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

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.

Journal of Artificial Intelligence Research 72 (2021) 1251-1279 Submitted 02/2021; published 12/2021

Learning Optimal Decision Sets and Lists with SAT

Jinqiang Yu JINQIANG.YU@MONASH.EDU
Alexey Ignatiev ALEXEY.IGNATIEV@MONASH.EDU
Peter J. Stuckey PETER.STUCKEY@MONASH.EDU
Pierre Le Bodic PIERRE.LEBODIC@MONASH.EDU
Department of Data Science and Artificial Intelligence
Monash University, Australia

Abstract

Decision sets and decision lists are two of the most easily explainable machine learning models. Given the renewed emphasis on explainable machine learning decisions, both of these machine learning models are becoming increasingly attractive, as they combine small size and clear explainability. In this paper, we define size as the total number of literals in the SAT encoding of these rule-based models as opposed to earlier work that concentrates on the number of rules. In this paper, we develop approaches to computing minimum-size "perfect" decision sets and decision lists, which are perfectly accurate on the training data, and minimal in size, making use of modern SAT solving technology. We also provide a new method for determining optimal sparse alternatives, which trade off size and accuracy. The experiments in this paper demonstrate that the optimal decision sets computed by the SAT-based approach are comparable with the best heuristic methods, but much more succinct, and thus, more explainable. We contrast the size and test accuracy of optimal decisions lists versus optimal decision sets, as well as other state-of-the-art methods for determining optimal decision lists. Finally, we examine the size of average explanations generated by decision sets and decision lists.

1. Introduction

With the increasing use of Machine Learning (ML) models to automate decisions[1], there has been an upsurge in interest in *explainable artificial intelligence* (XAI)[2] where these models can explain, in a manner understandable by humans, why they made a decision. This in turn has led to a re-examination of machine learning models that are implicitly easy to explain.

1. (Jordan & Mitchell, 2015; LeCun, Bengio, & Hinton, 2015; Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, et al., 2015)
2. (Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, & Müller, 2010; Ribeiro, Singh, & Guestrin, 2016; Doshi-Velez & Kim, 2017; Lundberg & Lee, 2017; Ribeiro, Singh, & Guestrin, 2018; Shih, Choi, & Darwiche, 2018; Li, Liu, Chen, & Rudin, 2018; Montavon, Samek, & Müller, 2018; Lipton, 2018; Monroe, 2018; Evans & Grefenstette, 2018; Ignatiev, Narodytska, & Marques-Silva, 2019b, 2019c; Guidotti, Monreale, Ruggieri, Turini, Giannotti, & Pedreschi, 2019b; Guidotti, Monreale, Giannotti, Pedreschi, Ruggieri, & Turini, 2019a; Darwiche, 2020; Darwiche & Hirth, 2020; Ignatiev, 2020; Marques-Silva, Gerspacher, Cooper, Ignatiev, & Narodytska, 2020)

Arguably the most explainable forms of ML models are decision trees[3], decision lists[4] and decision sets[5], since these encode simple logical rules. Of these, decision sets provide the simplest explanation, since if a rule "fires" for a given data instance, then this rule is the only explanation required. To explain decision lists, i.e. ordered sets of decision rules, we additionally need to take the order of rules into account.

In order to make explanations easy for humans to understand they should be as concise as possible.[6] There has been considerable investigation into producing the smallest possible optimal perfect decision trees (Narodytska et al., 2018; Verhaeghe et al., 2019), where the decision tree agrees perfectly with the training data, as well as producing the smallest possible sparse decision trees (Hu et al., 2019; Aglin et al., 2020), where there is a trade-off between size of the decision tree versus its accuracy on the training data.

Recent work has examined learning decision sets where the number of rules is minimized first and the number of literals is minimized afterwards (Ignatiev et al., 2018), as well as constructing a CNF classifier that minimizes the number of literals in the fixed number of rules (Malioutov & Meel, 2018; Ghosh & Meel, 2019) to explain the instances of the positive class. The latter two works also suffer from the limitation that only one class, class 1, is predicted by the rules, and class 0 is represented by the fact that no rule applies. Unfortunately, the explanation of negative instances is not succinct as explaining a class 0 example requires (the negation of) all rules.

We argue that previous work has not used the best measure of explainability, since, for instance, 2 rules each involving 80 conditions are significantly less explainable than 4 rules each involving only 10 conditions. Therefore, we investigate directly building decision sets that minimize the size redefined as the total number of literals used. This contributes to smaller decision sets (with respect to literals) which tend to be attractive to explain decisions. Note that the measure of size can also apply to decision lists.

There has been investigation of the method of optimizing decision lists (Angelino et al., 2017; Rudin & Ertekin, 2018; Angelino et al., 2018) that relies on a two-phase approach. First, decision rules are mined using some association rule mining techniques (Agrawal, Imieliński, & Swami, 1993), then an optimal order of the rules is found via search. In contrast, the method proposed in this paper directly generates all the rules of the optimal decision list as part of the search. This means it can generate decision rules which are meaningful in the context of their position in the target decision list, but could by themselves not provide valuable information on the training data, and therefore would not have been

3. (Hu, Rudin, & Seltzer, 2019; Aglin, Nijssen, & Schaus, 2020; Narodytska, Ignatiev, Pereira, & Marques-Silva, 2018; Verhaeghe, Nijssen, Pesant, Quimper, & Schaus, 2019; Apté & Weiss, 1997; Marques-Silva et al., 2017; Dufour, 2014; Bessiere, Hebrard, & O'Sullivan, 2009; Avellaneda, 2020; Janota & Morgado, 2020; Schidler & Szeider, 2021)
4. (Hancock, Jiang, Li, & Tromp, 1996; Rivest, 1987; Rudin & Ertekin, 2018; Angelino, Larus-Stone, Alabi, Seltzer, & Rudin, 2018; Angelino, Larus-Stone, Alabi, Seltzer, & Rudin, 2017; Wang, Rudin, Doshi-Velez, Liu, Klampfl, & MacNeille, 2017; Clark & Niblett, 1989; Yu, Ignatiev, Le Bodic, & Stuckey, 2020a)
5. (Kim, Khanna, & Koyejo, 2016; Doshi-Velez & Kim, 2017; Miller, 2019; Clark & Niblett, 1989; Clark & Boswell, 1991; Lakkaraju, Bach, & Leskovec, 2016; Ignatiev, Pereira, Narodytska, & Marques-Silva, 2018; Malioutov & Meel, 2018; Dash, Günlük, & Wei, 2018; Ghosh & Meel, 2019; Yu, Ignatiev, Stuckey, & Le Bodic, 2020b; Ignatiev, Lam, Stuckey, & Marques-Silva, 2021)
6. Interpretability is generally accepted to be a *subjective* concept, without a rigorous definition (Lipton, 2018). In this paper we measure interpretability in terms of the overall succinctness of the information provided by a model to explain a given prediction (see Section 6 for details).

mined by the approach of (Angelino et al., 2017; Rudin & Ertekin, 2018; Angelino et al., 2018). The two-step process can also generate many candidate rules to order when the number of features is large. Finally, this earlier approach does not optimize rules in terms of reducing the total number of literals, which may result in larger decision list models.

The previous methods focus on sparse decision lists, which trade size for accuracy on the training data. In contrast, we can also find optimal perfect decision lists, which agree completely with the training data.

Although the definition of size triggers harder computation in SAT models, the resulting machine learning models can be considerably smaller. However, for *sparse* decision sets and decision lists, the new measure is no more challenging for computation than the traditional rule count measure, and provides better granularity decisions on sparseness.

In summary, the contributions of this paper are:

- An approach to building optimal decision sets and decision lists in terms of the total number of literals required.
- The first method we are aware of to determine optimal decision lists, which agree with all the training data.
- The first SAT-based approach to find optimal sparse decision sets and decision lists that allow a trade-off between size and accuracy.
- We introduce the notion of *explanation size* which, for a particular example, determines how much information is required to explain the classification given by a machine learning model. We compare explanation size for decision lists and decision sets.
- Experiments demonstrating that our optimal sparse decision sets are as accurate as the best heuristic methods, but more concise.
- Experiments demonstrating that our approach to optimal sparse decision lists generates more accurate decision lists than previous methods

The paper is organized as follows. Section 2 presents the notation and definitions used throughout the paper. Section 3 outlines related work. The innovative SAT-based encodings for the inference of decision sets are described in Section 4. Section 5 illustrates encodings for computing decision lists. Section 6 introduces the notion of explanation size. The analysis of experimental results is discussed in Section 7. Finally, Section 8 concludes the paper.

2. Preliminaries

(Maximum) Satisfiability. The input of a Boolean satisfiability problem (SAT) (Biere, Heule, van Maaren, & Walsh, 2009) consists of a formula over a set of propositional variables using various logic operators on these variables. Solving a SAT problem consists in determining whether there exists an assignment of *true* or *false* value to each variable, called a *truth assignment*, such that the entire formula is *satisfied*. Otherwise, the formula is *unsatisfiable*. In the more specific *conjunctive normal form* (CNF), the formula is a conjunction

of clauses, and each clause is a disjunction of *literals*. A literal is a variable or its negation. Hence, a CNF formula can be satisfied if and only if at least one literal per clause can be set to *true*.

In the context of unsatisfiable formulas, the maximum satisfiability (MaxSAT) problem consists in finding a truth assignment that maximizes the number of satisfied clauses. In the Partial Weighted MaxSAT variant (Biere et al., 2009, Chapter 19), each clause c is either *soft* and has a weight w_c, or it is *hard*. An optimal solution then consists of a truth assignment that satisfies all hard clauses and maximizes the sum of the weights of the satisfied soft clauses.

Classification Problems. We consider a classification problem with a set of features $\mathcal{F} = \{f_1, \ldots, f_K\}$ and a label $\mathcal{C}$, all binary (or binarized using standard techniques (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, & Duchesnay, 2011)). The complete space of feature values (or *feature space* (Han, Kamber, & Pei, 2012)) is $\mathcal{U} \triangleq \prod_{r=1}^{K} \{f_r, \neg f_r\}$. The training set is denoted by $\mathcal{E} = \{e_1, \ldots, e_M\}$ and each instance $e_i \in \mathcal{E}$ is a pair (π_i, c_i) where $\pi_i \in \mathcal{U}$ and $c_i \in \mathcal{C}$. Furthermore, we assume without loss of generality in our context that dataset $\mathcal{E}$ *partially* defines a Boolean function $\phi : \mathcal{U} \to \mathcal{C}$, i.e. for any examples e_i and e_j in $\mathcal{E}$ associated with a same set of feature values, their classes are also the same.

The objective of classification in machine learning is to devise a function $\hat{\phi}$ that matches the actual function ϕ on the training data $\mathcal{E}$ and generalizes *suitably well* on unseen test data (Fürnkranz, Gamberger, & Lavrac, 2012; Han et al., 2012; Mitchell, 1997; Quinlan, 1993). In many settings (including *sparse*[7] decision sets and lists), function $\hat{\phi}$ is not required to match the actual function ϕ on the complete set of examples $\mathcal{E}$ and instead an *accuracy* measure is considered. Moreover, in classification problems one conventionally has to deal with an optimization problem, to optimize either with respect to the complexity of $\hat{\phi}$, or with respect to the accuracy of the learnt function (to make it match the actual function ϕ on a maximum number of examples), or both.

Rules, Decision Sets and Decision Lists. We can naturally represent a binary feature $f \in \mathcal{F}$ as a Boolean variable, and its two possible values as a literal or its negation, denoted f and $\neg f$. A *rule* has the form IF "instance satisfies formula" THEN "predict c", where the formula is a conjunction on a subset of the feature literals.

A *Decision Set* (DS) is an *unordered* set of rules. A decision set misclassifies an instance if no rule matches the instance, or if there exists a rule that predicts a wrong class.

A *Decision List* (DL) is an *ordered* set of rules. The first rule of a decision list that matches an instance is the one that classifies the instance. Decision lists are often written as a single cascade of IF-THEN-ELSEs, with the last rule often a "catch-all", or *default rule*, which matches all (remaining) instances to some class.

Example 1. Consider the following set of 8 items (shown as columns):

7. In contrast to *perfect* rule-based ML models, which aim at getting the highest possible training accuracy, *sparse* rule-based ML models trade off training accuracy for model size.

Item No.		1	2	3	4	5	6	7	8
Features	A	1	1	0	0	0	0	0	0
	B	0	1	1	1	0	0	0	0
	C	1	0	0	1	1	1	0	0
	D	0	1	1	0	0	1	1	1
	E	0	1	0	1	1	0	0	1
Class	H	1	1	0	0	1	1	0	0

A valid and optimal decision set for this data is

$$\begin{array}{llll} \mathsf{if} & A & \mathsf{then} & H \\ \mathsf{if} & \neg A \wedge \neg C & \mathsf{then} & \neg H \\ \mathsf{if} & \neg B \wedge C & \mathsf{then} & H \\ \mathsf{if} & \neg A \wedge B & \mathsf{then} & \neg H \end{array}$$

The *size* of this decision set is 11 (one for each literal on the left side and one for each rule, or alternatively, one for each literal on the left hand side and right hand side). Note how rules can overlap: both the first and third rule classify item 1.

A valid and optimal decision list for the data above is

$$\begin{array}{ll} & \mathsf{if}\ A\ \mathsf{then}\ H \\ \mathsf{else} & \mathsf{if}\ B\ \mathsf{then}\ \neg H \\ \mathsf{else} & \mathsf{if}\ C\ \mathsf{then}\ H \\ \mathsf{else} & \mathsf{if}\ \mathit{true}\ \mathsf{then}\ \neg H \end{array}$$

The size of the decision list is 7, and there is no overlap of rules by definition: item 1 is classified by the first rule only. Note that the last rule is a default rule.

Throughout the paper, each rule will be represented as a *path graph*, i.e. a graph reduced to a single path, where each literal is a *node* on the path. For example, the first two rules of the decision set would be represented as the two path graphs:

By construction, the *leaf* (i.e. last) node of each path corresponds to a class literal (see Example 2 for full example). Our SAT models will represent a Decision set (or list) as a concatenation of these path graphs into a single path graph. □

3. Related Work

The first work we are aware of for synthesizing a formula to cover a given partially defined (by $\mathcal{E}$ in our notation) Boolean function ϕ is by Michalski (1969).[8] Their general algorithm called AQ can be applied to the classification problem when $\mathcal{C} = \{c\}$ to construct a DNF formula by greedily choosing a minimal subset of features of each positive instance which does not cover any negative instance, until all positive instances are covered. The approach is also extended to multi-classification problems.

8. We thank the anonymous reviewers for pointing this out.

Decision sets and decision lists are rule-based predictive models that date back to the late 80s (Rivest, 1987; Clark & Niblett, 1989; Clark & Boswell, 1991). To the best of our knowledge, Kamath, Karmarkar, Ramakrishnan, and Resende (1992) proposed the first logic-based approach to constructing decision sets. Specifically, they introduced a SAT-based algorithm to synthesize a formula in disjunctive normal form (DNF) that matches a given set of training samples, which is tackled by an interior point method afterwards. Later, Lakkaraju et al. (2016) defined an approach to yielding a set of rules and minimizing a linear combination of criteria, e.g. the upper bound of the size in a rule, the number of rules, the number of overlapping rules, and misclassification.

The latest related approach to constructing decision sets is provided by Ignatiev et al. (2018). They develop a SAT model to iteratively compute minimal perfect decision sets where the training examples are perfectly classified and the number of rules is minimized. Then, the total number of literals required in a decision set is lexicographically minimized. However, as will be demonstrated later, the method computes larger decision sets than our approach that aims to minimize the total number of literals in a target decision set. Their method achieves better scalability to generate perfect decision sets as the more coarse-grained optimization measure is applied, i.e. the number of rules. The SAT method (Ignatiev et al., 2018) is also demonstrated to comprehensively outstrip the heuristic approach of Lakkaraju et al. (2016).

Malioutov and Meel (2018) provide a MaxSAT approach for the problem of binary classification, where they *fix the number of rules* and define the size in the model as the total number of literals across all clauses. Their proposed model minimizes a linear combination of Hamming loss and size, controlling the trade-off between explainability and accuracy instead of computing a perfect binary classifier. The scalability of this method is enhanced by Ghosh and Meel (2019), where rules are computed iteratively on partitions of the training data. However, only a single formula that classifies the positive items is computed rather than a *decision set* as defined in other work (Clark & Boswell, 1991; Lakkaraju et al., 2016; Ignatiev et al., 2018). Although the representation in the approach is smaller, explainability is reduced for negative examples where the whole formula is used to explain their classification.

Integer Programming (IP) has also been used to construct optimal rule-based models that only classify positive examples. Dash et al. (2018) introduce an IP approach for binary classification where an instance is classified as positive if and only if at least one rule of the model fires the example. The objective function of the model is to minimize a variation of Hamming loss, which is the number of misclassified positive instances, plus, for each misclassified negative example, the number of clauses misclassifying it. The bound on the size of each clause determines the complexity of this model. Column generation (Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance, 1998) is used in the IP model as it has one binary variable for each possible clause. However, since the model is only solved heuristically for large datasets, the pricing problem tends to be too expensive.

One recent approach (Rudin & Ertekin, 2018) provides decision lists that have some optimality guarantee. Given a fixed set of decision rules, it chooses a minimum-size ordered subset of these rules; the order essentially terminates when a default rule is chosen. The authors model the problems as an IP and solve it with a MIP (Mixed IP) solver. The objective is a combination of training accuracy and sparsity, minimizing misclassifications where every rule used incurs a "cost" of C misclassifications, and every literal used costs

C_1 misclassifications. The method is slow, and somewhat restricted by the time required to generate all potential possible rules as input. They consider datasets with up to 3000 examples and 60 features, but cannot prove optimality of their solutions on the data tested. One advantage of the approach is that it is easy to customize, for example, favoring the use of certain features, or extending to cost-sensitive learning.

To the best of our knowledge, the first method to generate *optimal decision lists* extends the approach of Rudin and Ertekin (2018) using the same idea of ordering a fixed set of decision rules, but using a bespoke branch-and-bound algorithm (Angelino et al., 2018). The method makes use of bounding methods and symmetry elimination techniques. They minimize regularized misclassification, where each rule costs ρM misclassification errors where M is the number of training examples. The approach relies on the sparsification parameter ρ to limit the set of rules it needs to consider. It can find and prove optimal solutions to large problems (hundreds of thousands of examples), the main limitation is on the number of features, since the number of possible decision rules grows exponentially in the number of features. The datasets they consider have up to 28 (binary) features, and at most 189 decision rules are considered.

4. Decision Set Encoding

This section introduces two SAT-based approaches to learning decision sets of minimum *total* size, referring to the total number of literals used in a model. We recall that the number of training data is M, whereas the number of features is K. For binary classification problems we consider that there is one class pseudo-feature $\mathcal{C} = \{c\}$ and examples have this feature set to *true* or *false*. For three or more classes, we assume a one-hot encoding (Pedregosa et al., 2011), i.e. each class value is represented by a single binary feature, with each example having exactly one such feature from $\mathcal{C}$ set to *true*. We will describe the approaches to computing perfect decision sets that perfectly classify the training examples and sparse models that can trade off accuracy for size.

4.1 Iterative SAT Model for Perfect Decision Sets

We first introduce a SAT-based model which determines whether there exists a perfect decision set of a given size N. In order to find a decision set of minimum size, the SAT model is iteratively solved while increasing N by 1 until it is satisfied. All rules are encoded as a single sequence of feature literals, and a class literal ends each rule. The model also keeps track of which examples are valid (agree) with previous literals in the rule, and checks whether the examples that reach the end of a rule (i.e. a leaf node) match the correct class. The Boolean variables used in the encoding are described below:

- s_{jr}: node j is a literal on feature $f_r \in \mathcal{F} \cup \mathcal{C}$;
- t_j: truth value of the literal for node j;
- v_{ij}: example $e_i \in \mathcal{E}$ is valid at node j;

The model is as follows:

- Only one feature (or a class feature) is used in a node:

$$\forall_{j\in[N]} \sum_{r\in[K+|\mathcal{C}|]} s_{jr} = 1 \tag{1}$$

- The last node is a leaf:

$$\bigvee_{c\in\mathcal{C}} s_{Nc} \tag{2}$$

- All examples are valid at the first node:

$$\forall_{i\in[M]}\ v_{i1} \tag{3}$$

- An example e_i is valid at node $j+1$ iff j is a leaf node, or e_i is valid at node j and e_i and node j agrees on the value of the feature s_{jr} selected for that node:

$$\forall_{i\in[M]}\forall_{j\in[N-1]}\ v_{ij+1} \leftrightarrow (\bigvee_{c\in\mathcal{C}} s_{jc}) \vee (v_{ij} \wedge \bigvee_{r\in[K]} (s_{jr} \wedge (t_j = \pi_i[r]))) \tag{4}$$

- If example e_i is valid at a leaf node j, it should agree on the class feature:

$$\begin{aligned} \forall_{i\in[M]}\forall_{j\in[N]}\ (s_{jc} \wedge v_{ij}) \rightarrow (t_j = c_i) \quad & \mathcal{C} = \{c\} \\ \forall_{i\in[M]}\forall_{j\in[N]}\forall_{c\in\mathcal{C}}\ (s_{jc} \wedge v_{ij}) \rightarrow c_i \quad & |\mathcal{C}| \geq 3 \end{aligned} \tag{5}$$

- When there are 3 or more classes we restrict leaf nodes to only consider *true* examples of the class:

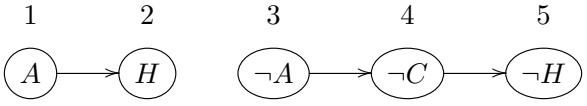
$$\forall_{j\in[N]}\forall_{c\in\mathcal{C}}\ s_{jc} \rightarrow t_j \qquad |\mathcal{C}| \geq 3 \tag{6}$$

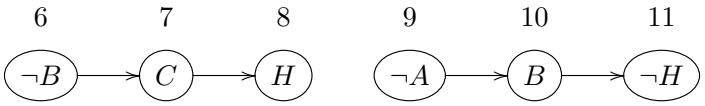
- For every example there should be at least one leaf node where it is valid:

$$\forall_{i\in[M]} \bigvee_{j\in[N]} ((\bigvee_{c\in\mathcal{C}} s_{jc}) \wedge v_{ij}) \tag{7}$$

The model described above represents a non-clausal Boolean formula, which can be clausified with the utilization of auxiliary variables (Tseitin, 1968). Also note that any of the known cardinality encodings can be used to denote the sum in constraint (1) (Biere et al., 2009, Chapter 2) (also see (Asín, Nieuwenhuis, Oliveras, & Rodríguez-Carbonell, 2009; Bailleux & Boufkhad, 2003; Batcher, 1968; Sinz, 2005)). Finally, the size (in terms of the number of literals) of the suggested SAT encoding is $\mathcal{O}(N \times M \times K)$, which results from constraint (4).

Example 2. Consider a solution for 11 nodes for the dataset of Example 1. The sequence of nodes that represent the rules are shown below:





The interesting (true) decisions for each node are given in the following table

Node j	1	2	3	4	5	6	7	8	9	10	11
s_{jr}	s_{1A}	s_{2H}	s_{3A}	s_{4C}	s_{5H}	s_{6B}	s_{7C}	s_{8H}	s_{9A}	s_{10B}	s_{11H}
t_j	1	1	0	0	0	0	1	1	0	1	0
v_{ij}	v_{11}	v_{12}	v_{13}	v_{24}	v_{75}	v_{16}	v_{17}	v_{18}	v_{19}	v_{210}	v_{311}
	$\vdots$	v_{22}	$\vdots$	$\vdots$	v_{85}	$\vdots$	v_{57}	v_{58}	$\vdots$	$\vdots$	v_{411}
	v_{81}		v_{83}	v_{84}		v_{86}	$\vdots$	v_{68}	v_{89}	v_{810}	
							v_{87}				v_{411}

The selected variable at the end of each rule is class H. Note that all examples are valid at the start node in each rule, and each feature literal leads to the reduction of the valid set for the next node. In each leaf node j, the valid instances of the correct class are determined by the truth value t_j of that node. □

The iterative SAT model tries to compute a perfect decision set of size N. If this fails, we increase the predefined size by 1 and resolve, until a solution is found or resource limits (typically computation time) are reached. A potential question is why binary search is not used instead. The issue is that the complexity of deciding satisfiability of the described model grows (potentially) exponentially with N, and therefore, predefining a large N can lead to the failure of solving the whole problem.

Example 3. Consider the data shown in Example 1. The iterative SAT model initially tries to find a decision set of size 1, which fails, then of size 2, etc. until the model reaches size 11 where it can determine the perfect decision set: $A \Rightarrow H$, $\neg A \wedge \neg C \Rightarrow \neg H$, $\neg B \wedge C \Rightarrow H$, $\neg A \wedge B \Rightarrow \neg H$ of size 11 by finding the model shown in Example 2. □

4.2 MaxSAT Model for Perfect Decision Sets

Instead of using the proposed iterative SAT-based model, which iterates over varying size N of the perfect decision set, we can modify the model to determine a perfect decision set of size at most N, and formulate a MaxSAT problem such that the number of nodes used can be minimized. Let us add a flag variable u_j for every available node. Variable u_j is *true* whenever the node j is unused and *false* otherwise. The model is as follows:

- A node either uses a feature or is unused:

$$\forall_{j\in[N]}\ u_j + \sum_{r\in[K+|\mathcal{C}|]} s_{jr} = 1 \tag{8}$$

- If a node j is unused then so are all the following nodes

$$\forall_{j\in[N-1]}\ u_j \rightarrow u_{j+1} \tag{9}$$

- The last used node is a leaf:

$$\forall_{j\in[N-1]}\ u_{j+1} \rightarrow u_j \vee \bigvee_{c\in\mathcal{C}} s_{jc} \tag{10}$$

$$u_N \vee \bigvee_{c\in\mathcal{C}} s_{Nc} \tag{11}$$

The constraints above together with constraints (3), (4), (5), (6), and (7) make up the hard clauses of the MaxSAT formula, i.e. every clause of them must be satisfied. The model maximizes $\sum_{j\in[N]} u_j$, which is the representation of the list of unit soft clauses of the form $(u_j, 1)$.

In the worst case, the model is still run iteratively. We first propose an upper bound N on the size of the decision set. The model tries to search for a decision set of size no more than N. If this fails, we need to increase N by some number (say 10) and retry, until the solution is found or resource limits are reached.

Example 4. Revisiting the solution illustrated in Example 1 when N is set to 13 we find the solution shown in Example 2 extended in which the last two nodes are unused: $u_{12} = u_{13} = \mathit{true}$. The last used node 11 is clearly a leaf. Note that truth value (t_j) and validity (v_{ij}) variables are irrelevant to unused nodes j. □

4.3 MaxSAT Model for Sparse Decision Sets

We can extend the MaxSAT model to look for *sparse* decision sets, which trade off training accuracy for model size, rather than *perfect* decision sets, which aim at the highest possible training accuracy. The objective to minimize is the total number of misclassifications (including non-classifications where no prediction information is provided for the example by any decision rule) plus the product of the number of nodes in a decision set and a discount factor Λ, which records that Λ fewer misclassifications are worth the addition of one node to the decision set. Typically we consider $\Lambda = \lceil \lambda M \rceil$ where λ is the regularized penalty of nodes in terms of misclassifications.

We introduce variable m_i to represent that example $i \in [M]$ is misclassified. Consider the following constraints:

- If example e_i is valid at a leaf node j then e_i agrees on the class feature or e_i is misclassified:

$$\begin{aligned} &\forall_{i\in[M]}\forall_{j\in[N]}\ (s_{jc} \wedge v_{ij}) \rightarrow (t_j = c_i \vee m_i) \quad && \mathcal{C} = \{c\} \\ &\forall_{i\in[M]}\forall_{j\in[N]}\forall_{c\in\mathcal{C}}\ (s_{jc} \wedge v_{ij}) \rightarrow (c_i \vee m_i) \quad && |\mathcal{C}| \geq 3 \end{aligned} \tag{12}$$

- For every example there should be at least one leaf literal where it is valid or the example is misclassified (actually *non-classified*):

$$\forall_{i\in[M]}\ m_i \vee \bigvee_{j\in[N]}\ (\bigvee_{c\in\mathcal{C}} s_{jc} \wedge v_{ij}) \tag{13}$$

together with all the MaxSAT constraints from Section 4.2 except constraints (5) and (7). The objective function of the proposed model is

$$\sum_{i\in[M]} m_i + \sum_{j\in[N]} \Lambda(1-u_j) + N\Lambda$$

represented as soft clauses $(\neg m_i, 1)$, $i \in [M]$, and (u_j, Λ), $j \in [N]$.

Note that the value of regularized penalty λ is critical. As the value of λ becomes higher, the emphasis of the problem shifts more to "sparsification" of the target decision set, rather than its accuracy. In other words, higher values of λ (and hence of Λ as well) contribute to simpler decision sets where its accuracy on training examples is sacrificed.

4.4 Separated Models for Decision Sets

A convenient feature of optimal decision sets is the following: the union of minimal decision sets for each $c \in \mathcal{C}$ that correctly classifies all examples of class c and does not misclassify any examples not of class c as class c, is a minimal target decision set for the whole problem.

That means we can compute perfect decision sets for $|\mathcal{C}|$ classes by separately computing $|\mathcal{C}|$ perfect decision sets, one for each class. The union of these $|\mathcal{C}|$ models, which we call *"separated model"*, is clearly not much smaller than the complete model, as each separated model still includes constraint (4) for each example leading to the size $\mathcal{O}(N \times M \times K)$. The advantage arises because the minimal size required for each separated model is smaller.

Example 5. Consider the dataset shown in Example 1. We can iteratively compute decision rules for the positive examples: $A \Rightarrow H$, $\neg B \wedge C \Rightarrow H$ of size 5, and the negative examples: $\neg A \wedge \neg C \Rightarrow \neg H$, $\neg A \wedge B \Rightarrow \neg H$ of size 6. This is faster than solving the problems together where the complete model iterates from size 1 to size 11 to ultimately find the same solution. □

For sparse decision sets, the definition of *misclassifications* needs to be modified in order that a separated solution is available. Suppose that an example $e_i \in \mathcal{E}$ is of class $c_i \in \mathcal{C}$ then the *number of misclassifications* of example e_i is counted as follows:

- If example e_i is not classified as class c_i, then this counts as one misclassification.
- If example e_i is classified as class $c_j \in \mathcal{C}$, $c_j \neq c_i$ that counts as one misclassification per class.

With the modified definition of misclassification, we can construct minimal decision sets per class separately and then join them together.

5. Decision List Encoding

We can modify the SAT encoding for decision sets described in Section 4 fairly naturally to instead define decision lists.

