of bias assessment

Any systematic review needs to provide an overall assessment of the risk of bias (RoB) for the primary study and each domain it reflects upon [22].

The approach we used relies on the suggestions by Dybå and Dingsøyr [29]. Out of 11 original questions we have selected the five most relevant for our business-driven review. We have assessed each study with the first three questions as "Yes" and "No", while the remaining questions in terms of a four-level scale ("Very low", "Low RoB", "Moderate RoB", and "High RoB") as recommended in the SEGRESS guidelines by Kitchenham et al. [22]. The evaluation was done by two researchers independently, and the results are available in supplementary material in Appendix A.

(1) Is the paper based on research?
(2) Is there a clear statement of the aims of the research?
(3) Is there an adequate description of the research context?
(4) Is there a clear statement of findings?
(5) Is the study of value for practice?

Secondly, we expect that the risk of bias within the main focus of our study (research done in industry settings) cannot be neglected and needs to be understood by Nokia. Due to many external factors and limitations that need to be accounted for in vivo, purely academic studies have more opportunities to mitigate reporting biases.

Also, we suggest the risk of bias due to missing results in our analysis is high (see also threats to validity in Section 4.3), as there are no reliable methods for identifying research done in vivo besides automatic search and manual selection by keywords and full-text reading, as well as no secondary research to benchmark with. As a consequence, we do not generalise our findings but focus on individual learnings and conclusions derived from analysed papers, accounting in that the overall certainty of conclusions is lowered.

2.6. Data extraction and synthesis

Fig. 1 illustrates the overview of the selection process, where 397 publications were found based on our defined search terms in six online databases. After screening and application of the selection criteria, 32 primary studies remained for further analysis. During the advanced search execution, we automatically extracted the following data structure and combined it into a singular CSV file:

- Title,
- Authors,
- Year,
- Abstract,
- Database.

Secondly, we manually combined a BibTeX reference file. During synthesis, we have accumulated and combined the inputs from both researchers for data inclusion/exclusion, RoB, and quality of evidence evaluation. Next, we have built graphs and charts from the evaluations of selected primary studies to formulate and visualise responses to the posed research questions. All mentioned artefacts are available in Appendix A.

3. Results

All 32 selected papers (see Fig. 1) were read in full and evaluated according to our relevance assessment Section 2.4. The number of papers per year is presented in Fig. 2.

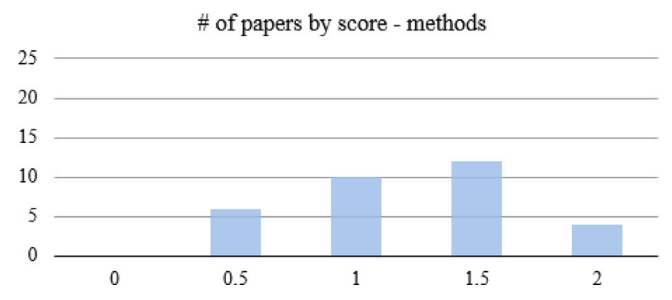

Fig. 3. Relevance assessment of papers from the perspective of methods used in ML SDP in commercial applications.

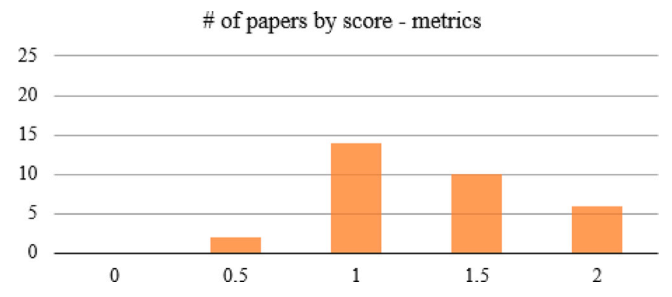

Fig. 4. Relevance assessment of papers from the perspective of features used in ML SDP in commercial applications.

Full list of papers is available below and in references section Systematic Literature Review: [SLR9,SLR10,SLR11,SLR12,SLR13,SLR14, SLR15,SLR16,SLR3,SLR17,SLR18,SLR19,SLR7,SLR20,SLR21,SLR22,SLR23, SLR24,SLR25,SLR26,SLR27,SLR28,SLR29,SLR30,SLR31,SLR32,SLR1,SLR2, SLR4,SLR5,SLR6,SLR8]

3.1. Study characteristics and individual results

We have built a simple statistic showing our relevance assessment of all included publications based on our full-text reading to answer the research questions. Secondly, we chose and summarised the three most impactful works for each research question to highlight the most notable implications of the review effort. Notably, a handful of studies were chosen in more than one category, which shows that they accurately addressed multiple challenges of industrial applications.

RQ1. What methods are used for ML SDP in commercial applications?

Our evaluation of the relevance of the research from the perspective of the methods used is presented in Fig. 3. There were many good studies, either providing a comprehensive explanation of the techniques used, using very sophisticated methods that had not been tried before in any setting, or employing a wide variety of classifiers to select the best performing one. Overall, methods score was the highest of the six RQs.

Three selected papers from the perspective of the methods used include the following:

- Melo et al. [SLR22] created a guide to support the prediction of change proneness for maintenance teams to enable the identification of change-prone classes in the early phases of software development to improve software quality, based on the back-end source code of a WEB application. The guide proposes a straightforward step-wise approach: phase 1 designing the data set - choosing the independent variables, choosing the dependent variables, and collecting features; phase 2 - applying prediction - statistical analyses, normalisation, outlier detection, feature selection, resampling and cross-validation, tuning the prediction model, ensuring the reproducibility. The guideline is then validated in vivo on a web application's back-end source code, using ten classifiers: LR, LightGBM, XGBoost, DT, RF, KNN, Adaboost, Gradient Boost, SVM with Linear Kernel and SVM with RBF kernel. Notably, the case study used Python 3.7 on the Anaconda platform and Jupyter Notebook 5.6.0, which is readily available to any practitioner.
- The highest number of different techniques was analysed by Malhotra and Sharma [SLR21]. The study on fault prediction models for identifying fault-prone classes in web applications compares 14 machine learning techniques and explores their relationship with 18 object-oriented metrics. The study uses open-source Apache Click and Apache Rave projects developed under the Apache Software Foundation to rank the predictive ability of different fault prediction models and then statistically compare them. The techniques used in the study are: Statistical classifiers (BN, LR), Decision tree (DT, REPT, RT, J48), Support Vector Machines (SVM, VP, SMO), Neural Networks (MLP), and Ensemble learning (Bag, RF, LB, AB). MLP turned out to be the most effective methodology on the data set used, validated with a significant pair-wise difference.
- We chose to highlight the study carried out by Altinger et al. [SLR12], as it analyses fault prediction on high-quality industry-grade software using the readily available WEKA tool (see also in RQ3 Section 3.1). Authors use as many as seven different classifiers to analyse the effects of under- and oversampling the training data, concluding that different classifiers are affected differently. Furthermore, the paper is one of the first ones applying ML SDP to commercial software from the automotive industry.

Table 5
Top18 most frequently used classifiers in analysed papers.

Classifier	# of uses	Classifier	# of uses	Classifier	# of uses
NB	11	KNN	5	BN	2
RF	11	J48	4	Bagging	2
LR	10	CNN	3	GB	2
SVM	9	AdaBoost	3	MLP	2
DT	6	XGBoost	2	LSTM	2
NN	6	DL	2		

While counting the most popular methods, we have consolidated variants of the same classifiers. We found 108 instances of classifiers in our selected studies, 38 unique ones, and 17 that occurred more than once (see Table 5). The most popular classifiers among our selected industry papers are the same as suggested by Pachouly et al. [15] as most popular among all ML SDP studies.

RQ2. What features used for ML SDP in commercial applications?

Our evaluation of the relevance of research from the perspective of the metrics used is visualised in Fig. 4. The metrics discussed in the reviewed papers were naturally dependent on the subject studied. The majority provided a sufficient explanation of their meaning and how they were used; however, only a subset specifically addressed how they are gathered in the industrial examples, as it may be substantially more difficult than in more artificial settings. (see Fig. 4).

Three selected research studies from the perspective of the used features include below papers:

- dos Santos and Figueiredo [SLR23] conducted an exploratory study of software features for defect prediction. The authors analyse three groups: class-level metrics, entropy metrics, and change metrics. However, the study is barely an industry project (the analysed dataset contains five large Java projects: Eclipse JDT and PDE, Equinox, Lucene, and Mylyn); it offers a clear explanation of seven used ML algorithms and, even more importantly, an exhaustive presentation of three types of metrics.

Class-Level Metrics contain 17 items in two groups: Chidamber & Kemerer and Object-Oriented. Secondly, the entropy level includes five metrics related to entropy. Finally, eight change metrics are also used. Although it is difficult to imagine that such metrics would be available in an entirely industrial setting, it showcases the potential possibilities. Furthermore, the study also highlights the importance of the understandability of the features, which may be valuable for implementation in commercial settings.

- A study by Wang and Khoshgoftaar [SLR28] attempts to determine the most optimal set of software features for defect prediction. The experiments are conducted based on software metrics and defect data collected from a vast telecommunications software system. The software measurement dataset contains 42 software metrics, including 24 product metrics, 14 process metrics, and four execution metrics. Out of the three classes of feature selection, the wrapper-based subset selection approach performed best, filter-based subset evaluators second, and the feature ranking performed worst.
- Shippey et al. [SLR10] published a paper specifically on the automatic identification of code features for SDP. Furthermore, the authors validate their method in vivo using a large international telecommunications company based in the UK. They successfully apply Abstract Syntax Tree (AST) n-grams to identify features of defective Java code and improve defect prediction performance. The models were created with the default set of static code metrics calculated by the JHawk program, which contains many method-level and software code metrics. The results show that in some systems, the AST n-grams are meaningfully related to faults with substantial effect sizes and that AST n-grams can significantly affect the performance of SDP models.

