A Systematic Literature Review of Explainable AI for Software Engineering

Ahmad Haji Mohammadkhani[a], Nitin Sai Bommi[b], Mariem Daboussi[c], Onkar Sabnis[d], Chakkrit Tantithamthavorn[e], Hadi Hemmati[f]

[a]*University of Calgary, Calgary, Alberta, Canada*
[b]*University of Hyderabad, Hyderabad, India*
[c]*INSAT , Tunis, Tunisia*
[d]*Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India*
[e]*Monash University, Melbourne, Victoria, Australia*
[f]*York University, Toronto, Ontario, Canada*

Abstract

Context: In recent years, leveraging machine learning (ML) techniques has become one of the main solutions to tackle many software engineering (SE) tasks, in research studies (ML4SE). This has been achieved by utilizing state-of-the-art models that tend to be more complex and black-box, which is led to less explainable solutions that reduce trust and uptake of ML4SE solutions by professionals in the industry.

Objective: One potential remedy is to offer explainable AI (XAI) methods to provide the missing explainability. In this paper, we aim to explore to what extent XAI has been studied in the SE community (XAI4SE) and provide a comprehensive view of the current state-of-the-art as well as challenge and roadmap for future work.

Method: We conduct a systematic literature review on 24 (out of 869 primary studies that were selected by keyword search) most relevant published studies in XAI4SE. We have three research questions that were answered by meta-analysis of the collected data per paper.

Results: Our study reveals that among the identified studies, software maintenance (%68) and particularly defect prediction has the highest share on the SE stages and tasks being studied. Additionally, we found that XAI methods were mainly applied to classic ML models rather than more complex models. We also noticed a clear lack of standard evaluation metrics for XAI methods in the literature which has caused confusion among researchers and a lack of benchmarks for comparisons.

Conclusions: XAI has been identified as a helpful tool by most studies, which we cover in the systematic review. However, XAI4SE is a relatively new domain with a lot of untouched potentials, including the SE tasks to help with, the ML4SE methods to explain, and the types of explanations to offer. This study encourages the researchers to work on the identified challenges and roadmap reported in the paper.

Keywords: Explainable AI for Software Engineering (XAI4SE), Systematic Review,, Machine Learning for Software Engineering (ML4SE), Explainable AI, Interpretable AI

1. Introduction

In the past few decades, advancement in Machine Learning (ML) has exponentially increased with the combination of powerful machines, robust algorithms, and easier access to vast amounts of data. As a result, at present, ML models have been developed in many critical domains such as in healthcare, banking, finance, terrorism detection [1, 2, 3].

In the field of software engineering as well, ML has been dominating many studies and activities. AI-based solutions have been applied on many automation tasks such as source code [4], test case [5], patch [6], specification generation [7], prediction tasks such as defect or vulnerability prediction [8], and recommendation tasks such as API recommendation [9].

Easy access to an abundance of software repositories (e.g., GitHub, StackOverflow, execution logs, etc.) has made SE an ideal field for data-driven ML algorithms to thrive. During the past decade, there has been many publications in sub-domains of SE research such as Mining Software Repositories (Software Analytics) and Automated Software Engineering that study applying ML techniques on various SE tasks such as API recom-

mendation [9], risk prediction [10], code comprehension [11], code generation [4], code retrieval [12], code summarization [13], bug-fixing processes [14], software testing [5], fault localization [15], effort estimation [16], documentation [17], clone detection [18], program synthesis [19], etc.

Often case, the models with higher complexity such as SVMs [20], and deep neural networks [21] have achieved higher predictive accuracy in these tasks and thus recommended to be used by practitioners. However, similar to other fields, like NLP or Image Processing, as the ML models become more complex and achieve better accuracies, understanding their decision-making process becomes much harder. State-of-the-art models evolved from using algorithms like decision trees or logistic regression that are intrinsically explainable, to using Deep Learning (DL) architectures, which basically offer no explanation for their decisions. Even though the final results may improve, more questions arise regarding their black-box nature.

This black-box nature of the solution can cause issues with trusting these models at different levels. It will provoke distrust in managers and customers, especially in more critical tasks. For instance, in the case of SE tasks whenever an automated solution is suggested to be deployed, e.g., a generated code or patch, etc.) where the wrong predictions/generations may cause expensive failure or extra manual overhead to check and fix, leading to a lack of adoption in practice [22]. Also, ethical challenges and the risk of immoral biases emerge [23] and it leaves the researchers with less clue about the right direction for improving their models [24].

In other words, acquiring the explainable output from the so-called black-box models, is one of the key problems of the existing machine learning models, to be adopted in practice. Though the black-box models themselves learn the connection between the features and the final output, they do not clearly describe the actual relationship between the given set of features and the individual prediction bestowed by the models. It is extremely important to know why a model makes a specific prediction to retain the model's reliability, in many domain including most SE tasks. For that, the model should be able to offer explanations for an individual prediction or the model's overall approach to the practitioners.

Software practitioners as the main users of ML models in SE would only be interested in adopting ML-based code recommendation, bug repair tools, etc., if they are persuaded that the models have learned what they "should have" and predict or generate what they are "meant to", from the perspective of the human experts. Without establishing such trust any solution is seemed as unreliable with potential unexpected consequences that wastes more resources and time, than it has saved by automation. In addition, if the outputs of the models are not informative enough, they can only lead to more ambiguity rather than helping the developers and users.

As another example, let's look at the defect prediction problem, which is one of the most studied tasks in SE by researchers when it comes to applying ML in SE. It has taken advantage of ML/DL models for many years. The goal is to predict if a piece of code or file is defective or not. In this scenario, if the model finds a file defective, but not any more information about its decision, the developer will have no clue on what to look for if the model provides the reasons for the file's being defective or specify some lines or tokens as the possible defective parts, it would be easy for the software engineers to minimize the bugs based on the grounds provided and validate the ML-based recommendation.

In short, model explainability facilitates the debugging process, bias detection (recourse to individuals who are adversely affected by the model predictions), assess when to trust model predictions in decision-making, and determining if they are suitable for deployment.

Thus, there is a compelling need to overcome the trade-off between accuracy and interpretability to provide trustworthy, fair, robust models for real-world applications. As a consequence, AI researchers have focused on the space of eXplainable Artificial Intelligence (XAI) [25], which provides an explainable and understandable interpretation of the behavior of AI systems.

Responding to the growing attraction in the industry and academia to use AI models in SE in recent years and the critical need for interpretation of AI models that are getting more complex, we conducted a systematic literature review (SLR). The SLR is based on relevant journal and conference papers that offer explainability methods for SE tasks or propose ML models that include any type of explanation. We conducted this research to provide practitioners and researchers a better insight into the efforts that have been made in this topic so far and the state-of-the-art of XAI in software engineering. In order to do so we have defined three research questions.

- **RQ1: What are the main software engineering areas and tasks for which Explainable AI approaches have been used to date?**

RQ1 aims to understand to what extent XAI has been used already in the SE community, which can poten-

tially reveal which areas need further study. As our analysis show, "software maintenance" is the highest explored area and defect prediction is the most favorable task for the XAI researchers in SE by far, but areas such as testing and program repair have not used XAI much, even though they have used ML a lot.

- **RQ2: What are the Explainable AI approaches adopted for SE tasks?**

The goal of RQ2 is to understand what XAI methods the SE community are using, which in turn will help us guide the future work by identifying methods that have shown good results already and potetnial methods that are not studied yet. Our analysis shows that most of the explanations so far are coming from self-explaining models and in the form of model internals (e.g., variable coefficients). Also, we can see that LIME and ANOVA are two of the mostly utilized XAI methods. We also see that higher-level explanations such as visualization and explanations in natural language are less sutied in XAI4SE.

- **RQ3: How useful XAI has been for SE?**

In RQ3, we focus on the usefulness of XAI methods and how the offered explanation has helped the users to improve or better understand the models. The goal is to have a realistic view if the potential benefits that can steer the future research in this domian. To answer this question, we collect the original authors' view, based on their results, on the usefulness, not our subjective opinion. One of the most interesting improvements reported in the studies is the usage of XAI to make the defect prediction models more precise while finding bugs and defects. Also, we discuss the lack of standard evaluation metrics according to the studies and how human-centered evaluations should be beneficial.

Finally, we have summarized the limitations and challenges of XAI4SE and provided a roadmap for future works.

To the extent of our knowledge, this is the first SLR for XAI in software engineering. The key contributions of this paper are:

- We present a consolidated body of knowledge of research on different aspects of XAI in software engineering
- We conducted a comprehensive search on the most influential venues in the field and selected 24 related papers that match our criteria.
- We analyzed these 24 papers to get a better insight into different characteristics of the problem in hand (e.g. their objective, datasets, representation, etc.), the models that are used, and the varieties of explainability that they offer.
- Finally, we identified gaps and limitations and have suggested a roadmap for the future.

2. Explainable AI in a Nutshell

One of the common issues in the literature on Explainable AI is that there are several related concepts in this domain that are often used interchangeably in different articles. Therefore, in this section, we provide a brief description and the definitions of the terms that are widely used in Explainable AI jargon which help to understand the related literature and the concepts more precisely. This will also make the basis for the definition of XAI in SE, which will be used in this paper.

2.1. Explainability and Interpretability

The most common term which hinders the establishment of concepts in XAI is the interchangeable use of interpretability and explainability in the literature. Although these terms are closely related there are some works that identify and distinguish these terms.

Though there are no clear mathematical definitions for interpretability and explainability, few works in this line attempt to clarify these terms. According to Miller [26], model interpretability means "the degree to which a human can understand the cause of a decision". Another popular definition came from Doshi-Velez and Kim [27], where they define it as "the ability to explain or to present in understandable terms to a human". Based on these explanations *Interpretability* is more like the cause and effect relationship of inputs and outputs of the model.

