Information and Software Technology 170 (2024) 107448

Contents lists available at ScienceDirect

Information and Software Technology

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

VALIDATE: A deep dive into vulnerability prediction datasets[☆]

Matteo Esposito [a,b,*], Davide Falessi [a]

[a] University of Rome Tor Vergata, Via del Politecnico, 1, Rome, 00132, Lazio, Italy
[b] Multitel S.r.l., Via Modigliani, 27, Conversano (BA), 70014, Puglia, Italy

ARTICLE INFO

Dataset link: https://validatetool.org/

Keywords:
Security
Replicability
Vulnerability
Machine learning
Repository
Dataset

ABSTRACT

Context: Vulnerabilities are an essential issue today, as they cause economic damage to the industry and endanger our daily life by threatening critical national security infrastructures. Vulnerability prediction supports software engineers in preventing the use of vulnerabilities by malicious attackers, thus improving the security and reliability of software. Datasets are vital to vulnerability prediction studies, as machine learning models require a dataset. Dataset creation is time-consuming, error-prone, and difficult to validate.
Objectives: This study aims to characterise the datasets of prediction studies in terms of availability and features. Moreover, to support researchers in finding and sharing datasets, we provide the first VulnerAbiLty predIction DatAseT rEpository (*VALIDATE*).
Methods: We perform a systematic literature review of the datasets of vulnerability prediction studies.
Results: Our results show that out of 50 primary studies, only 22 studies (i.e., 38%) provide a reachable dataset. Of these 22 studies, only one study provides a dataset in a stable repository.
Conclusions: Our repository of 31 datasets, 22 reachable plus nine datasets provided by authors via email, supports researchers in finding datasets of interest, hence avoiding reinventing the wheel; this translates into less effort, more reliability, and more reproducibility in dataset creation and use.

1. Introduction

Our daily life relies on multiple instances of software that perform everyday tasks [1] such as household [2], autonomous driving [3] and digital governance [4]. Vulnerabilities are an issue today as they are widespread across multiple domains. Vulnerabilities cause economic damage to industry [5,6]; they endanger our daily lives by threatening critical security infrastructures [7–9].

The more software spreads across new domains, the wider the attack surfaces; thus, the need to focus on vulnerability prevention [10]. The spread of cyber-warfare increases the importance of preventing vulnerabilities [11]. Vulnerability prediction helps software engineers prevent the use of vulnerabilities by malicious attackers, thus improving the security and reliability of software [12]. Vulnerability prediction studies (**VPS**) support advances in the identification of vulnerabilities in terms of various performance aspects such as cost and accuracy [13,14].

Datasets are vital to VPS as every Machine Learning (**ML**) model requires a dataset. The dataset creation is time-consuming, error-prone and difficult to validate [15]. Datasets are also vital to support replication; many recent studies stressed the issue of replicability and reproducibility in ML studies [16–20].

This study aims to characterise VPS datasets, briefly datasets, in terms of availability and features. Furthermore, to help researchers find and share datasets, we provide the first VulnerAbiLity predIction DatAseT rEpository (**VALIDATE**). Our contributions are as follows:

- providing the first systematic dataset review (**SDR**) related to VPS. Our SDR supports researchers in finding datasets of interest, hence avoiding reinventing the wheel, i.e., creating a dataset when a similar one is available. Reusing datasets of interest results in less effort, more reliability, and more reproducibility in VPS. Although there are many datasets, no dataset might meet the researchers' needs; therefore, researchers can use our characterisation to provide evidence of the need to create a new dataset,
- providing VALIDATE, an online tool to search for a dataset with specific features. VALIDATE supports our characterisation by allowing the users to navigate and retrieve datasets easily, and
- providing a non-exact replica of a VPS. The aim of our non-exact replica is twofold: (i) it shows how researchers can use VALIDATE, and (ii) it improves the body of knowledge of vulnerability prediction.

☆ **Open access funding** provided by Università degli Studi di Roma Tor Vergata within the CRUI-CARE Agreement.
* Corresponding author at: University of Rome Tor Vergata, Via del Politecnico, 1, Rome, 00132, Lazio, Italy.
E-mail addresses: m.esposito@ing.uniroma2.it (M. Esposito), falessi@ing.uniroma2.it (D. Falessi).

https://doi.org/10.1016/j.infsof.2024.107448
Received 20 November 2023; Received in revised form 12 March 2024; Accepted 14 March 2024
Available online 19 March 2024

VALIDATE is available on GitHub Pages.[1] We are also open to community contributions. Researchers may use the Dataset Submission Template located in the VALIDATE Community repository[2] to submit new datasets. Moreover, we provide a brief user guide, a YouTube video tutorial link[3] and all the original studies references in the Replication Package [21].

We performed an SDR which differs significantly in aim from the conventional SLRs. SLR focuses on reviewing and synthesising previous studies results [22], whereas our SDR focuses on analysing the datasets used in previous studies. Our SDR provides a scientific and practical contribution, showing gaps and availability of datasets types.

In our context, we have a one-to-one relation between a study and its dataset (if provided); i.e., if a study provided more than one dataset, these have been merged by the original authors into one repository. Our results show that of the 50 primary studies, only 22 studies (i.e., 38%) provide a reachable dataset; only one study provides a dataset in a stable repository. Out of the 31 gathered datasets, 22 reachable plus nine datasets provided by authors via email, no datasets have been manually checked for the absence of any vulnerability in software entities. Specifically, researchers labelled entities as negative when they do not have a specific vulnerability; however, the entity might have other vulnerabilities. Thus, negative labels might actually be positive. Therefore, VPS might have overestimated the false positive rate [23].

The remainder of this paper is structured as follows. We present the research background and related work in Section 2. We describe the study design in Section 3.3. In Section 4, we discuss the SDR results. In Section 5, we present our non-exact replica of the selected study, imitating what researchers can do with our tool. We discuss the SDR results in Section 6. We explain the threats to the validity of our study in Section 7. We provide our conclusions in Section 8.

2. Background and related work

This section introduces ML for Software Engineering (**SE**) and discusses related works in VPS, emphasising reproducibility.

ML can support SE in many tasks [24]. Zhang and Tsai [25] defined the field of SE as a “fertile ground where many software development and maintenance tasks could be formulated as learning problems and approached in terms of learning algorithms”. ML can tackle problems that humans usually have a mere grasp or no knowledge at all [25] or optimise solutions to otherwise well-known problems but with inefficient solutions [26,27]. Researchers focus on systematically reviewing the tasks of SE that benefit from the ML “unparalleled capabilities” [24, 28–30]. For instance, Lyu et al. [31] discuss the development process that benefited from emerging AIOps models. Kapur and Sodhi [32] discuss the effort estimation based on metrics such as software features similarity and developer activity. Durelli et al. [33] conduct a mapping study on ML applications for software testing.

The activity of mining software repositories (**MSR**) led to ideas and challenges for empirical studies in SE [34–36]. Hassan et al. [37] point out the effectiveness of MSR in empirically validating new ideas and techniques. MSR activity supports the creation of datasets containing useful information for predictive models [38–42]. Several studies describe tools, techniques, and advantages in automatic mining [43–45]. Bavota [46] presents issues in mining software repositories, including the lack of meaningful content in commit messages [47], misclassification of data [48,49] and the missing link between ticket and commit [50]. Zhou et al. [51] investigate on how automatically identify security patches through commit-related data; Zou et al. [52] uses ML to protect the return pointer. Finally, Vandehei et al. [53], Falessi et al. [54] highlight the importance, in the data preparation stage, of correctly labelling the data according to the ticketing system and the information from the version control system (**VCS**) platform.

Finally, Ibrahim et al. [55] discuss the significant threat posed by software vulnerabilities in opensource projects, which can compromise system integrity, availability, and confidentiality. Their analysis of the top 100 PHP opensource projects reveals that 27% of them exhibit security vulnerabilities. Increasing cyber-warfare [11] and data breaches [56] threaten critical infrastructure and user privacy [7–9]. Furthermore, our recent study [57] reveals that vulnerability default severity might be inaccurate.

2.1. Machine learning for vulnerability prediction studies

ML for VPS led to several studies that approached challenges and novel ideas in the field [58,58–63]. Croft et al. [64] conduct an SLR in the data preparation phase of a VPS study. They show that data is the crucial component of any data-driven application; nevertheless, the preparation phase of a dataset is still full of challenges. Our study relies on Croft et al. [64] study selection as all datasets require the data preparation phase. We focus on the entire dataset rather than only on the data preparation phase. Thus, our selection of studies extends from Croft et al. [64]. We conduct a new SLR adopting Croft et al. [64] search strings and inclusion and exclusion criteria. We expand the search scope by adding two specific keywords, “dataset” and “repository”, and by broadening the time range, including studies published until January 2023. Finally, it is essential to emphasise that although Croft et al. [64] focuses on the data preparation phase of VPS, we focus on characterising existing VPS datasets across 9 dimensions.

Jabeen et al. [65] experimented on the effectiveness of different ML and statistical techniques for software vulnerability prediction. The authors use goodness-of-fit and criteria on prediction capabilities to assess the performances. Jabeen et al. [65] show that ML techniques are more effective than statistical ones.

Zheng et al. [66] investigate the factors that can affect the vulnerability detection capabilities of ML models. Zheng et al. [66] use the CountVectorizer[4] to extract features from text, to improve the performance of conventional ML models. The authors show that deep learning models can perform better than traditional ML models. Zhu et al. [67] highlight a “perception gap” between deep learning and human experts in understanding code semantics. In real-world scenarios, deep learning-based methods underperform by over 50% compared to controlled experiments, prompting a deep dive into this phenomenon and exploring current solutions to narrow the gap.

Partenza et al. [68] assess the capabilities of a periodic neural network called ASTNN. To experiment with ASTNN the authors used the Juliet test case suite outperforming the authors’ previous research on project Achilles. However, Partenza et al. [68] show that the same neural network performance dropped when tested against OWASP real-world vulnerabilities. The author’s research highlights that ad hoc datasets, like Juliet, are unsuitable for ML training due to their asymmetries in the complexity of vulnerable and non-vulnerable codes and unconfirmed cases.

Yu et al. [69] discusses the application of active learning for vulnerability identification, introducing HARMLESS, an incremental support vector machine that achieves high recall by inspecting a small portion of source code files. Despite known challenges such as the high computational and annotation costs for new instances, which may reduce the anticipated advantages in human-effort cost reduction, HARMLESS demonstrates effective vulnerability detection.

Jabeen et al. [65], Zheng et al. [66] and Partenza et al. [68] highlight the impacts of ML techniques on datasets confirming the fundamental idea of VALIDATE, i.e., researchers should focus on ML or statistical techniques aimed at improving previous results rather than reinventing the wheel (i.e., creating datasets). More specifically, Partenza

[1] https://validatetool.org/
[2] https://go.validatetool.org/template
[3] https://go.validatetool.org/tutorial

[4] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

et al. [68] reports problems using ad hoc datasets such as Juliet to train specific ML algorithms. Our SDR reveals that working in the VPS field frequently employs ad-hoc datasets for their ML training and validation phases. Researchers may use VALIDATE to address this problem to identify datasets that align with their innovative ideas and research goals.

2.2. Replicability and reproducibility

Giray [30] analyses state of the art and challenges in the engineering of ML systems. They show that the random nature of ML-based systems hinders SE tasks in engineering; moreover, they discuss the lack of tools and a well-proven methodology for engineering processes. The replicability and reproducibility (**R&R**), are among the challenges in engineering ML systems. On a similar topic, Liu et al. [17] focus on R&R in Deep Learning for SE. They show that 10% of the studies have at least one research question about R&R; 62% of the studies do not provide a good source code or original data. The PROMISE repository [70] is the first repository of freely available datasets for SE. As with VALIDATE, with the PROMISE repository, the authors want to support researchers in new ideas rather than spending effort ”reinventing the wheel”, i.e., creating a new dataset that might already exist. PROMISE has been online since 2005 and supports all fields of SE. Based on the original PROMISE idea and the need for reproducibility highlighted in Liu et al. [17], we developed VALIDATE to serve a specific SE domain, i.e., VPS. It should be noted that PROMISE exists in three distinct versions.[5],[6],[7] At time of writing, there is only one operative version of PROMISE[7], although the oldest among the three. The advantages of VALIDATE compared to PROMISE are:

- VALIDATE dataset repository includes studies related to each dataset: Cheikhi and Abran [71] comprehensively overview the PROMISE and ISBSG dataset repositories. They reveal a lack of studies associated with specific PROMISE datasets; they note that 70 of 84 PROMISE datasets need more practical usage information.
- VALIDATE allows users to search for datasets based on attributes to improve PROMISE on this aspect: Cheikhi and Abran [71] show that only 37 of 84 PROMISE datasets comprehensively describe data and attributes of the dataset. Of course, this search is possible since VALIDATE is specific to a subdomain of SE.
- Public donated dataset peer review: VALIDATE allows the community to publicly donate VPS datasets by opening an issue report[8] and filling in the necessary filter values. While PROMISE allows users to donate their dataset by email, using GitHub issues allows a public peer review of the candidate dataset.
- Community contributions: VALIDATE users can submit an issue requesting a change or fix a bug in VALIDATE.

PROMISE has the advantage, compared to VALIDATE, of providing the only means to share the repository about all SE subdomains.

2.3. The impact of datasets and feature selection on the accuracy of ML

To our knowledge, no previous study has investigated optimal dataset selection. Dataset selection is essential in machine learning-related tasks [66]. The dataset characteristics and the representativeness and the relevance of the data can impact the feature selection (FS) [72,73], the model accuracy and the generalisation capabilities [64,74]. In this sub-section, we present related work showcasing the importance of the dataset characteristics and selection process in the overall ML/DL application.

Alelyani et al. [72] show how dataset characteristics impact the stability of FS algorithms. The authors extensively analysed the inner properties of different datasets to assess how those affect the consistency of the FS algorithm. Furthermore, Oreski et al. [73] focused on seven characterisation methodologies for datasets and five FS techniques. The author reveals how specific characteristics of a dataset deeply impact the accuracy and the time complexity of FS techniques. Thus, the characteristics of datasets are essential in dimensionality reduction, hence critically influencing the accuracy and generalisation capabilities of predictive models.

Zheng et al. [66] investigate four factors influencing machine learning-based vulnerability detection, including data quality and classification models. As a result, selecting appropriate datasets impacts the performance of a classification or neural network. Likewise, [64,66,74] focusing on the data preparation phase of vulnerability prediction studies, emphasise the importance of carefully selecting appropriate datasets for specific research ideas. Nong et al. [74] address the shortage of systematic research on open science practices within software engineering, mainly focusing on deep learning-based software vulnerability detection. The authors performed a comprehensive literature review on 55 original studies, underscoring the importance of meticulous dataset selection. Their investigation reveals that the utilisation of imbalanced or artificially generated datasets resulted in overly optimistic performance assessments, thereby compromising the replicability of most techniques.