There is an overwhelming number of metrics used in the analysed materials, so we decided against synthesising all in detail. However, having our overarching business introduction goals in mind, we focused on selecting the ones that emerged as considered the most important by the authors and coincided with the conclusions done by Pachouly et al. [15]. The most important training metrics are focused on process, source code, and historical defects. The metrics used in the industry seem to be very widely distributed and we were not able to identify specific ones that repeat as considered to be most effective or most frequently used.

Notably, metrics used in the industry seem to be restricted mainly by the availability and limitations of the environment the study is run in. A further consideration is available in the RQ2 part of the discussion in Section 4.1.

RQ3. What frameworks used for ML SDP in commercial applications?

Our evaluation of the relevance of research from the perspective of the frameworks used is visualised in Fig. 5. We consider the frameworks used in the analysed literature to be well described. Many have been designed from the beginning to be used in a specific industrial setting and fit a particular business purpose. However, the downside of such an approach may be low transferability. Only few were more generic and verified in vivo among more standard datasets. A broader usage in different circumstances allowed accurate verification and more improvement opportunities. See Fig. 5.

Three selected papers from the perspective of the used frameworks include:

- A study by Kawalerowicz and Madeyski [SLR6] describes a continuous build outcome prediction in a real software project utilising Jaskier [30] — a tool built for that purpose by the authors. The tool uses ML models that predict the continuous integration (CI) build results based on historical results combined with metrics gathered and calculated in real-time from the software repository. In the experiment, 310 project days coming from a total of 9 developers were analysed to show whether the prediction of the result of the build can reduce the number of failed builds. Notably, the tool itself is open-source and "consists of three developer clients (for Visual Studio, VSCode and command line), a web service to facilitate the data exchange between the clients and prediction model living in the cloud or on on-premises machine learning server", which offers significant potential for customisation and industry implementation.
- Bowes et al. [SLR27] addressed the claim that, despite the vast amount of research done on ML SDP, it has not been transferred to industrial practise. Therefore, the authors introduce the Ensemble Learning for Fault Finding (ELFF) tool to improve the availability of defect prediction tools that practitioners can use during their day-to-day operations. It contains two main parts — the back-end that gathers historical defect information, collects source code metrics, and performs defect prediction, and the front-end offers results visualisation with an integrated development environment. Importantly, ELFF was validated in vivo in a large telecommunication project and was positively evaluated by its users.
- A study by Hryszko and Madeyski [SLR8] was carried out in Volvo Group on a document management system, to which the researchers employ their machine learning-based tool (DePress Madeyski and Majchrzak [31]) to analyse the cost-effectiveness of software defect prediction in an industrial setting. The tool used is an open-source software measurement framework jointly developed by Wroclaw University of Science and Technology and Capgemini for defect prediction in commercial software. DePress is built on KNIME and allows the development of graphical workflows with a graphical interface. The desired data is collected from the SVN version control system and Jira as the defect report application. The obtained results show that even using default settings, low investment, and high effectiveness give a cost-efficient solution that significantly improves software quality.

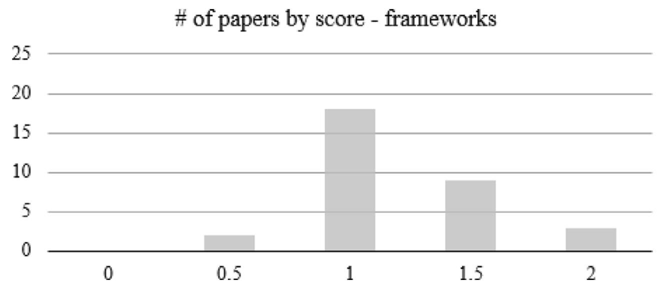

Fig. 5. Relevance assessment of papers from the perspective of frameworks used in ML SDP in commercial applications.

While counting the most popular frameworks, we have all papers and excreted the information on the frameworks used for building the models. We found 16 categories in our selected studies, highlighted in Table 6. Unfortunately, many authors provide insight into how their tooling works without indicating how it was written. From established frameworks, WEKA, Scikit-Learn, and Keras, TensorFlow combination are the most popular among our selected papers.

RQ4. What datasets are used for ML SDP in commercial applications?

Our evaluation of the relevance of research from the perspective of the datasets used is visualised in Fig. 6. The quality of the description varied as many studies only briefly described the dataset used (being unable to disclose even the name of the company or for the sake of brevity). Also, we obtained a wide plethora of industries studied from

Table 6
Breakdown of used frameworks in analysed papers.

Framework	# of studies	Framework	# of studies
WEKA	7	PyTorch	1
Own tool, no information	5	Word2Vec, Doc2Vec	1
Own tool, Python	5	Azure ML Studio	1
Keras, Tensorflow	3	MATLAB	1
No information	3	DGCNN	1
Scikit-Learn	2	JIT-SDP	1
Code2Vec	1	Java apps. and R scripts	1

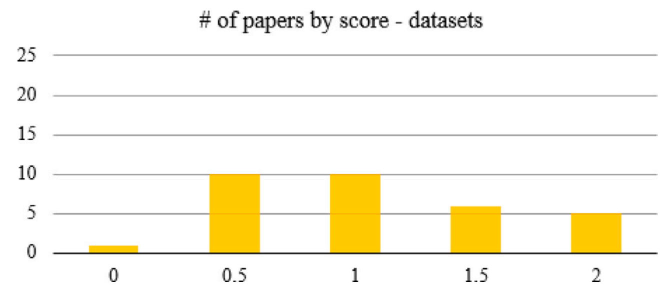

Fig. 6. Relevance assessment of papers from the perspective of datasets used in ML SDP in commercial applications.

maritime, finance, automotive, to telecommunication, which was by far the most popular source of data. See Fig. 6.

We selected four papers from the perspective of the datasets used to be highlighted.

- Pradhan et al. [SLR1] studied defect prediction for large-scale software systems. The research focuses on challenges resulting from the scale of hundreds of millions of lines of code based on an example of Cisco's IOS-XE, consisting of more than 2200 software components and tens of thousands of source files. To address said challenges, the authors propose that software defect prediction for large-scale software should include four components: data definition, quality attributes selection, ML algorithm, and SDP life cycle. Using this approach, several models are presented, with the remark that the solutions are specific to the dataset used in the study, but general conclusions on the importance of these challenges should apply to all large-scale software.
- An approach used in a subset of studies, with an excellent example from Shippey et al. [SLR10], is first to train the method and validate using a popular database like NASA, Eclipse, PROMISE, and secondly to use an in vivo dataset. Such an approach allows for benefiting from benchmarks with other studies on the same sets, as they are well understood and explored, and validating in a real-life setting in the second step to confirm industry applicability. The authors highlight that the Eclipse open-source Java system analysed in their study was chosen as already used extensively in defect prediction studies and allowed validation of the proposed technique for locating faulty code. Secondly, the authors collected fault data from two commercial telecommunications systems to test the solution to different problems.
- Another excellent example of employing such a step-wise approach is the work done by Bowes et al. [SLR7]. The authors highlight that datasets can significantly affect predictive performance and aim to analyse datasets with varied characteristics from different contexts. The research compares the performance of four classifiers across NASA, open source, and commercial datasets (unnamed UK-based telecommunications company), uncovering significant inconsistencies in performance. Most importantly, defect prediction models may perform similarly when judged by performance measures but may identify different defects. Thus, ensembles of classifiers are recommended, as using only one classifier is not likely to comprehensively detect defects.
- The industry study by Zong [SLR25] focuses on a classification-based software defect prediction model for an unnamed finance software system. The authors point out that financial software systems are different from traditional in several ways and that there is no available benchmark for software defect prediction in the finance industry. Therefore, the acquisition and use of commercial data is the only way to validate the proposed solution. The authors then propose a binary classification defect prediction model with features specific to finance software systems. Moreover, they discuss how to apply the prediction output and how it benefits the software development process.

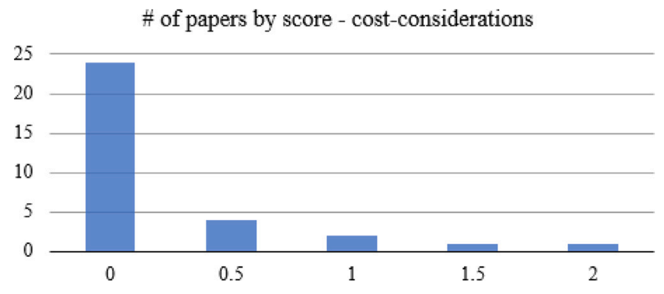

Fig. 7. Relevance assessment of papers from the perspective of cost considerations used in ML SDP in commercial applications.

One of the most concerning observations we have made is that majority of the studies were unable to disclose the commercial datasets that were used and did not introduce them to the public domain for further research (as was done, for example, by Jureczko and Madeyski [32]). Without more industry datasets being available publicly for further investigation, the overall business adoption of ML SDP will be slowed down and replications impossible. Therefore, publishing the datasets, e.g., in services like Zenodo, should be of utmost importance. Second, we suggest that authors who do not have the possibility to publish the dataset should state this clearly in the paper rather than not comment on these circumstances at all.

Full or partial dataset publication was found in the following 17 (out of 32) of the analysed papers that have been published between 01.01.2015 and 08.04.2022: [SLR2,SLR3,SLR11,SLR23,SLR14,SLR4,SLR6,SLR26,SLR20,SLR21,SLR22,SLR31,SLR30,SLR15,SLR9,SLR29,SLR24].

RQ5. What cost considerations are used for ML SDP in commercial applications?

Our evaluation of the relevance of research from the perspective of cost considerations used is visualised in Fig. 7. The topic of cost-effectiveness of implemented solutions was very rarely discussed in the reviewed publications. The majority opened with a justification of the research, but was limited to generic statements without further consideration during the implementation in the real environment. Therefore, we selected and summarised only two publications that offer a comprehensive understanding of the cost considerations behind the conducted research. See Fig. 7.