In contrast, *Explainability* is related to the internal logic and mechanics that are inside a machine learning algorithm. If the model is explainable, the human can understand the behavior of the model in the training and decision-making process. An interpretable model does not necessarily explain the internal logic of the ML model. This explains that interpretability does not exactly entail explainability, but an explainable model is always interpretable [28]. Therefore, it is necessary to address both interpretability and explainability aspects to better understand the ML model's logical behavior based on its inputs and outputs.

Regardless of this debate, since many studies have used these terms interchangeably and this SLR wants to cover all the related work, in both literature collection and analysis phases in this paper, we have ignored the differences between these two terms.

2.2. Objectives of XAI

Basically, explainability can be used to evaluate, justify, improve, or manage AI, or even learn from it. It is necessary to understand AI's risks and its failures by understanding the model's behavior.

In this section, we explain the main objectives of explainability in AI, in general (not limited to SE) so that later in the paper we can assess with aspects of generic XAI has been used and identified and useful in the SE domain. It's worth mentioning that these objectives are not separate concepts and may overlap in many aspects or be called with other names in different articles, but here, we gathered the most mentioned objectives in the literature.

An XAI method can aim to explain different aspects of an ML model. For instance, the practitioners may want to focus on the input data in an ML model to help them properly balance the training dataset. In another work, researchers may focus on the final output of the model to be able to provide a human-understandable explanation to the model's end user. In this section, we will go through some of these aspects of XAI and some suggested questions that can guide researchers' efforts, while focusing on these aspects.

2.2.1. Justification

One of the main objectives of explaining a model is to justify the model's decision in order to increase its credibility. This objective's aim is to gain the trust of users or people who are affected by the AI model. For this purpose, it's not necessary to explain different components or algorithms of a model, but it's required to find the connection between inputs and outputs [28].

To answer this need, it's important to know why an instance gives a specific output. It would also be helpful to elaborate on the feature(s) of the instance that determines the prediction or in some cases, to explore the reasons for getting the same output for two different inputs. Sometimes it's important to know how much change and in what direction for an instance is required or tolerable, in order to see a different output. Being able to answer such questions will make it easier for the users to trust the models.

2.2.2. Improvement

Improvement is one of the most important and strongest inspirations behind many explainability efforts [29]. Working with black-box complex models, it is a common problem that the model starts to make wrong decisions but the reasoning behind it is unclear to developers or researchers. The problem can be due to imbalanced training data or an inadequate internal part or overfitting of the model on specific features. Providing an explanation can be helpful in these situations.

2.2.3. Understanding

Understanding a model is another motivation that is mentioned in the literature. It's a very general term but that usually pairs up with other objectives or makes them possible. Understanding a model may lead to its improvement, test its fairness, or increase the level of user confidence.

2.2.4. Fairness

Fairness is a widely used term in the space of AI and ML. Hence, understanding the term fairness is important before discussing how fairness relates to Explainability of ML models. Broadly speaking, fairness is the quality or state of being fair or having impartial judgment. Fairness has been discussed in many disciplines such as law, social science, quantitative fields, philosophy etc. [30]

In machine learning, fairness can be pursued, in order to prevent biases and discriminations that may happen in different parts of an ML. It can appear in the training data where the dataset is imbalanced, in algorithms where for instance using specific optimization functions may lead to biased decision-making, or in the output representation where wrong conclusions are drawn from an observation [31].

XAI methods can significantly help researchers to consciously detect and eliminate biases and achieve fairness.

2.2.5. Transparency

Transparency is the opposite of black-boxness, that means when an ML model makes a decision, its whole process can be comprehended by a human [32]. Full transparency is usually impossible to fulfill, but it can be achieved in three levels that are: simulatable, decomposable, and algorithmic transparency [33]. Simulatable transparency means the model can be readily interpreted or simulated by a human, which means having a specific input, the user must be able to calculate the output having the model's parameters. Decomposable transparency is when smaller components of the model can be explained or explainable, separately. Finally, algorithmic transparent models are those whose training can get investigated mathematically.

2.3. Modality of Explanation

Model interpretability can be achieved if the model is understandable without further explanation [34]. Ac-

cording to the interpretability literature, the model understanding can be achieved by either considering inherently interpretable predictive models (i.e.; Linear regression, Random forest, Decision trees, etc.) or post-hoc interpretability approaches [35].

2.3.1. Intrinsically Interpretable

Intrinsically interpretable models are also called inherently interpretable models or self-explaining models. These models usually provide some level of transparency in their design and structure or they might generate additional outputs such as attention maps [36], disentangles representations, or textual or multi-model explanations alongside their predictions. Decision trees are one of the most famous self-explaining ML models. However, these inherently interpretable models often suffer from the accuracy-interpretability trade-off.

2.3.2. Post-hoc Interpretability

This approach approximates the black-box model using a transparent surrogate model without changing the base model. In this approach, the explained model is treated like a black-box and can be obtained even without any access to its internal components which is ideal when the model is too complex. However, post-hoc methods can also be applied intrinsicly and be model-specific [37]. Some of the most common post-hoc methods are LIME [38] and SHAP [39] which provide an explanation regarding defined features.

2.4. Scope of Explanation: Local and Global Explainability

Portability is an important aspect of post-hoc interpretability. It explains how far the explainer method is dependent on access to the internals of the global training model or the explained model.

Explainability methods are called *model-specific* or decompositional or white-box models, if the interpretability algorithm is restricted to explain only one specific family of models. In contrast, the algorithms which explain the output of different global models by considering the global model as a black-box, are called *model-agnostic* [40].

According to the present deployments in the space of interpretability, post-hoc interpretability can be further classified as global interpretability and local interpretability based on the scale of interpretation.

2.4.1. Global Interpretability

In order to explain the model output globally in a comprehensive way, the knowledge about the trained algorithm, trained model, and input data would be sufficient. The basic idea behind the holistic global interpretability is understanding how the model makes decisions based on the entire feature set, parameters, and structures. The outcomes of holistic-level interpretability are to help identify: (a) the features' importance levels, and (b) the feature interactions with each other.

2.4.2. Local Interpretability

When discussing local interpretability for a single prediction, it considers a single instance and argues that how the model's prediction for the particular instance corresponds to that instance's features. The prediction will be on a linear or monotonic relationship of some features in the local context. For example, the condition of all patients might not linearly depend on the patients' blood pressure. But, if we look at one specific patient and examine that patient's condition, while watching the small changes of his blood pressure, there is a higher chance of finding a linear relationship in the sub-region of our data. This is a reason to expect more accurate explanations from local explanations compared to global explanations.

3. Methodology

Our methodology for this review was mostly inspired by the guidelines proposed by Kitchenham for conducting an SLR [41]. The proposed method includes three main stages: planning, conducting the review, and reporting the results.

The planning phase comprises two steps. Identifying the need for a review which is discussed in Section 1 and developing a review protocol in order to remove the researchers' bias in the reviewing process. Our review protocol's main components include research questions, search strategy, paper selection, inclusion and exclusion criteria, and data extraction. This review protocol is recurringly modified and evolved during the process. Finally, the third step is basically reporting the results and the analysis.

The first two steps are discussed in the following section while the results are reported in Section 4 and analyzed in Section 5.

3.1. Research Questions

Regarding the purpose of this SLR which is investigating the utilization of XAI methods for ML/DL models in software engineering, we formulated the following RQs:

1. **RQ1: What are the main software engineering areas and tasks for which Explainable AI approaches have been used to date?**
   (a) What are the main software engineering areas and tasks for which XAI has been applied?
   (b) What are the ML4SE works that offer explainability?
   (c) What are the objectives of explainability in each research paper reviewed?
2. **RQ2: What are the Explainable AI approaches adopted for SE tasks?**
   (a) What kind of XAI techniques have been used?
   (b) What kind of explanations do they offer?
3. **RQ3: How useful XAI has been for SE?**
   (a) What are the objectives of applying XAI on SE and how they have been useful?
   (b) What are the means of evaluating XAI in SE?

Each of these RQs are aimed to help us shed some light on different areas of our review. RQ1 is focusing on SE tasks that took advantage of any kind of XAI method to explain their models. While RQ2 is concentrated on the XAI methods and how they have been utilized. Finaly, RQ3 cares about how successful XAI has been in doing its specified task. Each of these RQs also has sub-questions that together, form the answer to their respective main question.

3.2. Search Strategy

The search strategy that is used in this work, starts by choosing the digital libraries to be searched. To perform the primary search phase, five main scientific publication databases in software engineering and AI have been selected, i.e., ACM Digital Library, IEEE, Springer, Wiley Online Library, and Elsevier.

We considered three main categories of "Explainable AI", "Software Engineering", and "Artificial Intelligence" to create our search strings. By integrating alternative terms and synonyms for each of them, we gathered a list of keywords in three categories as shown in Table 1.

Using the boolean operators AND and OR, we formulated a search string that is: ("Interpretable" OR "Interpretability" OR "Explainability" OR "local model" OR "global model" OR "model-agnostic" OR "explainer model" OR "explanations" OR "black-box models" OR "rule-based" OR "counterfactual" OR "causal") AND ("SQA" OR "Software Analytics" OR "defect prediction" OR "API recommendation" OR "risk prediction" OR "code comprehension" OR "code generation" OR "code retrieval" OR "code summarization" OR "bug-fixing processes" OR "software testing" OR "effort estimation" OR "documentation generation" OR "clone detection" OR "program synthesis" OR "image to code" OR "security" OR "program repair" OR "vulnerability detection" OR "intrusion detection system" OR "Malware detection") AND ("Machine learning" OR "classification" OR "neural networks" OR "neural networks" OR "image processing" OR "text mining" OR "text analysis" OR "classification" OR "clustering" OR "rule mining" OR "association rules" OR "NLP" OR "Natural Language Processing" OR "embedding" OR "transformer" OR "adversarial learning")

The OR operators will check if any of the search terms in one category exist in the paper and the AND operator will make sure that all three categories are found in the papers.