Finally, regarding the importance of FS algorithm, Zhang et al. [75] conducted the first extensive empirical study on the correlation between various application features and vulnerability proliferation. Seven FS techinques were applied to nine feature subsets selected from 34 collected features, this allowed the authors to discover that application complexity alone is not the sole determinant of vulnerability discovery. More specifically, human-related factors also significantly explain the proliferation of vulnerabilities.

3. Systematic dataset review

We perform the first SDR in the field of VPS [76,77]. We build our methodology on findings and observations of Croft et al. [64], Nong et al. [74].

3.1. Goal and research questions

We formalised the goal of this study according to the Goal Question Metric (GQM) approach [78] as follows:

Investigate	datasets,
for the purpose of	characterisation,
with respect to	nine dimensions,
from the point of view of	researchers,
in the context of	VPS.

Based on the aforementioned goal, we defined our main research question, which serves as the primary focus of our investigation: *What are the defining characteristics of state-of-the-art datasets in VPS?*”

3.2. Research methodology

The SDR consist of two phases. Phase 1 gathers studies using PICO [79]. Phase 2 focuses on filtering the studies according to inclusion and exclusion criteria.

Phase 1 involves the collection of 2802 studies spread across three different sources. Specifically:

- ACM Digital Library: 69 studies, 67 of which coincide with Croft et al. [64],

[5] http://promise.site.uottawa.ca/SERepository/
[6] http://code.google.com/p/promisedata/
[7] http://openscience.us/repo
[8] https://go.validatetool.org/community

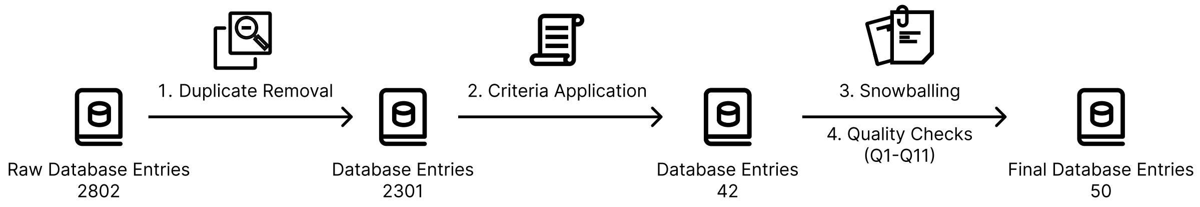


Fig. 1. Workflow for study selection and inclusion.

- IEEEXplorer: 2360 studies, 1118 of which coincide with Croft et al. [64], and
- Scopus: 373 studies, 357 of which coincide with Croft et al. [64].

We use an SLR procedure based on Zhang et al. [76],Kitchenham [77], Lewowski and Madeyski [80]. Moreover, our primary study selection is inspired by Croft et al. [64]. Specifically, Croft et al. [64] performed a study selection process in February 2021 and obtained 61 studies. Table 1 describes our PICO enquiry. Croft et al. [64] provides the base of our query to which we add to the population Category the word "repository" and "dataset" to expand and accommodate the different scopes of the literature review. Croft et al. [64] limited their research to studies published until February 2021. Our query extends the selection of studies of Croft et al. [64] to those published up to January 2023.

We derive our groups and dimensions from studies highlighting aspects of ML for VPS. For instance, Falessi et al. [81] point out the importance of preserving the mined data time order. Similarly, several studies approached VPS with different grains [82–87]; thus, we decided to define a dimension on the predicted entity granularity. Zhang et al. [88] combined software metrics and text features for VPS. We acknowledge this idea of combination, and therefore we decide to focus our attention on different feature sets [89–92] by defining a dimension that analyses the available feature sets of the datasets. Russo et al. [93] point out the relevance of CVE in VPS [94–97]; therefore, we focus our attention on the availability of CVE info in the datasets.

Fig. 1 presents our steps in the selection process of studies, i.e., Phase 2. We divide phase 2 into three specific steps:

1. Duplicate Removal: we eliminate duplicates in the three distinct data sources. This resulted in the discarding of 501 studies.
2. Criteria Application: we apply inclusion and exclusion criteria to the remaining 2301 studies, eliminating an additional 2251 studies. Croft et al. [64] proposed a set of criteria in their systematic literature review (SLR). We find Croft et al. [64] inclusion and exclusion criteria to match our aim, so we decided to avoid reinventing the wheel. Table 2 presents our inclusion and exclusion criteria inspired from [64].
3. Snowball Sampling: we use the snowball sampling technique [98, 99] to incorporate potentially related studies that the query may not have captured. This results in the identification of 8 additional studies.
4. Quality Assessment: Table 3 presents the quality checklist inspired by the established guidelines by Kitchenham [77]. According to Table 3, we assessed whether the original studies' authors clearly stated the design's aims correctly aligned with them. Furthermore, we evaluated if the metrics used in the measurement procedures of the original studies were appropriate for answering the research questions and if the sample represented the specific vulnerability type and granularity. We check whether the authors justified smaller sample sizes and if they thoroughly described the specific tools used in the study. At the end of the fourth step in Fig. 1, all the papers selected satisfied questions one to 12. Finally, Fig. 2 graphically presents the last quality checks, i.e., Q12 and Q13 (see Section 3.3.1).

The final selection of the studies in this paper consists of 50 studies, including 171 unique projects and seven programming languages.

Table 1
PICO query string.

Category	Subject	Search Terms
Population	Software	"software" OR "code" OR "repository" OR "dataset"
Intervention	Machine Learning Static Application	"learn" OR "neuralnetwork" OR "artificial intelligence" OR "AI-based" OR "predict" NOT("fuzz" OR "test" OR "attack" OR "adversarial" OR "malware" OR "description")
Comparison	–	–
Outcomes	Software Vulnerability Prediction	"vulnerability" AND ("predict" OR "detect" OR "classify" O "identify" OR "discover" OR "uncover" OR"locate")

Table 2
Inclusion and exclusion criterias.

Inclusion Criteria	
I1.	The study relates to the field of VPS, and informs the practice of Software Engineering
I2.	The study presents a unique VPS process or evaluation.
I3.	The study is a full paper longer than six pages.
Exclusion Criteria	
E1.	Solely a literature review or survey article.
E2.	Non peer-reviewed academic literature.
E3.	Academic articles other than conference or journal papers, such as book chapters or dissertations.
E4.	Studies not written in English.
E5.	Studies whose full-text is unavailable.
E6.	Studies published to a venue unrelated to the discipline of Computer Science.
E7.	Studies published to a journal or conference with a CORE ranking of less than A and H-index less than 40, and that have a citation count of less than 20.

3.3. Coding

This study systematically characterises datasets by categorising them into 9 dimensions through thematic synthesis [100]. Each of the abovementioned dimensions represents a specific characteristic of the dataset.

3.3.1. Dataset availability

Given the increasing importance of reproducibility and replicability in SE studies, [16–20], and since the dataset is vital to replicate VPS, in this dimension, we characterise whether VPS provides a dataset. In addition, we want to understand the stability of the storage location used for the dataset. We used Cruzes and Dybå [100] recommendations for deriving these dimensions and their values. Our dimensions aim to support researchers in finding a dataset of interest. The existence of possible overlaps or relations across dimensions does not impact the aim of the dimensions. To assess the availability of the datasets, we read the studies and manually checked for any link or reference to the dataset. Eventually, once we found a reference to a dataset in the form of an URL, we manually followed the reference and downloaded the referenced dataset. We note that the type of storage location of datasets

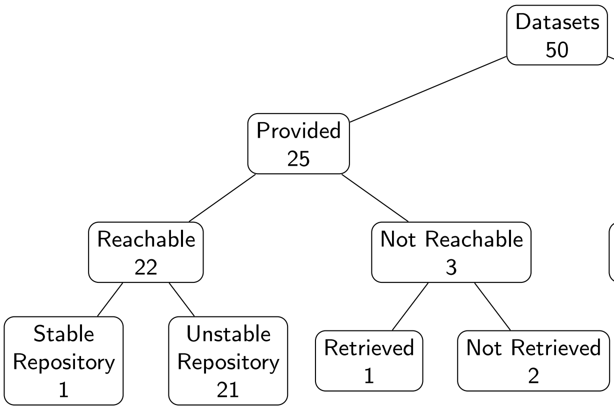


Fig. 2. Availability of datasets.

Table 3
Quality checklist.

Design	
Q1	Are the aims clearly stated?
Q2	Was the study designed with these questions in mind?
Q3	Do the study measures allow the questions to be answered?
Q4	Is the sample representative of the population to which the results will generalise?
Q5	Was the sample size justified?
Q6	If the study involves technology assessment, is the technology clearly defined?
Q7	Are the measures used in the study fully defined?
Q8	Are the measures used in the study the most relevant for answering the research questions?
Q9	Is the study's scope (size and length) sufficient to identify changes in the outcomes of interest?
Analysis	
Q10	Were the basic data adequately described?
Conclusions	
Q11	Are all study questions answered?
Availability of Datasets	
Q12	Was the replication package provided in the paper?
Q13	Is the replication package available?

Table 4
Interpretation of κ values for measuring agreement.

Value of κ	Interpretation
$\kappa < 0$	No agreement
$0 \leq \kappa < 0.4$	Poor agreement
$0.4 \leq \kappa < 0.6$	Discrete agreement
$0.6 \leq \kappa < 0.8$	Good agreement
$0.8 \leq \kappa < 1$	Excellent agreement

is an important aspect. We classify the location as *Stable Repository* if the authors provide an external reference to the dataset, the dataset can be downloaded, and the dataset is in a remote stable repository, e.g. Zenodo. We classify the location as *Unstable Repository* if the authors provide an external reference to the dataset, the dataset can be downloaded, and the dataset is in a remote unstable repository, for example, GitHub or personal websites. It is important to discriminate between stable and unstable locations because datasets in unstable locations can be deleted or permanently moved. Hence, a study might not be replicable if its dataset is stored in an unstable repository. If the dataset was unavailable or unavailable, we emailed all authors requesting such an unavailable dataset. We waited for the answer for three months and thanked all authors who could provide the dataset. All dimensions are based on datasets available in their studies or retrieved via emails; we call these datasets the gathered datasets (**GD**).

The two authors of the paper acted as coders. We used Microsoft Forms to input the data and then outputted a spreadsheet; this supported smooth individual coding. We used the outputted spreadsheet to compare the answers provided by each coder. In case of disagreement, we discussed the matter thoroughly to understand the reasons and then reached a consensus. We analyse our agreements via Cohen's κ [101]. When assigning items to different categories, the kappa statistic is commonly used to evaluate the agreement between raters or classifiers. In the context of VPS datasets, it can be used to measure the level of agreement between two independent authors who categorised the datasets according to different characteristics. When comparing the observed level of agreement to the expected level of agreement, Cohen's κ provides a metric for the reliability and consistency of the categorisations. Both authors characterised the dimensions individually. In the event of disagreements, we reached a consensus through discussion. Table 4 presents the interpretation κ as suggested by Cohen [101], Sim and Wright [102].

In the following subsections, we present the 9 dimensions along which we characterise the datasets. We note that all of them are relevant to vulnerability prediction, whereas some are only for VPS, e.g., CVE/CWE information. VALIDATE supports researchers in finding the dataset of interest, and it is independent of the specific decision-making approach (see Section 2.3).

3.3.2. Dimension 1: Granularity of the labelled entity

Different VPS can focus on entities of different granularity such as classes and files [103], commits [104], methods [105], code fragments [106], or machine code [107]. Understanding the granularity is important since Morrison et al. [15], analysing Microsoft products, reports that the granularity of the predicted entity impacts the accuracy and actionability of the prediction model. Le et al. [104] and Chakraborty et al. [108] focus on comparing the granularity of the predicted entity. This dimension aims to characterise the trends in the granularity of predicted entities. Our methodology consists in downloading the dataset and manually inspecting it. We also checked the granularity is defined in the study.

3.3.3. Dimension 2: Nature of the labelled entity

The data in the dataset can be collected from open-source projects hosted online through VCS like GitHub or the data can be synthetically created by an authority like NIST [109] with the SARD [110]. Understanding the domain of the data is important since Chakraborty et al. [108] discuss the domain of the labelled entities and their impact on the model's performance. Specifically, synthetic entities might not fully grasp the complexity of real-world projects, thus harming the prediction model's ability to generalise. This dimension aims to characterise the trends in data domain; i.e., synthetic versus open-source. Our methodology consists of finding details of the source of the collected data.

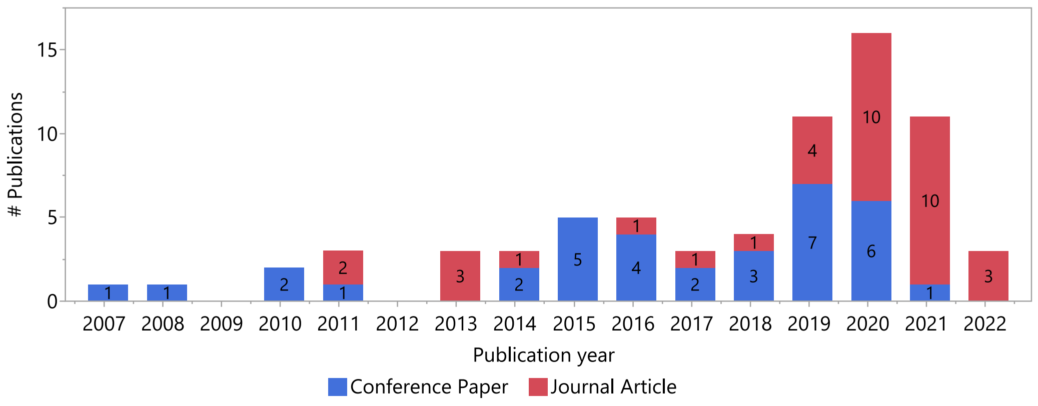


Fig. 3. Publication and venue distribution over the years.

3.3.4. *Dimension 3: Type of labelling*

Multiclass VPS aim to determine which code is affected by a vulnerability and the type of predicted vulnerability. For instance, Zou et al. [111] proposed and evaluated the first deep learning-based system for multiclass vulnerability prediction. This dimension aims to characterise the trends in the type of prediction: binary or multiclass. Our methodology consists in looking for details about the class type of the label.

3.3.5. *Dimension 4: Feature sets availability*

The features are the input to an ML model [112,113]. Different features aim to achieve different goals [114]. In SE, we can measure many aspects of a software entity, such as software or a development process. Chidamber and Kemerer [115] introduce object-oriented software metrics. Moser et al. [116] introduce software development metrics also known as change metrics that aim to measure the process. Recent studies focused on the interpretation and use of specific metrics for VPS [117] and their possible subsets [118]. This dimension characterises the set features in the dataset. Our methodology consists of analysing the name of the features in the dataset and their definition in the correlated study.