5.1 MaxSAT Model for Perfect Decision Lists

The MaxSAT model introduced in Section 4.2 determines whether there exists a perfect decision set of at most a given size N. We can modify this to determine a perfect *decision*

list of size at most N by keeping track of which items have previously been classified by a previous rule, and preventing them from being considered (in)valid in later rules. The sequence of literals is viewed as a path graph, with one feature literal per node.

We introduce a new variable n_{ij} to represent that example $e_i \in \mathcal{E}$ is not previously classified by any nodes before j.

The model is as follows:

- All examples are not previously classified at the first node:

$$\forall_{i\in[M]}\ n_{i1} \tag{14}$$

- An example e_i is previously unclassified at node $j+1$ iff it was previously unclassified, and either j is not a leaf node or it was invalid at the previous leaf node (so not classified by the rule that finished there):

$$\forall_{i\in[M]}\forall_{j\in[N-1]}\ n_{ij+1} \leftrightarrow n_{ij} \wedge ((\bigwedge_{c\in\mathcal{C}} \neg s_{jc}) \vee \neg v_{ij}) \tag{15}$$

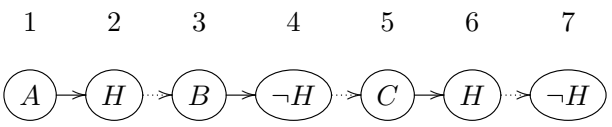
- An example e_i is valid at node $j+1$ iff j is a leaf node and it was previously unclassified, or e_i is valid at node j and e_i and node j agree on the value of the feature s_{jr} selected for that node:

$$\forall_{i\in[M]}\forall_{j\in[N-1]} \quad \begin{array}{l} v_{ij+1} \leftrightarrow (((\bigvee_{c\in\mathcal{C}} s_{jc}) \wedge n_{ij+1}) \vee \\ (v_{ij} \wedge \bigvee_{r\in[K]} (s_{jr} \wedge (t_j = \pi_i[r])))) \end{array} \tag{16}$$

The constraints above together with constraints (3), and (5)–(10) make up the hard constraints of the MaxSAT model. The model maximizes $\sum_{j\in[N]} u_j$, which can be trivially represented as a list of unit soft clauses of the form $(u_j, 1)$. The size (in terms of the number of literals) of the proposed SAT encoding is $\mathcal{O}(N \times M \times K)$, which results from constraints (15) and (16).

The differences between the above model and the MaxSAT model of decision sets is the addition of the n_{ij} variables to track which items have been previously classified, and their use in constraint (16), as well as the rules to compute them given in constraints (14) and (15).

Example 6. Consider a solution for 7 nodes for the data of Example 1. The representation of the decision list is shown below:



The interesting (true) decisions for each node are as follows:

Node j	1	2	3	4	5	6	7
s_{jr}	s_{1A}	s_{2H}	s_{3B}	s_{4H}	s_{5V}	s_{6H}	s_{7H}
t_j	1	1	1	0	1	1	0
v_{ij}	v_{11}	v_{12}	v_{33}	v_{34}	v_{55}	v_{56}	v_{77}
	$\vdots$	v_{22}	$\vdots$	v_{44}	$\vdots$	v_{66}	v_{87}
	v_{81}		v_{83}		v_{85}		
n_{ij}	n_{11}	n_{12}	n_{33}	n_{34}	n_{55}	n_{56}	n_{77}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	n_{87}
	n_{81}	n_{82}	n_{83}	n_{84}	n_{85}	n_{86}	

The selected variable at the end of each rule is class H. Note that at the start and after each leaf node all previously unclassified examples are valid, and each feature literal triggers the decrease of the valid set for the next node. The valid instances of the correct class are determined by the truth value t_j in each leaf node j. □

The MaxSAT model tries to find a decision list of size at most N. If this fails, we can increase N by some amount and resolve, until either resource limits (typically computation time) are reached or a solution is found.

5.2 MaxSAT Model for Sparse Decision Lists

We modify the MaxSAT model for sparse decision sets introduced in Section 4.3 to look for sparse decision lists that trade off accuracy for size. The same objective function and discount factor Λ are used in the model where Λ fewer misclassifications are worth the addition of one node to the decision list.

The hard part of the model consists of the constraints (12) and (13) with all the hard constraints of the model for perfect decision lists except constraints (5) and (7). The objective function is

$$\sum_{i\in[M]} m_i + \sum_{j\in[N]} \Lambda(1-u_j) + N\Lambda$$

represented as soft clauses $(\neg m_i, 1)$, $i \in [M]$, and (u_j, Λ), $j \in [N]$.

5.3 Separated Models for Decision Lists

Separated models for decision sets have been introduced in Section 4.4 where the size of the union of $|\mathcal{C}|$ models is the same as the size of the complete model.

For decision lists this property no longer holds. If we compile decision lists separately for each class, we must still order the decision lists of different classes. Therefore, no optimal decision list might be expressed as rules for one class, followed by another class.

Example 7. Consider the dataset of Example 1. Recall that an optimal decision list shown in Example 1 has 7 literals.

An optimal decision list that is separated in class order is

$$\begin{array}{ll} & \mathsf{if}\ A\ \mathsf{then}\ H \\ \mathsf{else} & \mathsf{if}\ \neg B \wedge C\ \mathsf{then}\ H \\ \mathsf{else} & \mathsf{if}\ B\ \mathsf{then}\ \neg H \\ \mathsf{else} & \mathsf{if}\ true\ \mathsf{then}\ \neg H \end{array}$$

requiring one more literal.

Given that separated models are important for scaling this approach to larger problems, we need to consider approaches for defining decision lists in a separated form. We consider a number of different approaches:

fixed σ Given a permutation σ of classes, find an optimal decision list for the first class in σ, then make an optimal decision list for the second class ignoring items already classified by the decision list for the first class. Then consider the third class, etc.

greedy Make an optimal decision list for each class independently: choose the one that is best under some metric. Fix its solution as the first part of the decision list. Calculate I' as the items not classified by this decision list. Make an optimal decision list for each remaining class independently. Again, choose the best one and fix it. Continue until all classes are considered, or I' becomes empty.

For the fixed permutation case, one can try all possible permutations, if there are not too many, e.g. $|\mathcal{C}| \leq 3$, or use a heuristic to choose a permutation σ. One heuristic we consider is sorting the classes by increasing/decreasing number of their respective items in the training set. Alternatively, we consider ordering the classes greedily based on the post-hoc analysis of the accuracy or cost of individual class representations obtained on the training data. Here, training accuracy for the representation of class c is $1 - \frac{\sum_{i\in[M],c_i=c} m_i}{|\{e_i\in\mathcal{E},c_i=c\}|}$ while the cost of representation of class c is assumed to be $N - \sum_{j\in[N]} u_j + \left\lceil \frac{\sum_{i\in[M]} m_i}{\Lambda} \right\rceil$.

Note that for separated sparse models, the objective is effectively different. Using the same objective for each class separately means that we count a misclassification once for every class it is detected by. This is arguably more informative. As we cannot guarantee the same optimal solutions anyway (due to order restrictions), this seems acceptable.

6. Explanation Size

Given two different ML models, we can ask *which model gives the smallest explanation* on a particular data instance. By optimizing the size of a decision list or decision set, we believe the size of the explanations it creates will be small, but this is not completely accurate. The explanation size of an ML model can be far smaller than the whole model. The implicit notion of *explanation size* we are trying to capture is, if a customer/user were to ask why our model made a decision for their case, how would we explain that decision? Note that we also define explanation size for the cases where a decision set makes no decision, either since no rule fires, or two contradictory rules fire. We define the explanation size of a model $\hat{\phi}$ on an example instance e as follows (note that alternative definitions of explanation size may exist, both for decision sets and decision lists).

If $\hat{\phi}$ is a decision set and the rules in $\hat{\phi}$ that fire on example e are $\{\textbf{if } \pi_i \textbf{ then } c_i\}$, $\forall i \in [n]$, $n \leq N = |\hat{\phi}|$, then

- if all the classes $c_1, \ldots, c_n$ agree, i.e. $c_i = c'$, $\forall i \in [n]$, $c' \in \mathcal{C}$, then the explanation size for example e is $\frac{\sum_{i=1}^{n} |\textbf{if } \pi_i \textbf{ then } c_i|}{n}$, that is, the average of the rules, any of which could explain the example.
- if not all classes agree for e then the explanation size is the sum of averages of the rules for all the conflicting classes predicted for e; wlog. assume that $c_i = c'$, $c' \in \mathcal{C}$, $1 \leq i \leq k < n$ and $c_j = c''$, $c'' \in \mathcal{C}$, $k+1 \leq j \leq n$, then the explanation size for example e is $\frac{\sum_{i=1}^{k} |\textbf{if } \pi_i \textbf{ then } c_i|}{k} + \frac{\sum_{j=k+1}^{n} |\textbf{if } \pi_j \textbf{ then } c_j|}{n-k}$; similar reasoning can be applied to situations of more than two conflicting classes.
- if no rule fires then the explanation size is $|\hat{\phi}|$, i.e. we need the whole decision set to explain why e is not classified.

If $\hat{\phi}$ is a decision list and $\textbf{if } \pi_j \textbf{ then } c_j$ is the first rule in $\hat{\phi}$ that fires on example e then

- the explanation size is $\sum_{i=1}^{j} |\pi_i| + 1$ as we need to explain why none of the previous rules fired, and why rule j did.
- if no rule fires for e then the explanation size is $|\hat{\phi}|$, i.e. we need the whole model to explain why e is not classified. Note that in practice this does not occur since the last rule will be a default rule, and all examples will be classified.

Note that it is easy to extend the notion of explanation size to decision trees (as the path from root to leaf) though decision tree models are not considered in this paper.

7. Experimental Results

This section describes the results of experimental assessment of the proposed approach to computing optimal decision sets and decision lists. Firstly, we discuss the comparison between the decision set approaches and state-of-the-art decision sets in terms of performance, accuracy and model size. Afterwards, we compare the proposed approach to decision lists with the SAT-based decision sets as well as the only previous approach to optimal sparse decision lists we are aware of (Angelino et al., 2017; Wang et al., 2017).

Experimental results are obtained on the StarExec cluster[9] (Stump, Sutcliffe, & Tinelli, 2014), each computing node of which uses an Intel Xeon E5-2609 2.40GHz CPU with 128GByte of RAM, running CentOS 7.7. The time limit and memory limit used per process are 1800 seconds and 16 GB.

For the evaluation, we use the 71 datasets from the UCI Machine Learning Repository (Dua & Graff, 2017) and Penn Machine learning Benchmarks (Olson, La Cava, Orzechowski, Urbanowicz, & Moore, 2017). We also use 5-fold cross validation, which results in 355 pairs of training and test data split with the ratio 80% and 20%, respectively. Finally, feature domains are quantized into 2, 3, and 4 intervals and then *one-hot encoded* (Pedregosa et al., 2011). The number of one-hot encoded features (training instances, resp.)

9. `https://www.starexec.org/`

per dataset in the benchmark suite varies from 3 to 384 (from 14 to 67557, resp.). The total number of benchmark datasets is 1065 ($71 \times 5 \times 3$).

All the proposed models are implemented as collections of Python scripts[10] and solving is done by instrumenting incremental calls to SAT solver Glucose 3 (Audemard, Lagniez, & Simon, 2013) and MaxSAT solver RC2-B (Ignatiev, Morgado, & Marques-Silva, 2018, 2019a).

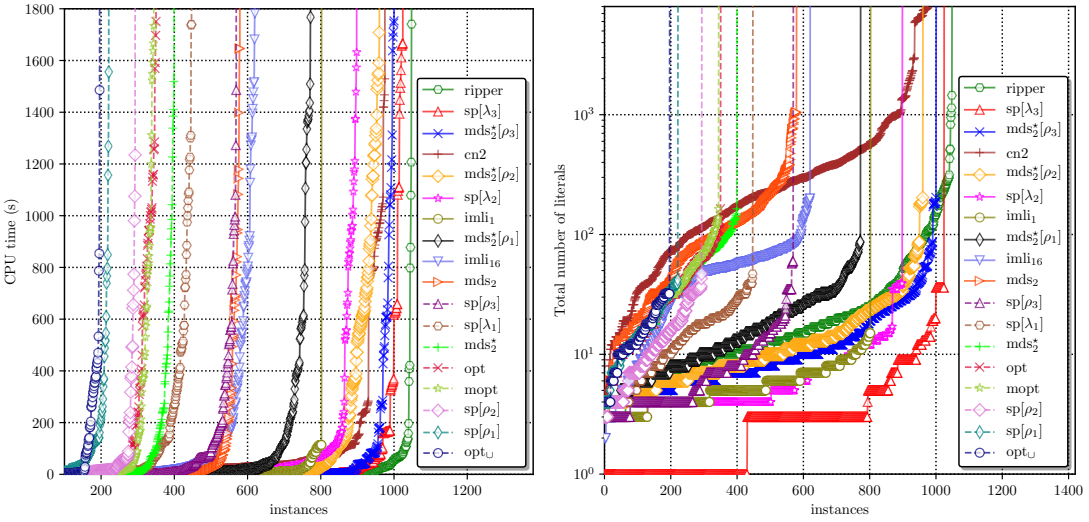
7.1 Experimental Results for Decision Sets

Implementation. All models in the implementation target independent computation of each class[11], as described in Section 4.4. The iterative SAT and MaxSAT models targeting perfect decision sets are referred to as *opt* and *mopt*. To explain the benefits of separated models over the models aggregating all classes, a complete SAT model is also included, which is referred to as $opt_{\cup}$. Lastly, some variants of the MaxSAT model for sparse decision sets named $sp[\lambda_i]$ are assessed with three different values of regularized penalty: $\lambda_1 = 0.005$, $\lambda_2 = 0.05$, and $\lambda_3 = 0.5$.

Competitors. MinDS$_2$ and MinDS$_2^\star$ (Ignatiev et al., 2018) for perfect decision sets referred to as mds_2 and $mds_2^\star$ are considered as the competitors in the implementation. The former algorithm minimized the number of rules, whereas the latter does lexicographic optimization, i.e. it minimizes the number of rules first and the total number of literals afterwards. In addition, we modified MinDS to be able to compute *sparse* decision sets similar to what is described in Section 4.3. The implementation is called $mds_2[\rho_i]$, tested with three values of regularized penalty $\rho_1 = 0.05$, $\rho_2 = 0.1$, and $\rho_3 = 0.5$. Note that $\rho_i \neq \lambda_i$, $i \in [3]$, as the two models use different measures where MinDS targets rules but the other one targets literals. To consider fair comparison of performance between the new model *sp* and of the sparse variant of $mds_2^\star$, we provide the configuration of $sp[\rho_i]$ with $\rho_i = \frac{\lambda_i}{K}$, where K refers to the number of features in the dataset. In other words, a rule is considered equivalent to K literals.

Apart from MinDS, some state-of-the-art algorithms were also considered, including a MaxSAT-based method IMLI (Ghosh & Meel, 2019) (i.e. the direct successor of MLIC (Malioutov & Meel, 2018)) as well as two heuristic approaches CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991), and RIPPER (Cohen, 1995). Note that IMLI and RIPPER compute only one class given the training data[12]. Both of these competitors incorporate a *default rule* which classifies a class (1) different from the constructed one and (2) represented by the majority of the training examples. Finally, IMLI aims to compute a target decision set given the number of rules k, which varies from 1 to 16. The best results (both regarding accuracy and performance) are demonstrated by the configuration aiming at the smallest possible number of rules, i.e. $k = 1$, while the worst results were shown for $k = 16$. Therefore, only the extreme values of k are used to compare the results, represented by $imli_1$ and $imli_{16}$.

10. `https://github.com/alexeyignatiev/minds/`
`https://github.com/jinqiang-yu/dlsat/`
11. The prototype adapts all the developed models to the case of multiple classes, which is motivated by the practical importance of non-binary classification.
12. The implementation of RIPPER considered in our experimental evaluation is taken from `https://github.com/imoscovitz/wittgenstein`. A reader may find other implementations having different capabilities.



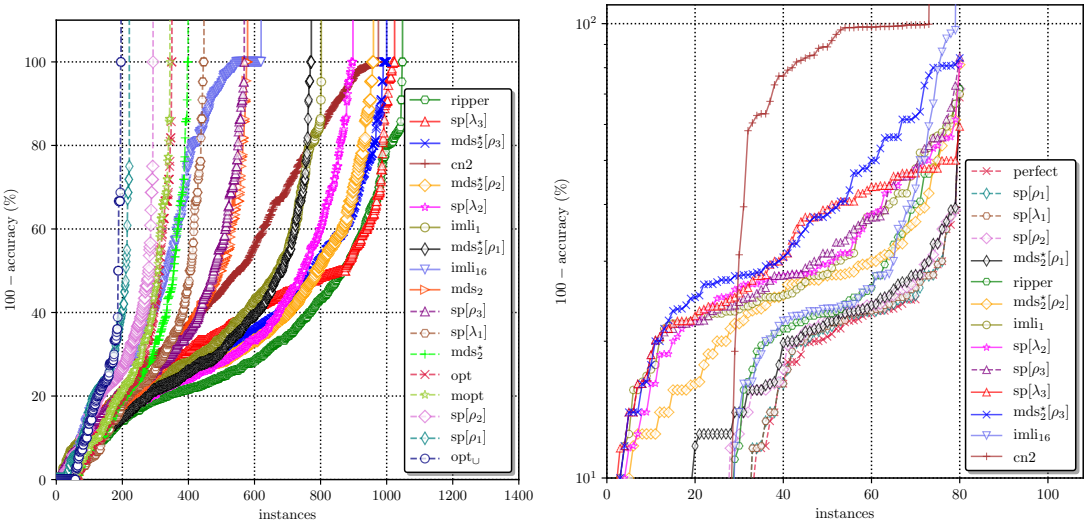
(a) Performance (b) Decision set size

Figure 1: Performance of all decision set competitors and the quality of solutions in terms of decision set size.

Performance. The performance of these models is shown in Figure 1a using a cactus plot. The plot shows for each method how many instances were solved by the method within each CPU time. As can be observed, the best results are demonstrated by *ripper*, which is able to compute 1048 selected datasets within the 1800s time limit. The second and third positions are occupied by the proposed MaxSAT models for sparse decision sets $sp[\lambda_3]$ and $mds^{\star}_2[\rho_3]$, which can successfully cope with 1024 and 1000 datasets respectively. *cn2* takes the fourth place with 975 datasets solved. $imli_1$ is the best configuration of *imli*, finishing with 802 instances solved, whereas the worst configuration $imli_{16}$ computes only 620 datasets. Finally, the worst performance is shown by the approaches targeting *perfectly accurate* decision sets *opt*, *mopt*, and $opt_{\cup}$ as well as the MaxSAT approaches for sparse decision sets with low regularized penalty $sp[\rho_1]$ and $sp[\rho_2]$. For example, $opt_{\cup}$ solves only 196 datasets.

Test Accuracy. The calculation of accuracy is as follows: (1) if a rule of a wrong class *"covers"* an instance, the instance is considered misclassified (though there are other rules of the correct class covering this instance); (2) if an instance is not covered by any rule of a correct class, the instance is considered misclassified. Afterwards, the accuracy is calculated as $\frac{M-E}{M} \times 100\%$, where M is the total number of instances and E is the total number of misclassified instances.

Confirmed by Figure 2b which depicts the accuracy among all the approaches for the datasets solved by *all* these approaches, perfect decision sets tend to have highest accuracy once they are computed. As can be observed, the virtual *perfect* approach, which represents all the tools aiming at perfect decision sets, i.e. *opt*, *mopt*, $opt_{\cup}$, mds_2, and $mds^{\star}_2$, outperforms the other approaches regarding testing accuracy. The average testing accuracy



(a) On the complete set of datasets

(b) Instances solved by all tools

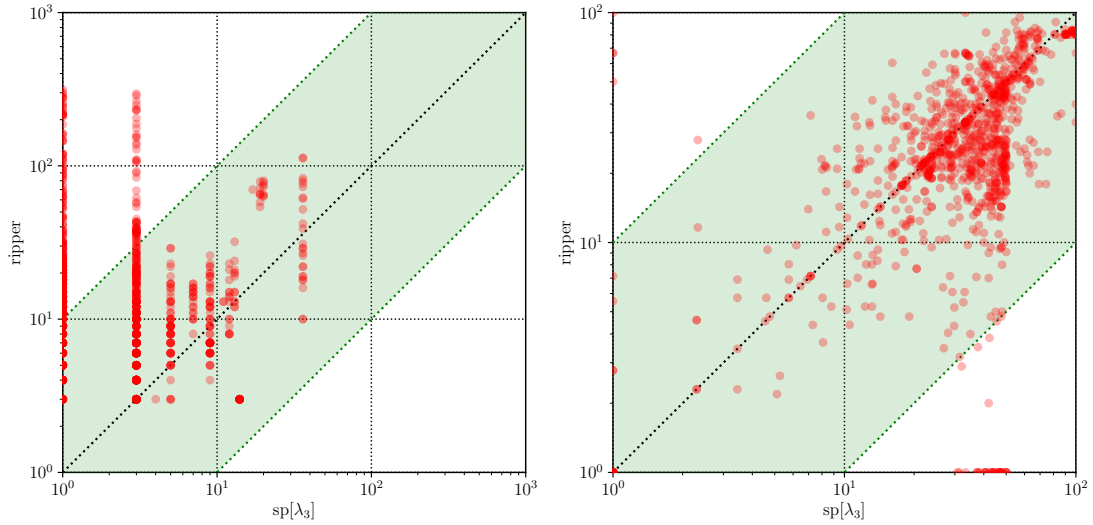
Figure 2: Accuracy of the considered decision set approaches.

on these solved datasets is 85.89%[13]. Conversely, *cn2* (43.73%) is the worst in terms of testing accuracy. Also, the average accuracy of $mds^\star_2[\rho_3]$, $sp[\lambda_3]$, $imli_1$, $imli_{16}$, and *ripper* is 61.71%, 67.42%, 76.06%, 77.42%, and 80.50% respectively.

The result changes significantly if we compare testing accuracy on the entire set of benchmark datasets, demonstrated in Figure 2a. Here, we *assume* that the accuracy for a dataset is 0% if a tool fails to train a model for the dataset within the given time limit. The figure shows that *ripper* (68.13% on average) achieves the best accuracy, followed by the sparse decision sets computed by $mds^\star_2[\rho_3]$ and $sp[\lambda_3]$ (61.23% and 60.91%, respectively). The average accuracy of $imli_1$ and $imli_{16}$ is 50.66% and 28.26%, whereas the average accuracy achieved by *cn2* is 47.49%.

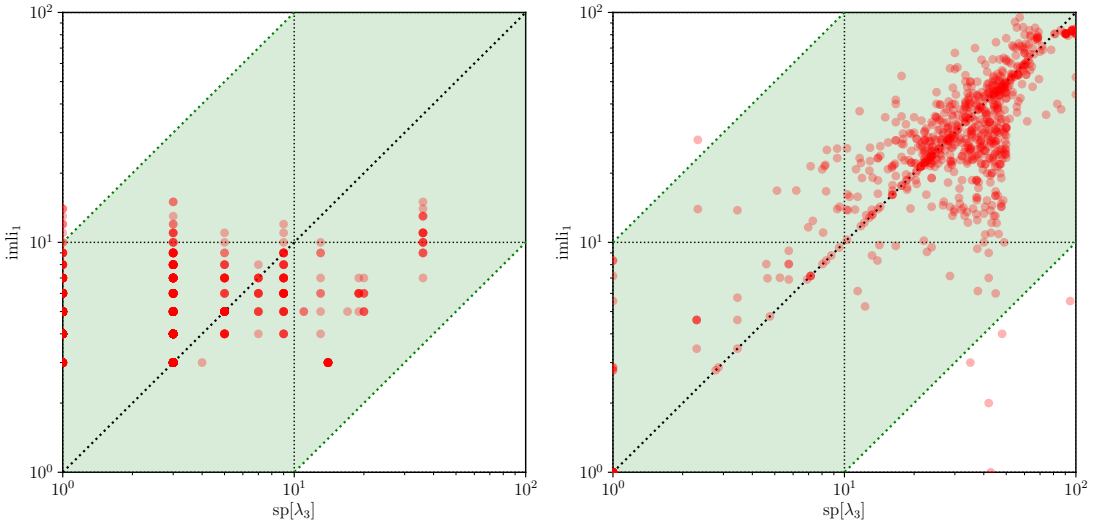
Decision Set Size Figure 1b illustrates the size in terms of the number of literals in a decision set model. The figure demonstrates that the winner is $sp[\lambda_3]$. The decision sets of $sp[\lambda_3]$ make up only one literal[14] for more than 400 datasets. For another bunch of around 400 instances, the solutions of $sp[\lambda_3]$ consist of 3 literals. Computing small solutions is a noticeable achievement as $sp[\lambda_3]$ also achieves overall high accuracy. The average size of decision sets computed by $sp[\lambda_3]$ is 4.18. Note that $imli_1$ occupies the second place with 5.57 literals per dataset though its solution always consists of only one rule. In comparison to this, the average solution size of *cn2* is 598.05. Finally, the result of $imli_{16}$ is 46.29, while the average solution of *ripper* has 35.14 literals.

13. This average value is the highest possible accuracy that can be achieved on these datasets whatever machine learning model is considered.
14. In a unit-size decision set, the literal is meant to assign a constant class. This can be seen as applying a *default rule*.



(a) Decision set size (b) Test accuracy

Figure 3: Comparison of $sp[\lambda_3]$ and *ripper*.

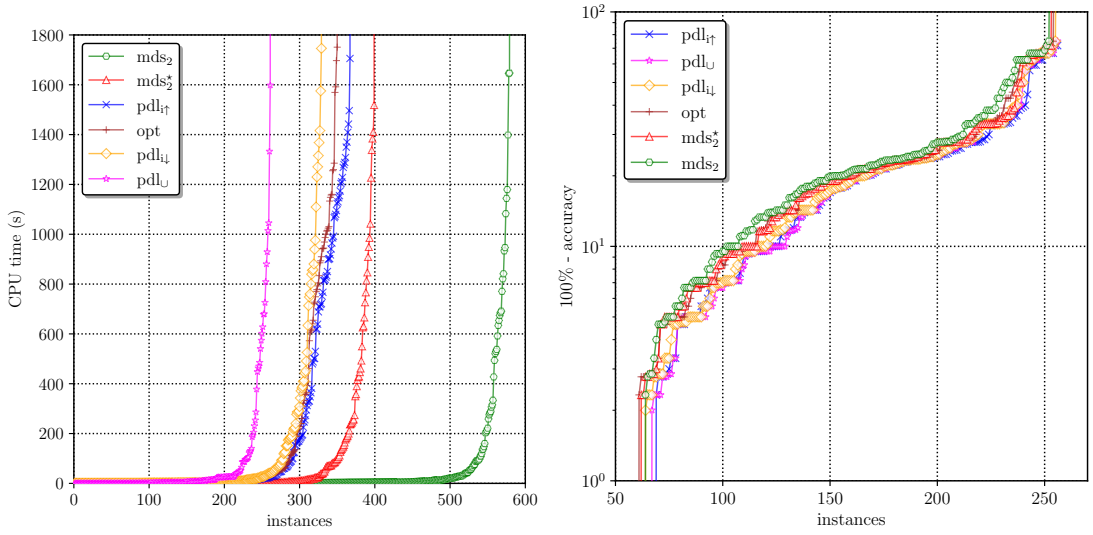


(a) Decision set size (b) Test accuracy

Figure 4: Comparison of $sp[\lambda_3]$ and $imli_1$.

As expected, perfect decision sets, i.e. those constructed by *opt*, *mopt*, $opt_{\cup}$, as well as mds_2, and $mds_2^{\star}$, tend to be larger in general. It should also be noted that the $mds_2^{\star}$ obtains perfect decision sets of size 40.70, whereas the sparse version gets 14.52 literals per solution.

A few more details. The comparison of $sp[\lambda_3]$ with *ripper* and $imli_1$ is depicted in Figure 3 and Figure 4 using scatter plots that plot the compared statistic of each method on the two axes. Note that the axes are log scaled since the differences in the methods



(a) Performance (b) Test accuracy

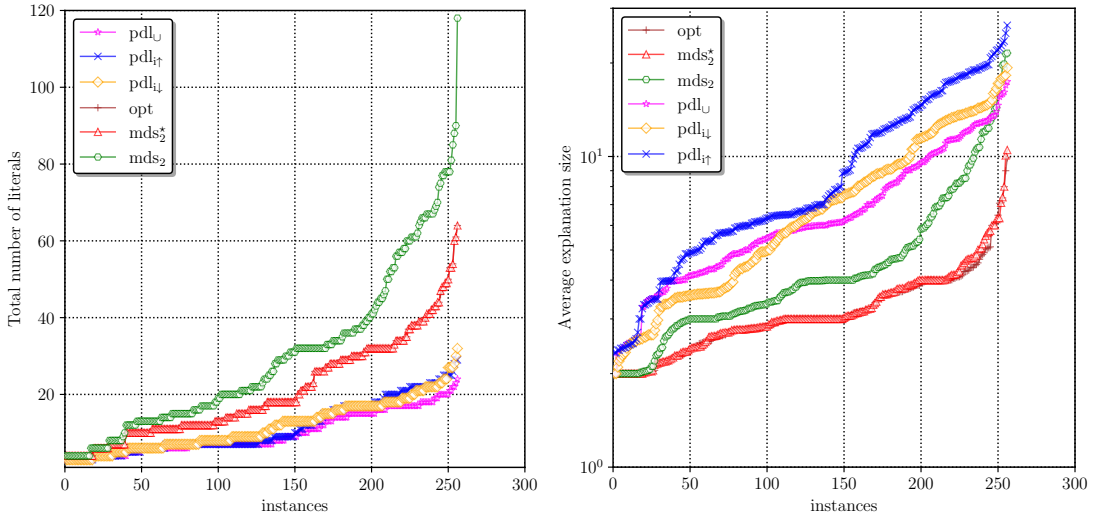
Figure 5: Performance and accuracy of perfect decision list and decision set models.

can be significant. All the results in these figures are obtained for the datasets solvable by each pair of contestants. As can be observed in Figure 4a and Figure 4b, $imli_1$ and $sp[\lambda_3]$ are comparable with respect to size and accuracy. Nevertheless, as $imli_1$ constructs decision sets representing only one class and it considerably underperforms $sp[\lambda_3]$, the latter method is considered a better alternative. Even if the accuracy of *ripper* is comparable with the accuracy of $sp[\lambda_3]$ (see Figure 3b) and the former approach achieves the best overall performance, the size of decision sets computed by *ripper* tends to be significantly larger than the one of its opponent (see Figure 3a).

