MONASH University

Explainable AI with the Use of Formal Reasoning

Jinqiang Yu

Doctor of Philosophy

A Thesis Submitted for the Degree of Doctor of Philosophy at
Monash University in 2024
Faculty of Information Technology

Copyright notice

Abstract

In recent years, machine learning (ML) and Artificial Intelligence (AI) techniques have experienced rapid advancements, reshaping numerous aspects of human lives. Owing to advancements in algorithms and improved computational capabilities, these impressive ML and AI approaches are adopted across a wide range of tasks, including those in the fields of natural language processing (NLP) and computer vision (CV). For instance, a significant breakthrough has been achieved with the development of large language models (LLMs) such as GPT-4, which have demonstrated exceptional performance in a variety of natural language processing (NLP) tasks.

Despite the considerable advancements in machine learning (ML) that have rapidly integrate complex ML models into our daily lives, they are considered "black-box" models, lacking transparency in the outputs they produce. Consequently, both domain experts and general users lack clear insights into outputs generated by these complex models, although they make decisions based on these outputs. As a result, there is a growing demand for transparency and accountability in decision-making processes since lacking them can lead to critical issues related to fairness and bias, as well as regulatory compliance. This arouses the need for rapid development in the research field of *eXplainable AI* (XAI). The aim of XAI is to connect the internal workings of ML/AI systems with human understanding and ultimately establish trustworthy AI. Various benefits of XAI are identified, including reducing bias in ML models, building trust in AI systems, and enhancing safety in safety-critical domains.

A wide variety of XAI methods towards enhancing transparency and trustworthiness of ML systems have emerged in recent years. Generally, XAI techniques can be classified according to various criteria, including intrinsic, post-hoc, model-agnostic, and model-specific methods. Among these techniques, model-agnostic methods are prominent in the XAI field, which produce two types of explanations, namely, feature selection and feature attribution explanations. A feature selection explanation identifies a set of features sufficient to get the prediction, whereas a feature attribution explanation indicates the importance of each feature for the prediction. However, model-agnostic methods face explanation quality challenges, such as issues with out-of-distribution sampling and unsoundness in the produced explanations. The limitations of these non-formal XAI approaches trigger a significant challenge to the reliability of model-agnostic explanations, especially in high-risk or safety-critical settings. Consequently, serious outcomes may arise from relying on such unsound explanations produced by model-agnostic methods.

As an alternative, formal eXplainable AI (FXAI) methods offer logic-based or formal explanations, aiming to provide rigorous and provable explanations for predictions produced by ML models. These formal XAI techniques are characterized by logic-based and model-specific XAI approaches, where explanations generated by formal methods are classified as two categories, namely, abductive explanations (AXp's) and contrastive explanations (CXp's). AXP's provide answers for "'why" questions, i.e. why a certain prediction was made, while CXP's address "why" nor "how" questions, i.e. why not another prediction was made or how to change the prediction. The thesis is motivated by rapid advancements of the XAI field and the shortcomings of prevailing model-agnostic approaches, focusing on formal explainability and aiming to tackle the challenges in model-agnostic methods as well as enhance formal explainability. Specifically, the thesis introduces a method to compute decision sets and lists that optimize either the number of literals used in the model or the trade-off between model size and accuracy. This addresses the explainability challenges encountered in interpretable models produced by prior studies. The thesis also propose an innovative anytime method for producing decision sets that are both accurate and interpretable. This involves compiling a gradient boosted tree into a decision set, resolving accuracy concerns associated with decision sets. Second, the thesis presents an approach to generating more succinct AXp's and more accurate CXp's with the use of background knowledge, improving the quality of AXp's and CXp's generated by existing approaches. Furthermore, the thesis introduces the first method to produce formal feature attribution and extends the approach to more efficiently producing or approximating formal feature attribution. Finally, a method to apply formal explainability in just-in-time (JIT) defect prediction is proposed. The proposed approach can efficiently generate explanations that are demonstrated to be accurate, robust, and actionable, whereas lack of this capability is identified in existing model-agnostic techniques.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature:

Print Name: Jinqiang Yu

Date: 20 April 2024

Publications during enrolment

The full citation of work declared in this "thesis by publication" compilation is as follows:

1. Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic. Learning optimal decision sets and lists with sat. *Journal of Artificial Intelligence Research,* 72, 1251-1279, 2021.

2. Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, Nina Narodytska, and Joao Marques-Silva. Eliminating the impossible, whatever remains must be true: On extracting and applying background knowledge in the context of formal explanations. *In Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 37, pp. 4123-4131, 2023.

3. Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. From formal boosted tree explanations to interpretable rule sets. *In 29th International Conference on Principles and Practice of Constraint Programming,* vol. 280, pp.38:1-38:21, 2023.

4. Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. On Formal Feature Attribution and Its Approximation. *arXiv preprint arXiv:2307.03380,* 2023.

5. Jinqiang Yu, Graham Farr, Alexey Ignatiev, and Peter J. Stuckey. Anytime Approximate Formal Feature Attribution. *arXiv preprint arXiv:2312.06973,* 2023.

6.Jinqiang Yu, Michael Fu, Alexey Ignatiev, Chakkrit Tantithamthavorn, and Peter J. Stuckey. A Formal Explainer for Just-In-Time Defect Predictions. *ACM Transactions on Software Engineering and Methodology,* 2024. (**Minor Revision submitted.**)

Thesis including published works declaration

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

This thesis includes *sixe* original papers published or accepted in peer reviewed journals, conferences and archives. The core theme of the thesis is *Explainable AI with the Use of Formal Reasoning*. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the *Faculty of Information Technology* under the supervision of *Prof. Peter J. Stuckey* and *Dr. Alexey Ignatiev*.

The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into team-based research. In the case of *Chapters 3–8* my contribution to the work involved the following:

Thesis Chapter	Publication Title	Status	Nature and % of student contribution	Co-author name(s) Nature and % of Co-author's contribution	Co-author(s), Monash student Y/N*
3	Learning Optimal Decision Sets and Lists with SAT	Published	Concept, conducting experiments and writing the manuscript. 65%.	1) Alexey Ignatiev. Concept, conducting experiments and input into manuscript. 20%. 2) Peter J. Stuckey. Concept and input into manuscript. 10%. 3) Pierre Le Bodic. Concept and input into manuscript. 5%.	No No No
4	From Formal Boosted Tree Explanations to Interpretable Rule Sets	Published	Concept, conducting experiments and writing the manuscript. 70%.	1) Alexey Ignatiev. Concept and input into manuscript. 15% 2) Peter J. Stuckey. Concept and input into manuscript. 15%.	No No
5	Eliminating the Impossible, Whatever Remains Must Be True: On Extracting and Applying Background Knowledge in the Context of Formal Explanations	Published	Concept, conducting experiments and writing the manuscript. 60%.	1) Alexey Ignatiev. Concept and input into manuscript. 15%. 2) Peter J. Stuckey. Concept and input into manuscript. 15%. 3) Nina Narodytska. Concept and input into manuscript. 5%. 4) Joao Marques-Silva. Concept and input into manuscript. 5%.	No No No No
6	On Formal Feature Attribution and Its Approximation	Published	Concept, conducting experiments and writing the manuscript. 70%.	1) Alexey Ignatiev. Concept and input into manuscript. 15%. 2) Peter J. Stuckey. Concept and input into manuscript. 15%.	No No
7	Anytime Approximate Formal Feature Attribution	Published	Concept, conducting experiments and writing the manuscript. 70%.	1) Graham Farr. Concept and input into manuscript. 10%. 2) Alexey Ignatiev. Concept and input into manuscript. 10%. 3) Peter J. Stuckey. Concept and input into manuscript. 10%.	No No No
8	A Formal Explainer for Just-In-Time Defect Predictions	Published	Concept, conducting experiments and writing the manuscript. 65%.	1) Michael Fu. Concept, implementing package and input into manuscript. 15%. 2) Alexey Ignatiev. Concept and input into manuscript. 10%. 3) Chakkrit Tantithamthavorn. Concept and input into manuscript. 5%. 4) Peter J. Stuckey. Concept and input into manuscript. 5%.	Yes No No No

I have not renumbered sections of submitted or published papers in order to generate a consistent presentation within the thesis.

Student name: Jinqiang Yu

Student signature: **Date:**

I hereby certify that the above declaration correctly reflects the nature and extent of the student's and co-authors' contributions to this work. In instances where I am not the responsible author I have consulted with the responsible author to agree on the respective contributions of the authors.

Main Supervisor name: Prof. Peter J. Stuckey

Main Supervisor signature: **Date:**

Acknowledgements

The journey of pursuing my Ph.D. is both challenging and rewarding in my life. Without the guidance, encouragement, and support from my supervisors, colleagues, and family, this achievement is not attainable.

I am deeply grateful for the supervision provided by Prof. Peter J. Stuckey and Dr. Alexey Ignatiev throughout my research journey. Their guidance and support have been invaluable, and their thoughtful manner to science has a significant factor in driving my work. I express my gratitude to them for granting me the scholarship from their funding, which has played a crucial role in supporting my Ph.D. studies. Furthermore, I am sincerely thankful to them for acquainting me with the exciting realm of explainable AI. This exploration of explainable AI has not only broadened my perspective but has also fueled my enthusiasm for AI research. I am fortunate to have supervisors who actively engage in every one of my research projects, providing constant guidance and assistance. Moreover, I highly value the flexibility they grant me to investigate a range of interesting problems. To both my supervisors, I extend my sincere gratitude for the dedication you have shown in guiding me, ultimately shaping me through the Ph.D journey.

I would like to express my gratitude to all members of the panel committee, Assoc. Prof. Arun Konagurthu, Assoc. Prof. Guido Tack, and Prof. Geoff Webb, for their invaluable feedback and support throughout my candidature. This thesis has significantly benefited from the suggestions provided by the examiners XXX and XXX. Additionally, I extend my appreciation to my collaborators Dr. Pierre Le Bodic, Prof. Joao Marques-Silva, Dr. Nina Narodytska, Dr. Chakkrit Tantithamthavorn, Michael Fu, and Prof. Graham Farr. Their dedication and hard work on my research projects are deeply appreciated. Their invaluable assistance and guidance are significant, without which I would not be able to complete these research works.

I will like to thank my fellow researchers and colleagues at the OPTIMA lab for fostering a stimulating and supportive environment that has contributed to my intellectual development. Also, I like to express my gratitude to Dr. Bojie Shen, Hendrik Bierlee, Kelvin Davis, Jiarong Fan, Shizhe Zhao, Dr. Allen Zhong, Dr. Terrence Mak, Sushmita Paul, Dr. Jip Dekker, and other members for engaging in numerous insightful research discussions with me, and for the interesting micro talks we enjoyed on Fridays. Their encouragement and support have been crucial in inspiring me to successfully complete my research journey.

Finally, I deeply appreciate my family for their love, unconditional support, and continuous encouragement during my Ph.D. journey. Their belief in my capabilities has

consistently provided me with motivation and strength even in the toughest time. I am sincerely thankful to them for being instrumental in my journey. Without their support, I would not have reached this point.

Contents

Copyright notice i

Abstract ii

Declaration iv

Publications during enrolment v

Thesis including published works declaration vi

Acknowledgements ix

1 Introduction 1
- 1.1 Achievements of Machine Learning . . . 1
- 1.2 Rise of eXplainable AI . . . 2
- 1.3 Motivations of Formal Explainability . . . 5
- 1.4 Research Questions and Contributions . . . 6
  - 1.4.1 Research Questions . . . 6
  - 1.4.2 Contributions . . . 7
  - 1.4.3 Contributions for RQ1. . . . 7
  - 1.4.4 Contributions for RQ2. . . . 7
  - 1.4.5 Contributions for RQ3. . . . 8
- 1.5 Thesis Organization . . . 9

2 Background 10
- 2.1 Classification and Machine Learning Models . . . 10
  - 2.1.1 Classification Problems . . . 10
  - 2.1.2 Decision Sets & Lists . . . 11
  - 2.1.3 Decision Trees and Gradient Boosted Trees . . . 12
  - 2.1.4 Neural Networks . . . 12
- 2.2 Explainable AI . . . 13
  - 2.2.1 ML Model Interpretability and Post-Hoc Explanations . . . 14
  - 2.2.2 Model-agnostic methods for post-hoc explainability . . . 15
    - 2.2.2.1 Model-agnostic methods for feature attribution . . . 15
    - 2.2.2.2 Model-agnostic methods for feature selection . . . 18
  - 2.2.3 Limitations of Model-agnostic methods . . . 18
  - 2.2.4 Formal explainability . . . 19
- 2.3 SAT & MaxSAT . . . 20

2.3.1 Boolean Satisfiability (SAT) 20
2.3.2 Maximum Satisfiability (MaxSAT) 20

3 Learning Optimal Decision Sets and Lists with SAT 22

4 From formal boosted tree explanations to interpretable rule sets 24

5 Extracting and Applying Background Knowledge in the Context of Formal Explanations 26

6 On Formal Feature Attribution and Its Approximation 28

7 Anytime Approximate Formal Feature Attribution 29

8 A Formal Explainer for Just-In-Time Defect Predictions 31

9 Conclusions and Future Work 63

Bibliography 67

Chapter 1

Introduction

1.1 Achievements of Machine Learning

The rapid progress in machine learning (ML) and Artificial Intelligence (AI) techniques has drastically transformed various aspects of human lives over recent years [2, 100]. These impressive ML and AI methods, due to their advancements in algorithms and enhanced computational capabilities, have found vast use across diverse tasks, such as those in natural language processing (NLP) and computer vision (CV) fields.

In the language technology field, a notable advancement has been made through the development of large language models (LLMs) [32, 90, 169], BERT [43], GPT-3 [27], GPT-4 [28], and LLaMA-2 [167]. These models have showcased remarkable performance across various natural language processing (NLP) tasks. Furthermore, they signify a notable advancement in natural language generation (NLG) and natural language understanding (NLU) capabilities, highlighting their adaptability across diverse applications such as chatbots and content generation. Leading technology companies have integrated LLMs into their commercial products and services to augment functionality. As an example, Microsoft integrates GPT into the new Bing to enhance search relevance ranking [119]. Meanwhile, advancements in the image generation realm, facilitated by of technologies such as Stable Difussion [147], variational autoencoder (VAE) [93], generative adversarial networks (GANs) [54], vision transformers [45], have significantly influenced the field, notably improving image generation capabilities across domains such as computer graphics and artistic design. Moreover, significant milestones have been achieved in the filed of reinforcement learning, illustrated by the outstanding performance of AI agents, such as AlphaGo [125, 162, 163]. These agents outperformed world champions in complex games like chess and Go, demonstrating the potential of AI in strategic decision-making.

The significance of ML advancements for society is also highlighted by the substantial contributions, particularly in the general AI/ML field, made by leading technology companies, such as Amazon, Facebook, Google, Microsoft, Baidu, Amazon, Oracle, and many others. Furthermore, the emergence of AI for Science (AI4Science) [5, 182], making a bridge to connect AI technologies with scientific fields, has proven to be an influential method for accelerating scientific research and discovery. AI4Science leverages AI's computational power to analyze complex scientific data, formulate hypotheses, and foster advancements in areas like materials science [29], climate science [40, 88], and genomics [50, 172].

While these accomplishments have raised awareness of the transformative potential of ML and AI, they have also emphasized the critical importance of dependable and trustworthy AI [91, 101, 113, 115, 158, 173]. Maintaining transparency, ethical standards, and responsible implementation are crucial to guarantee the positive societal effects of these technologies, despite their limitless potential.

1.2 Rise of eXplainable AI

Complex ML models are rapidly integrated into our daily lives due to the impressive progress in ML. These complex models, driven by their exceptional accuracy, significantly impact decision-making across various fields including healthcare [111, 131, 141, 142, 154], law [95, 137], finance [15, 53, 152], and transportation [1, 59, 129]. However, accuracy frequently sacrifices explainability, leaving both domain experts and general users without clear insights into the outcomes generated by these complex model although users need to make decisions based on them. Complex ML models are commonly referred to as "black boxes" [31, 108, 176, 181] owing to their opaque internal mechanisms and the intricate structure that significantly complicates model interpretation, making it challenging for humans to understand. With the growing adoption of complex ML models, there is a rising need for transparency and accountability [58, 98, 140, 170] in their decision-making processes. Users, stakeholders, and regulators require insight into the rationale behind the decisions or recommendations made by ML/AI systems. Lacking such transparency and accountability can lead to several critical issues as follows.

- Fairness and Bias: Due to the potential for ML/AI systems to generate biased or unfair decisions [10, 62], explainability is essential for understanding the mechanisms behind these biases and mitigating them in ML models. This guarantees that AI-based decisions do not worsen social disparities.

- Regulatory Compliance: In some sectors, including law and finance, regulations and standards mandate decision-making processes to be transparent and comprehensible. The absence of transparency can result in compliance challenges because stakeholders may find it difficult to understand and justify the decisions made by AI systems [34, 87, 155].
- Safety and Robustness: In safety-critical domains like autonomous vehicles and healthcare, it is crucial to gain insight into how AI systems handle unexpected situations. Lacking transparency presents considerable challenges in guaranteeing the safety and robustness of AI systems, especially in contexts where the operation of AI models can potentially trigger unexpected consequences [41].

In light of the identified critical issues in ML, there is an imperative demand to build trust in ML/AI systems. This need has led to rapid development in the research field of *eXplainable AI* (XAI), which is aimed at making a bridge between the internal mechanisms of ML/AI systems and human comprehension, ultimately targeting establishing *trustworthy AI* [113].

The significance of XAI. The significance of both XAI and trustworthy AI is emphasized by recent regulations and guidelines from influential entities [3, 4, 36, 47, 48, 113, 130, 168]. The main reasons of the significance of XAI are identified as follows.

- Reducing Bias: XAI can contribute to identifying and alleviating bias in ML models. With the assistance of XAI in gaining insight of the features and data points affecting decisions, professionals can tackle biases and achieve fairness in AI applications.
- Building Trust: Trust is a vital factor to consider in the adoption of AI. XAI enables stakeholders to obtain a better understanding of the decision-making process, therefore enhancing trust in the technology. This is especially vital in situations where AI automates or assists decision-making processes, such as in finance, where investors depend on AI-assisted investment advice, and in healthcare, where doctors require trust in AI-supported diagnoses.
- Boosting productivity: Methods facilitating explainability can quickly uncover errors and/or areas requiring improvement, simplifying tasks for ML operations (MLOps) teams who are responsible for efficiently monitoring and maintaining AI systems. For instance, the insight of particular features that drive the model's output enables technical teams to verify whether the patterns identified by the model have wide applicability and relevance for future predictions, or if they merely represent a particular data point.

- Enhancing Safety: In safety-critical fields, XAI facilitates users understanding the rationale behind the decision made by AI systems. For instance, in the context of autonomous vehicles, the understanding of driving system's actions can prevent accidents.
- Aiding Compliance: There is a rising call for transparency in ML algorithms, with governments and regulatory organizations increasingly paying attention to AI ethics. Aligning with these principles, XAI can assist organizations in complying with established regulations and ethical standards.

XAI Approaches. In recent years, various approaches to XAI have emerged, aiming to improve the transparency and trustworthiness of ML systems. In general, XAI techniques can be categorized based on different criteria, namely intrinsic, post-hoc, model-agnostic, and model-specific methods.

- *Intrinsic* approaches concentrate on structuring ML models in a manner that ensures they are inherently interpretable from the beginning. In general, models with straightforward architectures are referred to as *interpretable* ML models, frequently deemed intrinsically interpretable [148]. Arguably, the most explainable types of ML models include decision trees, decision lists, and decision sets, as they consist of straightforward logical rules. Among these, decision sets offer the most straightforward explanations, where if a rule in a decision set "fires" for a specific data instance, that rule alone serves as the explanation. Regarding decision lists, which consist of an ordered sets of rules, we are required to additionally consider the order of rules in the model.
- *Post-hoc* methods explain trained ML models by applying interpretation techniques. These techniques are applicable to complex and inherently non-interpretable models, including neural networks and transformers. The two primary branches of research in post-hoc explanations include *feature selection* techniques, exemplified by Anchors [145], and *feature attribution* methods, such as LIME [144] and SHAP [109].
- *Model-agnostic* techniques [109, 144, 145] are applicable to any ML model, including complex models. These techniques are often implemented after the model has finished its training phase, rendering them post hoc in nature. Model-agnostic approaches often generate explanations by examining the relationships between inputs and outputs, without depending on access to the internal information of the model, such as weights and structural details. Undoubtedly, model-agnostic methods are viewed as the prevailing approach in the domain of XAI.

- *Model-specific* methods are customized for specific categories of models and are not universally applicable. In contrast to model-agnostic approaches, model-specific methods can offer more detailed insights into particular types of models, although their applicability is restricted to those particular models. There is an increasing tendency to apply formal approaches for ML systems verification [158], with logic-based explainability playing a crucial role in this trend [37, 73, 113–115].

To gain a deeper insight into these XAI approaches, interested readers can refer to reference [9, 37, 60, 73, 113, 115, 126, 140, 175]. The majority of XAI methods explored in this thesis can be classified as intrinsic and post-hoc approaches.

1.3 Motivations of Formal Explainability

While model-agnostic methods are prevalent in the XAI domain, they encounter challenges regarding the quality of explanations. An emerging alternative is formal explainability, which represents a rising trend in applying automated reasoning techniques to explain and verify ML models. The challenges faced by model-agnostic methods and the emergence of formal explainability are outlined below.

Limitations of Model-agnostic Explanations. Numerous XAI techniques introduced in recent years are model-agnostic, yet they unfortunately encounter several issues. These issues include out-of-distribution sampling [96, 164, 178, 179] and unsoundness in generated explanations [73, 77, 114, 128]. Due to the limitations presented by these non-formal XAI methods, a notable challenge regarding the reliability of model-agnostic explanations is identified, particularly in high-risk or safety-critical environments [35, 61, 113, 137, 148, 150, 170]. Relying on such unsound explanations generated by model-agnostic methods can trigger severe consequences. In addition to unsoundness, other drawbacks of model-agnostic explanations have been identified [30, 44, 63–65, 86, 94].

Formal Explainability. Formal XAI (FXAI) approaches, providing *logic-based explanations* or *formal explanations*, stand as an alternative to model-agnostic methods. These formal XAI methods are characterized by model-specific and logic-based XAI methods [37, 73, 113, 114]. FXAI is aimed at offering rigorous and provable explanations for outputs made by ML models. There are two categories of formal explanations, namely, *abductive explanations* (AXp's) and *contrastive explanations* (CXp's). AXp's provide answers for *why* a prediction was made, while CXp's answer questions of *why not* another prediction was made or *how* to change the prediction produced

by the ML model. Several studies introduce formal methods for computing explanations [7, 107, 138, 139, 174], with one prominent approach in formal explainability building on abductive reasoning [14, 68–71, 73, 74, 76, 77, 81, 83, 114, 116, 117, 128, 161, 180].

Abductive Reasoning for Formal Explanations. In general, abduction-based approaches involves encoding a given ML model into a logical-based representation. Therefore, the efficiency of generating explanations through these methods relies on optimizing the encoding from ML models to logic-based representations, along with the capabilities of automated reasoning tools, e.g. SAT (Boolean satisfiability), SMT (satisfiability modulo theories), and QBF (quantified Boolean formula) solvers. However, recent research has presented a technique that uses diverse robustness tools to generate formal explanations [66], which does not explicitly demand the encoding from ML models to logic-based representations.

Explainability for Interpretable Models. Intrinsically interpretable ML models [20, 33, 106, 126, 149, 157], as their name implies, can offer explanations without requiring extra computation. Decision trees [26, 72, 135, 136], decision lists [146, 151], and decision sets [97, 122] are widely considered as the most interpretable ML models. Recent studies have revealed that formal explanations for decision trees can be notably more compact than explanations offered by interpretable ML models [84, 85], with decision tree explanations aligning with the paths within the tree. Moreover, the interpretability of decision lists and sets presents challenges [115]. These findings indicate that even interpretable models still require explanation or further interpretability improvement.

1.4 Research Questions and Contributions

1.4.1 Research Questions

Motivated by the rapid development in the XAI domain and the identified limitations of prevalent model-agnostic methods, this thesis focuses on XAI problems, in particular formal explainability, aiming to develop techniques to improve explainability of ML models and address the gap in formal explainabaility. The thesis is categorized into three research questions (RQs):

- RQ1: How can we learn ML models that are both accurate and interpretable?
- RQ2: How can we compute succinct and accurate explanations?
- RQ3: How can we apply formal explanations in real-world scenarios?

1.4.2 Contributions

1.4.3 Contributions for RQ1.

To address RQ1, two research works are introduced:

- Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic. Learning optimal decision sets and lists with sat. *Journal of Artificial Intelligence Research,* 72, 1251-1279, 2021.
- Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. From formal boosted tree explanations to interpretable rule sets. *In 29th International Conference on Principles and Practice of Constraint Programming,* vol. 280, pp.38:1-38:21, 2023.

First, we present a method for constructing decision sets and decision lists optimal in either the number of required literals or the trade-off between model size and accuracy. In this study, we introduce a novel metric for assessing the explainability of interpretable ML models, specifically focusing on the number of literals used. Previous research has primarily relied on a metric based on the number of rules, which may not accurately capture explainability. For instance, two rules with 70 conditions each are considerably less interpretable than six rules with only three conditions each. With the new explainability measure, we can generate more interpretable decision sets and lists.

Furthermore, we introduce a novel anytime approach to generating decision sets that focus on both accuracy and interpretability. Motivated by the issue in existing approaches to decision sets, which cannot offer any decision information if a dataset is not completely solved, we propose a method that establishes a connection between formal post-hoc explainability and interpretable decision set models. This involves distilling a gradient boosted tree model into a decision set as needed, making use of formal explanations in the process. This technique is a knowledge distillation method and in theory, it can be extended to other ML models.

1.4.4 Contributions for RQ2.

To tackle RQ2, three research works are proposed:

- Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, Nina Narodytska, and Joao Marques-Silva. Eliminating the impossible, whatever remains must be true: On extracting

and applying background knowledge in the context of formal explanations. *In Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 37, pp. 4123-4131, 2023.

- Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. On Formal Feature Attribution and Its Approximation. *arXiv preprint arXiv:2307.03380,* 2023.
- Jinqiang Yu, Graham Farr, Alexey Ignatiev, and Peter J. Stuckey. Anytime Approximate Formal Feature Attribution. *arXiv preprint arXiv:2312.06973,* 2023.

We introduce a method for leveraging background knowledge, i.e. correlations between features, to generate more concise "why" formal explanations which are presumably more understandable to humans, and to provide more precise "why not" explanations. These approaches tackle the problem of lengthy formal explanations, which are caused by the general formal methods that consider the entire feature space under the assumption of feature independence and uniform distribution [171]. This issue can make the explanations difficult for users to interpret, and also limit the practical applicability of the formal explainability techniques.

In addition to the drawback of formal explanations being lengthy, the formal approach also fails to provide feature attribution. To overcome this limitation, we propose a new formal method for feature attribution. By exhaustively enumerating all formal explanations, we establish a clear definition of formal feature attribution (FFA) as the proportion of explanations where a particular feature appears. However, we argue that formal feature attribution presents challenges for the second level of the polynomial hierarchy. Therefore, we further extend our work by developing an anytime approach that enables the efficient computation of approximate FFA and thus extending its practical applicability.