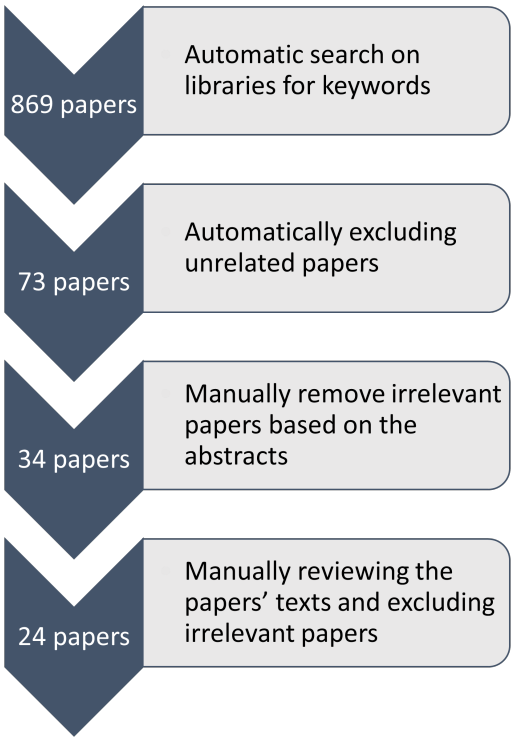

Figure 1: An overview of the searching strategy and its different stages with the number of studies in each step.

3.3. Study Selection/ Inclusion and Exclusion Criteria

First, we started by searching a smaller version of the alternative search terms (more generic terms) on the full documents of papers in all of the specified databases with no time limit for the papers on the date

Table 1: Key search terms and their respective alternative search terms.

Key Search Terms	Alternative Search Terms
Explainable AI	Interpretable, Interpretability, Explainability, local model, global model, model-agnostic, explainer model, explanations, interpretability algorithm, black-box models, rule-based, counterfactual, causal
Software Engineering	SQA, Software Analytics, defect prediction, security, vulnerability detection, intrusion detection system, API recommendation, risk prediction, code comprehension, code generation, code retrieval, code summarization, bug-fixing processes, software testing, effort estimation, documentation generation, clone detection, program synthesis, image to code, security, program repair
Artificial Intelligence	Machine learning, classification, neural networks, deep learning, image processing, text mining, text analysis, classification, clustering, rule mining, association rules, NLP, Natural Language Processing, embedding, transformer, adversarial learning

"06/textbackslash21/2022". This ended up with hundreds of irrelevant papers as the result. So it was decided to limit the search to the title, abstract, and keywords of the papers. Given that the terms used in title, abstract, and keywords are more specific and not very generic (i.e., a generic word such as "software engineering" might not be used in the title, abstract, and keywords but the specific task e.g., "defect prediction" will be.), we augmented the search terms in the new search to the current list to make sure we do not miss any relevant paper. To make sure that our findings are not biased to a limited set of specific terms, we augmented the list iteratively (4 iterations), by manually checking the citations and references of the collected papers and updating the list to cover all relevant papers. At this point, we could find a total of 869 papers which many of them were unrelated to SE and belonged to Health, Physiology, Medicine, or security. We used the search engine tools to filter the papers of unrelated fields that left us with 73 papers.

Lots of irrelevant papers were still among the search results so in order to find and select the relevant papers, we manually reviewed the abstracts of all the 73 papers (this job was divided between four authors. Then one author validated all decisions) and included or excluded papers based on the following criteria:

- Inclusion
  - Full text papers published as journal or conference papers that comply with Interpretability and explainability definition provided in Section 2.
  - Papers that are written in English Language.
  - Papers that are related to software engineering.
  - Papers that are available in full text.
- Exclusion
  - Books, gray literature (theses, unpublished and incomplete work), posters, secondary or review studies (SLR or SMS), surveys, discussions and keynotes are excluded.
  - Short papers where page count is less than 4 pages.
  - Papers about Software Engineering but not discussing about interpretability and explainability.
  - Papers about interpretability and explainability but not applying it on Software Engineering tasks.

So if a paper meets all of the defined inclusion criteria and also does not meet any of the exclusion criteria, we included it in the final papers list. Finally according to all of these filters and criteria, we were able to extract 25 papers that totally matched our interests. One paper [1] was discarded due to the problem of accessing its full

[1] https://ieeexplore.ieee.org/document/1374186

version in the digital libraries. So there are 24 papers remained, which can be seen in Table .6.

3.4. Data Extraction

For the data extraction phase, we defined a checklist of items that could help us extract the required information from each paper to answer the RQs. The checklist includes both quantitative and qualitative properties. The 24 papers were divided between four authors and one author verified and consolidated the results. We defined 17 main properties that could help us answer the RQs as below:

1. Publication details (Title, Authors, Published Date, Venue, Authors' affiliation)
2. What is the aim/ motivation/ goal of the research?
3. What are the key research problems addressed?
4. What phases of the SE are considered in the study?
5. What SE tasks are under experiment in the study? (For papers that perform experiments and report results)
6. Is the study conducted based on open-source data or data from industry?
7. What are the ML/DL models and techniques considered in the study and how they have been evaluated?
8. What are the XAI methods and techniques used in the study and how they are evaluated or represented (or visualized)?
9. What is the scope and modality of the explanation offered in the study?
10. What is the replication status of the experiments in the study?
11. The number of participants used in the study where human evaluation was involved?
12. What are the strengths and limitations of the study?
13. What are the key research gaps/ future work identified by each study?
14. Is the explainability among the main focuses of the study or just a side-benefit

3.5. Data Synthesis

Our data synthesis focused on summarizing and presenting the extracted data to convey meaningful information regarding the papers to address the RQs. In Section 4, we present all of our findings and results using tables, pie charts, bar charts, etc. in order to share insightful information from our analysis and studies.

Table 2: Authors with more than one contribution in the selected studies.

Author	# of papers
Tantithamthavorn, Chakkrit	5
Jiarpakdee, Jirayus	3
Dam, Hoa Khanh	2
Pornprasit, Chanathip	2
Thongtanunam, Patanamon	2
Matsumoto, Kenichi	2

4. Results

the results of the final selected papers after the search process and the meta-analysis on their publication places and years.

4.1. Selected Primary Studies

After filtering out or adding different research to our list of papers in multiple steps, finally we came up with a list of 24 papers that totally matched our search criteria. In Table.6, the paper's title, name of authors, year and source of publication, and publication type (journal or conference) of all these final selected studies are summarized.

4.2. Publication Statistics

As shown in Figure2, the first research in our selected studies goes back to 2008, published in Software Quality Journal. This study uses k nearest neighbor (kNN) and classification and regression trees generated based on the CART algorithm to perform the software effort prediction. Even though this study does not directly mention the XAI concept, it touches upon the explainability aspect in an implicit manner by being able to find the dominant predictor variables and how they're statistically significant using Anova method. After that, there is a small number of XAI works in SE each year but then from 2019, we can see a trending pattern and popularity of related research.

In the selected studies, a total of 75 authors contributed, while six mentioned in Table 2 are the authors in more than one paper. Also in terms of conferences and journals, as shown in Table 3, only three journals and one conference had more than one paper in the selected studies, which confirms that this field is still a new domain in the SE community.

Another interesting observation is that before 2018, most of the publications were in AI conferences or journals that weren't specific to SE, while after that, all

Table 3: Conferences [C] and journals [J] and number of papers published in each venue. The full name of each venue can be found in Table .6

Conference/Journal	# of papers
RAISE [C]	2
MSR [C]	2
Transactions on Software Engineering [J]	2
Software Quality Journal [J]	2
Empirical Software Engineering [J]	2
Symposium on Cloud Computing [C]	1
TOSEM [J]	1
IUI [C]	1
ICSE-NIER [C]	1
Symposium on Applied Computing [C]	1
ICSE-Companion [C]	1
ESEC/FSE [C]	1
RE [C]	1
ICMLA [C]	1
ASE [C]	1
ICECS [C]	1
ISSRE [C]	1
DSA [C]	1
ICTAI [C]	1

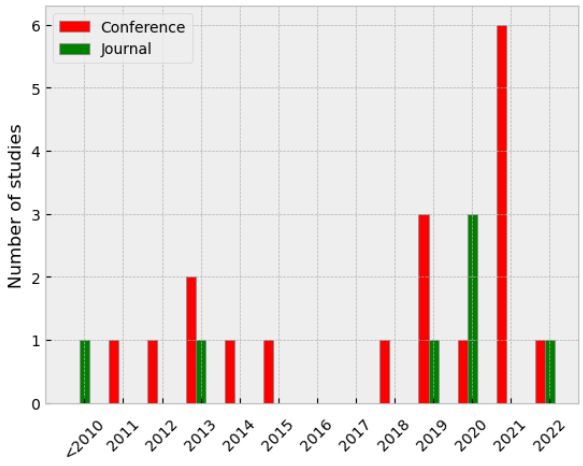

Figure 2: Distribution of selected studies over years and types of venues.

of the publications belong to SE-specific venues which shows the rising interest of the SE community in the field.

5. Synthesis and Analysis

5.1. RQ1 Results: What are the main software engineering areas and tasks for which Explainable AI approaches have been used to date?

In our selected studies, we surveyed different SE activities and SE tasks that XAI methods and studies have been focused on or utilized. We organized the tasks and activities based on the Software Development Life Cycle (SDLC) stages [42] and in this section, we are going to present and analyze the results to answer RQ1 of this study.

5.1.1. RQ1.1 results: What are the main software engineering areas and tasks for which XAI has been applied?

The most attractive activity for the researchers was software maintenance with 68% of the tasks being used for interpretation as shown in Figure 3. After that, software development has the largest share with 16%, and software management and software requirements each with two tasks in all of the studies have 8%. It's noteworthy that among the six stages of SDLC, software design and testing are two activities that have no presence in all of our selected studies.