3.3.6. *Dimension 5: Availability of either a CVE or a CWE information*

Mann and Christey [119] introduced the concept of Common Vulnerability Enumeration (CVE), also known as Common Vulnerabilities and Exposures. Martin et al. [120] introduced the concept of Common Weakness Enumeration (CWE), a dictionary of publicly known vulnerabilities and security hotspots maintained by The MITRE Corporation [121]. The CWE aims to classify a specific vulnerability that belongs to the release of a project maintained by a specific vendor with a unique identification number. CVE is fundamental for new vulnerability studies [122]. Therefore, the research community, following this new trend, made available large-scale datasets of CVE Infos [91,123–125] and tools to automatically retrieve CVE [126]. This dimension characterises the trend in the availability of CVE information. Our methodology involves looking for details about the availability of CVE info in the collected data.

3.3.7. *Dimension 6: Labelling process*

Labelling the data is crucial for VPS [64]. Due to automatism or not considering real-world scenarios, incorrect labelling can lead to model problems [127]. For example, Tantithamthavorn et al. [49], Eilers et al. [128], Falessi et al. [54] underlined the impact of mislabelled data on the performance of ML model. Therefore, the researchers developed tools and techniques [129,130] to cure software repositories and made manually curated data repositories available for SVP [91,129–131]. This dimension aims to investigate whether the authors manually curated the dataset. A dataset is manually curated if researchers checked at least one aspect during the labelling process; the next dimension focuses on the specific curated aspect. Our methodology consists in looking for how the data have been labelled.

3.3.8. *Dimension 7: Manually curated*

The labelling process may contain automatic steps. For example, a researcher might decide to check if a bug ticket is a bug or a feature [128] or can check which commit actually fixed the ticket or which line of code induced a vulnerability [132]; after this check, a set of automated steps leads to a dataset. This dimension aims to understand what has been manually curated in a manually curated dataset. Our methodology seeks details about the labelling phase of the collected data.

3.3.9. *Dimension 8: Ground truth source*

Ground truth (**GT**) source dimension characterise where authors gained their GT. The GT is an essential component of VPS, as it provides the basis for evaluating the effectiveness of different techniques. GT can suffer from issues related to data availability and quality [133]. Researchers often gain vulnerability data from public databases such as NVD [134–138]. These databases may not be comprehensive, as not all vulnerabilities are reported or discovered. The data quality can also vary, as the information may not be standardised or consistent [139]. Our methodology seeks details about the GT source of the collected data.

3.3.10. *Dimension 9: Negatives assessed*

As already said, mislabelling can impact the performance of prediction models [49,54,128]. There are two possible types of mislabelling: false positive or false negative. In the false positive case, a non-vulnerable code is labelled as vulnerable; in the false negative case, a vulnerable code is labelled as non-vulnerable. In the absence of evidence, we must assume that both mislabelling types impact prediction models' performance. This dimension aims to investigate whether researchers manually assessed both vulnerable and nonvulnerable code or if researchers assessed only the vulnerable code and labelled the remaining code as non-vulnerable. Our methodology consists of carefully checking if the study's authors manually checked if the code labelled as negative has no vulnerability.

4. Results

In this section, we discuss the results of the characterisation of our groups. However, before analysing the results, we analyse the level of agreement of the authors in characterising the dataset over the 9 dimensions. Regarding the author's agreement on each dimension: While we fully agreed (i.e., 100%) on over 80% of the dimensions, it is important to note instances of divergence. Notably, we found a variance in opinions, with an 80% agreement on Dimension 1 (Granularity of the labelled entity) and a 93% agreement on Dimension 4 (Feature sets availability).

4.1. Dataset availability

In this group, we assess whether primary studies provide a means of reproducing the results. The values of the current dimension are:

- **Provided**: the authors provide an external reference to the dataset.
- **Not Provided**: the authors do not provide an external reference to the dataset.
- **Reachable**: the authors provide an external reference to the dataset, and the dataset can be downloaded.
- **Not Reachable**: the authors provide an external reference to the dataset, and the dataset cannot be downloaded, i.e., the provided reference is broken.
- **Retrieved**: the authors do not provide an external reference to the dataset, and the authors provide the dataset to us after an e-mail request.
- **Not Retrieved**: (the authors do not provide the dataset to us after an email request) AND (the authors do not provide an external reference to the dataset) OR [(the dataset was provided) AND (it was unreachable)].
- **Stable Repository**: the authors provide an external reference to the dataset, the dataset can be downloaded and the dataset is in a remote stable repository, e.g., Zenodo [140].
- **Unstable Repository**: the authors provide an external reference to the dataset, the dataset can be downloaded, and the dataset is in a remote unstable repository, e.g., GitHub [141] without DOI commits [142].

Fig. 2, reports the availability of the datasets in our 50 primary studies according to the options mentioned above. According to Fig. 2 only 25 of the 50 studies provide their dataset but three of these are currently not reachable. Thus, only 44% of the studies provide a reachable dataset. Finally, 21 studies are currently hosted in an unstable repository; thus, only one dataset is hosted in a stable repository (i.e. Zenodo).

To obtain datasets that were not publicly available, we contacted the corresponding author of each respective dataset. If no corresponding author was specified, we contacted all authors. Three months were given to the authors to provide us with the datasets. After the three months had elapsed, we expressed our gratitude to the authors who had provided us with their datasets. We also sent a follow-up email to the authors who had not replied to the initial email, giving them an additional week to respond. After a total period of three months and one week, we began the process of classifying the datasets.

Fig. 3 presents the year of publication distribution of the 50 GDs published between 2007 and 2022 and their publication venue distribution. Notably, there is a spike in publications during 2020, with 16 out of the 50 datasets published during that year. Moreover we note a gap in 2009 and 2012. It is noteworthy that, despite conferences being the most preferred venue in the early 2010s, there has been a shift in the community's interest towards publishing in journals in the following years. In particular, in 2022, all of the studies that met our criteria for inclusion came from journal venues.

In conclusion, in this work, according to Table 3 we gathered a total of 31 datasets (GDs): 22 Reachable, 1 Not Reachable Retrieved, and 8 Not Provided Retrieved, which resulted in 21 unique datasets.

4.2. Dimensions

In this subsection, we characterise the 31 GDs along the 9 dimensions.

4.2.1. Dimension 1: Granularity of the labelled entity

In this dimension, we characterise the types of entities involved in the prediction. Studies can predict the following types of entity, ordered from the coarsest to the finest-grained.

- **Class or File**: in Object Oriented Programming (OOP) [143] a class is an extensible code template for object creation. Provides initial values for its state and implementations of its behaviour [144]. Usually, a class corresponds to a single file; nevertheless, in OOP it is possible to have multiple classes inside a single file. In non-OOP, a file contains only functions [145].
- **Commit**: the study predicts commits on the Version Control System (**VCS**) [137].
- **Fragment**: the study predicts only a fragment of the code (i.e., a specific line of a method/function) [82].
- **Machine Code** : instruction and data expressed in a form directly recognisable by the CPU [111]. The study predicts the representation of the source code by machine code [107].
- **Method**: the study predicts a method (e.g., Java) or a function (e.g., C) [146]

Fig. 4a reports the distribution of the granularity of the predicted entities. We note that:

- the most predicted entity is Class or File (17),
- methods are predicted less than Class or File (10) but more than the other types, and
- the least predicted entity is the commit, with only one study predicting it.

4.2.2. Dimension 2: Nature of the labelled entity

In this dimension, we characterise the nature of the predicted entities. The predicted entities can be of the following types:

- **Artificial**: The predicted entities are created manually for testing and evaluation purposes; examples include the Juliet Test Case from NIST/SARD. The main disadvantage of this type of entity is that it might not represent the industrial code [135].
- **Real**: The predicted entities are derived from open-source projects and online VCS. The main disadvantage of this nature of entities is that it may not be related to all types of vulnerability [147].
- **Mixed**: a combination of the above options [148].

Fig. 5d reports the nature of the entities. We can observe that:

- the most chosen nature is Real with 23 datasets,
- artificial data is a minority, with only six datasets, and
- only two datasets use a mixed data source.

4.2.3. Dimension 3: Type of labelling

In this dimension, we characterise the class of the label. The class of the label of the predicted entities can be of the following types:

- **Binary**: the label can only be true or false, i.e., the presence or absence of vulnerabilities [149].
- **Multiclass**: the label can have more than two values and represents the absence of a specific vulnerability, e.g., CWE-23: Relative Path Traversal, CWE-79: Improper Neutralisation of Input During Web Page Generation [111].

Fig. 5a reports the class of the label. We note that:

- 87% of the gathered dataset uses binary labelling, and
- 13% of the gathered datasets uses a multi-class label, using MITRE classification to label the entity.

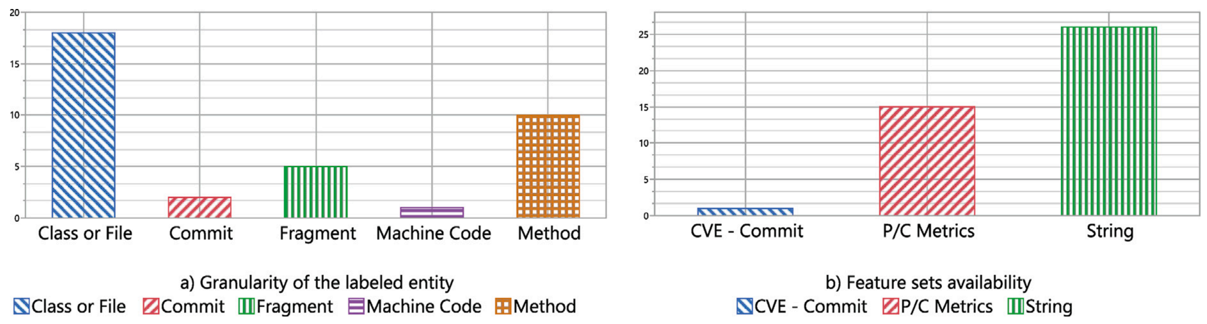


Fig. 4. Granularity of the labelled entity and Feature sets availability.

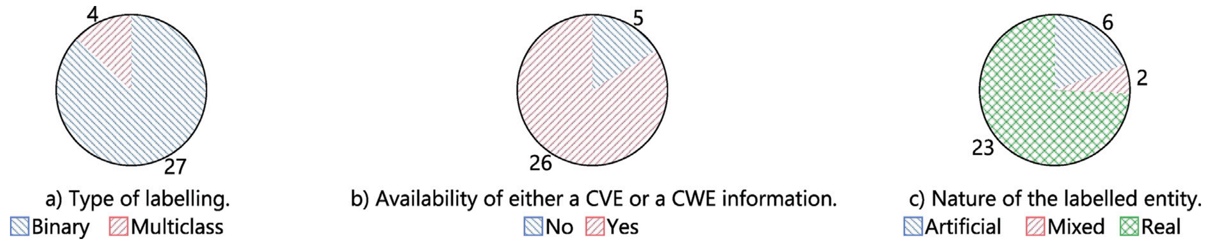


Fig. 5. Type of labelling, Nature of the labelled entity, and Availability of either a CVE or a CWE information.

4.2.4. *Dimension 4: Feature sets availability*

In this dimension, we analyse the feature sets in GDs. The possible feature sets can be of the following types:

- **String**: the feature set consists of strings derived from string operations in the source code or in the compiled machine code [150].
- **P/C Metric**: [115,151,152]: The feature set consists of process and complexity metrics [94].
- **CVE-Commit**: the feature set consists of characteristics of a CVE and its related commit [137].

Fig. 4b reports the feature sets available in GDs. We note that:

- the majority of GDs has string features, and
- the minority of GDs has CVE-commit.

4.2.5. *Dimension 5: Availability of either a CVE or a CWE information*

In this dimension, we analyse the information in the GD related to the vulnerable code. The values of the current dimension are:

- **CVE\CWE Info available**: the dataset reports either a CVE or a CWE information for each positive entity [89].
- **No CVE\CWE Info available**: the dataset reports neither a CVE nor a CWE information for each positive entity [135].

Fig. 5b reports how many GDs provide a CVE or a CWE information for each positive entity. We note that 26 GDs (84%) provide a CVE or a CWE.

4.2.6. *Dimension 6: Labelling process*

In this dimension, we characterise the process of labelling entities. The values of the current dimension are:

- **Automated**: the authors generated the dataset via a script [136].
- **Given**: the authors used the dataset as provided by some sort of official entity (e.g., the Juliet test case provided by NIST) [153].
- **Manual**: the authors manually curated at least one aspect of the dataset (e.g., SAP Dataset [91]).
- **Reused**: the authors reused the dataset from another study [154] reused their own datasets in Alves et al. [155].

Fig. 6c reports the type of labelling process. According to Fig. 6c:

- 15 out of the 31 GDs (48%) have at least one entity that is manually curated in the process, and
- only one out of the 31 GDs (3%) are reused from a previous study.

4.2.7. *Dimension 7: Manually curated*

This dimension investigates what is manually curated during the labelling process. We note that manual curation may only be done on a sample rather than the entire dataset, and a mixed dataset may only have a partial amount of entries containing CVE information. Researchers should analyse the datasets to understand whether the amount and type of curation fit their needs. The values of the current dimension are:

- **Code**: the author manually reviews the project's source code (code) or a fragment (slice) of it [133].
- **Commit**: the commit was manually reviewed [156].
- **Ticket**: the Jira ticket/NVD's CVE was manually reviewed [58].
- **Nothing**: nothing was manually curated [108].

Fig. 6a reports what is manually curated during labelling. According to Fig. 6a:

- most of the gathered datasets (i.e., 15 out of 31 GDs) have nothing manually curated, and
- the least curated entity is the commit with only 5 out of the 31 GDs.

4.2.8. *Dimension 8: Ground truth source*

In this dimension, we characterise the source of the GT. The authors used the following sources to obtain the GT:

- **Bugzilla**: GT from Bugzilla [12],
- **Fortify SCA**: GT from Fortify SCA tool [133],
- **Human Expert**: GT from human expert knowledge [69],
- **NVD**: GT from CVE information extracted from NVD [145],
- **SARD**: GT from ad-hoc test suites [157], and
- **SQLI-LABS**: GT from SQLI-LABS published data [96].