Finally, a drawback of both RIPPER and IMLI is that they construct a representation for one class only. Therefore, a concise explanation for the instances of non-computed classes cannot be provided by the two approaches. This is in clear contrast with our approaches, which offer users a concise representation of every class in the dataset.

7.2 Experimental Results for Decision Lists

Implementation The complete MaxSAT model is referred to as $pdl_{\cup}$. As was shown in Section 5.3, separated models do not guarantee optimality of the size of decision lists, and so we tested various ordering of the classes when computing separated decision lists. Concretely, $pdl_{i\uparrow}$ and $pdl_{i\downarrow}$ refer to the separated models that order classes by the increasing/decreasing number of training data in the classes. Sparse models are referred to as $sdl[\lambda]_{\circ}$, where λ is a regularized penalty and ordering $\circ$ is from $\{\cup, i\downarrow, i\uparrow, a\downarrow, a\uparrow, c\downarrow, c\uparrow\}$ meaning that decision list computation is all classes at once, or each class computed separately with the classes being ordered based on the increasing/decreasing *number/accuracy/cost* of training data in the classes, as defined in Section 5.3.



(a) Decision set/list size

(b) Average explanation size

Figure 6: Decision set/list size and average explanation size of perfect decision list and decision set models.

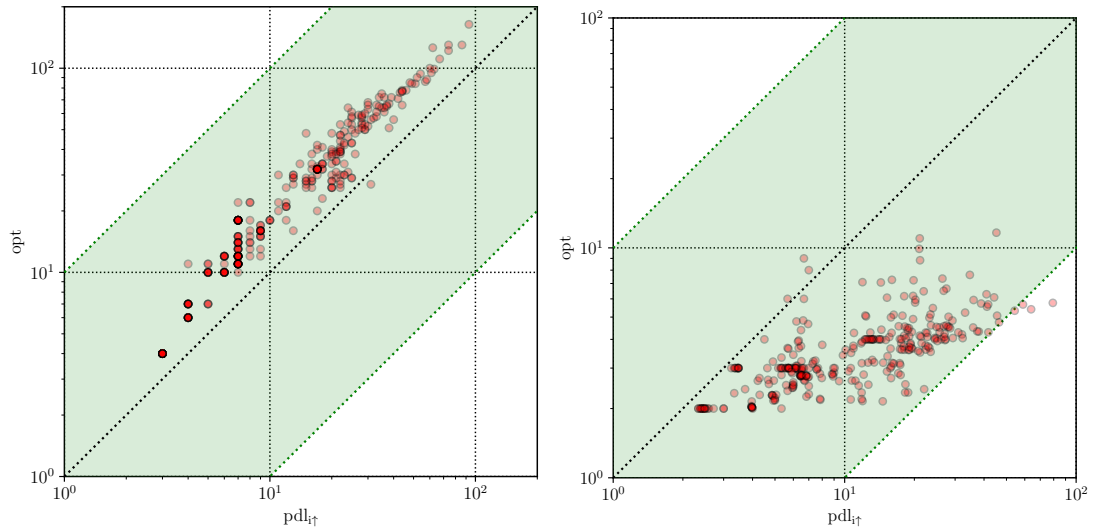
7.2.1 Perfect Models

We compare our prototype against state-of-the-art perfect decision set methods (Ignatiev et al., 2018) and the perfect decision set model described in Section 4.2, namely mds_2, $mds_2^\star$ and *opt*. While mds_2 generates a decision set with the smallest number of rules and *opt* minimizes the number of literals, $mds_2^\star$ does rule minimization followed by literal minimization. The comparison of perfect models is illustrated in Figure 5, Figure 6, and Figure 7.

Performance. The performance of the perfect models is shown in Figure 5a. As can be seen, mds_2 outperforms all the other rivals and trains 579 models. This should not come as a surprise since mds_2 minimizes the number of rules. It is followed by $mds_2^\star$, which sequentially applies rule and literal minimization – $mds_2^\star$ can solve 399 benchmarks. The best performing decision list model $pdl_{i\uparrow}$ comes third with 369 datasets handled successfully. The optimal decision set approach *opt* solves 350 instances. Finally, $pdl_{i\downarrow}$ and $pdl_\cup$ can train 329 and 261 decision lists respectively.

Test Accuracy. Test accuracy computed for the benchmarks solved by all the competitors is shown in the cactus plot of Figure 5b. Concretely, the plot depicts the value of *test error* in percent. On average, all the approaches perform similarly here and have test accuracy $\approx 80\%$. This is not surprising as all of them target perfectly accurate models.

Decision Set/List Size. The size calculated as the total number of literals in the decision set/list model is shown in Figure 6a. (Note that the plot is generated for the datasets successfully handled by all the shown competitors.) Observe that optimal perfect decision lists $pdl_\cup$ are the smallest among all the approaches with the average size being 9.9 per



(a) Decision set/list size detailed: $\text{pdl}_{i\uparrow}$ vs. opt (b) Explanation size detailed: $\text{pdl}_{i\uparrow}$ vs. opt

Figure 7: Comparison of $\text{pdl}_{i\uparrow}$ and opt in terms of decision set/list size and average explanation size.

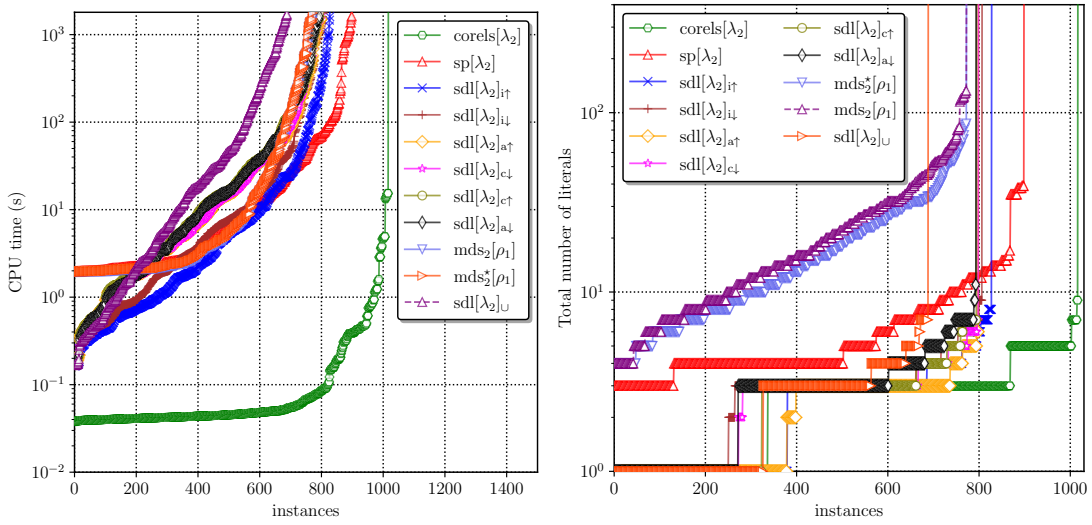
model. The second best model is $pdl_{i\uparrow}$ with 11.3 literals per model. Note that the smallest size decision sets obtained by *opt* have 20.6 literals on average. The largest models are of mds_2 with 29.6 literals per model on average. The pairwise comparison of decision set/list size for *opt* and $pdl_{i\uparrow}$ is detailed in the scatter plot of Figure 7a, which clearly demonstrates that perfect smallest size decision sets are usually larger than decision lists even when these are not guaranteed to be smallest in size.

Average Explanation Size. Although decision lists are smaller, the advantage of perfect decision sets is clearly the average explanation size per instance, which is calculated as described in Section 6. This data is shown in Figure 6b. For instance, it takes 3.3 literals on average to explain a prediction of decision sets produced by *opt*. For $mds_2^\star$ and mds_2 the numbers are 3.3 and 4.9, respectively. Explanations for decision lists are larger; the best result is shown by $pdl_\cup$, which has 6.9 literals per explanation. The best performing decision list model $pdl_{i\uparrow}$ has 9.6 literals per explanation. The detailed comparison of the average explanation size for *opt* and $pdl_{i\uparrow}$ is shown in the scatter plot of Figure 7b.

7.2.2 Sparse Models

The second part of our decision list evaluation compares sparse models. Here, the proposed approach is compared against sparse versions of decision sets *sp* and mds_2 of Section 4.2 and optimal sparse decision lists produced by *corels* (Angelino et al., 2017; Wang et al., 2017).[15] Although we tested 3 values for regularized penalty $\lambda \in \{0.005, 0.05, 0.5\}$, we report the results only for $\lambda_2 = 0.05$. As Section 7.1 showed, the best trade-off for sparse

15. There is a implementation of the RIPPER algorithm referred to as *JRip* that produces decision lists. However, we were unable to run it in our experimental environment. Since RIPPER is known to be more



(a) Performance (b) Decision set/list size

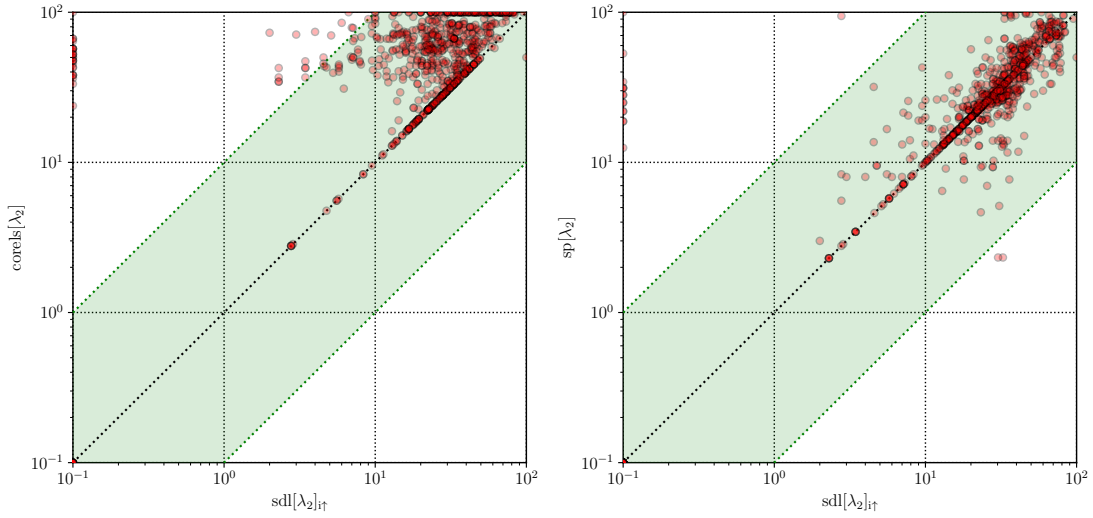
Figure 8: Performance and decision set/list size of sparse models.

decision sets was obtained for $\lambda_2 = 0.05$ and $\lambda_3 = 0.5$. However, decision lists obtained for λ_3 are usually too sparse as they end up having a single rule predicting a constant class. Therefore, hereinafter, the results are reported for configurations $sdl[\lambda_2]_*$ as well as for $sp[\lambda_2]$, $mds_2[\rho_1]$, $mds_2^\star[\rho_1]$, and $corels[\lambda_2]$. (Note that the value of regularized penalty $\rho_1 = 0.05$ is unchanged, also taken from Section 7.1. As mds_2 and $mds_2^\star$ minimize the number of rules, regularized penalty ρ_1 is applied wrt. the number of rules, which contrasts λ_2 applied to the number of literals). The results are shown in Figure 8, Figure 9, and Figure 10.

Performance. As can be observed in Figure 8a, $corels_{\lambda_2}$ is the fastest among the approaches for sparse models. It solves 1016 benchmarks. Sparse decision sets can be trained by $sp[\lambda_2]$ for 898 datasets while decision lists can be trained by $sdl[\lambda_2]_{i\uparrow}$ for 827 of them. Observe that class ordering $i\uparrow$ based on the increasing number of instances per class outperforms the other configurations of $sdl[\lambda_2]$, which can tackle ≈ 800 datasets each. The decision set competitors $mds_2[\lambda_2]$ and $mds_2^\star[\lambda_2]$ solve 772 benchmarks. Finally, aggregated computation of smallest decision lists of $sdl[\lambda_2]_\cup$ handles 688 datasets.

Test Accuracy. Although $corels[\lambda_2]$ outperforms its rivals in performance, the accuracy of its decision lists is not the best. The scatter plot in Figure 9a depicts the value of test error $e = 100\% - a$, where a is test accuracy, for $corels[\lambda_2]$ and $sdl[\lambda_2]_{i\uparrow}$. Observe that in many cases the accuracy of $sdl[\lambda_2]_{i\uparrow}$ is significantly higher than of $corels[\lambda_2]$: the average accuracy of $corels[\lambda_2]$ is 40.2% while the average accuracy of $sdl[\lambda_2]_{i\uparrow}$ is 69.9%. This clearly suggests that the sparsity measure used in our work enables us to train more

accurate (but less concise) than *corels* (Angelino et al., 2017; Wang et al., 2017), it not clear what the comparison of our approach against RIPPER would look like.



(a) Error: $\text{sdl}[\lambda_2]_{i\uparrow}$ vs. $\text{corels}[\lambda_2]$ (b) Error: $\text{sdl}[\lambda_2]_{i\uparrow}$ vs. $\text{sp}[\lambda_2]$

Figure 9: Test error comparison of $\text{sdl}[\lambda_2]_{i\uparrow}$, $\text{corels}[\lambda_2]$, and $\text{sp}[\lambda_2]$.

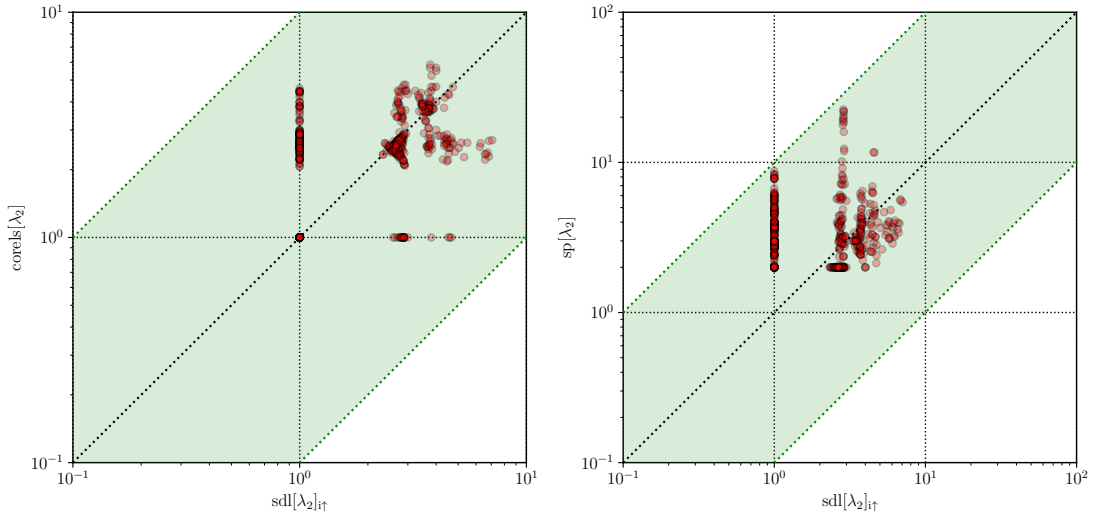
accurate decision lists. Also, as shown in Figure 9b, the accuracy of $sdl[\lambda_2]_{i\uparrow}$ is on a par with the accuracy of sparse decision sets of $sp[\lambda_2]$, which on average equals 67.6%.

Decision Set/List Size. As detailed in Figure 8b, the smallest models are obtained with sparse decision lists of $corels[\lambda_2]$ and $sdl[\lambda_2]_*$. The average number of literals in the lists produced by $corels[\lambda_2]$, $sdl[\lambda_2]_\cup$, and $sdl[\lambda_2]_{i\uparrow}$ is 2.7, 2.3, and 2.4, respectively (these numbers are calculated across the instances *solved* by the corresponding tools). Similar results are demonstrated by the other configurations of $sdl[\lambda_2]_*$. In contrast, the average size of sparse decision sets of $sp[\lambda_2]$, $mds_2[\lambda_2]$, and $mds_2^\star[\lambda_2]$ is 6.9, 22.5, and 17.9, respectively.

Average Explanation Size. Figure 10a and Figure 10b provide a comparison of $sdl[\lambda_2]_{i\uparrow}$ against $corels[\lambda_2]$ and $sp[\lambda_2]$ in terms of the average explanation size. In contrast to the case of perfect models, an average explanation for decision lists of $sdl[\lambda_2]_{i\uparrow}$ has 2.1 literals while explanations of sparse decision sets of $sp[\lambda_2]$ are of size 3.4. This suggests that sparse decision lists not only are smaller than sparse decision sets but they also provide a user with explanations that are more succinct. The average explanation size of the decision lists of $corels[\lambda_2]$ is 2.3. (The average numbers shown here are collected across all benchmarks solved by the corresponding tools).

8. Conclusion

In this paper, we developed the first approach to computing decision sets and decision lists where the total number of literals is minimized. The method can construct perfect decision sets and decision lists, i.e. those that perfectly classify the training instances; or sparse decision sets and lists, i.e. those that trade off model accuracy on training instances for size. The experimental results demonstrate that sparse models can outperform perfectly accurate models due to their (1) high accuracy overall and (2) better scalability. Second, the



(a) Explanations: sdl[λ_2]$_{i\uparrow}$ vs. corels[λ_2]

(b) Explanations: sdl[λ_2]$_{i\uparrow}$ vs. sp[λ_2]

Figure 10: Comparison of sdl[λ_2]$_{i\uparrow}$, corels[λ_2], and sp[λ_2] in terms of average explanation size.

regularized penalty substantially influences the efficiency of sparse decision sets and lists as these models are harder to compute but more accurate when the regularized penalty is smaller. This provides a reasonable trade-off in practice. Points 1 and 2 hold for the proposed models in this paper and also for the *sparse variants* of prior work aiming to minimize the number of rules (Ignatiev et al., 2018).

While existing heuristic approaches like RIPPER might scale significantly well and compute accurate decision sets, their solutions are often much larger than the sparse decision sets proposed in this paper, which causes them to be less explainable. The proposed method for sparse decision sets improves upon the previous state-of-the-art algorithms represented by prior logic-based approaches (Ignatiev et al., 2018; Malioutov & Meel, 2018; Ghosh & Meel, 2019) as well as by efficient heuristic methods (Cohen, 1995; Clark & Niblett, 1989; Clark & Boswell, 1991).

Although existing bespoke methods for optimal sparse decision lists are considerably more scalable, interestingly the accuracy of the models they construct are lower, probably because the size measure we use is more fine grained. There is also a question of how these methods scale with number of features. Finally, we provide the first comparison of decision sets and lists in terms of model size and explanation size. For perfect models, decision sets are preferable, but surprisingly this reverses for sparse models.

There is an interesting direction to extend this work. There is considerable symmetry in the proposed models, and while we tried adding symmetry breaking constraints to improve the models, what we tried did not make a significant difference. This deserves further exploration.

References

Aglin, G., Nijssen, S., & Schaus, P. (2020). Learning optimal decision trees using caching branch-and-bound search. In *AAAI*, pp. 3146–3153.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD*, p. 207–216.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., & Rudin, C. (2017). Learning certifiably optimal rule lists. In *KDD*, pp. 35–44.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., & Rudin, C. (2018). Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, *18*(234), 1–78.

Apté, C., & Weiss, S. (1997). Data mining with decision trees and decision rules. *Future generation computer systems*, *13*(2-3), 197–210.

Asín, R., Nieuwenhuis, R., Oliveras, A., & Rodríguez-Carbonell, E. (2009). Cardinality networks and their applications. In *SAT*, pp. 167–180.

Audemard, G., Lagniez, J., & Simon, L. (2013). Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *SAT*, pp. 309–317.

Avellaneda, F. (2020). Efficient inference of optimal decision trees. In *AAAI*, pp. 3195–3202.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. (2010). How to explain individual classification decisions. *Journal of Machine Learning Research*, *11*, 1803–1831.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF encoding of Boolean cardinality constraints. In *CP*, pp. 108–122.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, *46*(3), 316–329.

Batcher, K. E. (1968). Sorting networks and their applications. In *AFIPS*, Vol. 32, pp. 307–314.

Bessiere, C., Hebrard, E., & O'Sullivan, B. (2009). Minimising decision tree size as combinatorial optimisation. In *CP*, pp. 173–187.

Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.). (2009). *Handbook of Satisfiability*. IOS Press.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: some recent improvements. In *EWSL*, pp. 151–163.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–283.

Cohen, W. W. (1995). Fast effective rule induction. In *ICML*, pp. 115–123.

Darwiche, A. (2020). Three modern roles for logic in AI. In *PODS*, pp. 229–243. ACM.

Darwiche, A., & Hirth, A. (2020). On the reasons behind decisions. In *ECAI*, pp. 712–720.

Dash, S., Günlük, O., & Wei, D. (2018). Boolean decision rules via column generation. In *NeurIPS*, p. 4660–4670.

Doshi-Velez, F., & Kim, B. (2017). A roadmap for a rigorous science of interpretability. *CoRR*, *abs/1702.08608*.

Dua, D., & Graff, C. (2017). UCI machine learning repository..

Dufour, D. (2014). Finding cost-efficient decision trees. Master's thesis, University of Waterloo.

Evans, R., & Grefenstette, E. (2018). Learning explanatory rules from noisy data. *J. Artif. Intell. Res.*, *61*, 1–64.

Fürnkranz, J., Gamberger, D., & Lavrac, N. (2012). *Foundations of Rule Learning*. Springer.

Ghosh, B., & Meel, K. S. (2019). IMLI: an incremental framework for maxsat-based learning of interpretable classification rules. In *AIES*, pp. 203–210. ACM.

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., & Turini, F. (2019a). Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, *34*(6), 14–23.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2019b). A survey of methods for explaining black box models. *ACM Comput. Surv.*, *51*(5), 93:1–93:42.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann.

Hancock, T. R., Jiang, T., Li, M., & Tromp, J. (1996). Lower bounds on learning decision lists and trees. *Inf. Comput.*, *126*(2), 114–122.

Hu, X., Rudin, C., & Seltzer, M. (2019). Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*, pp. 7265–7273.

Ignatiev, A. (2020). Towards trustable explainable AI. In *IJCAI*, pp. 5154–5158.

Ignatiev, A., Lam, E., Stuckey, P. J., & Marques-Silva, J. (2021). A scalable two stage approach to computing optimal decision sets. In *34th AAAI Conference on Artificial Intelligence (AAAI 2021)*, p. To Appear.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2018). PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pp. 428–437.

Ignatiev, A., Morgado, A., & Marques-Silva, J. (2019a). RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, *11*(1), 53–64.

Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019b). Abduction-based explanations for machine learning models. In *AAAI*, pp. 1511–1519.

Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019c). On relating explanations and adversarial examples. In *NeurIPS*, pp. 15857–15867.

Ignatiev, A., Pereira, F., Narodytska, N., & Marques-Silva, J. (2018). A SAT-based approach to learn explainable decision sets. In *IJCAR*, pp. 627–645.

Janota, M., & Morgado, A. (2020). SAT-based encodings for optimal decision trees with explicit paths. In *SAT*, pp. 501–518.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kamath, A. P., Karmarkar, N., Ramakrishnan, K. G., & Resende, M. G. C. (1992). A continuous approach to inductive inference. *Math. Program.*, *57*, 215–238.

Kim, B., Khanna, R., & Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in neural information processing systems*, pp. 2280–2288.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pp. 1675–1684.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

Li, O., Liu, H., Chen, C., & Rudin, C. (2018). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*.

Lipton, Z. C. (2018). The mythos of model interpretability. *Commun. ACM*, *61*(10), 36–43.

Lundberg, S. M., & Lee, S. (2017). A unified approach to interpreting model predictions. In *NIPS*, pp. 4765–4774.

Malioutov, D., & Meel, K. S. (2018). MLIC: A maxsat-based framework for learning interpretable classification rules. In *CP*, pp. 312–327.

Marques-Silva, J., Gerspacher, T., Cooper, M. C., Ignatiev, A., & Narodytska, N. (2020). Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*.

Marques-Silva et al., J. (2017). Learning decision trees with SAT. Tech. rep., LASIGE, FCUL.

Michalski, R. S. (1969). On the quasi-minimal solution of the general covering problem. In *International Symposium on Information Processing*, pp. 125–128.

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

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Monroe, D. (2018). AI, explain yourself. *Commun. ACM*, *61*(11), 11–13.

Montavon, G., Samek, W., & Müller, K. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, *73*, 1–15.

Narodytska, N., Ignatiev, A., Pereira, F., & Marques-Silva, J. (2018). Learning optimal decision trees with SAT. In *IJCAI*, pp. 1362–1368.

Olson, R. S., La Cava, W., Orzechowski, P., Urbanowicz, R. J., & Moore, J. H. (2017). Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, *10*(1), 36.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kauffmann.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., & Rastogi, R. (Eds.), *KDD*, pp. 1135–1144. ACM.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI*.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*(3), 229–246.

Rudin, C., & Ertekin, S. (2018). Learning customized and optimized lists of rules with mathematical programming. *Mathematical Programming Computation*, *10*, 659–702.

Schidler, A., & Szeider, S. (2021). SAT-based decision tree learning for large data sets. In *AAAI*, pp. 3904–3912.

Shih, A., Choi, A., & Darwiche, A. (2018). A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pp. 5103–5111.

Sinz, C. (2005). Towards an optimal CNF encoding of Boolean cardinality constraints. In *CP*, pp. 827–831.

Stump, A., Sutcliffe, G., & Tinelli, C. (2014). Starexec: A cross-community infrastructure for logic solving. In *IJCAR*, pp. 367–373.

Tseitin, G. S. (1968). On the complexity of derivations in the propositional calculus. *Studies in Mathematics and Mathematical Logic*, *Part II*, 115–125.

Verhaeghe, H., Nijssen, S., Pesant, G., Quimper, C.-G., & Schaus, P. (2019). Learning optimal decision trees using constraint programming. In *Proceedings of CP-19*.

Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., & MacNeille, P. (2017). A bayesian framework for learning rule sets for interpretable classification. *Journal of Machine Learning Research*, *18*, 70:1–70:37.

Yu, J., Ignatiev, A., Le Bodic, P., & Stuckey, P. J. (2020a). Optimal decision lists using SAT. *CoRR*, *abs/2010.09919*.

Yu, J., Ignatiev, A., Stuckey, P. J., & Le Bodic, P. (2020b). Computing optimal decision sets with SAT. In *CP*, pp. 952–970.

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.

From Formal Boosted Tree Explanations to Interpretable Rule Sets

Jinqiang Yu
Department of Data Science and AI, Monash University, Australia
ARC Training Centre in OPTIMA, Australia

Alexey Ignatiev
Department of Data Science and AI, Monash University, Australia

Peter J. Stuckey
Department of Data Science and AI, Monash University, Australia
ARC Training Centre in OPTIMA, Australia

Abstract

The rapid rise of Artificial Intelligence (AI) and Machine Learning (ML) has invoked the need for *explainable AI* (XAI). One of the most prominent approaches to XAI is to train rule-based ML models, e.g. decision trees, lists and sets, that are deemed interpretable due to their transparent nature. Recent years have witnessed a large body of work in the area of constraints- and reasoning-based approaches to the inference of interpretable models, in particular decision sets (DSes). Despite being shown to outperform heuristic approaches in terms of accuracy, most of them suffer from scalability issues and often fail to handle large training data, in which case no solution is offered. Motivated by this limitation and the success of gradient boosted trees, we propose a novel anytime approach to producing DSes that are both accurate and interpretable. The approach makes use of the concept of a generalized formal explanation and builds on the recent advances in formal explainability of gradient boosted trees. Experimental results obtained on a wide range of datasets, demonstrate that our approach produces DSes that more accurate than those of the state-of-the-art algorithms and comparable with them in terms of explanation size.

2012 ACM Subject Classification Computing methodologies → Machine learning

Keywords and phrases Decision set; interpretable model; gradient boosted tree; BT compilation

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material *Software (Source Code)*: `https://github.com/jinqiang-yu/cpl/`

Acknowledgements This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009.

1 Introduction

Rapid development of Artificial Intelligence (AI) and Machine Learning (ML) have revolutionized all aspects of human lives in recent years [30, 1]. However, decisions made by most widely used ML models are hard for humans to understand hence the interest in the theory and practice of *Explainable AI* (XAI) rises.

One major approach to XAI is to compute *post-hoc* explanations for ML predictions to answer a *"why"* question [34, 44], i.e. why the prediction is made. Although heuristic approaches to post-hoc explanations prevail [34, 44, 43], they suffer from a number of weaknesses [21, 16, 49, 52]. Formal methods [48, 20, 37] provide alternative approaches to explanations that avoid these weaknesses. Another alternative approach to XAI is to compute *interpretable* ML models, i.e. logic-based models, including decision trees [40], decision lists [46], and decision sets [29]. These models enable decision makers to obtain

succinct explanations from the models directly. In this paper, we focus on the decision set (DS) models.

Decisions sets are particularly easy to explain: the rule that fired is an explanation of the decision. This led to an upsurge in interest of decision sets that are both interpretable and accurate. Recent work [50] uses propositional satisfiability (SAT) to generate minimum-size decision sets that are perfectly accurate on the training data, and demonstrates that decision sets that completely agree with the training data outperform others in terms of accuracy. A more scalable maximum satisfiability (MaxSAT) approach [18] to this problem was then proposed. Unfortunately, both of these methods are unable to provide any decision information if a dataset is not completely solved.

Motivated by these works and their limitations, this paper aims at making a bridge between formal post-hoc explainability and interpretable DS models. In particular, the paper focuses on developing a novel anytime approach to computing decision sets that are both interpretable and accurate, by compiling a gradient boosted tree model into a decision set on demand with the use of formal explanations. This is done with the use of the recent approach [17] to compute abductive explanations for gradient boosted trees using maximum satisfiability (MaxSAT). Furthermore, the paper proposes a range of post-hoc model reduction heuristics aiming at enhancing interpretability of the result models, done with MaxSAT and integer linear programming (ILP). The experimental results show that compared with other state-of-the-art methods, decision sets generated by the proposed approach are more accurate, and comparable with the competition in terms of interpretability.

2 Preliminaries

SAT and MaxSAT. The standard definitions for propositional satisfiability (SAT) and maximum satisfiability (MaxSAT) solving are assumed [3]. A propositional formula ϕ is said to be in *conjunctive normal form* (CNF) if it is a conjunction of clauses. A *clause* is a disjunction of literals, where a *literal* is either a Boolean variable b or its negation $\neg b$. A *truth* assignment μ is a mapping from the set of variables to $\{0, 1\}$. A clause is said to be *satisfied* by truth assignment μ if one of the literals in the clause is assigned value 1; otherwise, the clause is *falsified*. If all clauses in formula ϕ are satisfied by assignment μ, ϕ is satisfied; otherwise, assignment μ falsifies ϕ. A CNF formula ϕ is *unsatisfiable* if there exists no assignment satisfying ϕ.