1.4.5 Contributions for RQ3.

To solve RQ3, one research work is presented:

- Jinqiang Yu, Michael Fu, Alexey Ignatiev, Chakkrit Tantithamthavorn, and Peter J. Stuckey. A Formal Explainer for Just-In-Time Defect Predictions. *ACM Transactions on Software Engineering and Methodology,* 2024. (**Accepted.**)

We propose an method to apply formal explainability in a just-in-time (JIT) defect prediction. JIT defect prediction aims to predict whether a commit will potentially introduce software defects, helping teams in directing their limited resources towards

the most high-risk commits or pull requests. This proposed approach offers explanations that are not only provably correct, but also guaranteed to be minimal. This tackles the shortcomings observed in previous research, where model-agnostic techniques are used to explain JIT model predictions, resulting in explanations that are not formally sound, robust, and actionable. Our proposed approach is capable of efficiently producing explanations that are proven to be correct, robust, and actionable, while existing model-agnostic techniques lack this capability.

1.5 Thesis Organization

The thesis is organized as follows:

- Chapter 2 reviews the background related to XAI.
- Chapter 3 introduces the method to generate optimal decision sets and list.
- Chapter 4 shows the work to compile a gradient boosted tree into a decision set.
- Chapter 5 presents the technique to apply background to formal explainability.
- Chapter 6 illustrates the approach to computing formal feature attribution.
- Chapter 7 shows the anytime method to approximate formal feature attribution.
- Chapter 8 introduces work to apply formal explainability in jsut-in-time prediction.
- Chapter 9 presents the conclusions and feature work.

Chapter 2

Background

This chapter outlines the fundamental knowledge relevant to the work discussed in this thesis, including concepts and definitions. This chapter is divided into three sections. Section 2.1 introduces machine learning (ML) models designed for classification tasks. Specifically, our focus is on interpretable ML models, including decision sets, decision lists, and decision trees. In addition, we explore gradient boosted trees, which consist of a collection of decision trees, and introduce neural networks. In Section 2.2, we offer a concise overview of the history of eXplainable AI (XAI), with an emphasis on widely recognized model-agnostic techniques. However, such methods suffer from various significant issues, e.g., the unsoundness of explanations. Therefore, in this section, we provide further insight into an emerging research field: formal eXplainable AI (FXAI), which seeks to overcome these drawbacks and prioritize the rigor of explanations. Section 2.3 presents the concepts and definitions of Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT), which serve as foundational elements for formal explainability. Furthermore, we introduce two other prevalent concepts in formal explainability, namely, Minimal Unsatisfiable Subset (MUS) and Minimal Correction Subset (MCS).

2.1 Classification and Machine Learning Models

2.1.1 Classification Problems

A classification problem considers a set of features $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{1, 2, \ldots, k\}$. Each domain $\mathbb{D}_i$ for feature $i \in \mathcal{F}$ can be categorical or ordinal, i.e. integer, and the value of each feature i is taken from its corresponding domain $\mathbb{D}_i$. Therefore, the entire feature space is defined as $\mathbb{F} \triangleq \prod_{i=1}^{m} \mathbb{D}_i$. For boolean domains, $\mathbb{D}_i = \{0, 1\}$, $i \in \mathcal{F}$, and $\mathcal{F} = \{0, 1\}^m$. The notation $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$ represents a

specific point in feature space, where each $v_i \in \mathbb{D}_i$ is a constant taken for feature $i \in \mathcal{F}$. An *example* or *instance* is denoted by a concrete point $\mathbf{v} \in \mathbb{F}$ in feature space and its corresponding class $c \in \mathcal{K}$, i.e. an instance is represented by a pair $(\mathbf{v}, c)$. Given a subset $\mathcal{S} \in \mathcal{F}$, $\mathbf{v}_S$ is denoted as the partial point of $\mathbf{v}$, indicating the restriction of the entire point $\mathbf{v}$ to those features in $\mathcal{S}$. An arbitrary point in feature space is denoted by the notation $\mathbf{x} = (x_1, \ldots, x_m)$, where each $x_i \in \mathbf{x}$ is a variable which takes values from its corresponding domain $\mathbb{D}_i$ and represents feature $i \in \mathcal{F}$. Moreover, the set of classes $\mathcal{K}$ is assumed to contain finite classes; no extra restrictions are placed upon $\mathcal{K}$. Finally, a non-constant classification function $\kappa : \mathbb{F} \to \mathcal{K}$ is defined by a machine learning classifier $\mathcal{M}$, which is represented as a tuple $(\mathbb{F}, \mathcal{F}, \mathcal{K}, \kappa)$.

2.1.2 Decision Sets & Lists

Decision sets [75, 97, 122, 177] and decision lists [146, 151] are considered interpretable machine learning models, representing families of rule-based classifiers. These models offer users concise explanations directly from the models themselves.

A *decision rule* is structured as "IF antecedent THEN prediction", with the antecedent being a set of feature literals. A rule is considered to assign an instance $\mathbf{v} \in \mathcal{F}$ to class $c \in \mathcal{K}$ if its antecedent is *compatible* with $\mathbf{v}$ or *matches* $\mathbf{v}$ and its prediction is c.

A *decision set* (DS) consists of an unordered set of decision rules $\mathcal{R}$. An instance $(\mathbf{v}, c) \in \mathcal{E}$ is misclassified by a DS if either there are no rules in $\mathcal{R}$ compatible with $\mathbf{v}$, or if there is at least one rule that classifies $\mathbf{v}$ as a class $c' \in \mathcal{F}$ where $c' \neq c$.

A *decision list* (DL) is an ordered set of rules. The first rule of a decision list matches an instance is the one classifying the instance. Decision lists are typically represented as as a single cascade of IF-THEN-ELSEs, with the final rule serving as a default rule, which assigns all remaining instances to certain class.

Compared with DLs, there are a number of issues in DSs complicating their analysis. Overlapping is one of the issues, where two or more rules predicting different classes fires the same inputs [113, 115]. Coverage of feature space is another issue, where instances are not fired by any rule in DSs. To address this coverage problem, a default rule can be added, which assigns a certain class to instances fired by none of the other rules. However, this condition further complicates the reasoning process for DSs.

2.1.3 Decision Trees and Gradient Boosted Trees

Decision trees [26, 72, 135, 136] are another interpretable machine learning model, often considered to be more advantageous than most ML models as their decisions are generally straightforward to humans.

A *decision tree* (DT) $t = (V, E)$ consists of a set of nodes V and a set of edges E, forming a directed acyclic graph where there is at most one edge between any pair of nodes. The root node of a decision tree t has no incoming edges, while all other nodes have a single incoming edge. In this thesis, univariate decision trees are considered, where every non-terminal node is linked to exactly one feature $i \in \mathcal{F}$, and each terminal node corresponds to a value from a set of classes $\mathcal{K}$. Furthermore, each non-terminal node signifies a feature condition in the form of $x_i < d$, where feature $i \in \mathcal{F}$ and *splitting threshold* $d \in \mathbb{D}_i$. Each path in the DT t is represented as a sequence of nodes, e.g. $\mathcal{P} = \langle \mathfrak{n}_1, \ldots, \mathfrak{n}_n \rangle$, where $\mathfrak{n}_1$ is the root node and $\mathfrak{n}_n$ is the terminal node. Every pair $(\mathfrak{n}_j, \mathfrak{n}_{j+1})$ signifies an edge in the tree t. In each path $\mathcal{P} \in t$ from the root node to a terminal node, a feature may be tested multiple times, i.e., it is not read-once. An instance is assigned the class linked to the terminal node in the path that matches this instance.

While decision trees are easily understood by humans, they are prone to bias when they are small and tend to overfit when large. To address these shortcomings, tree ensembles were introduced, which involve generating multiple decision trees and aggregating their decisions to reach a final decision. Gradient boosted trees is one of the popular ensemble methods used to overcome the decision tree's limitations.

A *gradient boosted tree* (BT) is a tree ensemble $\mathfrak{T}$ constructed by iteratively building a sequence of small decision trees. At each stage, new trees are added to the ensemble to address the inaccuracies present in the existing tree ensemble. Concretely, a BT $\mathfrak{T}$ defines sets of decision trees $T_c \in \mathfrak{T}$ for each class $c \in [|\mathcal{K}|]$, where T_c consists of $N \in \mathbb{N}_{>0}$ trees t_{kz+c}, $z \in \{0, \ldots, N-1\}$, $k = |\mathcal{K}|$. Thus, each decision tree is associated with a certain class c and contributes to the weight class c. Given an instance $\mathbf{v} \in \mathbb{F}$, its class is determined by calculating the sum of scores assigned by trees for each class $w(\mathbf{v}, c) = \sum_{t \in T_c} t(\mathbf{v})$ and assigning the class with the highest score, i.e. $\operatorname{argmax}_{c \in [|\mathcal{K}|]} w(\mathbf{v}, c)$.

2.1.4 Neural Networks

While these interpretable machine learning models including decision sets, decision lists, and decision tree are straightforward to explain their predictions, they often lack the accuracy achieved by other state-of-the-art models. Neural networks [156], which are

the fundamental structures of advanced machine learning models, such as convolutional neural networks (CNN) [51, 99], recurrent neural networks (RNN) [46] and transformers [169], can be adopted to address the accuracy performance issues associated with these interpretable models

A *neural network* (NN) is a network N composed of artificial neurons where an instance $\mathbf{v} \in \mathbb{F}$ is passed through a sequence of layers $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ and each layer $l \in \mathcal{L}$ consists of a set of neurons. Here, the first layer $l_1 \in \mathcal{L}$ is the input layer, layers $\{l_2, \ldots, l_{n-1}\}$ are the hidden layers, and the final layer l_n acts as the output layer. Functionally, a neural network defines a mapping $N : \mathbb{F} \to \mathcal{K}$. In classification tasks, the output yields scores (or probabilities) across $|\mathcal{K}|$ classes, with the class receiving the highest score being the prediction of the input instance $\mathbf{v}$. Each neuron o in a neural network N forms interconnected nodes arranged in layers. These neurons collaborate to process and convey information. Each neuron o accepts inputs, processes them, and generates an output, which is subsequently conveyed to other neurons in the following layer of the network. In each layer $l_i \in \mathcal{L}$, multiple inputs are received from either the instance $\mathbf{v}$ (for the input layer l_1, i.e. $i = 1$) or the outputs of the neurons in the previous layer l_{i-1}, $i \in \{2, \ldots . n\}$. Each neuron o in layer l_i applies weights to them, aggregates them, and afterwards passes the result through an activation function to generate an output. Finally, the output layer l_n in a neural network N provides scores across $|\mathcal{K}|$ classes for an instance $\mathbf{v}$ and assign the class c with the highest score.

However, although neural networks can deliver exceptional performance, they often lack of explainability, leading to challenges in understanding their predictions. This highlights the necessity for explainable AI.

2.2 Explainable AI

Inspired by the growing adoption of machine learning (ML) across various domains, there is a rising demand for eXplainable AI (XAI). Its significance lies not only in fostering trust but also in validating ML models [57, 153]. XAI approaches to interpreting the behavior of ML models can be categorized based on different criteria, such as whether they are intrinsic or post-hoc, and whether they are model-agnostic or model-specific. The thesis will exclusively focus on *intrinsic* and *post-hoc* methods.

Intrinsic methods aim to construct ML models that are inherently understandable from the beginning. Models featuring straightforward architectures, like decision sets [97],

lists [151], and trees [72], are typically deemed intrinsically interpretable. They possess a natural ability to offer straightforward explanations without requiring additional computation.

Post-hoc approaches provide valuable insights into the predictions of individual data points, fostering trust at the level of individual data points. These methods produce two primary categories of post-hoc explanations, including *feature selection* and *feature attribution*.

2.2.1 ML Model Interpretability and Post-Hoc Explanations

Interpretability is commonly regarded as a subjective concept, with no formal definition [105]. An method to assess interpretability involves evaluating the succinctness of information provided by an ML model to justify a specific prediction.

There has been a surge in proposed methods for computing post-hoc explanations recently [124, 126]. The majority of well-known methods are model-agnostic [109, 144, 145], where they are allowed to generate explanations for any black-box ML model without the need of access to the model's internal architecture or parameters. Model-agnostic approaches are typically categorized as either *feature selection* or *feature attribution* methods, depending on the type of explanations they provide, as discussed below.

Feature Selection. A feature selection method aims to identify subsets of features $\omega \subseteq \mathcal{F}$, which are considered adequate for the prediction $c = \kappa(\mathbf{v})$ given instance $\mathbf{v}$. As noted above, most feature selection methods are model-agnostic, with Anchors [145] standing out as a notable example. The adequacy of the chosen feature set for a particular prediction is statistically assessed through extensively perturbing the target instance according the distributions observed in training data. This evaluation involves various metrics, such as fidelity, precision, and others. Therefore, feature selection explanations represented as a set of features $\omega \subseteq \mathcal{F}$ should be interpreted as the conjunction $\bigwedge_{i \in \omega} (x_i = v_i)$ that is considered sufficient for the prediction $c = \kappa(\mathbf{v})$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$.

Feature Attribution. An alternative perspective on post-hoc explanations is offered by feature attribution methods, such as LIME [144] and SHAP [109]. These methods attribute significance to all features of the model based on perturbations of the target instance, assigning a numerical value $w_i \in \mathbb{R}$ to denote the importance of each feature $i \in \mathcal{F}$. The features can subsequently be ranked from the most significant to the least significant based on their importance values. Consequently, a feature attribution explanation typically is represented as a linear form $\sum_{i \in \mathcal{F}} w_i \cdot x'_i$, where each $x'_i \in \{0, 1\}$

signifies the absence/ presence of $x_i \in \mathbf{x}$. This representation can also be viewed as an approximation of the original black-box model κ for the instance $\mathbf{v} \in \mathbb{F}$. Among various feature attribution methods, SHAP [12, 13, 109] often stands out as it targets approximating Shapley values, a robust derived from cooperative games in game theory [159].

2.2.2 Model-agnostic methods for post-hoc explainability

As discussed in Section 2.2.1, prevalent post-hoc explainability approaches can be generally classified into two categories: those relying on feature attribution and those relying on feature selection. The majority of prominent methods are model-agnostic, indicating that they can be employed with any black-box ML model without the need for access to the model's internal structure and parameters.

2.2.2.1 Model-agnostic methods for feature attribution

The best explanation provided for a straightforward model is the model itself as it precisely and succinctly illustrates its decision-making process and therefore it is straightforward to understand. With complex models, e.g. neural networks or ensemble methods, relying on the original model itself as a suitable explanation is not feasible, as these models feature complex structures that are challenging to understand. As a result, model-agnostic approaches adopt a simpler explanation model g, which is characterized as any interpretable approximation of the original model. Notable choices include LIME [144] and SHAP [109].

As introduced in LIME [144], many model-agnostic aims to explain a prediction $c = \kappa(\mathbf{x})$ given instance $\mathbf{x}$ and ML model $\mathcal{M}$. In an explanation model g, a simplified input $\mathbf{x}'$ is commonly used, where each $x'_i \in \mathbf{x}'$ denotes the absence/ presence of $x_i \in \mathbf{x}$. These approaches are conventionally referred to as *additive feature attribution* methods.

Definition 2.1 (Additive feature attribution approaches)**.** These methods provide an explanation model represented as a linear function of binary variables:

$$g(\mathbf{x}') = \phi_o + \sum_{i}^{|\mathcal{F}|} \phi_i x'_i \tag{2.1}$$

where $\mathbf{x}' \in \{0,1\}^{|\mathcal{F}|}$ and $\phi_i \in \mathbb{R}$ indicates the attribution of feature $i \in \mathcal{F}$.

Local Interpretable Model-agnostic Explanations (LIME). LIME [144] is a widely used model-agnostic additive feature attribution approach within the domain of XAI. The fundamental idea behind LIME is to approximate the functionality of a black-box

ML model $\mathcal{M}$ for a specific instance by constructing a simpler model g, often referred to as a "surrogate model" or "local model". LIME produces feature attribution explanations by perturbing the input instance of interest and examining the resulting changes in the model's predictions. Explanations are generated in the form of feature importance scores as outlined in Definition 2.1. These explanations allow users to understand the rationale behind the model's decision for a specific instance.

LIME aims to find an explanation represented as a model $g \in \mathcal{G}$ that integrates both interpretability and local fidelity, where $\mathcal{G}$ is a set of potential local explanation models. Concretely, they employ $\Omega(g)$ to represent a measure of the explanation's complexity, in contrast to interpretability. Additionally, they use $\pi_{\mathbf{x}}$ as a measure of proximity between an instance $\mathbf{x}$ and $\mathbf{v}$, thereby indicating the locality around $\mathbf{v}$. Finally, $\mathcal{L}(\kappa, g, \pi_{\mathbf{x}})$ is used to denote a measure of how unfaithful g approximates κ within the locality represented by $\pi_{\mathbf{x}}$. The explanation generated by LIME is obtained as follows:

Definition 2.2 (LIME)**.** Given a set of features $\mathcal{F}$, a classifier $\mathcal{M}$ linked with a classification function κ on these features, and a data point $\mathbf{v} \in \mathbb{F}$, the explanation generated by LIME is defined as

$$\xi = \arg\min_{g \in \mathcal{G}} \mathcal{L}(\kappa, g, \pi_{\mathbf{x}}) + \Omega(g) \tag{2.2}$$

In this context, LIME targets minimizing $\mathcal{L}(\kappa, g, \pi_{\mathbf{x}})$ while keeping $\Omega(g)$ sufficiently low, thus achieving explanations that balance interpretability and local fidelity.

SHapley Additive exPlanations (SHAP). Shapley values were initially proposed by L. Shapley [6, 159] within the domain of game theory. In cooperative games, Shapley values indicate the worth of the game contributed by each player. Shapley values have been widely adopted to provide explanations for the predictions of ML models, including methods proposed in [109, 165, 166], and numerous other works. In these methods, each feature (along with its corresponding value) is considered as an independent player when explaining an given instance. A numerical value is assigned to each feature, representing its contribution to the prediction.

SHapley Additive exPlanations (SHAP) [109, 126] is an example of such methods that calculates SHAP scores representing Shapley values in the domain of XAI. It stands out as one of the most widely adopted model-agnostic feature attribution techniques. SHAP computes the impact of each feature's presence or absence on model predictions by taking all possible feature combinations into account and then determining the average contribution of each feature across all potential combinations. While the exact computation of SHAP scores is acknowledged to be computationally hard [12, 13], the computation of SHAP scores becomes polynomial for some families of classifiers.

Consider $\mathcal{H} : 2^{\mathcal{F}} \rightarrow 2^{\mathbb{F}}$ defined as follows:

$$\mathcal{H}(S; \mathbf{v}) := \{\mathbf{x} \in \mathbb{F} \mid \wedge_{i \in S}\ x_i = v_i\} \tag{2.3}$$

$\mathcal{H}(S; \mathbf{v})$ represents all the data points in the feature space that share the same values of features as S with $\mathbf{v}$.

To generate SHAP scores, SHAP requires a probability distribution across the features. Let the probability of a data point be denoted as $\Pr(\cdot)$. Moreover, the *expected value* of a classification function κ is represented as $E[\kappa]$, with $E[\kappa|\mathbf{v}] = \kappa(\mathbf{v})$ for a complete data point $\mathbf{v}$. Consider $E[\kappa|\mathbf{v}_S]$ denoting the expected value of the boolean function $\kappa|\mathbf{v}_S$, defined as follows:

$$E[\kappa|\mathbf{v}_S] := \sum_{\mathbf{x} \in \mathcal{H}(S;\mathbf{v})} \kappa(\mathbf{x}) \cdot \Pr(\mathbf{x}|\mathbf{v}_S) \tag{2.4}$$

Consider $\phi : 2^{\mathcal{F}} \rightarrow \mathbb{R}$ defined as follows:

$$\phi(S; \mathcal{M}, \mathbf{v}) := E[\kappa|\mathbf{v}_S] \tag{2.5}$$

For a uniform distribution, $\phi(S; \mathcal{M}, \mathbf{v})$ is defined as:

$$\phi(S; \mathcal{M}, \mathbf{v}) = \frac{1}{2^{|\mathcal{F} \setminus S|}} \sum_{\mathbf{x} \in \mathcal{H}(S;\mathbf{v})} \kappa(\mathbf{x}) \tag{2.6}$$

The following definitions are used to simplify the notation:

$$\Delta(i, S; \mathcal{M}, \mathbf{v}) := \phi(S \cup \{i\}; \mathcal{M}, \mathbf{v}) - \phi(S; \mathcal{M}, \mathbf{v}) \tag{2.7}$$

$$\zeta(S; \mathcal{M}, \mathbf{v}) := \frac{|S|!\,(|\mathcal{F}| - |S| - 1)!}{|\mathcal{F}|!} \tag{2.8}$$

Definition 2.3 (SHAP Scores [12, 13, 42, 109])**.** Given a classifier $\mathcal{M}$ functioned on a set of features $\mathcal{F}$, along with a probability distribution Pr, and an instance $\mathbf{v} \in \mathbb{F}$, let $\text{SHAP} : \mathcal{F} \rightarrow \mathbb{R}$ be the SHAP score for feature $i \in \mathcal{F}$ on $\mathbf{v}$ in relation to $\mathcal{M}$, defined as:

$$\text{SHAP}(i; \mathcal{M}; \mathbf{v}) := \sum_{S \subseteq \mathcal{F} \setminus i} \zeta(S; \mathcal{M}, \mathbf{v}) \cdot \Delta(i, S; \mathcal{M}, \mathbf{v}) \tag{2.9}$$

Note that the sum of the expected value $\phi(\emptyset; \mathcal{M}, \mathbf{v})$ of the classification function κ and SHAP scores for all features $\sum_{i \in \mathcal{F}} \text{SHAP}(i; \mathcal{M}, \mathbf{v})$ is associated with the prediction of the

given instance $\kappa(\mathbf{v})$ [42, 165, 166]:

$$\phi(\emptyset; \mathcal{M}, \mathbf{v}) + \sum_{i \in \mathcal{F}} \text{SHAP}(i; \mathcal{M}, \mathbf{v}) = \kappa(\mathbf{v}) \tag{2.10}$$

2.2.2.2 Model-agnostic methods for feature selection

Anchor. Anchor Explanations (Anchor) [145] stands out as a well-known model-agnostic feature selection approach, offering explanations for predictions made by complex ML models. The core concept of Anchor is to explore concise and readily comprehensible "anchor rules" that adequately explain a model's predictions for individual instances. These anchor rules are straightforward, consisting of IF-THEN statements that capture important features or conditions determining a specific prediction by the model. Anchor explanations are succinct and easily understandable, offering users insights into why the model predicts a particular class for an individual instance.

Definition 2.4 (Anchors [145])**.** Given a classification function κ, and a data instance $\mathbf{v}$ to be explained, A is an anchor if

$$E_{\mathcal{D}(\mathbf{x} \mid A)}[1_{\kappa(\mathbf{x})=\kappa(\mathbf{v})}] \geq \delta,\ A(\mathbf{v}) = 1 \tag{2.11}$$

Here, $\mathcal{D}(\cdot \mid A)$ signifies the conditional distribution when the rule A is applied. δ ranging from 0 to 1 is a precision threshold for local fidelity. Only rules with a local fidelity of at least δ are deemed valid. Additionally, $1_{\kappa(\mathbf{x})=\kappa(\mathbf{v})}$ denotes the indicator function.

A represents a rule consist of a set of predicates, where $A(\mathbf{v})$ returns 1 only if all its feature predicates align with the feature values of $\mathbf{v}$.

2.2.3 Limitations of Model-agnostic methods

Model-agnostic methods overlook the complexities inherent in ML models and instead concentrate on examining its input-output behavior. Model-agnostic approaches are vulnerable to various significant challenges, such as producing unreliable explanations [73, 77, 114, 128] and encountering issues with out-of-distribution sampling [96, 164, 178, 179]. These challenges further deteriorate the trust of AI systems. For example, unsound explanations can result in catastrophic outcomes in situations that are high-risk or safety-critical [35, 61, 137, 148, 150, 170]. In addition to unsoundness, model-agnostic explanations have been found to have various other limitations [44, 67, 94, 96, 164].

2.2.4 Formal explainability

In contrast to model-agnostic methods [57, 109, 144, 145], which lack rigorous guarantees, recent research have investigated formal XAI (FXAI) approaches, which are rigorous model-based methods for explainability [73, 85, 113–115]. The goal of FXAI is to offer rigorous and provable explanations for predictions made by ML models. The present theoretical framework is constructed upon two distinct types of formal explanations: *abductive explanations (AXp's)* and *contrastive explanations (CXp's)*, both of which focus on feature selection.

Abductive Explanations (AXp's). The definition of abductive Explanations (AXp's) for ML models is built on previous studies [16, 38, 76, 114, 160]. *Abductive explanations (AXp's)* are minimal subsets of features that are formally demonstrated to be sufficient for explaining an ML prediction, given a formal representation of the classifier of interest. Specifically, for a given instance $\mathbf{v} \in \mathbb{F}$ and prediction $c = \kappa(\mathbf{v})$, an AXp is a minimal subset of features $\mathcal{X} \subseteq \mathcal{F}$, such that

$$\forall(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \rightarrow (\kappa(\mathbf{x}) = c) \tag{2.12}$$

An AXp is are guaranteed to be a minimal subset of features satisfying Equation 2.12, and thus another term for an AXp is a prime implicant (PI) explanation [76]. Like other feature selection explanations, AXp's address "why" questions, by explaining why a particular prediction was made for a specific point in the feature space. An AXp provides an answer to the question by presenting a minimal or irreducible set of features that suffice to determine the prediction.

Contrastive explanations (CXp's): Another way to explain a model's behavior is to explore explanations to answer a "why not" question (why not another prediction was made), or a "how" question (how to change the prediction). Explanations addressing "why not" or "how" questions are known as *contrastive explanations (CXp's)* [78, 114, 123]. Following previous research, we define a CXp as a minimal subset of features that are necessary to change the model's prediction if allowed to change their values. Concretely, for a given instance $\mathbf{v} \in \mathbb{F}$ and prediction $c = \kappa(\mathbf{v})$, a CXp is a minimal subset of features $\mathcal{Y} \in \mathcal{F}$, such that

$$\exists(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i) \wedge (\kappa(\mathbf{x}) \neq c) \tag{2.13}$$

Minimal hitting set duality. Recent studies have revealed that there is a *minimal hitting set duality* relationship between a AXp's and CXp's for a specific instance $\mathbf{v} \in \mathbb{F}$ [78, 143]. This duality suggests that each AXp for a prediction $c = \kappa(\mathbf{v})$ serves as a

minimal hitting set (MHS) of the set of all CXp's for c, and vice versa: each CXp is an MHS of the set of all AXp's.

An increasing number of recent studies on formal explanations includes (but is not restricted to) works [8, 11, 14, 24, 25, 39, 49, 55, 110, 114, 117, 171].

2.3 SAT & MaxSAT

Numerous recent studies focusing on computing interpretable ML models [75, 79] and producing formal explanations [74, 82] have leveraged Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT) technology. This section introduces the necessary concepts related to SAT and MaxSAT. Here, we adopt standard definitions for SAT and MaxSAT solving [22].

2.3.1 Boolean Satisfiability (SAT)

In a Boolean satisfiability (SAT) problem [22], the input comprises a propositional formula over a set of propositional variables with the use of different logical operators on these variables. A propositional formula is considered to be in *conjunctive normal form* (CNF) if it consists of a conjunction (logical "and") of clauses, where each *clause* is a disjunction (logical "or") of literals. A *literal* is either a propositional variable b or its negation $\neg b$. Whenever convenient, a clause is regarded as a *set of literals*, and a CNF formula are considered as a *set of clauses*. A *truth assignment* assigns a value from $\{0, 1\}$ to each variable in a CNF formula. For a truth assignment, a clause is deemed satisfied if there exists at least one of its literals assigned value 1; otherwise, it is falsified by the assignment. A formula is considered satisfied if all of its clauses are satisfied; conversely, it is falsified if any of its clauses are falsified. If there is no assignment satisfying a CNF formula, the formula is deemed unsatisfiable. Solving a SAT problem is to determine whether there exists a truth assignment satisfying the entire formula, i.e. all clauses in the formula are satisfied.