A particularly noteworthy finding is the high ratio of the defect prediction task compared to all other tasks in the studies. This task alone has the lead with 44% of the

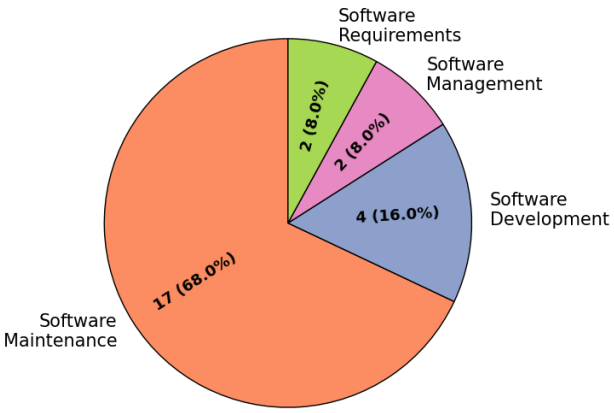

Figure 3: Distribution of studies per SDLC stage in the selected studies. Reported by #Papers (% proportions over all).

Table 4: The break-down of SDLC stages to tasks and their number of appearance in the selected studies.

Activity	Task	#
Requirement	Requirements engineering	1
	Requirements quality	1
Development	Code translation	1
	Code autocompletion	1
	Natural language to code	1
	Code Summarization	1
QA & Maintanance	Quality assessment	1
	Defect prediction	11
	Code reviews	1
	Reliability prediction	1
	Technical debt detection	1
	Valid bug report detection	1
	Variable misuse detection	1
Management	Effort Prediction	2

total number with a meaningful gap from the next task, which is effort prediction with only two studies and the other tasks each one being utilized only once. Also in general, QA and software maintenance tasks have the highest diversity with having seven different tasks. In Table4, we can see different tasks in each activity domain's broken-down.

Defect prediction is basically a classification problem to tag a code snippet, class, etc. as faulty or not faulty. This makes it a very ideal task for machine learning algorithms and also for XAI. It's also a very important task that ML models have shown promising results in.

We also examined a survey [42] on deep learning models for software engineering tasks and interestingly, we found 27 different tasks that already have taken advantage of black-box DL models, and yet, no explanation method has been used on them. Tasks such as "Code generation", "Code comment generation", "Test case generation", "Bug classification", etc. This can show the big gap between ML applications in SE and the utilization of XAI in the field.

Furthermore, we observed that among the reviewed studies that include an empirical evaluation, 16 papers only used open-source data, three papers only used publicly unavailable industrial data [43, 44, 45], while two papers used both [46, 47]. The data type in all the studies are either textual which can be source code or NL, or tabular which is a set of extracted features.

5.1.2. RQ1.2 Results: What are the ML4SE works that offer explainability?

To answer this question, we analyzed different ML models and the number of times that they have been used in the selected studies. As the sorted results show in Figure 4, while many models are rare and have only one use case, models like random forest and decision tree have been very popular with nine and seven uses. This is probably due to the fact that these classic models are usually being used as the baselines in many papers and also the fact that these models offer a level of self-explainability so it is expected to see them more in our selected studies.

The next popular models are regression models that are used for both regression problems (such as effort prediction) and classification problems (such as defect prediction). Regression models are used six times in total, while four of them are logistic regression, and linear and robust regression each have only one use-case.

Neural networks are also favored by having a total of 15 times being experimented, with adequate diversity among themselves. More complex code models like Code2Vec, Code2Seq, TransCode, and CodeBERT

can be found in the studies, as well as simpler networks like feed-forward neural networks, CNN, RNN, or basic transformer models are also among the explained models.

In Table 5 ML models are shown alongside the tasks they were used to solve. Models like the random forest or regression models are mostly utilized for defect prediction. The popularity of random forest for defect prediction is probably due to the fact that it's a self-explaining model which is a good fit for this specific task with high accuracy. Also, it is interesting to see the diversity of tasks that take advantage of the decision tree as a simple yet effective model.

DNN code models like Code2Vec, Code2Seq, Transcode, and CodeBert are always used for generation-based tasks such as code summarization or code translation which is understandable due to the complexity of these tasks and traditional models' incompatibility with them.

Regarding a large number of defect prediction studies, we also focused on those papers more specifically. Among the 11 works focused on this task, there is quite a diversity of ML models being used and even some studies cover different models for the purpose of their research. As mentioned, random forest is the most popular method that has been used in eight different papers either as the main model or as a benchmark to compare with. Logistic regression and deep learning neural networks are the second most popular models being used in five and three papers. There are other models that only have been used in two or one study. Models such as Support Vector Machine (SVM), k-Nearest Neighbours, and C5.0 Boosting. In these defect prediction tasks, standard evaluation metrics for ML models such as precision, recall, F1 score, and AUC are commonly used.

We also analyzed the SE tasks that these ML models have been applied to in terms of their ML problem type. As shown in Figure5, 18 tasks (more than 65% of the tasks experimented in the selected studies) fall into the classification problems. While only three tasks are considered regression problems (reliability prediction, requirements quality, and effort prediction), and four can be called generation problems (code translation, code autocompletion, natural language to code, and code summarization).

Classification tasks are among the simplest models to explain because of the fixed number of possible outputs that leads to a more straightforward evaluation, thus it makes sense for the researchers to focus on them. On the other hand, generation tasks are among the hardest since the output is usually harder to evaluate objectively. This gets more interesting when we see that three of those generation tasks are actually from one paper [48] that is doing a human evaluation of XAI methods for SE tasks. We will discuss the challenges of evaluation metrics in Section 5.3.2.

5.1.3. RQ1.3 Results: What are the objectives of explainability in each research paper reviewed?

In Section 2.2, we demonstrated different objectives of XAI. In our analysis of the selected studies of this work, we were interested to extract different proclaimed objectives of different studies. We found three main objectives explicitly or implicitly mentioned in the selected studies as: accountability, fairness, credibility, understanding, transparency, and improvement. Accountability and credibility have been used interchangeably in different works, meaning the reliability of the model for its users. Note that these objectives are not mutually exclusive and some papers follow multiple goals so the numbers add up to greater than the number of papers.

As shown in Figure 6, 20 papers that is the vast majority of studies have claimed improvement as (at least) one of their purposes, accountability in six, and understanding in three different papers have been mentioned. transparency and fairness are less common objectives while being the aim of two papers, and credibility has been mentioned only once.

Accountability and fairness are usually used in reviews/surveys [49, 48, 50] together, but the other objectives are coupled with the improvement in different research.

It's worth mentioning that while most of the objectives have been said with the same meaning as what is discussed in Section 2.2, improvement is an ambiguous term, used for multiple intentions. In the selected studies some researchers consider the explanation as it is, as the improvement [46]. In the other words, they believe that the fact that their model offers an explanation is an improvement compared to other models. Tree-based models that extract rules for their decision-making and present them are among this category.

Meanwhile, some studies have provided model-agnostic XAI tools and methods to help other researchers to understand and improve their models in the future [51] or even have used XAI methods to literally improve the results of models, especially in defect prediction. We will discuss them later in Section 5.3.1 where we discuss the helpfulness of XAI for SE models.

Table 5: Different ML models and the tasks they have been used for and number of uses

ML model	Task (# of Uses)	Total # of uses
Random Forest	Defect prediction (8) , Code reviews	9
Decision Tree	Effort prediction, defect prediction (2), Quality assessment, Requirements quality, Requirements engineering, Code reviews	7
Regression Model	Defect prediction (5), Requirements engineering	6
TransCode	Code translation, Code autocompletion, Natural language to code	3
CodeBERT	Code translation, Code autocompletion, Natural language to code	3
Feed Forward Network	Defect Prediction (2), Quality assessment	3
Naïve Bayes	Defect prediction (2)	2
kNN	Defect Prediction, Requirements engineering	2
SVM	Quality assessment, Defect Prediction	2
CNN	self-admitted technical debt detection, Valid Bug Report	2
Bayesian Network	Quality assessment, Defect Prediction	2
RNN	Variable Misuse detection	1
Fuzzy Logic	Software Reliability Prediction	1
Transformer	Variable Misuse detection, Defect prediction	1
Code2Vec	Code Summarization	1
Code2Seq	Code Summarization	1
Fuzzy C Means	Defect Prediction	1
Kurtanovic and Maalej's Classification	Requirements Classification	1
Gradient Boosting	Defect prediction	1
Genetic Algorithms	Software Reliability Prediction	1
Rule-based	Quality assessment	1
Case-based Reasoning	Quality assessment	1

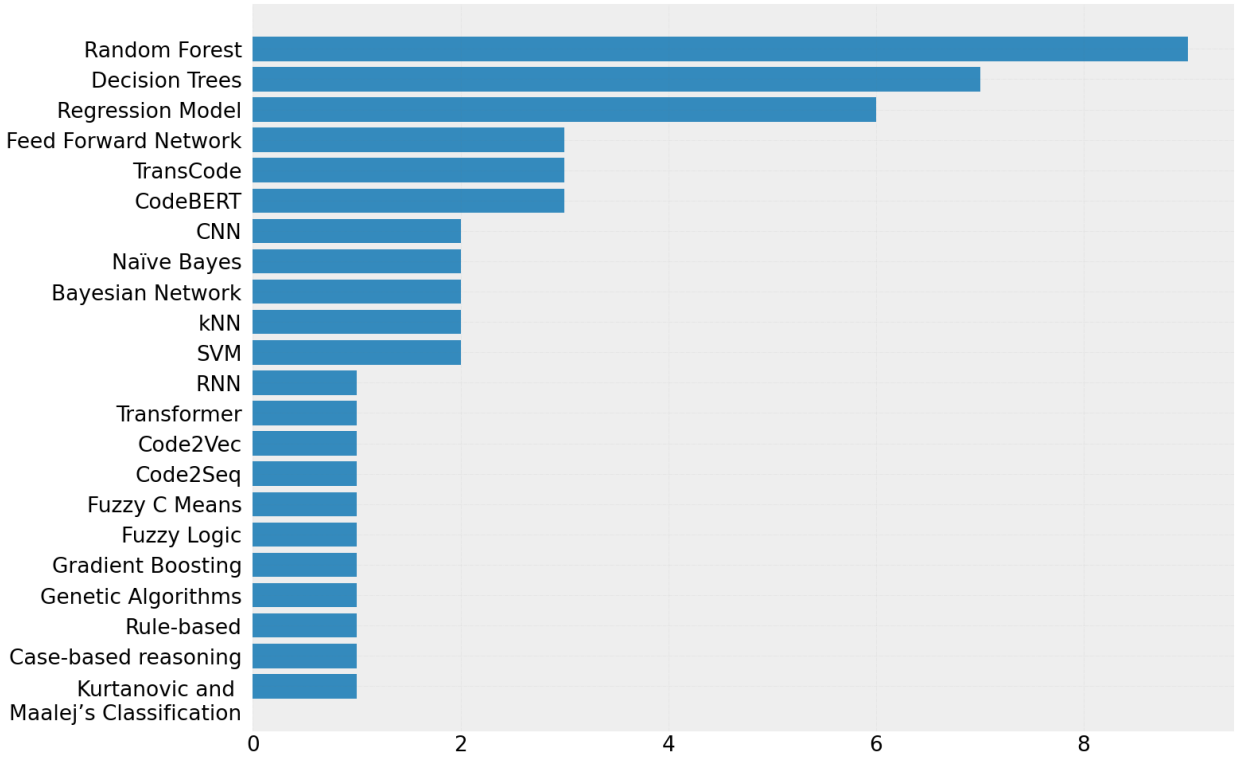

Figure 4: Number of experiments each ML model have been used in the selected studies