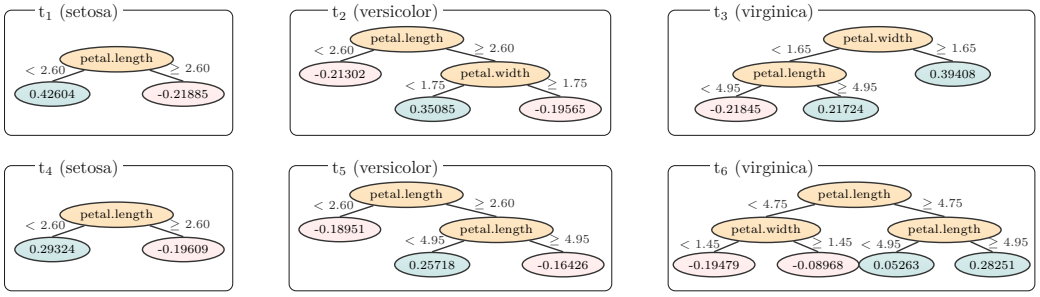
In the context of unsatisfiable formulas, the MaxSAT problem consists in finding a truth assignment that maximizes the number of satisfied clauses. Hereinafter, we use a variant of MaxSAT called Partial Weighted MaxSAT [3, Chapters 23 and 24]. The formula ϕ in this variant is represented as a conjunction of *hard* clauses $\mathcal{H}$, which must be satisfied, and *soft* clauses $\mathcal{S}$ where each of them is associated with a weight representing a preference to satisfy them, i.e. $\phi = \mathcal{H} \land \mathcal{S}$. Partial Weighted MaxSAT problems aim at finding a truth assignment μ that satisfies all hard clauses and maximizes the total weight of satisfied soft clauses.

Classification Problems. We consider classification problems with a set of classes[1] $\mathcal{K} = \{1, \ldots, k\}$, and a set of features $\mathcal{F} = \{1, \ldots, m\}$. The value of each feature $i \in \mathcal{F}$ is taken from its corresponding (numeric) domain D_i. As a result, the entire feature space is defined as $\mathbb{F} \triangleq \prod_{i=1}^{m} D_i$. A concrete point represented by $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, s.t. each v_i is a constant

[1] Non-integer class labels can be mapped to a set $\{1, \ldots, |\mathcal{K}|\}$.

IF	“petal.length” < 2.60	THEN	class = “setosa”
IF	$2.60 \leq$ “petal.length” $< 4.95 \wedge$ “petal.width” < 1.75	THEN	class = “versicolor”
IF	“petal.length” $\geq 2.60 \wedge$ “petal.width” ≥ 1.75	THEN	class = “virginica”
IF	“petal.length” ≥ 4.95	THEN	class = “virginica”

(a) Decision set



(b) BT model [5] consisting of 2 trees per class, each of depth ≤ 2, adopted from [17]

Figure 1 Example DS and BT models computed on the well-known *Iris* classification dataset.

value taken by feature $i \in \mathcal{F}$, together with its corresponding class $c \in \mathcal{K}$, represented by a pair $(\mathbf{v}, c)$, indicate a *data instance* or *example*. With a slight abuse of notation and whenever convenient, a data point $\mathbf{v} \in \mathbb{F}$ is also referred to as an instance. Finally, $\mathbf{x} = (x_1, \ldots, x_m)$ denotes a vector of feature variables $x_i \in D_i$, $i \in \mathcal{F}$, used for reasoning over points in $\mathbb{F}$.

A classifier defines a *classification function* τ: $\mathbb{F} \to \mathcal{K}$. The objective of classification problems is to learn a function τ to generalize well on unseen data given a training dataset $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$, where each instance $e_d \in \mathcal{E}$ is a pair of $(\mathbf{v}_d, c_d)$. Classification problems are conventionally posed as an optimization problem, i.e. either to minimize the complexity of τ, or maximize its accuracy, or both.

Rules, Decision Sets and Gradient Boosted Trees. Multiple ways exist to learn classifiers given data $\mathcal{E}$. This paper focuses on arguably one of the most interpretable models, i.e. decision sets, trained by *compiling* gradient boosted trees.

A *decision rule* is in the form of “IF antecedent THEN prediction”, where the antecedent is a set of feature literals. Informally, a rule is said to classify an instance $\mathbf{v} \in \mathbb{F}$ as 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) is an unordered set of decision rules $\mathcal{R}$. An instance $(\mathbf{v}, c) \in \mathcal{E}$ is misclassified by a DS if either there exists no rule in $\mathcal{R}$ matching $\mathbf{v}$, or there exists a rule classifying $\mathbf{v}$ as a class $c' \in \mathcal{K}$ s.t. $c' \neq c$.

A *gradient boosted tree* (BT) is a tree ensemble $\mathfrak{T}$ defining sets of decision trees $T_c \in \mathfrak{T}$ for each class $c \in [|\mathcal{K}|]$, where T_c comprises $N \in \mathbb{N}_{>0}$ trees t_{kz+c}, $z \in \{0, \ldots, N-1\}$, $k = |\mathcal{K}|$. Given an instance $\mathbf{v} \in \mathbb{F}$, its class is obtained by computing 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 which has the maximum score, i.e. $\operatorname{argmax}_{c \in [|\mathcal{K}|]} w(\mathbf{v}, c)$. Whenever convenient, $\mathfrak{n} \in t$ denotes a non-terminal node, where $t \in \mathfrak{T}$ represents an arbitrary decision tree. Moreover, each such $\mathfrak{n}$ indicates a feature condition in the form of $x_i < d$, where feature $i \in \mathcal{F}$ and *splitting threshold* $d \in \mathcal{D}_i$.

▶ **Example 1.** Figure 1 shows DS and BT models trained on the Iris dataset, which has 4 numeric features and 3 classes: *“setosa”*, *“versicolor”*, and *“virginica”*. Observe that

Table 1 Several instances extracted from *Iris* dataset.

#	sepal.length	sepal.width	petal.length	petal.width	class
e_1	5.1	3.5	1.4	0.2	setosa
e_2	7.7	2.6	6.9	2.3	virginica
e_3	5.6	2.5	3.9	1.1	versicolor
e_4	6.2	2.8	4.8	1.8	virginica
e_5	5.6	2.8	4.9	2.0	virginica

instance $\mathbf{v}_1 \in e_1$ shown in Table 1 is classified as *"setosa"* by the first rule of the DS. In the BT model, each class $c \in [3]$ is represented by 2 trees t_{3z+c}, $z \in \{0,1\}$. Thus, it also classifies $\mathbf{v}_1$ as *"setosa"*, since the score of this class $w(\mathbf{v}_1, 1) = t_1 + t_4 = 0.71928$ is higher than the score of *"versicolor"* $w(\mathbf{v}_1, 2) = t_2 + t_5 = -0.40253$ and the score of *"virginica"* $w(\mathbf{v}_1, 3) = t_3 + t_6 = -0.41324$. ◻

Interpretability and Explanations. Interpretability is not formally defined as it is considered to be a subjective concept [33]. In this paper interpretability is defined as the overall succinctness of the information offered by an ML model to justify a provided prediction. Moreover, following earlier work [48, 20], we equate explanations for ML models with *abductive explanations* (AXps), which are subset-minimal sets of features sufficient to explain a given prediction. Concretely, given an instance $\mathbf{v} \in \mathbb{F}$ and a prediction $c = \tau(\mathbf{v}) \in \mathcal{K}$, an AXp is a subset-minimal set of features $\mathcal{X} \subseteq \mathcal{F}$ such that

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

▶ **Example 2.** Consider the setup of Example 1. Given instance $\mathbf{v}_1$, observe that for any instance with *"petal.length"* $= 1.4$, the BT is guaranteed to predict *"setosa"* independently of the values of other features, since the weights for *"setosa"* and *"versicolor"* are 0.71928 and -0.40253 respectively as before, and the maximal weight for *"virginica"* is $0.39408 - 0.08968 = 0.30440$. Thus, the (only) AXp $\mathcal{X}$ for the prediction for e_1 made by the BT model is {*"petal.length"*}. ◻

Explanations in BTs. Formal reasoning has been recently applied to computing AXps for BT models, with the key difficulty being how to effectively reason about the aggregation over a large number of trees in a BT model. Recent work applied satisfiability modulo theory (SMT) [21] or mixed integer linear programming (MILP) solvers [42, 27] to directly address the linear summations arising in the BT encoding. Hereinafter, we build on the recent MaxSAT approach [17], which maps the aggregation reasoning to a set of MaxSAT queries to avoid a costly encoding of the linear constraints into CNF. Also, [17] demonstrates how a MaxSAT query can be made such that (1) holds if and only if the *optimal* value of the constructed objective function is negative.[2] In general, assuming that each feature $i \in \mathcal{F}$ is numeric (continuous), the approach orders the set of splitting thresholds $\{d_{i1}, ..., d_{ih_i}\}$ in a BT $\mathfrak{T}$ for each feature i, where h_i is the total number of thresholds of feature i in $\mathfrak{T}$ and $d_{ij} \in \mathcal{D}_i$ for $j \in [h_i]$. Given an instance $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, the above approach associates each value v_i with a single interval I'_i from the set of disjoint intervals $\mathbb{D}_i = \{ I_{i1} \equiv [\min(\mathcal{D}_i), d_{i1}),\ I_{i2} \equiv [d_{i1}, d_{i2}),\ \ldots,\ I_{ih_i+1} \equiv [d_{ih_i}, \max(\mathcal{D}_i)] \}$. Thus, AXp extraction

[2] The reader is referred to [17] for the details.

boils down to finding a subset-minimal subset $\mathcal{X} \in \mathcal{F}$ s.t.

$$\forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} x_i \in I'_i\right] \rightarrow (\tau(\mathbf{x}) = c) \tag{2}$$

▶ **Example 3.** Recall Example 2 and assume *"petal.length"* and *"petal.width"* have indices 3 and 4. Note that the sets of splitting thresholds for feature *"petal.length"* $\{d_{31} = 2.60, d_{32} = 4.75, d_{33} = 4.95\}$ and for feature "petal.width" $\{d_{41} = 1.45, d_{42} = 1.65, d_{43} = 1.75\}$. Let $min(\mathcal{D}_3) = -\infty$ and $min(\mathcal{D}_4) = 0.1$. Then we can associate the values of features 3 and 4 in our instance $\mathbf{v}_1 \in e_1$ with intervals $I_{31} \equiv (-\infty, 2.60)$ and $I_{41} \equiv [0.1, 1.45)$. Hence by (2), the AXp shown in Example 2 can in fact be seen as a rule $\langle$*IF "petal.length"* $<$ 2.60 *THEN class* $=$ *"setosa"*$\rangle$. □

3 Related Work

Interpretable decision sets are logic-based ML models that can be traced back to the 70s and 80s [39, 15, 4, 45]. To the best of our knowledge, [6] proposed the first approach to decision sets, which were introduced as the variant of decision lists [45, 7]. The first method making use of logic and optimization to synthesize a disjunction of rules that match a given dataset was proposed in [26]. Recent work [29] argued that decision sets are more interpretable than the other logic-based models, i.e. decision lists and decision trees. This work uses smooth local search to generate a set of rules first and heuristically minimizes a linear combination of criteria afterwards, e.g. the size of a rule, their maximum number, overlap or error.

Since then a number of works proposed the use of logic reasoning and optimization procedures to train DS models [22, 36, 12, 50, 18] claiming to significantly outperform the approach of [29] in terms of accuracy and performance. Among those, the works closest to ours are [22, 50, 18]. They proposed SAT-based approaches to computing smallest-size decision sets that *perfectly* agree with the training data by minimizing either the number of rules [22, 18] or the number of literals [50, 18] used in the model. Additionally, [50] is capable of computing *sparse* decisions sets that trade off training accuracy for model size. Despite the dramatic performance increase achieved in [18], all the approaches above suffer from scalability issues.

Post-hoc explainability is one of the major approaches to XAI. Besides a plethora of heuristic sampling-based methods to post-hoc explainability [43, 34, 44], a formal reasoning based approach to computing abductive explanations [48, 20] stands out. AXps can be related with prime implicants of the decision function (hence an alternative name *prime implicant explanations*, *PI-explanations*) associated with ML predictions and are guaranteed to capture the semantics of the ML models in the entire feature space. Although hard to compute in general, AXps were shown to be effectively computable for BT models by an incremental MaxSAT-based approach [17].

Our work aims at making a bridge between interpretable DS models and AXp computation by exploiting the latter for training the former. Given a BT model, it focuses on generating decision rules that agree with the BT. Each rule represents an AXp for the prediction made by the BT model, resulting in a DS model in a way *guided* by the original BT model. The approach is shown to outperform the prior logic-based approaches to DS inference in terms of test accuracy and performance. Note that despite prior attempts to train sparse models guided by tree ensembles [38], to our best knowledge, none of the existing works have applied formal post-hoc explanations to compile interpretable models.

Finally, our approach can be related to the existing line of work on *knowledge distillation* [11, 13], where an interpretable model is trained to approximate a hard-to-interpret

black-box model, which is often seen as teacher-to-student knowledge transfer. Note that in contrast to knowledge distillation, our approach is able to *compile* a BT into an *equivalent* DS if we consider the entire feature space, as shown below.

4 Decision Sets by Boosted Tree Compilation

Based on [17], this section details a MaxSAT-based approach to compiling a BT into a DS where each rule in the DS is equivalent to a prime implicant of the BT classification function.

4.1 Rule Extraction

Recall that an AXp, as defined in (1) and (2), can be seen as an *if-then* rule. Given a hard-to-interpret BT model, the AXp extraction approach of [17] can be modified to compute an interpretable DS consisting of a *set* of AXps for the BT. However, when the features are continuous (numeric), this potential approach suffers from the following issue. Recall that an AXp $\mathcal{X} \in \mathcal{F}$ indicates a set of *concrete* feature values that are sufficient to explain a prediction $c = \tau(\mathbf{v})$ for a certain instance $\mathbf{v} \in \mathbb{F}$. Although this same AXp can explain other instances compatible with it, its applicability in general is at the mercy of expressivity of the feature literals used in the AXp, i.e. equality literals and succinct interval membership in the case of (1) and (2), respectively. Motivated by this limitation, we propose to compute AXps over the literals intrinsic to the BT model aiming at getting feature intervals that are as general as possible, as detailed below.[3]

In contrast to the work of [17], which associates each feature value $v_i \in D_i$ with a single *narrowest* interval I'_i covering the value, we exploit all the splitting points used by the BT for feature i and identify all of the corresponding literals satisfied by the feature value v_i. Note that the original MaxSAT encoding [17] introduces a single Boolean variable o_{ij} for each literal $x_i < d_{ij}$ with d_{ij} being a j'th threshold used in the BT for feature i, s.t. $o_{ij} = 1$ iff $x_i < d_{ij}$ holds true. This way, each positive o_{ij} represents an upper bound on the value of x_i while each negative $\neg o_{ij}$ represents a lower bound on x_i.

▶ **Example 4.** Feature 3 (*"petal.length"*) from Example 3 has 3 thresholds: $d_{31} = 2.60$, $d_{32} = 4.75$, $d_{33} = 4.95$. Boolean variables o_{31}, o_{32}, and o_{33} are set to true iff $x_3 < 2.60$, $x_3 < 4.75$, and $x_3 < 4.95$, respectively. Let feature 3 take value 3.9 in the instance we want to explain. Observe how we can immediately assign literals $\neg o_{31}$, o_{32}, and o_{33} to true. □

Next, given an instance $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, let us construct a complete conjunction $\bigwedge_{i \in \mathcal{F}, j \in [h_i]} \widetilde{o}_{ij}$ of literals $\widetilde{o}_{ij}$ s.t. $\widetilde{o}_{ij}$ is to be replaced by o_{ij} if $v_i < d_{ij}$ and replaced by $\neg o_{ij}$ otherwise. By construction, this conjunction holds true for instance $\mathbf{v}$. Now, given this conjunction of literals, we can apply the existing approach of [17] to extract a subset-minimal explanation $\mathcal{Y} \subseteq \{\widetilde{o}_{ij} \mid i \in \mathcal{F}, j \in [h_i]\}$ for instance $\mathbf{v}$ over literals $\widetilde{o}_{ij}$ s.t.

$$\forall (\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{l \in \mathcal{Y}} l\right] \rightarrow (\tau(\mathbf{x}) = c) \tag{3}$$

Such an explanation $\mathcal{Y}$ may (or may not) define either a lower bound on feature i, an upper bound, or both, aiming to construct the *most general* interval for each feature $i \in \mathcal{Y}$. Hence, we informally refer to such explanations as *generalized* AXps or simply *rules* (hereinafter, we use both interchangeably).

[3] An alternative to our approach is *inflation* of abductive explanations, which is discussed in [23, 24]. Given an AXp, it aims at extending the set of values covered by each feature literal in the AXp while the AXp condition (1) still holds.

Algorithm 1 Deletion-based Rule Extraction

Function: RuleExtract($\mathfrak{T}, \mathbf{v}, c, \mathcal{E}$)
Input: $\mathfrak{T}$: BT defining $\tau(\mathbf{x})$, $\mathbf{v}$: Instance, c: Prediction, i.e. $c = \tau(\mathbf{v})$ $\mathcal{E}$: Training data
Output: $\mathcal{Y}$: Subset-minimal rule

1: $\langle \mathcal{H}, \mathcal{S} \rangle \leftarrow \texttt{Encode}(\mathfrak{T})$
2: $\mathcal{Y} \leftarrow \texttt{Init}(\mathfrak{T}, \mathbf{v})$
3: $\mathcal{Y} \leftarrow \texttt{Sort}(\mathcal{Y}, \mathcal{E})$
4: **for** $l \in \mathcal{Y}$ **do**
5: **if** $\texttt{EntCheck}(\langle \mathcal{H}, \mathcal{S} \rangle, c, \mathcal{Y} \setminus \{l\})$ **then**
6: $\mathcal{Y} \leftarrow \mathcal{Y} \setminus \{l\}$
7: **return** $\mathcal{Y}$

▶ **Example 5.** Consider instance $\mathbf{v}_3$ predicted as *"versicolor"* by the BT (observe that $v_3 = 3.9$ and $v_4 = 1.1$) and recall the thresholds for features 3 and 4 discussed in Example 3. We can compute a generalized AXp $\mathcal{Y} = \{\neg o_{31}, o_{33}, o_{43}\}$ representing the second rule of the DS shown in Figure 1a. The original approach of [17] would instead compute an AXp defining the narrowest intervals for features 3 and 4, representing a rule: $\langle$*IF* $2.60 \leq$ *"petal.length"* $<$ $4.75 \wedge$ *"petal.width"* < 1.45 *THEN class* $=$ *"versicolor"*$\rangle$, which is far less general than $\mathcal{Y}$. ◻

A possible rule extraction procedure is outlined in Algorithm 1. (Please ignore line 3 for now; feature sorting is described in Section 4.2). The input BT model $\mathfrak{T}$ is encoded into MaxSAT by applying the approach of [17]. Given an instance $\mathbf{v} \in \mathbb{F}$, the initial set of literals $\mathcal{Y} = \{\widetilde{o}_{ij} \,|\, i \in \mathcal{F}, j \in [h_i]\}$ is created. Note that any feature $i \in \mathcal{F}$ unused in the BT $\mathfrak{T}$ is excluded from $\mathcal{Y}$. The rest of the procedure implements the standard deletion-based AXp extraction [20], i.e. it iterates through all literals in $\mathcal{Y}$ one by one, and checks which of the them can be safely removed such that entailment (3) still holds.

▶ **Example 6.** Consider our running example model and instance $\mathbf{v}_2 \in e_2$ from Table 1 predicted as *"virginica"* by the BT $\mathfrak{T}$. Given the thresholds for features 3 and 4 in Example 3, set $\mathcal{Y}$ is initialized to $\{\neg o_{31}, \neg o_{32}, \neg o_{33}, \neg o_{41}, \neg o_{42}, \neg o_{43}\}$. The other two features are excluded from $\mathcal{Y}$ since they are irrelevant to the classification function in $\mathfrak{T}$. Applying Algorithm 1 results in extracting a subset-minimal generalized AXp $\mathcal{Y} = \{\neg o_{33}\}$, which represents the rule $\langle$*IF petal.length* ≥ 4.95 *THEN class* $=$ *"virginica"*$\rangle$. ◻

▶ Remark 7. Algorithm 1 relies on deciding whether formula (3) holds for each feature in explanation $\mathcal{Y}$. Here, this is done by means of a series of incremental core-guided MaxSAT oracle calls [19, 17]. One may wonder whether or not incomplete *anytime* MaxSAT solving [31, 35, 2, 32] can be applied in this setting. Although this may look plausible at first glance, time-restricted anytime MaxSAT algorithms can only *over-approximate* exact MaxSAT solutions while (3) holds *if and only if* the exact value of the objective function is negative. Therefore, an over-approximation of a MaxSAT solution is *never able* to prove the validity of (3) and so none of the features being tested can be discarded in the case of incomplete MaxSAT algorithms, which defies the purpose of Algorithm 1.

4.2 Boosted Tree Compilation

As mentioned above, generalized AXps can be seen as general decision rules that can be applied to an enormous number of instances. Therefore, it makes little sense to extract such rules for each instance in the feature space $\mathbb{F}$. Instead, one can devise an on-demand

Algorithm 2 Compile a BT into a DS

Function: Compile($\mathfrak{T}, \tau, \mathcal{C}$)
Input: $\mathfrak{T}$: BT defining $\tau(\mathbf{x})$, τ: Classification function in $\mathfrak{T}$, $\mathcal{C}$: Coverage set
Output: $\mathcal{R}$: Set of Rules

$\mathcal{R} \leftarrow \emptyset$
$\mathcal{C}_u \leftarrow \mathcal{C}$
while $\mathcal{C}_u \neq \emptyset$ **do**
  $\mathbf{v} \leftarrow \texttt{GetInst}(\mathcal{E}_u)$
  $\mathcal{Y} \leftarrow \texttt{RuleExtract}(\mathfrak{T}, \mathbf{v}, c = \tau(\mathbf{v}), \mathcal{E}_u)$
  $\mathcal{C}_c \leftarrow \texttt{GetCover}(\mathcal{Y}, \mathcal{C}_u)$
  $\mathcal{C}_u \leftarrow \mathcal{C}_u \backslash \mathcal{C}_c$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{X}'$
return $\mathcal{R}$

compilation process, i.e. given a *yet uncovered* instance $\mathbf{v} \in \mathbb{F}$, we can apply Algorithm 1 to extract a rule covering $\mathbf{v}$ (and some other instances). Clearly, *exhaustive* compilation of a BT, i.e. if the target is to cover all the instances in $\mathbb{F}$ with generalized AXps of the BT, is computationally expensive given that AXp extraction for tree ensembles is hard for D^{P} [25]. This can also lead to the large size of the resulting DSes making them hard to interpret. In practice, *local* compilation aiming at capturing the behavior of the BT on the training data only, is sufficient to generate a DS, which is both accurate and interpretable.

The proposed approach to compiling a BT $\mathfrak{T}$ into a DS $\mathcal{R}$ is shown in Algorithm 2. We initialize the set $\mathcal{C}_u$ of currently uncovered instances to be equal to $\mathcal{C}$, i.e. the set of examples we wish to cover. The algorithm represents a loop generating rules until the set of computed rules $\mathcal{R}$ covers all instances in coverage set data $\mathcal{C}_u$, i.e. until there is no uncovered instances in $\mathcal{C}$. Each iteration of the algorithm selects an instance $\mathbf{v}$ from $\mathcal{C}_u$. Afterwards, a generalized AXp $\mathcal{Y}$ for the prediction $c = \tau(\mathbf{v})$ by the BT $\mathfrak{T}$ (recall that $\mathfrak{T}$ is meant to compute classification function $\tau(\mathbf{x})$) is extracted by invoking Algorithm 1. The iteration proceeds by updating the set of rules $\mathcal{R}$ and the set of uncovered instances $\mathcal{C}_u$. The algorithm terminates when all the instances in the coverage set $\mathcal{C}$ are covered and returns a compiled DS $\mathcal{R}$.

▶ **Proposition 8.** *Let $\mathfrak{T}$ be a BT and $\mathcal{R}$ be a DS returned by Algorithm 2 for $\mathfrak{T}$. Then $\mathcal{R} \equiv \mathfrak{T}$ with respect to $\mathcal{C}$.*

We consider two usages of the algorithm: for *exhaustive compilation* the coverage set $\mathcal{C} = \mathbb{F}$ is all possible feature combinations (in practice we model this coverage set implicitly, rather than in its explicit exponential sized form), and for *training set compilation* where $\mathcal{C} = \mathcal{E}$ is the training set. Based on the properties of prime implicants, Proposition 8 states that as a generalized AXp $\mathcal{Y} \in \mathcal{R}$ is a formal explanation for a prediction made by BT $\mathfrak{T}$, a compiled DS captures the semantics of the original model $\mathfrak{T}$ on *coverage set* $\mathcal{C}$, assuming everything else is a *don't care*. Furthermore, if the process is applied subject to coverage set $\mathcal{C} = \mathbb{F}$, i.e. when we target the entire feature space $\mathbb{F}$, then $\mathcal{R}$ and $\mathfrak{T}$ behave identically, i.e. they compute the same classification function $\tau(\mathbf{x})$.

▶ **Corollary 9.** *Let Algorithm 2 return a DS $\mathcal{R}$ for a BT $\mathfrak{T}$. Then there is no instance in feature space $\mathbb{F}$ covered by two distinct rules $\mathcal{Y}_1, \mathcal{Y}_2 \in \mathcal{R}$ predicting inconsistent classes $c_1 \neq c_2$.*

As each generalized AXp for $\mathfrak{T}$ represents a prime implicant of the decision function $\tau(\mathbf{x})$

computed over literals $\widetilde{o}_{ij}$, the above corollary claims that there are no overlapping rules in the result DS $\mathcal{R}$. This contrasts with other modern approaches to DS inference, where rule overlap is known to be a problem [29, 22]. Note that this approach still suffers from another common issue of DS models: namely, if DS $\mathcal{R}$ is computed for the training data $\mathcal{E}$, there may still be instances in $\mathbb{F}$ uncovered by $\mathcal{R}$.

▶ **Example 10.** Consider the running example BT model shown in Figure 1b. Its compiled DS representation computed by Algorithm 2 is shown in Figure 1a. Observe that there is no rule overlap in the DS computed. In fact, as the DS is computed by taking into account feature space $\mathbb{F}$, it computes the same classification function as the original BT model. □

Feature Sorting. Intuitively, how general and so applicable a rule is depends on how frequently the features used in it appear in the training data $\mathcal{E}$ labeled with the target class. Thus, a simple heuristic to apply when extracting a rule for prediction $c = \tau(\mathbf{v})$ is to sort the initial state of $\mathcal{Y} = \{\widetilde{o}_{ij} \,|\, i \in \mathcal{F}, j \in [h_i]\}$ based on how frequently the corresponding literals $\widetilde{o}_{ij}$ apply in examples $\mathcal{E}$ labeled with c. This feature sorting represented by line 3 in Algorithm 1 in practice (according to our experiments) results in significantly more general rules and so overall smaller DSes.

Anytime Property. Most widely used reasoning-based algorithms to infer DSes provide a solution only if the computation is completed; otherwise, no decision set is reported. In contrast to these, the proposed approach is an *anytime* algorithm, i.e. it can return a *valid* DS $\mathcal{R}$ even though the compilation process is interrupted before all the coverage set instances $\mathcal{C}$ are covered. Furthermore, it can generate a more comprehensive DS $\mathcal{R}$, which covers more instances as it keeps going, i.e. after we have covered $\mathcal{C} \subseteq \mathbb{F}$ we can continue running the algorithm for the (unseen) instances of $\mathbb{F}$.

4.3 Post-Hoc Model Reduction

The compiled DS $\mathcal{R}$ can be large (in terms of either the number of rules or the total number of literals) since each generalized AXp $\mathcal{Y} \in \mathcal{R}$ may need a significant number of literals to explain a prediction made by BT $\mathfrak{T}$, or/and many rules are required to explain all instances of $\mathcal{C}$. Once the target DS is obtained, we can apply post-hoc heuristic methods for reducing its size and so making it more interpretable. The methods below are in a way inspired by the optimization problems studied in [18, 50]. Although these ideas are applicable to any DS inference method once the result model is devised, they do not look necessary for standard DS inference algorithms as they minimize the model while training. On the contrary, no minimization is applied in the rule enumeration process described above and so post-hoc model reduction plays a vital role in our approach to reduce the size of final DS models.

Reducing the Number of Rules. Given a set of rules $\mathcal{R}$, we can compute a minimum subset $\mathcal{R}^\star \subseteq \mathcal{R}$ that is still equivalent to the BT $\mathfrak{T}$ wrt. the coverage set $\mathcal{C}$ using discrete optimization, e.g. integer-linear programming (ILP). Concretely, the approach aims at selecting the smallest-size subset $\mathcal{R}^\star \subseteq \mathcal{R}$ that covers all instances in $\mathcal{C}$, where $\mathcal{R}$ is the compiled DS from $\mathfrak{T}$. Here, the size of $\mathcal{R}^\star$ is measured as the total number of literals used. This can be done by solving the following *set cover problem* [28]. Namely, for each rule $\mathcal{Y}_j \in \mathcal{R}$, we introduce a Boolean variable u_j such that $u_j = 1$ iff $\mathcal{Y}_j$ is included in $\mathcal{R}^\star$. Additionally, a Boolean variable y_{ij} is used to indicate that $\mathcal{Y}_j$ covers $e_i \in \mathcal{C}$. As a result,

the weighted set cover problem for minimizing the total number of literals used is as follows:

$$\text{minimize} \quad \sum_{j=1}^{|\mathcal{R}|} (|\mathcal{Y}_j| + 1) \cdot u_j \tag{4}$$

$$\text{subject to} \quad \forall_{i \in [n]} \sum_{j=1}^{|\mathcal{R}|} y_{ij} \cdot u_j \geq 1 \tag{5}$$

Reducing the Number of Literals. Additionally, one can minimize the total number of literals used in the rules of $\mathcal{R}$. Given a rule $\mathcal{Y} \in \mathcal{R}$, this can be done either lexicographically by maximizing rule accuracy followed by size minimization, or by optimizing both, or trading off misclassifications for rule size – in either case, a single MaxSAT call per rule to minimize can be made. The intution is that if a rule $\mathcal{Y}$ misclassifies k instances then its optimized version $\mathcal{Y}^\star \subseteq \mathcal{Y}$ should not result in many more misclassifications on training data $\mathcal{E}$. Recall that a rule misclassifies an instance $\mathbf{v}_k \in \mathcal{C}$ if it matches $\mathbf{v}_k$ but assigns it to a wrong class.