Fig. 6b presents the distribution of sources for the GT in VPS. We note that the NVD is the most prevalent source of the ground truth, i.e., 77% GDs used NVD. Additionally, human experts are the second

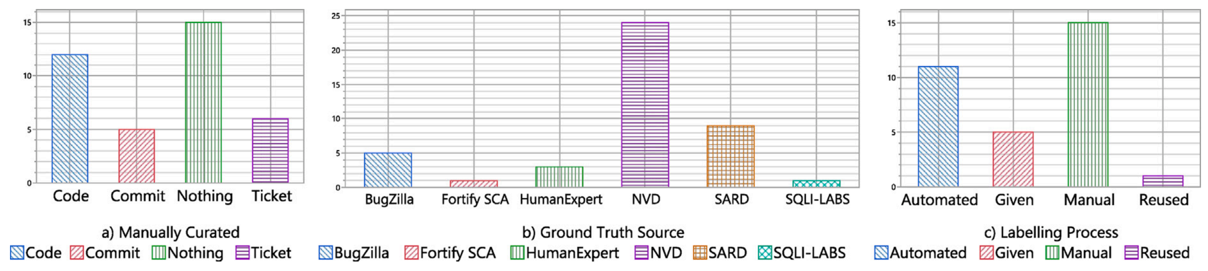


Fig. 6. Manually Curated, Negatives Assessed and, Labelling Process.

most effective source of the ground truth. Notably, in the single instance where a tool was employed, Fortify SCA was a static application security testing tool, no one used a dynamic security testing tool.

4.2.9. *Dimension 9: Negatives assessed*

In this dimension, we characterise the nature of negative entities. The values of the current dimension are:

- **Negatives Assessed**: what is labelled as not vulnerable results from an expert decision assessing the absence of vulnerability [95].
- **No Negatives Assessed**: what is labelled as not vulnerable is the complement of what experts assessed as the presence of vulnerability [12].

According to our findings:

- 71% of GDs do not assess negatives, and
- 29% of GDs do assess negatives.

We note that the 29% of GDs assessing negatives consider only one CWE at a time. In other words, a fragment of code deemed negative means that that fragment does not have a specific CWE, rather than that that fragment does not have any CWE. Therefore, no study assessed the absence of vulnerabilities in a code fragment.

5. Replication study

In this section, we show how VALIDATE can be used for conducting a non-exact replica study, thus replicating the use of VALIDATE by researchers.

5.1. *Dataset selection*

Among the possible datasets in VALIDATE, we were interested in finding an ”ML-ready” dataset with software product and process metrics as features. As an ML toolbox, we chose WEKA due to our successful research experience [53,54,81,158]. Thus, we selected the work of Tang et al. [84]; they evaluated three open-source projects, namely Moodle [159], PhpMyAdmin [160], Drupal [161]. Combining the information from the security advisors and the NVD, they created a dataset containing 223 vulnerabilities. They used such a dataset to investigate whether text mining prediction produces more accurate results than software features. Their results show that the text mining features delivered better performance than the software metrics.

5.2. *Design*

In this section, we explain the key design concept of our non-exact replica.

Table 5 reports the design elements of the original study and our non-exact replica. Our non-exact replica shares many design elements with the original study. We use the same six datasets of the original study based on the authors' open-source projects (i.e., Drupal, Moodle and PhpMyAdmin). As an ML tool, we use Weka [162] as the original paper. As per the independent variable (IV), we choose the same Metrics/Token. As accuracy metrics, we chose to use the same ones as in the original study: Inspection Ratio [84] ($\mathcal{I} = \frac{TP+FN}{TP+FP+TN+FN}$) and Recall [163–166] ($\mathcal{R} = \frac{TP}{TP+FN}$).

Our non-exact replica improves the original study on many design elements [167–172]. As a validation procedure, we use the same three-fold cross-validation of the original study. To avoid randomness in the split procedure that affects our results, we repeated the three-fold cross-validation 100 times instead of the three times in the original study. In VPS, feature selection is crucial as it allows researchers to identify and prioritise the most influential factors contributing to system vulnerabilities. Researchers can streamline their analysis and improve the accuracy of ML models [169]. Hence, as a feature selection, we use symmetric uncertainty (**SU**) as suggested by Zhao et al. [169]. Moreover, defect prediction and VPS datasets often exhibit imbalance [86]. We employ class balancing techniques to mitigate biases favouring majority classes in ML models. Notably, SMOTE [173] proves advantageous compared to other methods like oversampling and undersampling. SMOTE effectively tackles class imbalance by creating synthetic instances for the minority class, offering a robust solution without sacrificing data from the majority class to achieve dataset balance [170]. Finally, tuning ML models is essential in VPS because it allows for the optimisation of hyperparameters, enhancing the models performance and predictive accuracy [174]. Manual tuning involves adjusting parameters based on domain knowledge and experimentation to find the best configuration for a given dataset. In the context of VPS, where the nature of vulnerabilities and their manifestations in code can vary widely, manual tuning is particularly tedious and challenging. Auto-tuning models, such as AutoWeka [174], automate the hyperparameter tuning process, systematically exploring the parameter space to find the optimal configuration. The output of Auto-WEKA is the best classifier and the related best parameters given the provided dataset. Table 5 reports the resulting classifier and parameters for each project.

Our analysis procedure includes a statistical test to reject the null hypothesis of no difference between $\mathcal{I}$ and $\mathcal{R}$ in defect prediction provided by using Metrics or Tokens. Since our data strongly deviate from normality, we use the Wilcoxon signed-rank test [175] to test our null hypothesis.

5.3. *Results*

Fig. 7 reports the accuracy of text-mining-based models and software-metrics-based models. According to Fig. 7 the use of tokens provides higher $\mathcal{I}$ and $\mathcal{R}$ than the metrics. The results of the statistical test show a p-value lower than 0.0001 in the three datasets and two accuracy metrics. Therefore, our non-exact replica confirms the results of the original study even with a more sophisticated empirical procedure: text mining models have higher accuracy than software metrics-based models [84].

All experiments took about a week on one McAffee server model BG5500 running Windows Server 2022. The server has two Intel Xeon (R) CPU X5660, a base clock speed of 2.79 GHz and 72.0 GB of RAM with a clock speed of 1067 MHz.

Table 5
Study comparison.

	Original	Replica
Datasets	Drupal, PhpMyAdmin, Moodle	Drupal, PhpMyAdmin, Moodle
Machine Learning Tool	Weka	Weka
Independent Variable	Metrics/Token	Metrics/Token
Accuracy Metrics	Inspection | Recall	Inspection | Recall
Validation Techniques	1 time 3-fold cv	100 times 3-fold cv
Feature Selections	None	Symmetrical Uncertainty
Class Balancing	None	SMOTE
Hyperparameters Tuning	None	AutoWEKA
Statistical Test	None	Wilcoxon signed-rank
Classifiers	Random Forest	**Drupal**
		Metrics: RandomSubSpace + Random Forest
		Tokens: SGD
		Moodle
		Metrics: AdaBoostM1 + DecisionTable
		Tokens: SMO + NormalizedPolyKernel
		PhpMyAdmin
		Metrics: RandomForest
		Tokens: SMO + NormalizedPolyKernel

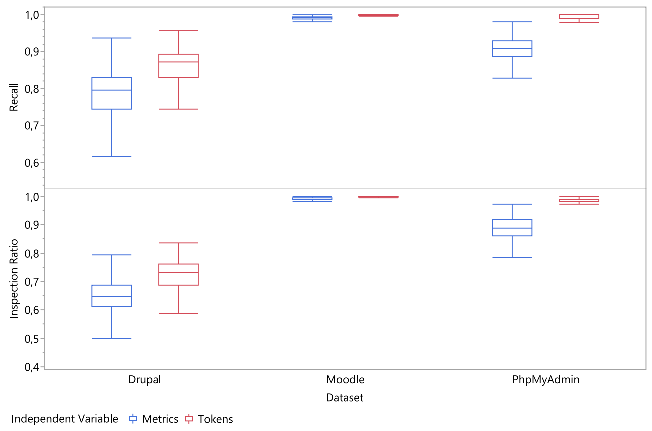


Fig. 7. Accuracy of text-mining-based models and software-metrics-based models.

6. Discussion

Our systematic review of the VPS datasets shows that 50% of the studies provide a dataset. This result is in line with Liu et al. [17], so it is necessary to enforce a replicability package for ML studies. Liem and Panichella [176] discuss the replicability and randomness in using ML in empirical SE studies. Specifically, they note a high variability in using ML techniques and tools with small to non-existent discussions on the randomicity of the specific ML technique or tool. Fig. 2 shows that out of the 22 studies that provide a reachable dataset (44% of the total), only one dataset is currently hosted in a stable repository. The main issue with unstable repositories lies in the volatility of the archive, which is due to their intrinsic non-trackable nature. For example, a GitHub repository can be archived without prior notice from the author; similarly, the central administration can update a personal website of an academic website. The issue of historical and locational traceability of digital objects leads GitHub to add academic features to its repositories [142]. Similarly to Mozilla Science Labs [177], Figshare [178], and Zenodo [140], GitHub allows users to generate DOI commits and integrate their repository with external archiving platforms. We encourage researchers to provide a reachable replication package and host it in a stable repository.

Regarding Dimension 1, Fig. 4 shows a majority of coarse-grained entities, that is, class or file; this result is in line with Morrison et al. [15],Arisholm and Briand [179].

Regarding Dimension 2, Fig. 5b shows that the majority (70% of the 31 GD) use real data (i.e., open-source projects) that are easier to analyse through their coarser grain than to build the project and analyse the byte-code, thus inducing the focus towards class or files. It should be noted that 85% of the GD uses a binary classification; therefore, many studies can use a large body of knowledge on classifiers. Although only 15% of the studies use multi-class classification, recently, the VPS community seems to be focussing on this kind of classification [111,180]. Regarding Dimension 6, Fig. 6c shows that the most used labelling process involves manually curating at least one aspect of the labelling process; nonetheless, it represents only 42% of the GD. We note that curating at least one aspect of the labelling process provides a more refined dataset; in the same vein, Ponta et al. [91] propose a five-year manually-reviewed dataset.

Our SDR shows that only 33% of the 31 GD assess the entity labelled negative (see Dimension 8, Fig. 6b). Furthermore, no study manually checked the absence of any vulnerability; i.e., what is labelled as negative is the absence of a particular vulnerability rather than all vulnerabilities. For instance, in the Juliet Test Case, “CWE321_Hard_Coded_Cryptographic_Key_ basic_81_goodG2B” is labelled as negative for CWE321 rather than for any CWE. Therefore, we can argue that what has been labelled as negative for a vulnerability might be positive for another vulnerability. Thus, past ML studies might have overestimated the false positive rate [23]. In conclusion, the labelling process is not yet standardised [64], and it is essential to assess the negative labels better.

Finally, on the possibility of reusing datasets according to their content, according to Dimension 4, Fig. 4b, the scientific community is driving its attention on the chance given by the NLP-alike technique for feature extraction directly from the source code or the machine code we call ”string feature”. As discussed in Section 3.3, the presence of a standardised vulnerability nomenclature, that is, CVE/CWE information, helps researchers focus on a specific family of vulnerabilities or gain more knowledge of their existence and interaction. Therefore, according to Fig. 5c, the scientific community appears to be aware of the issue since most GD, i.e., 71%, provide the CVE/CWE info.

Regarding Dimension 8, the findings from Fig. 6b have several implications for VPS. The first implication is that the prevalence of NVD as the most prolific source of ground truth suggests that it is the most accessible. Moreover, Fig. 6b suggests that utilising multiple sources to establish the ground truth or a combination of different sources, such as automated tools and human expertise, can provide a more comprehensive understanding of the potential vulnerability landscape, which is crucial for developing effective security measures.

According to Dimension 9, only 9% of the GD provides datasets compliant with this demand.

To understand coverage across multiple dimensions, we use a Burt table [181] which is a contingency table that displays the frequency of cases for two categorical variables in a suitable format for analysis. The Burt table helps identify the association patterns between two variables and provides an easy visual overview of the relationship between categorical variables. To interpret a Burt table, we should look for cells with high or low frequencies relative to the total number of cases. These indicate solid or weak associations between the two variables. Table 6 reports the Burt table values for each combination of categories in the MCA analysis.

To better understand Table 6, and specifically to facilitate the identification of unavailable datasets, we counted in Table 6, for each dimension value, the number of unavailable datasets in combination with all other dimensions values. For instance, if the number of missing combinations of a cell is zero, there is at least a dataset with that dimension value and all values of the other dimensions. A dimension with all zeros indicates the presence of at least a dataset with all values of that dimension and all the values of the other dimensions. A cell with many missing combinations indicates a low availability of datasets with that dimension value and all values of the other dimensions. Table 7 reports the number of missing combinations. According to Table 7:

- "CVE Info" and "Binary" are the two values with only one missing combinations. This means that datasets with "CVE Info" and "Cross Project" are available with all but one values of other dimensions.
- The value "Reused" of the dimension Labelling Process show the highest number of missing combinations, i.e., 25. This means there is a scarce availability of datasets with those values and values of other dimensions.
- The value "Human Expert" of the dimension Ground Truth Source show 22 missing combinations. Thus, too few studies involves human expert to curate or validate the ground truth

The above result implies that researchers can leverage existing datasets when in need of "CVE Info" and "Binary" features. In contrast, they will likely build their own when needing other features.

7. Threats to validity

This section discusses different threats to the validity [182] of the present study. We organise the discussion on two topics: SDR and non-exact replica.

Regarding SDR, our classification of datasets is driven by our experience in supporting SE through ML [167–172,183–186]. We did not use card sorting in our workflow as done in previous studies [187], because we felt that spreadsheets and comparisons proved more suited to our specific context as characterised by a relatively small number of dimensions, values, and coders. If on the one side, we are confident that our classification, and hence the filters in VALIDATE, can help researchers find useful datasets, on the other side there is a chance that researchers would like to search the datasets in VALIDATE according to unavailable filters. Researchers might require filters that differ in topic or granularity from the current ones. Therefore, we plan to accommodate future requests in changing the datasets classifications according to researchers' needs. The dataset provided by the researchers via email or through the repository referenced in the paper is assumed to be the data set used in the study. To mitigate this issue, we checked that the dataset was in accordance with the description in the paper of the data.

The authors of this paper manually performed the classification; therefore, there is the possibility that the classification is not reliable. We mitigated this threat to validity by:

- removing intrinsically subjective categories such as the ease of use or the ease of retrieving the datasets, and
- classifying each dataset in each category independently across researchers. Subsequently, we calculated Cohen [101] Kappa and checked a high level of agreement (see Section 4).

The exclusion criteria E7, i.e., citation count of less than 20, can hinder the validity of the work by excluding valuable studies that have not yet gained significant recognition in the academic community. For instance, the low number of studies in 2022 could be due to the exclusion criteria E7; i.e., recent relevant papers need more time to gain 20 citations. However, in light of the replicability principle, we have used the exclusion criteria in Croft et al. [64].

Regarding the non-exact replica, we use classifiers suggested by Auto-Weka. Auto-Weka has different parameters, including the running time and the optimised accuracy metric. We set the running time to two hours per dataset; we selected the proportion of accurate classifications as an optimised accuracy metric. There is a chance that with different parameters, auto-weka would find a better classifier that would lead to a different result from the current ones. Unfortunately, this is a common issue with tuning. To mitigate this issue, we tried different classifiers such as Random Forest [188] and IBK [189], and the current results still hold.