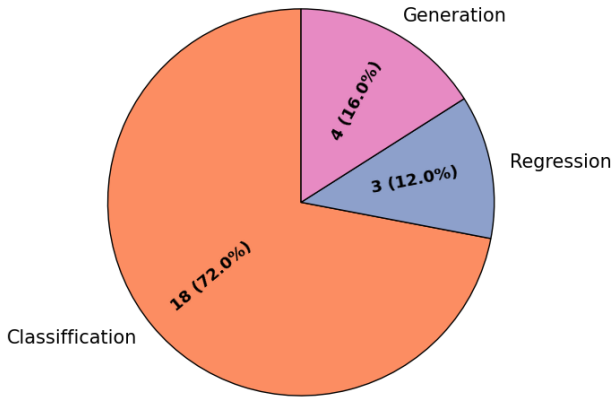

Figure 5: Distribution of SE task types in selected studies. Reported by #Papers (% proportions over all).

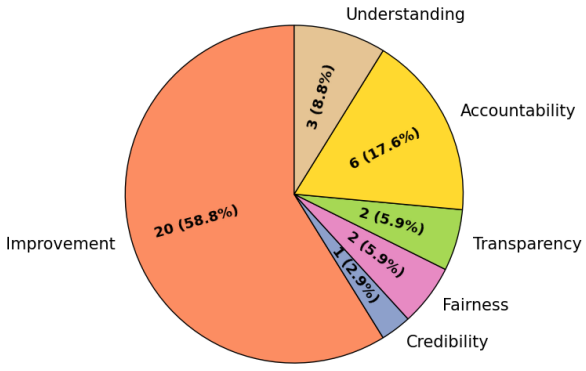

Figure 6: The XAI objectives stated (explicitly or implicitly) in the selected studies along with the number of papers that have mentioned them. Reported by #Papers (% proportions over all).

Answer to RQ1: *Among different stages of SDLC, QA and maintenance have been the most popular among the XAI researchers, and this popularity is hugely focused on the defect prediction task. This is likely due to 1) the popularity of this task among researchers and also 2) the adequacy of self-explaining traditional models like decision trees for these tasks.*
Our analysis also shows that, due to their inherent interpretability, classic models, like random forest, decision tree, regression models, and naive Bayes models, are among the most common ML models in the selected studies. DNN models are also another center of attention in terms of ML models, however generation-based tasks that are one of the main strengths of these models are still unexplored.

5.2. RQ2 Results: What are the Explainable AI approaches adopted for SE tasks?

In this section, we focus on the XAI methods that have been used to explain the SE models that we discussed in previous sections. This will include post-hoc XAI methods and self-explaining models.

5.2.1. RQ2.1 Results: What kind of XAI techniques have been used?

As we discussed in Section 2.2.5, there are two types of XAI approaches: (a) models that offer some level of explanation alongside the output, and (b) post-hoc XAI techniques that are applied on the model afterwards. In this section, we are discuss both cases among our selected studies. According to our analysis, 15 self-explaining models were used in the studies that need no or a small amount of effort to interpret while only 9 models were using post-hoc methods.

Tree-based models are the most popular self-explaining models among these studies with using a diverse set of algorithms. Decision trees using the C4.5 algorithm or its developed version, the PART algorithm that uses fuzzy logic to handle the classification task [45, 52, 53, 44, 54] or random forests are widely used in our primary selected studies [43, 55, 56, 57, 58].

The explanation offered in tree-based models is usually a set of rules or highlighted features that are quite self-explanatory for the user and usually need no further processing. Thus, the explanation in these models are either in form of feature summary statistics, or feature summary visualization using plots and graphs.

Deep neural networks are another observed type of ML models that even though are known as black-box models, but some researchers believe they have internal features that can be used for interpretation, after proper processing. For instance, [59] has used backtracking of a trained CNN model to extract the key phrases of the input and discover meaningful patterns in the code comments. As an explanation, they were able to find some 1-gram, 2-gram, and 3-gram patterns in natural language format that are compared to human-extracted patterns. Their model is able to cover most of the benchmark and also offers further comprehensive patterns with more diversity of vocabulary. In another study [60], the authors also use backtracking of a CNN model but for the valid bug report determination. As the input of this task is in NL format, it is very similar to classification tasks in NLP and it generates a natural language explanation.

Transformer models are also among the promising DNN models that have outperformed state-of-the-art models in some SE tasks by leveraging the attention mechanism. There is a controversial debate about the validation of transformer model's attention mechanism as an explanation method, in recent years. Among the selected studies, there is one that have used the self-attention mechanism of a transformer as an explanation [61]. They claim to find the importance of the input tokens by constructing the attention matrices. The generated matrix as well as a respective saliency map is the explanation that they present, but no evaluation is discussed.

Looking at the post-hoc XAI methods, most techniques that are used in SE are adopted from the NLP domain. LIME (Local Interpretable Model-Agnostic) [38] is one of the widely used XAI methods in NLP that also showed very useful in SE. In the reviewed papers, four different articles have used LIME directly in their studies [55, 56, 57, 58]. The method works based on feeding a black-box model with perturbed input data and observing the changes that happen on the output weights. This straightforward mechanism makes it ideal for NL and PL data. Perturbation can be defined on multiple levels from files in a commit to tokens in a code snippet and the results will be scores for the defined features (lines, tokens, etc.).

As we mentioned, LIME works based on input perturbation by generating synthetic instances(n) similar to a test instance (x) by a random perturbation method. The level and granularity of this perturbation depend on the task and objectives. In two of the studies that use LIME [55, 58], code tokens are considered as features that are supposed to be perturbed, while another study [57] uses tabular features of a code commit made by the developers (e.g. number of files modified, num-

ber of added or deleted lines, Number of developers who have modified the files, etc.). After the generation of synthetic instances, a defect prediction model is used to label them, and then, based on the predictions, a score between -1 to 1 is assigned to each feature. A positive score means a positive impact on the prediction and vice versa.

This model-agnostic and easy mechanism has led to the popularity of LIME among users, as another study, which is focused on the practitioners' perceptions of XAI for defect prediction models [56], claims that LIME gets the highest appeal among different XAI methods in terms of information usefulness, information, insightfulness, and information quality with more than 66% agreement rate.

The second favorite method is ANOVA (ANalysis Of VAriance), which is a statistical method that can be used to measure the importance of factors in a model. It does this by calculating the improvement in the Residual Sum of Squares (RSS) when a feature is added to the model. In our selected primary studies, This model-agnostic technique is used three times [56, 62, 63]. In addition, LIME and ANOVA have been selected as the first and second most preferred XAI methods, respectively, in a human evaluation of 50 software practitioners.

SIVAND is another model-agnostic XAI method that works based on Delta Debugging and reducing the input size, without changing the output [64]. The big picture is very similar to LIME but instead of just finding the most important atomic unit (token or character), it removes redundant units so the input gets smaller in size, but the prediction remains the same. They have tested their method on Code2Vec, Code2Seq, RNN, and Transformer models on two tasks of Method Name Prediction and Variable Misuse Detection and have a qualitative example-based evaluation of their models. For the Transformer model, they validate their performance by comparing the similarity of SIVAND and attention scores.

PyExplainer is another XAI method inspired by LIME and focused on Just In Time (JIT) defect prediction. It is a model-agnostic local rule-based technique that challenges the LIME's random perturbation mechanism by generating better synthetic neighbors, and explanation more unique and consistent. According to the mentioned criteria, PyExplainer outperforms LIME on Random Forest and Logistic Regression models for defect prediction.

Beside these model-agnostic methods, there are some other XAI techniques that are used for interpreting specific ML models. For instance, deconvolution is a method that exploits the structure of CNNs to extract the importance of features[59]. Flexible Code Counter/Classifier(CCFlex) is also an interesting model that was originally designed to classify lines of code but is used for code review and finding guideline violations[65].

Some other XAI methods are also mentioned in a study asking about software practitioners' perceptions of the goals of XAI methods for defect prediction models[56]. As mentioned earlier, LIME and ANOVA were the most preferred methods, while other techniques like Variable Importance, Partial Dependence, Decision Tree, BreakDown, SHAP, and Anchor were less favorable.

5.2.2. RQ2.2 Results: What kind of explanations do they offer?

We analyzed the selected studies in terms of the type of explanation that they offer. Following the previous discussion about self-explaining models, many models deliver "Model internals" as the explanation. Extracted rules and structures of tree-based models or the attention matrix of Transformer models are in this category. However, some works go further and more than the local explanations, presenting the "Feature summary statistic" or "Feature summary visualization" of their explanations with different visualization methods. The distribution of each type is shown in Figure 7. As it can be understood from the plot, some models have one, while others offer more types of explanation.