Inspired by [18], we introduce a Boolean variable p_k, which is true iff rule $\mathcal{Y}$ covers $\mathbf{v}_k$ — this holds if $\mathcal{Y}$ does not use any literals incompatible with $\mathbf{v}_k$. If $\mathcal{Y}_{\mathbf{v}_k} = \{\widetilde{o}_{ij} \,|\, i \in \mathcal{F}, j \in [h_i]\}$ are all the literals compatible with $\mathbf{v}_k$ then this can be modeled with constraints

$$\forall_{k \in [|\mathcal{C}|]} \cdot\, p_k \leftrightarrow \bigwedge_{l \in \mathcal{Y} \setminus \mathcal{Y}_{\mathbf{v}_k}} \neg l \tag{6}$$

Furthermore, let rule $\mathcal{Y}$ predict $c \in \mathcal{K}$ and let $\mathcal{C}_\ominus \subseteq \mathcal{C}$ contain all instances labeled with any other class. Thus, we can apply the objective below when minimizing rule $\mathcal{Y}$:

$$\sum_{l \in \mathcal{Y}} l + \sum_{k \in [|\mathcal{C}_\ominus|]} W \cdot p_k \tag{7}$$

If W is large enough, say $|\mathcal{C}| + 1$, this lexicographically minimizes misclassifications and then literals. If W is small, e.g. $1/\lambda \cdot |\mathcal{C}|$, this trades off $\lambda \cdot |\mathcal{C}|$ misclassifications for one literal.

5 Experimental Results

This section compares the proposed approach with the state-of-the-art DS learning algorithms on a variety of publicly available datasets in terms of accuracy, scalability, model and explanation size. The experiments are performed on an Intel Xeon 8260 CPU running Ubuntu 20.04.2 LTS, with the time limit of 3600s and the memory limit of 8GByte. Our experiments contain two parts, namely, exhaustive BT compilation and training-set BT compilation.

Prototype implementation. A prototype of the compilation-based approach to generating DSes was developed as a set of Python scripts using $\mathcal{C} = \mathcal{E}$, hereinafter referred to as *cpl*. The implementation of BT compilation exploits [17] and, therefore, makes use of the RC2 MaxSAT solver [19].[4] The BTs to be compiled are computed by XGBoost [5]; the number of trees per class in a BT model is 50 and the maximum depth of each tree is 3. Post-hoc literal reduction is done again with RC2 [19]. Let cpl_l denote the implementation applying

[4] Real weights in the objective function are not conventionally supported by MaxSAT solvers; the only other solver to support real weights besides RC2 is LMHS [47].

lexicographic optimization while $cpl_{l\lambda_1}$ trades off model accuracy for the number of literals used, with $\lambda_1 = 0.005$. Let cpl_r denote the implementation with post-hoc rule reduction applied using the Gurobi ILP solver [14]. The configuration with both post-hoc lexicographic optimization and rule reduction is denoted cpl_{lr}. Finally, the proposed approach applying exhaustive compilation $\mathcal{C} = \mathbb{F}$ is referred to as cpl_f.

Competition. Our approach is compared against: *twostg* a two-stage MaxSAT approach [18] for DSes perfectly accurate on the training data; *opt* another MaxSAT approach [50] for perfectly accurate DSes; sp_{λ_1} a sparse alternative to *opt* by the same authors (with $\lambda_1 = 0.005$) optimizing like $cpl_{l\lambda_1}$; $imli_1$ and $imli_{16}$ using MaxSAT-based IMLI [12] to minimize the number of literals given a predefined number of rules (we use 1 or 16); *ids* a state-of-the-art approach [29] based on smooth local search;[5] *ripper* a popular heuristic DS algorithm RIPPER [8]; and CN2 (referred to as *cn2*) another heuristic algorithm [7, 6].[6]

Datasets. For the evaluation, 59 publicly available datasets from UCI Machine Learning Repository [9] and Penn Machine Learning Benchmarks [41] are considered. We apply 5-fold cross validation, resulting in 295 pairs of training and test (unseen) data. For the sake of a fair comparison, the datasets used are preprocessed so that each original feature $i \in \mathcal{F}$ is replaced with a number of non-intersecting feature intervals $x_i < d_{ij}$ defined by the XGBoost model (see Section 2). This guarantees that all competitors tackle the same problem instances.

5.1 Exhaustive BT Compilation

The first experiment compares exhaustive compilation, where $\mathcal{C} = \mathbb{F}$ is the entire feature space. This is impractical except for 6 small benchmarks.

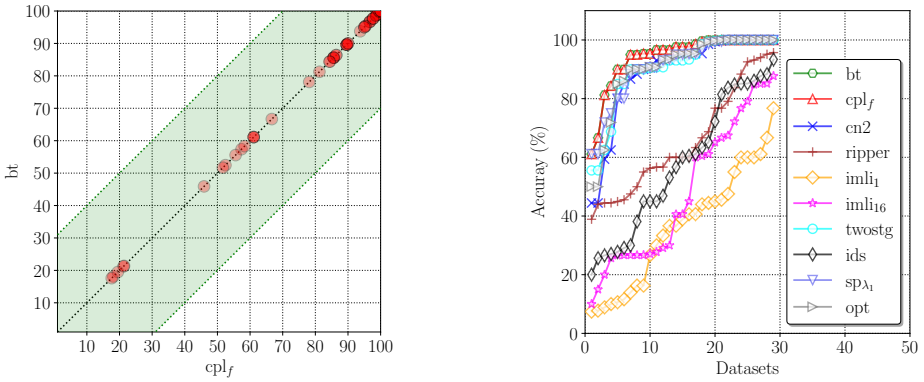
Results. Here we compare cpl_f with the competition in terms of accuracy, the total number of literals used and explanation size. We present the results as cactus plots showing the number of datasets that e.g. reach a certain accuracy, or finish in a certain runtime, for each method. These experimental results are shown in Figures 2 and 3 as well as the average results across folds are described in Table 2 where only the results of the datasets *completely* solved by compared competitors are presented. Note that cpl_f is nowhere near as scalable as the approaches described in the later experiments, but it is the *most accurate* approach to creating DSes we are aware of.

Test accuracy. An instance is considered misclassified if either there exists a rule of a wrong class that covers it, or it is not covered by any rule of the correct class. Thus, the test accuracy in this paper is calculated as $\frac{n-g}{n}$, where n is the total number of instances in the test data and g is the total number of misclassified instances. If an approach fails to train a model within the time limit, we assume its accuracy to be 0% for this dataset.

As can be seen in Figure 2b and Table 2, the best accuracy is achieved by BTs and cpl_f. In fact, these models share the same accuracy (this is also confirmed in Figure 2a), which

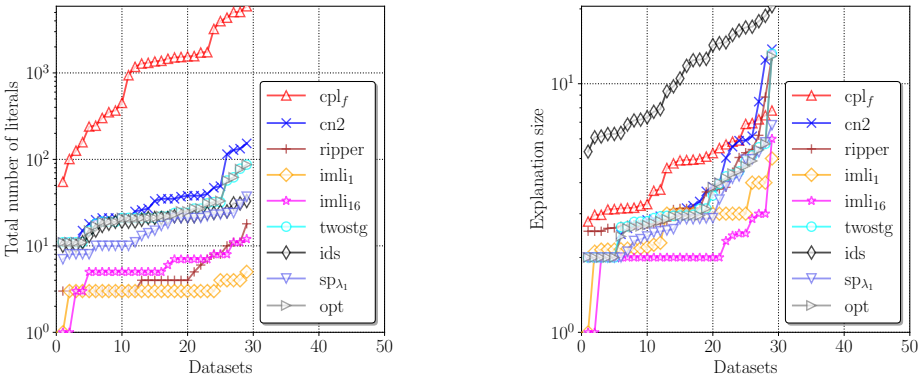
[5] Since the original implementation performs poorly [22], here we consider the new implementation of IDS [10], which is claimed to be orders of magnitude faster than the original implementation.

[6] Note that since RIPPER and IMLI compute a single class only given the training data, both of these competitors are augmented with a default rule predicting a class (1) different from the target class and (2) represented by the majority of training instances. Other algorithms, including our approach, incorporate a default rule that assigns a class based on the majority class in the training instances.



(a) Comparison with BT **(b)** Comparison with the others

Figure 2 Accuracy of exhaustive compilation. The *standard* interpretation of cactus plots is assumed, i.e. a plot sorts the datapoints for each method by the y-axis value, and then shows them in increasing order independently of other methods. Thus, the order of datasets/folds differs for different methods. Also, the order of datasets for the same method differs in different subplots.



(a) Number of literals used **(b)** Explanation size

Figure 3 Succinctness of exhaustive compilation.

should not come as a surprise given that cpl_f replicates the behavior of the BT in the entire feature space $\mathbb{F}$ (see Proposition 8).

Model Complexity. In general, complexity of a DS model can be measured by the total number of literals used in this DS. The total number of literals used in DS models is compared in Figure 3a and Table 2. Though the accuracy of DSes trained by cpl_f outperforms the other competitors, these models are significantly larger, which is no surprise given that cpl_f computes many more rules with no post-hoc reduction applied.

Explanation size. Explanation size is defined as the number of literals required to explain an instance.[7] This is arguably more important than the model size, since it defines "how hard" it is to understand an individual explanation. A small DS model tends to provide

[7] See [51] for details.

Table 2 Accuracy, number of literals used, and explanation size across folds.

Approach	Dataset					
	cardiotocography	hayes-roth	iris	new-thyroid	orbit	zoo
			Accuracy (%)			
bt	**100.0**	**84.38**	**96.0**	**96.74**	**99.66**	**96.0**
cpl$_f$	**100.0**	**84.38**	**96.0**	**96.74**	**99.66**	**96.0**
sp_{λ_1}	100.0	73.44	94.0	91.63	99.43	89.05
opt	100.0	70.63	93.33	91.63	99.54	93.05
twostg	100.0	71.25	92.67	92.09	99.54	91.1
cn2	100.0	62.5	92.67	93.02	99.54	89.1
ripper	45.3	66.25	57.33	80.93	94.11	60.33
ids	27.23	43.75	58.67	76.28	85.29	40.62
$imli_{16}$	27.23	38.75	25.34	69.77	70.55	43.33
$imli_1$	45.3	39.37	32.67	26.98	8.93	60.33
			Number of literals used			
cpl$_f$	3120.0	76.0	214.0	3614.2	729.8	1422
sp_{λ_1}	21.0	33.5	9.0	15.4	10.0	23.2
opt	21.0	63.6	19.4	23.0	11.8	30.0
twostg	21.0	64.2	19.8	22.6	11.8	29.8
cn2	21.0	116.2	27.2	36.6	13.2	40.8
ripper	**3.0**	12.8	5.0	8.2	4.0	**3.0**
ids	21.0	21.6	19.8	20.0	25.0	14.2
$imli_{16}$	5.0	**2.2**	7.4	7.4	6.4	5.0
$imli_1$	**3.0**	**2.2**	**3.0**	**4.2**	**3.0**	**3.0**
			Explanation size			
cpl$_f$	7.26	3.76	3.02	4.9	3.18	5.4
sp_{λ_1}	**2.0**	6.31	2.45	4.13	2.86	3.64
opt	**2.0**	5.41	2.76	4.3	2.94	2.96
twostg	**2.0**	5.4	2.87	4.23	2.94	3.33
cn2	**2.0**	6.94	3.02	4.47	3.02	4.05
ripper	2.73	10.15	4.3	4.3	3.15	2.59
ids	16.08	18.23	13.06	7.74	6.23	9.28
$imli_{16}$	**2.0**	**2.2**	**2.1**	**1.97**	**2.8**	2.46
$imli_1$	2.18	**2.2**	3.0	4.0	3.0	**2.2**

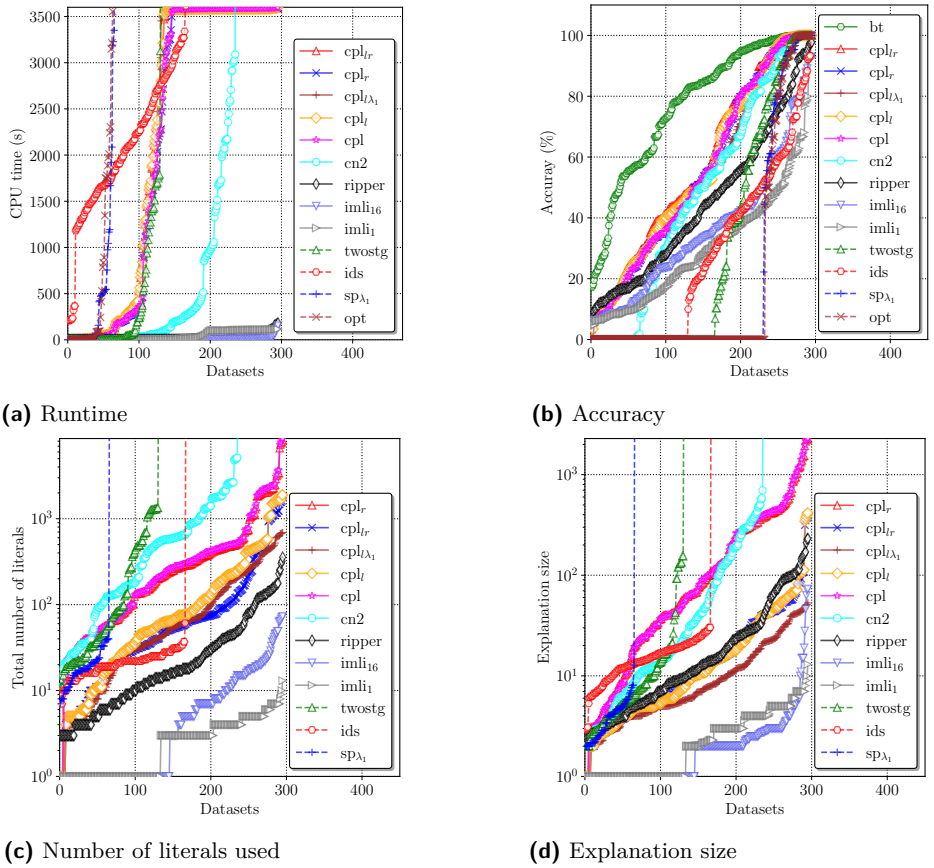
compact explanations but it is not always accurate. As can be seen in Figure 3b and Table 2 and similar to the total number of literals used in DSes, cpl_f requires more literals to explain an instance than all competitors except *ids*.

A crucial observation to make here is that we test explanation size for each of the test instances available. Although test data are meant to extrapolate the overall unseen data, such approximation of the unseen feature space is not ideal. As a result, there may be numerous instances in $\mathbb{F}$ *uncovered* by all the approaches but cpl_f, in which case it will be the *only* approach providing a user not only with a prediction but also with a succinct explanation of the prediction made.

5.2 BT Compilation Targeting Training Data

Compilation to cover the training set $\mathcal{C} = \mathcal{E}$ is much more efficient, and the main usage we expect of our algorithms.

Scalability. Figure 4a depicts scalability of all selected algorithms on the 295 considered datasets. Note that runtime of our approach includes BT training time. The best performance is demonstrated by the proposed implementation, i.e. *cpl* and cpl_*, $* \in \{l, r, lr, l\lambda_1\}$, where all selected datasets are solved within the time limit. This is not surprising since the approach is an anytime algorithm that can always return a valid DS. As for other competitors, the heuristic method *ripper* and the MaxSAT approaches $imli_1$ as well as $imli_{16}$ also solve all considered datasets. Next is the heuristic algorithm *cn2*, where 235 datasets are solved



(a) Runtime **(b)** Accuracy

(c) Number of literals used **(d)** Explanation size

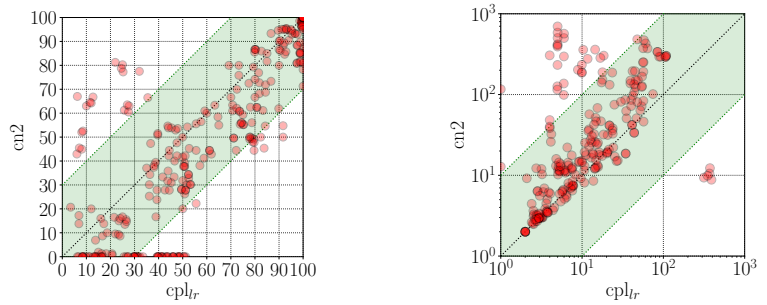
Figure 4 Summary of experimental results when the competitors aim at training a DS given training data $\mathcal{E}$ (i.e. $\mathcal{C} = \mathcal{E}$).

within the 3600s time limit. Followed by *ids*, which solves 166 considered datasets. The two-stage MaxSAT approach *twostg* successfully addresses 130 datasets, while the other MaxSAT algorithm for perfect decision sets *opt* and its sparse alternative sp_{λ_1} solve 65 and 63 datasets respectively.

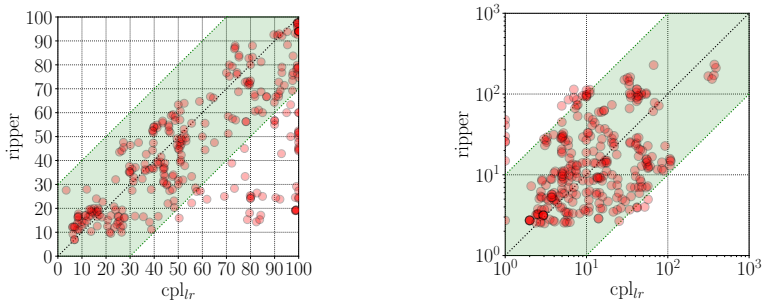
Test Accuracy. The accuracy among the selected approaches is shown in Figure 4b. The average accuracy among all selected datasets for BTs is 77.34%, beating all DS approaches. The highest accuracy among DSes is achieved by all the configurations of the proposed approach, i.e. *cpl* and cpl_*, where the average accuracy ranges from 54.01% ($cpl_{l\lambda_1}$) to 57.49% (cpl_{lr}).[8] Unsurprisingly, the accuracy in $cpl_{l\lambda_1}$ is lower than the other configurations since $cpl_{l\lambda_1}$ trades off training accuracy on the number of literals in the computation process.

Next most accurate are the heuristic methods *cn2* (48.03%) followed by *ripper* (44.81%). The average accuracy of $imli_{16}$ and $imli_1$ is 35.47% and 29.7% respectively, while the average accuracy of *twostg* is 29.6% and *ids* is 26.78. Finally, the worst accuracy is demonstrated by sp_{λ_1} and *opt* (18.84% and 18.27% on average respectively) as these tools fail to provide

[8] Note that most datasets we used represent non-binary classification. Also, DSes are not to be compared with BTs. As Figure 4b shows (and as our work aims to demonstrate), our approach outperforms the state-of-the-art DS inference methods in terms of accuracy.

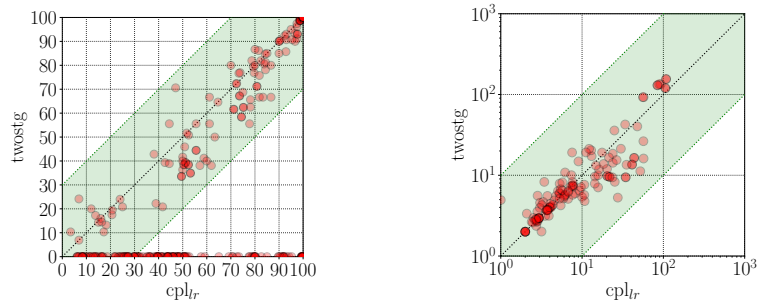


(a) Acry: vs. *cn2* **(b)** Xp size: vs. *cn2*

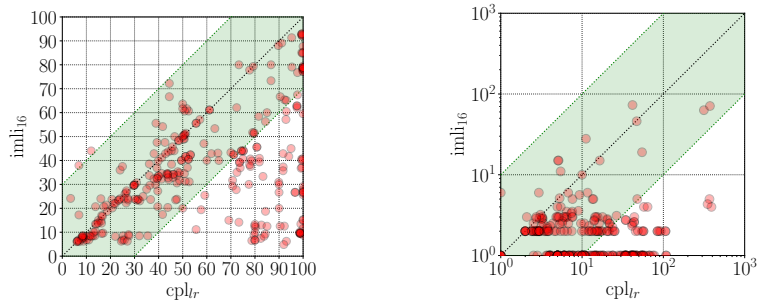


(c) Acry: vs. *ripper* **(d)** Xp sz: vs. *ripper*

Figure 5 Comparison of cpl_{lr} vs. *cn2* and *ripper* in terms of accuracy and explanation size.



(a) Acry: vs. *twostg* **(b)** Xp sz: vs. *twostg*



(c) Acry: vs. $imli_{16}$ **(d)** Xp sz: vs. $imli_{16}$

Figure 6 cpl_{lr} vs. $imli_{16}$ and *twostg* in terms of accuracy and explanation size.

prediction information for many datasets within the time limit. We will omit further discussion of *sp* and opt_{λ_1} since they solve so few datasets.

Model Complexity. Figure 4c illustrates the comparison among selected approaches regarding the total number of literals used in each DS solution. The average number of literals are in order: $imli_1$ (2.77), $imli_{16}$ (8.26), *ids* (21.14), *ripper* (38.47), $cpl_{l\lambda_1}$ (118.47), cpl_{lr} (157.53), cpl_l (213.27), *twostg* (265.98), cpl_r (584.39) *cpl* (620.82), *cn2* (700.49). Clearly, rule reduction and literal reduction can significantly reduce the size of the model without significantly affecting accuracy. Note how our approaches while significantly larger than the least accurate competitors, are significantly smaller than the most accurate competitor *cn2*.

Explanation Size. Figure 4d shows the explanation size for each competitor. The average explanation sizes are in order: $imli_1$ (2.61), $imli_{16}$ (3.00), $cpl_{l\lambda_1}$ (12.14), *ids* (15.28), *twostg* (17.5), cpl_{lr} (25.34), cpl_l (26.18), *ripper* (29.08), *cn2* (81.93), cpl_r (234.46), *cpl* (240.88). Figure 4d demonstrates that post-hoc literal reduction not only helps decrease the number of literals required to explain DS models, but also enables DSes to remain accurate, whereas rule reduction does not contribute to smaller explanations. With literal reduction applied our approaches are very competitive in terms of explanation size.

Detailed Comparison. While cactus plots allow us to compare many methods over a large suite of benchmarks, they do not allow direct comparison on individual benchmarks. We provide a detailed comparison of cpl_{lr} versus other decision set inference approaches in Figures 5 and 6, including *cn2*, *ripper*, *twostg*, and $imli_{16}$.[9] The scatter plots depicting explanation size are obtained for the datasets solvable by both competitors. Note that cpl_{lr} can generate more accurate DSes than the competitors. Also observe that the explanation size of DSes computed by cpl_{lr} is smaller than *cn2* and comparable with *twostg*. Although the explanation size of DSes in cpl_{lr} is larger than *ripper* and $imli_{16}$, the two approaches are less interpretable as they compute DSes representing only one class.

Summary. The experimental results were performed on various datasets, demonstrating that our approach computes DSes that outperform the state-of-the-art competitors in terms of accuracy and yield comparable explanation size to them.

6 Conclusions

This paper introduced a novel *anytime* approach to generating decision sets by means of on-demand extraction of generalized abductive explanations for boosted tree models. It can be used for exhaustive compilation of a BT model wrt. the entire feature space, or target a set of training instances. Augmented by a number of post-hoc model reduction techniques, the approach is shown to compute decision sets that are more accurate than decisions sets computed by the state-of-the-art algorithms and comparable with them in terms of explanation size.

As the proposed approach targets generating a decision set by compiling a BT, a natural line of future work is to extend the proposed approach to compile BTs into the other interpretable models, i.e. decision trees and decision lists, making use of AXp extraction for BTs. Additionally, another future work is to apply AXp extraction to compile other accurate black box models, e.g. neural networks, into decision sets.

[9] The average results across the folds are given in the appendix.

References

1. ACM. Fathers of the deep learning revolution receive ACM A.M. Turing award. http://tiny.cc/9plzpz, 2018.
2. Jeremias Berg, Emir Demirovic, and Peter J. Stuckey. Core-boosted linear search for incomplete MaxSAT. In *CPAIOR*, pages 39–56, 2019.
3. Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2021.
4. Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.
5. Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.
6. Peter Clark and Robin Boswell. Rule induction with CN2: some recent improvements. In *EWSL*, pages 151–163, 1991.
7. Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
8. William W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.
9. Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml.
10. Jiri Filip and Tomas Kliegr. Pyids-python implementation of interpretable decision sets algorithm by lakkaraju et al, 2016. In *RuleML+ RR*, 2019.
11. Nicholas Frosst and Geoffrey E. Hinton. Distilling a neural network into a soft decision tree. In *CEx@AI*IA*, volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.
12. 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.
13. Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819, 2021.
14. Gurobi Optimization. Gurobi optimizer reference manual, 2022. http://www.gurobi.com/.
15. Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.
16. Alexey Ignatiev. Towards trustable explainable AI. In *IJCAI*, pages 5154–5158, 2020.
17. Alexey Ignatiev, Yacine Izza, Peter J. Stuckey, and João Marques-Silva. Using MaxSAT for efficient explanations of tree ensembles. In *AAAI*, pages 3776–3785, 2022.
18. Alexey Ignatiev, Edward Lam, Peter J. Stuckey, and Joao Marques-Silva. A scalable two stage approach to computing optimal decision sets. In *AAAI*, pages 3806–3814, 2021.
19. Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. RC2: an efficient MaxSAT solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019.
20. Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.
21. Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019. URL: http://arxiv.org/abs/1907.02509, arXiv:1907.02509.
22. 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.
23. Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. On tackling explanation redundancy in decision trees. *J. Artif. Intell. Res.*, 75:261–321, 2022.
24. Yacine Izza, Alexey Ignatiev, Peter J. Stuckey, and Joao Marques-Silva. Delivering inflated explanations. *CoRR*, abs/2306.15272, 2023.
25. Yacine Izza and Joao Marques-Silva. On explaining random forests with SAT. In *IJCAI*, July 2021.
26. Anil P. Kamath, Narendra Karmarkar, K. G. Ramakrishnan, and Mauricio G. C. Resende. A continuous approach to inductive inference. *Math. Program.*, 57:215–238, 1992.

27 Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, Yuichi Ike, Kento Uemura, and Hiroki Arimura. Ordered counterfactual explanation by mixed-integer linear optimization. In *AAAI*, pages 11564–11574. AAAI Press, 2021.

28 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

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

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

31 Zhendong Lei and Shaowei Cai. Solving (weighted) partial maxsat by dynamic local search for SAT. In *IJCAI*, pages 1346–1352, 2018.

32 Zhendong Lei and Shaowei Cai. Nudist: An efficient local search algorithm for (weighted) partial maxsat. *Comput. J.*, 63(9):1321–1337, 2020.

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

34 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017.

35 Chuan Luo, Shaowei Cai, Kaile Su, and Wenxuan Huang. CCEHC: an efficient local search algorithm for weighted partial maximum satisfiability. *Artif. Intell.*, 243:26–44, 2017.

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

37 Joao Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350, 2022.

38 Hayden McTavish, Chudi Zhong, Reto Achermann, Ilias Karimalis, Jacques Chen, Cynthia Rudin, and Margo I. Seltzer. Fast sparse decision tree optimization via reference ensembles. In *AAAI*, pages 9604–9613, 2022.

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

40 Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and João Marques-Silva. Learning optimal decision trees with SAT. In *IJCAI*, pages 1362–1368, 2018.

41 Randal S. Olson, William G. La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10(1):36:1–36:13, 2017.

42 Axel Parmentier and Thibaut Vidal. Optimal counterfactual explanations in tree ensembles. In *ICML*, volume 139 of *PMLR*, pages 8422–8431, 2021.

43 Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144, 2016.

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

45 Ronald L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.

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

47 Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP hybrid MaxSAT solver. In *SAT*, pages 539–546, 2016.

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

49 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.

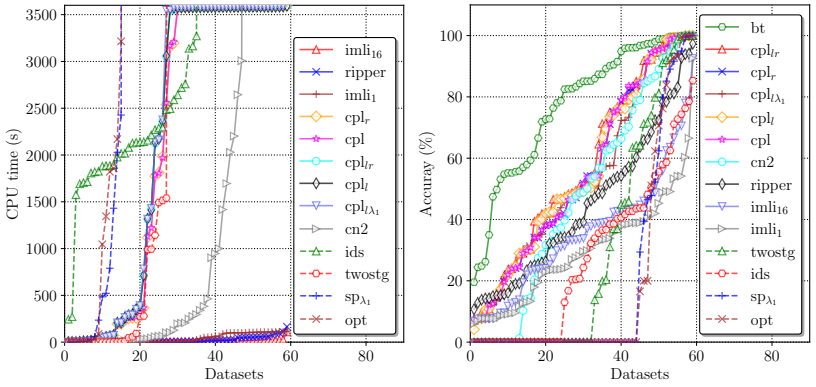
50 Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic. Computing optimal decision sets with SAT. In *CP*, pages 952–970, 2020.

51 Jinqiang Yu, Alexey Ignatiev, Peter J Stuckey, and Pierre Le Bodic. Learning optimal decision sets and lists with sat. *JAIR*, 72:1251–1279, 2021.

553 **52** Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, Nina Narodytska, and Joao Marques-Silva.
554 Eliminating the impossible, whatever remains must be true: On extracting and applying
555 background knowledge in the context of formal explanations. In *AAAI*, 2023.