We selected only two papers that offer a wider commentary on the cost consciousness of studied topic:

- In the Volvo study already highlighted for the used framework, Hryszko and Madeyski [SLR8] committed a complete publication on cost-effectiveness and placed a significant focus on providing financial justification for using software defect prediction in commercial settings. The investment cost is calculated based on tool acquisition, training, data collection, and prediction preparation. Benefits are calculated as a multiplication of the average person-hour work cost saved by test engineers. As a result, the simulated

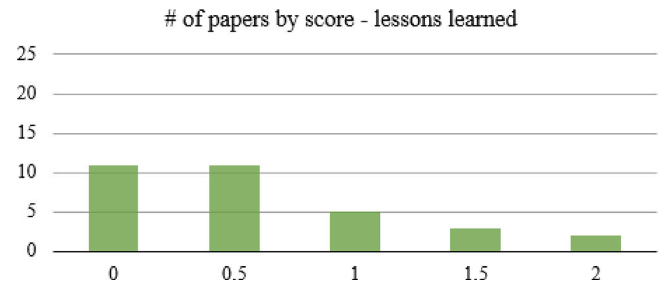

Fig. 8. Relevance assessment of papers from the perspective of lessons learnt used in ML SDP in commercial applications.

estimations show that quality assurance costs can be reduced by almost 30%, the estimated Return on Investment (ROI) is a staggering 73 (7300%), and Benefits Cost Ratio (BCR) equals 74. This is an excellent illustration of how SDP can benefit commercial companies in their quality assurance effort.

- The study by Kang et al. [SLR32] focuses on predicting just-in-time software defects to reduce post-release quality costs in the maritime industry. The authors offer a description of the proprietary software systems used on large ships that are used for maritime transportation, including workstations, displays, controllers, and control networks, the details behind the proposed solution, and an analysis of the results. However, a genuinely distinguishing factor for the publication is a cost–benefit analysis. Two of the five research questions focus on increasing test efficiency and reducing post-release quality cost. Both considerations adhere precisely to the business reasoning needed for similar solutions to be applied in the industry, making this publication a prime example of how to increase in vivo applicability of academic research.

RQ6. What learnings come from the commercial applications of ML SDP?

Our evaluation of the relevance of research from the perspective of the lessons learnt is visualised in Fig. 8. By definition, reviewed publications executed in vivo contain valuable insight into leanings or commercial applications. Most offered such suggestions as part of the typical description of the proceedings; therefore, we sought dedicated discussions on aspects such as lessons learnt or commentary on received feedback. Additionally, some are named and organised in the form of a guideline that practitioners can follow. See Fig. 8.

Three selected papers from the perspective of the lessons learnt include the following:

- Dam et al. [SLR11] provided a whole paper in the form of a lessons learnt from applying ML SDP in practice. The study was run on the PROMISE repository and an open-source project by Samsung. The models are built with Long Short Term Memory network, which directly matches the Abstract Syntax Tree representation of the source code of both projects. The authors offer six extensive lessons learnt from developing the model and validating the results on topics like explainability, training time, or heterogeneity of code bases. Furthermore, the paper underlines the importance of feedback received from practitioners using the proposed solution.
- In the study Zhao et al. [SLR18] 11 real-world online service systems from two large commercial banks were analysed, demonstrating the effectiveness of their tool for real-time incident prediction with alerts. In addition to describing the approach and implementation, the study includes dedicated sections on interpretability, success stories, and lessons learnt. All three contain valuable insight for practitioners aiming to start similar efforts. Interpretability adheres to XAI (the only other publication that touches upon XAI is dos Santos and Figueiredo [SLR23] — the need to explain prediction results and help engineers understand incidents in practice. Success stories cover four cases of the tool that accurately forecast incidents in real-world operations. Finally, the lessons learnt offer three observations that are specifically highlighted for practitioners who want to use such an approach in the future.
- Already praised for its cost–benefit section, work by Kang et al. [SLR32] also offers a comprehensive description of the lessons learnt. Employing just-in-time software defect prediction to reduce post-release quality costs in the maritime industry has led the authors to many insightful conclusions and meaningful recommendations. The highlights include the proven applicability of SP to a made-to-order sector where each application has different requirements and the need for software architecture to support separate modules that provide commonality and offer variability. Furthermore, the section describes interviews conducted with various stakeholders to elicit feedback (on management aspects like implementation cost estimations and developers on the additional overhead). Finally, there are also seven distinct guidelines for practitioners aiming to employ defect prediction in the future.

3.2. Reporting biases and certainty of evidence

As described in Section 2.5, the risk of bias in reviewed studies is relatively high. Researchers frequently mentioned the limitations posed during the research in the industry as a significant portion of influencing factors were outside of their control. Nevertheless, our main effort to introduce ML SDP in Nokia system-level testing of 5G will have similar limitations increasing the overall risk and endangering the provided certainty of evidence. Therefore, considering the business goals behind our effort, we focus primarily on the subject and content of the primary publications.

Our risk of bias evaluation had the following Kappa coefficients [21] measuring the consensus between both researchers:

$$\kappa_{(1)} = 0.65 \quad \kappa_{(2)} = 1 \quad \kappa_{(3)} = 1 \quad \kappa_{(4)} = 0.81 \quad \kappa_{(5)} = 0.72$$

The overall agreement was good/substantial to very good/almost perfect (see [21, Table 6.2]); nevertheless, each risk evaluation disagreement on the five selected questions described in Section 2.5 has been discussed and resolved.

Secondly, reporting bias may be negatively influenced by additional stakeholders interested in the results in industry settings. Both practitioners participating in the experiments and management approving any cost related to the activities expect high returns. That may put further pressure on researchers. Another significant constraint represents the limited time spent on in vivo research as it is expected to be brief and practical rather than thorough and complete. Therefore, publications on any newly proposed solutions should be twofold. One in a purely academic setting, satisfying rigorous requirements for the risk of bias and certainty of the evidence, and the second with in vivo validation highlighting the validated results, but also risks resulting from additional limitations and confirmation bias.

All in all, uncertainty is embedded in all business endeavours and can never be fully understood. Therefore, increased reporting risks and threats to validity in business-driven efforts need to be clearly discussed, but also accepted to a certain degree.

4. Discussion

In order to derive meaningful conclusions that practitioners can benefit from this research, we have discussed each RQ from the perspective of business applicability and, where possible, we also referenced similar studies. Importantly, to further validate our results and review the discussion, we invited three experienced practitioners from Nokia to help

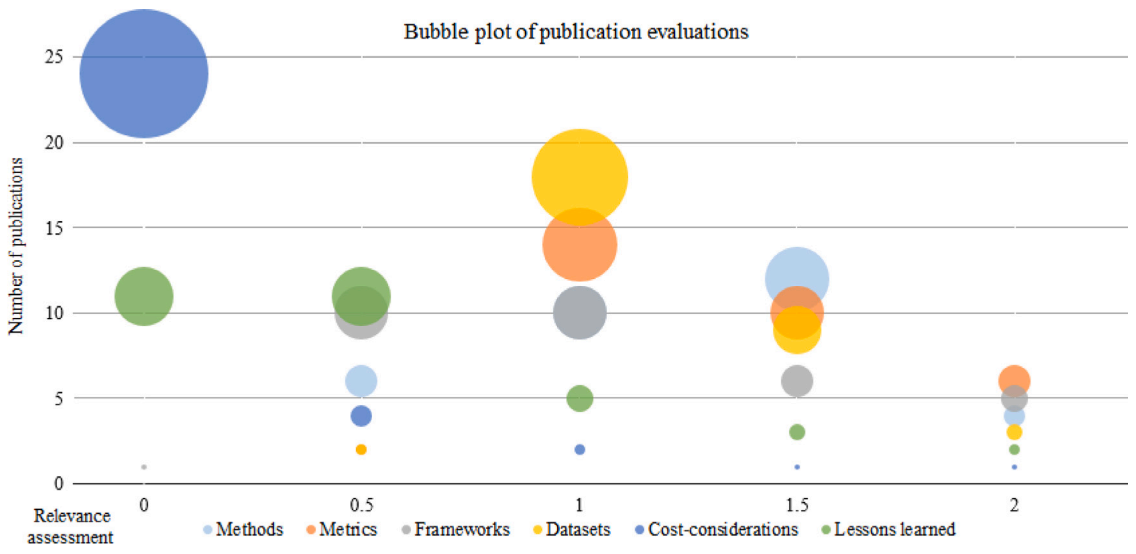

Fig. 9. Bubble plot of publication relevance assessments.

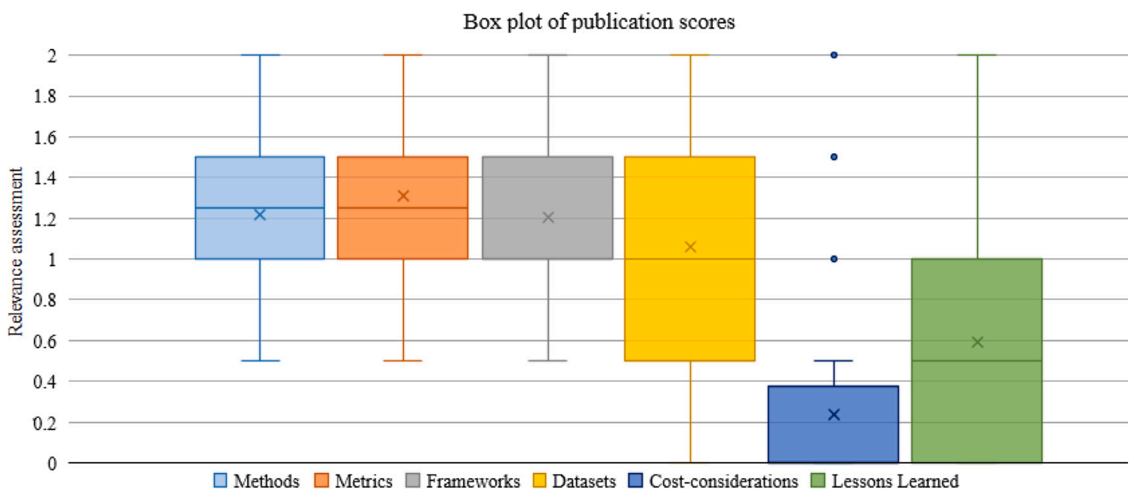

Fig. 10. Box plot of publication evaluations.

us during the discussion effort. The purpose of inviting practitioners was to increase the value of the proposed recommendations and share further insight from an industry perspective.