2.3.2 Maximum Satisfiability (MaxSAT)

In the case of unsatisfiable formulas, the maximum satisfiability (MaxSAT) problem is introduced, which aims to identify a truth assignment that maximizes the number of satisfied clauses. Although numerous variants of MaxSAT are available [22, Chapters 23 and 24], the thesis is focused on Partial Unweighted MaxSAT, which can be formulated as follows. A formula Φ is composed of a conjunction of *hard* clauses $\mathcal{H}$ and

soft clauses $\mathcal{S}$, i.e. $\Phi = \mathcal{H} \wedge \mathcal{S}$ (or $\Phi = \mathcal{H} \cup \mathcal{S}$ in the set theory notation), where hard clauses $\mathcal{H}$ must be satisfied while soft clauses $\mathcal{S}$ indicate a preference for satisfying these clauses. The aim of the Partial Unweighted MaxSAT problem is to identify a truth assignment satisfying all the hard clauses and maximizing the total number of satisfied soft clauses. In the examination of an unsatisfiable formula Φ, there is often an interest in finding *minimal unsatisfiable subsets* (MUSes) and *minimal correction subsets* (MCSes) of Φ. These subsets can be defined as follows.

In the analysis of an unsatisfiable formula Φ, one is also often interested in identifying *minimal unsatisfiable subsets* (MUSes) and *minimal correction subsets* (MCSes) of Φ, which can be defined as follows.

Definition 2.5 (Minimal Unsatisfiable Subset (MUS))**.** Consider a formula consisting of an unsatisfiable set of clauses $\Phi = \mathcal{H} \cup \mathcal{S}$, where $\Phi \models \bot$. A subset of clauses $\mu \subseteq \mathcal{S}$ is seen as a *Minimal Unsatisfiable Subset* (MUS) iff $\mathcal{H} \cup \mu \models \bot$ and $\mathcal{H} \cup \mu' \not\models \bot$ holds for $\forall \mu' \subsetneq \mu$.

Informally, an MUS can be considered as a minimal explanation of the unsatisfiability of a formula Φ since it presents the minimal (irreducible) information required to be augmented to the hard clauses $\mathcal{H}$ in order to achieve unsatisfiability. Alternatively, there may be interest in *correcting* the formula by removing some clauses from $\mathcal{S}$ to obtain satisfiability.

Definition 2.6 (Minimal Correction Subset (MCS))**.** Consider a formula comprising an unsatisfiable set of clauses $\Phi = \mathcal{H} \cup \mathcal{S}$, i.e. $\Phi \models \bot$. A subset of clauses $\sigma \subseteq \mathcal{S}$ is considered as a *Minimal Correction Subset* (MCS) iff $\mathcal{H} \cup \mathcal{S} \setminus \sigma \not\models \bot$ and $\mathcal{H} \cup \mathcal{S} \setminus \sigma' \models \bot$ holds for $\forall \sigma' \subsetneq \sigma$.

Informally, an MCS can be viewed as the minimal approach to "correcting" the unsatisfiability of an unsatisfiable formula Φ. One of the key findings in analyzing unsatisfiable CNF formulas is the minimal hitting set (MHS) duality relationship between MUSes and MCSes [23, 143]. Concretely, given the sets of all MUSes and MCSes of formula Φ denoted as $\mathbb{U}_\Phi$ and $\mathbb{C}_\Phi$ respectively, we have $\mathbb{U}_\Phi = \text{MHS}(\mathbb{C}_\Phi)$ and $\mathbb{C}_\Phi = \text{MHS}(\mathbb{U}_\Phi)$, where $\text{MHS}(S)$ represents the minimal hitting sets of S, i.e. $\text{MHS}(S)$ is the minimal sets sharing at least one element with each subset in S. Formally, $\text{mins}(S) = \{s \in S \mid \forall t \subsetneq s,\ t \notin S\}$ identifies the subset-minimal elements of a set of sets, $\text{HS}(S) = \{t \subseteq (\cup S) \mid \forall s \in S,\ t \cap s \neq \emptyset\}$, and $\text{MHS}(S) = \text{mins}(\text{HS}(S))$. This finding of the MHS duality relationship has been widely used in developing algorithms for MUSes and MCSes [18, 102, 103], and has also applied in various other contexts. In recent years, there has been a surge in the development of novel algorithms for extracting and enumerating MUSes and MCSe [17, 21, 56, 103, 120, 121, 127, 134].

Chapter 3

Learning Optimal Decision Sets and Lists with SAT

This chapter is based on:

- Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic. Learning optimal decision sets and lists with sat. *Journal of Artificial Intelligence Research,* 72, 1251-1279, 2021.

The most interpretable ML models include decision sets and decision lists, since these models consist of straightforward logical rules. Decision sets offer the simplest explanation because when a rule that "fires" for a given data point, that rule alone suffices as the explanation, while for decision lists, i.e. ordered sets of rules, we have to also consider the order of rules in the model. To ensure these models are easily understandable to humans, their succinctness should be maximized. A recent study has proposed an method to compute decision sets, focusing initially on minimizing the number of rules and afterwards minimizing the number of literals. Additionally, there has been investigation [52, 112] into developing a CNF classifier that minimizes the number of literals within a fixed number of rules, aimed at generating interpretable decision rules to explain instances of the positive class. However, we argue that previous studies have not used the most best explainability measure. For example, two rules, each consisting of 80 conditions, are notably less interpretable than four rules, each comprising only 10 conditions. Motivated by this, we explore directly computing decision sets and lists that minimize their size, redefined as the total number of literals in the model. This lead to smaller decision sets and decision lists (in terms of literals), which are considered more appealing for explaining decisions. This chapter introduces methods for learning "perfect" decision sets and decision lists with minimum size, which are perfectly accurate on

the training data, with the use of SAT solving technology. Furthermore, this chapter presents a novel approach to identifying optimal sparse alternatives, trading of accuracy and size.

Chapter 4

From formal boosted tree explanations to interpretable rule sets

This chapter is based on:

- Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. From formal boosted tree explanations to interpretable rule sets. *In 29th International Conference on Principles and Practice of Constraint Programming,* vol. 280, pp.38:1-38:21, 2023.

Explaining decision sets is notably straightforward: the rule that "fires" a given instance serves as the explanation for that instance. This led to increased interest in decision sets that are both easy to understand to accurate. The method presented in Chapter 3 produces decision sets of minimum size that achieve perfect accuracy on the training data and demonstrates that decision sets that fully align with the training data exhibit superior accuracy compared to others. A more scalable method [80] to generate perfectly accurate decision sets was the introduced. However, neither of these methods can offer any decision information if a dataset is not entirely solved. Inspired by these studies and their limitations, this chapter focuses on establish a connection between formal post-hoc explainability and decision sets. Specifically, this chapter aims to develop an innovative anytime method to generate decision sets that are both accurate and interpretable. This is achieved by distilling a gradient boosted tree model into a decision set on demand with the use of abductive explanations (AXp's). In addition, the chapter introduces several post-hoc model reduction techniques aimed at improving the interpretability of the result decision sets with the use of MaxSAT and integer linear programming (ILP). Empirical results conducted on various datasets show that our method generates decision

sets outperforming those produced by state-of-the-art approaches in terms of accuracy, while remaining comparable regarding explanation size.

Chapter 5

Extracting and Applying Background Knowledge in the Context of Formal Explanations

This chapter is based on:

- Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, Nina Narodytska, and Joao Marques-Silva. Eliminating the impossible, whatever remains must be true: On extracting and applying background knowledge in the context of formal explanations. *In Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 37, pp. 4123-4131, 2023.

While the formal explainability method provides provably correct and minimal explanations, a few limitations have been identified. For instance, to ensure provable correctness of explanations, formal methods must consider the entire feature space, assuming that features are uniformly distributed and independent [171]. This requires a formal reasoner to examine all potential combinations of feature values, even those that are unlikely to occur in practical contexts. While this ensures the correctness of abductive explanations (AXp's), it may result in unnecessarily long explanations. Additionally, it raises concerns about the validity of contrastive explanations (CXp's), as the counterexamples they depend on may be not meaningful. Inspired by the limitation, this chapter is aimed at generating both AXp's and CXp's using background knowledge and presents the following contributions. First, this chapter introduces an efficient method to mine background knowledge represented by precise if-then rules for a given training dataset. This method builds on a recent technique to learn decision sets [80]. Second, we presents a novel method to produce AXp's and CXp's subject to extracted background

knowledge. Also, this chapter theoretically demonstrates that incorporating background knowledge enhances the quality of both AXp's and CXp's, and therefore helps establish trust in the underlying AI systems. Finally, inspired by [73], we argue that background knowledge facilitates a more precise assessment of the correctness of model-agnostic explainers by preventing the consideration of impossible combinations of feature values.

Chapter 6

On Formal Feature Attribution and Its Approximation

This chapter is based on:

- Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. On Formal Feature Attribution and Its Approximation. *arXiv preprint arXiv:2307.03380,* 2023.

Serving as the alternative of model-agnostic XAI methods, formal XAI (FXAI) approaches suffer from their own drawbacks, such as scalability issues and the need to construct a logical representation of ML models. Also, formal explanations are often larger compared to their model-agnostic counterparts as they do not take into account the reasoning about (unknown) data distributions. Lastly, and notably, FXAI approaches have not been applied to address feature attribution problems. Motivated by the aforementioned limitations, this chapter introduces a novel formal method to generate feature attribution, leveraging the achievements of established FXAI techniques [114]. Through exhaustive enumeration of all AXp's, we can define formal feature attribution (FFA) as the proportion of occurrences of a given feature within these AXp's. Arguably, computing formal feature attribution is hard for the second level of the polynomial hierarchy. While computing *exact* FFA can be challening, this chapter demonstrates that existing anytime formal explanation enumeration techniques can be effectively used to approximate FFA. Empirical results conduced on publicly accessible tabular and image datasets show the practical effectiveness of the proposed method and its advantage over SHAP and LIME, as well as in a real-world application of XAI in the field of software engineering [118, 133].

Chapter 7

Anytime Approximate Formal Feature Attribution

This chapter is based on:

- Jinqiang Yu, Graham Farr, Alexey Ignatiev, and Peter J. Stuckey. Anytime Approximate Formal Feature Attribution. *arXiv preprint arXiv:2312.06973,* 2023.

Chapter 6 provides a clear and crisp definition of FFA, but its computation presents challenges, since determining whether a feature has a non-zero attribution is as least as hard as determining its relevance. In Chapter 6, we demonstrate that FFA computation can be efficiently achieved by leveraging the hitting set duality between AXp's and CXp's. While attempting to enumerate CXp's, a side effect of the algorithm is the discovery of AXp's. Indeed, the algorithm typically identifies numerous AXp's before encountering the first CXp. In this case, the AXp's at the beginning are ensured to be diverse, as they must be broad in scope to guarantee that the CXp is large enough to hit all relevant AXp's for exact FFA. Therefore, collecting AXp's as side effects of CXp enumeration proves effective in the initial stages of the enumeration process. However, as we collect an increasing number of AXp's as side effects, we eventually reach some point where significantly more CXp's are generated than AXp's. Experimental results indicate that if we aim to enumerate all AXp's, it is preferable not to depend on the side effect behavior, but rather to directly enumerate AXp's. This presents a dilemma: for fast and accurate approximations of FFA, we enumerate CXp's and generate AXp's as a side effect, while to produce exact FFA, we compute all AXp's, and it is more advantageous to directly enumerate them. In this chapter, we introduce an anytime method to produce approximate FFA, where we initially enumerate CXp's and then switch to AXp enumeration dynamically when the rate of AXp discovery through CXp

enumeration decrease. Thus, we can quickly obtain accurate approximations while also accelerating the process of arriving at the complete set of AXp's compared to pure CXp enumeration. The second contribution of this work involves exploring this alternative approach and demonstrating that even with a(n) (in)complete set of CXp's provided, determining FFA remains computationally challenging, as it is #P-hard even when all CXp's are of size two.

Chapter 8

A Formal Explainer for Just-In-Time Defect Predictions

This chapter is based on:

- Jinqiang Yu, Michael Fu, Alexey Ignatiev, Chakkrit Tantithamthavorn, and Peter J. Stuckey. A Formal Explainer for Just-In-Time Defect Predictions. *ACM Transactions on Software Engineering and Methodology,* 2024. (**Accepted.**)

Software companies often rapidly release software products at a rapid pace in short-term development cycles. The rapid pace of software release presents significant challenges because of the exponential increase of highly complex source code, particularly for under-resourced software quality assurance (SQA) teams. Due to the time-consuming and costly nature of various SQA activities, e.g., code review, developers are unable to thoroughly guarantee the highest quality for all newly developed code commits or pull requests given limited time and resources. Just-in-Time (JIT) defect prediction [89, 92, 104, 132] is proposed to predict whether a commit will introduce software defects in the future. This allows teams to allocate their finite SQA resources to the most critical commits or pull requests. However, JIT defect prediction largely remains a black-box approach, offering predictions that lack explainability and actionable insights for practitioners. Previous research has applied a range of model-agnostic techniques to explain the predictions made by JIT models. Unfortunately, explanations produced by these methods are still not formally sound, robust, and actionable. Motivated by the limitation of model-agnostic approaches, this chapter introduces FoX, a Formal eXplainer for JIT defect prediction. With the use of formal reasoning about the functionality of JIT defect prediction models, FoX can offer explanations that are not only provably correct but also guaranteed to be minimal. Experimental results demonstrate that FoX

can effectively produce explanations that are provably correct, correct, and actionable. In contrast, explanations generated by model-agnostic methods do not hold this The survey conducted among 54 software practitioners offers valuable insights into the usefulness and trustworthiness of FoX. 74% of respondents found it to be trustworthy, and 86% agreed with the usefulness of FoX. Hence, this chapter serves as a significant advancement towards providing trustable explanations for JIT models, enabling domain experts and practitioners to gain a clearer understanding of why a commit is predicted as defective and how to mitigate the risk.

A Formal Explainer for Just-In-Time Defect Predictions

JINQIANG YU, Monash University, Australia and Australian Research Council OPTIMA ITTC, Australia
MICHAEL FU, Monash University, Australia
ALEXEY IGNATIEV, Monash University, Australia
CHAKKRIT TANTITHAMTHAVORN, Monash University, Australia
PETER J. STUCKEY, Monash University, Australia and Australian Research Council OPTIMA ITTC, Australia

Just-In-Time (JIT) defect prediction has been proposed to help teams to prioritize the limited resources on the most risky commits (or pull requests), yet it remains largely a black-box, whose predictions are not explainable nor actionable to practitioners. Thus, prior studies have applied various model-agnostic techniques to explain the predictions of JIT models. Yet, explanations generated from existing model-agnostic techniques are still not formally sound, robust, and actionable. In this paper, we propose FoX, a Formal eXplainer for JIT Defect Prediction, which builds on formal reasoning about the behaviour of JIT defect prediction models and hence is able to provide provably correct explanations, which are additionally guaranteed to be minimal. Our experimental results show that FoX is able to efficiently generate provably-correct, robust, and actionable explanations while existing model-agnostic techniques cannot. Our survey study with 54 software practitioners provides valuable insights into the usefulness and trustworthiness of our FoX approach. 86% of participants agreed that our approach is useful, while 74% of participants found it trustworthy. Thus, this paper serves as an important stepping stone towards trustable explanations for JIT models to help domain experts and practitioners better understand why a commit is predicted as defective and what to do to mitigate the risk.

ACM Reference Format:
Jinqiang Yu, Michael Fu, Alexey Ignatiev, Chakkrit Tantithamthavorn, and Peter J. Stuckey. xxxx. A Formal Explainer for Just-In-Time Defect Predictions. 1, 1 (April xxxx), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Modern software companies often continuously release software products at a rapid pace in short-term cycles. Such rapid-release software development often poses the greatest challenges to under-resourced Software Quality Assurance (SQA) teams due to the exponential growth of highly-complex source code. Since various SQA activities are time-consuming and expensive (e.g., code review), developers cannot exhaustively ensure that all newly developed code commits or pull requests are of the highest quality given the limited time and resources.

Just-in-time (JIT) defect prediction [24, 27, 33, 41] has been proposed to predict if a commit will introduce software defects in the future, enabling teams to prioritize their limited SQA resources on the most risky commits/pull requests. In the past decades, various Artificial Intelligence/Machine

Authors' addresses: Jinqiang Yu, jinqiang.yu@monash.edu, Monash University, Australia and Australian Research Council OPTIMA ITTC, Australia; Michael Fu, yeh.fu@monash.edu, Monash University, Australia; Alexey Ignatiev, alexey.ignatiev@monash.edu, Monash University, Australia; Chakkrit Tantithamthavorn, chakkrit@monash.edu, Monash University, Australia; Peter J. Stuckey, peter.stuckey@monash.edu, Monash University, Australia and Australian Research Council OPTIMA ITTC, Australia.

XXXX-XXXX/xxxx/4-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

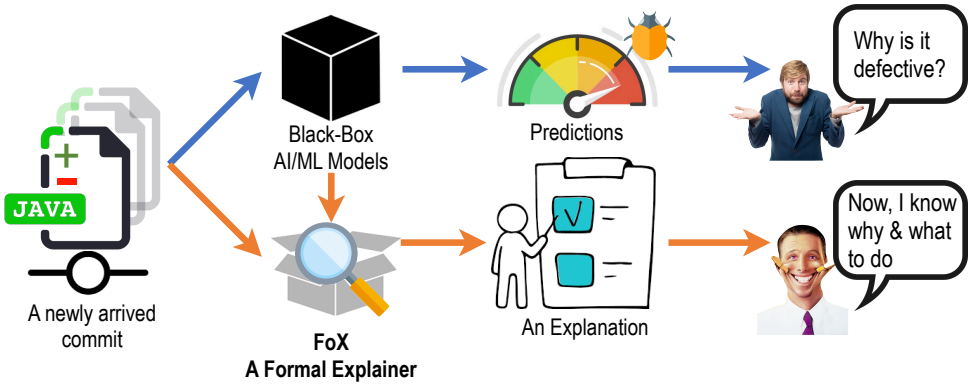

Fig. 1. A motivation on the need of Explainable AI for Just-In-Time defect prediction. Currently, practitioners and domain experts often ask many *why?* questions (e.g., why a commit is predicted as defective) and *how?* questions (e.g., how should developers mitigate the risk) [23].

Learning (AI/ML) techniques have been used to build JIT defect prediction models. Such defect prediction technologies have empowered many recent AI-powered code quality tools (e.g., Microsoft's AI Code Defect,[1] Amazon's CodeGuru,[2] CQSE GmbH's TeamScale,[3] Sealight's Code Quality Intelligence,[4] and CodeScene's Code Quality Analytics[5]), demonstrating the significance of Just-in-time (JIT) defect prediction to real-world practice.

Recently, the *explainability* and *actionability* of AI/ML models in SE have become one of the most challenging research problems that remain largely unsolved. Importantly, most of the current JIT defect prediction models are often treated as a black-box. Their predictions are not explainable nor actionable to practitioners (see Figure 1)—i.e., they are not able to provide clear evidence to explain why the given instance is predicted as defective or not defective. Particularly, practitioners and domain experts often asked many *why?* questions (e.g., why a commit is predicted as defective) [11, 22, 23, 56] and *how?* questions (e.g., how should developers mitigate the risk) [7, 30, 32, 40, 42, 43, 60]. In addition, due to the growing size and complexity of modern AI/ML-based JIT defect models, it is also very difficult for domain experts to understand how the models work and whether the models are trained correctly. Thus, a lack of explainability and actionability can lead to a lack of trust in JIT defect prediction models, hindering their adoption in practice [11, 22, 23, 55].

The state-of-the-art in the explainability of JIT defect prediction models is represented by local model-agnostic approaches. One successful local model-agnostic approach referred to as PyExplainer has been recently proposed in an award-winning paper at ASE'21 [42]. PyExplainer's explanations are greatly beneficial to help developers better understand what features contributed the most to the predictions, enabling teams to focus on the most important aspects that are associated with software defects instead of focusing on the less important ones. The key principle of model-agnostic techniques (these also include the prominent explainers LIME [45], SHAP [34] and Anchors [46] among many others) is to generate explanations based on extensive sampling in the vicinity of the concrete instance being explained. The sampling procedure randomly generates a large number of synthetic instances and results in either directly reporting a conjunction of

[1]https://www.microsoft.com/en-us/ai/ai-lab-code-defect
[2]https://aws.amazon.com/codeguru/
[3]https://www.cqse.eu/en/teamscale/overview/
[4]https://www.sealights.io/product/release-quality-analysis/
[5]https://codescene.com/

feature values deemed relevant for the prediction [46], or training a surrogate local model on these synthetic instances [34, 45], which aims at approximating the original model.

Unfortunately, the reliance on extensive sampling although a vital component in making these explainers model-agnostic also makes them in general *incorrect* [16, 17, 39] and in particular susceptible to *out-of-distribution attacks* [31, 50]. Finally, the random nature of the sampling procedure results in the model-agnostic explanations being *non-robust* [2, 22, 49, 51], i.e. running the same explanation method on the same instance multiple times may produce different explanations. These issues may negatively impact the operational decision-makings of under-resourced SQA teams and exacerbate the problem of trust in AI.

Motivated by the aforementioned limitations of the sampling-based methods, *in this paper*, we propose FoX, a Formal eXplainer for Just-In-Time Defect Prediction, which builds on *formal reasoning* about the behavior of JIT defect prediction models and hence is able to provide a user with *provably correct* explanations, which are additionally guaranteed to be minimal. FoX leverages the formal explainability principles that have not been explored in software engineering [19, 20, 35, 48] in order to compute a single *abductive* explanation, i.e. answering *why a model predicted (or not) a commit as defect-introducing*, and a single *contrastive* explanation (aka. counterfactual explanations), i.e. saying *what developers should do to reverse the prediction of the model*. Furthermore, FoX can provide a user with a given number of both abductive and contrastive explanations, or enumerate them exhaustively [18] as well as report *formal feature attribution* [62], i.e. a list of numeric values representing *how important* the corresponding features are for a given prediction. Note that FoX hinges on the use of formal methods applied to the original JIT defect prediction models, and so on the success of modern propositional satisfiability (SAT) solving [3]. In particular, given a model and an instance both encoded into propositional logic, FoX makes a series of SAT oracle calls for computing the requested number of abductive and/or contrastive explanations for the prediction made. Thanks to the use of logic, the approach is able to capture the behavior of the original model and hence the explanations it reports are guaranteed to be formally correct, i.e. they hold in the *entire feature space*.

In light of the theoretical correctness guarantees of formal explanations [36], we experimentally evaluate FoX along two additional dimensions: robustness and speed, and compare with four state-of-the-art model-agnostic techniques i.e., LIME [45], SHAP [34], Anchor [46], PyExplainer [42]. By evaluating FoX and the competitors on two JIT defect prediction models (i.e., Logistic Regression and Random Forest) that are trained on a total of 40,798 commits from two large-scale software systems (i.e., OpenStack and Qt), the results show that (1) model-agnostic explanations cannot compete with formal explanations generated by FoX in terms of formal correctness; (2) in stark contrast to heuristic explanations, formal explanations of FoX are 100% robust (i.e., deterministic); and (3) explanations of FoX can be efficiently computed and enumerated, i.e. explanation generation for both LRs and RFs requires less than a second. Thus, we conclude that FoX is able to efficiently generate provably-correct, robust, and actionable explanations, addressing various limitations raised by prior studies of explainable defect prediction [2, 22, 49, 51]. These explanations are expected to help domain experts and practitioners better understand why a commit is predicted as defective through abductive explanations and what to do to mitigate the risk of having defects in the future thanks to contrastive explanations. To explore the practicality of our FoX approach, we conduct a user study with 54 software practitioners. Specifically, we evaluated the usefulness and trustworthiness of our approach. Our survey indicates that the explanation generated by FoX for JIT defect predictions improves the usefulness from 72% to 86% and improves the trustworthiness from 53% to 74% based on practitioners' perceptions. These results highlight the practicality of our FoX approach that can enhance the trust between JIT defect prediction and software practitioners.

Novelty. To the best of our knowledge, this paper is the first to:

- Present FoX – The first Formal eXplainer for Just-In-Time Defect Prediction that leverages the formal explainable AI principles [12, 18–20, 35, 48], providing provably correct and succinct explanations that can answer not only *why?* and *how?* questions but also giving formally defined feature importance scores [62].
- Empirically demonstrate that FoX is able to efficiently generate robust and actionable explanations while the four state-of-the-art model-agnostic techniques that were previously used in defect prediction (i.e., LIME [45], SHAP [34], Anchor [46], PyExplainer [42]) cannot, besides lacking formal correctness.

Open Science. To facilitate the reusability and replicability of our work, we provide a public replication package[6]. To ease the adoption of FoX by practitioners, we publish FoX as a Python Package[7], which is available in both `conda` and `pip` (Package Installer for Python). The FoX Python package is also well-tested, achieving a code coverage of 98% measured by CodeCov with A+ quality and 0 alerts graded by LGTM.

Paper Organization. Section 2 motivates this work and discusses the limitations of prior approaches. Section 3 defines the necessary concepts and argues how formal explanations address the aforementioned limitations. Section 4 describes the proposed approach. Section 5 details the experimental setup while Section 6 presents the results. Section 8 outlines related work. Section 9 discusses the limitations and future work, while Section 10 concludes the paper.

2 BACKGROUND & MOTIVATION

In this section, we motivate and formulate the problem with respect to the findings of prior work based on a few illustrative examples.

The explainability of AI models is one of the grand research challenges [54] for AI in SE (see http://xai4se.github.io), since practitioners often do not trust the predictions [11, 55, 56, 56, 60], hindering the adoption of AI-powered software development tools in practice. Recently, Explainable AI has been actively investigated in the domain of defect prediction [7, 22, 23, 30, 40, 42, 43, 56]. For example, recent works have shown some successful case studies to make defect prediction models more practical [41, 58], explainable [22, 26], and actionable [42, 43].

The heart of Explainable AI for SE is the use of model-agnostic techniques to explain the predictions of AI/ML models. Examples of widely-used model-agnostic techniques in SE include LIME [45], SHAP [34], Anchors [46], and PyExplainer [42]. Since many AI/ML models are treated as a black-box, the primary objective of model-agnostic techniques is to build a local explainable model that mimics the behaviors of the global black-box model for the prediction of an instance to be explained.