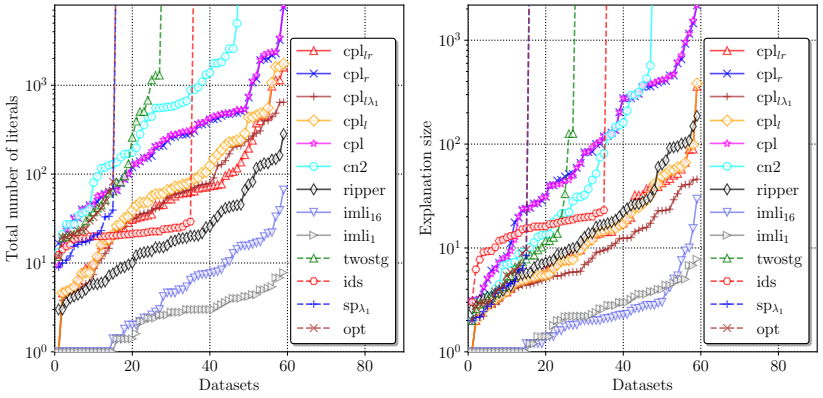
Appendices

A Summaries of Results Across Folds



(a) Runtime **(b)** Accuracy

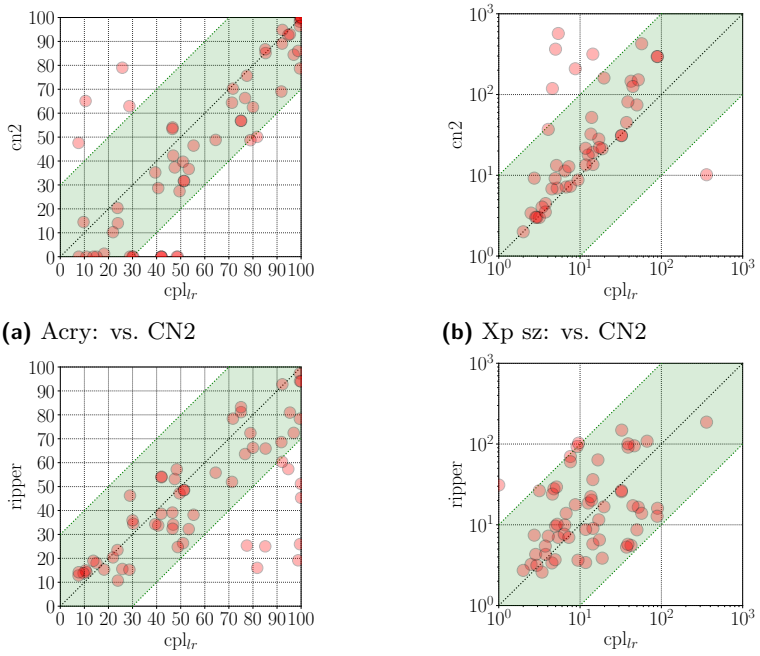
Figure 7 Experimental results of runtime and accuracy across folds.



(a) Number of literals used **(b)** Explanation Size

Figure 8 Experimental results of model complexity and explanation size across folds.

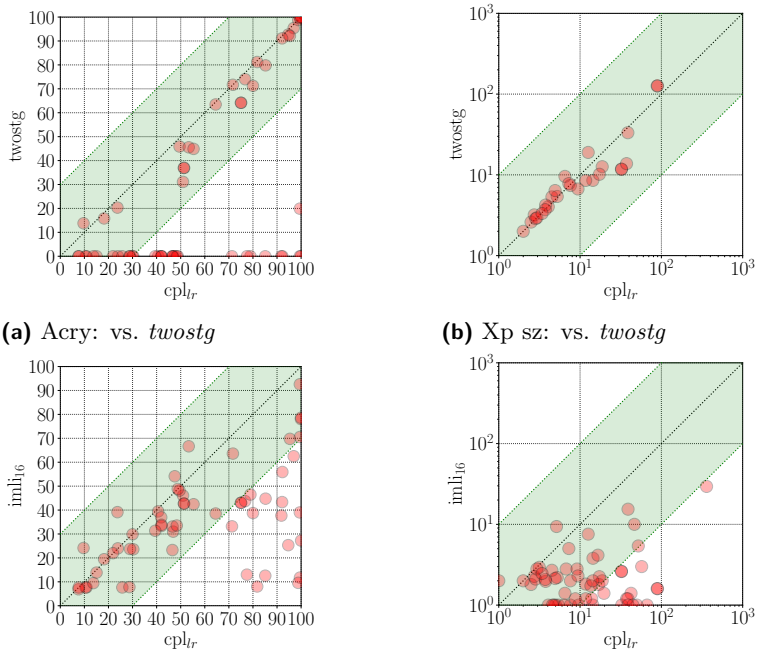
Figures 7 and 8 illustrate the average experimental results across folds regarding scalability, accuracy, model complexity, and explanation size. Since 5-fold cross validation is used, these results for each dataset are obtained from the average of 5 pairs of training and test data. Here, observations similar to those described in Section 5 can be made, i.e. the best scalability and accuracy among selected DS competitors are both demonstrated by *cpl* and cpl_*, $* \in \{l, r, lr, l\lambda_1\}$, while $imli_1$ and $imli_{16}$ show the smallest model complexity and explanation size.



(a) Acry: vs. CN2 **(b)** Xp sz: vs. CN2

(c) Acry: vs. RIPPER **(d)** Xp sz: vs. RIPPER

Figure 9 cpl_{lr} vs. CN2 and RIPPER across folds in terms of accuracy and explanation size.



(a) Acry: vs. *twostg* **(b)** Xp sz: vs. *twostg*

(c) Acry: vs. $imli_{16}$ **(d)** Xp sz: vs. $imli_{16}$

Figure 10 cpl_{lr} vs. $imli_{16}$ and *twostg* Across Folds in terms of accuracy and explanation size.

B Detailed Comparisons Across Folds

In this appendix, we provide a detailed comparison of cpl_{lr} versus other decision set inference approaches across folds.

Figure 9 and Figure 10 detail the comparisons of cpl_{lr} with CN2, RIPPER, $imli_{16}$ and *twostg* in terms of average accuracy and explanation size across folds. As can be seen in Figure 9a, the accuracy of DSes generated by cpl_{lr} is higher than the accuracy of CN2, where the average accuracy is 57.49% and 48.03%, respectively. Additionally, Figure 9b demonstrate that the explanation size of DSes produced by CN2 (81.93 on average) can be two orders of magnitude larger than the explanation size of cpl_{lr} (25.88 on average).

Figure 9c illustrate that the average accuracy in RIPPER is 44.81%, which is 12.68% lower than the accuracy in cpl_{lr}. Although Figure 9d depict that RIPPER is comparable with cpl_{lr} regarding explanation size (29.08 and 25.34 on average respectively), RIPPER is less interpretable as it computes DSes representing only one class.

As can be observed in Figure 10a, the accuracy of *twostg* (29.67% on average) is 27.82% lower than the accuracy in cpl_{lr} while Figure 10b illustrate that the explanation size is comparable between the two approaches. Finally, Figure 10c demonstrate that the accuracy of $imli_{16}$ is 22.02% lower than the accuracy of cpl_{lr} on average. However, as can be seen in Figure 10d, the explanation size of $imli_{16}$ is smaller than the explanation size of cpl_{lr} but $imli_{16}$ generates DSes targeting only a single class, which significantly diminish the interpretability of computed DSes.

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.

Eliminating The Impossible, Whatever Remains Must Be True

Jinqiang Yu[1,4], **Alexey Ignatiev**[1], **Peter J. Stuckey**[1,4], **Nina Narodytska**[2], **Joao Marques-Silva**[3]

[1] Monash University, Melbourne, Australia
[2] VMWare Research, Palo Alto, USA
[3] IRIT, CNRS, Toulouse, France
[4] ARC Training Centre in OPTIMA, Melbourne, Australia
{jinqiang.yu,alexey.ignatiev,peter.stuckey}@monash.edu, nnarodytska@vmware.com, joao.marques-silva@irit.fr

Abstract

The rise of AI methods to make predictions and decisions has led to a pressing need for more explainable artificial intelligence (XAI) methods. One common approach for XAI is to produce a post-hoc explanation, explaining why a black box ML model made a certain prediction. Formal approaches to post-hoc explanations provide succinct reasons for *why* a prediction was made, as well as *why not* another prediction was made. But these approaches assume that features are independent and uniformly distributed. While this means that "why" explanations are correct, they may be longer than required. It also means the "why not" explanations may be suspect as the counterexamples they rely on may not be meaningful. In this paper, we show how one can apply background knowledge to give more succinct "why" formal explanations, that are presumably easier to interpret by humans, and give more accurate "why not" explanations. In addition, we show how to use existing rule induction techniques to efficiently extract background information from a dataset.

1 Introduction

Recent years have witnessed rapid advances in Artificial Intelligence (AI) and Machine Learning (ML) algorithms revolutionizing all aspects of human lives (LeCun, Bengio, and Hinton 2015; ACM 2018). An ever growing range of practical applications of AI and ML, on the one hand, and a number of critical issues observed in modern AI systems (e.g. decision bias (Angwin et al. 2016) and brittleness (Szegedy et al. 2014)), on the other hand, gave rise to the quickly advancing area of theory and practice of Explainable AI (XAI).

Several major approaches to XAI exist. Besides tackling XAI through computing *interpretable* ML models directly (Rudin 2019), or through the use of interpretable models for approximating complex *black-box* ML models (Ribeiro, Singh, and Guestrin 2016), the most prominent approach to XAI is to compute *post-hoc explanations* to ML predictions on demand (Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018). Prior work distinguishes post-hoc (*abductive*) explanations answering a *"why?"* question and (*contrastive*) explanations targeting a *"why not?"* question (Miller 2019). Heuristic approaches to post-hoc explainability are known to suffer from a number of fundamental explanation quality issues (Narodytska et al. 2019; Ignatiev, Narodytska, and Marques-Silva 2019c; Camburu et al. 2019; Ignatiev 2020), including the existence of out-of-distribution attacks (Slack et al. 2020). A promising alternative is formal explainability where explanations are computed as prime implicants of the decision function associated with ML predictions (Shih, Choi, and Darwiche 2018). Formal explanations have also been related with abductive reasoning (Ignatiev, Narodytska, and Marques-Silva 2019a,b).

Although provably correct and minimal, formal explanations have a few limitations. To provide provable correctness guarantees, formal approaches have to take into account the complete feature space assuming that the features are independent and uniformly distributed (Wäldchen et al. 2021). This makes a formal reasoner check all the combinations of feature values, including those that realistically can *never appear* in practice. This leads to unnecessarily long explanations that are hard for a human decision maker to interpret.

Motivated by this limitation, our work focuses on computing both abductive and contrastive formal explanations making use of background knowledge, and makes the following contributions. First, given a training data, we propose an efficient generic approach to extracting background knowledge in the form of highly accurate *if-then* rules, building on a recent formal method for learning decision sets (Ignatiev et al. 2021). Second, we propose a novel approach to computing formal explanations subject to background knowledge, *independent* of the nature of the background knowledge. Third, we prove theoretically that the use of background knowledge positively affects the quality of both abductive and contrastive explanations, thus, helping to build trust in the underlying AI systems. Finally, motivated by (Ignatiev 2020), we argue that background knowledge helps one assess the correctness of heuristic ML explainers more accurately since it blocks impossible combinations of features values.

2 Preliminaries

SAT and MaxSAT. Definitions standard in *propositional satisfiability* (SAT) and *maximum satisfiability* (MaxSAT) solving are assumed (Biere et al. 2021). SAT and MaxSAT formulas are assumed to be propositional. A propositional formula φ is considered to be in *conjunctive normal form* (CNF) if it is a conjunction (logical *"and"*) of clauses, where a *clause* is a disjunction (logical *"or"*) of literals, and a

Edu.	Status	Occup.	Rel.	Sex	Hrs/w	Target
H-School	Married	Sales	Husband	M	40-45	≥ 50k
Bachelors	Married	Sales	Wife	F	≤ 40	≥ 50k
Masters	Married	Prof	Wife	F	≥ 45	≥ 50k
Masters	Married	Prof	Wife	F	≤ 40	≥ 50k
Dropout	Separated	Service	Not-in-fam	M	≤ 40	< 50k
Dropout	Never	B-Collar	Unmarried	M	≥ 45	≥ 50k

Table 1: Several examples extracted from *adult* dataset.

literal is either a Boolean variable b or its *negation* $\neg b$. Whenever convenient, a clause is treated as a set of literals. In the context of *unsatisfiable* formulas, i.e. when there is no way to assign the values 0 or 1 to all the variables of formula φ s.t. φ evaluates to 1, the maximum satisfiability problem is to find a truth assignment that maximizes the number of satisfied clauses. Hereinafter, we use a variant of MaxSAT called Partial (Unweighted) MaxSAT (Biere et al. 2021, Chapters 23 and 24). The formula φ in Partial (Unweighted) MaxSAT is a conjunction of *hard* clauses $\mathcal{H}$, which must be satisfied, and *soft* clauses $\mathcal{S}$, which represent a preference to satisfy them, i.e. $\varphi = \mathcal{H} \land \mathcal{S}$. The aim is to find a truth assignment that satisfies all hard clauses while maximizing the total number of satisfied soft clauses.

Hereinafter, propositional formulas are applied for reasoning about the behavior of machine learning models used as well as to represent background knowledge constraints.

Classification Problems. Classification problems consider a set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_k\}$, and a set of features $\mathcal{F} = \{1, \ldots, m\}$. The value of each feature $i \in \mathcal{F}$ is taken from a domain $\mathcal{D}_i$, which can be integer, real-valued or Boolean. Therefore, the complete feature space is defined as $\mathbb{F} \triangleq \prod_{i=1}^{m} \mathcal{D}_i$. A concrete point in feature space is represented by $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, where each $v_i \in \mathbf{v}$ is a constant taken by feature $i \in \mathcal{F}$. An *instance* or *example* is denoted by a specific point $\mathbf{v} \in \mathbb{F}$ in feature space and its corresponding class $c \in \mathcal{K}$, i.e. a pair $(\mathbf{v}, c)$ represents an instance. Moreover, the notation $\mathbf{x} = (x_1, \ldots, x_m)$ denotes an arbitrary point in feature space, where each $x_i \in \mathbf{x}$ is a variable taking values from its corresponding domain $\mathcal{D}_i$ and representing feature $i \in \mathcal{F}$.

A classifier defines a classification function $\tau : \mathbb{F} \to \mathcal{K}$. There are many ways to learn classifiers for a given dataset. In this paper, we consider: *decision lists* (DLs) (Rivest 1987; Clark and Niblett 1989), *boosted trees* (BTs) (Friedman 2001; Chen and Guestrin 2016), and *binarized neural networks* (BNNs) (Hubara et al. 2016).

Example 1. *Consider the data shown in Table 1. It represents a snapshot of instances taken from a simplified version*[1] *of the* adult *dataset (Kohavi 1996). Figure 1 illustrates DL and BT models trained for this dataset. Observe that for instance* $\mathbf{v} = \{$*Education = HighSchool, Status = Married, Occupation = Sales, Relationship = Husband, Sex = Male, Hours/w = 40 to 45*$\}$ *from Table 1, rule* R_2 *in the DL in Figure 1a predicts* $\geq 50k$*. Similarly, the sum of the weights (*0.1063*,* 0.0707 *and* -0.0128 *in the* 3 *trees, respectively) for prediction* $\geq 50k$ *is positive (*0.1642*) in the BT in Figure 1b, and so the BT model also predicts* $\geq 50k$ *for the aforementioned instance* $\mathbf{v}$*.*

[1]For simplicity, the running example used throughout the text will correspond to a *simplified* version of the *adult* dataset (Kohavi 1996), where some of the features are dropped.

Interpretability and Explanations. Interpretability is not formally defined since it is a subjective concept (Lipton 2018). In this paper, we define interpretability as the conciseness of the computed explanations for an ML model to justify a provided prediction. The definition of explanation for an ML model is built on earlier work (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019a; Darwiche and Hirth 2020; Audemard, Koriche, and Marquis 2020; Marques-Silva and Ignatiev 2022), where explanations are equated with *abductive explanations* (AXps), which are subset-minimal sets of features sufficing to explain a given ML prediction. Concretely, given an instance $\mathbf{v} \in \mathbb{F}$ and a computed prediction $c \in \mathcal{K}$, i.e. $\tau(\mathbf{v}) = c$, an AXp is a subset-minimal set of features $\mathcal{X} \subseteq \mathcal{F}$, such that

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

Abductive explanations are also prime implicants of the decision predicate τ_c and hence a *prime implicant* (PI) explanation is another name for an AXp.

Example 2. *Consider the models in Figure 1 and instance* $\mathbf{v}$ *from Example 1. By examining the DL model, specifying Education = HighSchool, Status = Married, Occupation = Sales, and Relationship = Husband guarantees that any compatible instance is classified by* R_2 *independent of the values of other features, i.e. Sex and Hours/w. Similarly, the prediction of an instance is guaranteed to be* $\geq 50k$ *in Figure 1b as long as the feature values above are used, since the sum of weights is promised to be* $0.1063 + 0.0707 + -0.0128 = 0.1642$ *for class* $\geq 50k$*. Therefore, the (only) AXp* $\mathcal{X}$ *for the prediction of* $\mathbf{v}$ *is* $\{$*Education, Status, Occupation, Relationship*$\}$ *in both models.*

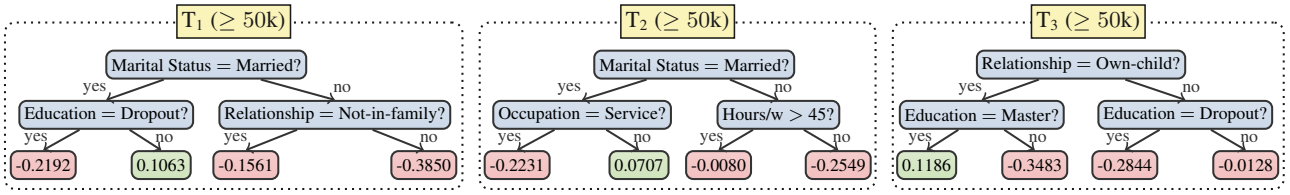
We also consider *contrastive explanations* (CXps) defined as subset-minimal sets of features that are necessary to change the prediction if the features of a CXp are allowed to take arbitrary values from their domains. Formally and following (Ignatiev et al. 2020), a CXp for prediction $\tau(\mathbf{v}) = c$ is defined as a minimal subset $\mathcal{Y} \subseteq \mathcal{F}$ s.t.

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

Example 3. *Consider the setup of Example 2. Given either model,* $\mathcal{Y} = \{$*Occupation*$\}$ *is a CXp for instance* $\mathbf{v}$ *because the prediction for* $\mathbf{v}$ *can be changed if feature 'Occupation' is allowed to take another value, e.g. if the value is changed to 'Service'. Similarly, changing the value of feature 'Occupation' to 'Service' triggers that the weights in the* 3 *trees become* 0.1063*,* -0.2231 *and* -0.0128*. Therefore, the total weight is* -0.0982*, i.e. the prediction is changed. By further examining the two models, other subsets of features can be identified as CXps for* $\mathbf{v}$*. The set of CXps is* $\mathbb{Y} =$

R_0:	IF	Education = Dropout	THEN	Target $<$ 50k
R_1:	ELSE IF	Occupation = Service	THEN	Target $<$ 50k
R_2:	ELSE IF	Status = Married $\wedge$ Relationship = Husband	THEN	Target $\geq$ 50k
R_3:	ELSE IF	Status = Married $\wedge$ Relationship = Wife	THEN	Target $\geq$ 50k
R_{DEF}:	ELSE		THEN	Target $<$ 50k

(a) Decision list.



(b) Boosted tree (Chen and Guestrin 2016) consisting of 3 trees with the depth of each tree at most 2.

Figure 1: Example DL and BT models trained on the well-known *adult* classification dataset.

$\{\{$*Education*$\}$, $\{$*Status*$\}$, $\{$*Occupation*$\}$, $\{$*Relationship*$\}\}$, *while the set of AXps demonstrated in Example 2 is* $\mathbb{X} = \{\{$*Education, Status, Occupation, Relationship*$\}\}$.

(Ignatiev et al. 2020) builds on the seminal work of Reiter (Reiter 1987) to establish a minimal hitting set (MHS) duality relationship between AXps and CXps, i.e. each CXp *minimally hit* every AXp, and vice-versa.

Example 4. *Observe how the MHS duality holds for the sets of AXps* $\mathbb{X}$ *and the set of CXps* $\mathbb{Y}$ *shown in Example 3. The only AXp minimally hits all the CXps and vice versa.*

There is a growing body of recent work on formal explanations (Marques-Silva et al. 2020, 2021; Izza and Marques-Silva 2021; Ignatiev and Marques-Silva 2021; Arenas et al. 2021; Wäldchen et al. 2021; Darwiche and Marquis 2021; Malfa et al. 2021; Boumazouza et al. 2021; Blanc, Lange, and Tan 2021; Izza, Ignatiev, and Marques-Silva 2022; Gorji and Rubin 2022; Ignatiev et al. 2022; Huang et al. 2022; Marques-Silva and Ignatiev 2022; Amgoud and Ben-Naim 2022; Ferreira et al. 2022; Arenas et al. 2022).

3 Extracting Background Knowledge

Recent work (Gorji and Rubin 2022) argues that background knowledge is helpful in the context of formal explanations. If identified, background knowledge may help forbid some of the combinations of feature values that would otherwise have to be taken into account by a formal reasoner, thus, slowing the reasoner down and making the explanations unnecessarily long. But the question of how such knowledge can be obtained in an automated way remains open.

Example 5. *Assume that Table 1 represents* trustable *information. The following two rules can be extracted:*

- IF *Relationship* $=$ *Husband* THEN *Status* $=$ *Married*
- IF *Relationship* $=$ *Wife* THEN *Status* $=$ *Married*

These rules may be used to discard feature Status *when computing explanations as long as* Relationship *equals either* Husband *or* Wife *because of the implications identified.*

We describe the MaxSAT approach to extract background knowledge representing implicit relations between features of a dataset if the dataset is assumed to be trustable. It builds on the recent two-stage approach (Ignatiev et al. 2021) to learning smallest size decision sets. Concretely, we apply the first stage of (Ignatiev et al. 2021) which enumerates individual decision rules given a dataset, using MaxSAT.

Without diving into the details, the idea of (Ignatiev et al. 2021) is as follows. Given training data $\mathcal{E}$ and target class $c \in \mathcal{K}$, a MaxSAT solver is invoked multiple times, each producing a unique subset-minimal (irreducible) rule in the form of *"IF* antecedent *THEN* prediction c*"*, where the antecedent is a set of feature values. The MaxSAT solver is fed with various CNF constraints and an objective function targeting rule size minimization. The approach also detects and blocks *symmetric rules*, i.e. those that do not contribute new information to the rule-based representation of class $c \in \mathcal{K}$.

We can modify the MaxSAT approach outlined above to learning background knowledge in the form of decision rules, i.e. identifying the dependency of a feature $i \in \mathcal{F}$ on other features $j \in \mathcal{F} \setminus \{i\}$. For this, we need to discard the prediction column from the dataset $\mathcal{E}$ and instead focus on a feature $i \in \mathcal{F}$, consider some of its values $v_{ij} \in \mathcal{D}_i$ and "pretend" to compute decision rules for a "fake class" $x_i = v_{ij}$. Thanks to the properties of the approach of (Ignatiev et al. 2021), all the rules computed are guaranteed to be subset-minimal and to respect training data $\mathcal{E}$. Once all the rules for feature $i \in \mathcal{F}$ and value $v_{ij} \in \mathcal{D}_i$ are computed, the same exercise can be repeated for all the values in $\mathcal{D}_i \setminus \{v_{ij}\}$ but, more importantly, all the other features.

Example 6. *Consider again Table 1. The two rules shown in Example 5 are computed by our rule learning approach if we focus on feature* Status*. The following two rules can be extracted when feature* Relationship *is focused on instead:*

- IF *Status* $=$ *Married* $\wedge$ *Sex* $=$ *M.* THEN *Rel.* $=$ *Husband*
- IF *Status* $=$ *Married* $\wedge$ *Sex* $=$ *F.* THEN *Rel.* $=$ *Wife*

Duplicate Rules. As mentioned above, all rules generated with the MaxSAT approach of (Ignatiev et al. 2021) are guaranteed to be subset-minimal. Furthermore, none of

Algorithm 1: Rule Extraction

Input: Dataset $\mathcal{E}$, extraction limit λ
Output: Rules φ

$\mathcal{E}_f, \mathcal{F} \leftarrow \texttt{DropClass}(\mathcal{E}), \texttt{ExtractFeatures}(\mathcal{E})$
$\varphi, B \leftarrow \emptyset, \emptyset$ *# to extract and block rules, resp.*
for $i \in \mathcal{F}$ **do**
  for rule $\in \texttt{EnumerateRules}(\mathcal{E}_f, i, B)$ **do**
    if $\texttt{limit}(\text{rule}, \lambda)$ is $true$ **then**
      break
    $\varphi \leftarrow \varphi \cup \text{rule}$
  $B \leftarrow \varphi$
return φ

the rules enumerated is symmetric with another rule if considered in the *if-then* form. However, when the rules are treated as clauses, i.e. a disjunction of Boolean literals, some rules may duplicate the other. Indeed, recall that a rule of size $k \leq |\mathcal{F}|$ is of the form $(f_1 \land \ldots \land f_{k-1}) \rightarrow f_k$ where each f_i represents a literal $(x_i = v_{i_{j_i}})$, $i \in \mathcal{F}$ and $v_{i_{j_i}} \in \mathcal{D}_i$. Clearly, this same proposition can be equivalently represented as a clause $(\neg f_1 \lor \ldots \lor \neg f_{k-1} \lor f_k)$. Observe that the same clause can be used to represent another rule $(f_1 \land \ldots \land f_{k-2} \land \neg f_k) \rightarrow \neg f_{k-1}$, which can thus be seen as symmetric in the *clausal form*. This way, a clause of size k represents k possible rules. However, due to symmetry, it suffices to compute only one of them and block all the "duplicates" by adding its clausal representation to the MaxSAT solver. This novel symmetry breaking mechanism significantly improves the scalability of our approach.

Example 7. *Consider a rule* $\{$ *IF Status* $=$ *Married* $\land$ *Sex* $=$ *Male THEN Relationship* $=$ *Husband* $\}$ *computed for feature Relationship. This rule is represented as a clause*

$(\textit{Status} \neq \textit{Married} \lor \textit{Sex} \neq \textit{M.} \lor \textit{Relationship} = \textit{Husband})$

There are two duplicates in other contexts:

- IF *Status* $=$ *Married* $\land$ *Rel.* $\neq$ *Husband* THEN *Sex* = *F.*
- IF *Sex* $=$ *M.* $\land$ *Rel.* $\neq$ *Husband* THEN *Status* $\neq$ *Married*

Extraction limit. Even if we remove duplicate rules, there can still be many rules to enumerate for an entire dataset. Many of them will never, or only rarely, contribute to reducing the size of explanations of the classifier. Extracting these *low value* rules is unnecessary in the rule extracting process. In practice, we noticed that some rules (e.g. long rules or rules having a low support) never contribute to explanation reduction. Hence, we apply an *extraction limit* to prevent exhaustive rule enumeration, which enables us to focus only on *most useful* rules. Here, extraction limit can be a restriction of a user's choice, e.g. a total extraction runtime, a limit on the number of rules, rule support or size, etc.

A high-level view on the overall rule extraction approach is provided in Algorithm 1. Initially, the class column from the original dataset $\mathcal{E}$ is dropped and the features $\mathcal{F}$ in $\mathcal{E}$ are acquired. For each feature $i \in \mathcal{F}$, the algorithm enumerates the decision rules targeting i until the extraction limit is met or no more rules can be found. The rules previously learned are blocked in the clausal form to avoid computing their duplicates. Finally, the algorithm returns the rules extracted.

Our approach computes only rules that are perfectly consistent with the *known* data, which makes sense if the data is extensive and trustworthy. In practical settings, however, some of the data are unknown, i.e. the rules computed may be inconsistent with unseen parts of the feature space $\mathbb{F}$. If testing and validation data are available, then the rules can be tested against them. We can then exclude the rules that are not *sufficiently* accurate wrt. test and/or validation data.

4 Knowledge-Assisted Explanations

We assume the obtained background knowledge can be represented as a formula φ. Under that assumption, (Gorji and Rubin 2022) proposes to compute AXps for positive predictions of a Boolean classifier $\tau : \mathbb{F} \rightarrow \{0, 1\}$ taking into account constraints φ. Observe that formula φ can be seen as representing a predicate $\varphi : \mathbb{F} \rightarrow \{0, 1\}$, the truth value of which, i.e. $\varphi(\mathbf{x})$, can be tested for an instance $\mathbf{v} \in \mathbb{F}$. The approach of (Gorji and Rubin 2022) relies on *compiling* a Boolean classifier $\tau(\mathbf{x})$ into a *tractable* representation (Shih, Choi, and Darwiche 2018) and proposes to compute an AXp $\mathcal{X} \subseteq \mathcal{F}$ for prediction $\tau(\mathbf{v}) = 1$, $\mathbf{v} \in \mathbb{F}$, subject to constraints φ as a prime implicant of $[\varphi(\mathbf{x}) \rightarrow \tau(\mathbf{x}) = 1]$.

Observe that we can generalize this idea to the context of computing formal AXps and CXps for *any classifier* that admits a logical representation suitable for making reasoning oracle calls wrt. formulas (1) and (2). Namely, given a prediction $\tau(\mathbf{x}) = c$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, an abductive explanation $\mathcal{X} \subseteq \mathcal{F}$ subject to background knowledge φ is such that:

$$\forall(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{j \in \mathcal{X}} (x_j = v_j) \rightarrow [\varphi(\mathbf{x}) \rightarrow (\tau(\mathbf{x}) = c)] \quad (3)$$

More importantly, the same can be done with respect to contrastive explanations. Given a prediction $\tau(\mathbf{x}) = c$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, a contrastive explanation $\mathcal{Y} \subseteq \mathcal{F}$ subject to background knowledge φ is such that the following holds:

$$\exists(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i) \land [\varphi(\mathbf{x}) \land (\tau(\mathbf{x}) \neq c)] \quad (4)$$

Note that (3) and (4) are the negation of each other, i.e. a subset of features $\mathcal{Y} \subseteq \mathcal{F}$ *is* a CXp for prediction $\tau(\mathbf{x}) = c$ iff $\mathcal{X} = \mathcal{F} \setminus \mathcal{Y}$ *is not* an AXp. This means when dealing with either AXps or CXps, one can reason about (un)satisfiability of formula $\bigwedge_{i \in \mathcal{Z}} (x_i = v_i) \land [\varphi(\mathbf{x}) \land (\tau(\mathbf{x}) \neq c)]$ with $\mathcal{Z}$ being either $\mathcal{X}$ or $\mathcal{F} \setminus \mathcal{Y}$ depending on the kind of target explanation. Therefore, if background knowledge φ is a *conjunction* of constraints, e.g. rules, we can integrate them in the existing formal explainability setup of (Ignatiev, Narodytska, and Marques-Silva 2019a) with no additional overhead.