Secondly, we would like to highlight again the discrepancy in the quality and quantity of evidence between methods, features, frameworks, and datasets (RQ1–RQ4), which were much more comprehensive than cost considerations and lessons learnt (RQ5 and RQ6). The discrepancy can be observed in the below bubble plot (Fig. 9). The bubble size reflects the number of the most frequent evaluations for each criterion, and it is clearly visible that we found much more evidence for the first four categories.

We created a box plot to visualise the distribution of grades and their skewness by displaying the data quartiles, averages, and variability outside the upper and lower boundaries (Fig. 10). All RQs have an extensive distribution and variability, showing a meaningful difference in our perceived relevance. Also, the discrepancy in the evaluations between RQ1–R4 (methods, features, frameworks, and datasets) and the much lower amount of evidence for RQ5–RQ6 (cost considerations and lessons learnt) can be observed. Although all categories have publications what were evaluated with the highest possible score ("2"), none of the methods, features, and frameworks categories received the lowest one ("0") .

Over the years, ML SDP branched out into many tracks tackling different obstacles for creating meaningful and accurate models form existing data (examples provided below), however, we found limited traces of utilising those aspects in the reviewed industrial papers:

- Semi-supervised learning [33] — no papers,
- Prediction without historical data [34] — no papers,
- Cross-project defect prediction [35] — two papers: [SLR9,SLR5],
- Cross-company defect prediction [36] — one paper: [SLR9],
- Heterogeneous defect prediction [37] — no papers,
- Noise in data [38,39] — despite majority of papers highlight significant data imbalance, only standard mitigation techniques are used,
- Just-in-time defect prediction [40] — [SLR32,SLR5,SLR6].

Finally, we would like to highlight that the ML-based approach to SDP also has several limitations that hamper industrial application in comparison to search-based [41] and rule-based techniques [42]. We have identified similar obstacles to the ones described by Pachouly et al. [15], Pradhan et al. [SLR1], Tantithamthavorn and Hassan [43]:

- Challenging availability of data in terms of its quantity and quality,
- Many performance measurement misinterpretation pitfalls,
- Difficult to achieve interpretability of predictions,
- Demanding implementation and longer calculation times.

4.1. Answers to research questions

Based on the search, selection, and relevance assessment process execution described in Section 2, as well as the observations provided in Section 3, together with practitioners from Nokia we have drawn several conclusions for machine learning in software defect prediction highlighted below.

RQ1. What methods are used for ML SDP in commercial applications?

We found the methods used in our reviewed papers to provide very valuable conclusions (Section 3). However, a significant part of the papers offered an analysis of only one method, which we find incomplete. A better approach to proposing a new technique or its adaptation to new datasets is to benchmark against well-established practises. Thus, we evaluated most highly the solutions where results from several techniques were compared to select the best in given circumstances, to account for the "no free lunch" (NFL) theorems [2,3]. In the reviewed papers, we also observed that almost all prevalent techniques were validated in industry. Thus, majority of techniques can potentially find a practical application with satisfactory results, as from a business perspective, the key differentiators for each dataset are the features used (and available) and the framework that should allow several methods to be executed and compared [SLR21]. Even more so when different methods find different errors, as pointed out by Bowes et al. [SLR7].

Second, we have not observed significant differences between the set of methods used in our industry-focused study and the findings of other literature reviews in the field. There are many good SLRs (see Section 1.3) to compare our data with; however, we chose Pachouly et al. [15] as the most recent and directly attempting to answer a subset of our RQs. The aforementioned publication is not industry-focused, but it allows insight into whether similar conclusions can be drawn in academic and practical considerations. Most importantly, we confirm also in industry research the most popular methods are Linear Regression, Naive Bayes, Logistic Regression, Decision Tree, SVM, KNN, and Random Forest (see Table 5).

Importantly, we have found several deep learning applications in the industry [SLR11,SLR15,SLR16,SLR3,SLR17,SLR29,SLR30,SLR2]. Despite differences in the approaches used, the mentioned studies confirm the applicability of deep learning techniques in industry and show significant future potential. Employing multiple layers requires greater effort during introduction and maintenance, but often rewards via higher predictive performance, which may be attractive to more risk-averse commercial environments such as software products for healthcare, automotive, or finance.

Lastly, on a broader scale, the purpose of ML SDP is to complement more traditional test processes. Therefore, out of all the main performance measures [44] practitioners tent to value fewer false-positives (misclassification or error rate) as well as very high accuracy or precision of the prediction. Dealing with false-positives is always noise for an R&D organisation, whereas every true fault may be an added value confirming, or in the best case, uncovering a new existing defect.

RQ2. Which features are used for ML SDP in commercial applications?

Similarly to the methods reviewed, we found the features used in the publications to be satisfactory; however, they were slightly closer to the middle value of "1" (Section 3). A significant part of the models was trained on a very high number of features (as many as 18 in [SLR21] in 14 in [SLR32]). Also, we agree with Pachouly et al. [15] that "most of the studies employed the following features to train the model: process-related metrics, attributes taken from historical defects, source code metrics in general, and object-oriented metrics".

The difficulty comes from the availability of metrics that can be gathered in vivo and, what can be even more difficult, what is historically available. From this perspective, in many cases, it is the model that needs to be robust enough to work with what is available to be reusable. 0f course, dedicated solutions optimised to work in one particular dataset and one company, this difficulty can be overcome with more ease, and cooperation with practitioners to introduce more metrics is more feasible. A review of software metrics for fault prediction using ML approaches with PROMISE repository dataset done by Meiliana et al. [45] has shown that, on average, three software metrics are sufficient to build practical software prediction models.

Therefore, we conclude that from a features perspective, there is no need to include a large number of metrics that improve final performance by a small difference, but balance with what is possible and available in the target context.

RQ3. What frameworks are used for ML SDP in commercial applications?

Many excellent frameworks were described in the reviewed publications. None were evaluated on a "0", with majority on "1" and "1.5" (Section 3). However, the publications varied in the number of details offered on the internal mechanics of the tool. A good practise seems to be not to try to fit the tool description and the result analysis in one article, but to split it into two separate publications that offer more information on each (for example, the research in [SLR6] and the tool in [30]). One distinguishing factor for the offered solution is proven compatibility with popular frameworks like Jenkins, or any Microsoft-based platforms that are widely used in the industry. Offering a straightforward interface towards an already used solution largely decreases the implementation difficulty and automatically provides access to already existing data without additional modifications.

Furthermore, many publications focus only on how the tool works and not on how other researchers and practitioners could use it (see RQ6 below). We did not encounter the same newly proposed tooling for use by different researchers, as each created their own. However, a significant part of the research reused the well-established TensorFlow[7] ([SLR15]) and WEKA[8] ([SLR13,NM1,SLR28,SLR32]).

Unfortunately, only two authors ([SLR23,SLR18]) reflected on the concept of explainable AI (XAI), which can be critically important in specific industries and is considered a very attractive concept from the perspective of practitioners. Finally, after reviewing all the tools, we conclude that the dream state for a reusable tool would be an online, standalone, non-intrusive software that is easy to install, has a clear user-friendly interface, offers XAI interpretations, has defined interfaces to well-known databases that are already popular in industry, and has implementation-focused documentation. Any tool offering at least part of these characteristics has a better chance of being implemented on a larger scale.

RQ4. What datasets are used for ML SDP in commercial applications?

Considering that the automatic search query and the selection process used by design scrutinised the datasets on which the solutions were validated, the relevance evaluation of the evidence was moderately high. Interestingly, despite the fact that the methods and features having higher overall scores, datasets had the most studies evaluated at the maximum grade of "2" (Section 3). Therefore, we received a wide spread of industries from finance, finance through transportation, automotive, to telecommunications. The latter was the most frequently used, showing an increased interest in pursuing ML-based solutions within the industry.

[7] TensorFlow is an end-to-end open-source platform for machine learning offering multiple tools and libraries allowing to build and deploy ML applications. URL: https://www.tensorflow.org/.

[8] Weka is an open-source software collection of machine learning algorithms and tools for data mining tasks. URL: https://www.cs.waikato.ac.nz/ml/weka/.

Notably, we identified a particularly efficient approach to validate the solution proposed in the study on a few different datasets ([SLR10,SLR11,SLR7,SLR5]). The approach was to first validate on well-explored sets like NASA or PROMISE. The benefit came from the sets being very well known and having plenty of research to benchmark against. The solution was then used on a lesser-known open-source project to pilot and study the performance in circumstances closer to a real environment. Finally, the proposed methods were tested on an industrial dataset to verify the solution in vivo and have a complete set of research results. Solutions validated in such a way have the best chances of being most effective in various circumstances. Secondly, many of the reviewed papers purposefully aimed to validate the solutions on many datasets, but came from one category of several open-source projects or a few sets of data from the same company or industry. Of course, the more datasets the solution is validated against, the better.

Nevertheless, as studied by Stradowski and Madeyski [5], Lanza et al. [16], Li et al. [19] the overall statistics of all ML SDP primary publications show that industry application is relatively low. Due to the "no free lunch" (NFL) theorems [2,3] and the fact that datasets can be very different and verification in one instance does not bring full benefit from the perspective of industry application. Therefore, validation on several datasets is critically essential, and we would like to suggest that researchers more frequently reach out to companies asking if they would be interested in validating new methods on proprietary data. We acknowledge that many attempts will fail, but there are chances that some success will increase the chances that more solutions can be validated in vivo. Moreover, even if the data cannot be directly cleared for publication, there are ways to anonymise results, not disclose the company's name, or build an artificial dataset with similar characteristics.

RQ5. What cost considerations are used for ML SDP in commercial applications?