Such model-agnostic techniques often consists of four steps: (1) generate synthetic instances around an instance to be explained; (2) obtain the predictions of such synthetic instances using the global model; (3) build a local explainable model to learn the relationship between synthetic instances and their predictions from the global model; and (4) generate an explanation from the local explainable model. Depending on the form of explanations model-agnostic approaches offer, they are conventionally classified as *feature selection* or *feature attribution* approaches briefly discussed below. Both lines of work aim at identifying the *most important features*, which are a set of features (i.e., independent variables) that have the strongest influence on the model prediction for a given

[6]https://github.com/trustablefox/exp_replication
[7]https://github.com/trustablefox/foxplainer

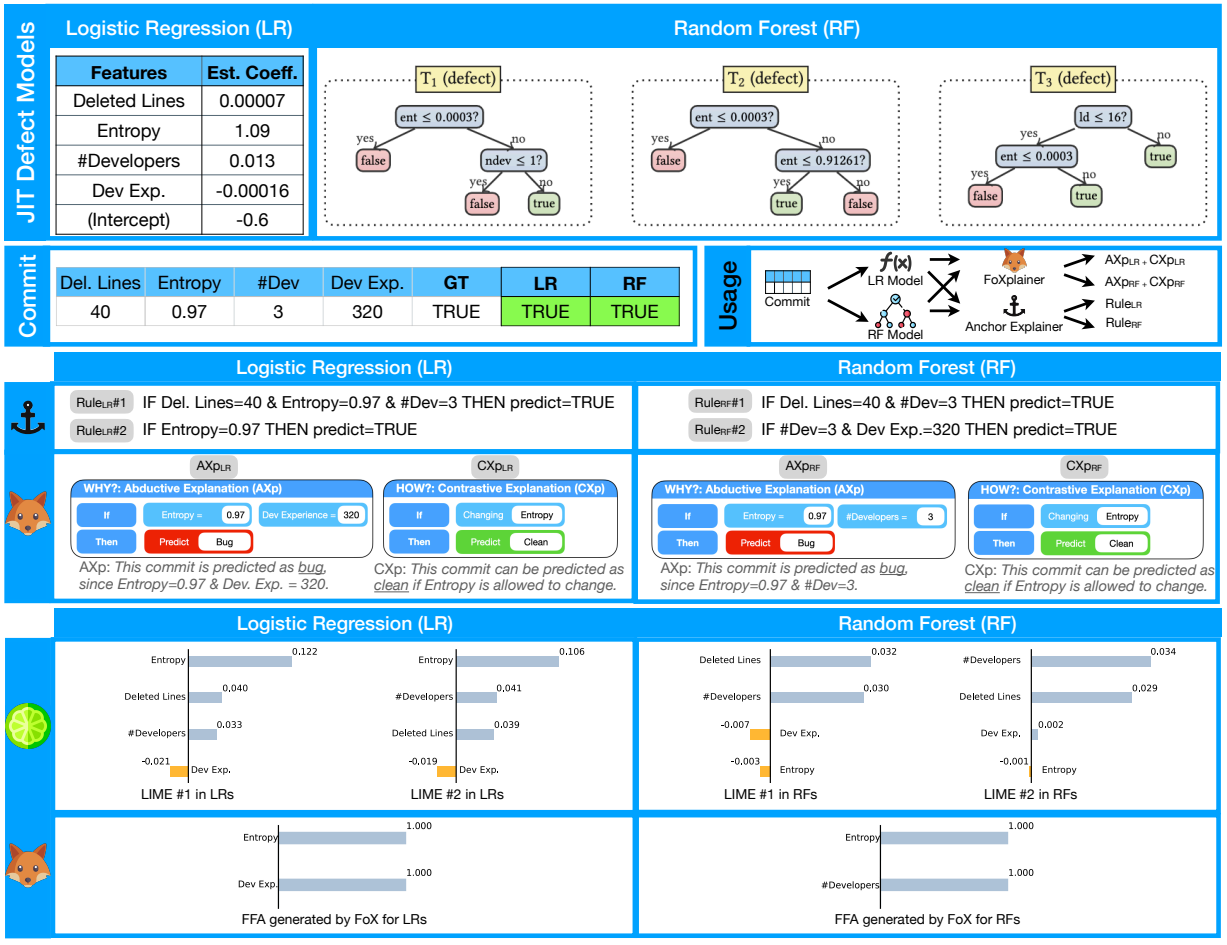

Fig. 2. An illustrative example of explanations generated by FoX, Anchor [46], and LIME [45] for Just-In-Time defect prediction. FFA refers to Formal Feature Attribution (see Definition 3).

instance (i.e., the strongest relationship between features and a prediction outcome), meaning that features that are not important must have little to no influence on the model prediction. Note that both forms of model-agnostic techniques often suffer from a few key limitations also detailed below.

Feature Selection. A feature selection[8] approach identifies subsets of features that are deemed *responsible* for a given prediction. (Two well-known examples of feature selection approaches are Anchor [46] and PyExplainer [42].) The sufficiency of the selected set of features for a given prediction is determined by these explainers statistically based on extensive sampling around the instance of interest, by assessing a few measures like *fidelity*, *precision*, among others.

Feature Attribution. A different view on post-hoc explanations is provided by feature attribution approaches, e.g. LIME [45] and SHAP [34]. Based on random sampling in the neighborhood of the target instance, these approaches attribute responsibility to all model's features by assigning a numeric value of importance to each feature. Given these importance values, the features can then be ranked from most important to least important. As a result, given a set of features $\mathcal{F}$, a feature attribution explanation is conventionally provided as a linear form $\sum_{i \in \mathcal{F}} w_i \cdot x_i$, which can be also

[8]Hereinafter, the term *feature selection* denotes the principles underlying a family of ML explanation approaches rather than the techniques *of the same name* used in the context of data mining.

seen as approximating the original black-box model in the *local* neighborhood of instance being explained. Among other feature attribution approaches, SHAP [34] is often claimed to stand out as it aims at approximating Shapley values, a powerful concept originally introduced by L. Shapley in the context of cooperative games in game theory [47] as a mechanism for assessing player importance in cooperative games.

Limitation ①: Explanations are not formally correct. Explanations generated by existing model-agnostic techniques heavily rely on extensive sampling and various interpretable ML models (e.g., linear regression, LASSO, decision tree) to be used to explain the behavior of the original black-box model. However, due to their statistical nature such model-agnostic techniques do not have a mechanism to provably guarantee that explanations are correct with respect to the original model. For instance, Ignatiev [17] has recently argued that in the context of general AI classification tasks, most prominent model-agnostic explainers report explanations that do not logically capture the semantics of the original models and are likely to be incorrect. Similarly, Lakkaraju *et al.* [31, 50] showed how out-of-distribution sampling can be used to fool a model-agnostic explainer. Finally, Huang and Marques-Silva [16] argued that SHAP may often fail to give a correct estimation of the actual feature importance thus failing to produce reasonable explanations, both in terms of attribution values and in terms of feature ranking. As a result, the set of most important features that such explainers claim to be the reason for the prediction may in fact be insufficient for the prediction (i.e. some of the other important features are missing), or it may be too conservative (i.e. it contains features that are irrelevant for the prediction).

An Illustrative Running Example. Consider example logistic regression (LR) and random forest (RF) based JIT defect prediction models shown in Figure 2. These models are trained using 4 selected features (i.e., *ld*, *ent*, *ndev* and *dev_exp*) on the training dataset of the Qt project.[9] Given an example commit $\mathbf{v}$ (#lines_deleted = 40, entropy = 0.97, #developers = 3, #developer_experience = 320), the commit is predicted as *true* by both JIT defect models, meaning that this commit is likely to introduce defects in the future based on the given characteristics.

Example 1 (Incorrect Explanation)**.** *First, let us consider a model-agnostic feature selection explanation generated by Anchor [46] for the LR prediction generated as "IF* ld = 40 & ent = 0.97 & ndev = 3 *THEN prediction is* true*", meaning that the LR model shown in Figure 2 predicts this commit as* true *due to these three features. This indicates that these three features alone (i.e.,* ld*,* ent*, and* ndev*) are* sufficient *for the given commit prediction made by the model. In other words, other features* can be excluded *from consideration as having no effect on the prediction. However, this explanation is* incorrect *because there exists at least one* counterexample *instance compatible with the explanation on all the selected features that is still predicted differently. For example, by changing the value 320 of* dev_exp *to 3,400, we get the JIT model prediction* false*. This illustrates that feature* ld*, which is claimed by Anchor to be irrelevant, still affects the model's prediction for the instance being explained. Observe how the counterexample constructed reveals incorrectness of the explanation.*

Now, consider a feature attribution explanation that LIME [45] computes for the RF prediction: {ld: 0.032, ndev: 0.030, dev_exp: -0.007, ent: -0.003}*. This explanation indicates that all features contribute to the prediction although two of them exhibit negative contributions. Observe that the feature* dev_exp *is not included in the example RF model and thus it should not have any contribution to the prediction. However, LIME fails to generate a correct explanation since it assigns a non-zero attribution value to the feature* dev_exp *in the explanation.* □

Limitation ②: Explanations are not robust. Explanations generated by existing model-agnostic techniques heavily rely on synthetic instances that are randomly generated around

[9]For simplicity, all but 4 features are discarded here. Note that the experimental results shown in Section 6 address the *original* datasets, with no simplification involved.

an instance to be explained. Similarly, prior studies [2, 22, 49, 51] also raised concerns that such model-agnostic techniques are often non-deterministic. Thus, the randomness within the neighbourhood generation process may produce different sets of synthetic instances, which often leads to producing different explanations.

Example 2 (Non-robust Explanations)**.** *Given our running example commit (#lines_deleted = 40, entropy = 0.97, #developers = 3, #developer_experience = 320) and the RF model, Figure 2 shows that if Anchor [46] runs from scratch twice, two different explanations are computed, i.e. "IF* ld = 40 & ndev = 3 *THEN prediction is* true*" and "IF* ndev = 3 & dev_exp = 320 *THEN prediction is* true*", which exemplifies the issue of non-robustness. A similar issue can be observed in LIME [45]. As depicted in Figure 2, LIME generates different feature attribution explanations when running from scratch twice, i.e.* {ld: 0.032, ndev: 0.030, dev_exp: -0.007, ent: -0.003} *and* {ld: 0.029, ndev: 0.034, dev_exp: 0.002, ent: -0.001}*. Importantly, these explanations not only offer different feature attributions but also result in different feature rankings based on those attributions.* □

Limitation ③: Explanations are not actionable. Explanations generated by existing model-agnostic techniques for JIT defect predictions only focus on answering *why?* questions (e.g., why a commit is predicted as defective) [11, 22, 23, 56]. However, the explanations for the *how?* questions (e.g., how should developers proceed to mitigate the risk) that are desirable by practitioners [7, 30, 32, 40, 42, 43, 60] have not yet been effectively generated. Observe that Anchor's explanations in the examples above, while being incorrect and non-robust, may help a user understand *why* the corresponding prediction was made by the model – but they provide the user with no clue of *how* the prediction could be changed.

3 PRELIMINARIES

Given the significant limitations yet high impact of prior work [42, 45, 46], we propose FoX, a practical, formal reasoning-based approach capable of generating both provably correct *abductive* and *contrastive explanations* for JIT defect models. Below, we present the usage scenario of the proposed approach followed by its technical details.

3.1 Usage Scenario

In software development, the ML-based defect prediction bot could offer significant value in improving code quality. In particular, our approach can be integrated seamlessly with GitHub commit actions, which functions as a proactive tool to identify potentially defective commits in real time.

Upon a developer pushing a commit to a repository on GitHub, FoX is automatically triggered to analyze whether this commit is defective using ML models. Subsequently, the prediction results are displayed directly within the GitHub interface (as shown in Figure 10), offering insightful explanations for defective commits. Specifically in this example, FoX offers model explanations including factors such as low relative reviewer experience, low reviewer awareness, and high commit age. These explanations help clarify why the ML model generates such a prediction. Furthermore, actionable guidance is offered to support developers mitigate identified risks. In particular, developers are advised to assign more experienced developers for review, enhance reviewer awareness through training and communication, and ensure frequent updates for ongoing scrutiny and validation.

Upon receiving the prediction results, explanations, and mitigation strategies, developers can acknowledge the suggestions and take necessary actions accordingly. In conclusion, by integrating the ML-based explainable defect prediction into the software development workflow, development

teams can proactively identify and address potential defects, thereby enhancing the overall quality of their software products.

3.2 Necessary Notation

First, let us formally define a JIT defect prediction model. A JIT defect prediction model is defined under the standard classification scenario characterized by a set of features $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_0, c_1\}$, where $c_0 = \mathit{false}$ and $c_1 = \mathit{true}$, i.e. non-defective and defective. Let the domain of feature $i \in \mathcal{F}$ be $\mathcal{D}_i$ and so the complete space of feature values is $\mathbb{F} = \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_m$. A specific point in feature space $\mathbb{F}$, also referred to as an *instance*, is denoted by $\mathbf{v} = \langle v_1, \ldots, v_m \rangle$ and represents an individual commit. In general, an arbitrary point in feature space is denoted by $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$, where each variable $x_i \in \mathbf{x}$ takes values from its domain $\mathcal{D}_i$ and represents feature $i \in \mathcal{F}$. As a result, an ML classifier is assumed to define a classification function: $\tau : \mathbb{F} \to \mathcal{K}$.

In this paper, we focus on the two following classes of ML models. First, the *(binary) logistic regression* (LR) model predicts a commit $\mathbf{v} \in \mathbb{F}$ by evaluating the probability $p(\mathbf{v})$ of class *true*, given the log-odds $f(\mathbf{v})$ of this class, which is a linear combination of a set of features $\mathcal{F}$ and an intercept w_0, where each feature $i \in \mathcal{F}$ is associated with a coefficient $w_i \in \mathbb{R}$. Second, the *random forest* (RF) model [4] is an ensemble model that consists of decision trees T for a set of classes $\mathcal{K}$. Given a commit $\mathbf{v} \in \mathbb{F}$, each decision tree assigns a class to $\mathbf{v}$, and the final RF prediction for $\mathbf{v}$ is determined as the majority vote over all the classes assigned by the individual trees.

3.3 Formal Explanations

Let us define formal *abductive* (AXp) and *contrastive* (CXp) explanations and argue how they address the aforementioned limitations. Observe that both types of formal explanations below exploit the concept of *subset-minimality*. A set $\mathcal{S}$ is called subset-minimal with respect to some property $\mathfrak{P}$ if property $\mathfrak{P}$ holds for $\mathcal{S}$ but it does not hold for any of its *strict* subsets, i.e. for $\mathcal{S}' \subsetneq \mathcal{S}$. Note that the role of property $\mathfrak{P}$ in the definitions below is played by the corresponding conditions (1) and (2).

Definition 1: Abductive Explanation (AXp). Building on earlier work [12, 19, 36, 48], an AXp is defined as a subset-minimal set of features *sufficient* to explain the JIT defect prediction made by an ML model τ. More formally, given an instance $\mathbf{v} \in \mathbb{F}$ and a JIT defect prediction $c = \tau(\mathbf{v})$, an AXp $\mathcal{X}$ is a subset-minimal set of features, such that the following holds:

$$\forall (\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \right] \rightarrow (\tau(\mathbf{x}) = c) \tag{1}$$

Informally, (1) states that given an AXp $\mathcal{X}$ for the prediction $c = \tau(\mathbf{v})$, for any instance $\mathbf{x}$ in the *entire* feature space $\mathbb{F}$ *compatible* with instance $\mathbf{v}$ on the features of $\mathcal{X}$, classifier τ is guaranteed to predict c, no matter what values the other features (excluded from AXp $\mathcal{X}$) take in instance $\mathbf{x}$.

Example 3:. *Consider our running example. By examining the Qt dataset, the lower and upper bounds for the domains of the features considered are as follows:* $0 \leq x_{ld} \leq 309,728$, $0.0 \leq x_{ent} \leq 1.0$, $0 \leq x_{ndev} \leq 313$, *and* $0 \leq x_{dev\_exp} \leq 3,419$. *Given the possible values of the features, observe that a valid AXp* $\mathcal{X}$ *for the LR model's* τ_{lr} *prediction is shown in Figure 2. This AXp indicates that specifying* $ent = 0.97$ *and* $dev\_exp = 320$ *guarantees that the prediction for any compatible instance must be true, independently of the values of the other features, i.e. features* ld *and* ndev. *Indeed, even when* x_{ld} *and* x_{ndev} *take their lower bound values (both are 0), although the probability* $p(\mathbf{x})$ *of* true *gets smaller, it is still greater than 0.5 and, thus,* $\tau_{lr}(\mathbf{x}) = true$. *Similar reasoning can be applied in the case of the RF model, where the majority of the trees are guaranteed to predict* true *as long as* $ent = 0.97$ *and* $ndev = 3$. □

Clearly, the requirement imposed by (1) guarantees that the feature values of explanation $\mathcal{X}$ are sufficient for prediction c to hold in the entire feature space $\mathbb{F}$, i.e. there is no counterexample instance in $\mathbb{F}$ that would be compatible with explanation $\mathcal{X}$ but predicted differently than c. Hence, by definition, abductive explanations are *formally correct*. This is not the case for model-agnostic explanations, as was illustrated above for Anchor.

Note that multiple AXps may exist given an instance $\mathbf{v} \in \mathbb{F}$. Although not guaranteed to be robust by definition, AXps computed by the proposed algorithms are robust due to the deterministic nature of the underlying reasoners used. This is also confirmed by our experimental results provided in Section 6.

Finally, although AXps address 2 out of 3 limitations above, they can only provide a user with an indication for *why* a certain prediction was made, i.e. they are not designed to answer a *how?* question. This is what contrastive explanations defined below are capable of.

Definition 2: Contrastive Explanation (CXp). Following recent work [18, 36, 38], a CXp is a subset-minimal set of features that, if allowed to change their values, are *necessary* to flip the prediction provided by the JIT defect prediction model. Formally, given a JIT defect prediction $c = \tau(\mathbf{v})$, a CXp $\mathcal{Y}$ is a subset-minimal set of features, such that the following holds:

$$\exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i)\right] \land (\tau(\mathbf{x}) \neq c) \tag{2}$$

In other words, if (2) holds for prediction $c = \tau(\mathbf{v})$, it means that there exists an instance $\mathbf{x}$ in the feature space $\mathbb{F}$ where all the features $i \notin \mathcal{Y}$ are equal to the values of instance $\mathbf{v}$ being explained but the features of $\mathcal{Y}$ are different, such that $\tau(\mathbf{x}) \neq c$.

Example 4. *Consider the instance* $\mathbf{v}$ *from* (3) *classified as* true *by both LR and RF models shown in Figure 2. Figure 2 shows a valid CXp* $\mathcal{Y} = \{ent\}$ *computed for the prediction made by the LR model* τ_{lr}*, since this prediction can be flipped if this feature is allowed to take another value from its domain despite other features equating to the values of* $\mathbf{v}$*. Namely, if the value of* ent *is changed to* 0.0 *given* $0.0 \leq x_{ent} \leq 1.0$*, the probability of* true *prediction is* $p(\mathbf{x}) < 0.5$*, i.e. the prediction is changed to* false*. Similarly, the same CXp (Figure 2) for the RF model indicates that changing the value of feature* ent *to* 0.0 *forces the 3 trees of the RF model* τ_{rf} *to predict* false*,* false *and* true*, respectively, which results in the majority vote (and so the overall RF prediction)* false. □

Observe that the *formal correctness* observations can be made with respect to contrastive explanations too, using the same rationale. Indeed, the validity of (2) guarantees the existence of an instance that is classified differently even though it is close to the original instance $\mathbf{v}$. The algorithms we apply for computing CXps have a deterministic nature and thus the CXps we compute are guaranteed to be *robust* (this is also confirmed by our experimental results). Finally, contrastive explanations are designed to provide a user with an answer to the *how?* question, i.e. how the instance needs to be changed in order to change the prediction. Furthermore, subset-minimality of a CXp implies that the changes suggested by the CXp are *minimal* with respect to the original instance $\mathbf{v}$.

Finally, recent work has shown that given a prediction $c = \tau(\mathbf{v})$, AXps and CXps are related through the *minimal hitting set duality* [18, 44]. Given a set of sets $\mathbb{S}$, a *hitting set* of $\mathbb{S}$ is a set H such that $\forall S \in \mathbb{S}, S \cup H \neq \emptyset$, i.e. H "hits" every set in $\mathbb{S}$. A hitting set H for $\mathbb{S}$ is *minimal* if none of its strict subsets is also a hitting set. The minimal hitting set duality represents the fact that given a prediction $c = \tau(\mathbf{v})$, each AXp for the prediction is a *minimal hitting set* (MHS) of the set of all CXps $\mathbb{C}_\tau(\mathbf{v}, c)$ for that prediction, and the other way around: each CXp is an MHS of the set of all AXps $\mathbb{A}_\tau(\mathbf{v}, c)$. By slightly abusing the notation, $\mathbb{A}_\tau(\mathbf{v}, c) = \text{MHS}(\mathbb{C}_\tau(\mathbf{v}, c))$ and $\mathbb{C}_\tau(\mathbf{v}, c) = \text{MHS}(\mathbb{A}_\tau(\mathbf{v}, c))$. The algorithms we use in our approach heavily rely on this duality relation.

Algorithm 1 MARCO-like Anytime Explanation Enumeration

```
1:  procedure XpEnum(τ, v, c)
2:      (𝔸, ℂ) ← (∅, ∅)                          ▷ Sets of AXp's and CXp's to collect.
3:      while resources available do
4:          𝒴 ← MinimalHS(𝔸, ℂ)                  ▷ Get a new MHS of 𝔸 subject to ℂ.
5:          if 𝒴 = ⊥ then break                  ▷ Stop if none is computed.
6:          if ∃(x ∈ 𝔽). ⋀_{i∉𝒴}(x_i = v_i) ∧ (τ(x) ≠ c) then   ▷ Check CXp condition (2) for 𝒴.
7:              ℂ ← ℂ ∪ {𝒴}                      ▷ 𝒴 appears to be a CXp.
8:          else                                 ▷ There must be a missing AXp 𝒳 ⊆ ℱ \ 𝒴.
9:              𝒳 ← ExtractAXp(ℱ \ 𝒴, τ, v, c)   ▷ Get AXp 𝒳 by iteratively checking (1) [19].
10:             𝔸 ← 𝔸 ∪ {𝒳}                      ▷ Collect new AXp 𝒳.
        return 𝔸, ℂ
```

Example 5. *Consider our running example. There is a single AXp for the target instance and so the full set of AXps for the RF prediction of instance* $\mathbf{v}$ *is* $\mathbb{X} = \{\{ent, ndev\}\}$*. The full set of CXps for the same prediction is* $\mathbb{Y} = \{\{ent\}, \{ndev\}\}$*. Observe how the only AXp minimally hits each CXp from* $\mathbb{Y}$ *and, vice versa, each CXp minimally hits the AXp from* $\mathbb{X}$*.* □

Definition 3: Formal Feature Attribution (FFA). Given the definition of AXp's above, we can now illustrate the *formal feature attribution* (FFA) function by Yu *et al* [62]. Denoted as $\text{ffa}_\tau(i, (\mathbf{v}, c))$, it returns for a classification $\tau(\mathbf{v}) = c$ how important feature $i \in \mathcal{F}$ is in making this classification, defined as the proportion of AXp's for the classification $\mathbb{A}_\tau(\mathbf{v}, c)$, which include feature i, i.e.

$$\text{ffa}_\tau(i, (\mathbf{v}, c)) = \frac{|\{\mathcal{X} \mid \mathcal{X} \in \mathbb{A}_\tau(\mathbf{v}, c), i \in \mathcal{X})|}{|\mathbb{A}_\tau(\mathbf{v}, c)|} \tag{3}$$

Yu *et al* [62] define an anytime algorithm for computing FFA shown in Algorithm 1. The algorithm collects AXp's $\mathbb{A}$ and CXp's $\mathbb{C}$. They are initialized to empty. While we still have resources, we generate a minimal hitting set $\mathcal{Y} \in \text{MHS}(\mathbb{A})$ of the current known AXp's $\mathbb{A}$ and not already in $\mathbb{C}$ with the call MinimalHS$(\mathbb{A}, \mathbb{C})$. If no (new) hitting set exists then we are finished and exit the loop. We then check if (2) holds in which case we add the candidate to the set of CXp's $\mathbb{C}$. Otherwise, we know that $\mathcal{F} \setminus \mathcal{Y}$ is a correct (non-minimal) abductive explanation, i.e. it satisfies (1). We use the call ExtractAXp to minimize the resulting explanation, returning an AXp $\mathcal{X}$ which is added to the collection of AXp's $\mathbb{A}$. ExtractAXp tries to remove features i from $\mathcal{F} \setminus \mathcal{Y}$ one by one while still satisfying (1). When resources are exhausted, the loop exits and we return the set of AXp's and CXp's currently discovered.

Example 6. *Consider our running example in Figure 2. As illustrated in Example 5, the complete set of AXps for the RF prediction of instance* $\mathbf{v}$ *is* $\mathbb{X} = \{\{ent, ndev\}\}$*. Therefore, with Equation* (3) *we can compute that the FFA for* $\mathbf{v}$ *is* {ent: 1, ndev: 1}*, indicating that both the features* ent *and* ndev *make an equal contribution to the RF prediction, while the other two features, i.e.* ld *and* dev_exp*, hold no contribution.* □

3.4 On Propositional Satisfiability

Due to the need to logically reason about the behavior of an ML model of interest, e.g. to be able to check the validity of (1) and (2) in the enumeration process described above, the general setup of FoX makes use of the propositional satisfiability (SAT) reasoning. As a result, the notation and standard definitions widely used in SAT solving are assumed [3]. Namely, we assume formulas to be propositional. A propositional formula φ is said to be in *conjunctive normal form* (CNF) if

it is a conjunction of clauses, where each *clause* is a disjunction of literals. A *literal* is either a Boolean variable taking values 0 or 1, or its negation. Whenever convenient, a clause is considered to be a set of literals. A *truth assignment* μ is a mapping from the variables in φ to $\{0, 1\}$. Namely, $\mu(i) = b$, $b \in \{0, 1\}$ represents the fact the assignment μ sets the i'th variable to b. By slightly abusing the notation, we will use μ_i to represent either positive or negative literal on the i'th variable, depending on whether $\mu(i) = 1$ or $\mu(i) = 0$. A clause is satisfied by a truth assignment μ if at least one of literals in the clause is assigned value 1; otherwise, the clause is falsified. If there exists an assignment μ that satisfies all clauses in a CNF formula φ then formula φ is satisfiable; otherwise, it is unsatisfiable.

4 FOX: FORMALLY EXPLAINING JIT DEFECT PREDICTIONS

In this section, we introduce the method we use for extracting formal explanations for JIT defect predictions made by LR and RF models. Note that it builds on the existing principled approach to computing formal abductive and contrastive explanations studied in the general context of XAI [12, 19, 36, 48] and their enumeration [18]. By building on the above, we describe FoX as follows.

Overview. Figure 3 depicts an overview of FoX, which comprises 3 main steps. First, a user (a software developer) selects an ML model (LR or RF) and a target instance, i.e. a commit, such that an explanation for the model's prediction is sought for the instance. Second, FoX encodes both the model and the instance into a logical formalism (into a SAT formula) such that formal reasoning about satisfiability of the corresponding propositional formula can be applied. Note that, as was shown in [35], LR models admit effective reasoning procedures that operate *directly* on the original linear representation of the classifier, i.e. no logical encoding is necessary for LR models (more on this below), which is still needed for RF models. Given a suitable representation of the classifier, this step then iteratively extracts either a required number of AXps and/or CXps or enumerates them exhaustively. All generated explanations are stored in a solution pool. (We should say here that there can be a multitude of valid explanations for each model prediction, and some of them may be less interesting to the user than the other. As such, explanation enumeration helps the user obtain multiple possible reasons for the prediction and opt for the best ones depending on given criteria.) In the third step, explanations are selected from the pool based on the users' explanation preferences. Alternatively, the tool reports a feature importance explanation in the form of FFA based on all the AXps and CXps enumerated. Afterwards, these explanations are sent back to the user to help them prioritize QA resources on the most risky commits.