Following (Ignatiev et al. 2020) and applying the same arguments, an immediate observation to make is that in the presence of background knowledge, the minimal hitting set duality between AXps and CXps holds:

Proposition 1. *Let* $\mathbf{v} \in \mathbb{F}$ *be an instance such that* $\tau(\mathbf{v}) = c$, $c \in \mathcal{K}$, *and background knowledge* φ *is compatible with* $\mathbf{v}$. *Then any AXp* $\mathcal{X}$ *for prediction* $\tau(\mathbf{v}) = c$ *minimally hits any CXp for this prediction, and vice versa.*

Proposition 1 enables us to apply algorithms originally studied in the context of over-constrained systems (Bailey

IF	Status = Married	THEN	Target $\geq$ 50k
ELSE IF	Sex = M $\wedge$ Rel. $\neq$ Husband	THEN	Target $<$ 50k
ELSE		THEN	Target $\geq$ 50k

Figure 2: A DL for selected examples of *adult* dataset.

and Stuckey 2005; Liffiton and Sakallah 2008; Belov, Lynce, and Marques-Silva 2012; Marques-Silva et al. 2013; Mencia, Previti, and Marques-Silva 2015; Ignatiev et al. 2015; Liffiton et al. 2016; Bendík, Cerná, and Benes 2018) to explore all AXps and CXps for ML predictions. In particular, the existing explanation extraction and enumeration algorithms (Ignatiev, Narodytska, and Marques-Silva 2019a; Ignatiev et al. 2020) can be readily applied by taking into account background knowledge, as shown in (3) and (4).

Gorji et al. (Gorji and Rubin 2022) proved that subset-minimal AXps computed subject to additional constraints for Boolean classifiers tend to be smaller than their unconstrained "counterparts". The rationale is that when additional constraints are imposed, some of the features $i \in \mathcal{F}$ may be dropped from an AXp because the equalities $x_i = v_i$ falsify the constraints, i.e. they represent data instances that are *not permitted* by the constraints. Based on their result, the following generalization can be proved to hold:

Proposition 2. *Consider $\mathbf{v} \in \mathbb{F}$ such that $\tau(\mathbf{v}) = c$, $c \in \mathcal{K}$, and background knowledge φ is compatible with $\mathbf{v}$. Then for any subset-minimal AXp $\mathcal{X} \subseteq \mathcal{F}$ for prediction $\tau(\mathbf{v}) = c$, there is a subset-minimal AXp $\mathcal{X}' \subseteq \mathcal{F}$ for $\tau(\mathbf{v}) = c$ subject to background knowledge φ such that $\mathcal{X}' \subseteq \mathcal{X}$.*

Remark 1. *The opposite, i.e. that given AXp $\mathcal{X}'$ subject to background knowledge φ, there must exist a subset-minimal AXp $\mathcal{X} \supseteq \mathcal{X}'$ without knowledge φ, in general does not hold.*

Example 8. *Consider the DL in Figure 2. Given an instance $\mathbf{v} = \{$Education = Dropout, Status = Separated, Occupation = Service, Relationship = Not-in-Family, Sex = Male, Hours/w = $\leq 40\}$, the prediction enforced by R_1 is $\leq 50k$ and the AXp is $\mathcal{X} = \{$Status, Relationship, Sex$\}$. Let a single constraint φ be $\{$Sex = Male $\wedge$ Relationship = Not-in-Family $\rightarrow$ Status = Separated$\}$. Feature 'Status' can be dropped because the constraint φ ensures it to be set to the "right value" if the other two features are set as required, and hence R_0 is guaranteed not to fire. Thus, we can compute a smaller AXp $\mathcal{X}' = \{$ Relationship, Sex $\}$.*

While using background knowledge φ pays off in terms of interpretability of abductive explanations, this cannot be said wrt. contrastive explanations. Surprisingly and as the following result proves, background knowledge can only contribute to increase the size of contrastive explanations.

Proposition 3. *Consider $\mathbf{v} \in \mathbb{F}$ such that $\tau(\mathbf{v}) = c$, $c \in \mathcal{K}$, and background knowledge φ is compatible with $\mathbf{v}$. Then for any subset-minimal CXp $\mathcal{Y}' \subseteq \mathcal{F}$ for prediction $\tau(\mathbf{v}) = c$ subject to knowledge φ, there is subset-minimal CXp $\mathcal{Y} \subseteq \mathcal{F}$ is a CXp for prediction $\tau(\mathbf{v}) = c$ such that $\mathcal{Y}' \supseteq \mathcal{Y}$.*

Remark 2. *The reverse direction: given a CXp $\mathcal{Y}$ generated without using background knowledge, there must exist a CXp $\mathcal{Y}' \supseteq \mathcal{Y}$ using background knowledge, does not hold.*

One may wonder then why background knowledge is useful when computing CXps. The reason is that the CXps generated using background knowledge are *correct* under the assumption that the background knowledge describes the actual relationships between features. On the contrary, CXps generated without using background knowledge are *only correct* under the assumption that every combination of feature values is possible, i.e. all features are independent and their values are uniformly distributed across the feature space, which hardly ever occurs in practice.

Example 9. *Consider the setup of Example 8. Observe that a CXp for the prediction is $\mathcal{Y} = \{$ Status $\}$. Its correctness relies on the fact that changing Status to Married changes the prediction to $\geq 50k$. But given the background knowledge φ, this is clearly erroneous. Since the other fixed features in instance $\mathbf{v}$ are $\{$ Sex = Male, Education = Dropout, Occupation = Service, Relationship = Not-in-family, Hours/w = ≤ 40 $\}$, the modification is inconsistent with knowledge φ. This demonstrates the weakness of CXps as they rely on the assumption that any tuple of feature values in $\mathbb{F}$ is possible. Applying constraint φ leads to a larger CXp $\mathcal{Y}' \triangleq \{$ Status, Relationship $\}$. This clearly does allow the prediction to change and it is compatible with φ.*

5 Experimental Results

Here we provide a summary of the experimental results. The full experimental analysis can be found in (Yu et al. 2022).

Setup and Prototype Implementation. The experiments were run on an Intel Xeon 8260 CPU running Ubuntu 20.04.2 LTS, with a memory limit of 8 GByte. A prototype of the approach to extracting background knowledge and computing AXps and CXps applying background knowledge was developed as a set of Python scripts.[2] The implementation of knowledge extraction builds on (Ignatiev et al. 2021) and extensively uses modern SAT technology (Ignatiev, Morgado, and Marques-Silva 2018, 2019). Also, explanation enumeration for DLs and BNNs makes use of the SAT technology while for BTs we apply satisfiability modulo theories (SMT) solvers (Gario and Micheli 2015).

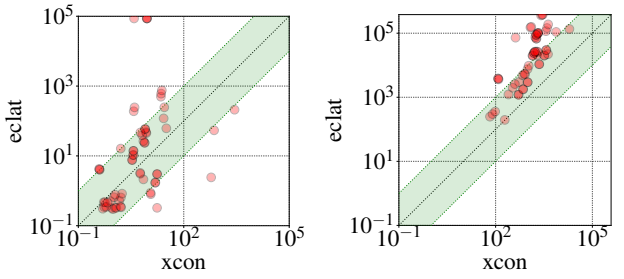
A few words should be said about the competition considered. First, we compared our knowledge extraction approach to the Apriori and Eclat algorithms (Agrawal and Srikant 1994; Zaki et al. 1997). In our experiments, these algorithms behave almost identically with Eclat solving one more instance; as a result, we use Eclat as the best competitor. When running Eclat, we apply the same setup as used for our approach. Finally, heuristic explainers are represented by LIME (Ribeiro, Singh, and Guestrin 2016), SHAP (Lundberg and Lee 2017), and Anchor (Ribeiro, Singh, and Guestrin 2018) in their default configurations.

Datasets. The benchmarks considered include a selection of datasets publicly available from UCI Machine Learning Repository (Dua and Graff 2017) and Penn Machine Learning Benchmarks (Olson et al. 2017). In total, 24 datasets

[2] https://github.com/jinqiang-yu/xcon

	rule_1	rule_2	rule_3	rule_4	rule_5	rule_{all}
Accuracy (%)	99.22	99.35	99.29	99.16	99.06	99.08

Table 2: Average accuracy of individual rules over test data.



(a) Runtime comparison. (b) Number of rules comparison.

Figure 3: *Eclat* vs. *xcon* – performance comparison.

are selected. Whenever applicable, numeric features in all benchmarks were quantized into 4, 5, or 6 intervals. Therefore, the total number of quantized datasets considered is 62.

Machine Learning Models. We used CN2 (Clark and Niblett 1989) to train the DL models studied. BTs were computed by XGBoost (Ribeiro, Singh, and Guestrin 2016) s.t. each class is represented by 25 trees of depth 3. BNNs were trained by PyTorch (Paszke et al. 2019). Three configurations of hidden layers[3] were used when training BNNs to achieve sufficient test accuracy. As usual, each of the 62 datasets was randomly split into 80% of training and 20% of test data, respectively. The average test accuracy of the DL, BT, and BNN models was 76.47%, 76.17%, and 80.31%.

5.1 Knowledge Extraction

Knowledge extraction is tested using 5-fold cross validation. The average accuracy of each rule is measured as the proportion of test instances that violate that rule, averaged over the folds. We consider rules of length 1 to 5, and extracting all possible rules (*all*). Table 2 compares the average accuracy of the background knowledge extracted. The average accuracy of *all rules* and $rules_s$, $s \in \{1, \ldots, 5\}$, exceeds 99%.

Additionally, we compare the overall performance of exhaustive rule extraction against rule extraction with the size limit 5. On average, exhaustive (limited, resp.) rule enumeration ends up computing 2116.29 (1964.24, resp.) rules per dataset. We also compare *xcon* against Eclat in terms of the overall performance. For a fair comparison, we set Eclat to extract only rules of confidence 100%, i.e. all the rules extracted are perfectly consistent with the *known* data. Figure 3a compares the runtime of rule extraction and the number of extracted rules of size up to 5 across all 5 train-test pairs. Clearly *xcon* can extract rules on par with Eclat, and is able to tackle all 62 datasets, while Eclat fails on 4.

[3]The 3 configurations are classified as *small*, *medium* and *large*. The size of the hidden layers of these 3 configurations is as follows: large: (64, 32, 24, 2); medium: (32, 16, 8, 2); small: (10, 5, 5, 2).

Figure 3b shows the number of extracted rules for the 58 datasets solved by both approaches. Eclat always extracts more rules because it uses a less expressive language, i.e. it cannot use the *negation* of feature literals.

5.2 Knowledge-Assisted Explanations

Let us evaluate the benefit of computing formal explations using background knowledge, once more restricting to background rules of size at most 5. For each of the 62 datasets, we selected all *test* instances and enumerated 20 *smallest size* AXps or CXps for each such instance. Let $xcon_*$ s.t. $* \in \{dl, bt, bnn\}$ denote the approaches for formally explaining DL, BT, and BNN models, and $xcon^r_*$ is the approach taking into account background knowledge.

Scalability. Figures 4a and 4b compare the average runtime for computing a single AXp or CXp for the DL models. Clearly background knowledge quickens AXp explanation, while slowing CXp explanation. The slowdown for CXps is caused by the fact that the CXps get much larger, which requires a larger number of oracle calls. For BTs and BNNs the use of background knowledge neither substantially improves nor degrades the computation of AXps or CXps.

Explanation Quality. The change of smallest size of AXps and CXps in an instance for DLs is shown in Figure 4c, and 4d. Clearly background knowledge substantially reduces the size of AXps, and substantially increases the size of CXps. Similar results arise for BTs and BNNs. The experiments above were repeated using background knowledge extracted with Eclat. The results are similar.

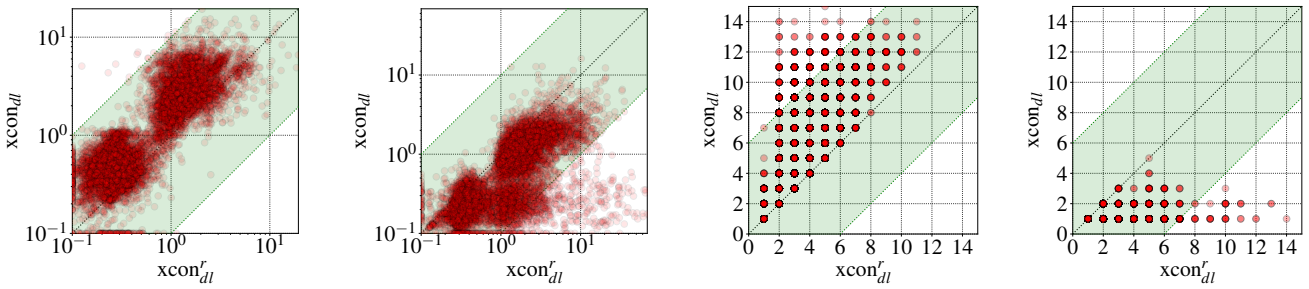
5.3 Formal vs. Heuristic Explanations

Following (Ignatiev, Narodytska, and Marques-Silva 2019a; Narodytska et al. 2019; Ignatiev 2020), we apply formal explanations to assess the runtime and explanation quality for the heuristic approaches LIME, SHAP, and Anchor. The idea is to show the importance of trustable background knowledge when targeting a more accurate quality assessment.

Scalability. Table 3 illustrates the runtime of a single explanation extraction for a data instance across the 62 datasets performed by LIME, SHAP, Anchor, $xcon_*$, and $xcon^r_*$. Here, $xcon^r_*$ and $xcon_*$ represent the proposed approach to computing AXps or CXps with/without background knowledge, s.t. $* \in \{axp, cxp\}$. Observe that both $xcon_*$ and $xcon^r_*$ outperform LIME and Anchor for all the 3 models, explaining a data instance in a fraction of a second. LIME and Anchor are 1-2 orders of magnitude slower for DL and BNN models, while LIME outweighs Anchor when generating explanations for BTs. The worst performance for DL and BNN models is demonstrated by SHAP while, surprisingly, SHAP outperforms the other competitors for BTs models.

Correctness. The average correctness of computed explanations is shown in Table 4. [4] Here, an explanation is said to be correct if it answers a *"why"* question and satisfies (1) (or (3) in the presence of background knowledge) or it answers a *"why not"* question and satisfies (2) (or (4) in the presence

[4]LIME/SHAP assign weights to *all the features*. We use only those whose weight contributes to the decision made based on sign.



(a) AXp runtime for DLs. (b) CXp runtime for DLs. (c) AXp size for DLs. (d) CXp size for DLs.

Figure 4: Impact of *xcon* rules on runtime (ms) and explanation size for DLs.

Model	Runtime per explanation (ms)						
	xcon$_{axp}$	xcon$^r_{axp}$	xcon$_{cxp}$	xcon$^r_{cxp}$	LIME	SHAP	Anchor
DL	2	1	1	2	3755	42 555	3800
BT	80	82	97	151	98	6	351
BNN	196	152	179	199	15 607	183 058	11 384

Table 3: Average runtime per explanation.

Explainer	Correctness (%)					
	Without knowledge			With knowledge		
	DL	BT	BNN	DL	BT	BNN
LIME	6.06	38.26	8.2	31.06	60.63	47.88
SHAP	49.47	72.89	58.89	91.72	93.75	95.0
Anchor	24.03	13.85	6.57	73.85	73.0	70.1

Table 4: Average correctness of LIME, SHAP and Anchor.

of background knowledge). Table 4 shows that the average correctness is higher when background knowledge is applied as the number of features required in a minimal correct explanation answering a *"why"* question can drop, which is demonstrated in Section 5.2. However, heuristic approaches are not able to achieve 100% correctness.

Heuristic explainers consistently demonstrate low correctness when no background knowledge is applied, which confirms the earlier results of (Ignatiev 2020). This changes dramatically when we apply the background knowledge because some of the counterexamples invalidating heuristic explanations are forbidden by the knowledge extracted. Assuming that this knowledge is valid, these correctness results better reflect the reality and so are more trustable.

6 Related Work

Many methods for extracting knowledge from a dataset of rules exist (Hipp, Güntzer, and Nakhaeizadeh 2000; Zhang and Zhang 2002; Agrawal and Srikant 1994; Zaki et al. 1997; Izza et al. 2020; Belaid, Bessiere, and Lazaar 2019). For use as background knowledge, we aim at very high confidence in the rules, ideally they should be *completely* valid for the feature space. Traditional rule mining is more interested in rules with high support and less focused on validity, although it can be adapted to this case (see our experimental results above). Although the explanation methods we apply in the presence of background knowledge are agnostic about where it comes from, the motivation for our rule extraction method is twofold: (1) the rules are computed in a clausal form and (2) their high quality is guaranteed by the use of the strict optimization problem formulation.

The most prominent approaches to post-hoc explainability are of heuristic nature (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Ribeiro, Singh, and Guestrin 2018) and based on sampling in the vicinity of the instances being explained. None of these approaches can handle background knowledge. Approaches to formal explainability are represented by compilation of classifiers into tractable representations (Shih, Choi, and Darwiche 2018) and reasoning-based explanation approaches (Ignatiev, Narodytska, and Marques-Silva 2019a; Marques-Silva and Ignatiev 2022). The closest related work is (Gorji and Rubin 2022). Based on compilation of a binary classifier into a binary decision diagram (BDD), it conjoins concocted background knowledge to give more succinct *"why"* explanations for the classifier. This approach is restricted to much smaller examples than we consider here, since the compilation of a classifier into a BDD tends to explode with the feature space. The SAT and SMT based approaches to explanation we use are far more scalable. Finally, we consider a much broader class of classifiers, and also examine *"why not"* explanations and how they can be improved by using background knowledge.

7 Conclusions

Using background knowledge is highly advantageous for producing formal explanations of machine learning models. For abductive explanations (AXps), the use of background knowledge substantially shortens explanations, making them easier to understand, *and* improves the speed of producing explanations. For contrastive explanations (CXps), while the background knowledge lengthens them and may increase the time required to generate an explanation, the resulting explanations are far more useful since they do not rely on the (usually unsupportable) assumption that all tuples in the feature space are possible. Furthermore and as this paper shows, background knowledge can be applied in the context of heuristic explanations when an accurate analysis of their correctness is required.

Acknowledgments

This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009. This work was also partially supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement no. ANR-19-PI3A-0004, and by the H2020-ICT38 project COALA "Cognitive Assisted agile manufacturing for a Labor force supported by trustworthy Artificial intelligence".

References

ACM. 2018. Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award. http://tiny.cc/9plzpz. Accessed: 2022-07-08.

Agrawal, R.; and Srikant, R. 1994. Fast algorithms for mining association rules. In *VLDB*, 487–499.

Amgoud, L.; and Ben-Naim, J. 2022. Axiomatic Foundations of Explainability. In Raedt, L. D., ed., *IJCAI*, 636–642.

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. 2016. Machine Bias. http://tiny.cc/dd7mjz.

Arenas, M.; Baez, D.; Barceló, P.; Pérez, J.; and Subercaseaux, B. 2021. Foundations of Symbolic Languages for Model Interpretability. In *NeurIPS*.

Arenas, M.; Barceló, P.; Romero, M.; and Subercaseaux, B. 2022. On Computing Probabilistic Explanations for Decision Trees. *CoRR*, abs/2207.12213.

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On Tractable XAI Queries based on Compiled Representations. In *KR*, 838–849.

Bailey, J.; and Stuckey, P. J. 2005. Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. In *PADL*, 174–186.

Belaid, M.; Bessiere, C.; and Lazaar, N. 2019. Constraint Programming for Association Rules. In Berger-Wolf, T. Y.; and Chawla, N. V., eds., *SIAM*, 127–135.

Belov, A.; Lynce, I.; and Marques-Silva, J. 2012. Towards efficient MUS extraction. *AI Commun.*, 25(2): 97–116.

Bendík, J.; Cerná, I.; and Benes, N. 2018. Recursive Online Enumeration of All Minimal Unsatisfiable Subsets. In *ATVA*, 143–159.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability*. IOS Press. ISBN 978-1-64368-160-3.

Blanc, G.; Lange, J.; and Tan, L. 2021. Provably efficient, succinct, and precise explanations. In *NeurIPS*.

Boumazouza, R.; Alili, F. C.; Mazure, B.; and Tabia, K. 2021. ASTERYX: A model-Agnostic SaT-basEd appRoach for sYmbolic and score-based eXplanations. In *CIKM*, 120–129.

Camburu, O.; Giunchiglia, E.; Foerster, J.; Lukasiewicz, T.; and Blunsom, P. 2019. Can I Trust the Explainer? Verifying Post-hoc Explanatory Methods. *CoRR*, abs/1910.02065.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD*, 785–794.

Clark, P.; and Niblett, T. 1989. The CN2 Induction Algorithm. *Machine Learning*, 3: 261–283.

Darwiche, A.; and Hirth, A. 2020. On the Reasons Behind Decisions. In *ECAI*, 712–720.

Darwiche, A.; and Marquis, P. 2021. On Quantifying Literals in Boolean Logic and Its Applications to Explainable AI. *J. Artif. Intell. Res.*, 72: 285–328.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. Accessed: 2022-07-08.

Ferreira, J.; de Sousa Ribeiro, M.; Gonçalves, R.; and Leite, J. 2022. Looking Inside the Black-Box: Logic-based Explanations for Neural Networks. In *KR*, 432–442.

Friedman, J. H. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5): 1189–1232.

Gario, M.; and Micheli, A. 2015. PySMT: a Solver-Agnostic Library for Fast Prototyping of SMT-Based Algorithms. In *SMT Workshop*.

Gorji, N.; and Rubin, S. 2022. Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints. In *AAAI*, 5660–5667.

Hipp, J.; Güntzer, U.; and Nakhaeizadeh, G. 2000. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDD Explor.*, 2(1): 58–64.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2022. Tractable Explanations for d-DNNF Classifiers. In *AAAI*, 5719–5728.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks. In *NIPS*, 4107–4115.

Ignatiev, A. 2020. Towards Trustable Explainable AI. In *IJCAI*, 5154–5158.

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using MaxSAT for Efficient Explanations of Tree Ensembles. In *AAAI*, 3776–3785.

Ignatiev, A.; Lam, E.; Stuckey, P. J.; and Marques-Silva, J. 2021. A Scalable Two Stage Approach to Computing Optimal Decision Sets. In *AAAI*, 3806–3814.

Ignatiev, A.; and Marques-Silva, J. 2021. SAT-Based Rigorous Explanations for Decision Lists. In *SAT*, 251–269.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2019. RC2: an Efficient MaxSAT Solver. *J. Satisf. Boolean Model. Comput.*, 11(1): 53–64.

Ignatiev, A.; Narodytska, N.; Asher, N.; and Marques-Silva, J. 2020. From Contrastive to Abductive Explanations and Back Again. In *AI*IA*, 335–355.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019a. Abduction-Based Explanations for Machine Learning Models. In *AAAI*, 1511–1519.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019b. On Relating Explanations and Adversarial Examples. In *NeurIPS*, 15857–15867.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019c. On Validating, Repairing and Refining Heuristic ML Explanations. *CoRR*, abs/1907.02509.

Ignatiev, A.; Previti, A.; Liffiton, M. H.; and Marques-Silva, J. 2015. Smallest MUS Extraction with Minimal Hitting Set Dualization. In *CP*, 173–182.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2022. On Tackling Explanation Redundancy in Decision Trees. *J. Artif. Intell. Res.*, 75: 261–321.

Izza, Y.; Jabbour, S.; Raddaoui, B.; and Boudane, A. 2020. On the Enumeration of Association Rules: A Decomposition-based Approach. In Bessiere, C., ed., *IJCAI*, 1265–1271.

Izza, Y.; and Marques-Silva, J. 2021. On Explaining Random Forests with SAT. In *IJCAI*.

Kohavi, R. 1996. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *KDD*, 202–207.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521(7553): 436.

Liffiton, M. H.; Previti, A.; Malik, A.; and Marques-Silva, J. 2016. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2): 223–250.

Liffiton, M. H.; and Sakallah, K. A. 2008. Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. *J. Autom. Reasoning*, 40(1): 1–33.

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

Lundberg, S. M.; and Lee, S. 2017. A Unified Approach to Interpreting Model Predictions. In *NeurIPS*, 4765–4774.

Malfa, E. L.; Michelmore, R.; Zbrzezny, A. M.; Paoletti, N.; and Kwiatkowska, M. 2021. On Guaranteed Optimal Robust Explanations for NLP Models. In *IJCAI*, 2658–2665.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2020. Explaining Naive Bayes and Other Linear Classifiers with Polynomial Time and Delay. In *NeurIPS*.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2021. Explanations for Monotonic Classifiers. In *ICML*, 7469–7479.

Marques-Silva, J.; Heras, F.; Janota, M.; Previti, A.; and Belov, A. 2013. On Computing Minimal Correction Subsets. In *IJCAI*, 615–622.

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *AAAI*, 3806–3814.

Mencia, C.; Previti, A.; and Marques-Silva, J. 2015. Literal-Based MCS Extraction. In *IJCAI*, 1973–1979.

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

Narodytska, N.; Shrotri, A. A.; Meel, K. S.; Ignatiev, A.; and Marques-Silva, J. 2019. Assessing Heuristic Machine Learning Explanations with Model Counting. In *SAT*, 267–278.

Olson, R. S.; Cava, W. G. L.; Orzechowski, P.; Urbanowicz, R. J.; and Moore, J. H. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10(1): 36:1–36:13.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 8024–8035.

Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.*, 32(1): 57–95.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*, 1135–1144.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*, 1527–1535.

Rivest, R. L. 1987. Learning Decision Lists. *Mach. Learn.*, 2(3): 229–246.

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

Shih, A.; Choi, A.; and Darwiche, A. 2018. A Symbolic Approach to Explaining Bayesian Network Classifiers. In *IJCAI*, 5103–5111.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *AIES*, 180–186.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR (Poster)*.

Wäldchen, S.; MacDonald, J.; Hauch, S.; and Kutyniok, G. 2021. The Computational Complexity of Understanding Binary Classifier Decisions. *J. Artif. Intell. Res.*, 70: 351–387.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; Narodytska, N.; and Marques-Silva, J. 2022. Eliminating The Impossible, Whatever Remains Must Be True. *CoRR*, abs/2206.09551.

Zaki, M. J.; Parthasarathy, S.; Ogihara, M.; and Li, W. 1997. New Algorithms for Fast Discovery of Association Rules. In *KDD*, 283–286.

Zhang, C.; and Zhang, S. 2002. *Association Rule Mining, Models and Algorithms*.

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].

On Formal Feature Attribution and Its Approximation

Jinqiang Yu **Alexey Ignatiev** **Peter J. Stuckey**
Department of Data Science and AI, Faculty of IT
Monash University, Melbourne, Victoria, Australia
{jinqiang.yu,alexey.ignatiev,peter.stuckey}@monash.edu

Abstract

Recent years have witnessed the widespread use of artificial intelligence (AI) algorithms and machine learning (ML) models. Despite their tremendous success, a number of vital problems like ML model brittleness, their fairness, and the lack of interpretability warrant the need for the active developments in explainable artificial intelligence (XAI) and formal ML model verification. The two major lines of work in XAI include *feature selection* methods, e.g. Anchors, and *feature attribution* techniques, e.g. LIME and SHAP. Despite their promise, most of the existing feature selection and attribution approaches are susceptible to a range of critical issues, including explanation unsoundness and *out-of-distribution* sampling. A recent formal approach to XAI (FXAI) although serving as an alternative to the above and free of these issues suffers from a few other limitations. For instance and besides the scalability limitation, the formal approach is unable to tackle the feature attribution problem. Additionally, a formal explanation despite being formally sound is typically quite large, which hampers its applicability in practical settings. Motivated by the above, this paper proposes a way to apply the apparatus of formal XAI to the case of feature attribution based on formal explanation enumeration. Formal feature attribution (FFA) is argued to be advantageous over the existing methods, both formal and non-formal. Given the practical complexity of the problem, the paper then proposes an efficient technique for approximating exact FFA. Finally, it offers experimental evidence of the effectiveness of the proposed approximate FFA in comparison to the existing feature attribution algorithms not only in terms of feature importance and but also in terms of their relative order.[1]

1 Introduction

Thanks to the unprecedented fast growth and the tremendous success, Artificial Intelligence (AI) and Machine Learning (ML) have become a universally acclaimed standard in automated decision making causing a major disruption in computing and the use of technology in general [1, 37, 43, 59]. An ever growing range of practical applications of AI and ML, on the one hand, and a number of critical issues observed in modern AI systems (e.g. decision bias [3] and brittleness [76]), on the other hand, gave rise to the quickly advancing area of theory and practice of Explainable AI (XAI).

Numerous methods exist to explain decisions made by what is called black-box ML models [58, 60]. Here, *model-agnostic* approaches based on random sampling prevail [58], with the most popular being *feature selection* [68] and *feature attribution* [48, 68] approaches. Despite their promise, model-agnostic approaches are susceptible to a range of critical issues, like unsoundness of explanations [22, 28] and *out-of-distribution sampling* [42, 74], which exacerbates the problem of trust in AI.

An alternative to model-agnostic explainers is represented by the methods building on the success of formal reasoning applied to the logical representations of ML models [53, 73]. Aiming to address the limitations of model-agnostic approaches, formal XAI (FXAI) methods themselves suffer from a few downsides, including the lack of scalability and the requirement to build a complete logical representation of the ML model. Formal explanations also tend to be larger than their model-agnostic

[1] Source code and complete experimental setup are available at https://github.com/ffattr/ffa.git.

Preprint. Under review.

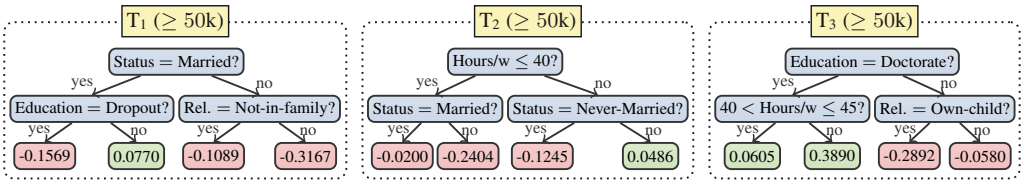
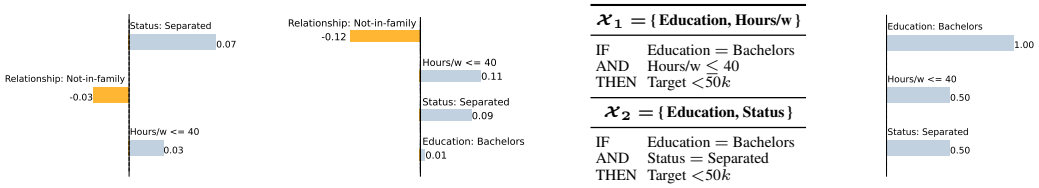


Figure 1: Example boosted tree model [12] trained on the well-known *adult* classification dataset.



$\mathcal{X}_1$ = { Education, Hours/w }

IF Education = Bachelors
AND Hours/w ≤ 40
THEN Target $<50k$

$\mathcal{X}_2$ = { Education, Status }

IF Education = Bachelors
AND Status = Separated
THEN Target $<50k$

(a) LIME (b) SHAP (c) AXp's $\mathcal{X}_1$ and $\mathcal{X}_2$ (d) FFA

Figure 2: Examples of feature attribution reported by LIME and SHAP, as well as both AXp's (no more AXp's exist) followed by FFA for the instance $\mathbf{v}$ shown in Example 1.

counterparts because they do not reason about (unknown) data distribution [77]. Finally and most importantly, FXAI methods have not been applied so far to answer feature attribution questions.