Cost considerations had the worst relevance evaluation scores out of all six categories. As many as 33 publications had a "0" from both evaluations, eight had one evaluation of "0.5", and only one had the highest score of "2" (Section 3). Therefore, only very little evidence was analysed. Each business implementation requires a certain level of return on investment calculation, and even the most basic project charters include cost–benefit analysis [46]. If the published research helped to understand how many defects can be avoided by implementing SDP solutions and at what cost, it could increase the business adoption of proposed solutions. On the other hand, it is worth acknowledging that it may be difficult to estimate how expensive the development of the tooling, gathering the data, and maintaining such a solution could be. However, in some cases, this decision is more straightforward than in others. For example, if there is an expensive infrastructure needed to test, if finding defects is very difficult or time consuming, or each escaped defect is costly, then the cost of development could be relatively low in comparison. Secondly, if there are existing suitable tools that can be used with only modifications, that significantly reduces expenditure. We have observed that, at least in the reviewed papers, researchers seldom leave an invitation to use their tooling and frameworks by others.

The cost considerations that could be offered in ML SDP research does not need to be very precise or sophisticated. Accurate calculations and decisions must be made within the industry. However, researchers could greatly help to make initial feasibility decisions by providing rough estimates of implementation costs based on the experiences of the author. Even rough estimates in working hours or Full-Time Equivalent (FTE) per month that is needed to set up the tool, practitioners could compare it to the cost of escaped failure in their industry and context. Secondly, the simplest way that we would recommend in a preliminary financial benefit analysis is a Return on Investment (ROI) or Internal Rate of Return (IRR) calculation [46].

We did not find any other research we could compare our conclusions regarding cost considerations in research for ML SDP introduction in business. We hope this could be improved in the future, and we strongly encourage academics to offer this information in their primary and secondary studies. Following the observations on low industry-academia collaboration by Garousi and Felderer [17], we agree that the majority of research lacks cost–benefit analysis, discussing the time and effort needed for implementation and what potential savings it can bring. As businesses need to be cost-effective, we identify it to be a key enabler to bridge the gap and judging by our results, there is significant room for improvement.

RQ6. What learnings come from the commercial applications of ML SDP?

Identification of lessons learnt is essential in process improvement initiatives in many business situations [47]. Unfortunately, the research publications we reviewed were evaluated relatively low in this aspect (Section 3). Naturally, all publications reported on the methods and results, but usually offered very scarce insight into the implementation process itself. In comparison, the evidence on methods, features, frameworks, and datasets was satisfactory. But any specific lessons learnt offered by the authors and practitioners designing and maintaining their solutions are very beneficial to similar future efforts.

From an implementation perspective, the most important aspects to understand could be: how difficult is setting up the tool, what competence is needed (like programming skills in Python, R, and specific libraries, as well as specific predictive modelling techniques), what other tools are needed and what maintenance effort is needed. Without such considerations explained in the publication, the value is diminished to the method described and the results obtained, with limited potential to be repeated in another business context.

In addition, technical implementation difficulties are not the only ones worth discussing. Another important consideration is dealing with the change management issues encountered and overcome during the introduction of a new process. Such insight can make all the difference in industry application. Secondly, an outstanding practise we found (for example, [SLR27]) is to gather feedback from practitioners on using the researched solutions, where even simple input from practitioners on the time, effort and satisfaction of the obtained results can offer meaningful insight.

In the following, we highlighted the key points that were raised in the reviewed papers from the perspective of practical lessons learnt, not only in standard research description but also emphasised as such.

- Dam et al. [SLR11] start the article with a motivating example, where the authors begin with an example which illustrates the difficulties of using the already existing approaches, justifying the need for their research.
- An 'implications for practitioners' section offered by dos Santos and Figueiredo [SLR23] emphasised that practitioners should not expect the same models to explain defect predictions for different projects and recommend within project training.
- Bowes et al. [SLR27] reported training practitioners on their tool and implementing weekly email questionnaires to collect input and enable evaluation and improvement of their solution.
- An important lesson learnt raised by Kang et al. [SLR32] is the need to gather feedback from experts on how much overhead was added by the introduction of the tool.
- The second recommendation by Kang et al. [SLR32] is based on the feedback received from management that provided cost estimates that were beneficial to understanding the effect of cost reduction as it was essential to obtain managerial support for implementation.
- Lastly, the entire publication by Melo et al. [SLR22] is a comprehensive step-wise guideline on change-proneness prediction. The guideline approach is the best form of publication if the authors aim for their proposed method to be tested in vivo.

Addressing the above topics in publications would increase the chances of adoption and make it easier and more successful when it takes place. Therefore, we recommend academics to include a lessons learnt section in their publications, for example done by Kang et al. [SLR32], Zhao et al. [SLR18].

Finally, we would like to highlight what works well, and thus what are the benefits, in terms of progress towards the wider industrial application of ML SDP solutions:

- The first positive aspect is that there are several studies in vivo across many industries published over recent years. Therefore, there seems to be an interest among companies and practitioners in implementing the said solutions (see Fig. 2).
- The researchers are able to obtain good predictive results, despite using a wide plethora of different algorithms. The solutions created so far in academic settings are sufficient enough for researchers to find effective solutions in different contexts (see Section 3).
- Established ML frameworks, like WEKA, Scikit-Learn, Keras, and TensorFlow, are the most popular among our selected papers, showing their usefulness in vivo.
- Proposed ML frameworks can be successfully integrated into the existing testing machinery for automatic SDP for example: ELFF [SLR27], Jaskier [SLR6], DePress [SLR8] or [NM9,NM10]).
- Despite not being frequently explored, the studies that do exist on cost-considerations show high profitability potential [SLR32, SLR8].

ML-based approach to SDP also has several risks that may hamper industrial application. The papers in the scope of our SLR indicated the following (see Section 4 for more details):

- Limited availability and quality of in vivo data that is feasible ML SDP (including replicability aspects).
- In the analysed literature, we have not encountered sufficient consideration of the researcher bias, discussed in detail by Shepperd et al. [9], but there is no evidence as well that this important factor could be neglected in industrial settings.
- Using inadequate performance measurement metrics, for example, not dealing with class-imbalance problem.
- Consideration of only singular test phases of the test cycle.
- Not sufficient explanation and reporting of the actual implementation of the solutions (including scarcely provided lessons learned).
- Lack of a satisfactory explanation of the predictions.
- In the analysed literature, we have not encountered any discussion about processes of new technology introduction and relevant change management.
- Not sufficient consideration of the costs and benefits of the introduced solutions.

4.2. Further recommendations

Together with the invited practitioners from Nokia (Section 4) we have defined the following recommendations based on the main findings described in Section 4.1:

- From the perspective of utilised ML methods, details of model operation seems not to be crucial. The most important factors for industry application are focused on low error rate, interpretability of results, and ability to benchmark different models for reuse purposes. We have not found justification for maximising the accuracy metrics as critically important.
- We have not identified clear need to include a large number of features that minimally improve final performance. A more feasible approach is to use what is already available in the company.
- Tools offering a subset of the below characteristics have a higher chance of being reusable in the industry: online, standalone, non-intrusive, easy to install, comprehensive documentation, user-friendly, defined interfaces to well-known databases, and allowing XAI interpretation.
- When working on new solutions, researchers should reach out to local companies asking them to implement them in vivo. There are chances that some of the proposals can be accepted; even if the data cannot be directly cleared for publishing. Furthermore, there are ways to randomise, not publish specific details, or not disclose the company's name.
- Researchers could immensely increase the chances of their solutions being implemented in vivo if the publications would support the initial feasibility decisions by providing rough estimates of implementation costs based on the authors' experiences. Even rough estimates that allow to understand the effort behind the application is a great addition.
- Including a lesson learnt section in empirical papers offers significant value to any further attempts to replicate the solution in other contexts. Any commentary on the difficulties faced, received feedback, or effort needed could increase the chances of further internalisation of ML SDP in the industry.

Second, based on the observations made during our selection process, introducing a clear statement in the title, abstract, and keywords of the published studies could significantly increase accessibility — a prime example is Arrieta et al. [NM2], where "An Industrial Case Study on Elevators Dispatching Algorithms" is included in the title. It highlights not only validation in a real-world setting, but also circumstance and context.

Third, as the incentive for companies to publish their research in academia is low and the entry threshold of publishing is high, a portion of evidence is available in grey literature. Various sources offer a wide selection of less rigorous reports on the usage of ML SDP in different companies. Following Kitchenham et al. [48] including such publications in future SLR research would be very valuable.

Another vital concern to raise is the issue of reproducible research in the context of in vivo validation [49]. We did not specifically review for reproducibility; however, disclosing company-owned data, processes, or research results can be more complex than in studies done on public sets. Therefore, gradual validation on academic and open-source sets that have no disclosure limitations is critical before the in vivo studies are conducted which may not get full publishing opportunities.

Next, any ML SDP effort in an industrial setting must be considered from a top-down perspective. Starting with a consideration of what the company's mission is, what the business goals are, and how the introduced solution may contribute to the company's success. Therefore, the solution should be robust enough to suit current needs, processes, tools, and data.

Our last recommendation is focused on our research done for Nokia and reflects on the significant potential for software practitioners to benefit from this literature review. In particular, we will use the results of the study and the selected contributions from the analysed literature (works like Rana et al. [50]) in our future effort to address the challenges in improving the quality and minimising the cost of software testing of 5G systems at Nokia as described by Stradowski and Madeyski [4].