The ongoing evolution of vulnerability datasets, driven by the inclusion of undisclosed vulnerabilities, can potentially change our ground truth. Moreover, interpretability is essential in VPS as it facilitates model improvement and enables effective decision-making by clearly understanding how and why vulnerabilities are identified, thus promoting overall cybersecurity resilience. For instance, authors can revise a published dataset by changing a few values. We aim to keep VALIDATE up to date with researchers' changes.

We decided not to include size as a dimension to differentiate datasets because our datasets are highly heterogeneous in type. For instance, no single metric can characterise the size of datasets containing code and datasets containing project characteristics.

8. Conclusion

In this study, we analysed the availability and characteristics of the prediction studies' datasets. We performed an SDR of VPS datasets. Our results show that of the 50 primary studies, only 22 studies provide a reachable dataset. Of these 22 studies, only one study provides a dataset in a stable repository. Therefore, researchers should focus on where to publish their datasets.

We provide the first repository for VPS datasets to support researchers in finding and sharing datasets (VALIDATE). Our repository of datasets supports researchers in finding datasets of interest, hence avoiding reinventing the wheel; this translates into less effort, more reliability, and more reproducibility in dataset creation and use. Although there are many datasets, no dataset might meet the needs of a researcher. Therefore, our repository provides evidence of the need to create a new dataset. An exciting result of our SDR is that no study manually verified the absence of any vulnerability, i.e., negative labels were not adequately evaluated. Therefore, previous studies may have overestimated false negatives.

We show how VALIDATE can be used by performing a non-exact replica featuring a more sophisticated design. Our non-exact replica confirms the original study's results: text-mining-based models have higher accuracy than software-metrics-based models. Therefore, the availability of ready-to-use datasets helps in replication.

We provide a replication package that contains all the data used to generate graphs, statistical analysis, VALIDATE user guide, and all the references to the original studies [21].

In the future, we plan to improve the knowledge of our field in the following key areas:

Table 6
Burt table.

Dimension		Labelling Process				Granularity				
	Values	Automated	Given	Manual	Reused	Class or File	Commit	Fragment	Machine Code	Method
Labelling Process	Automated	28	0	0	0	15	0	2	0	11
	Given	0	10	0	0	5	0	5	0	0
	Manual	0	0	38	0	24	2	2	1	9
	Reused	0	0	0	1	0	0	1	0	0
Granularity	Class or File	15	5	24	0	44	0	0	0	0
	Commit	0	0	2	0	0	2	0	0	0
	Fragment	2	5	2	1	0	0	10	0	0
	Machine Code	0	0	1	0	0	0	0	1	0
	Method	11	0	9	0	0	0	0	0	20
Manually Curated	Code	2	2	18	0	10	0	6	1	5
	Commit	0	0	9	0	6	1	0	0	2
	Nothing	26	8	0	1	20	0	4	0	11
	Ticket	0	0	11	0	8	1	0	0	2
Assessed Negatives		0	8	11	0	7	0	5	0	7
CVE Info		23	10	36	0	40	2	9	1	17
Source Type	Artificial	2	10	0	1	5	0	8	0	0
	Mixed	0	0	3	0	0	0	2	0	1
	Real	26	0	35	0	39	2	0	1	19
Label Type	Binary	24	6	36	1	39	2	5	1	20
	Multiclass	4	4	2	0	5	0	5	0	0
Feature Set	String	0	0	3	0	3	0	0	0	0
	CVE - Commit	0	0	2	0	0	2	0	0	0
	P/C Metrics	13	0	15	1	22	0	1	0	6
	String	15	10	18	0	19	0	9	1	14
Ground Truth Source	BugZilla	7	0	3	0	6	0	0	0	4
	Fortify SCA	0	0	6	0	6	0	0	0	0
	HumanExpert	0	0	4	0	4	0	0	0	0
	NVD	17	3	21	1	22	2	6	1	11
	SARD	4	6	4	0	5	0	4	0	5
	SQLI-LABS	0	1	0	0	1	0	0	0	0

Dimension		Manually Curated				Assessed Negatives	CVE Info	Source Type			Label Type	
	Values	Code	Commit	Nothing	Ticket			Artificial	Mixed	Real	Binary	Multiclass
Labelling Process	Automated	2	0	26	0	0	23	2	0	26	24	4
	Given	2	0	8	0	8	10	10	0	0	6	4
	Manual	18	9	0	11	11	36	0	3	35	36	2
	Reused	0	0	1	0	0	0	1	0	0	1	0
Granularity	Class or File	10	6	20	8	7	40	5	0	39	39	5
	Commit	0	1	0	1	0	2	0	0	2	2	0
	Fragment	6	0	4	0	5	9	8	2	0	5	5
	Machine Code	1	0	0	0	0	1	0	0	1	1	0
	Method	5	2	11	2	7	17	0	1	19	20	0
Manually Curated	Code	22	0	0	0	7	22	4	3	15	16	6
	Commit	0	9	0	0	2	9	0	0	9	9	0
	Nothing	0	0	35	0	8	29	9	0	26	31	4
	Ticket	0	0	0	11	2	9	0	0	11	11	0
Assessed Negatives		7	2	8	2	19	19	8	2	9	15	4
CVE Info		22	9	29	9	19	69	12	3	54	59	10
Source Type	Artificial	4	0	9	0	8	12	13	0	0	7	6
	Mixed	3	0	0	0	2	3	0	3	0	3	0
	Real	15	9	26	11	9	54	0	0	61	57	4
Label Type	Binary	16	9	31	11	15	59	7	3	57	67	0
	Multiclass	6	0	4	0	4	10	6	0	4	0	10
Feature Set	String	1	1	0	1	0	3	0	0	3	3	0
	CVE - Commit	0	1	0	1	0	2	0	0	2	2	0
	P/C Metrics	6	3	14	6	2	25	1	0	28	27	2
	String	15	4	21	3	17	39	12	3	28	35	8
Ground Truth Source	BugZilla	1	0	7	2	0	8	0	0	10	10	0
	Fortify SCA	2	2	0	2	0	6	0	0	6	6	0
	HumanExpert	2	0	0	2	0	3	0	0	4	4	0
	NVD	14	6	18	4	10	37	6	1	35	35	7
	SARD	3	1	9	1	8	14	6	2	6	11	3
	SQLI-LABS	0	0	1	0	1	1	1	0	0	1	0

(continued on next page)

- **Security by design**: Since many security vulnerabilities are at the design level [190], we plan to investigate how to codify architectural styles and patterns into datasets ready to use for vulnerability prediction.
- **Analyse study replicability**. Recent studies [16,17,191–198] focused on the replicability of empirical SE experiments, while other studies, such as the one conducted by Neto [199] focused on strategies to help researchers create replicability packages. We envision a model to automatically analyse the replicability of studies and to support researchers in facilitating the replicability of future studies.
- **Automated characterisation of the studies and their datasets.** In this study, we manually curated the content of VALIDATE. In the future, we plan to automate the categorisation process by creating an ML/DL model to categorise new datasets automatically.

Table 6 (*continued*).

Dimension		Feature Set				Ground Truth Source					
	Values	String	CVE - Commit	P/C Metrics	String	BugZilla	Fortify SCA	HumanExpert	NVD	SARD	SQLI-LABS
Labelling Process	Automated	0	0	13	15	7	0	0	17	4	0
	Given	0	0	0	10	0	0	0	3	6	1
	Manual	3	2	15	18	3	6	4	21	4	0
	Reused	0	0	1	0	0	0	0	1	0	0
Granularity	Class or File	3	0	22	19	6	6	4	22	5	1
	Commit	0	2	0	0	0	0	0	2	0	0
	Fragment	0	0	1	9	0	0	0	6	4	0
	Machine Code	0	0	0	1	0	0	0	1	0	0
	Method	0	0	6	14	4	0	0	11	5	0
Manually Curated	Code	1	0	6	15	1	2	2	14	3	0
	Commit	1	1	3	4	0	2	0	6	1	0
	Nothing	0	0	14	21	7	0	0	18	9	1
	Ticket	1	1	6	3	2	2	2	4	1	0
Assessed Negatives		0	0	2	17	0	0	0	10	8	1
CVE Info		3	2	25	39	8	6	3	37	14	1
Source Type	Artificial	0	0	1	12	0	0	0	6	6	1
	Mixed	0	0	0	3	0	0	0	1	2	0
	Real	3	2	28	28	10	6	4	35	6	0
Label Type	Binary	3	2	27	35	10	6	4	35	11	1
	Multiclass	0	0	2	8	0	0	0	7	3	0
Feature Set	String	3	0	0	0	0	0	0	3	0	0
	CVE - Commit	0	2	0	0	0	0	0	2	0	0
	P/C Metrics	0	0	29	0	7	3	3	14	2	0
	String	0	0	0	43	3	3	1	23	12	1
Ground Truth Source	BugZilla	0	0	7	3	10	0	0	0	0	0
	Fortify SCA	0	0	3	3	0	6	0	0	0	0
	HumanExpert	0	0	3	1	0	0	4	0	0	0
	NVD	3	2	14	23	0	0	0	42	0	0
	SARD	0	0	2	12	0	0	0	0	14	0
	SQLI-LABS	0	0	0	1	0	0	0	0	0	1

Table 7
Burt table summary.

Dimension	Values	No. of missing combinations
Labelling Process	Automated	14
	Given	17
	Manual	6
	Reused	25
Granularity	Class or File	7
	Commit	22
	Fragment	14
	Machine Code	24
	Method	14
Manually Curated	Code	9
	Commit	14
	Nothing	11
	Ticket	13
Assessed Negatives		10
CVE Info		1
Source Type	Artificial	14
	Mixed	21
	Real	6
Label Type	Binary	1
	Multiclass	15
Feature Set	String	22
	CVE - Commit	22
	P/C Metrics	8
	String	5
Ground Truth Source	BugZilla	18
	Fortify SCA	21
	HumanExpert	22
	NVD	5
	SARD	11
	SQLI-LABS	23

CRediT authorship contribution statement

Matteo Esposito: Writing – review & editing, Writing – original draft, Visualization, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Davide Falessi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://validatetool.org/.

References

[1] Y. Yoo, Computing in everyday life: A call for research on experiential computing, MIS Q. 34 (2010) 213–231, URL: http://misq.org/computing-in-everyday-live-a-call-for-research-on-experiential-computing.html.

[2] T. Beauvisage, Computer usage in daily life, in: D.R.O. Jr., R.B. Arthur, K. Hinckley, M.R. Morris, S.E. Hudson, S. Greenberg (Eds.), Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4–9, 2009, ACM, 2009, pp. 575–584, http://dx.doi.org/10.1145/1518701.1518791.

[3] M.A. Khan, H. El-Sayed, S. Malik, M.T. Zia, J. Khan, N. Alkaabi, H.A. Ignatious, Level-5 autonomous driving - are we there yet? A review of research literature, ACM Comput. Surv. 55 (2023) 27:1–27:38, http://dx.doi.org/10.1145/3485767.

[4] Y. Charalabidis, Z. Lachana, On the science foundation of digital governance and transformation, in: Y. Charalabidis, M.A. Cunha, D. Sarantis (Eds.), ICEGOV 2020: 13th International Conference on Theory and Practice of Electronic Governance, Athens, Greece, 23–25 September, 2020, ACM, 2020, pp. 214–221, http://dx.doi.org/10.1145/3428502.3428532.

[5] R. Telang, S. Wattal, An empirical analysis of the impact of software vulnerability announcements on firm stock price, IEEE Trans. Softw. Eng. 33 (2007) 544–557, http://dx.doi.org/10.1109/TSE.2007.70712.

[6] G. Tassey, The economic impacts of inadequate infrastructure for software testing, 2002.

[7] C. Ten, G. Manimaran, C. Liu, Cybersecurity for critical infrastructures: Attack and defense modeling, IEEE Trans. Syst. Man Cybern. A 40 (2010) 853–865.

[8] J. Jang-Jaccard, S. Nepal, A survey of emerging threats in cybersecurity, J. Comput. System Sci. 80 (2014) 973–993.

[9] Y. Li, Q. Liu, A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments, Energy Rep. 7 (2021) 8176–8186, http://dx.doi.org/10.1016/j.egyr.2021.08.126, URL: https://www.sciencedirect.com/science/article/pii/S2352484721007289.

[10] H. Hanif, M.H.N.B.M. Nasir, M.F.A. Razak, A. Firdaus, N.B. Anuar, The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches, J. Netw. Comput. Appl. 179 (2021) 103009, http://dx.doi.org/10.1016/j.jnca.2021.103009.

[11] V.A.F. Almeida, D. Doneda, J. de Souza Abreu, Cyberwarfare and digital governance, IEEE Internet Comput. 21 (2017) 68–71, http://dx.doi.org/10.1109/MIC.2017.23.

[12] Y. Shin, L.A. Williams, Can traditional fault prediction models be used for vulnerability prediction? Empir. Softw. Eng. 18 (2013) 25–59, http://dx.doi.org/10.1007/s10664-011-9190-8.

[13] T. Zimmermann, N. Nagappan, L.A. Williams, Searching for a needle in a haystack: Predicting security vulnerabilities for windows vista, in: ICST, IEEE Computer Society, 2010, pp. 421–428.

[14] T.L. Graves, A.F. Karr, J.S. Marron, H.P. Siy, Predicting fault incidence using software change history, IEEE Trans. Softw. Eng. 26 (2000) 653–661.

[15] P. Morrison, K. Herzig, B. Murphy, L.A. Williams, Challenges with applying vulnerability prediction models, in: D.M. Nicol (Ed.), Proceedings of the 2015 Symposium and Bootcamp on the Science of Security, HotSoS 2015, Urbana, IL, USA, April 21–22, 2015, ACM, 2015, pp. 4:1–4:9, http://dx.doi.org/10.1145/2746194.2746198.

[16] J.M. González-Barahona, G. Robles, On the reproducibility of empirical software engineering studies based on data retrieved from development repositories, Empir. Softw. Eng. 17 (2012) 75–89.

[17] C. Liu, C. Gao, X. Xia, D. Lo, J.C. Grundy, X. Yang, On the reproducibility and replicability of deep learning in software engineering, ACM Trans. Softw. Eng. Methodol. 31 (2022) 15:1–15:46.

[18] C. Drummond, Replicability is not reproducibility: Nor is it good science, in: Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML, 2009.

[19] S. Kapoor, A. Narayanan, Leakage and the reproducibility crisis in ml-based science, 2022, http://dx.doi.org/10.48550/ARXIV.2207.07048, URL: https://arxiv.org/abs/2207.07048.

[20] R. Pan, M. Bagherzadeh, T.A. Ghaleb, L.C. Briand, Test case selection and prioritization using machine learning: a systematic literature review, Empir. Softw. Eng. 27 (2022) 29, http://dx.doi.org/10.1007/s10664-021-10066-6.