A few words should be said regarding the extraction of a *single* formal explanation (i.e. ExtractAXp or ExtractCXp in Algorithms 1 and 2). In general [19], explanation extraction works as follows. Given a target instance $\mathbf{v} \in \mathbb{F}$, the procedure (i) considers a working set of features $\mathcal{S}$ (for instance, $\mathcal{S}$ can be initially set to include all features $\mathcal{F}$) and (ii) traverses each element $i \in \mathcal{S}$ *one-by-one* checking if feature i can be discarded from $\mathcal{S}$. Depending on the type of explanation we are interested in (AXp or CXp), this check is implemented by calling an *oracle* deciding whether or not set $\mathcal{S} \setminus \{i\}$ satisfies the explanation condition (conditions (1) or (2)). If it does, then feature i is discarded; otherwise, it is kept in $\mathcal{S}$. The algorithm proceeds until all features of $\mathcal{S}$ are tested and ends up making $\mathcal{S}$ a subset-minimal AXp or CXp (depending on the condition checked). Note that in general, the checks above involve calling a formal reasoner, e.g. a SAT solver, which solves an NP-hard problem [9] for each of the features tested. However, as was mentioned above, some ML models admit polynomial-time procedures that can replace a potentially expensive NP-oracle call. These include LR models discussed in Section 4.1.

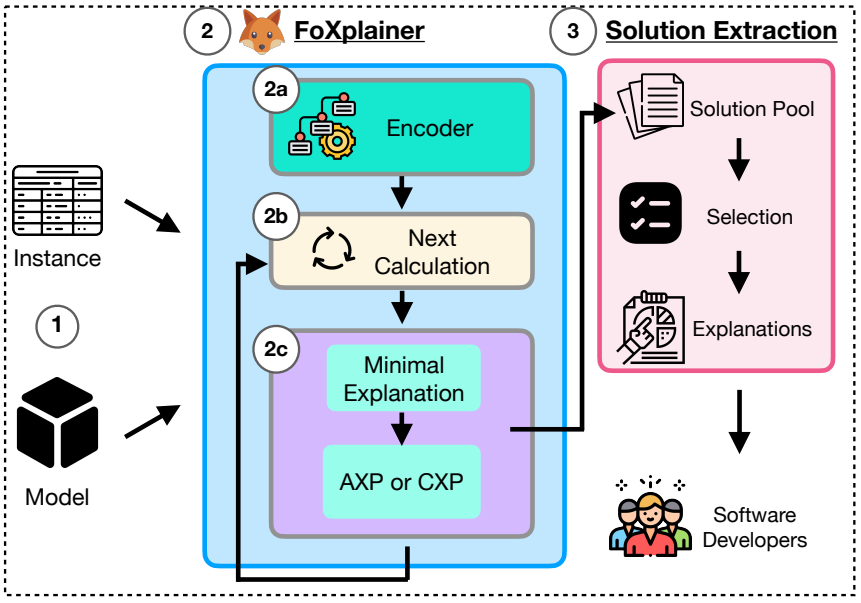

Fig. 3. An overview of FoX to extract explanations. Given an instance to be explained and an ML model, FoX consists of 3 main components: (1) formal encoding, (2) explanation enumeration, and (3) solution pool with the ability to select explanations given user preferences. FoX extracts two types of formal explanations, i.e. AXps and CXps. It can also report FFA.

Algorithm 2 SAT-Based Formal Explanation Enumeration [35]

procedure XpEnum(τ, $\mathbf{v}$, c)
  $(\mathbb{A}, \mathbb{C}) \leftarrow (\emptyset, \emptyset)$ ▷ *Sets of AXp's and CXp's to collect.*
  $\varphi \leftarrow \emptyset$ ▷ *Space to explore.*
  while resources available **do**
    $\mu \leftarrow \text{SAT}(\varphi)$ ▷ *Another unexplored subset?*
    if $\mu = \bot$ **then break** ▷ *Stop if none is computed, i.e. φ is unsatisfiable.*
    $\mathcal{S} \leftarrow \{i \in \mathcal{F} \mid \mu_i = 1\}$ ▷ *Extract the set from model μ.*
    if $\forall(\mathbf{x} \in \mathbb{F}). [\bigwedge_{i \in \mathcal{S}}(x_i = v_i)] \rightarrow (\tau(\mathbf{x}) = c)$ **then** ▷ *Check AXp condition (1) for $\mathcal{S}$.*
      $\mathcal{X} \leftarrow \text{ExtractAXp}(\mathcal{S}, \tau, \mathbf{v}, c)$ ▷ *Get AXp $\mathcal{X}$ by iteratively checking (1) [19].*
      $\mathbb{A} \leftarrow \mathbb{A} \cup \{\mathcal{X}\}$ ▷ *Collect new AXp $\mathcal{X}$.*
      $\varphi \leftarrow \varphi \cup \{(\bigvee_{i \in \mathcal{X}} \neg\mu_i)\}$ ▷ *Block new AXp $\mathcal{X}$.*
    else ▷ *There must be a missing CXp $\mathcal{Y} \subseteq \mathcal{F} \setminus \mathcal{S}$.*
      $\mathcal{Y} \leftarrow \text{ExtractCXp}(\mathcal{F} \setminus \mathcal{S}, \tau, \mathbf{v}, c)$ ▷ *Get CXp $\mathcal{Y}$ by iteratively checking (2) [18].*
      $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathcal{Y}\}$ ▷ *Collect new CXp $\mathcal{Y}$.*
      $\varphi \leftarrow \varphi \cup \{(\bigvee_{i \in \mathcal{Y}} \mu_i)\}$ ▷ *Block new CXp $\mathcal{Y}$.*
  return $\mathbb{A}$, $\mathbb{C}$

4.1 Special Case of Formal LR Explanations

Recent work [35] proposed polynomial-time algorithms to extracting a single AXp or CXp for *monotonic* classifiers. Note that monotonicity of the target classifier (on each feature) is the only requirement on the classifier imposed by [35]. Also observe that LR classifiers, as being a weighted linear sum, are monotonic on all features. As a result, we can apply the approach of [35] to LR classifiers and consider a set of ordered classes $\mathcal{K} = \{c_0 = false, c_1 = true\}$ in JIT defect predictions, where $c_0 < c_1$. Without loss of generality, assume that our LR classifier τ makes use of the first m

features with coefficients $\mathbf{w} = \{w_1, \ldots, w_m\}$, $m \leq |\mathcal{F}|$, s.t $w_i \geq 0\ \forall i \in \{1, \ldots, m\}$. Then, given two concrete instances $\underline{\mathbf{v}} \in \mathbb{F}$ and $\overline{\mathbf{v}} \in \mathbb{F}$ such that $\underline{\mathbf{v}}_i \cdot w_i \leq \overline{\mathbf{v}}_i \cdot w_i$ for any $i \in \{1, \ldots, m\}$, monotonicity of τ ensures that $\tau(\underline{\mathbf{v}}) \leq \tau(\overline{\mathbf{v}})$.

Given the above, also observe that in our setup, each feature $i \in \mathcal{F}$ takes values from a domain $\mathcal{D}_i$ and so has concrete lower bound $\lambda_i = \min \mathcal{D}_i$ and upper bound $\mu_i = \max \mathcal{D}_i$. (Since by assumption, each $w_i \geq 0$, the values of λ_i and μ_i also define the lower and upper bounds on $w_i \cdot \lambda_i$ and $w_i \cdot \mu_i$.) Recall that we seek an explanation for the prediction $c = \tau(\mathbf{v})$ for a particular instance $\mathbf{v} \in \mathbb{F}$ and given feature i let us denote by $\underline{\mathbf{v}}$ and $\overline{\mathbf{v}}$ the instances where the value $v_i \in \mathcal{D}_i$ is replaced by λ_i and μ_i, respectively.

The key observation of [35] is that checking whether or not a feature $i \in \mathcal{S} = \mathcal{F}$ should be included in the explanation can be replaced by a simple test of whether making this feature *free* allows τ to change its value when all the other features from $\mathcal{S} \setminus \{i\}$ are fixed to the values of $\mathbf{v}$. Essentially, this can be done by testing whether $\tau(\underline{\mathbf{v}}) = \tau(\overline{\mathbf{v}})$. If the equality is violated then feature i is important as changing its value also changes the value of τ. In this case, it is put back to set $\mathcal{S}$. Otherwise, feature i is made free, i.e. it is dropped from $\mathcal{S}$. Afterwards, we proceed by considering feature $i + 1$ in a similar fashion. Importantly, if i is made free then the lower and upper bound instances $\underline{\mathbf{v}}$ and $\overline{\mathbf{v}}$ are kept updated as *extreme* instances for our target instance $\mathbf{v}$. This way, during the overall process, the values of the features i in $\underline{\mathbf{v}}$ ($\overline{\mathbf{v}}$, resp.) are either kept equal to v_i or take value λ_i (μ_i, resp.).

Traverse	Working set $\mathcal{S}$	Feature i	$\mathcal{S} \setminus i : \tau(\underline{\mathbf{v}}) = \tau(\overline{\mathbf{v}})$?	Discard?
1	$\{ld, ent, ndev, dev\_exp\}$	ld	true	true
2	$\{ent, ndev, dev\_exp\}$	ent	false	false
3	$\{ent, ndev, dev\_exp\}$	$ndev$	true	true
4	$\{ent, dev\_exp\}$	dev_exp	false	false
—	$\{ent, dev\_exp\}$	—	—	—

Fig. 4. Single AXp Extraction in LR Models.

Example 7. *Consider the commit* $\mathbf{v}$ *and LR model* τ *described in Figure 2. Figure 4 illustrates the process of a single AXp extraction in LR models.*[10] *The procedure initializes the working set* $\mathcal{S} = \{ld, ent, ndev, dev\_exp\}$ *and then traverses each feature* $i \in \mathcal{S}$. *Feature* i *is discarded if* $\tau(\underline{\mathbf{v}}) = \tau(\overline{\mathbf{v}})$ *when fixing* $\mathcal{S} \setminus \{i\}$, *i.e.* $\mathcal{S} \setminus \{i\}$ *satisfies* (1). *As can be observed in the first row in Figure 4 where feature ld is traversed,* $\underline{\mathbf{v}} = \langle \underline{v}_{ld} = \lambda_{ld} = 0, \underline{v}_{ent} = 0.97, \underline{v}_{ndev} = 3, \underline{v}_{dev\_exp} = 320 \rangle$ *and* $\overline{\mathbf{v}} = \langle \underline{v}_{ld} = \mu_{ld} = 309,728, \underline{v}_{ent} = 0.97, \underline{v}_{ndev} = 3, \underline{v}_{dev\_exp} = 320 \rangle$ *for* $\mathcal{S} \setminus \{i\} = \{ent, ndev, dev\_exp\}$ *such that* $\tau(\underline{\mathbf{v}}) = \tau(\overline{\mathbf{v}})$, *and thus feature* ld *is discarded. Similarly, the third row indicates that feature* ndev *is discarded since* $\tau(\underline{\mathbf{v}}) \neq \tau(\overline{\mathbf{v}})$ *for* $\mathcal{S} \setminus \{i\} = \{ent, dev\_exp\}$. □

4.2 Formal Explanation Enumeration

Note that in the explanation extraction procedure outlined above (both the general SAT-based case and the simplified case of LR models), we start from the complete set of features $\mathcal{F}$. In practice, however, one can bootstrap the procedure with a subset of features $\mathcal{S} \subseteq \mathcal{F}$ that is guaranteed to satisfy the corresponding explanation condition (either (1) or (2)). This fact is exploited by the formal explanation enumeration [18, 35]. A high-level view on formal explanation enumeration is outlined in Algorithm 2 (readers are referred to [18, 35] for more details). Note that it differs from the procedure of Algorithm 1 as it does not rely on minimal hitting set enumeration when

[10] In a similar vein, when extracting a CXp, the procedure replaces (2) checks for subsets $\mathcal{S} \setminus \{i\}$ with checks testing whether or not $\tau(\underline{\mathbf{v}}) \neq \tau(\overline{\mathbf{v}})$ for $\mathcal{S} \setminus \{i\}$.

Table 1. An overview of the studied JIT defect datasets provided by McIntosh and Kamei [37].

Project	Training Data				Testing Data			
	Start Date	End Date	# Commits	# Defective Commits	Start Date	End Date	# Commits	# Defective Commits
Openstack	11/30/2011	08/13/2013	9,246	980 (11%)	08/13/2013	02/28/2014	3,963	646 (16%)
Qt	06/18/2011	05/08/2013	19,312	1,577 (8%)	05/08/2013	03/18/2014	8,277	476 (6%)

computing candidate features subsets to test. This is because in some practical scenarios, e.g. for LR models, it is cheaper to extract a subset-minimal explanation by iteratively checking either condition (1) or (2) than to delegate this task to the minimal hitting set enumerator in use.

To make the enumeration work, one needs to keep track of all subsets of $\mathcal{F}$ that are already seen and iteratively identify yet unseen subset $\mathcal{S} \subseteq \mathcal{F}$ (see lines 5–7) to check whether or not this new subset $\mathcal{S}$ satisfies (1) (cf. line 8). If it does then AXp extraction can be performed starting from set $\mathcal{S}$ and applying the linear search feature traversal augmented with oracle calls checking (1). Otherwise, $\mathcal{S}$ does not satisfy (1). Observe that in this case subset $\mathcal{F} \setminus \mathcal{S}$ satisfies (2) (cf. line 12), i.e. CXp extraction can be performed starting from subset $\mathcal{F} \setminus S$. In either case, a subset-minimal explanation is extracted and collected in each iteration of the enumeration algorithm.

Note that the space of all possible subsets to explore is modeled in Algorithm 2 with the use a CNF formula φ on Boolean variables μ_i, $i \in \mathcal{F}$ (see line 3). (Recall from Section 3.4 that we use μ to represent a truth assignment, and μ_i are meant to represent individual Boolean literals as defined by this assignment.) The meaning of each variable μ_i is that the corresponding feature i is *selected* for inclusion in the target subset $\mathcal{S}$ if and only if $\mu_i = 1$. At the beginning, formula φ has no clauses, which makes *any* subset of features initially possible. As soon as an AXp $\mathcal{X}$ is computed, the variables μ_i, $i \in \mathcal{X}$, are used to construct a clause $(\bigvee_{i \in \mathcal{X}} \neg \mu_i)$ and add it to formula φ, thus, forbidding the same explanation to be computed in the next iterations of the algorithm (line 11). On the other hand, if a CXp $\mathcal{Y}$ is computed, the corresponding variables u_i are used to add to formula φ another clause $(\bigvee_{i \in \mathcal{X}} \mu_i)$, which forces the next iterations to include some of the features of CXp $\mathcal{Y}$ (line 15). Overall, this algorithmic setup ensures that the enumeration process is guided by the already computed CXps and that no previously computed AXp is repeated. Observe that this is correct thanks to the known minimal hitting set duality between AXps and CXps [18] illustrated in Example 5 and also exploited in Algorithm 1. Finally, depending on our setup, the process can proceed until we complete explanation enumeration or until available computational resources, e.g. time, are exhausted. As a result, we can provide a user with a computed set of AXps or CXps we well as compute and report FFA values.

Iteration	$\mathcal{S}$	$\mathcal{S}$ satisfies (1)?	$\mathcal{X}$	$\mathcal{Y}$	Clause added to φ
1	$\{\}$	false	—	$\{ndev\}$	(u_{ndev})
2	$\{ndev\}$	false	—	$\{ent\}$	(u_{ent})
3	$\{ent, ndev\}$	true	$\{ent, ndev\}$	—	$(\neg u_{ent} \vee \neg u_{ndev})$

Fig. 5. Explanation Enumeration.

Example 8. *Consider our running commit* **v** *and RF model* τ *depicted in Figure 2. Figure 5 illustrates the AXp and CXp enumeration process of Algorithm 2. In the first iteration, an empty target set* $\mathcal{S}$ *is identified. Since* $\mathcal{S}$ *is not able to satisfy* (1)*,* $\mathcal{F} \setminus \mathcal{S} = \{ld, ent, ndev, dev\_exp\}$ *is the initial working set for the single CXp extraction, and thus a CXp* $\mathcal{Y} = \{ndev\}$ *is extracted and feature* ndev *must be included for the target sets in the next iterations. Similarly, the next candidate* $\mathcal{S}' = \{ndev\}$ *fails to satisfy* (1) *and so its complement* $\mathcal{F} \setminus \mathcal{S}'$ *is used to extract a CXp* $\mathcal{Y}' = \{ent\}$*; observe that feature* ent *has to be selected for the later candidate sets. Afterwards,* $\mathcal{S}'' = \{ent, ndev\}$ *is selected as the target set*

Table 2. The commit-level software features for Just-In-Time Defect Prediction provided by McIntosh and Kamei [37]. The asterisk symbol indicates the features selected by AutoSpearman that ensure non-collinearity among features.

Category	Name	Description
Size	LA*	The number of lines added by a commit.
	LD*	The number of lines deleted by a commit.
Diffusion	NS*	The number of modified subsystems.
	ND*	The number of modified directories.
	NF	The number of modified files.
	Complexity	The entropy of modified lines that spread across changed files.
History	Unique changes	The number of prior changes to the modified files.
	Developers	The number of developers who have changed the modified files in the past.
	Age*	The time interval between the last and current changes.
Experience	Prior changes	The number of prior changes that an author has participated in.
	Recent changes	The number of prior changes that an author has participated in weighted by the age of the changes.
	Subsystem changes	The number of prior changes to the modified subsystem(s) that an author has participated in.
	Awareness	The proportion of the prior changes to the modified subsystem(s) that an author has participated in.
Review	Iterations	The number of times that a change was revised prior to integration.
	Reviewers*	The number of reviewers who have voted on whether a change should be integrated or abandonded.
	Comments	The number of non-automated, non-owner comments posted during the review of a change.
	Review window*	The length of time between the creation of a review request and its final approval for integration.

in the third iteration. As $\mathcal{S}''$ satisfies (1)*, it is then used for extracting a single AXp $\mathcal{X} = \{ent, ndev\}$, and either* ent *or* ndev *is forbidden in next iterations. Finally, formula φ becomes unsatisfiable indicating that there exists no unseen set to explore, and the enumeration process terminates. At this point, we can report either the only AXp identified or two CXps enumerated. Finally, we can use the AXp to figure out that $\mathit{ffa}_\tau(1, (\mathbf{v}, c)) = 0$, $\mathit{ffa}_\tau(2, (\mathbf{v}, c)) = 1$, $\mathit{ffa}_\tau(3, (\mathbf{v}, c)) = 1$, and $\mathit{ffa}_\tau(4, (\mathbf{v}, c)) = 0$ as features 2 and 3 (*ent *and* ndev*) participate in the only AXp available for instance* $\mathbf{v}$ *while features 1 and 4 (*ld *and* dev_exp*) do not.* □

5 EXPERIMENTAL DESIGN

In this section, we present the studied datasets and explain the details of our experimental design.

Studied JIT Datasets. In our experiment, we build JIT defect models based on the datasets provided by McIntosh and Kamei [37], which were previously used in the PyExplainer paper [42]. These JIT defect datasets represent two large-scale open-source software projects, i.e. Openstack and Qt. They are chosen in order to ensure a fair comparison with the PyExplainer paper [42]. In addition, these datasets are widely-used as a benchmark JIT defect suite [14, 15, 37, 41, 42] and have been manually verified to ensure the validity of the SZZ algorithm [52] to reduce the number of false positives and false negatives [10, 61]. Openstack is an open-source software for cloud infrastructure service. Qt is a cross-platform application development framework.

Table 1 provides an overview of the studied datasets. For each dataset, there are 17 commit-level features that span across 5 dimensions, i.e., Size (e.g., lines added, lines deleted), Diffusion (e.g., #modified files), History (#developers), Experience, and Code Review Activities. The list of the studied features is provided in Table 2.

Experiment Design. To ensure a fair comparison, we use the same experimental setup as the PyExplainer paper [42] as follows:

(Step 1) Data Splitting. To avoid any temporal bias in our experiment, we first preserve the order of commits by sorting based on their commit date [41, 57]. Then, we use a time-wise hold-out validation technique (as used by McIntosh and Kamei [37]) to split the dataset into training (70%)

and testing (30%) datasets. The use of the time-wise hold-out validation technique ensures that the commits that appear later will not be used in the model training. Similarly, the commits that appear earlier will not be used in the model evaluation.

(Step 2) Build Global JIT Defect Models. For each training dataset, we first mitigate collinearity using AutoSpearman and handle class imbalance using SMOTE prior to building JIT defect models. After using AutoSpearman, we finally select 7 features that are not highly-correlated with each other. To ensure that the predictions of our JIT defect models are highly accurate, we apply a class rebalancing technique, as suggested by prior work [1, 53]. Since the defective ratio of our studied JIT defect datasets are highly imbalanced (i.e., 8%-16%), we apply SMOTE [6] to handle class imbalance *only on the training dataset*. Then, we build global JIT defect models using the training data of each project. Same as the PyExplainer paper [42], we use two classification techniques, i.e., Logistic Regression (LR) and Random Forest (RF). We find that our global JIT defect models achieve an AUC of 0.66 (LR) and 0.75 (RF) for OpenStack and an AUC of 0.64 (LR) and 0.74 (RF) for Qt, which are the same as the PyExplainer paper [42].

(Step 3) Explanation Generation. Then, we apply FoX to generate all (abductive and contrastive) explanations for each prediction of JIT defect models. Similarly, given a prediction, we also generate an explanation using the four baseline model-agnostic techniques, namely, LIME [45], SHAP [34], Anchor [46], PyExplainer [42]. These four approaches have been previously used in many prior studies of explainable defect prediction models [7, 22, 23, 30, 40, 42, 43].

6 EXPERIMENTAL RESULTS

In this section, we compare FoX with four model-agnostic techniques, i.e., LIME [45], SHAP [34], Anchor [46], PyExplainer [42] with respect to correctness, robustness and efficiency. Anchor and PyExplainer are feature selection approaches, generating features that are deemed sufficient for a given prediction , while LIME and SHAP produce feature attributions, revealing the contributions of individual features to the prediction. Note that FoX can compute features adequate for a given prediction and also compute FFA indicating the contributions of features.

(RQ1) Does FoX generate more correct explanations than existing approaches for Just-In-Time defect predictions?

Approach. To answer RQ1, we analyze the percentage of provably correct explanations computed by the four model-agnostic techniques. For feature selection approaches, e.g. PyExplainer and Anchor, the provably correct explanations mean *why?* explanations that satisfy (1) or *how?* explanations that satisfy (2), while for feature attribution approaches, e.g. LIME and SHAP, FFA serves as provably correct feature attributions.

Therefore, feature attributions generated by LIME and SHAP are compared with FFA using three metrics, namely errors, Kendall's Tau [25] and rank-based overlap (RBO) [59]. The *error* is quantified by the sum of absolute differences across all features, i.e. Manhattan distance, while Kendall's Tau and RBO are used to compare rankings of feature attributions.[11] Kendall's Tau and RBO are assessed within the ranges of [-1, 1] and [0, 1], respectively. A higher value for both metrics signifies stronger alignment or closeness between a ranking and FFA.

Since existing model-agnostic techniques are computationally expensive, we only randomly select 500 different instances in each testing dataset for evaluation. As PyExplainer generates at most 2,000 explanations answering *why* questions in each instance to be explained, we only analyze the correctness of the explanation with the highest importance score. For Anchor, we measure the

[11] Kendall's Tau is a correlation coefficient that evaluates the ordinal relationship between two ranked lists, providing a way to measure the similarity in the order of values. RBO is a metric that measures the closeness between two ranked lists, considering both the order and the depth of the overlap.

Table 3. (RQ1) Comparison between FoX's FFA against LIME and SHAP. (↘) Error = Better. (↗) Higher Kendall's Tau and RBO = Better.

Approach	Model	Openstack			Qt		
		Error	Tau	RBO	Error	Tau	RBO
LIME	LR	4.818	0.047	0.551	5.630	-0.086	0.447
	RF	6.089	0.064	0.516	7.847	0.063	0.358
SHAP	LR	5.098	0.003	0.531	5.215	-0.129	0.437
	RF	5.645	0.118	0.596	6.867	0.134	0.306

correctness of the only explanation computed in each instance. FoX enumerates all explanations for an instance, including AXps and CXps. Thus, we analyze the correctness of all the AXps and CXps computed. To compare LIME and SHAP with FFA, we take their outcomes substituting negative attributions by their positive counterpart (i.e. taking the absolute value) and then normalize the values into the range of [0, 1], which corresponds to the same scale as FFA.

Result. **In contrast to the explanations provided by FoX, heuristic explanations for Just-In-Time defect predictions are susceptible to the lack of correctness.** First, let us consider feature selection explanations. By definition, explanations computed by FoX are guaranteed to 100% correct for all instance predictions made by LRs and RFs for both studied datasets. On the other hand, according to our results, none of the explanations, i.e. 0%, computed by Anchor and PyExplainer are correct for either of the models in the two selected datasets. This stark difference clearly demonstrates that FoX is advantageous over the model-agnostic feature selection competition with respect to explanation correctness.

As for feature attribution approaches, the importance of features as reported by model-agnostic LIME and SHAP is compared against FFA generated by FoX and detailed in Table 3. For each dataset, we compute the metric for each individual instance and afterwards average the outcomes to acquire the final result of the dataset. Observe that the errors of LIME's feature attributions in LRs are 4.818 and 5.630 for Openstack and Qt, respectively, while the corresponding values in SHAP are 5.098 and 5.215. On the other hand, the errors in RFs are higher compared to LRs. LIME's errors are 6.089 and 7.847 in RFs in these two datasets, whereas SHAP's errors are 5.645 and 6.867. LIME and SHAP also demonstrate similar performance with respect to the two ranking comparison metrics. The values of Kendall's Tau for LIME (resp. SHAP) range from -0.086 to 0.064 (resp. from -0.129 to 0.134). In terms of RBO values, LIME shows results between 0.358 and 0.551, whereas the values in SHAP span from 0.306 to 0.596. In summary, as indicated by Table 3, both LIME and SHAP fail to achieve strong closeness to FFA.

(RQ2) Does FoX generate more robust explanations than existing approaches for Just-In-Time defect predictions?