4.3. Threats to validity

The threats to validity discussed below, include only the issues that have not been explained in the previous sections (mainly Section 2 and Section 3). Each highlighted risk had a mitigation reference, or confirmation that it has not been addressed. We followed the most common categories of threats from Zhou et al. [51] to disclose possible inadequacies and increase the overall quality of the conducted SLR:

- **Construct validity:** the most crucial threat to the correctness of measures for the researched concepts was the possibly incomplete and incorrect information in the selected primary studies. We do not directly mitigate the threats related to the heterogeneity, publication bias, researcher bias, or generalisability of the primary studies. The industrial settings may vary significantly and usually require a deep contextual and technical understanding of its operations. Therefore, it may also be challenging to synthesise the goals and outcomes precisely and understandably to a wider audience. To limit the impact and increase the applicability of conclusions to our case, we invited an experienced practitioners from Nokia to participate and review the discussion of our results.
- **External validity:** We do not wish to claim our conclusions can be generalised to the whole challenge of business applicability of ML SDP. We did not use all possible means to establish a golden standard, nor did we analyse all potential threats to validity. The goal of our research, translated to study questions, was to understand the methods, features, frameworks, and cost considerations used in published research done in industry settings and to gather lessons learnt. This goal comes from our overarching aim to improve the quality and minimise the cost of software testing of the 5G technology at Nokia [4,5]. Notably, we would like to also highlight a risk resulting from the lack of comparable business-driven studies to benchmark against. Nevertheless, the conclusions we draw from our six RQs apply to any practitioners interested in ML SDP and not only to those in Nokia. The findings come from multiple industries and different quality assurance processes, reflecting the overall state of ML SDP adoptions. Secondly, the results should also be interesting for researchers willing to understand and advance the industry adoption of ML SDP.
- **Internal validity:** it is possible that some relevant papers were not included in our review. The first reason is the efficiency and specificity of the electronic databases we used. Secondly, some relevant research might have been missed due to search string imperfection. Also, our sensitivity of 85.71% is acceptable; however, does not give certainty. Thus, we applied the snowballing technique to limit the impact and decrease the overlooked research pool. However, it is worth pointing out that we failed to identify a well-established way of distinguishing research done in an industrial context (see Section 4.2).
- **Conclusion validity:** We have contacted the authors of the 14 main works highlighted in Section 3 to confirm our interpretations. For the sake of reproducibility, we described our research process in detail in Section 2. Secondly, all gathered data is available Appendix A, and a dedicated SEGRESS checklist in Appendix B. However, a significant conclusion validity threat arises from the researcher bias and manual selection criteria application and manual evaluation of the primary research. To mitigate individual biases, we have performed a cross-check of the assessments by two researchers. Nevertheless, other researchers might obtain a slightly different subset of publications and relevance evaluation scores.

5. Conclusions

Despite a positive attitude towards academics [52], practitioners rarely include academic research on software defect prediction in vivo operations. The reasons behind this are the low number of in vivo research being published (partly due to the low interest of companies to do so), no established way to differentiate industry-based studies for searchability, and the in vivo context is very specific and not easily generalisable to other circumstances. Therefore, we decided to conduct a more extensive SLR study to evaluate the current research in terms of business applicability and suggest opportunities for improvement.

Our research concentrated on analysing and understanding the methods, features, frameworks, cost considerations, and lessons learnt offered in published research performed in real-world commercial settings. Out of 397 publications on machine learning software defect prediction found by automatic search through six online databases, we selected and investigated 32 studies. The research offers ample evidence regarding the methods, features, frameworks, and datasets, showing significant academic advancements in the field. On the other hand, reviewed papers provided very little evidence of the cost-effectiveness of the proposed solutions, and scarce lessons learnt that could help practitioners implement similar approaches in the future. The most important lessons learnt we identified include: gathering feedback from users in a structured manner, comparing several methods and features to identify the most effective combinations, validating proposed solutions on mixed datasets (research, open-source, and industry), and performing return on investment (ROI) calculations.

We hope that our study is a step towards bridging the gap and increasing the availability of real-world examples in published research. Secondly, it serves as a baseline and preparation for the next phase of employing machine learning software defect prediction in Nokia to increase the quality and lower the cost of 5G technology development [4].

CRediT authorship contribution statement

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

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.infsof.2023.107192. Lech Madeyski reports financial support was provided by Polish Ministry of Science and Higher Education. Szymon Stradowski reports a relationship with Nokia Solutions and Networks Oy that includes: Employment and equity or stocks.

Data availability

Research data are available in Supplementary Material: https://doi.org/10.5281/zenodo.7476403.

Acknowledgements

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

Appendix A. Raw results

Original forms are available in Supplementary Material: https://doi.org/10.5281/zenodo.7476403

Appendix B. SEGRESS checklist

See Table B.7.

Table B.7
SEGRESS item checklist.

SEGRESS item checklist			
Topic	Item#	Details	Location
Title	1	Identification of report topic and type as a systematic review.	Title, title page
Abstract	2	Provision of a structured summary of the entire report.	Abstract, page 1
Opening		Introduction of the larger problem, broad context for the work, and importance of the work.	Section 1, page 2
Rationale	3	Rationale for the review in the context of existing knowledge and how it contributes to the larger problem.	Section 1.1, page 2
Objectives	4	Stating the research questions addressed by the review and how they contribute to the larger problem.	Section 1.2, page 3
Eligibility criteria	5	Specify the inclusion and exclusion criteria based on the topic of interest.	Section 2.2, page 8
Information sources	6	Definition of all sources searched to identify studies.	Section 2.1, page 6
Search strategy	7	Provision of the full search strategies for all databases and snowballing.	Section 2.1, page 6
Selection process	8	Description of methods used to decide whether a study met the inclusion criteria.	Section 2.2, page 8
Data collection process	9	Specification of the methods used to collect data from reports.	Section 2.1, page 6
Data items	10	Listing and definition of all outcomes for which data were sought.	Section 2.4, page 10
Study risk of bias assessment	11	Specification of the methods used to assess risk of bias in the included studies.	Section 2.5, page 11
Effect measures	12	Specification of each outcome of the effect measures.	Not used
Analysis and synthesis methods	13	Description of the processes used to decide which studies were eligible for each synthesis.	Section 3, page 12
Reporting bias assessment	14	Presentation of the methods used to assess risk of bias due to missing results in a synthesis.	Section 3.2, page 24
Certainty assessment	15	Presentation of the methods used to assess certainty in the body of evidence.	Section 3.2, page 24
Study selection	16	Description of the results of the search and selection process including near-misses.	Section 2.2, page 8
Study characteristics	17	Citation of each included study and presentation of its characteristics.	Appendix A, page 42
Risk of bias in studies	18	Presentation of the assessments of risk of bias for each included study.	Section 3.2, page 24
Results of individual studies	19	Provision of summary statistics and for each study.	Section 3, page 12
Results of analyses and syntheses	20	Presentation of the results of all statistical analyses and syntheses conducted.	Section 3, page 12
Reporting biases	21	Presentation of the assessments of risk of bias due to missing results.	Section 4.3, page 35
Certainty of evidence	22	Presentation of the assessments of certainty in the body of evidence for each outcome.	Section 4.3, page 35
Discussion	23	Interpretation of the results in the context of other evidence.	Section 4, page 25
Registration and protocol	24	Registration information for the review.	Not registered
Support	25	Description of sources of financial or non-financial support for the review.	Section 5, page 36
Competing interests	26	Declaration of any competing interests of review authors.	Section 5, page 37
Availability of data, code, and other materials	27	Reporting which other deliverables can be publicly found and where.	Appendix A, page 49

References

[1] S. Dhall, A. Chug, Software defect prediction using supervised learning algorithm and unsupervised learning algorithm, IET Conference Publications (2013) 173–179, http://dx.doi.org/10.1049/cp.2013.2313.

[2] D.H. Wolpert, The lack of a priori distinctions between learning algorithms, Neural Comput. 8 (7) (1996) 1341–1390, http://dx.doi.org/10.1162/neco.1996.8.7.1341.

[3] D.H. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.

[4] S. Stradowski, L. Madeyski, Exploring the challenges in software testing of the 5G system at Nokia: A survey, Inf. Softw. Technol. (2022) 107067, http://dx.doi.org/10.1016/j.infsof.2022.107067, URL: https://www.sciencedirect.com/science/article/pii/S0950584922001768.

[5] S. Stradowski, L. Madeyski, Machine learning in software defect prediction: A business-driven systematic mapping study, Inf. Softw. Technol. (2023) 107128, http://dx.doi.org/10.1016/j.infsof.2022.107128, URL: https://www.sciencedirect.com/science/article/pii/S0950584922002373.

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

[7] C. Catal, B. Diri, A systematic review of software fault prediction studies, Expert Syst. Appl. 36 (4) (2009) 7346–7354, http://dx.doi.org/10.1016/j.eswa.2008.10.027.

[8] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Trans. Softw. Eng. 38 (6) (2012) 1276–1304, http://dx.doi.org/10.1109/TSE.2011.103.

[9] M. Shepperd, D. Bowes, T. Hall, Researcher bias: The use of machine learning in software defect prediction, IEEE Trans. Softw. Eng. 40 (6) (2014) 603–616.

[10] R. Malhotra, A systematic review of machine learning techniques for software fault prediction, Appl. Soft Comput. 27 (2015) 504–518, http://dx.doi.org/10.1016/j.asoc.2014.11.023.

[11] R. Wahono, A systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks, Journal of Software Engineering 1 (1) (2015) 1–16.

[12] K.S. Meiliana, H.L.H.S. Warnars, F.L. Gaol, E. Abdurachman, B. Soewito, Software metrics for fault prediction using machine learning approaches: A literature

review with PROMISE repository dataset, in: 2017 IEEE International Conference on Cybernetics and Computational Intelligence, CyberneticsCom, 2017, pp. 19–23, http://dx.doi.org/10.1109/CYBERNETICSCOM.2017.8311708.

[13] 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 Trans. Reliab. 68 (3) (2019) 1189–1212, http://dx.doi.org/10.1109/TR.2019.2892517.

[14] L. Son, N. Pritam, M. Khari, R. Kumar, P. Phuong, T. Pham, Empirical study of software defect prediction: A systematic mapping, Symmetry 11 (2019) 212, http://dx.doi.org/10.3390/sym11020212.

[15] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, A. Abraham, A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools, Eng. Appl. Artif. Intell. 111 (2022) 104773, http://dx.doi.org/10.1016/j.engappai.2022.104773.

[16] M. Lanza, A. Mocci, L. Ponzanelli, The tragedy of defect prediction, prince of empirical software engineering research, IEEE Softw. 33 (6) (2016) 102–105, http://dx.doi.org/10.1109/MS.2016.156.

[17] V. Garousi, M. Felderer, Worlds apart - industrial and Academic Focus Areas in software testing, IEEE Softw. 34 (5) (2017) 38–45.