[21] M. Esposito, D. Falessi, Replication package for "VALIDATE: A deep dive into vulnerability prediction datasets", 2023, http://dx.doi.org/10.5281/zenodo.10135002.

[22] B.A. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, S.G. Linkman, Systematic literature reviews in software engineering - A systematic literature review, Inf. Softw. Technol. 51 (2009) 7–15, http://dx.doi.org/10.1016/j.infsof.2008.09.009.

[23] F. Cheirdari, G. Karabatis, Analyzing false positive source code vulnerabilities using static analysis tools, in: N. Abe, H. Liu, C. Pu, X. Hu, N.K. Ahmed, M. Qiao, Y. Song, D. Kossmann, B. Liu, K. Lee, J. Tang, J. He, J.S. Saltz (Eds.), IEEE International Conference on Big Data, IEEE BigData 2018, Seattle, WA, USA, December 10–13, 2018, IEEE, 2018, pp. 4782–4788, http://dx.doi.org/10.1109/BigData.2018.8622456.

[24] D. Zhang, J.J.P. Tsai, Machine Learning Applications in Software Engineering, in: Series on Software Engineering and Knowledge Engineering, World Scientific Publishing Co. Inc., USA, 2005.

[25] D. Zhang, J.J.P. Tsai, Machine learning and software engineering, Softw. Qual. J. 11 (2003) 87–119, http://dx.doi.org/10.1023/A:1023760326768.

[26] R.W. Schwanke, S.J. Hanson, Using neural networks to modularize software, Mach. Learn. 15 (1994) 137–168, http://dx.doi.org/10.1007/BF00993275.

[27] C. Ryan, Automatic re-engineering of software using genetic programming, 2000, http://dx.doi.org/10.1007/978-1-4615-4631-3.

[28] Y. Yang, X. Xia, D. Lo, T. Bi, J. Grundy, X. Yang, Predictive models in software engineering: Challenges and opportunities, ACM Trans. Softw. Eng. Methodol. 31 (2022) http://dx.doi.org/10.1145/3503509.

[29] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A.M. Vollmer, S. Wagner, Software engineering for AI-based systems: A survey, ACM Trans. Softw. Eng. Methodol. 31 (2022) http://dx.doi.org/10.1145/3487043.

[30] G. Giray, A software engineering perspective on engineering machine learning systems: State of the art and challenges, J. Syst. Softw. 180 (2021) 111031.

[31] Y. Lyu, G.K. Rajbahadur, D. Lin, B. Chen, Z.M.J. Jiang, Towards a consistent interpretation of aiops models, ACM Trans. Softw. Eng. Methodol. 31 (2022) 16:1–16:38.

[32] R. Kapur, B. Sodhi, Oss effort estimation using software features similarity and developer activity-based metrics, ACM Trans. Softw. Eng. Methodol. 31 (2022) http://dx.doi.org/10.1145/3485819.

[33] V.H.S. Durelli, R.S. Durelli, S.S. Borges, A.T. Endo, M.M. Eler, D.R.C. Dias, M. de Paiva Guimarães, Machine learning applied to software testing: A systematic mapping study, IEEE Trans. Reliab. 68 (2019) 1189–1212.

[34] C. Ayala, B. Turhan, X. Franch, N. Juristo, Use and misuse of the term experiment in mining software repositories research, IEEE Trans. Softw. Eng. (2021) 1, http://dx.doi.org/10.1109/TSE.2021.3113558.

[35] Z. Wan, X. Xia, D. Lo, G.C. Murphy, How does machine learning change software development practices? IEEE Trans. Softw. Eng. 47 (2021) 1857–1871.

[36] A. Mashkoor, T. Menzies, A. Egyed, R. Ramler, Artificial intelligence and software engineering: Are we ready? Computer 55 (2022) 24–28.

[37] A.E. Hassan, A. Mockus, R.C. Holt, P.M. Johnson, Guest editor's introduction: Special issue on mining software repositories, IEEE Trans. Softw. Eng. 31 (2005) 426–428.

[38] O. Vandecruys, D. Martens, B. Baesens, C. Mues, M.D. Backer, R. Haesen, Mining software repositories for comprehensible software fault prediction models, J. Syst. Softw. 81 (2008) 823–839.

[39] R. Abdalkareem, S. Mujahid, E. Shihab, A machine learning approach to improve the detection of ci skip commits, IEEE Trans. Softw. Eng. (2020).

[40] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, D. Poshyvanyk, Machine learning-based prototyping of graphical user interfaces for mobile apps, IEEE Trans. Softw. Eng. 46 (2018) 196–221.

[41] S. Kim, E.J. Whitehead, Y. Zhang, Classifying software changes: Clean or buggy? IEEE Trans. Softw. Eng. 34 (2008) 181–196.

[42] V.B. Livshits, T. Zimmermann, Dynamine: finding common error patterns by mining software revision histories, in: ESEC/SIGSOFT FSE, ACM, 2005, pp. 296–305.

[43] L. Neil, S. Mittal, A. Joshi, Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports, in: 2018 IEEE International Conference on Intelligence and Security Informatics, ISI 2018, Miami, FL, USA, November 9–11, 2018, IEEE, 2018, pp. 7–12, http://dx.doi.org/10.1109/ISI.2018.8587375.

[44] J. Liang, O. Mizuno, Analyzing involvements of reviewers through mining a code review repository, in: K. Matsuda, K. Matsumoto, A. Monden (Eds.), 2011 Joint Conf of 21st Int'l Workshop on Software Measurement and the 6th Int'l Conference on Software Process and Product Measurement, IWSM/Mensura 2011, Nara, Japan, November 3–4, 2011, IEEE Computer Society, 2011, pp. 126–132, http://dx.doi.org/10.1109/IWSM-MENSURA.2011.33.

[45] C.C. Williams, J.K. Hollingsworth, Automatic mining of source code repositories to improve bug finding techniques, IEEE Trans. Softw. Eng. 31 (2005) 466–480, http://dx.doi.org/10.1109/TSE.2005.63.

[46] G. Bavota, Mining unstructured data in software repositories: Current and future trends, in: Leaders of Tomorrow Symposium: Future of Software Engineering, FOSE@SANER 2016, Osaka, Japan, March 14, 2016, IEEE Computer Society, 2016, pp. 1–12, http://dx.doi.org/10.1109/SANER.2016.47.

[47] K. Herzig, S. Just, A. Zeller, The impact of tangled code changes on defect prediction models, Empir. Softw. Eng. 21 (2016) 303–336, http://dx.doi.org/10.1007/s10664-015-9376-6.

[48] G. Antoniol, K. Ayari, M.D. Penta, F. Khomh, Y. Guéhéneuc, Is it a bug or an enhancement?: a text-based approach to classify change requests, in: I. Onut, A. Jaramillo, G. Jourdan, D.C. Petriu, W. Chen (Eds.), Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, CASCON 2018, Markham, Ontario, Canada, October 29–31, 2018, ACM, 2018, pp. 2–16, URL: https://dl.acm.org/citation.cfm?id=3291293.

[49] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, A. Ihara, K. Matsumoto, The impact of mislabelling on the performance and interpretation of defect prediction models, in: A. Bertolino, G. Canfora, S.G. Elbaum (Eds.), 37th IEEE/ACM International Conference on Software Engineering, Vol. 1, ICSE 2015, Florence, Italy, May 16–24, 2015, IEEE Computer Society, 2015, pp. 812–823, http://dx.doi.org/10.1109/ICSE.2015.93.

[50] A. Bachmann, C. Bird, F. Rahman, P.T. Devanbu, A. Bernstein, The missing links: bugs and bug-fix commits, in: G. Roman, A. van der Hoek (Eds.), Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7–11, 2010, ACM, 2010, pp. 97–106, http://dx.doi.org/10.1145/1882291.1882308.

[51] Y. Zhou, J.K. Siow, C. Wang, S. Liu, Y. Liu, SPI: automated identification of security patches via commits, ACM Trans. Softw. Eng. Methodol. 31 (2022) 13:1–13:27.

[52] C. Zou, X. Wang, Y. Gao, J. Xue, Buddy stacks: Protecting return addresses with efficient thread-local storage and runtime re-randomization, ACM Trans. Softw. Eng. Methodol. 31 (2022) http://dx.doi.org/10.1145/3494516.

[53] B. Vandehei, D.A. da Costa, D. Falessi, Leveraging the defects life cycle to label affected versions and defective classes, ACM Trans. Softw. Eng. Methodol. 30 (2021) 24:1–24:35.

[54] D. Falessi, A. Ahluwalia, M.D. Penta, The impact of dormant defects on defect prediction: A study of 19 apache projects, ACM Trans. Softw. Eng. Methodol. 31 (2022) 4:1–4:26.

[55] A. Ibrahim, M. El-Ramly, A. Badr, Beware of the vulnerability! how vulnerable are github's most popular php applications? in: 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications, AICCSA, 2019, pp. 1–7, http://dx.doi.org/10.1109/AICCSA47632.2019.9035265.

[56] S. Khan, I. Kabanov, Y. Hua, S.E. Madnick, A systematic analysis of the capital one data breach: Critical lessons learned, ACM Trans. Priv. Secur. 26 (2023) 3:1–3:29.

[57] M. Esposito, S. Moreschini, V. Lenarduzzi, D. Hästbacka, D. Falessi, Can we trust the default vulnerabilities severity? in: 2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation, SCAM, 2023, pp. 265–270, http://dx.doi.org/10.1109/SCAM59687.2023.00037.

[58] J. Stuckman, J. Walden, R. Scandariato, The effect of dimensionality reduction on software vulnerability prediction models, IEEE Trans. Reliab. 66 (2016) 17–37.

[59] L.K. Shar, L.C. Briand, H.B.K. Tan, Web application vulnerability prediction using hybrid program analysis and machine learning, IEEE Trans. Dependable Secur. Comput. 12 (2014) 688–707.

[60] Y. Li, S. Ji, C. Lyu, Y. Chen, J. Chen, Q. Gu, C. Wu, R. Beyah, V-fuzz: Vulnerability prediction-assisted evolutionary fuzzing for binary programs, IEEE Trans. Cybern. (2020).

[61] W.E. Wong, Within-project and cross-project defect prediction" r "damba: Detecting android malware by orgb analysis" r "large-scale empirical studies on effort-aware security vulnerability prediction methods" r "learning code context information to predict comment, IEEE Trans. Reliab. 69 (2020).

[62] P. Oser, R.W. van der Heijden, S. Lüders, F. Kargl, Risk prediction of iot devices based on vulnerability analysis, ACM Trans. Priv. Secur. 25 (2022) 1–36.

[63] S. Rahimi, M. Zargham, Vulnerability scrying method for software vulnerability discovery prediction without a vulnerability database, IEEE Trans. Reliab. 62 (2013) 395–407.

[64] R. Croft, Y. Xie, M.A. Babar, Data preparation for software vulnerability prediction: A systematic literature review, IEEE Trans. Softw. Eng. (2022) 1, http://dx.doi.org/10.1109/TSE.2022.3171202.

[65] G. Jabeen, S. Rahim, W. Afzal, D. Khan, A.A. Khan, Z. Hussain, T. Bibi, Machine learning techniques for software vulnerability prediction: a comparative study, Appl. Intell. 52 (2022) 17614–17635.

[66] W. Zheng, J. Gao, X. Wu, F. Liu, Y. Xun, G. Liu, X. Chen, The impact factors on the performance of machine learning-based vulnerability detection: A comparative study, J. Syst. Softw. 168 (2020) 110659.

[67] Y. Zhu, G. Lin, L. Song, J. Zhang, The application of neural network for software vulnerability detection: a review, Neural Comput. Appl. 35 (2023) 1279–1301, http://dx.doi.org/10.1007/s00521-022-08046-y.

[68] G. Partenza, T. Amburgey, L. Deng, J. Dehlinger, S. Chakraborty, Automatic identification of vulnerable code: Investigations with an ast-based neural network, in: COMPSAC, IEEE, 2021, pp. 1475–1482.

[69] Z. Yu, C. Theisen, L.A. Williams, T. Menzies, Improving vulnerability inspection efficiency using active learning, IEEE Trans. Softw. Eng. 47 (2021) 2401–2420, http://dx.doi.org/10.1109/TSE.2019.2949275.

[70] J. Sayyad Shirabad, T. Menzies, The PROMISE repository of software engineering databases, in: School of Information Technology and Engineering, University of Ottawa, Canada, 2005, URL: http://promise.site.uottawa.ca/SERepository.

[71] L. Cheikhi, A. Abran, Promise and ISBSG software engineering data repositories: A survey, in: 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, Ankara, Turkey, October 23–26, 2013, IEEE Computer Society, 2013, pp. 17–24, http://dx.doi.org/10.1109/IWSM-Mensura.2013.13.

[72] S. Alelyani, H. Liu, L. Wang, The effect of the characteristics of the dataset on the selection stability, in: IEEE 23rd International Conference on Tools with Artificial Intelligence, ICTAI 2011, Boca Raton, FL, USA, November 7–9, 2011, IEEE Computer Society, 2011, pp. 970–977, http://dx.doi.org/10.1109/ICTAI.2011.167.

[73] D. Oreski, S. Oreski, B. Klicek, Effects of dataset characteristics on the performance of feature selection techniques, Appl. Soft Comput. 52 (2017) 109–119, http://dx.doi.org/10.1016/j.asoc.2016.12.023.

[74] Y. Nong, R. Sharma, A. Hamou-Lhadj, X. Luo, H. Cai, Open science in software engineering: A study on deep learning-based vulnerability detection, IEEE Trans. Softw. Eng. 49 (2023) 1983–2005, http://dx.doi.org/10.1109/TSE.2022.3207149.

[75] M. Zhang, X. de Carné de Carnavalet, L. Wang, A. Ragab, Large-scale empirical study of important features indicative of discovered vulnerabilities to assess application security, IEEE Trans. Inf. Forensics Secur. 14 (2019) 2315–2330, http://dx.doi.org/10.1109/TIFS.2019.2895963.

[76] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, Inf. Softw. Technol. 53 (2011) 625–637.

[77] B. Kitchenham, Procedures for Performing Systematic Reviews, Technical Report 2004, 2004.

[78] V.R. Basili, G. Caldiera, H.D. Rombach, The goal question metric approach, Encycl. Softw. Eng. (1994).

[79] A.R. Henderson, Evidence-Based Medicine—How to Practice and Teach EBM. D. L. Sackett, W. S. Richardson, W. Rosenberg, and R. B. Haynes. New York: Churchill Livingstone, 1997 250pp. Paperback, 24.99. ISBN 0-443-05686-2, Clin. Chem. 43 (1997) 2014, http://dx.doi.org/10.1093/clinchem/43.10.2014.

[80] T. Lewowski, L. Madeyski, How far are we from reproducible research on code smell detection? A systematic literature review, Inf. Softw. Technol. 144 (2022) 106783, http://dx.doi.org/10.1016/j.infsof.2021.106783.