Approach. To answer RQ2, we evaluate the percentage of robust explanations generated by all the competitors. A higher percentage of robust explanations in an approach indicates that the approach is more robust. We run each experiment twice and compare the explanations computed for the same instance afterwards. For feature selection approaches, if a method can compute *identical explanations* in two separate experiments then we conclude that the approach computes a robust explanation for the instance. On the other hand, an explanation in feature attribution approaches is deemed robust if the approach consistently produces *the same ranking* of feature attributions for a given prediction in the two experiments. Similar to RQ1, we consider the most important

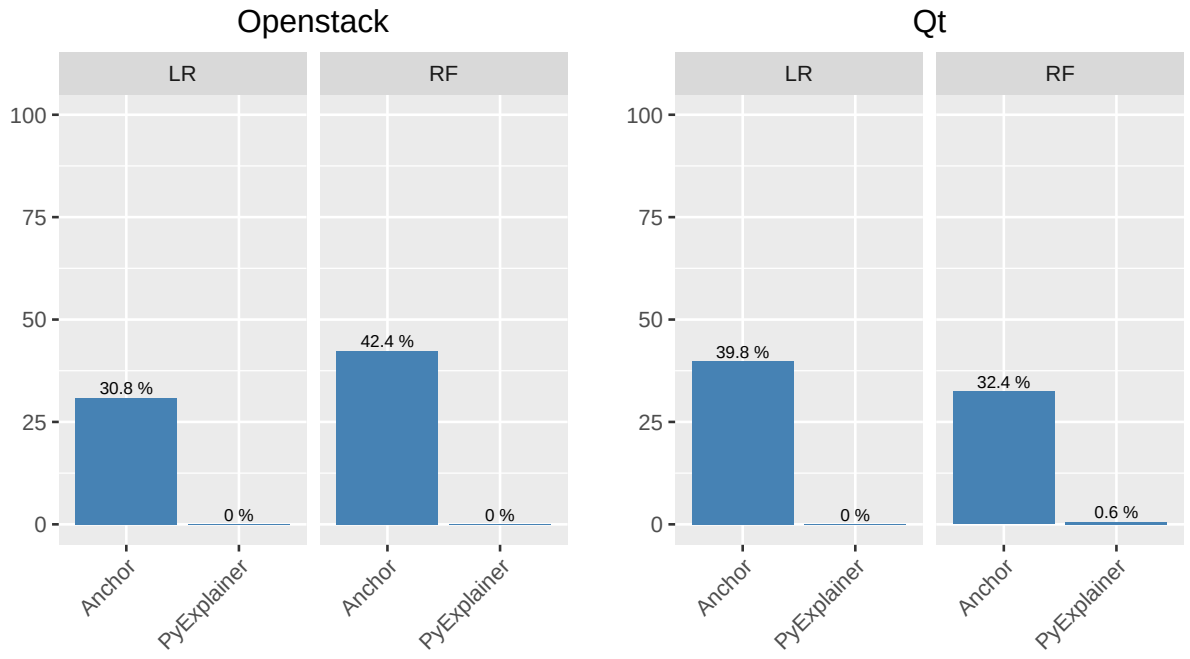

Fig. 6. (RQ2) Robustness of feature selection methods. FoX achieves perfect robustness (i.e., 100%) on both datasets and thus is not shown in the bar chart. (↗) Higher Robustness = Better.

explanation for an instance in PyExplainer, while we replace the solutions of LIME and SHAP by their absolute values.

Result. **FoX is guaranteed to compute 100% robust explanations.** The results of feature selection approaches regarding the percentage of robust explanations for instances to be explained are shown in Figure 6. We should emphasize that FoX always generates robust explanations for all LR and RF predictions for two studied datasets. Due to the deterministic nature of the modern SAT solvers, FoX computes not just a single robust explanation but instead it enumerates all explanations in the same order, and the order can be dictated by user-defined preferences.

In contrast, PyExplainer is not able to extract robust explanations for the studied ML models and datasets. For Openstack dataset Anchor computes 30.8% and 42.4% robust explanations for the LR and RF models respectively, while this approach generates 39.8% and 32.4% robust explanations for the predictions of instances in Qt dataset for the two models. Figure 7 illustrates the percentage of robust explanations for instances in feature attribution approaches. As FoX always computes robust explanations in the same order, FoX always generates robust FFA. 100% robust explanations can also be generated by SHAP which is deterministic. On the contrary, LIME fails to generate any robust explanation. In summary, due to the deterministic nature of the modern SAT solvers, the explanations generated by FoX are guaranteed to be robust and this achievement is only demonstrated in one of the four other model-agnostic techniques, i.e. SHAP.

(RQ3) Does FoX generate explanations faster than existing approaches for Just-In-Time defect predictions?

Approach. To answer RQ3, we assess the average runtime of generating one explanation per instance measured for FoX and the other four techniques. Lower runtime for computing an explanation denotes that the approach can more efficiently explain a JIT defect prediction. In contrast to

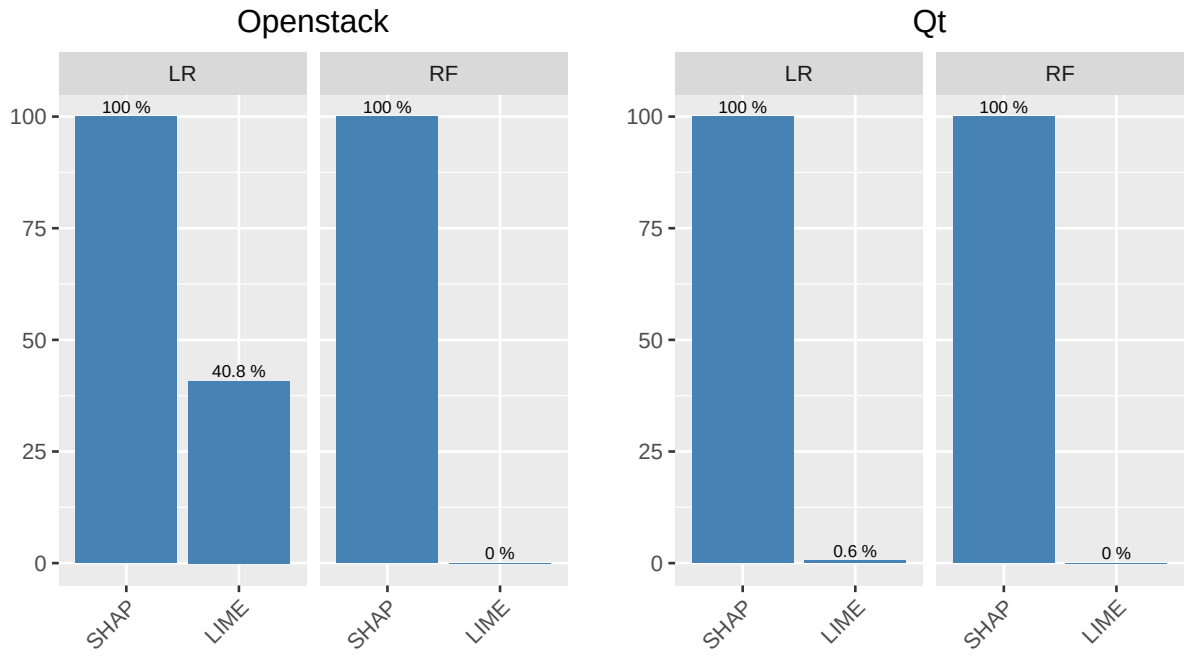

Fig. 7. (RQ2) Robustness of feature attribution methods. FoX achieves perfect robustness (i.e., 100%) on both datasets and thus is not shown in the bar chart. (↗) Higher Robustness = Better.

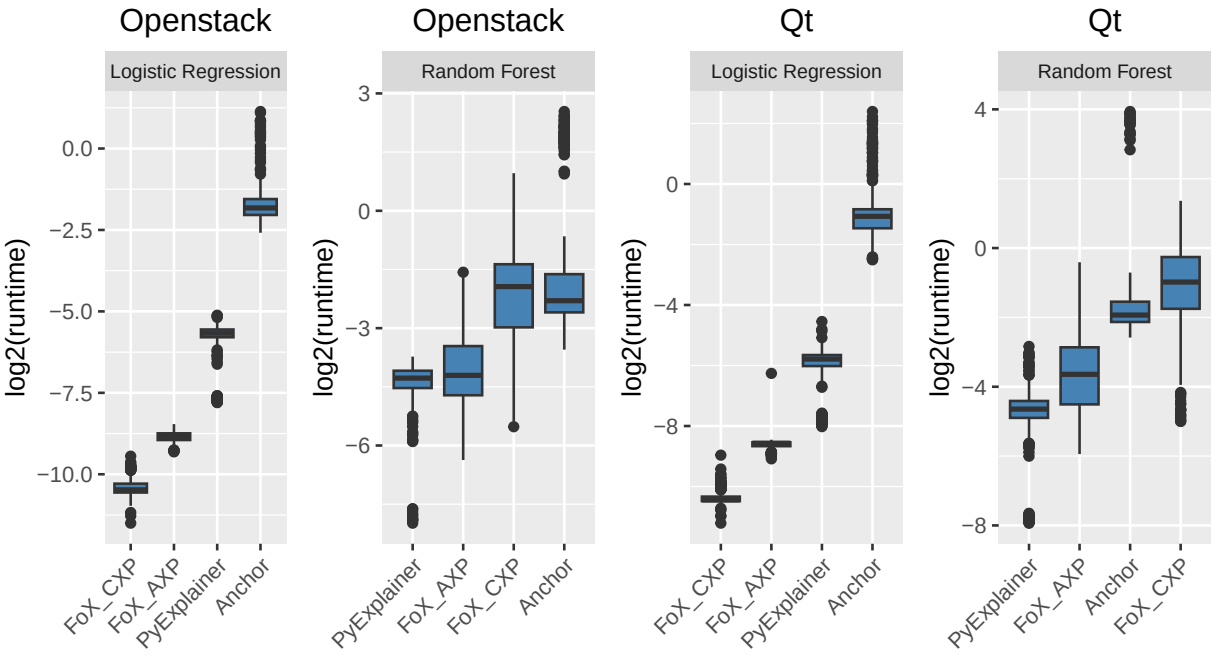

Fig. 8. (RQ3) Runtime of feature selection methods. (↘) Lower Runtime = Better.

other feature selection approaches that are only able to compute explanations answering *why* questions, FoX enumerates both AXps and CXps for a prediction. As a result, we analyze the time

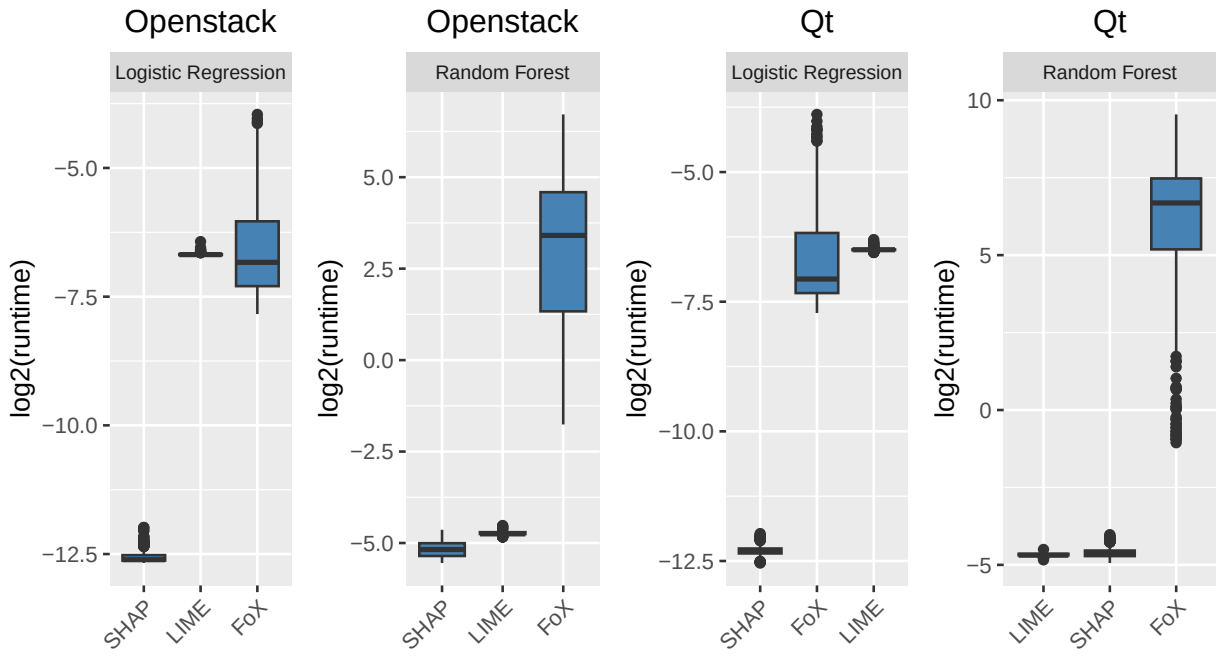

Fig. 9. (RQ3) Runtime of feature feature attribution methods. (↘) Lower Runtime = Better.

spent on generating an AXp and CXp separately represented by FoX_{AXp} and FoX_{CXp}, respectively. Moreover, we also evaluate the time taken by FoX to generate FFA.

Result. **FoX takes less than 0.003 seconds and less than 0.835 seconds to generate a feature selection based explanation for LRs and RFs on average, respectively, while 0.013s and 30.347s are required to generate an FFA for these two models.** Figure 8 depicts the runtime of computing an explanation for two models and both studied datasets in feature selection approaches. For LR models, FoX demonstrates high efficiency with the median time for generating a single AXp being 0.0022 seconds for Openstack and 0.0026 seconds for Qt. Also, FoX takes 0.0008 seconds to produce a CXp for most instances in both datasets. The median runtime of producing an explanation for Openstack by PyExplainer and Anchor is 0.0197 and 0.2825 seconds, whereas the runtime is 0.0181 and 0.4764 seconds for Qt. For RF models, PyExplainer spends 0.0516 seconds to generate an explanation for most instances to be explained in Openstack, while the runtime drops to 0.0400 seconds for Qt. The median runtime of computing an AXp by FoX is 0.0542 seconds for Openstack and 0.0801 seconds for Qt, while CXp extraction takes 0.2609 and 0.5066 seconds for these datasets, respectively. Observe that the runtime increase exhibited by FoX for RF models is caused by the need to represent the models logically and make a large series of SAT oracle calls, as described in Section 4.2. Anchor takes 0.2033 and 0.2620 seconds to generate an explanation for most instances in Openstack and Qt.

The runtime of generating a feature attribution based explanation for selected models and datasets is illustrated in Figure 9. For LR models, FoX remains high efficiency, with the median time of 0.0088 seconds to generate a FFA for an instance in Openstack and 0.0075 seconds for Qt. The median runtime required to generate a feature attribution by LIME and SHAP is 0.0104 and 0.0002 seconds, respectively, for Openstack, and 0.0113 and 0.0002 seconds, respectively, for Qt. For RF models, FoX takes a median time of 10.6197 seconds to generate FFA for an instance in the Openstack dataset and 102.8650 seconds in the Qt dataset. In contrast, the median runtime

for generating a feature attribution in the Openstack dataset using LIME and SHAP is 0.0362 and 0.0276 seconds, respectively. In the Qt dataset, the corresponding runtime is 0.0374 seconds for LIME and 0.0403 seconds for SHAP.

7 A USER STUDY OF EXPLAINABLE JUST-IN-TIME DEFECT PREDICTION

The quantitative evaluation of FoX is demonstrated in Section 6. However, it is essential to delve into the practical utility of explainable Just-In-Time (JIT) defect prediction for software practitioners, a vital aspect yet to be explored. To bridge this gap in knowledge, we conducted a user study to assess software developers' perceptions regarding the usefulness and trustworthiness of JIT defect prediction with and without explanations. This investigation seeks to unveil the practicality and real-world implications of our approach.

Following the guidelines provided by Kitchenham and Pfleeger [28], we conducted our study according to the following steps: (1) design and develop a survey, (2) recruit and select participants, and (3) verify data and analyze data. We explain the details of each step below.

7.1 Survey Design

Step 1 – Design and development of the survey: We designed our survey as a cross-sectional study where participants provided their responses at one fixed point in time. The survey consists of 7 closed-ended questions and 5 open-ended questions. We use multiple-choice questions and a Likert scale from 1 to 5 for closed-ended questions. Our survey consists of two parts: preliminary questions and participants' perceptions of AI-generated software vulnerability repairs.

Part I: Demographics. The survey commences with a query, "(D1) What is your role in your organization?", to ensure that our survey captures responses from the intended target participants. Subsequently, a demographics question, "(D2) What is the level of your professional experience?", is presented to ensure a diverse distribution of responses across software practitioners with varying degrees of professional experience.

Part II-A: JIT Defect Prediction Without Explanations. To simulate a realistic code review scenario, we asked participants to imagine themselves as software developers immersed in the continuous task of reviewing numerous commits and pull requests. In the dynamic world of software development, introducing new code can inadvertently lead to software bugs and errors. This is where AI-based defect prediction comes into play. The AI-based defect prediction is designed to predict potentially defective commits, providing a proactive approach to prioritize code reviews efficiently. Specifically, we told the participants that the defect prediction model will provide a warning for a defective commit as presented in the upper part of Figure 10.

We then asked the participants to assess the usefulness and trustworthiness of the defect prediction, along with providing reasons for their assessments. In particular, four inquiries were presented to the participants, commencing with "(Q1.1) How do you perceive the usefulness of the AI-based defect prediction?"; followed by "(Q1.2) Please justify your answer to Q1.1."; then "(Q1.3) Do you trust the AI-based defect prediction?"; and concluded with "(Q1.4) Please justify your answer to Q1.3.".

Part II-B: JIT Defect Prediction With Explanations. In this part of the survey, we asked participants to imagine that they were in the same situation as in the previous Part II-A. However, this time, the AI-based defect prediction provides both predictions and explanations of why the model generated such a prediction along with actionable guidance to help you mitigate the defective commit as presented in the lower part of Figure 10.

We then asked the participants to assess the usefulness and trustworthiness of the defect prediction with explanations and actionable guidance, while also providing reasons for their assessments. In particular, four inquiries were presented to the participants. These began with "(Q2.1) How

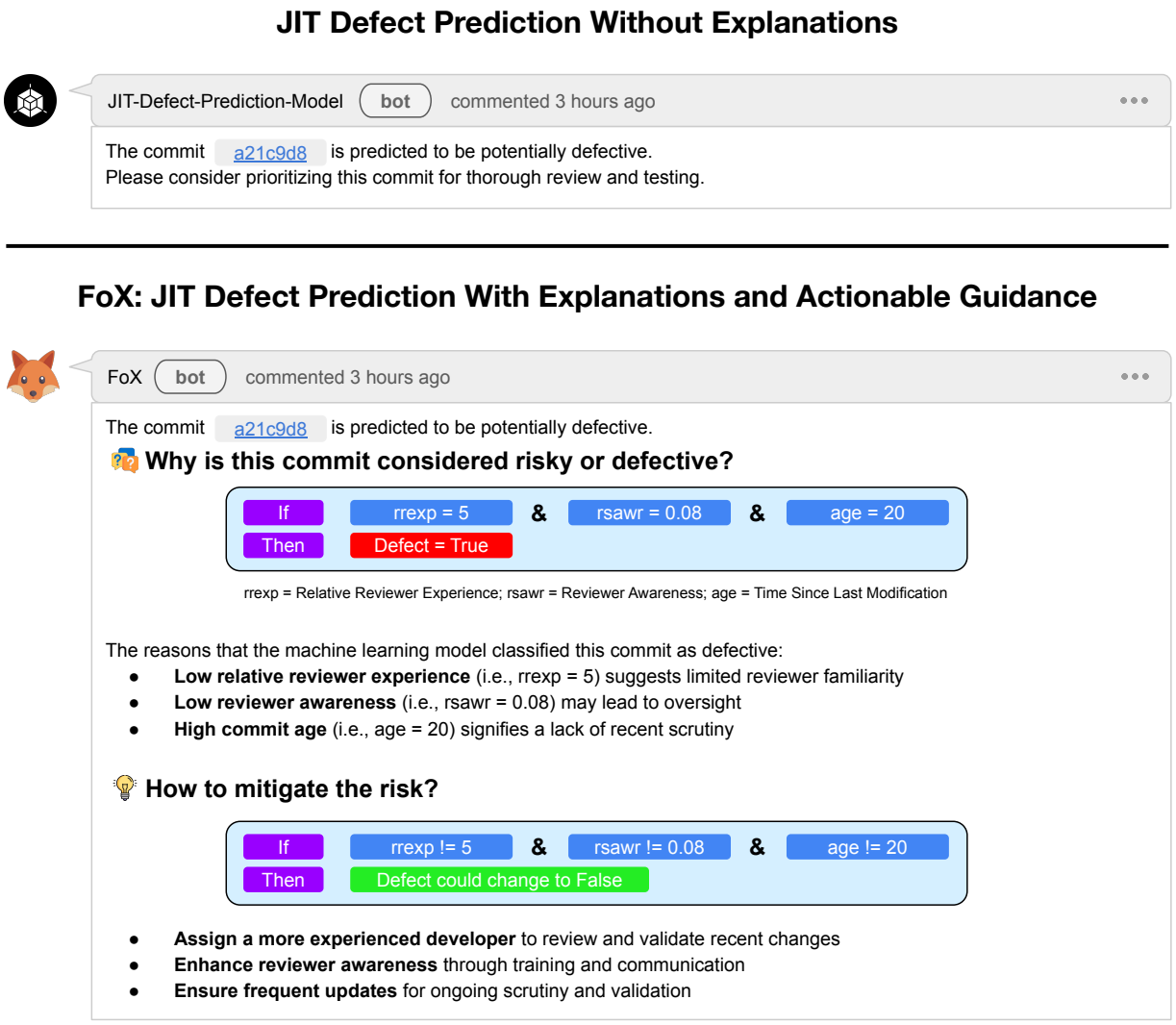

Fig. 10. The upper part shows the Just-In-Time (JIT) defect prediction without any explanations, as presented in Part II-A of our user study. The lower part shows FoX, which presents the JIT defect prediction with explanations and actionable guidance, as presented in Part II-B of our user study.

do you perceive the usefulness of the AI-based defect prediction with "Explanations (WHY)" and "Actionable Guidance (HOW)"?"; followed by "(Q2.2) Please justify your answer to Q2.1."; then "(Q2.3) Do you trust the AI-based defect prediction with "Explanations (WHY)" and "Actionable Guidance (HOW)"?"; then "(Q2.4) Please justify your answer to Q2.3."; then "(Q2.5) Would you consider adopting AI-based defect prediction with explanations if they are integrated into your CI/CD pipeline (e.g., a GitHub action) for free with no conditions?"; and concluded with "(Q2.6) What is your expectation of explainable defect prediction and how can we improve them?"

We employed Google Forms as the platform for our online survey. Upon accessing the landing page, each participant was welcomed with a detailed introductory statement. This statement clarified the study's objectives, the reasoning behind participant selection, potential benefits and risks, and our dedication to maintaining confidentiality. The survey was intentionally brief, taking approximately 15 minutes to complete, and guaranteed complete anonymity for all participants. It is worth noting that our survey underwent a thorough evaluation process and obtained ethical approval from the Monash University Human Research Ethics Committee (MUHREC ID: 41735).

Step 2: Recruit and select participants: We reached out to practitioners with software engineering-related experience through LinkedIn and Facebook platforms. Specifically, we advertised our survey by posting on LinkedIn and Facebook, providing accessible links. Additionally, we reached out directly to target groups via direct messages. Ultimately, we received a total of 54 responses over a two-week recruitment period.
Step 3: Verify data and analyze data: To ensure the integrity of our survey responses, especially concerning open-ended questions, we carried out a comprehensive manual review. We identified and excluded 0 invalid responses, such as those with unanswered open-ended questions or incomprehensible responses, from the total of 54 received. Consequently, we included all 54 responses for analysis. Closed-ended responses underwent quantitative analysis and were depicted using Likert scales in stacked bar plots. Furthermore, we conducted a detailed manual examination of responses to open-ended questions to gain deeper insights into participants' perspectives.

7.2 Survey Results

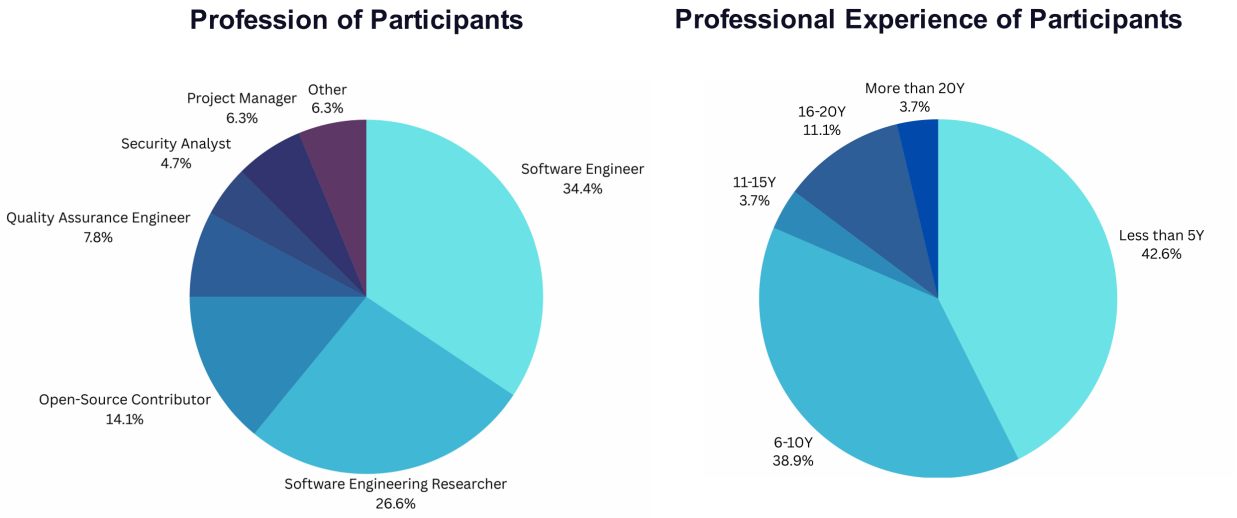

Fig. 11. The demographics of our survey participants in terms of their profession and professional experience.