In the category of "Feature summary visualization", there are multiple XAI methods presented in the studies. While there are works that are satisfied by "Raw declarative representations" of their interpretations, many XAI methods implement more informative ways. "Natural language explanation" is used five times, while "Saliency maps" and "Plots and charts" each are used three times. There are also two methods that present their results in interactive tools to the users. The distribution of different visualization techniques in the studies is illustrated in Figure8.

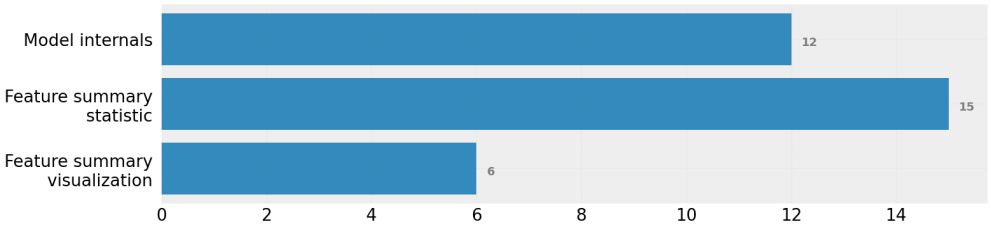

Figure 7: Different types of explanation offered in the selected studies.

Answer to RQ2: *Our investigation shows that beside different ML models that have a level of interpretebility and are quite common for SE tasks (such as random forest and decision trees), there is a variety of model-agnostic XAI methods that are used in SE to explain black-box models. LIME which is a perturbation XAI method is the most favorite method in the selected studies, while ANOVA is the second most-used method. There are also some other proposed methods used in the studies such as deconvolutioning for CNN models, or SIVAND which has similarities with LIME.*
The explanations offered by XAI methods can get categorized as "Feature summary statistic", "Model internals", or "Feature summary visualization" that have been used in the studies, respectively in descending order.

5.3. RQ3 Results: How useful XAI has been for SE?

In this section, we are going to focus on the helpfulness of XAI methods that are used for SE tasks.

5.3.1. RQ3.1 Results: What are the objectives of applying XAI on SE and how they have been useful?

In Section 5.1.3, we discussed different objectives of research in our selected studies. Some of them have used explanation as an additional output for their users, while a few others used the XAI methods to improve the performance of the models.

One interesting representation of these improvements can be seen in defect prediction. While many defect prediction models only classify if a file or a class is faulty, some researchers have taken advantage of XAI methods to specifically find the buggy line or tokens of the code [55, 56, 58]. This is interesting since it improves defect prediction's practicality and its potential for adoption by industry.

SIVAND is another successful XAI method, where its finding improves the performance of the code models it is applied to, and also offers valuable knowledge about them [64]. They are able to considerably reduce the size of the inputs, and yet achieve the same accuracy. This means a great reduction in the models' required time to make a prediction. Furthermore, they offer interesting knowledge about code models by claiming the presence of alleged patterns in them. For instance, they believe while Transformer models can understand more "global" information in codes, RNN models are prone to local information. They also believe these models are quite vulnerable to perturbations as they rely on too few features.

Another noteworthy and exemplary use of XAI can be seen in another work [60] where by backtracking a trained CNN network and manually inspecting the explanation, the authors are able to find valid bug report patterns. Further analysis of the identified bug reports, they verify the importance and effect of XAI on their model and the problem.

5.3.2. RQ3.2 Results: What are the means of evaluating XAI in SE?

Looking at the evaluation of the XAI methods (not the ML models) across the selected papers, we can see that the struggle is apparent. While studying these selected papers, we observed that there is little consistency among selected papers in terms of evaluation. Many studies have totally ignored the evaluation phase and only offer some visualization for their XAI with no evidence of its quality or reliability.

Most existing XAI evaluation methods [66], [59], [65] are qualitative and use human subjects to assess the explanations. The problem with this approach, specially with small scale subject pools, is that they are heavily biased and may not be generalizable to other users. In an attempt to find out how thoroughly evaluate XAI in SE, in one work [48] the authors asked 43 software engineers in 9 workshops to express their explainability needs for some specific tasks.

With respect to qualitative assessment of XAI methods in SE, there are some works that based on their task

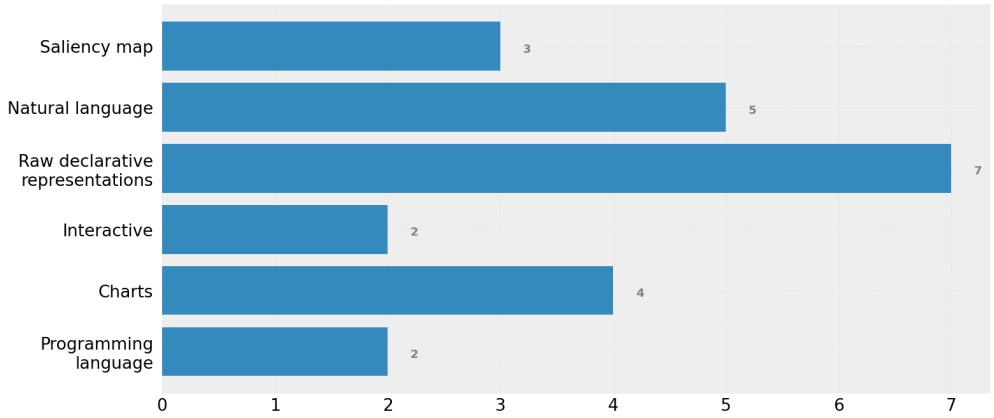

Figure 8: Distribution of different visualizations used as explanation methods in the selected studies.

(e.g., defect predicion), are able to define more common metrics such as Mean Squared Root (MSR), or Wald test [62]. In a few other studies, researchers defined innovative metrics or used some less-known metrics from generic the XAI domain. For instance, in one work [51] the authors measure the percentage of unique explanations generated by XAI method as a metric. Another work [52] utilizes the average length of the rules that their XAI extracts and another study [57] uses Variable Stability Index (VSI) and Coefficient Stability Index (CSI), as metrics.

Answer to RQ3: *Our findings show that some XAI methods were quite helpful and were applied either for providing proper explanations to researchers that led to finding interesting patterns in the model's decision-making, or for increasing the granularity of models' results, leading to the higher precision for the respective SE task.*
We also noticed some confusion and a lack of standard routines among the researchers in terms of evaluating XAI methods that are usually compensated by defining innovative and task-specific metrics or using qualitative evaluation by a human.

6. Challenges and road ahead

In general, there are valuable works that have been published in the field of XAI for SE. Interesting ideas have been suggested, and an encouraging growing trend has started. However, despite all of these efforts, there is a long way ahead. ML models are improving their performance and growing more complex gradually and yet there is a noticeable lack of interest from software practitioners in using them in the real world, partly due to the lack of transparency and trust issues.

As we mentioned in Section 5.1.1 there are still unexplored areas such as software design and testing, and tasks in SE that already have taken advantage of ML models and yet have not been studied in terms of explainability. Code comment generation, code generation, vulnerability detection, test case generation, bug classification, and program repair are some of the most important tasks that require more attention.

Additionally, if we consider the mentioned unexplored tasks, we can see a pattern of overlooking generation-based tasks (e.g. code generation, test case generation, and program repair). In our analysis, we discussed how classification problems are preferable for XAI due to their limited output space, but looking at recent advancements of AI models in generation-based tasks, we believe there is an undeniable need for XAI methods for those tasks.

A further note on this topic is these unexplored tasks and in general, all generation-based tasks is that their advancements rely heavily on deep neural networks. DNNs have been a great asset for researchers when it comes to searching in a huge space (i.e. vocabulary of words), thus they are ideal for this type of task. Code models and their success in code document generation, code translation, and code repair are good examples. However, they are incomparably more complex than classical ML models, so they are harder to explain. Nonetheless, this complexity is the exact reason that they require explanation.

Furthermore, by analyzing the XAI techniques used in SE models, we see that the generated explanations are mostly meant for developers and programmers and not other types of less-technical end users (e.g., managers,

executives, and other stakeholders) as there are fewer high-level explanations (such as visualizations or natural language rules) offered. While in other areas such as Image processing or NLP, the explanations are more user-friendly. This problem is partly because of the type of data and nature of SE tasks, but still, a lack of effort in giving such visualizations is noticeable.

Finally, if we look at our selected studies, we can see a clear lack of evaluation consistency among them. Some studies just explain the ML models for the sake of explaining and pay no more interest in further analysis, evaluation, or justification for the provided explanations. Considering the fact that explanations are supposed to help the models' users, we believe when objective evaluation is not possible or sufficient, trying human evaluation by experts, is a satisfactory solution.

7. Related Works

There have been lots of surveys and literature reviews addressing the XAI topic in general or for specific fields. For instance, a study surveys some traditional ML models such as SVM, linear, and tree-based models and notes a trade-off between their performance and reliability [67]. They also mention the ambiguity and confusion among works addressing XAI in terms of taxonomy and definitions.

In an attempt to standardize the efforts in XAI, some studies has tried to clarify some of the definitions and terminology and insisted in distinguishing the terms "interpretability" and "explainability" as different terms with the first only being one of the features of the latter [28]. They also claim that different approaches to solve the problem of black-box models, usually are unilateral and fail to address different aspects of explainability. A similar approach is taken by others that offer a more updated insight and also specify challenges and opportunities ahead[68].

Besides works that surveyed or reviewed the literature on XAI, in general, there are valuable attempts to particularly review the state of XAI in specific fields or models. As expected, medical applications that have taken great advantage of ML models, in recent years, are also among the most critical and sensitive domains. In a research, more than 200 papers that have used XAI in deep learning models for medical image processing are analyzed, which itself shows the high interest of researchers in the field [69]. They also provide some suggestions for future works including the need for medical experts to actively participate in designing interpretation methods.