[81] D. Falessi, J. Huang, L. Narayana, J.F. Thai, B. Turhan, On the need of preserving order of data when validating within-project defect classifiers, Empir. Softw. Eng. 25 (2020) 4805–4830.

[82] M. Zagane, M.K. Abdi, M. Alenezi, Deep learning for software vulnerabilities detection using code metrics, IEEE Access 8 (2020) 74562–74570, http://dx.doi.org/10.1109/ACCESS.2020.2988557.

[83] M.A. Albahar, A modified maximal divergence sequential auto-encoder and time delay neural network models for vulnerable binary codes detection, IEEE Access 8 (2020) 14999–15006, http://dx.doi.org/10.1109/ACCESS.2020.2965726.

[84] Y. Tang, F. Zhao, Y. Yang, H. Lu, Y. Zhou, B. Xu, Predicting vulnerable components via text mining or software metrics? an effort-aware perspective, in: 2015 IEEE International Conference on Software Quality, Reliability and Security, QRS 2015, Vancouver, BC, Canada, August 3–5, 2015, IEEE, 2015, pp. 27–36, http://dx.doi.org/10.1109/QRS.2015.15.

[85] T. Nguyen, T. Le, K. Nguyen, O.Y. de Vel, P. Montague, J.C. Grundy, D. Phung, Deep cost-sensitive kernel machine for binary software vulnerability detection, in: PAKDD (2), in: Lecture Notes in Computer Science, vol. 12085, Springer, 2020, pp. 164–177.

[86] M. Esposito, D. Falessi, Uncovering the hidden risks: The importance of predicting bugginess in untouched methods, in: 2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation, SCAM, 2023, pp. 277–282, http://dx.doi.org/10.1109/SCAM59687.2023.00039.

[87] D. Falessi, S.M. Laureani, J. Çarka, M. Esposito, D.A. da Costa, Enhancing the defectiveness prediction of methods and classes via JIT, Empir. Softw. Eng. 28 (2023) 37, http://dx.doi.org/10.1007/s10664-022-10261-z.

[88] Y. Zhang, D. Lo, X. Xia, B. Xu, J. Sun, S. Li, Combining software metrics and text features for vulnerable file prediction, in: ICECCS, IEEE Computer Society, 2015, pp. 40–49.

[89] H. Wang, G. Ye, Z. Tang, S.H. Tan, S. Huang, D. Fang, Y. Feng, L. Bian, Z. Wang, Combining graph-based learning with automated data collection for code vulnerability detection, IEEE Trans. Inf. Forensics Secur. 16 (2021) 1943–1958, http://dx.doi.org/10.1109/TIFS.2020.3044773.

[90] G. Lin, J. Zhang, W. Luo, L. Pan, Y. Xiang, O.Y. de Vel, P. Montague, Cross-project transfer representation learning for vulnerable function discovery, IEEE Trans. Ind. Inform. 14 (2018) 3289–3297, http://dx.doi.org/10.1109/TII.2018.2821768.

[91] S.E. Ponta, H. Plate, A. Sabetta, M. Bezzi, C. Dangremont, A manually-curated dataset of fixes to vulnerabilities of open-source software, in: M.D. Storey, B. Adams, S. Haiduc (Eds.), Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada, IEEE / ACM, 2019, pp. 383–387, http://dx.doi.org/10.1109/MSR.2019.00064.

[92] R. Li, C. Feng, X. Zhang, C. Tang, A lightweight assisted vulnerability discovery method using deep neural networks, IEEE Access 7 (2019) 80079–80092.

[93] E.R. Russo, A.D. Sorbo, C.A. Visaggio, G. Canfora, Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities, J. Syst. Softw. 156 (2019) 84–99.

[94] X. Chen, Y. Zhao, Z. Cui, G. Meng, Y. Liu, Z. Wang, Large-scale empirical studies on effort-aware security vulnerability prediction methods, IEEE Trans. Reliab. 69 (2020) 70–87, http://dx.doi.org/10.1109/TR.2019.2924932.

[95] N.P.D.S. Medeiros, N.R. Ivaki, P. Costa, M. Vieira, Software metrics as indicators of security vulnerabilities, in: 28th IEEE International Symposium on Software Reliability Engineering, ISSRE 2017, Toulouse, France, October 23–26, 2017, IEEE Computer Society, 2017, pp. 216–227, http://dx.doi.org/10.1109/ISSRE.2017.11.

[96] Y. Fang, S. Han, C. Huang, R. Wu, Tap: A static analysis model for php vulnerabilities based on token and deep learning technology, PLoS One 14 (2019) e0225196.

[97] D. Mitropoulos, G. Gousios, P. Papadopoulos, V. Karakoidas, P. Louridas, D. Spinellis, The vulnerability dataset of a large software ecosystem, in: BADGERS@ESORICS, IEEE, 2014, pp. 69–74.

[98] K.R. Felizardo, E. Mendes, M. Kalinowski, É.F. de Souza, N.L. Vijaykumar, Using forward snowballing to update systematic reviews in software engineering, in: ESEM, ACM, 2016, pp. 53:1–53:6.

[99] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: EASE, ACM, 2014, pp. 38:1–38:10.

[100] D.S. Cruzes, T. Dybå, Recommended steps for thematic synthesis in software engineering, in: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22–23, 2011, IEEE Computer Society, 2011, pp. 275–284, http://dx.doi.org/10.1109/ESEM.2011.36.

[101] J. Cohen, A coefficient of agreement for nominal scales, Educ. Psychol. Meas. 20 (1960) 37–46.

[102] J. Sim, C.C. Wright, The kappa statistic in reliability studies: Use, interpretation, and sample size requirements, Phys. Ther. 85 (2005) 257–268, http://dx.doi.org/10.1093/ptj/85.3.257.

[103] G. Lin, J. Zhang, W. Luo, L. Pan, O.Y. de Vel, P. Montague, Y. Xiang, Software vulnerability discovery via learning multi-domain knowledge bases, IEEE Trans. Dependable Secur. Comput. 18 (2021) 2469–2485, http://dx.doi.org/10.1109/TDSC.2019.2954088.

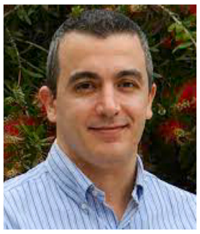
[104] T.H.M. Le, D. Hin, R. Croft, M.A. Babar, Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning, in: 36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15–19, 2021, IEEE, 2021, pp. 717–729, http://dx.doi.org/10.1109/ASE51524.2021.9678622.

[105] L. Pascarella, F. Palomba, A. Bacchelli, On the performance of method-level bug prediction: A negative result, J. Syst. Softw. 161 (2020) http://dx.doi.org/10.1016/j.jss.2019.110493.

[106] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, Z. Chen, Sysevr: A framework for using deep learning to detect software vulnerabilities, IEEE Trans. Dependable Secur. Comput. 19 (2022) 2244–2258, http://dx.doi.org/10.1109/TDSC.2021.3051525.
[107] S. Liu, M. Dibaei, Y. Tai, C. Chen, J. Zhang, Y. Xiang, Cyber vulnerability intelligence for internet of things binary, IEEE Trans. Ind. Inform. 16 (2020) 2154–2163, http://dx.doi.org/10.1109/TII.2019.2942800.
[108] S. Chakraborty, R. Krishna, Y. Ding, B. Ray, Deep learning based vulnerability detection: Are we there yet, IEEE Trans. Softw. Eng. (2021).
[109] U.S. Congress, National institute of standards and technology, 1901, URL: https://www.nist.gov/.
[110] P. Black, A software assurance reference dataset: Thousands of programs with known bugs, J. Res. Natl. Inst. Stand. Technol. 123 (2018) http://dx.doi.org/10.6028/jres.123.005.
[111] D. Zou, S. Wang, S. Xu, Z. Li, H. Jin, μvuldeepecker: A deep learning-based system for multiclass vulnerability detection, IEEE Trans. Dependable Secur. Comput. 18 (2021) 2224–2236, http://dx.doi.org/10.1109/TDSC.2019.2942930.
[112] E. Frank, M.A. Hall, G. Holmes, R.B. Kirkby, B. Pfahringer, I.H. Witten, Weka: A machine learning workbench for data mining, in: Data Mining and Knowledge Discovery Handbook, Springer, 2005, pp. 1305–1314, http://dx.doi.org/10.1007/0-387-25465-X_62, URL: https://hdl.handle.net/10289/1497 [62].
[113] R. Roscher, B. Bohn, M.F. Duarte, J. Garcke, Explainable machine learning for scientific insights and discoveries, IEEE Access 8 (2020) 42200–42216, http://dx.doi.org/10.1109/ACCESS.2020.2976199.
[114] Meiliana, S. Karim, 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.
[115] S.R. Chidamber, C.F. Kemerer, A metrics suite for object oriented design, IEEE Trans. Softw. Eng. 20 (1994) 476–493.
[116] R. Moser, W. Pedrycz, G. Succi, A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, in: ICSE, ACM, 2008, pp. 181–190.
[117] K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, Investigating effect of design metrics on fault proneness in object-oriented systems, J. Object Technol. 6 (2007) 127–141.
[118] H. Wang, T.M. Khoshgoftaar, N. Seliya, How many software metrics should be selected for defect prediction? in: FLAIRS Conference, AAAI Press, 2011.
[119] D.E. Mann, S.M. Christey, Towards a common enumeration of vulnerabilities, in: 2nd Workshop on Research with Security Vulnerability Databases, Purdue University, West Lafayette, Indiana, 1999.
[120] B. Martin, M. Brown, A. Paller, D. Kirby, Common Weakness Emunaration, first ed., MITRE Corporation, 2011, URL: http://cwe.mire.org/top25/.
[121] MIT, 1958. URL: https://www.mitre.org.
[122] Yanming Yang, Xin Xia, David Lo, John Grundy, A survey on deep learning for software engineering, ACM Comput. Surv. (ISSN: 0360-0300) 54 (10s) (2022) http://dx.doi.org/10.1145/3505243.
[123] S. Reis, R. Abreu, A ground-truth dataset of real security patches, 2021, CoRR abs/2110.09635, URL: https://arxiv.org/abs/2110.09635.
[124] X. Wu, W. Zheng, X. Chen, F. Wang, D. Mu, Cve-assisted large-scale security bug report dataset construction method, J. Syst. Softw. 160 (2020) http://dx.doi.org/10.1016/j.jss.2019.110456.
[125] J. Fan, Y. Li, S. Wang, T.N. Nguyen, A C/C++ code vulnerability dataset with code changes and CVE summaries, in: S. Kim, G. Gousios, S. Nadi, J. Hejderup (Eds.), MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29–30 June, 2020, ACM, 2020, pp. 508–512, http://dx.doi.org/10.1145/3379597.3387501.
[126] G.P. Bhandari, A. Naseer, L. Moonen, Cvefixes: automated collection of vulnerabilities and their fixes from open-source software, in: S. McIntosh, X. Xia, S. Amasaki (Eds.), PROMISE '21: 17th International Conference on Predictive Models and Data Analytics in Software Engineering, Athens Greece, August 19–20, 2021, ACM, 2021, pp. 30–39, http://dx.doi.org/10.1145/3475960.3475985.
[127] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y.L. Traon, M. Harman, The importance of accounting for real-world labelling when predicting software vulnerabilities, in: M. Dumas, D. Pfahl, S. Apel, A. Russo (Eds.), Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26–30, 2019, ACM, 2019, pp. 695–705, http://dx.doi.org/10.1145/3338906.3338941.
[128] D. Eilers, C. Köpp, C. Gleue, M.H. Breitner, It's not a bug, it's a feature: How visual model evaluation can help to incorporate human domain knowledge in data science, in: Y.J. Kim, R. Agarwal, J.K. Lee (Eds.), Proceedings of the International Conference on Information Systems - Transforming Society with Digital Innovation, ICIS 2017, Seoul, South Korea, December 10–13, 2017, Association for Information Systems, 2017, URL: http://aisel.aisnet.org/icis2017/DataScience/Presentations/15.
[129] P. Pickerill, H.J. Jungen, M. Ochodek, M. Mackowiak, M. Staron, PHANTOM: curating github for engineered software projects using time-series clustering, Empir. Softw. Eng. 25 (2020) 2897–2929, http://dx.doi.org/10.1007/s10664-020-09825-8.
[130] C. Sas, A. Capiluppi, Labelgit: A dataset for software repositories classification using attributed dependency graphs, 2021, CoRR abs/2103.08890, URL: https://arxiv.org/abs/2103.08890.
[131] R. Widyasari, S.Q. Sim, C. Lok, H. Qi, J. Phan, Q. Tay, C. Tan, F. Wee, J.E. Tan, Y. Yieh, B. Goh, F. Thung, H.J. Kang, T. Hoang, D. Lo, E.L. Ouh, Bugsinpy: a database of existing bugs in python programs to enable controlled testing and debugging studies, in: P. Devanbu, M.B. Cohen, T. Zimmermann (Eds.), ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8–13, 2020, ACM, 2020, pp. 1556–1560, http://dx.doi.org/10.1145/3368089.3417943.
[132] C. Pornprasit, C. Tantithamthavorn, Deeplinedp: Towards a deep learning approach for line-level defect prediction, IEEE Trans. Softw. Eng. (2023) 1, http://dx.doi.org/10.1109/TSE.2022.3144348.
[133] R. Scandariato, J. Walden, A. Hovsepyan, W. Joosen, Predicting vulnerable software components via text mining, IEEE Trans. Softw. Eng. 40 (2014) 993–1006, http://dx.doi.org/10.1109/TSE.2014.2340398.
[134] J. Stuckman, J. Walden, R. Scandariato, The effect of dimensionality reduction on software vulnerability prediction models, IEEE Trans. Reliab. 66 (2017) 17–37, http://dx.doi.org/10.1109/TR.2016.2630503.
[135] R. Li, C. Feng, X. Zhang, C. Tang, A lightweight assisted vulnerability discovery method using deep neural networks, IEEE Access 7 (2019) 80079–80092, http://dx.doi.org/10.1109/ACCESS.2019.2923227.
[136] J. Fan, Y. Li, S. Wang, T.N. Nguyen, A C/C++ code vulnerability dataset with code changes and CVE summaries, in: S. Kim, G. Gousios, S. Nadi, J. Hejderup (Eds.), MSR 20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29–30 June, 2020, ACM, 2020, pp. 508–512, http://dx.doi.org/10.1145/3379597.3387501.
[137] S.E. Ponta, H. Plate, A. Sabetta, M. Bezzi, C. Dangremont, A manually-curated dataset of fixes to vulnerabilities of open-source software, in: M.D. Storey, B. Adams, S. Haiduc (Eds.), Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26–27 May 2019, Montreal, Canada, IEEE / ACM, 2019, pp. 383–387, http://dx.doi.org/10.1109/MSR.2019.00064.
[138] G. Nikitopoulos, K. Dritsa, P. Louridas, D. Mitropoulos, Crossvul: a cross-language vulnerability dataset with commit data, in: D. Spinellis, G. Gousios, M. Chechik, M.D. Penta (Eds.), ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23–28, 2021, ACM, 2021, pp. 1565–1569, http://dx.doi.org/10.1145/3468264.3473122.
[139] A. Dann, H. Plate, B. Hermann, S.E. Ponta, E. Bodden, Identifying challenges for OSS vulnerability scanners - A study & test suite, IEEE Trans. Softw. Eng. 48 (2022) 3613–3625, http://dx.doi.org/10.1109/TSE.2021.3101739.
[140] European Organization For Nuclear Research, OpenAIRE, Zenodo, 2013, http://dx.doi.org/10.25495/7GXK-RD71, URL: https://www.zenodo.org/.
[141] M. GitHub Inc. (2007), Github, 2007, URL: https://github.com/.
[142] A. Smith, Improving github for science | the github blog, 2014, URL: https://github.blog/2014-05-14-improving-github-for-science/.
[143] B. Stroustrup, What is object-oriented programming? in: ECOOP, in: Lecture Notes in Computer Science, vol. 276, Springer, 1987, pp. 51–70.
[144] E. Gamma, R. Helm, R.E. Johnson, J.M. Vlissides, Design patterns: Abstraction and reuse of object-oriented design, in: ECOOP, in: Lecture Notes in Computer Science, vol. 707, Springer, 1993, pp. 406–431.
[145] J. Walden, J. Stuckman, R. Scandariato, Predicting vulnerable components: Software metrics vs text mining, in: 25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3–6, 2014, IEEE Computer Society, 2014, pp. 23–33, http://dx.doi.org/10.1109/ISSRE.2014.32.
[146] X. Cheng, H. Wang, J. Hua, M. Zhang, G. Xu, L. Yi, Y. Sui, Static detection of control-flow-related vulnerabilities using graph embedding, in: J. Pang, J. Sun (Eds.), 24th International Conference on Engineering of Complex Computer Systems, ICECCS 2019, Guangzhou, China, November 10–13, 2019, IEEE, 2019, pp. 41–50, http://dx.doi.org/10.1109/ICECCS.2019.00012.
[147] S. Liu, G. Lin, Q. Han, S. Wen, J. Zhang, Y. Xiang, Deepbalance: Deep-learning and fuzzy oversampling for vulnerability detection, IEEE Trans. Fuzzy Syst. 28 (2020) 1329–1343, http://dx.doi.org/10.1109/TFUZZ.2019.2958558.
[148] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, Y. Zhong, Vuldeepecker: A deep learning-based system for vulnerability detection, in: 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18–21, 2018, The Internet Society, 2018, URL: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-2_Li_paper.pdf.
[149] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y.L. Traon, M. Harman, The importance of accounting for real-world labelling when predicting software vulnerabilities, in: M. Dumas, D. Pfahl, S. Apel, A. Russo (Eds.), Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26–30, 2019, ACM, 2019, pp. 695–705, http://dx.doi.org/10.1145/3338906.3338941.