Part I: Demographics. Figure 11 presents the overall respondent demographic. In terms of the profession of the participants, 34% ($\frac{19}{54}$) of them are software engineers, 27% ($\frac{15}{54}$) of them are software engineering researchers, 14% ($\frac{7}{54}$) of them are open-source contributors, while the other 25% ($\frac{13}{54}$) are software quality assurance engineers, software project managers, and software security analysts. In terms of the level of their professional experience, 42% ($\frac{23}{54}$) of them have less than 5 years of experience, 39% ($\frac{21}{54}$) have 6-10 years of experience, 15% ($\frac{8}{54}$) have 11-20 years of experience, while the other 4% ($\frac{2}{54}$) has more than 20 years of experience.
Part II-A: JIT Defect Prediction Without Explanations. Figure 12 summarizes the answers to (Q1.1)-(Q1.4) regarding participants' perception of the JIT defect prediction without explanations. **As presented in (Q1.1) and (Q1.3) results, 72% ($\frac{39}{54}$) of participants perceived the defect prediction presented in the upper part of Figure 10 as useful, while only 53% ($\frac{29}{54}$) of participants trusted the prediction.**

Some participants expressed strong support for the JIT defect prediction. For example, a software engineering (SE) researcher with 6-10 years of experience stated, *"A good code defect detection tool can help developers or users avoid risks. For instance, unauthorized individuals may exploit the loopholes created by code defects to engage in illicit activities. Particularly in commercial applications,*

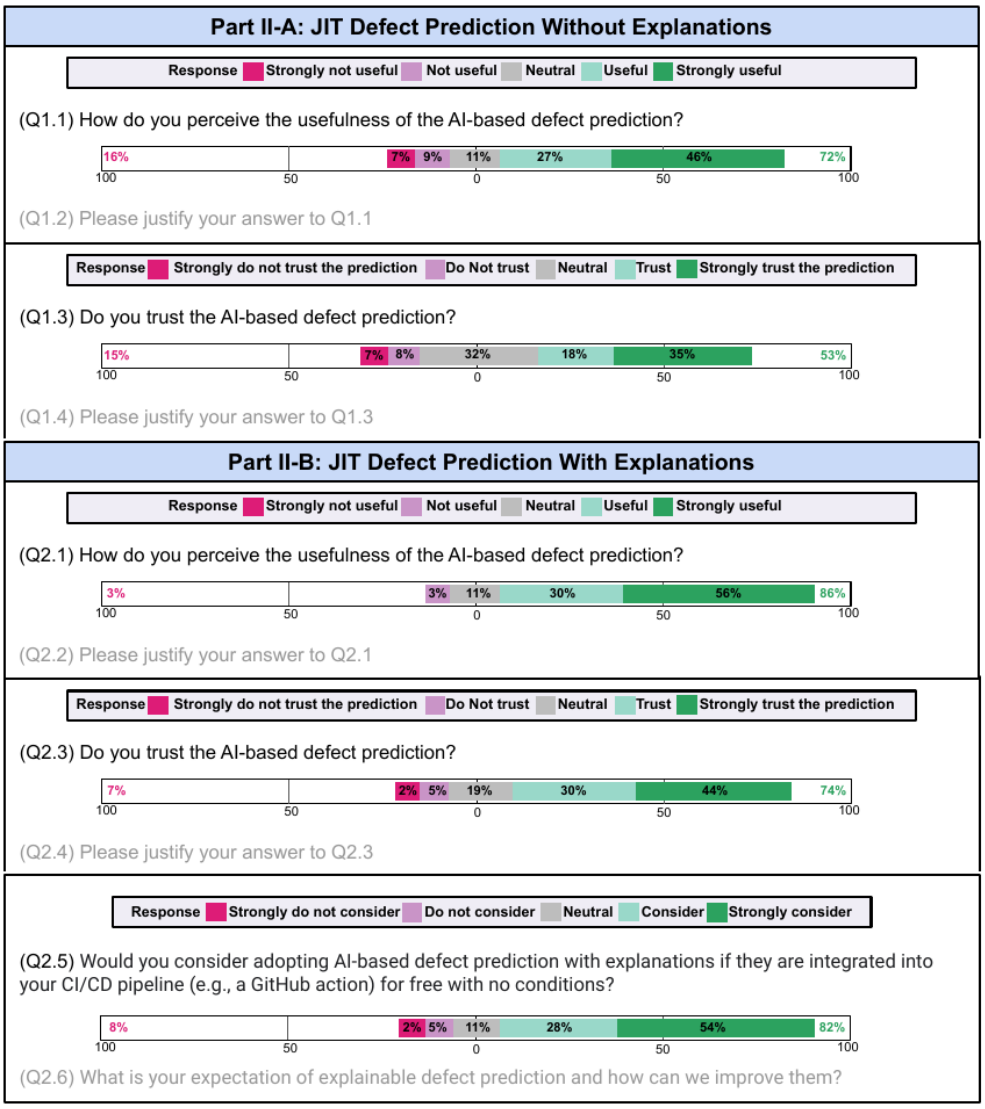

Fig. 12. (Survey Results) A summary of the survey questions (i.e., Part II-A: Q1.1-Q1.4 and Part II-B: Q2.1-2.6) and the results obtained from 54 participants.

a reliable code defect detection model can not only reduce a significant amount of ineffective human effort but also ensure the security of the application." Another SE researcher with less than 5 years of experience mentioned, *"With the development of AI technology, the accuracy of AI-based code review is getting higher and higher. Although it is not 100% reliable, it can still detect some potential bugs."*

However, some participants raised concerns about the explainability and transparency of the defect prediction model, which are important factors in building trust between developers and end users. A SE researcher with less than 5 years of experience noted, *"Without an explanation of the prediction, it cannot reduce the time to review the commit."* Another software testing engineer with 6-10 years of experience expressed, *"It's a black box. We can't know the reason behind the prediction. Hard to trust the prediction."* Additionally, an experienced software security analyst with 16-20 years of experience commented, *"Only prediction seems suspicious, I don't think this would help automate my workflow".*

Overall, most participants found the JIT defect prediction to be valuable as it can streamline human effort and identify potential bugs. However, only half expressed trust in the prediction, citing

concerns about the opaque nature of machine learning models and the absence of explanations. This underscores the practical necessity of our proposed explainable JIT defect prediction approach. **Part II-B: JIT Defect Prediction With Explanations**. Figure 12 provides an overview of participants' perceptions of our JIT defect prediction with explanations, covering questions (Q2.1)-(Q2.6). **As illustrated in the findings from (Q2.1) and (Q2.3), 86% ($\frac{46}{54}$) of participants regarded the defect prediction depicted in the lower portion of Figure 10 as useful. This marks a 14% increase compared to the defect prediction without explanations. Furthermore, 74% ($\frac{40}{54}$) of participants expressed trust in the prediction, representing a notable 21% increase compared to the scenario without explanations.** These results underscore the enhanced usefulness and trustworthiness of our approach among software practitioners. Last but not least, 82% (($\frac{44}{54}$)) of participants indicated their willingness to adopt FoX if integrated into the CI/CD pipeline, such as GitHub action. This result suggests a strong inclination towards the adoption of our approach within software development workflows.

Specifically, the participants perceived that FoX is useful and trustworthy due to various reasons stated in (Q2.2) and (Q2.4):

- **Time & Efficiency** – R7 (an open-source contributor with less than 5 years of experience): *AI-based defect prediction and prevention leverage advanced algorithms to proactively identify potential software defects, enabling organizations to save time, reduce costs, and deliver higher-quality software*; R18 (a SE researcher with 6-10 years of experience): *Save my time to double check and provide me with how*; R45 (a full-stack software engineer with 6-10 years of experience): *Increased accuracy and efficiency: AI can analyze vast amounts of data to identify patterns and anomalies that humans might miss, potentially leading to more accurate defect prediction. This can save time and resources by identifying potential issues early in the development process.*
- **Insight & Understanding** – R19 (a full-stack software engineer with less than 5 years of experience): *This gives a more detailed explanation of the detection, and a brief suggestion on how the risk can be mitigated*; R25 (a software testing engineer with 6-10 years of experience): *Understand why we get this prediction. Explanations are provided*; R32 (a SE researcher with less than 5 years of experience): *The information provided is more than just prediction*; R44 (a full-stack software engineer with 6-10 years of experience): *Providing explanations for why a certain commit or piece of code is flagged as potentially defective helps developers understand the underlying reasons behind the prediction*; R46 (a full-stack software engineer with 6-10 years of experience): *Detecting and predicting a defect is not enough, the explanation gives the reason behind the defect and gives possible solutions to mitigate the defect and give guidance to writing a more effective code.*
- **Trust & Confidence** – R15 (a SE researcher with less than 5 years of experience): *As depicted in Q2.2., providing explanations makes me trust the prediction*; R17 (a SE researcher with less than 5 years of experience): *With the explanation, the prediction of AI system becomes more trustable*; R43 (a software security analyst with 16-20 years of experience): *Given that the information is transparent, I would trust this prediction more than the one provided in the previous section*; R54 (a full-stack software engineer with 6-10 years of experience): *Predicting a defect is only one end of the game and is incomplete without explanations or actionable guidance. The explanation and actionable guidance offers solutions to the defects predicted, why they occur and how it can be sorted out thereby rendering additional boost to the confidence in the prediction.*
- **Practicality & Guidance** – R17 (a SE researcher with less than 5 years of experience): *The explanation is clear and the guidance looks convincing*; R32 (a SE researcher with less than 5

years of experience): *It's practical to have explanations and guidance along with the predictions*; R44 (a full-stack software engineer with 6-10 years of experience): *Providing explanations for why a certain commit or piece of code is flagged as potentially defective helps developers understand the underlying reasons behind the prediction. In addition to explanations, actionable guidance offers concrete suggestions or recommendations on how to address potential defects identified by the AI-based system. This could include specific code changes, best practices, or alternative approaches to mitigate the risk of introducing bugs*; R47 (a full-stack software engineer with 6-10 years of experience): *The reason why I strongly trust the prediction is that it guides the action of the masses.*

Furthermore, the participants' expectations regarding explainable defect prediction, as indicated in their responses to (Q2.6), can be summarized as follows:

- **Role-Specific Defect Prediction** – R11 (a full-stack software engineer with 6-10 years of experience): *We might consider that different roles in the IT team might need different information for defect prediction (developer vs tester vs auditor).*
- **Privacy** – R19 (a full-stack software engineer with less than 5 years of experience): *Reliability and explainability. Privacy issues.*
- **Guiding Corrections** – R25 (a software testing engineer with 6-10 years of experience): *We need to understand why we get the prediction and get hints to change the prediction to the desired one, i.e. non-defective.*
- **More Accurate Models** – R28 (a SE researcher with less than 5 years of experience): *I would expect this system would be able to at least accurately predict 70% to 80% of defects.*
- **User Feedback Mechanisms** – R44 (a full-stack software engineer with 6-10 years of experience): *Implementing feedback mechanisms that enable developers to provide input on the relevance and usefulness of explanations. Gathering feedback from users helps improve the quality and effectiveness of explanations over time by addressing common misunderstandings or areas of confusion.*

Summary. Our survey study with 54 software practitioners provides valuable insights into the usefulness and trustworthiness of our proposed explainable JIT defect prediction. In the scenario where only the defect prediction was presented, 72% ($\frac{39}{54}$) of participants deemed JIT defect prediction useful, even without explanations. However, only 53% ($\frac{29}{54}$) of participants trusted the prediction. This discrepancy underscores the pressing need for our proposed explainable JIT defect prediction approach to bridge the trust gap between defect prediction and end users. In contrast, in the scenario where our defect prediction with explanations and actionable guidance was presented, 86% ($\frac{46}{54}$) of participants found our explainable defect prediction useful. Furthermore, 74% ($\frac{40}{54}$) of participants trusted our explainable defect prediction. They attributed their value to increased efficiency, prediction understanding, trust, and practicality of the actionable guidance. In addition, participants voiced expectations for enhancements, emphasizing the importance of improving model accuracy, integrating a feedback loop to incorporate input from human experts, and tailoring the defect prediction model to users' roles and requirements. Our findings highlight the potential and significance of explainable JIT defect prediction, while also pinpointing areas that require further development and refinement.

8 RELATED WORK

Explainable AI in SE. The explainability of AI models in SE is increasingly important, since prior studies point out that practitioners often do not understand the reasons behind the predictions of software analytics [11, 23, 32, 56]. Such a lack of trust in the predictions hinders the adoption of AI-powered software development tools in practice. Thus, Explainable AI for SE (XAI4SE) is a

newly emerging research topic which aims to increase the explainability and actionability of AI models in SE. Various Explainable AI approaches have been explored in software engineering.

In stark contrast to prior work on explainable AI for SE and to the best of our knowledge, this paper is the first to leverage the formal approach to XAI, a quickly developing area of research that makes a bridge between explainable AI and formal reasoning applied to ML models by exploiting propositional and first-order logic [12, 19, 20, 35, 48]. This way, our paper serves as an example of the synergy between software engineering and formal methods, similar to what has been observed in the recent past in the area of formal software verification [5, 8, 13, 21, 29].

Explainable AI for Defect Prediction. Recent works have shown some successful use cases to make defect prediction models more practical [23, 41, 58], explainable [22, 26], and actionable [43]. However, some of these studies only apply existing model-agnostic techniques from the Explainable AI domain. Recently, Pornprasit *et al.* [42] proposed PyExplainer, a rule-based model-agnostic technique for Just-In-Time defect prediction. However, there exists many limitations (e.g., correctness, robustness, actionability) that have not been addressed.

Different from prior studies of explainable AI for defect prediction, to the best of our knowledge, this paper is the first to address the key limitations of explainability techniques (e.g., PyExplainer, LIME, SHAP, etc) for Just-In-Time defect prediction, demonstrating the significant advancement of explainable AI for defect prediction.

9 LIMITATIONS AND FUTURE WORK

Though experimental results demonstrate that FoX can efficiently compute provably-correct, robust, and actionable explanations, outperforming the four compared state-of-the-art model-agnostic approaches, some limitations may constrict the application of FoX.

Application to other classifiers. Conceptually, FoX can apply to all kinds of classifiers [19, 36]. Nothing prevents one to apply this technology to any classifier that admits a logical representation in some decidable fragment of first-order logic. In practice, however, some classifiers are hard to reason about formally, e.g. deep neural networks, and so it may be computationally expensive to generate provably correct explanations. In this sense, the success and future applicability of formal explainability depends on the future advances in the underlying formal reasoning technology (SAT, SMT, MILP, CP, etc.). In recent JIT defect prediction studies, language models (e.g., BERT) have been employed to directly extract characteristics of defective entities, showcasing promising performance compared to traditional machine learning models. With the rising adoption of language models in JIT defect prediction, it is crucial to explore efficient methods to produce formal explanations tailored for these models. In the future, this investigation will be one of our research directions, aimed at enhancing the practicability of our proposed approach for the advanced JIT defect prediction models. By improving the applicability of our method, practitioners can leverage it effectively to produce correct and robust explanations for language models.

Scalability. Related to the above, although FoX has been empirically demonstrated to be computationally efficient, it can suffer from some scalability problems if a target RF model is extremely complex and its propositional encoding is accordingly large. This is because SAT is a well-known example of an NP-complete problem [9], and the search space a SAT solver has to deal with is determined by the number of propositional variables in the target formula, which can be further affected by the number of clauses. Furthermore, exact FFA computation requires one to enumerate *all* formal explanations, which is computationally even more challenging, but Yu *et al* [62] show that we can *approximate* FFA very closely (i.e. much closer than the results for alternate methods in Table 3) using Algorithm 1 with a small cutoff time. Hence the disadvantages in feature attribution runtime shown in Figure 9 can easily be ameliorated. Note that while extracting a single formal explanation for LR models can be done in polynomial time [35], exact FFA computation is still

quite competitive due to the need to enumerate all such explanations. In the future, we will aim at addressing the scalability problems associated with RF models by proposing alternative (simpler) RF encodings into propositional logic as well as investigating alternative ways for effective FFA approximation.

Explanation Ordering. Users may opt to select only the first generated explanation even though FoX can generate a pool of candidate explanations. Although the generic explanation enumeration approach applied in FoX, supports generating explanations with the preference of small size, the bespoke alternative for the case of monotonic classifiers (see Section 4.1) is unable to do so. Nevertheless, one may still apply the same generic approach to monotonic classifiers as well (although the performance of explanation enumeration may be slightly affected). Thus, our future work will also include the extension of explanation size ordering for monotonic classifiers and the support for more feature orderings in FoX.

Actionability of Explanations. In this article, our focus is on proposing, evaluating, and implementing a more robust formal technique for explaining JIT defect predictions. Additionally, we conduct a user study to assess the qualitative aspects of our proposed approach. While recognizing the importance of clarifying the actionability of these explanations—specifically, how helpful feature-level JIT explanations are to developers—we acknowledge that this depends on how metrics are operationalized, which falls outside the scope of our current work. Thus, in future research, we plan to conduct a user study involving debugging tasks within a CI/CD pipeline. This study aims to evaluate the extent to which metrics provided by JIT explanations aid developers in saving time and effort, with a focus on recording human behaviors to understand their responses to the reported explanations more comprehensively.

10 CONCLUSION

In this paper, we propose FoX, the first formal explainer for Just-In-Time defect prediction. The FoX explainer builds on the use of formal reasoning by exploiting propositional logic in order to efficiently generate provably-correct, robust, and subset-minimal explanations answering the *why?* questions, e.g. why a commit is predicted as defective. Our formal explainer can also generate provably-correct, robust, and subset-minimal contrastive *how?* explanations, e.g. how should developers mitigate the risk, which are more actionable than the usual abductive explanations, serving as an important step towards actionable software analytics. As a result, the generated explanations are expected to help researchers better design actionable defect prediction approaches and help practitioners better focus on the most important aspects associated with software defects to improve the operational decisions of SQA teams.

REFERENCES

[1] Amritanshu Agrawal and Tim Menzies. 2018. Is Better Data Better Than Better Data Miners?: On the Benefits of Tuning SMOTE for Defect Prediction. In *ICSE*. 1050–1061.
[2] Reem Aleithan. 2021. Explainable just-in-time bug prediction: are we there yet?. In *ICSE-Companion*. 129–131.
[3] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (Eds.). 2021. *Handbook of Satisfiability*. IOS Press.
[4] Leo Breiman. 2001. Random Forests. *Mach. Learn.* 45, 1 (2001), 5–32.
[5] Cristian Cadar and Koushik Sen. 2013. Symbolic execution for software testing: three decades later. *Commun. ACM* 56, 2 (2013), 82–90.
[6] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* (2002), 321–357.
[7] Di Chen, Wei Fu, Rahul Krishna, and Tim Menzies. 2018. Applications of Psychological Science for Actionable Analytics. In *ESEC/FSE*. 456–467.
[8] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron A. Peled, and Helmut Veith. 2018. *Model checking*.
[9] Stephen A. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *STOC*. ACM, 151–158.

[10] Daniel Alencar da Costa, Shane McIntosh, Weiyi Shang, Uirá Kulesza, Roberta Coelho, and Ahmed E Hassan. 2017. A Framework for Evaluating the Results of the SZZ Approach for Identifying Bug-introducing Changes. *IEEE Transactions on Software Engineering (TSE)* 43, 7 (2017), 641–657.
[11] Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. 2018. Explainable Software Analytics. In *ICSE-NIER*. 53–56.
[12] Adnan Darwiche and Auguste Hirth. 2020. On the Reasons Behind Decisions. In *ECAI*. 712–720.
[13] Vijay Victor D'Silva, Daniel Kroening, and Georg Weissenbacher. 2008. A Survey of Automated Techniques for Formal Software Verification. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 27, 7 (2008), 1165–1178.
[14] Thong Hoang, Hoa Khanh Dam, Yasutaka Kamei, David Lo, and Naoyasu Ubayashi. 2019. DeepJIT: an end-to-end deep learning framework for just-in-time defect prediction. In *MSR*. 34–45.
[15] Thong Hoang, Hong Jin Kang, David Lo, and Julia Lawall. 2020. CC2Vec: Distributed representations of code changes. In *ICSE*. 518–529.
[16] Xuanxiang Huang and Joao Marques-Silva. 2023. The Inadequacy of Shapley Values for Explainability. *CoRR* abs/2302.08160 (2023).
[17] Alexey Ignatiev. 2020. Towards Trustable Explainable AI. In *IJCAI*. 5154–5158.
[18] Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva. 2020. From Contrastive to Abductive Explanations and Back Again. In *AI*IA*. 335–355.
[19] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. 2019. Abduction-Based Explanations for Machine Learning Models. In *AAAI*. 1511–1519.
[20] Yacine Izza and João Marques-Silva. 2021. On Explaining Random Forests with SAT. In *IJCAI*. 2584–2591.
[21] Ranjit Jhala and Rupak Majumdar. 2009. Software model checking. *ACM Comput. Surv.* 41, 4 (2009), 21:1–21:54.
[22] Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, Hoa Khanh Dam, and John Grundy. 2020. An Empirical Study of Model-Agnostic Techniques for Defect Prediction Models. *IEEE Transactions on Software Engineering (TSE)* (2020), 166–185.
[23] Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, and John Grundy. 2021. Practitioners' Perceptions of the Goals and Visual Explanations of Defect Prediction Models. In *MSR*. 432–443.
[24] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E. Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. 2013. A Large-Scale Empirical Study of Just-In-Time Quality Assurance. *IEEE Transactions on Software Engineering (TSE)* 39, 6 (2013), 757–773.
[25] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.
[26] Chaiyakarn Khanan, Worawit Luewichana, Krissakorn Pruktharathikoon, Jirayus Jiarpakdee, Chakkrit Tantithamthavorn, Morakot Choetkiertikul, Chaiyong Ragkhitwetsagul, and Thanwadee Sunetnanta. 2020. JITBot: An Explainable Just-In-Time Defect Prediction Bot. In *ASE*. 1336–1339.
[27] Sunghun Kim, Thomas Zimmermann, E James Whitehead Jr, and Andreas Zeller. 2007. Predicting Faults from Cached History. In *ICSE*. 489–498.
[28] Barbara A Kitchenham and Shari L Pfleeger. 2008. Personal opinion surveys. In *Guide to Advanced Empirical Software Engineering*. Springer, 63–92.
[29] Anvesh Komuravelli, Arie Gurfinkel, Sagar Chaki, and Edmund M. Clarke. 2013. Automatic Abstraction in SMT-Based Unbounded Software Model Checking. In *CAV*. 846–862.
[30] Rahul Krishna and Tim Menzies. 2020. Learning Actionable Analytics from Multiple Software Projects. *Empirical Software Engineering (EMSE)* (2020), 3468–3500.
[31] Himabindu Lakkaraju and Osbert Bastani. 2020. "How do I fool you?": Manipulating User Trust via Misleading Black Box Explanations. In *AIES*. 79–85.
[32] Chris Lewis, Zhongpeng Lin, Caitlin Sadowski, Xiaoyan Zhu, Rong Ou, and E James Whitehead Jr. 2013. Does Bug Prediction Support Human Developers? Findings from a Google Case Study. In *ICSE*. 372–381.
[33] Dayi Lin, Chakkrit Tantithamthavorn, and Ahmed E Hassan. 2021. The Impact of Data Merging on the Interpretation of Cross-Project Just-In-Time Defect Models. *IEEE Transactions on Software Engineering* (2021).
[34] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NIPS*. 4765–4774.
[35] João Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska. 2021. Explanations for Monotonic Classifiers. In *ICML*. 7469–7479.
[36] Joao Marques-Silva and Alexey Ignatiev. 2022. Delivering Trustworthy AI through Formal XAI. In *AAAI*. 12342–12350.
[37] Shane McIntosh and Yasutaka Kamei. 2017. Are fix-inducing changes a moving target? a longitudinal case study of just-in-time defect prediction. *IEEE Transactions on Software Engineering (TSE)* (2017), 412–428.
[38] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* 267 (2019), 1–38.
[39] Nina Narodytska, Aditya A. Shrotri, Kuldeep S. Meel, Alexey Ignatiev, and Joao Marques-Silva. 2019. Assessing Heuristic Machine Learning Explanations with Model Counting. In *SAT*. 267–278.
[40] Kewen Peng and Tim Menzies. 2021. Defect Reduction Planning (using TimeLIME). *IEEE Transactions on Software Engineering (TSE)* (2021).

[41] Chanathip Pornprasit and Chakkrit Tantithamthavorn. 2021. JITLine: A Simpler, Better, Faster, Finer-grained Just-In-Time Defect Prediction. In *MSR*. 369–379.
[42] Chanathip Pornprasit, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Michael Fu, and Patanamon Thongtanunam. 2021. PyExplainer: Explaining the Predictions of Just-In-Time Defect Models. In *ASE*. 407–418.
[43] Dilini Rajapaksha, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Christoph Bergmeir, John Grundy, and Wray Buntine. 2021. SQAPlanner: Generating Data-Informed Software Quality Improvement Plans. *IEEE Transactions on Software Engineering (TSE)* (2021).
[44] Raymond Reiter. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.* 32, 1 (1987), 57–95.
[45] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the Predictions of Any Classifier. In *KDD*. 1135–1144.
[46] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*. 1527–1535.
[47] LLoyd S. Shapley. 1953. A Value of n-Person Games. *Contributions to the Theory of Games* 2, 28 (1953), 307–317.
[48] Andy Shih, Arthur Choi, and Adnan Darwiche. 2018. A Symbolic Approach to Explaining Bayesian Network Classifiers. In *IJCAI*. 5103–5111.
[49] Jiho Shin, Reem Aleithan, Jaechang Nam, Junjie Wang, and Song Wang. 2021. Explainable Software Defect Prediction: Are We There Yet? *arXiv preprint arXiv:2111.10901* (2021).
[50] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *AIES*. 180–186.
[51] Dylan Z Slack, Sophie Hilgard, Sameer Singh, and Himabindu Lakkaraju. 2021. Reliable Post hoc Explanations: Modeling Uncertainty in Explainability. In *NeurIPS*.
[52] Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. 2005. When do changes induce fixes?. In *MSR*. 1–5.
[53] Chakkrit Tantithamthavorn, Ahmed E Hassan, and Kenichi Matsumoto. 2020. The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. *IEEE Transactions on Software Engineering (TSE)* 46, 11 (2020), 1200–1219.
[54] Chakkrit Tantithamthavorn and Jirayus Jiarpakdee. 2021. Explainable AI for Software Engineering. In *ASE*. 1–2.
[55] Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, and John Grundy. 2020. Explainable AI for Software Engineering. *arXiv preprint arXiv:2012.01614* (2020).
[56] Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, and John Grundy. 2021. Actionable Analytics: Stop Telling Me What It Is; Please Tell Me What To Do. *IEEE Software* 38, 4 (2021), 115–120.
[57] Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E Hassan, and Kenichi Matsumoto. 2017. An Empirical Comparison of Model Validation Techniques for Defect Prediction Models. *IEEE Transactions on Software Engineering (TSE)* 43, 1 (2017), 1–18.
[58] Supatsara Wattanakriengkrai, Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Hideaki Hata, and Kenichi Matsumoto. 2020. Predicting Defective Lines Using a Model-Agnostic Technique. *IEEE Transactions on Software Engineering (TSE)* (2020).
[59] William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)* 28, 4 (2010), 1–38.
[60] Ye Yang, Davide Falessi, Tim Menzies, and Jairus Hihn. 2017. Actionable analytics for software engineering. *IEEE Software* (2017), 51–53.
[61] Suraj Yathish, Jirayus Jiarpakdee, Patanamon Thongtanunam, and Chakkrit Tantithamthavorn. 2019. Mining Software Defects: Should We Consider Affected Releases?. In *ICSE*. 654–665.
[62] Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. 2023. On Formal Feature Attribution and Its Approximation. *CoRR* abs/2307.03380 (2023). https://doi.org/10.48550/arXiv.2307.03380 arXiv:2307.03380

Chapter 9

Conclusions and Future Work

Conclusions

While machine learning (ML) and Artificial Intelligence (AI) advancements have been widely applied across diverse domains like finance, the lack of transparency in these systems triggers significant concerns, e.g. fairness, bias, safety and robustness. As a result, there is a growing interest of eXplainable AI (XAI) aimed to establish trust in ML/ AI systems, by explaining or illustrating the behavior of ML models in human-understandable ways. Among various approaches to XAI, model-agnostic approaches are stand out as the prevailing ones although they encounter challenges regarding the quality of generated explanations. As an alternative, formal XAI (FXAI) methods offer logic-based or formal explanations, aimed to provide provable and robust explanations for ML predictions. This thesis concentrates on XAI challenges, especially formal explainability, targeting enhancing the interpretability of ML models and addressing gaps in formal explainability. The primary contributions of this thesis are outlined as follows:

- Chapter 3 introduces methods to generate decision sets and decision lists that are optimal either in terms of the number of required literals or the trade-off between model size and accuracy. In this study, we define size as the total number literals used in these rule-based ML models, in contrast with previous research that primarily focuses on the number of rules in the model, which cannot adequately capture the explainability of such models. The empirical results illustrate that the decision sets and lists computed by the proposed approaches are able to achieve decent trade-off between interpretability (represented by model size) and accuracy.
- In Chapter 4, we present an innovative anytime approach to producing decision sets through the on-demand extraction of generalized abductive explanations for

boosted trees. The proposed method can be used to compile a gradient boosted tree with respect to either the complete feature space, or a set of target training instances. Augmented by post-hoc model size reduction approaches, this method is demonstrated to generate decision sets that outperform those computed by state-of-the-art algorithms in terms of accuracy accuracy and remain comparable with them regarding explanation size. Note that the presented method can be categorized as a knowledge distillation technique and theoretically, it can extend to any other ML models.

- In Chapter 5, we propose a technique to apply background knowledge to enhance the quality of explanations. There are substantial advantages of incorporating background knowledge in generating formal explanations for ML models. In the context of abductive explanations (AXp's), integrating background knowledge significantly reduces the length of explanations, making them more explainable, and improve the efficiency of explanation generation. In the case of contrastive explanations (CXps), although applying background knowledge may increase the explanation size and potentially require more time for explanation generation, the resulting explanations are significantly more precise because they do not depend on the (often unsupportable) assumption that all tuples in the feature space are feasible. Moreover, as demonstrated in Chapter 5, background knowledge can be integrated into the context of heuristic explanations, particularly when an accurate analysis of is need.