Natural language processing is also one of the areas that benefitted from surveys and reviews to examine the state of XAI in the literature. For instance, A survey examines 50 papers that are related to XAI in NLP and categorizes and analyzes them according to different aspects like locality and globality or being self-explaining or post-hoc [35]. Furthermore, they note a debate among researchers about the very existence of a trade-off between explainability and performance. Similar to many other surveys in other fields, they also mention the lack of clear terminology and understanding of XAI methods and provide insightful guidelines.

There are also reviews or surveys that instead of tasks or fields of study focus on different types of data or models. Among these works, there are studies that focus on time-series data specifically [70], or analyze the advances in XAI for tabular data [71]. There are other researches that study different XAI methods for deep neural networks [72] or specifically reviews XAI in deep reinforcement learning models [73].

Software engineering is also a field that has benefited from ML models for a long time and SLRs have been conducted to analyze the literature for ML or DL techniques application in SE [42, 74]. However, to the best of our knowledge, this is the first work offering a systematic literature review for XAI in software engineering.

8. Conclusion

To the best of our knowledge, this is the first study to systematically review XAI in the domain of SE. By reviewing 24 relevant papers (out of over 800 automatically selected ones) and analyzing the collected data, we report the current state of research in this area and identify challenges and road ahead for future research.

In particular, the review shows that among different stages of SDLC, QA and maintenance have been the most popular among the XAI researchers, and this popularity is hugely focused on the defect prediction task. On the other hand generation-based tasks (e.g. code generation, test case generation, and program repair) which are popular in the ML4SE community are rarely studied in the XAI4SE literature.

The data also shows that the number one impact among all targeted objectives of applying XAI to ML4SE models has been improving the original ML model.

We also observe that self-explainable ML models, such as classic random forest, decision tree, regression models, and naive Bayes are among the most used XAI

techniques and post-hoc technique are less used in the community, so far.

In terms of the type of explanations, XAI4SE has been mainly focused on ML developers as the end user and provides low-level debugging type explanations such as "feature summary statistic" (e.g., Tables showing some extracted key features and their distribution [59]) and "model internals" (e.g., self-attention scores of Transformers [61] and backtracking CNN layers of a NN [60]) , and less on higher-level natural language-based explanations or visualization.

Finally, a lack of standard ways for evaluating XAI methods was clear among the studies, which has led to many different project-specific evaluation metrics, as well as human subject assessments.

References

[1] F. Doshi-Velez, Y. Ge, I. Kohane, Comorbidity clusters in autism spectrum disorders: an electronic health record time-series analysis, Pediatrics 133 (1) (2014) e54–e63.

[2] A. E. Khandani, A. J. Kim, A. W. Lo, Consumer credit-risk models via machine-learning algorithms, Journal of Banking & Finance 34 (11) (2010) 2767–2787.

[3] H. H. Le, J.-L. Viviani, Predicting bank failure: An improvement by implementing a machine-learning approach to classical financial ratios, Research in International Business and Finance 44 (2018) 16–25.

[4] A. Svyatkovskiy, S. K. Deng, S. Fu, N. Sundaresan, Intellicode compose: Code generation using transformer, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2020, pp. 1433–1443.

[5] J. Koo, C. Saumya, M. Kulkarni, S. Bagchi, Pyse: Automatic worst-case test generation by reinforcement learning, in: 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), IEEE, 2019, pp. 136–147.

[6] M. White, M. Tufano, M. Martinez, M. Monperrus, D. Poshyvanyk, Sorting and transforming program repair ingredients via deep learning code similarities, in: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2019, pp. 479–490.

[7] F. Ataiefard, M. J. Mashhadi, H. Hemmati, N. Walkinshaw, Deep state inference: Toward behavioral model inference of black-box software systems, IEEE Transactions on Software Engineering (2021).

[8] H. K. Dam, T. Tran, T. Pham, S. W. Ng, J. Grundy, A. Ghose, Automatic feature learning for predicting vulnerable software components, IEEE Transactions on Software Engineering 47 (1) (2018) 67–85.

[9] X. He, L. Xu, X. Zhang, R. Hao, Y. Feng, B. Xu, Pyart: Python api recommendation in real-time, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 1634–1645.

[10] A. S. Filippetto, R. Lima, J. L. V. Barbosa, A risk prediction model for software project management based on similarity analysis of context histories, Information and Software Technology 131 (2021) 106497.

[11] T. Ben-Nun, A. S. Jakobovits, T. Hoefler, Neural code comprehension: A learnable representation of code semantics, Advances in Neural Information Processing Systems 31 (2018).

[12] W. Gu, Z. Li, C. Gao, C. Wang, H. Zhang, Z. Xu, M. R. Lyu, Cradle: Deep code retrieval based on semantic dependency learning, Neural Networks 141 (2021) 385–394.

[13] Y. Shido, Y. Kobayashi, A. Yamamoto, A. Miyamoto, T. Matsumura, Automatic source code summarization with extended tree-lstm, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

[14] M. Tufano, C. Watson, G. Bavota, M. D. Penta, M. White, D. Poshyvanyk, An empirical study on learning bug-fixing patches in the wild via neural machine translation, ACM Transactions on Software Engineering and Methodology (TOSEM) 28 (4) (2019) 1–29.

[15] N. Miryeganeh, S. Hashtroudi, H. Hemmati, Globug: using global data in fault localization, Journal of Systems and Software 177 (2021) 110961.

[16] P. Phannachitta, On an optimal analogy-based software effort estimation, Information and Software Technology 125 (2020) 106330.

[17] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, et al., Codebert: A pre-trained model for programming and natural languages, arXiv preprint arXiv:2002.08155 (2020).

[18] D. Perez, S. Chiba, Cross-language clone detection by learning over abstract syntax trees, in: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), IEEE, 2019, pp. 518–528.

[19] E. C. Shin, M. Allamanis, M. Brockschmidt, A. Polozov, Program synthesis and semantic parsing with learned code idioms, Advances in Neural Information Processing Systems 32 (2019).

[20] B. Xu, A. Shirani, D. Lo, M. A. Alipour, Prediction of relatedness in stack overflow: deep learning vs. svm: a reproducibility study, in: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2018, pp. 1–10.

[21] M. J. Mashhadi, H. Hemmati, Hybrid deep neural networks to infer state models of black-box systems, in: 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2020, pp. 299–311.

[22] H. K. Dam, T. Tran, A. Ghose, Explainable software analytics, in: Proceedings of the 40th international conference on software engineering: New ideas and emerging results, 2018, pp. 53–56.

[23] J. Chakraborty, T. Xia, F. M. Fahid, T. Menzies, Software engineering for fairness: A case study with hyperparameter optimization, arXiv preprint arXiv:1905.05786 (2019).

[24] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, T. Zimmermann, Software engineering for machine learning: A case study, in: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE, 2019, pp. 291–300.

[25] D. Gunning, D. Aha, Darpa's explainable artificial intelligence (xai) program, AI magazine 40 (2) (2019) 44–58.

[26] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence 267 (2019) 1–38.

[27] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning, arXiv preprint arXiv:1702.08608 (2017).

[28] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: An overview of interpretability of machine learning, in: 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA), IEEE, 2018, pp. 80–89.

[29] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, J. Zhu, Explainable ai: A brief survey on history, research areas, approaches and challenges, in: CCF international conference on natural language processing and Chinese computing, Springer, 2019, pp.

563–574.

[30] D. K. Mulligan, J. A. Kroll, N. Kohli, R. Y. Wong, This thing called fairness: Disciplinary confusion realizing a value in technology, Proceedings of the ACM on Human-Computer Interaction 3 (CSCW) (2019) 1–36.

[31] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, ACM Computing Surveys (CSUR) 54 (6) (2021) 1–35.

[32] R. Roscher, B. Bohn, M. F. Duarte, J. Garcke, Explainable machine learning for scientific insights and discoveries, Ieee Access 8 (2020) 42200–42216.

[33] P. E. Love, W. Fang, J. Matthews, S. Porter, H. Luo, L. Ding, Explainable artificial intelligence: Precepts, methods, and opportunities for research in construction, arXiv preprint arXiv:2211.06579 (2022).

[34] J. E. Zini, M. Awad, On the explainability of natural language processing deep models, ACM Computing Surveys (CSUR) (2022).

[35] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, P. Sen, A survey of the state of explainable ai for natural language processing, arXiv preprint arXiv:2010.00711 (2020).

[36] A. H. Mohammadkhani, C. Tantithamthavorn, H. Hemmati, Explainable ai for pre-trained code models: What do they learn? when they do not work?, arXiv preprint arXiv:2211.12821 (2022).

[37] D. V. Carvalho, E. M. Pereira, J. S. Cardoso, Machine learning interpretability: A survey on methods and metrics, Electronics 8 (8) (2019) 832.

[38] M. T. Ribeiro, S. Singh, C. Guestrin, " why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[39] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).

[40] Z. C. Lipton, The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery., Queue 16 (3) (2018) 31–57.

[41] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele University 33 (2004) (2004) 1–26.

[42] Y. Yang, X. Xia, D. Lo, J. Grundy, A survey on deep learning for software engineering, ACM Computing Surveys (CSUR) 54 (10s) (2022) 1–73.

[43] T. Pushphavathi, V. Suma, V. Ramaswamy, A novel method for software defect prediction: hybrid of fcm and random forest, in: 2014 International Conference on Electronics and Communication Systems (ICECS), IEEE, 2014, pp. 1–5.

[44] V. Moreno, G. Génova, E. Parra, A. Fraga, Application of machine learning techniques to the flexible assessment and improvement of requirements quality, Software Quality Journal 28 (4) (2020) 1645–1674.

[45] M. P. Basgalupp, R. C. Barros, T. S. da Silva, A. C. de Carvalho, Software effort prediction: A hyper-heuristic decision-tree based approach, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, 2013, pp. 1109–1116.

[46] J. Jiarpakdee, Towards a more reliable interpretation of defect models, in: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, 2019, pp. 210–213.

[47] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, S. Çevikol, Requirements classification with interpretable machine learning and dependency parsing, in: 2019 IEEE 27th International Requirements Engineering Conference (RE), IEEE, 2019, pp. 142–152.