[150] V. Nguyen, T. Le, O.Y. de Vel, P. Montague, J.C. Grundy, D. Phung, H.W. Lauw and R.C. Wong and A. Ntoulas and E. Lim and S. Ng and S.J. Pan, Dual-component deep domain adaptation: A new approach for cross project software vulnerability detection, in: Advances in Knowledge Discovery and Data Mining - 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I, in: Lecture Notes in Computer Science, vol. 12084, Springer, 2020, pp. 699–711, http://dx.doi.org/10.1007/978-3-030-47426-3_54.
[151] T. McCabe, A complexity measure, IEEE Trans. Softw. Eng. SE-2 (1976) 308–320, http://dx.doi.org/10.1109/TSE.1976.233837.
[152] S.D. Palma, D.D. Nucci, F. Palomba, D.A. Tamburri, Within-project defect prediction of infrastructure-as-code using product and process metrics, IEEE Trans. Softw. Eng. 48 (2022) 2086–2104.
[153] N. Saccente, J. Dehlinger, L. Deng, S. Chakraborty, Y. Xiong, Project achilles: A prototype tool for static method-level vulnerability detection of java source code using a recurrent neural network, in: 34th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASE Workshops 2019, San Diego, CA, USA, November 11–15, 2019, IEEE, 2019, pp. 114–121, http://dx.doi.org/10.1109/ASEW.2019.00040.
[154] H. Alves, B. Fonseca, N. Antunes, Software metrics and security vulnerabilities: Dataset and exploratory study, in: 12th European Dependable Computing Conference, EDCC 2016, Gothenburg, Sweden, September 5–9, 2016, IEEE Computer Society, 2016, pp. 37–44, http://dx.doi.org/10.1109/EDCC.2016.34.
[155] H. Alves, B. Fonseca, N. Antunes, Experimenting machine learning techniques to predict vulnerabilities, in: 2016 Seventh Latin-American Symposium on Dependable Computing, LADC 2016, Cali, Colombia, October 19–21, 2016, IEEE Computer Society, 2016, pp. 151–156, http://dx.doi.org/10.1109/LADC.2016.32.
[156] A. Gkortzis, D. Mitropoulos, D. Spinellis, Vulinoss: a dataset of security vulnerabilities in open-source systems, in: A. Zaidman, Y. Kamei, E. Hill (Eds.), Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28–29, 2018, ACM, 2018, pp. 18–21, http://dx.doi.org/10.1145/3196398.3196454.
[157] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, Z. Chen, Sysevr: A framework for using deep learning to detect software vulnerabilities, IEEE Trans. Dependable Secur. Comput. 19 (2022) 2244–2258, http://dx.doi.org/10.1109/TDSC.2021.3051525.
[158] D. Falessi, J. Roll, J.L.C. Guo, J. Cleland-Huang, Leveraging historical associations between requirements and source code to identify impacted classes, IEEE Trans. Softw. Eng. 46 (2020) 420–441.
[159] M. Dougiamas, Moodle - open-source learning platform, 2002, URL: https://moodle.org/.
[160] T. phpMyAdmin Project, Phpmyadmin, 1998, URL: https://www.phpmyadmin.net/.
[161] D. community, Drupal - open source cms, 2001, URL: https://www.drupal.org/.
[162] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, third ed., Morgan Kaufmann, Elsevier, 2011, URL: https://www.worldcat.org/oclc/262433473.
[163] J. Yerushalmy, Statistical problems in assessing methods of medical diagnosis, with special reference to X-ray techniques, Public Health Rep. (1896-1970) 62 (1947) 1432–1449, URL: http://www.jstor.org/stable/4586294.
[164] D.G. Altman, J.M. Bland, Statistics notes: Diagnostic tests 1: sensitivity and specificity, BMJ 308 (1994) 1552, http://dx.doi.org/10.1136/bmj.308.6943.1552, URL: https://www.bmj.com/content/308/6943/1552, arXiv:https://www.bmj.com/content/308/6943/1552.full.pdf.
[165] T. Fawcett, An introduction to ROC analysis, Pattern Recognit. Lett. 27 (2006) 861–874.
[166] D.M.W. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2020, CoRR abs/2010.16061.
[167] J. Jiarpakdee, C. Tantithamthavorn, C. Treude, The impact of automated feature selection techniques on the interpretation of defect models, Empir. Softw. Eng. 25 (2020) 3590–3638, http://dx.doi.org/10.1007/s10664-020-09848-1.
[168] G.K. Rajbahadur, S. Wang, Y. Kamei, A.E. Hassan, The impact of feature importance methods on the interpretation of defect classifiers, 2022, CoRR abs/2202.02389, URL: https://arxiv.org/abs/2202.02389.
[169] K. Zhao, Z. Xu, M. Yan, T. Zhang, D. Yang, W. Li, A comprehensive investigation of the impact of feature selection techniques on crashing fault residence prediction models, Inf. Softw. Technol. 139 (2021) 106652, http://dx.doi.org/10.1016/j.infsof.2021.106652.
[170] C. Tantithamthavorn, A.E. Hassan, K. Matsumoto, The impact of class rebalancing techniques on the performance and interpretation of defect prediction models, IEEE Trans. Softw. Eng. 46 (2020) 1200–1219, http://dx.doi.org/10.1109/TSE.2018.2876537.
[171] W. Fu, T. Menzies, X. Shen, Tuning for software analytics: Is it really necessary? Inf. Softw. Technol. 76 (2016) 135–146, http://dx.doi.org/10.1016/j.infsof.2016.04.017.
[172] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, K. Matsumoto, The impact of automated parameter optimization on defect prediction models, IEEE Trans. Softw. Eng. 45 (2019) 683–711, http://dx.doi.org/10.1109/TSE.2018.2794977.
[173] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[174] C. Thornton, F. Hutter, H.H. Hoos, K. Leyton-Brown, Auto-weka: combined selection and hyperparameter optimization of classification algorithms, in: KDD, ACM, 2013, pp. 847–855.
[175] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80, http://dx.doi.org/10.2307/3001968, URL: https://app.dimensions.ai/details/publication/pub.1102728208.
[176] C.C.S. Liem, A. Panichella, Run, forest, run? on randomization and reproducibility in predictive software engineering, 2020, CoRR abs/2012.08387.
[177] T.M. Foundation, Mozilla labs, 1998, URL: https://labs.mozilla.org/.
[178] D. Science, Figshare, 2011, URL: https://figshare.com/.
[179] E. Arisholm, L.C. Briand, Predicting fault-prone components in a java legacy system, in: ISESE, ACM, 2006, pp. 8–17.
[180] H. Kekül, B. Ergen, H. Arslan, A multiclass hybrid approach to estimating software vulnerability vectors and severity score, J. Inf. Secur. Appl. 63 (2021) 103028.
[181] D. Karlis, M. greenacre (2007) correspondence analysis in practice, 2009.
[182] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012, http://dx.doi.org/10.1007/978-3-642-29044-2.
[183] S. Bayley, D. Falessi, Optimizing prediction intervals by tuning random forest via meta-validation, 2018, CoRR abs/1801.07194.
[184] D. Falessi, G. Cantone, The effort savings from using NLP to classify equivalent requirements, IEEE Softw. 36 (2019) 48–55.
[185] A. Ahluwalia, M.D. Penta, D. Falessi, On the need of removing last releases of data when using or validating defect prediction models, 2020, CoRR abs/2003.14376.
[186] A.D. Rodriguez, J. Cleland-Huang, D. Falessi, Leveraging intermediate artifacts to improve automated trace link retrieval, in: ICSME, IEEE, 2021, pp. 81–92.
[187] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schröter, C. Weiss, What makes a good bug report? IEEE Trans. Softw. Eng. 36 (2010) 618–643, http://dx.doi.org/10.1109/TSE.2010.63.
[188] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[189] D. Aha, D. Kibler, Instance-based learning algorithms, Mach. Learn. 6 (1991) 37–66.
[190] S. Rehman, K. Mustafa, Research on software design level security vulnerabilities, ACM SIGSOFT Softw. Eng. Notes 34 (2009) 1–5, http://dx.doi.org/10.1145/1640162.1640171.
[191] G. Rodríguez-Pérez, G. Robles, J.M. González-Barahona, Reproducibility and credibility in empirical software engineering: A case study based on a systematic literature review of the use of the SZZ algorithm, Inf. Softw. Technol. 99 (2018) 164–176, http://dx.doi.org/10.1016/j.infsof.2018.03.009.
[192] C.V.C. de Magalhães, F.Q.B. da Silva, R.E.S. Santos, M. Suassuna, Investigations about replication of empirical studies in software engineering: A systematic mapping study, Inf. Softw. Technol. 64 (2015) 76–101, http://dx.doi.org/10.1016/j.infsof.2015.02.001.
[193] J.C. Carver, N.J. Juzgado, M.T. Baldassarre, S. Vegas, Replications of software engineering experiments, Empir. Softw. Eng. 19 (2014) 267–276, http://dx.doi.org/10.1007/s10664-013-9290-8.
[194] N.J. Juzgado, Towards understanding replication of software engineering experiments, in: 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, Maryland, USA, October 10–11, 2013, IEEE Computer Society, 2013, p. 4, http://dx.doi.org/10.1109/ESEM.2013.64.
[195] M. Cruz, B. Bernárdez, A. Durán, J.A. Galindo, A. Ruiz-Cortés, Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018, IEEE Access 8 (2020) 26773–26791, http://dx.doi.org/10.1109/ACCESS.2019.2952191.
[196] S. Amann, S. Beyer, K. Kevic, H.C. Gall, Software mining studies: Goals, approaches, artifacts, and replicability, in: B. Meyer, M. Nordio (Eds.), Software Engineering - International Summer Schools, LASER 2013-2014, Elba, Italy, Revised Tutorial Lectures, in: Lecture Notes in Computer Science, vol. 8987, Springer, 2014, pp. 121–158, http://dx.doi.org/10.1007/978-3-319-28406-4_5.
[197] G. Robles, Replicating MSR: A study of the potential replicability of papers published in the mining software repositories proceedings, in: J. Whitehead, T. Zimmermann (Eds.), Proceedings of the 7th International Working Conference on Mining Software Repositories, MSR 2010 (Co-Located with ICSE), Cape Town, South Africa, May 2–3, 2010, Proceedings, IEEE Computer Society, 2010, pp. 171–180, http://dx.doi.org/10.1109/MSR.2010.5463348.
[198] R. Dror, G. Baumer, M. Bogomolov, R. Reichart, Replicability analysis for natural language processing: Testing significance with multiple datasets, Trans. Assoc. Comput. Linguist. 5 (2017) 471–486, http://dx.doi.org/10.1162/tacl_a_00074.
[199] A.A. Neto, A strategy to support replications of controlled experiments in software engineering, ACM SIGSOFT Softw. Eng. Notes 44 (2019) 23, http://dx.doi.org/10.1145/3356773.3356796.

Matteo Esposito is a Ph.D. student at the University of Rome Tor Vergata, Italy. He is also R&D Vice-Director for Multitel SRL. He is the Chairman of the ACM Rome Tor Vergata Student Chapter. His main research interest is machine learning to support secure software engineering. He received his MSc and BSc degrees in Computer Science Engineering from the University of Rome Tor Vergata, Italy.



Davide Falessi is an Associate Professor of Software Engineering at the University of Rome Tor Vergata, Italy. He is the Associate Editor in Software Economics of IEEE Software and a senior member of IEEE. He is a reviewer board member of the IEEE Transactions on Software Engineering. He has been the Guest Editor of special issues in several journals, including the Empirical Software Engineering Journal, the Journal of Systems and Software and IEEE Software. His main research interest is in devising and empirically assessing scalable solutions for developing software-intensive systems. He received his Ph.D., MSc, and BSc degrees in Computer Engineering from the University of Rome Tor Vergata, Italy.