- Chapter 6 introduces the first method for formal feature attribution (FFA), using the proportion of abductive explanations where a feature appears to indicate its significance. Experimental results demonstrate that we can compute exact FFA for numerous classification tasks, and in cases where exact computation is infeasible, we can compute effective approximations. Additionally, we found existing model-agnostic method to generate to feature attribution disagree with FFA. In some cases, they are considerably different from FFA, such as assigning no weight to a feature that is present in (a significant number of) explanations, or assigning a (large) non-zero weight to a feature that holds no relevance for the prediction. Overall, Chapter 6 argues that if we acknowledge FFA as a valid measure of feature attribution, it is important to explore approaches that generate good approximate FFA more efficiently.

- In Chapter 7, motivated by the difficulty of exact computation for many classifiers and datasets, we introduce an anytime method for FFA approximation. As illustrated in this chapter, computing FFA remains challenging even when the set

of CXp's is available. Therefore, there is a demand for anytime method to generate FFA. Surprisingly, starting with CXp enumeration to produce AXps results in rapidly achieving good approximations of FFA. However, in the longer turn this method proves to be less effective compared to simply enumerating AXps. This work demonstrates the integration of these approaches by strategically switching the enumeration phase, ensuring that information computed in the underlying MARCO enumeration algorithm is preserved. This presents a highly practical method to generate FFA.

- Chapter 8 introduces the first formal explainer for just-in-time (JIT) defect prediction. This novel explainer leverages formal reasoning by exploiting propositional logic to efficiently produce explanations that are provably correct, robust, and subset-minimal, addressing "why" questions, such as why a commit is predicted as defective. The proposed formal explainer is also capable of producing contrastive "how" explanations that are provably correct, robust, and subset-minimal, offering more actionable insights compared to abductive explanations. This marks a significant step towards actionable software analytics. Therefore, these generated explanations are able to assist researchers in designing actionable defect prediction methods and help practitioner direct their attention towards the most crucial aspects related to software defects. This, in turn, enhances the operational decision-making of software quality assurance (SQA) teams.

Feature work

While this thesis presents numerous advancements in the field of XAI, there are still several directions that need to be addressed in future research. Some future directions are outlined as follows.

Applying Formal Explainability. Although ML and AI are widely used in diverse fields, such as healthcare, law, finance, and transportation, the opacity of ML/ AI systems can hinder users from gaining clear insights into the outputs produced by these systems. Consequently, these systems fail to establish trust between humans and the predictions generated by ML models. Inspired by the limitation, in Chapter 8 we applies formal explainability for just-in-time (JIT) defect prediction, aiming to establish trust for the predictions. To broaden the applicability of ML and AI applications, one future research direction will focus on applying formal explainability across various fields. This will help users trust the predictions generated by ML models, facilitating the wider adoption of ML and AI in diverse domains.

Improving Scalability of Formal Explainability. With the increasing need to deploy large and complex ML/ AI systems across diverse domains, the scalability of XAI approaches has emerged as a notable concern. Unfortunately, formal explainability encounters scalability challenges, particularly with certain complex classifier families, such as neural networks. These models are computationally expensive or hard to reason formally to produce formal explanations due to their complex structure. To address this scalability limitation, recent research [19] suggests a novel method to approximate formal explanations, with the use of technologies aimed at evaluating the robustness of neural networks. Additionally, recent research [19] introduces innovative algorithms for computing formal explanations, achieved by finding a direct relationship between the practical complexity of robustness and formal explainability. Inspired by the scalability issue and these studies, one feature research direction will focus on developing methods to efficiently generate formal explanations for complex ML models.

Bibliography

[1] Rusul Abduljabbar, Hussein Dia, Sohani Liyanage, and Saeed Asadi Bagloee. Applications of artificial intelligence in transport: An overview. *Sustainability*, 11: 189, 2019.

[2] ACM. Fathers of the deep learning revolution receive ACM A.M. Turing award. http://tiny.cc/9plzpz, 2018.

[3] HLEG AI. Ethics guidelines for trustworthy ai. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai, 2010.

[4] HLEG AI. Assessment list for trustworthy artificial intelligence (altai) for self-assessment. https://bit.ly/3jAeHds, 2020.

[5] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: a preliminary study using GPT-4. *CoRR*, abs/2311.07361, 2023.

[6] Encarnación Algaba, Vito Fragnelli, and Joaquín Sánchez-Soriano. *Handbook of the Shapley value*. CRC Press, 2019.

[7] Leila Amgoud. Non-monotonic explanation functions. In *ECSQARU*, pages 19–31, 2021.

[8] Leila Amgoud and Jonathan Ben-Naim. Axiomatic foundations of explainability. In Luc De Raedt, editor, *IJCAI*, pages 636–642, 2022.

[9] Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11:e1424, 2021.

[10] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. http://tiny.cc/dd7mjz, 2016.

[11] Marcelo Arenas, Daniel Baez, Pablo Barceló, Jorge Pérez, and Bernardo Subercaseaux. Foundations of symbolic languages for model interpretability. In *NeurIPS*, pages 11690–11701, 2021.

[12] Marcelo Arenas, Pablo Barceló, Leopoldo E. Bertossi, and Mikaël Monet. The tractability of SHAP-score-based explanations for classification over deterministic and decomposable Boolean circuits. In *AAAI*, pages 6670–6678, 2021.

[13] Marcelo Arenas, Pablo Barceló, Leopoldo E. Bertossi, and Mikaël Monet. On the complexity of SHAP-score-based explanations: Tractability via knowledge compilation and non-approximability results. *CoRR*, abs/2104.08015, 2021.

[14] Marcelo Arenas, Pablo Barceló, Miguel A. Romero Orth, and Bernardo Subercaseaux. On computing probabilistic explanations for decision trees. In *NeurIPS*, 2022.

[15] Henri Arslanian and Fabrice Fischer. *The future of finance: The impact of FinTech, AI, and crypto on financial services*. Springer, 2019.

[16] Gilles Audemard, Frédéric Koriche, and Pierre Marquis. On tractable xai queries based on compiled representations. In *KR*, pages 838–849, 2020.

[17] Fahiem Bacchus and George Katsirelos. Using minimal correction sets to more efficiently compute minimal unsatisfiable sets. In *CAV*, pages 70–86, 2015.

[18] James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pages 174–186, 2005.

[19] Shahaf Bassan and Guy Katz. Towards formal xai: formally approximate minimal explanations of neural networks. In *TACAS*, pages 187–207, 2023.

[20] Ali Behrouz, Mathias Lécuyer, Cynthia Rudin, and Margo Seltzer. Fast optimization of weighted sparse decision trees for use in optimal treatment regimes and optimal policy design. In *CEUR workshop proceedings*, volume 3318, 2022.

[21] Jaroslav Bendík, Ivana Cerná, and Nikola Benes. Recursive online enumeration of all minimal unsatisfiable subsets. In *ATVA*, pages 143–159, 2018.

[22] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability: Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 2021. IOS Press.

[23] Elazar Birnbaum and Eliezer L. Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46, 2003.

[24] Guy Blanc, Jane Lange, and Li-Yang Tan. Provably efficient, succinct, and precise explanations. In *NeurIPS*, 2021.

[25] Ryma Boumazouza, Fahima Cheikh Alili, Bertrand Mazure, and Karim Tabia. ASTERYX: A model-Agnostic SaT-basEd appRoach for sYmbolic and score-based eXplanations. In *CIKM*, pages 120–129, 2021.

[26] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.

[27] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020.

[28] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712, 2023.

[29] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559:547–555, 2018.

[30] Oana-Maria Camburu, Eleonora Giunchiglia, Jakob N. Foerster, Thomas Lukasiewicz, and Phil Blunsom. Can I trust the explainer? verifying post-hoc explanatory methods. *CoRR*, abs/1910.02065, 2019.

[31] Manuel Carabantes. Black-box artificial intelligence: an epistemological and critical analysis. *AI & society*, 35(2):309–317, 2020.

[32] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2023.

[33] Chaofan Chen, Kangcheng Lin, Cynthia Rudin, Yaron Shaposhnik, Sijia Wang, and Tong Wang. A holistic approach to interpretability in financial lending: Models, visualizations, and summary-explanations. *Decision Support Systems*, 152: 113647, 2022.

[34] Mark Coeckelbergh. *AI ethics*. Mit Press, 2020.

[35] Mike Daily, Swarup Medasani, Reinhold Behringer, and Mohan Trivedi. Self-driving cars. *Computer*, 50(12):18–23, 2017.

[36] DARPA. DARPA explainable Artificial Intelligence (XAI) program. https://www.darpa.mil/program/explainable-artificial-intelligence, 2016.

[37] Adnan Darwiche. Logic for explainable ai. In *LICS*, pages 1–11, 2023.

[38] Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. In *ECAI*, pages 712–720, 2020.

[39] Adnan Darwiche and Pierre Marquis. On quantifying literals in Boolean logic and its applications to explainable AI. *J. Artif. Intell. Res.*, 72:285–328, 2021.

[40] Catherine O de Burgh-Day and Tennessee Leeuwenburg. Machine learning for numerical weather and climate modelling: a review. *Geoscientific Model Development*, 16:6433–6477, 2023.

[41] Alexandra DeArman. The wild, wild west: A case study of self-driving vehicle testing in arizona. *Ariz. L. Rev.*, 61:983, 2019.

[42] Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of SHAP explanations. *J. Artif. Intell. Res.*, 74:851–886, 2022.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *NAACL-HLT*, pages 4171–4186, 2019.

[44] Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. *ECAI*, pages 2473–2480, 2020.

[45] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[46] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14, 1990.

[47] EU. Artificial intelligence act. http://tiny.cc/ahcnuz, 2021.

[48] EU. Coordinated plan on artificial intelligence – 2021 review. https://bit.ly/3hJG2HF, 2021.

[49] João Ferreira, Manuel de Sousa Ribeiro, Ricardo Gonçalves, and João Leite. Looking inside the black-box: Logic-based explanations for neural networks. In *KR*, page 432–442, 2022.

[50] Veniamin Fishman, Maria Sindeeva, Nikolay Chekanov, Tatiana Shashkova, Nikita Ivanisenko, and Olga Kardymon. Ai in genomics and epigenomics. In *Artificial Intelligence for Healthy Longevity*, pages 217–243, 2023.

[51] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36, 1980.

[52] Bishwamittra Ghosh and Kuldeep S. Meel. IMLI: an incremental framework for maxsat-based learning of interpretable classification rules. In *AIES*, pages 203–210. ACM, 2019.

[53] John W Goodell, Satish Kumar, Weng Marc Lim, and Debidutta Pattnaik. Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. *Journal of Behavioral and Experimental Finance*, 32:100577, 2021.

[54] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[55] Niku Gorji and Sasha Rubin. Sufficient reasons for classifier decisions in the presence of domain constraints. In *AAAI*, pages 5660–5667, 2022.

[56] Éric Grégoire, Yacine Izza, and Jean-Marie Lagniez. Boosting MCSes enumeration. In *IJCAI*, pages 1309–1315, 2018.

[57] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.

[58] Benjamin Haibe-Kains, George Alexandru Adam, Ahmed Hosny, Farnoosh Khodakarami, Massive Analysis Quality Control (MAQC) Society Board of Directors Shraddha Thakkar 35 Kusko Rebecca 36 Sansone Susanna-Assunta 37 Tong Weida 35 Wolfinger Russ D. 38 Mason Christopher E. 39 Jones Wendell 40 Dopazo Joaquin 41 Furlanello Cesare 42, Levi Waldron, Bo Wang, Chris McIntosh, Anna Goldenberg, Anshul Kundaje, et al. Transparency and reproducibility in artificial intelligence. *Nature*, 586:E14–E16, 2020.

[59] Ammar Haydari and Yasin Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23:11–32, 2020.

[60] Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable ai methods-a brief overview. In *xxAI*, pages 13–38, 2022.

[61] Joo-Wha Hong, Ignacio Cruz, and Dmitri Williams. Ai, you can drive my car: How we evaluate human drivers vs. self-driving cars. *Computers in Human Behavior*, 125:106944, 2021.

[62] Xiyang Hu, Yan Huang, Beibei Li, and Tian Lu. Uncovering the source of machine bias. *CoRR*, abs/2201.03092, 2022.

[63] Xuanxiang Huang and João Marques-Silva. The inadequacy of shapley values for explainability. *CoRR*, abs/2302.08160, 2023.

[64] Xuanxiang Huang and João Marques-Silva. A refutation of shapley values for explainability. *CoRR*, abs/2309.03041, 2023.

[65] Xuanxiang Huang and João Marques-Silva. Refutation of shapley values for XAI - additional evidence. *CoRR*, abs/2310.00416, 2023.

[66] Xuanxiang Huang and João Marques-Silva. From robustness to explainability and back again. *CoRR*, abs/2306.03048, 2023.

[67] Xuanxiang Huang and João Marques-Silva. The inadequacy of Shapley values for explainability. *CoRR*, abs/2302.08160, 2023.

[68] Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On efficiently explaining graph-based classifiers. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *KR*, pages 356–367, 2021.

[69] Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, Martin Cooper, Nicholas Asher, and Joao Marques-Silva. Tractable explanations for d-dnnf classifiers. In *AAAI*, volume 36, pages 5719–5728, 2022.

[70] Xuanxiang Huang, Martin C. Cooper, António Morgado, Jordi Planes, and João Marques-Silva. Feature necessity & relevancy in ML classifier explanations. In *TACAS (1)*, pages 167–186, 2023.

[71] Xuanxiang Huang, Yacine Izza, and Joao Marques-Silva. Solving explainability queries with quantification: The case of feature relevancy. In *AAAI*, volume 37, pages 3996–4006, 2023.

[72] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[73] Alexey Ignatiev. Towards trustable explainable ai. In *IJCAI*, pages 5154–5158, 2020.

[74] Alexey Ignatiev and João P. Marques Silva. Sat-based rigorous explanations for decision lists. In Chu-Min Li and Felip Manyà, editors, *SAT*, volume 12831 of *Lecture Notes in Computer Science*, pages 251–269, 2021.

[75] Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and João Marques-Silva. A SAT-based approach to learn explainable decision sets. In *IJCAR*, pages 627–645, 2018.

[76] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

[77] Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. On validating, repairing and refining heuristic ml explanations. *CoRR*, abs/1907.02509, 2019.

[78] Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva. From contrastive to abductive explanations and back again. In *AI*IA*, pages 335–355, 2020.

[79] Alexey Ignatiev, Edward Lam, Peter J. Stuckey, and Joao Marques-Silva. A scalable two stage approach to computing optimal decision sets. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*, page to appear, 2021.

[80] Alexey Ignatiev, Edward Lam, Peter J. Stuckey, and Joao Marques-Silva. A scalable two stage approach to computing optimal decision sets. In *34th AAAI Conference on Artificial Intelligence (AAAI 2021)*, page To Appear, 2021.

[81] Alexey Ignatiev, Joao Marques-Silva, Nina Narodytska, and Peter J Stuckey. Reasoning-based learning of interpretable ml models. In *IJCAI*, pages 4458–4465, 2021.

[82] Alexey Ignatiev, Yaccine Izza, Peter J Stuckey, and Joao Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *AAAI*. AAAI, 2022.

[83] Yacine Izza and João Marques-Silva. On explaining random forests with SAT. In *IJCAI*, pages 2584–2591, 2021.

[84] Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On explaining decision trees. *CoRR*, abs/2010.11034, 2020.

[85] Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On tackling explanation redundancy in decision trees. *J. Artif. Intell. Res.*, 75:261–321, 2022.

[86] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *AISTATS*, pages 2907–2916, 2020.

[87] Anna Jobin, Marcello Ienca, and Effy Vayena. The global landscape of ai ethics guidelines. *Nature machine intelligence*, 1:389–399, 2019.

[88] Christopher Kadow, David Matthew Hall, and Uwe Ulbrich. Artificial intelligence reconstructs missing climate information. *Nature Geoscience*, 13:408–413, 2020.

[89] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E. Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. A Large-Scale Empirical Study of Just-In-Time Quality Assurance. *IEEE Transactions on Software Engineering (TSE)*, 39 (6):757–773, 2013.

[90] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

[91] Davinder Kaur, Suleyman Uslu, Kaley J Rittichier, and Arjan Durresi. Trustworthy artificial intelligence: a review. *ACM computing surveys (CSUR)*, 55:1–38, 2022.

[92] Sunghun Kim, Thomas Zimmermann, E James Whitehead Jr, and Andreas Zeller. Predicting Faults from Cached History. In *ICSE*, pages 489–498, 2007.

[93] Diederik P. Kingma and Max Welling. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1312.6114, 2013.

[94] I. Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle A. Friedler. Problems with shapley-value-based explanations as feature importance measures. In *ICML*, volume 119, pages 5491–5500, 2020.

[95] Jishitha Kuppala, K Kalyana Srinivas, P Anudeep, R Sravanth Kumar, and PA Harsha Vardhini. Benefits of artificial intelligence in the legal system and law enforcement. In *MECON*, pages 221–225, 2022.

[96] Himabindu Lakkaraju and Osbert Bastani. "how do I fool you?": Manipulating user trust via misleading black box explanations. In *AIES*, pages 79–85, 2020.

[97] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684, 2016.

[98] Stefan Larsson and Fredrik Heintz. Transparency in artificial intelligence. *Internet Policy Review*, 9, 2020.

[99] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86, 1998.

[100] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436, 2015.

[101] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Computing Surveys*, 55: 1–46, 2023.

[102] Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40:1–33, 2008.

[103] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2):223–250, 2016.

[104] Dayi Lin, Chakkrit Tantithamthavorn, and Ahmed E Hassan. The impact of data merging on the interpretation of cross-project just-in-time defect models. *IEEE Transactions on Software Engineering*, 2021.

[105] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61 (10):36–43, 2018.

[106] Jiachang Liu, Chudi Zhong, Boxuan Li, Margo Seltzer, and Cynthia Rudin. Fasterrisk: Fast and accurate interpretable risk scores. *NeurIPS*, 35:17760–17773, 2022.

[107] Xinghan Liu and Emiliano Lorini. A logic for binary classifiers and their explanation. In *CLAR*, pages 302–321, 2021.

[108] Octavio Loyola-Gonzalez. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE access*, 7:154096–154113, 2019.

[109] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774, 2017.

[110] Emanuele La Malfa, Rhiannon Michelmore, Agnieszka M. Zbrzezny, Nicola Paoletti, and Marta Kwiatkowska. On guaranteed optimal robust explanations for NLP models. In *IJCAI*, pages 2658–2665, 2021.

[111] Paras Malik, Monika Pathania, Vyas Kumar Rathaur, et al. Overview of artificial intelligence in medicine. *Journal of family medicine and primary care*, 8:2328–2331, 2019.

[112] Dmitry Malioutov and Kuldeep S. Meel. MLIC: A maxsat-based framework for learning interpretable classification rules. In *CP*, pages 312–327, 2018.

[113] João Marques-Silva. Logic-based explainability in machine learning. In Leopoldo E. Bertossi and Guohui Xiao, editors, *Reasoning Web*, pages 24–104, 2022.

[114] João Marques-Silva and Alexey Ignatiev. Delivering trustworthy ai through formal xai. In *AAAI*, pages 12342–12350, 2022.

[115] João Marques-Silva and Alexey Ignatiev. No silver bullet: interpretable ML models must be explained. *Frontiers Artif. Intell.*, 6, 2023.

[116] Joao Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska. Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*, 2020.

[117] João Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska. Explanations for monotonic classifiers. In *ICML*, pages 7469–7479, 2021.

[118] Shane McIntosh and Yasutaka Kamei. Are fix-inducing changes a moving target? A longitudinal case study of just-in-time defect prediction. *IEEE Trans. Software Eng.*, 44(5):412–428, 2018.

[119] Yusuf Mehdi. Reinventing search with a new ai-powered microsoft bing and edge, your copilot for the web. https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edge-your-copilot-f 2023.

[120] Carlos Mencía, Alessandro Previti, and João Marques-Silva. Literal-based MCS extraction. In *IJCAI*, pages 1973–1979, 2015.

[121] Carlos Mencia, Alexey Ignatiev, Alessandro Previti, and Joao Marques-Silva. MCS extraction with sublinear oracle queries. In *SAT*, pages 342–360, 2016.

[122] Ryszard S Michalski. On the quasi-minimal solution of the general covering problem. In *International Symposium on Information Processing*, pages 125–128, 1969.

[123] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019.

[124] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[125] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[126] Christoph Molnar. *Interpretable Machine Learning*. Leanpub, 2020. http://tiny.cc/6c76tz.

[127] Nina Narodytska, Nikolaj Bjørner, Maria-Cristina V. Marinescu, and Mooly Sagiv. Core-guided minimal correction set and core enumeration. In *IJCAI*, pages 1353–1361, 2018.

[128] Nina Narodytska, Aditya A. Shrotri, Kuldeep S. Meel, Alexey Ignatiev, and Joao Marques-Silva. Assessing heuristic machine learning explanations with model counting. In *SAT*, pages 267–278, 2019.

[129] Alexandros Nikitas, Kalliopi Michalakopoulou, Eric Tchouamou Njoya, and Dimitris Karampatzakis. Artificial intelligence, transport and the smart city: Definitions and dimensions of a new mobility era. *Sustainability*, 12:2789, 2020.

[130] OECD. Recommendation of the council on artificial intelligence. https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449, 2021.

[131] Andreas S Panayides, Amir Amini, Nenad D Filipovic, Ashish Sharma, Sotirios A Tsaftaris, Alistair Young, David Foran, Nhan Do, Spyretta Golemati, Tahsin Kurc, et al. Ai in medical imaging informatics: current challenges and future directions. *IEEE journal of biomedical and health informatics*, 24:1837–1857, 2020.

[132] Chanathip Pornprasit and Chakkrit Tantithamthavorn. JITLine: A Simpler, Better, Faster, Finer-grained Just-In-Time Defect Prediction. In *MSR*, pages 369–379, 2021.

[133] Chanathip Pornprasit, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Michael Fu, and Patanamon Thongtanunam. PyExplainer: Explaining the predictions of Just-In-Time defect models. In *ASE*, pages 407–418, 2021.

[134] Alessandro Previti, Carlos Mencía, Matti Järvisalo, and João Marques-Silva. Premise set caching for enumerating minimal correction subsets. In *AAAI*, pages 6633–6640, 2018.

[135] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[136] J. Ross Quinlan. *C4.5: Programs for machine learning*. Morgan Kauffmann, 1993.

[137] Stephan Raaijmakers. Artificial intelligence for law enforcement: challenges and opportunities. *IEEE security & privacy*, 17:74–77, 2019.

[138] Antonio Rago, Oana Cocarascu, Christos Bechlivanidis, and Francesca Toni. Argumentation as a framework for interactive explanations for recommendations. In *KR*, volume 17, pages 805–815, 2020.

[139] Antonio Rago, Oana Cocarascu, Christos Bechlivanidis, David Lagnado, and Francesca Toni. Argumentative explanations for interactive recommendations. *Artif. Intell.*, 296:103506, 2021.

[140] Arun Rai. Explainable ai: From black box to glass box. *Journal of the Academy of Marketing Science*, 48:137–141, 2020.

[141] Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. Ai in health and medicine. *Nature medicine*, 28:31–38, 2022.

[142] Sandeep Reddy, Sonia Allan, Simon Coghlan, and Paul Cooper. A governance model for the application of ai in health care. *Journal of the American Medical Informatics Association*, 27:491–497, 2020.

[143] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32: 57–95, 1987.

[144] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the Predictions of Any Classifier. In *KDD*, pages 1135–1144, 2016.

[145] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535, 2018.

[146] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[147] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10674–10685, 2022.

[148] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019.

[149] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1, 2019.

[150] Cynthia Rudin. Why black box machine learning should be avoided for high-stakes decisions, in brief. *Nature Reviews Methods Primers*, 2:81, 2022.

[151] Cynthia Rudin and Seyda Ertekin. Learning customized and optimized lists of rules with mathematical programming. *Mathematical Programming Computation*, 10:659–702, 2018.

[152] Lukas Ryll, Mary Emma Barton, Bryan Zheng Zhang, R Jesse McWaters, Emmanuel Schizas, Rui Hao, Keith Bear, Massimo Preziuso, Elizabeth Seger, Robert Wardrop, et al. Transforming paradigms: A global ai in financial services survey. https://ssrn.com/abstract=3532038, 2020.

[153] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296, 2017.

[154] Deepti Saraswat, Pronaya Bhattacharya, Ashwin Verma, Vivek Kumar Prasad, Sudeep Tanwar, Gulshan Sharma, Pitshou N Bokoro, and Ravi Sharma. Explainable ai for healthcare 5.0: opportunities and challenges. *IEEE Access*, 10: 84486–84517, 2022.

[155] Daniel Schiff, Justin Biddle, Jason Borenstein, and Kelly Laas. What's next for ai ethics, policy, and governance? a global overview. In *AIES*, pages 153–158, 2020.

[156] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61, 2015.

[157] Lesia Semenova, Cynthia Rudin, and Ronald Parr. On the existence of simpler machine learning models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1827–1858, 2022.

[158] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry. Toward verified artificial intelligence. *Communications of the ACM*, 65:46–55, 2022.

[159] LLoyd S. Shapley. A value of n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[160] Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018.

[161] Aditya A Shrotri, Nina Narodytska, Alexey Ignatiev, Kuldeep S Meel, Joao Marques-Silva, and Moshe Y Vardi. Constraint-driven explanations for black-box ml models. In *AAAI*, volume 36, pages 8304–8314, 2022.

[162] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[163] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 2018.

[164] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In *AIES*, pages 180–186, 2020.

[165] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11, 2010.

[166] Erik Strumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.*, 41(3), 2014.

[167] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[168] UNESCO. Draft recommendation on the ethics of artificial intelligence. https://unesdoc.unesco.org/ark:/48223/pf0000374266, 2021.

[169] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[170] Warren J Von Eschenbach. Transparency and the black box problem: Why we do not trust ai. *Philosophy & Technology*, 34(4):1607–1622, 2021.

[171] Stephan Wäldchen, Jan MacDonald, Sascha Hauch, and Gitta Kutyniok. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.*, 70:351–387, 2021.

[172] Sean Whalen, Jacob Schreiber, William S Noble, and Katherine S Pollard. Navigating the pitfalls of applying machine learning in genomics. *Nature Reviews Genetics*, 23:169–181, 2022.

[173] Jeannette M Wing. Trustworthy ai. *Communications of the ACM*, 64:64–71, 2021.

[174] Lior Wolf, Tomer Galanti, and Tamir Hazan. A formal approach to explainability. In *AIES*, pages 255–261, 2019.

[175] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. In *NLPCC*, pages 563–574, 2019.

[176] Guang Yang, Qinghao Ye, and Jun Xia. Unbox the black-box for the medical explainable ai via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. *Information Fusion*, 77:29–52, 2022.

[177] Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic. Computing optimal decision sets with SAT. In *CP*, pages 952–970, 2020.

[178] Jinqiang Yu, Graham Farr, Alexey Ignatiev, and Peter J. Stuckey. Anytime approximate formal feature attribution. *CoRR*, abs/2312.06973, 2023.

[179] Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. On formal feature attribution and its approximation. *CoRR*, abs/2307.03380, 2023.

[180] Jinqiang Yu, Alexey Ignatiev, Peter J Stuckey, Nina Narodytska, and Joao Marques-Silva. Eliminating the impossible, whatever remains must be true: On extracting and applying background knowledge in the context of formal explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4123–4131, 2023.

[181] Ronald Yu and Gabriele Spina Alì. What's inside the black box? ai challenges for lawyers and researchers. *Legal Information Management*, 19:2–13, 2019.

[182] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *CoRR*, abs/2307.08423, 2023.