[48] J. Sun, Q. V. Liao, M. Muller, M. Agarwal, S. Houde, K. Talamadupula, J. D. Weisz, Investigating explainability of generative ai for code through scenario-based design, in: 27th International Conference on Intelligent User Interfaces, 2022, pp. 212–228.

[49] S. Suneja, Y. Zheng, Y. Zhuang, J. A. Laredo, A. Morari, Towards reliable ai for source code understanding, in: Proceedings of the ACM Symposium on Cloud Computing, 2021, pp. 403–411.

[50] H. K. Dam, T. Tran, A. Ghose, Explainable software analytics, in: Proceedings of the 40th international conference on software engineering: New ideas and emerging results, 2018, pp. 53–56.

[51] C. Pornprasit, C. Tantithamthavorn, J. Jiarpakdee, M. Fu, P. Thongtanunam, Pyexplainer: Explaining the predictions of just-in-time defect models, in: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2021, pp. 407–418.

[52] T. Diamantopoulos, A. Symeonidis, Towards interpretable defect-prone component analysis using genetic fuzzy systems, in: 2015 IEEE/ACM 4th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, IEEE, 2015, pp. 32–38.

[53] H. Lounis, T. F. Gayed, M. Boukadoum, Machine-learning models for software quality: a compromise between performance and intelligibility, in: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, IEEE, 2011, pp. 919–921.

[54] A. Okutan, O. T. Yıldız, Software defect prediction using bayesian networks, Empirical Software Engineering 19 (1) (2014) 154–181.

[55] C. Pornprasit, C. K. Tantithamthavorn, Jitline: A simpler, better, faster, finer-grained just-in-time defect prediction, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 369–379.

[56] J. Jiarpakdee, C. K. Tantithamthavorn, J. Grundy, Practitioners' perceptions of the goals and visual explanations of defect prediction models, in: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), IEEE, 2021, pp. 432–443.

[57] W. Zheng, T. Shen, X. Chen, Just-in-time defect prediction technology based on interpretability technology, in: 2021 8th International Conference on Dependable Systems and Their Applications (DSA), IEEE, 2021, pp. 78–89.

[58] S. Wattanakriengkrai, P. Thongtanunam, C. Tantithamthavorn, H. Hata, K. Matsumoto, Predicting defective lines using a model-agnostic technique, IEEE Transactions on Software Engineering (2020).

[59] X. Ren, Z. Xing, X. Xia, D. Lo, X. Wang, J. Grundy, Neural network-based detection of self-admitted technical debt: From performance to explainability, ACM transactions on software engineering and methodology (TOSEM) 28 (3) (2019) 1–45.

[60] J. He, L. Xu, Y. Fan, Z. Xu, M. Yan, Y. Lei, Deep learning based valid bug reports determination and explanation, in: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2020, pp. 184–194.

[61] J. Humphreys, H. K. Dam, An explainable deep model for defect prediction, in: 2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), IEEE, 2019, pp. 49–55.

[62] C. Tantithamthavorn, A. E. Hassan, K. Matsumoto, The impact of class rebalancing techniques on the performance and interpretation of defect prediction models, IEEE Transactions on Software Engineering 46 (11) (2018) 1200–1219.

[63] Q. Liu, W. Z. Qin, R. Mintram, M. Ross, Evaluation of preliminary data analysis framework in software cost estimation based on isbsg r9 data, Software quality journal 16 (3) (2008) 411–458.

[64] M. R. I. Rabin, V. J. Hellendoorn, M. A. Alipour, Understanding neural code intelligence through program simplification, in:

Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 441–452.

[65] M. Ochodek, R. Hebig, W. Meding, G. Frost, M. Staron, Recognizing lines of code violating company-specific coding guidelines using machine learning, in: Accelerating Digital Transformation, Springer, 2019, pp. 211–251.

[66] T. Mori, Superposed naive bayes for accurate and interpretable prediction, in: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), IEEE, 2015, pp. 1228–1233.

[67] F. K. Došilović, M. Brčić, N. Hlupić, Explainable artificial intelligence: A survey, in: 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO), IEEE, 2018, pp. 0210–0215.

[68] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information fusion 58 (2020) 82–115.

[69] B. H. van der Velden, H. J. Kuijf, K. G. Gilhuijs, M. A. Viergever, Explainable artificial intelligence (xai) in deep learning-based medical image analysis, Medical Image Analysis (2022) 102470.

[70] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, N. Díaz-Rodríguez, Explainable artificial intelligence (xai) on timeseries data: A survey, arXiv preprint arXiv:2104.00950 (2021).

[71] M. Sahakyan, Z. Aung, T. Rahwan, Explainable artificial intelligence for tabular data: A survey, IEEE Access 9 (2021) 135392–135422.

[72] G. Montavon, W. Samek, K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digital signal processing 73 (2018) 1–15.

[73] A. Heuillet, F. Couthouis, N. Díaz-Rodríguez, Explainability in deep reinforcement learning, Knowledge-Based Systems 214 (2021) 106685.

[74] C. Watson, N. Cooper, D. N. Palacio, K. Moran, D. Poshyvanyk, A systematic literature review on the use of deep learning in software engineering research, ACM Transactions on Software Engineering and Methodology (TOSEM) 31 (2) (2022) 1–58.

Table .6: List of the selected 24 papers. Conference: [C], Journal: [J]

Ref	Title	Authors	Publication Venue	Publication Year
[48]	Investigating Explainability of Generative AI for Code through Scenario-Based Design	J. Sun, Q. V. Liao, M. Muller, M. Agarwal, S. Houde, K. Talamadupula, J.D. Weisz	Intelligent User Interfaces (IUI) [C]	2022
[58]	Predicting Defective Lines Using a Model-Agnostic Technique	S. Wattanakriengkrai, P. Thongtanunam, C. Tantithamthavorn, H. Hata, K. Matsumoto	Transactions on Software Engineering [J]	2022
[55]	JITLine: A Simpler, Better, Faster, Finer-grained Just-In-Time Defect Prediction	C. Pornprasit, C. Tantithamthavorn	Mining Software Repositories (MSR) [C]	2021
[57]	Just-in-Time Defect Prediction Technology based on Interpretability Technology	W. Zheng, T. Shen, X. Chen	Dependable Systems and their Applications (DSA) [C]	2021
[56]	Practitioners' Perceptions of the Goals and Visual Explanations of Defect Prediction Models	J. Jiarpakdee, C. Tantithamthavorn, J. Grundy	Mining Software Repositories (MSR) [C]	2021
[51]	PyExplainer: Explaining the Predictions of Just-In-Time Defect Models	C. Pornprasit, C. Tantithamthavorn; J. Jiarpakdee, M. Fu, P. Thongtanunam	Automated Software Engineering (ASE) [C]	2021
[49]	Towards Reliable AI for Source Code Understanding	S. Suneja; Y. Zheng; Y. Zhuang; J. Laredo; A. Morari	Symposium on Cloud Computing [C]	2021
[64]	Understanding Neural Code Intelligence through Program Simplification	M. R. I. Rabin, V. J. Hellendoorn, M. A. Alipour	European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) [C]	2021
[44]	Application of machine learning techniques to the flexible assessment and improvement of requirements quality	V. Moreno, G. Génova, E. Parra, A. Fraga	Software Quality [J]	2020
[60]	Deep Learning Based Valid Bug Reports Determination and Explanation	J. He, L. Xu, Y. Fan, Z. Xu, M. Yan, Y. Lei	International Symposium on Software Reliability Engineering (ISSRE) [C]	2020
[65]	Recognizing lines of code violating company-specific coding guidelines using machine learning: A Method and Its Evaluation	M. Ochodek, R. Hebig, W. Meding, G. Frost, M. Staron	Empirical Software Engineering [J]	2020
[62]	The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models	C. Tantithamthavorn, A. E. Hassan, K. Matsumoto	Transactions on Software Engineering [J]	2020

[61]	An Explainable Deep Model for Defect Prediction	J. Humphreys, H. K. Dam	Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE) [C]	2019
[59]	Neural Network-Based Detection of Self-Admitted Technical Debt: From Performance to Explainability	X. Ren, Z. Xing, X. Xia, D. Lo, X. Wang, J. Grundy.	Transactions on Software Engineering and Methodology (TOSEM) [J]	2019
[47]	Requirements Classification with Interpretable Machine Learning and Dependency Parsing	F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, S. Çevikol	Requirements Engineering (RE) [C]	2019
[46]	Towards a More Reliable Interpretation of Defect Models	J. Jiarpakdee	International Conference on Software Engineering: Companion (ICSE-Companion) [C]	2019
[50]	Explainable Software Analytics	H. K. Dam, T. Tran, A. Ghose	International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER) [C]	2018
[66]	Superposed Naive Bayes for Accurate and Interpretable Prediction	T. Mori	International Conference on Machine Learning and Applications (ICMLA) [C]	2015
[43]	A novel method for software defect prediction: Hybrid of FCM and random forest	T.P. Pushphavathi, V. Suma, V. Ramaswamy	International Conference on Electronics and Communication Systems (ICECS) [C]	2014
[54]	Software defect prediction using Bayesian networks	A. Okutan, O. T. Yıldız	Empirical Software Engineering [J]	2014
[52]	Towards Interpretable Defect-Prone Component Analysis Using Genetic Fuzzy Systems	T. Diamantopoulos, A. Symeonidis	Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE) [C]	2014
[45]	Software Effort Prediction: A Hyper-Heuristic Decision-Tree Based Approach	M. P. Basgalupp, R. C. Barros, T. S. da Silva, A. C. P. L. F. de Carvalho	Symposium on Applied Computing [C]	2013
[53]	Machine-Learning Models for Software Quality: A Compromise between Performance and Intelligibility	H. Lounis, T. F. Gayed, M. Boukadoum	International Conference on Tools with Artificial Intelligence [C]	2011
[63]	Evaluation of preliminary data analysis framework in software cost estimation based on ISBSG R9 Data	Q. Liu, W. Z. Qin, R. Mintram, M. Ross	Software Quality [J]	2008