[18] D. Bowes, T. Hall, J. Petrić, Software defect prediction: Do different classifiers find the same defects? Softw. Qual. J. 26 (2) (2018) 525–552, http://dx.doi.org/10.1007/s11219-016-9353-3.

[19] Z. Li, X.-Y. Jing, X. Zhu, Progress on approaches to software defect prediction, IET Softw. 12 (3) (2018) 161–175, http://dx.doi.org/10.1049/iet-sen.2017.0148.

[20] I. Sarker, Machine learning: Algorithms, real-world applications and research directions, SN Comput. Sci. 2 (160) (2021) http://dx.doi.org/10.1007/s42979-021-00592-x.

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

[22] B.A. Kitchenham, L. Madeyski, D. Budgen, SEGRESS: Software engineering guidelines for reporting secondary studies, IEEE Trans. Softw. Eng. (2023) http://dx.doi.org/10.1109/TSE.2022.3174092.

[23] O. Dieste, A. Griman, N. Juristo, Developing search strategies for detecting relevant experiments, in: First International Symposium on Empirical Software Engineering and Measurement, Vol. 4, ESEM 2007, 2009, pp. 513–539, http://dx.doi.org/10.1007/s10664-008-9091-7,

[24] H. Grodzicka, A. Ziobrowski, Z. Łakomiak, M. Kawa, L. Madeyski, Code smell prediction employing machine learning meets emerging Java language constructs, in: Data-Centric Business and Applications: Towards Software Development, Vol. 4, Springer International Publishing, Cham, 2020, pp. 137–167, http://dx.doi.org/10.1007/978-3-030-34706-2_8.

[25] 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, ACM, New York, NY, USA, 2014, pp. 1–10, http://dx.doi.org/10.1145/2601248.2601268.

[26] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, Inf. Softw. Technol. 53 (6) (2011) 625–637, http://dx.doi.org/10.1016/j.infsof.2010.12.010.

[27] T. Lewowski, L. Madeyski, Code smells detection using artificial intelligence techniques: A business-driven systematic review, in: N. Kryvinska, A. Poniszewska-Marańda (Eds.), Developments in Information & Knowledge Management for Business Applications : Vol. 3, Springer International Publishing, Cham, 2022, pp. 285–319, http://dx.doi.org/10.1007/978-3-030-77916-0_12.

[28] H. Munir, M. Moayyed, K. Petersen, Considering rigor and relevance when evaluating test driven development: A systematic review, Inf. Softw. Technol. 56 (2014) http://dx.doi.org/10.1016/j.infsof.2014.01.002.

[29] T. Dybå, T. Dingsøyr, Strength of evidence in systematic reviews in software engineering, in: ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2008, pp. 178–187, http://dx.doi.org/10.1145/1414004.1414034.

[30] M. Kawalerowicz, L. Madeyski, Jaskier: A supporting software tool for continuous build outcome prediction practice, Advances and Trends in Artificial Intelligence. From Theory to Practice, 2021, pp. 426–438, http://dx.doi.org/10.1007/978-3-030-79463-7_36.

[31] L. Madeyski, M. Majchrzak, Software measurement and defect prediction with depress extensible framework, Found. Comput. Decis. Sci. 39 (4) (2014) 249–270, http://dx.doi.org/10.2478/fcds-2014-0014.

[32] M. Jureczko, L. Madeyski, Towards identifying software project clusters with regard to defect prediction, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE '10, ACM, New York, NY, USA, 2010, pp. 9:1–9:10, http://dx.doi.org/10.1145/1868328.1868342.

[33] M. Li, H. Zhang, R. Wu, Z.-H. Zhou, Sample-based software defect prediction with active and semi-supervised learning, Autom. Softw. Eng. 19 (2012) 201–230.

[34] C. Catal, U. Sevim, B. Diri, Metrics-driven software quality prediction without prior fault data, in: Electronic Engineering and Computing Technology, 60, 2010, pp. 189–199, http://dx.doi.org/10.1007/978-90-481-8776-8_17,

[35] S. Hosseini, B. Turhan, D. Gunarathna, A systematic literature review and meta-analysis on cross project defect prediction, IEEE Trans. Softw. Eng. 45 (2) (2019) 111–147, http://dx.doi.org/10.1109/TSE.2017.2770124.

[36] F. Peters, T. Menzies, A. Marcus, Better cross company defect prediction, in: 2013 10th Working Conference on Mining Software Repositories, MSR, 2013, pp. 409–418, http://dx.doi.org/10.1109/MSR.2013.6624057.

[37] J. Nam, S. Kim, Heterogeneous defect prediction, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, in: ESEC/FSE 2015, ACM, New York, NY, USA, 2015, pp. 508–519, http://dx.doi.org/10.1145/2786805.2786814.

[38] O. Alan, C. Catal, Thresholds based outlier detection approach for mining class outliers: An empirical case study on software measurement datasets, Expert Syst. Appl. 38 (4) (2011) 3440–3445, http://dx.doi.org/10.1016/j.eswa.2010.08.130, URL: https://www.sciencedirect.com/science/article/pii/S0957417410009383.

[39] S. Kim, H. Zhang, R. Wu, L. Gong, Dealing with noise in defect prediction, in: 2011 33rd International Conference on Software Engineering, ICSE, 2011, pp. 481–490, http://dx.doi.org/10.1145/1985793.1985859.

[40] X. Yang, D. Lo, X. Xia, J. Sun, TLEL: A two-layer ensemble learning approach for just-in-time defect prediction, Inf. Softw. Technol. 87 (2017) 206–220, http://dx.doi.org/10.1016/j.infsof.2017.03.007, URL: https://www.sciencedirect.com/science/article/pii/S0950584917302501.

[41] A. Perera, A. Aleti, M. Böhme, B. Turhan, Defect prediction guided search-based software testing, in: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ACM, 2020, pp. 448–460, http://dx.doi.org/10.1145/3324884.3416612.

[42] B. Dhanalaxmi, G. Naidu, K. Anuradha, A rule-based prediction method for defect detection in software system, J. Theor. Appl. Inf. Technol. 95 (2017) 3403–3412.

[43] C. Tantithamthavorn, A.E. Hassan, An experience report on defect modelling in practice: Pitfalls and challenges, in: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, in: ICSE-SEIP '18, ACM, New York, NY, USA, 2018, pp. 286–295, http://dx.doi.org/10.1145/3183519.3183547.

[44] M. Rizwan, A. Nadeem, M.A. Sindhu, Analyses of classifier's performance measures used in software fault prediction studies, IEEE Access 7 (2019) 82764–82775, http://dx.doi.org/10.1109/ACCESS.2019.2923821.

[45] S.K. Meiliana, H.L.H.S. Warnars, F.L. Gaol, E. Abdurachman, B. Soewito, Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset, in: 2017 IEEE International Conference on Cybernetics and Computational Intelligence, 2017, pp. 19–23, http://dx.doi.org/10.1109/CYBERNETICSCOM.2017.8311708.

[46] J.J. Phillips, Chapter 2 - ROI model, in: Return on Investment in Training and Performance Improvement Programs (Second Edition), second ed., in: Improving Human Performance, Butterworth-Heinemann, Boston, 2003, pp. 32–57, URL: https://www.sciencedirect.com/science/article/pii/B9780750676014500055.

[47] T. Pyzdek, The Six Sigma Handbook: a Complete Guide for Green Belts, Black Belts, and Managers at All Levels, McGraw-Hill Companies, 2003, pp. 269–273, http://dx.doi.org/10.1036/0071415963.

[48] B. Kitchenham, L. Madeyski, D. Budgen, How should software engineering secondary studies include grey material? IEEE Trans. Softw. Eng. 49 (2) (2023) 872–882, http://dx.doi.org/10.1109/TSE.2022.3165938.

[49] L. Madeyski, B. Kitchenham, Would wider adoption of reproducible research be beneficial for empirical software engineering research? J. Intell. Fuzzy Systems 32 (2) (2017) 1509–1521, http://dx.doi.org/10.3233/JIFS-169146.

[50] R. Rana, M. Staron, J. Hansson, M. Nilsson, W. Meding, A framework for adoption of machine learning in industry for software defect prediction, in: Proceedings of the 9th International Conference on Software Engineering and Applications - ICSOFT-EA, ICSOFT 2014, SciTePress, INSTICC, 2014, pp. 383–392, http://dx.doi.org/10.5220/0005099303830392.

[51] 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, http://dx.doi.org/10.1109/APSEC.2016.031.

[52] 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, in: ESEC/FSE 2015, ACM, New York, NY, USA, 2015, pp. 415–425, http://dx.doi.org/10.1145/2786805.2786809.

Systematic Literature Review - References

[SLR1] S. Pradhan, V. Nanniyur, P. Vissapragada, On the defect prediction for large scale software systems – From defect density to machine learning, in: 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), 2020, pp. 374–381, http://dx.doi.org/10.1109/QRS51102.2020.00056.

[SLR2] J Briem, J Briem, H Sellik, P Rapoport, G Gousios, M Aniche, OffSide: Learning to identify mistakes in boundary conditions, in: IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), 2020, pp. 203–208, http://dx.doi.org/10.1145/3387940.3391464.

[SLR3] X. Cheng, H. Wang, J. Hua, G. Xu, Y. Sui, DeepWukong: Statically detecting software vulnerabilities using deep graph neural network, ACM Trans. Softw. Eng. Methodol. 30 (3) (2021) http://dx.doi.org/10.1145/3436877.

[SLR4] L. Gomes, R. Torres, M. Côrtes, On the prediction of long-lived bugs: An analysis and comparative study using FLOSS projects, Inf. Softw. Technol. 132 (2020) 106508, http://dx.doi.org/10.1016/j.infsof.2020.106508.

[SLR5] S. Tabassum, L. Minku, D. Feng, G. Cabral, L. Song, An investigation of cross-project learning in online just-in-time software defect prediction, in: ICSE 2020 Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, 2020, pp. 554–565, http://dx.doi.org/10.1145/3377811.3380403.

[SLR6] M. Kawalerowicz, L. Madeyski, Continuous build outcome prediction: A small-n experiment in settings of a real software project, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories, 2021, pp. 412–425, http://dx.doi.org/10.1007/978-3-030-79463-7_35.

[SLR7] D. Bowes, T. Hall, J. Petrić, Software defect prediction: Do different classifiers find the same defects? Softw. Qual. J. 26 (2) (2018) 525–552, http://dx.doi.org/10.1007/s11219-016-9353-3.

[SLR8] J. Hryszko, L. Madeyski, Cost effectiveness of software defect prediction in an industrial project, Found. Comput. Decis. Sci. 43 (1) (2018) 7–35, http://dx.doi.org/10.1515/fcds-2018-0002.

[SLR9] F. Qin, Z. Zheng, Y. Qiao, K. Trivedi, Studying aging-related bug prediction using cross-project models, IEEE Trans. Reliab. PP (2018) 1–20, http://dx.doi.org/10.1109/TR.2018.2864960.

[SLR10] T. Shippey, D. Bowes, T. Hall, Automatically identifying code features for software defect prediction: Using AST N-grams, Inf. Softw. Technol. 106 (2019) 142–160.

[SLR11] H.K. Dam, T. Pham, S.W. Ng, T. Tran, J. Grundy, A. Ghose, T. Kim, C.-J. Kim, Lessons learned from using a deep tree-based model for software defect prediction in practice, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories, MSR, 2019, pp. 46–57.

[SLR12] H. Altinger, S. Herbold, F. Schneemann, J. Grabowski, F. Wotawa, Performance tuning for automotive software fault prediction, in: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering, SANER, 2017, pp. 526–530, http://dx.doi.org/10.1109/SANER.2017.7884667.

[SLR13] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y. Le Traon, M. Harman, The importance of accounting for real-world labelling when predicting software vulnerabilities, in: ESEC/FSE 2019, ACM, New York, NY, USA, 2019, pp. 695–705, http://dx.doi.org/10.1145/3338906.3338941.

[SLR14] X. Du, B. Chen, Y. Li, J. Guo, Y. Zhou, Y. Liu, Y. Jiang, Leopard: Identifying vulnerable code for vulnerability assessment through program metrics, in: Proceedings of the 41st International Conference on Software Engineering, ICSE '19, IEEE Press, 2019, pp. 60–71.

[SLR15] M. Pradel, K. Sen, DeepBugs: A learning approach to name-based bug detection, Proc. ACM Program. Lang. 2 (OOPSLA) (2018).

[SLR16] J. Chen, S. Zhang, X. He, Q. Lin, H. Zhang, D. Hao, Y. Kang, F. Gao, Z. Xu, Y. Dang, D. Zhang, How incidental are the incidents? Characterizing and prioritizing incidents for large-scale online service systems, in: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ASE '20, ACM, New York, NY, USA, 2020, pp. 373–384, http://dx.doi.org/10.1145/3324884.3416624.

[SLR17] R. Chopra, S. Roy, R. Malhotra, Transductive instance transfer learning for cross-language defect prediction, in: 2022 4th Asia Pacific Information Technology Conference, New York, NY, USA, 2022, pp. 176–182, http://dx.doi.org/10.1145/3512353.3512379.

[SLR18] N. Zhao, J. Chen, Z. Wang, X. Peng, G. Wang, Y. Wu, F. Zhou, Z. Feng, X. Nie, W. Zhang, K. Sui, D. Pei, Real-time incident prediction for online service systems, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, New York, NY, USA, 2020, pp. 315–326.

[SLR19] W. Zheng, H. Lu, Y. Zhou, J. Liang, H. Zheng, Y. Deng, IFeedback: Exploiting user feedback for real-time issue detection in large-scale online service systems, in: Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering, ASE '19, IEEE Press, 2019, pp. 352–363, http://dx.doi.org/10.1109/ASE.2019.00041.

[SLR20] H. Li, X. Yang, Y. Li, L.-Y. Hao, T.-L. Zhang, Evolutionary extreme learning machine with sparse cost matrix for imbalanced learning, ISA Trans. 100 (2020) 198–209, http://dx.doi.org/10.1016/j.isatra.2019.11.020.

[SLR21] R. Malhotra, A. Sharma, Analyzing machine learning techniques for fault prediction using web applications, J. Inform. Process. Syst. 14 (2018) 751–770, http://dx.doi.org/10.3745/JIPS.04.0077.

[SLR22] C.S. Melo, M. Cruz, A.D.F. Martins, T. Matos, J.M. da Silva Monteiro Filho, J.C. Machado, A practical guide to support change-proneness prediction, in: International Conference on Enterprise Information Systems, 2019.

[SLR23] G.E. dos Santos, E. Figueiredo, Failure of one, fall of many: An exploratory study of software features for defect prediction, in: 2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation, SCAM, 2020, pp. 98–109, http://dx.doi.org/10.1109/SCAM51674.2020.00016.

[SLR24] T. Yu, W. Wen, X. Han, J. Hayes, ConPredictor: Concurrency defect prediction in real-world applications, in: IEEE Transactions on Software Engineering, IEEE Trans. Softw. Eng. 45 (2018) 558–575, http://dx.doi.org/10.1109/TSE.2018.2791521.

[SLR25] 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, in: ICSEB 2019, ACM, New York, NY, USA, 2019, pp. 60–65, http://dx.doi.org/10.1145/3374549.3374553.

[SLR26] Y. Kim, S. Mun, S. Yoo, M. Kim, Precise learn-to-rank fault localization using dynamic and static features of target programs, ACM Trans. Softw. Eng. Methodol. 28 (2019) 1–34, http://dx.doi.org/10.1145/3345628.

[SLR27] D. Bowes, S. Counsell, T. Hall, J. Petrić, T. Shippey, Getting defect prediction into industrial practice: The ELFF tool, in: 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2017, pp. 44–47, http://dx.doi.org/10.1109/ISSREW.2017.11.

[SLR28] H. Wang, T. Khoshgoftaar, A study on software metric selection for software fault prediction, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1045–1050, http://dx.doi.org/10.1109/ICMLA.2019.00176.

[SLR29] H. Sellik, O. Paridon, G. Gousios, M. Aniche, Learning off-by-one mistakes: An empirical study, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), 2021, pp. 58–67.

[SLR30] 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, http://dx.doi.org/10.1109/ICTAI.2017.00019.

[SLR31] A. Phan, L. Nguyen, Convolutional neural networks on assembly code for predicting software defects, in: 2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES), 2017, pp. 37–42, http://dx.doi.org/10.1109/IESYS.2017.8233558.

[SLR32] J. Kang, D. Ryu, J. Baik, Predicting just-in-time software defects to reduce post-release quality costs in the maritime industry, Softw. - Pract. Exp. 51 (4) (2021) 748–771, http://dx.doi.org/10.1002/spe.2927.

Near-Miss - References

[NM1] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh, P. Runeson, Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts, Empirical Softw. Engg. 21 (4) (2016) 1533–1578, http://dx.doi.org/10.1007/s10664-015-9401-9.

[NM2] A. Arrieta, J. Ayerdi, M. Illarramendi, A. Agirre, G. Sagardui, M. Arratibel, Using machine learning to build test oracles: An industrial case study on elevators dispatching algorithms, in: 2021 IEEE/ACM International Conference on Automation of Software Test, AST, 2021, pp. 30–39, http://dx.doi.org/10.1109/AST52587.2021.00012.

[NM3] A. Bhattacharyya, C. Amza, PReT: A tool for automatic phase-based regression testing, in: 2018 IEEE International Conference on Cloud Computing Technology and Science, CloudCom, IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 284–289, http://dx.doi.org/10.1109/CloudCom2018.2018.00062.

[NM4] H. Khosrowjerdi, K. Meinke, A. Rasmusson, Virtualized-fault injection testing: A machine learning approach, in: 2018 IEEE 11th International Conference on Software Testing, Verification and Validation, ICST, 2018, pp. 297–308, http://dx.doi.org/10.1109/ICST.2018.00037.

[NM5] G. Gadelha, F. Ramalho, T. Massoni, Traceability recovery between bug reports and test cases-a Mozilla Firefox case study, Autom. Softw. Eng. 28 (2021) http://dx.doi.org/10.1007/s10515-021-00287-w.

[NM6] H.-Y. Li, M. Li, Z.-H. Zhou, Towards one reusable model for various software defect mining tasks, in: Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019, Macau, China, 2019, Proceedings, Part III, Springer-Verlag, Berlin, Heidelberg, 2019, pp. 212–224, http://dx.doi.org/10.1007/978-3-030-16142-2_17.

[NM7] D. She, R. Krishna, L. Yan, S. Jana, B. Ray, MTFuzz: Fuzzing with a multi-task neural network, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, 2020, http://dx.doi.org/10.1145/3368089.3409723.

[NM8] D. Marijan, A. Gotlieb, A. Sapkota, Neural network classification for improving continuous regression testing, in: 2020 IEEE International Conference On Artificial Intelligence Testing (AITest), 2020, pp. 123–124, http://dx.doi.org/10.1109/AITEST49225.2020.00025.

[NM9] M. Gokceoglu, H. Sozer, Automated defect prioritization based on defects resolved at various project periods, J. Syst. Softw. 179 (2021) 110993, http://dx.doi.org/10.1016/j.jss.2021.110993.

[NM10] D. Elsner, F. Hauer, A. Pretschner, S. Reimer, Empirically evaluating readily available information for regression test optimization in continuous integration, in: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, ACM, New York, NY, USA, 2021, pp. 491–504.

[NM11] B. Agrawal, M. Mishra, Demo: Automatically retrainable self improving model for the automated classification of software incidents into multiple classes, in: IEEE 41st International Conference on Distributed Computing Systems, ICDCS, 2021, pp. 1110–1113, http://dx.doi.org/10.1109/ICDCS51616.2021.00113.

[NM12] A. Sharif, D. Marijan, M. Liaaen, DeepOrder: Deep learning for test case prioritization in continuous integration testing, 2021, arXiv abs/2110.07443.

[NM13] W. Zhang, Efficient bug triage for industrial environments, in: 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME, 2020, pp. 727–735, http://dx.doi.org/10.1109/ICSME46990.2020.00082.