Motivated by the above, we define a novel formal approach to feature attribution, which builds on the success of existing FXAI methods [53]. By exhaustively enumerating all formal explanations, we can give a crisp definition of *formal feature attribution* (FFA) as the proportion of explanations in which a given feature occurs. We argue that formal feature attribution is hard for the second level of the polynomial hierarchy. Although it can be challenging to compute exact FFA in practice, we show that existing anytime formal explanation enumeration methods can be applied to efficiently approximate FFA. Our experimental results demonstrate the effectiveness of the proposed approach in practice and its advantage over SHAP and LIME given publicly available tabular and image datasets, as well as on a real application of XAI in the domain of Software Engineering [57, 64].

2 Background

This section briefly overviews the status quo in XAI and background knowledge the paper builds on.

2.1 Classification Problems

Classification problems consider a set of classes $\mathcal{K} = \{1, 2, \ldots, k\}$[2], and a set of features $\mathcal{F} = \{1, \ldots, m\}$. The value of each feature $i \in \mathcal{F}$ is taken from a domain $\mathbb{D}_i$, which can be categorical or ordinal, i.e. integer, real-valued or Boolean. Therefore, the complete feature space is defined as $\mathbb{F} \triangleq \prod_{i=1}^{m} \mathbb{D}_i$. A concrete point in feature space is represented by $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$, where each component $v_i \in \mathbb{D}_i$ is a constant taken by feature $i \in \mathcal{F}$. An *instance* or *example* is denoted by a specific point $\mathbf{v} \in \mathbb{F}$ in feature space and its corresponding class $c \in \mathcal{K}$, i.e. a pair $(\mathbf{v}, c)$ represents an instance. Additionally, the notation $\mathbf{x} = (x_1, \ldots, x_m)$ denotes an arbitrary point in feature space, where each component x_i is a variable taking values from its corresponding domain $\mathbb{D}_i$ and representing feature $i \in \mathcal{F}$. A classifier defines a non-constant classification function $\kappa : \mathbb{F} \to \mathcal{K}$.

Many ways exist to learn classifiers κ given training data, i.e. a collection of labeled instances $(\mathbf{v}, c)$, including decision trees [27] and their ensembles [11, 12], decision lists [69], neural networks [43], etc. Hereinafter, this paper considers boosted tree (BT) models trained with the use of XGBoost [12].

Example 1. *Figure 1 shows a BT model trained for a simplified version of the* adult *dataset [41]. For a data instance* $\mathbf{v} = \{$*Education = Bachelors, Status = Separated, Occupation = Sales, Relation-*

[2]Any set of classes $\{c_1, \ldots, c_k\}$ can always be mapped into the set of the corresponding indices $\{1, \ldots, k\}$.

ship $=$ *Not-in-family*, *Sex* $=$ *Male*, *Hours/w* $\leq$ *40*$\}$, *the model predicts <50k because the sum of the weights in the 3 trees for this instance equals* $-0.4073 = (-0.1089 - 0.2404 - 0.0580) < 0$.

2.2 ML Model Interpretability and Post-Hoc Explanations

Interpretability is generally accepted to be a subjective concept, without a formal definition [47]. One way to measure interpretability is in terms of the succinctness of information provided by an ML model to justify a given prediction. Recent years have witnessed an upsurge in the interest in devising and applying interpretable models in safety-critical applications [60, 70]. An alternative to interpretable models is post-hoc explanation of *black-box* models, which this paper focuses on.

Numerous methods to compute explanations have been proposed recently [58, 60]. The lion's share of these comprise what is called *model-agnostic* approaches to explainability [48, 67, 68] of heuristic nature that resort to extensive sampling in the vicinity of an instance being explained in order to "estimate" the behavior of the classifier in this local vicinity of the instance. In this regard, they rely on estimating input data distribution by building on the information about the training data [42]. Depending on the form of explanations model-agnostic approaches offer, they are conventionally classified as *feature selection* or *feature attribution* approaches briefly discussed below.

Feature Selection. A feature selection approach identifies subsets of features that are deemed *sufficient* for a given prediction $c = \kappa(\mathbf{v})$. As mentioned above, the majority of feature selection approaches are model-agnostic with one prominent example being Anchors [68]. As such, the sufficiency of the selected set of features for a given prediction is determined statistically based on extensive sampling around the instance of interest, by assessing a few measures like *fidelity*, *precision*, among others. As a result, feature selection explanations given as a set of features $\omega \subseteq \mathcal{F}$ should be interpreted as the conjunction $\bigwedge_{i\in\omega}(x_i = v_i)$ deemed responsible for prediction $c = \kappa(\mathbf{v})$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$. Due to the statistical nature of these explainers, they are known to suffer from various explanation quality issues [28, 42, 75]. An additional line of work on *formal* explainability [30, 73] also tackles feature selection while offering guarantees of soundness; these are discussed below.

Feature Attribution. A different view on post-hoc explanations is provided by feature attribution approaches, e.g. LIME [67] and SHAP [48]. 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 $w_i \in \mathbb{R}$ of importance to each feature $i \in \mathcal{F}$. Given these importance values, the features can then be ranked from most important to least important. As a result, 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 seen as approximating the original black-box explainer κ in the *local* neighborhood of instance $\mathbf{v} \in \mathbb{F}$. Among other feature attribution approaches, SHAP [5, 6, 48] is often claimed to stand out as it aims at approximating Shapley values, a powerful concept originating from cooperative games in game theory [72].

Formal Explainability. In this work, we build on formal explainability proposed in earlier work [8, 14, 30, 53, 73]. where explanations are equated with *abductive explanations* (AXp's). Abductive explanations are *subset-minimal* sets of features formally proved to suffice to explain an ML prediction given a formal representation of the classifier of interest. Concretely, given an instance $\mathbf{v} \in \mathbb{F}$ and a prediction $c = \kappa(\mathbf{v})$, an AXp is a subset-minimal set of features $\mathcal{X} \subseteq \mathcal{F}$, such that

$$\forall(\mathbf{x} \in \mathbb{F}).\, \bigwedge\nolimits_{i\in\mathcal{X}}(x_i = v_i) \rightarrow (\kappa(\mathbf{x}) = c) \tag{1}$$

Abductive explanations are guaranteed to be subset-minimal sets of features proved to satisfy (1). As other feature selection explanations, they answer *why* a certain prediction was made. An alternate way to explain a model's behavior is to seek an answer *why not* another prediction was made, or, in other words, *how* to change the prediction. Explanations answering *why not* questions are referred to as *contrastive explanations* (CXp's) [31, 53, 58]. As in prior work, we define a CXp as a subset-minimal set of features that, if allowed to change their values, are *necessary* to change the prediction of the model. Formally, a CXp for prediction $c = \kappa(\mathbf{v})$ is a subset-minimal set of features $\mathcal{Y} \subseteq \mathcal{F}$, such that

$$\exists(\mathbf{x} \in \mathbb{F}).\, \bigwedge\nolimits_{i\notin\mathcal{Y}}(x_i = v_i) \land (\kappa(\mathbf{x}) \neq c) \tag{2}$$

Finally, recent work has shown that AXp's and CXp's for a given instance $\mathbf{v} \in \mathbb{F}$ are related through the *minimal hitting set duality* [31, 66]. The duality implies that each AXp for a prediction $c = \kappa(\mathbf{v})$

is a *minimal hitting set*[3] (MHS) of the set of all CXp's for that prediction, and the other way around: each CXp is an MHS of the set of all AXp's. The explanation enumeration algorithm [31] applied in this paper heavily relies on this duality relation and is inspired by the MARCO algorithm originating from the area of over-constrained systems [44, 45, 65]. A growing body of recent work on formal explanations is represented (but not limited) by [2, 4, 7, 9, 10, 13, 15, 19, 21, 23, 24, 26, 29, 32–36, 49–56, 77, 80].

Example 2. *In the context of Example 1, feature attribution computed by LIME and SHAP as well as all 2 AXp's are shown in Figure 2. AXp $\mathcal{X}_1$ indicates that specifying Education $=$ Bachelors and Hours/w $\leq$ 40 guarantees that any compatible instance is classified as $<$ 50k independent of the values of other features, e.g. Status and Relationship, since the maximal sum of weights is $0.0770 - 0.0200 - 0.0580 = -0.0010 < 0$ as long as the feature values above are used. Observe that another AXp $\mathcal{X}_2$ for $\mathbf{v}$ is {Education, Status}. Since both of the two AXp's for $\mathbf{v}$ consist of two features, it is difficult to judge which one is better without a formal feature importance assessment.*

3 Why Formal Feature Attribution?

On the one hand, abductive explanations serve as a viable alternative to non-formal feature selection approaches because they (i) guarantee subset-minimality of the selected sets of features and (ii) are computed via formal reasoning over the behavior of the corresponding ML model. Having said that, they suffer from a few issues. First, observe that deciding the validity of (1) requires a formal reasoner to take into account the complete feature space $\mathbb{F}$, assuming that the features are independent and uniformly distributed [77]. In other words, the reasoner has to check all the combinations of feature values, including those that *never appear in practice*. This makes AXp's being unnecessarily *conservative* (long), i.e. they may be hard for a human decision maker to interpret. Second, AXp's are not aimed at providing feature attribution. The abundance of various AXp's for a single data instance [30], e.g. see Example 2, exacerbates this issue as it becomes unclear for a user which of the AXp's to use to make an informed decision in a particular situation.

On the other hand, non-formal feature attribution in general is known to be susceptible to out-of-distribution sampling [42, 74] while SHAP is shown to fail to effectively approximate Shapley values [22]. Moreover and quite surprisingly, [22] argued that even the use of exact Shapley values is inadequate as a measure of feature importance. Our results below confirm that both LIME and SHAP often fail to grasp the real feature attribution in a number of practical scenarios.

To address the above limitations, we propose the concept of *formal feature attribution* (FFA) as defined next. (An insight on this was also given in [22].) Let us denote the set of all formal abductive explanations for a prediction $c = \kappa(\mathbf{v})$ by $\mathbb{A}_\kappa(\mathbf{v}, c)$. Then formal feature attribution of a feature $i \in \mathcal{F}$ can be defined as the proportion of abductive explanations where it occurs. More formally,

Definition 1: (FFA). The *formal feature attribution* $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ of a feature $i \in \mathcal{F}$ to an instance $(\mathbf{v}, c)$ for machine learning model κ is

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

Formal feature attribution has some nice properties. First, it has a strict and formal definition, i.e. we can, assuming we are able to compute the complete set of AXp's for an instance, exactly define it for all features $i \in \mathcal{F}$. Second, it is fairly easy to explain to a user of the classification system, even if they are non-expert. Namely, it is the percentage of (formal abductive) explanations that make use of a particular feature i. Third, as we shall see later, even though we may not be able to compute all AXp's exhaustively, we can still get good approximations fast.

Example 3. *Recall Example 2. As there are 2 AXp's for instance $\mathbf{v}$, the prediction can be attributed to the 3 features with non-zero FFA shown in Figure 2d. Also, observe how both LIME and SHAP (see Figure 2a and Figure 2b) assign non-zero attribution to the feature Relationship, which is in fact irrelevant for the prediction, but overlook the highest importance of feature Education.*

One criticism of the above definition is that it does not take into account the length of explanations where the feature arises. Arguably if a feature arises in many AXp's of size 2, it should be considered

[3] 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.

more important than a feature which arises in the same number of AXp's but where each is of size 10. An alternate definition, which tries to take this into account, is the weighted formal feature attribution (WFFA), i.e. the *average* proportion of AXp's that include feature $i \in \mathcal{F}$. Formally,

Definition 2: (WFFA). The *weighted formal feature attribution* $\text{wffa}_\kappa(i, (\mathbf{v}, c))$ of a feature $i \in \mathcal{F}$ to an instance $(\mathbf{v}, c)$ for machine learning model κ is

$$\text{wffa}_\kappa(i, (\mathbf{v}, c)) = \frac{\sum_{\mathcal{X} \in \mathbb{A}_\kappa(\mathbf{v}, c), i \in \mathcal{X}} |\mathcal{X}|^{-1}}{|\mathbb{A}_\kappa(\mathbf{v}, c)|} \tag{4}$$

Note that these attribution values are not on the same scale although they are convertible:

$$\sum_{i \in \mathcal{F}} \text{ffa}_\kappa(i, (\mathbf{v}, c)) = \frac{\sum_{\mathcal{X} \in \mathbb{A}_\kappa(\mathbf{v}, c)} |\mathcal{X}|}{|\mathbb{A}_\kappa(\mathbf{v}, c)|} \times \sum_{i \in \mathcal{F}} \text{wffa}_\kappa(i, (\mathbf{v}, c)).$$

FFA can be related to the problem of *feature relevancy* [25], where a feature is said to be *relevant* if it belongs to at least one AXp. Indeed, feature $i \in \mathcal{F}$ is relevant for prediction $c = \kappa(\mathbf{v})$ if and only if $\text{ffa}_\kappa(i, (\mathbf{v}, c)) > 0$. As a result, the following claim can be made.

Proposition 1. *Given a feature $i \in \mathcal{F}$ and a prediction $c = \kappa(\mathbf{v})$, deciding whether $\textit{ffa}_\kappa(i, (\mathbf{v}, c)) > \omega$, $\omega \in (0, 1]$, is at least as hard as deciding whether feature i is relevant for the prediction.*

The above result indicates that computing exact FFA values may be expensive in practice. For example and in light of [25], one can conclude that the decision version of the problem is Σ_2^{P}-hard in the case of DNF classifiers.

Similarly and using the relation between FFA and feature relevancy above, we can note that the decision version of the problem is in Σ_2^{P} as long as deciding the validity of (1) is in NP, which in general is the case (unless the problem is simpler, e.g. for decision trees [34]). Namely, the following result is a simple consequence of the membership result for the feature relevance problem [25].

Proposition 2. *Deciding whether $\textit{ffa}_\kappa(i, (\mathbf{v}, c)) > 0$ is in Σ_2^P if deciding* (1) *is in NP.*

4 Approximating Formal Feature Attribution

As the previous section argues and as our experimental results confirm, it may be challenging in practice to compute exact FFA values due to the general complexity of the problem. Although some ML models admit efficient formal encodings and reasoning procedures, effective principal methods for FFA approximation seem necessary. This section proposes one such method.

Normally, formal explanation enumeration is done by exploiting the MHS duality between AXp's and CXp's and the use of MARCO-like [45] algorithms aiming at efficient exploration of minimal hitting sets of either AXp's or CXp's [31, 44, 45, 65]. Depending on the target type of formal explanation, MARCO exhaustively enumerates all such explanations one by one, each time extracting a candidate minimal hitting set and checking if it is a desired explanation. If it is then it is recorded and blocked such that this candidate is never repeated again. Otherwise, a dual explanation is extracted from the subset of features complementary to the candidate [30], gets recorded and blocked so that it is hit by each future candidate. The procedure proceeds until no more hitting sets of the set of dual explanations can be extracted, which signifies that all target explanations are enumerated. Observe that while doing so, MARCO also enumerates all the dual explanations as a kind of "side effect".

One of the properties of MARCO used in our approximation approach is that it is an *anytime* algorithm, i.e. we can run it for as long as we need to get a sufficient number of explanations. This means we can stop it by using a timeout or upon collecting a certain number of explanations.

The main insight of FFA approximation is as follows. Recall that to compute FFA, we are interested in AXp enumeration. Although intuitively this suggests the use of MARCO targeting AXp's, for the sake of fast and high-quality FFA approximation, we propose to target CXp enumeration with AXp's as dual explanations computed "unintentionally". The reason for this is twofold: (i) we need to get a good FFA approximation as fast as we can and (ii) according to our practical observations, MARCO needs to amass a large number of dual explanations before it can start producing target explanations. This is because the hitting set enumerator is initially "blind" and knows nothing about the features

Algorithm 1 MARCO-like Anytime Explanation Enumeration

procedure XpEnum(κ, $\mathbf{v}$, c)
  $(\mathbb{A}, \mathbb{C}) \leftarrow (\emptyset, \emptyset)$ ▷ *Sets of AXp's and CXp's to collect.*
  while true **do**
    $\mathcal{Y} \leftarrow$ MinimalHS($\mathbb{A}, \mathbb{C}$) ▷ *Get a new MHS of $\mathbb{A}$ subject to $\mathbb{C}$.*
    if $\mathcal{Y} = \bot$ **then break** ▷ *Stop if none is computed.*
    if $\exists(\mathbf{x} \in \mathbb{F}). \bigwedge_{i \notin \mathcal{Y}}(x_i = v_i) \land (\kappa(\mathbf{x}) \neq c)$ **then** ▷ *Check CXp condition (2) for $\mathcal{Y}$.*
      $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathcal{Y}\}$ ▷ *$\mathcal{Y}$ appears to be a CXp.*
    else ▷ *There must be a missing AXp $\mathcal{X} \subseteq \mathcal{F} \setminus \mathcal{Y}$.*
      $\mathcal{X} \leftarrow$ ExtractAXp($\mathcal{F} \setminus \mathcal{Y}, \kappa, \mathbf{v}, c$) ▷ *Get AXp $\mathcal{X}$ by iteratively checking (1) [30].*
      $\mathbb{A} \leftarrow \mathbb{A} \cup \{\mathcal{X}\}$ ▷ *Collect new AXp $\mathcal{X}$.*
  return $\mathbb{A}$, $\mathbb{C}$

it should pay attention to — it uncovers this information gradually by collecting dual explanations to hit. This way a large number of dual explanations can quickly be enumerated during this initial phase of grasping the search space, essentially "for free". Our experimental results demonstrate the effectiveness of this strategy in terms of monotone convergence of approximate FFA to the exact FFA with the increase of the time limit. A high-level view of the version of MARCO used in our approach targeting CXp enumeration and amassing AXp's as dual explanations is shown in Algorithm 1.

5 Experimental Evidence

This section assesses the formal feature attribution for gradient boosted trees (BT) [12] on multiple widely used images and tabular datasets, and compares FFA with LIME and SHAP. In addition, it also demonstrates the use of FFA in a real-world scenario of Just-in-Time (JIT) defect prediction, which assists teams in prioritizing their limited resources on high-risk commits or pull requests [64].

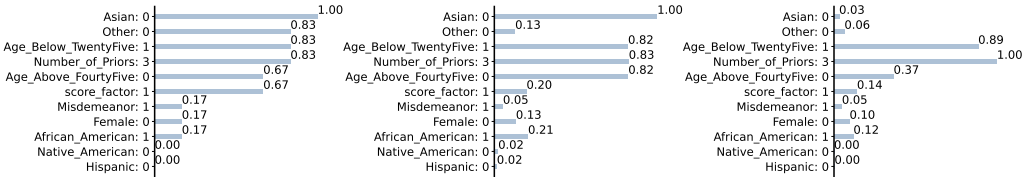
Setup and Prototype Implementation. All experiments were performed on an Intel Xeon 8260 CPU running Ubuntu 20.04.2 LTS, with the memory limit of 8 GByte. A prototype of the approach implementing Algorithm 1 and thus producing FFA was developed as a set of Python scripts and builds on [32]. As the FFA and WFFA values turn out to be almost identical (subject to normalization) in our experiments, here we report only FFA. WFFA results can be found in supplementary material.

Datasets and Machine Learning Models. The well-known MNIST dataset [16, 62] of handwritten digits 0–9 is considered, with two concrete binary classification tasks created: 1 vs. 3 and 1 vs. 7. We also consider PneumoniaMNIST [79], a binary classification dataset to distinguish X-ray images of pneumonia from normal cases. To demonstrate extraction of *exact* FFA values for the above datasets, we also examine their downscaled versions, i.e. reduced from $28 \times 28 \times 1$ to $10 \times 10 \times 1$. We also consider 11 tabular datasets often applied in the area of ML explainability and fairness [3, 17, 18, 20, 61, 71]. All the considered datasets are randomly split into 80% training and and 20% test data. For images, 15 test instances are randomly selected in each test set for explanation while all tabular test instances are explained. For all datasets, gradient boosted trees (BTs) are trained by XGBoost [12], where each BT consists of 25 trees of depth 3 per class.[4] Finally, we show the use of FFA on 2 JIT defect prediction datasets [64], with 500 instances per dataset chosen for analysis.

5.1 Formal Feature Attribution

In this section, we restrict ourselves to examples where we can compute the *exact* FFA values for explanations by computing all AXp's. To compare with LIME and SHAP, we take their solutions, replace negative attributions by the positive counterpart (in a sense taking the absolute value) and then normalize the values into $[0, 1]$. We then compare these approaches with the computed FFA values, which are also in $[0, 1]$. The *error* is measured as Manhattan distance, i.e. the sum of absolute differences across all features. We also compare feature rankings according to the competitors (again using absolute values for LIME and SHAP) using Kendall's Tau [39] and rank-biased overlap (RBO) [78]

[4]Test accuracy for MNIST digits is 0.99, while it is 0.83 for PneumoniaMNIST. This holds both for the 28×28 and 10×10 versions of the datasets. The average accuracy across the 11 selected tabular datasets is 0.80.



(a) FFA (b) LIME (c) SHAP

Figure 3: Explanations for an instance of Compas $\mathbf{v} = \{\text{\#Priors} = 3, \text{Score\_factor} = 1, \text{Age\_Above\_FourtyFive} = 0, \text{Age\_Below\_TwentyFive} = 1, \text{African\_American} = 1, \text{Asian} = 0, \text{Hispanic} = 0, \text{Native\_American} = 0, \text{Other} = 0, \text{Female} = 0, \text{Misdemeanor} = 1\}$ predicted as Two_yr_Recidivism = true.

Table 1: LIME and SHAP versus FFA on tabular data.

Dataset ($\lvert\mathcal{F}\rvert$)	**adult** (12)	**appendicitis** (7)	**australian** (14)	**cars** (8)	**compas** (11)	**heart-statlog** (13)	**hungarian** (13)	**lending** (9)	**liver-disorder** (6)	**pima** (8)	**recidivism** (15)
Approach						**Error**					
LIME	4.48	2.25	5.13	1.53	3.28	4.48	4.56	1.39	2.39	2.72	4.73
SHAP	4.47	2.01	4.49	1.40	2.67	3.71	4.14	1.44	2.28	3.00	4.76
						Kendall's Tau					
LIME	0.07	0.11	0.22	-0.11	-0.11	0.17	0.04	-0.36	-0.22	0.17	0.05
SHAP	0.03	0.12	0.27	-0.10	-0.10	0.17	0.20	-0.39	-0.21	0.07	0.12
						RBO					
LIME	0.54	0.66	0.49	0.63	0.55	0.56	0.41	0.59	0.66	0.68	0.39
SHAP	0.49	0.67	0.55	0.66	0.59	0.52	0.49	0.61	0.67	0.63	0.44

metrics.[5] Kendall's Tau and RBO are measured on a scale $[-1, 1]$ and $[0, 1]$, respectively. A higher value in both metrics indicates better agreement or closeness between a ranking and FFA.

Tabular Data. Figure 3 exemplifies a comparison of FFA, LIME and SHAP on an instance of the Compas dataset [3]. While FFA and LIME agree on the most important feature, "Asian", SHAP gives it very little weight. Neither LIME nor SHAP agree with FFA, though there is clearly some similarity.

Table 1 details the comparison conducted on 11 tabular datasets, including *adult*, *compas*, and *recidivism* datasets commonly used in XAI. For each dataset, we calculate the metric for each individual instance and then average the outcomes to obtain the final result for that dataset. As can be observed, the errors of LIME's feature attribution across these datasets span from 1.39 to 5.13. SHAP demonstrates similar errors within a range $[1.40, 4.76]$. LIME and SHAP also exhibit comparable performance in relation to the two ranking comparison metrics. The values of Kendall's Tau for LIME (resp. SHAP) are between -0.36 and 0.22 (resp. -0.39 and 0.27). Regarding the RBO values, LIME exhibits values between 0.39 and 0.68, whereas SHAP demonstrates values ranging from 0.44 to 0.67. Overall, as Table 1 indicates, both LIME and SHAP fail to get close enough to FFA.

10×10 Digits. We now compare the results on 10×10 downscaled MNIST digits and PneumoniaMNIST images, where it is feasible to compute all AXp's. Table 2 compares LIME's, SHAP's feature attribution and approximate FFA. Here, we run AXp enumeration for a number of seconds, which is denoted as FFA_*, $* \in \mathbb{R}^+$. The runtime required for each image by LIME and SHAP is less than one second. The results show that the errors of our approximation are small, even after 10 seconds it beats both LIME and SHAP, and decreases as we generate more AXp's. The results for the orderings show again that after 10 seconds, FFA_* ordering gets closer to the exact FFA than both LIME and SHAP. Observe how LIME is particularly far away from the *exact* FFA ordering.

Summary. *These results make us confident that we can get useful approximations to the exact FFA without exhaustively computing all AXp's while feature attribution determined by LIME and SHAP is quite erroneous and fails to provide a human-decision maker with useful insights, despite being fast.*

[5] Kendall's Tau is a correlation coefficient assessing the ordinal association between two ranked lists, offering a measure of similarity in the order of values; on the other hand, RBO is a metric that measures the similarity between two ranked lists, taking into account both the order and the depth of the overlap.

Table 2: Comparison on 10×10 Images of FFA versus LIME, SHAP and FFA approximations.

Dataset	LIME	SHAP	FFA_{10}	FFA_{30}	FFA_{60}	FFA_{120}	FFA_{600}	FFA_{1200}
($\lvert\mathcal{F}\rvert = 100$)				**Error**				
10×10-mnist-1vs3	11.50	10.07	5.74	5.33	4.97	4.62	3.37	2.67
10×10-mnist-1vs7	12.64	8.28	4.16	3.58	2.94	2.50	1.42	1.01
10×10-pneumoniamnist	17.32	17.90	5.37	4.32	3.78	3.39	2.22	1.64
				Kendall's Tau				
10×10-mnist-1vs3	-0.15	0.48	0.49	0.57	0.62	0.65	0.74	0.80
10×10-mnist-1vs7	-0.33	0.47	0.52	0.63	0.70	0.77	0.85	0.89
10×10-pneumoniamnist	-0.02	0.24	0.58	0.71	0.79	0.80	0.89	0.92
				RBO				
10×10-mnist-1vs3	0.20	0.50	0.61	0.65	0.69	0.74	0.81	0.84
10×10-mnist-1vs7	0.19	0.58	0.73	0.77	0.81	0.86	0.90	0.90
10×10-pneumoniamnist	0.21	0.37	0.61	0.70	0.73	0.77	0.83	0.87

(a) LIME (b) SHAP (c) FFA_{10} (d) FFA_{30} (e) FFA_{120} (f) FFA_{600} (g) $\text{FFA}_{1.2k}$ (h) $\text{FFA}_{3.6k}$ (i) $\text{FFA}_{7.2k}$

Figure 4: 28×28 MNIST 1 vs. 3. The prediction is digit 3. The *plasma* gradient is used ranging from deep purple for the least important features to vibrant yellow for the most important features.

Table 3: Comparison on 28×28 Images of FFA_{7200} versus LIME, SHAP and FFA approximations.

Dataset	LIME	SHAP	FFA_{10}	FFA_{30}	FFA_{120}	FFA_{600}	FFA_{1200}	FFA_{3600}
($\lvert\mathcal{F}\rvert = 784$)				**Error**				
28×28-mnist-1vs3	49.66	22.77	9.44	7.61	6.81	4.51	3.13	2.69
28×28-mnist-1vs7	55.10	24.92	11.78	9.58	6.94	4.51	3.30	2.18
28×28-pneumoniamnist	62.94	31.55	8.17	7.81	5.69	4.89	3.77	3.10
				Kendall's Tau				
28×28-mnist-1vs3	-0.80	0.42	0.44	0.62	0.69	0.80	0.86	0.87
28×28-mnist-1vs7	-0.79	0.34	0.40	0.56	0.72	0.82	0.87	0.92
28×28-pneumoniamnist	-0.66	0.24	0.34	0.50	0.67	0.76	0.80	0.87
				RBO				
28×28-mnist-1vs3	0.03	0.40	0.43	0.50	0.61	0.78	0.83	0.88
28×28-mnist-1vs7	0.03	0.34	0.40	0.45	0.58	0.76	0.83	0.93
28×28-pneumoniamnist	0.03	0.23	0.31	0.35	0.42	0.59	0.66	0.83

5.2 Approximating Formal Feature Attribution

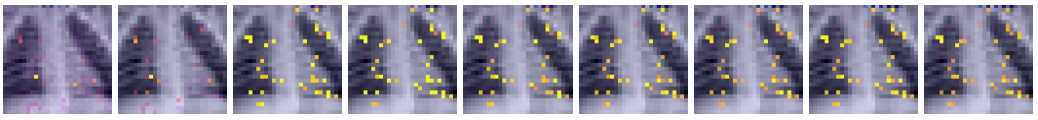
Since the problem of formal feature attribution "lives" in Σ_2^{P}, it is not surprising that computing FFA may be challenging in practice. Table 2 suggests that our approach gets good FFA approximations even if we only collect AXp's for a short time. Here we compare the fidelity of our approach versus the approximate FFA computed after 2 hours (7200s). Figure 4, 5, and 6 depict feature attributions generated by LIME, SHAP and FFA_* for the three selected 28×28 images. The comparison between LIME, SHAP, and the approximate FFA computation is detailed in Table 3. The LIME and SHAP processing time for each image is less than one second. The average findings detailed in Table 3 are consistent with those shown in Table 2. Namely, FFA approximation yields better errors, Kendall's Tau and RBO values, outperforming both LIME, and SHAP after 10 seconds. Furthermore, the results demonstrate that after 10 seconds our approach places feature attributions closer to FFA_{7200} compared to both LIME and SHAP hinting on the features that are truly relevant for the prediction.

5.3 Application in Just-in-Time Defect Prediction

Just-in-Time (JIT) defect prediction [38, 40, 46, 63] has been recently proposed to predict if a commit will introduce software defects in the future, enabling development teams to prioritize their limited Software Quality Assurance resources on the most risky commits/pull requests. The approach of JIT

(a) LIME (b) SHAP (c) FFA_{10} (d) FFA_{30} (e) FFA_{120} (f) FFA_{600} (g) $\text{FFA}_{1.2k}$ (h) $\text{FFA}_{3.6k}$ (i) $\text{FFA}_{7.2k}$

Figure 5: 28×28 MNIST 1 vs. 7. The prediction is digit 7.



(a) LIME (b) SHAP (c) FFA_{10} (d) FFA_{30} (e) FFA_{120} (f) FFA_{600} (g) $\text{FFA}_{1.2k}$ (h) $\text{FFA}_{3.6k}$ (i) $\text{FFA}_{7.2k}$

Figure 6: 28×28 PneumoniaMNIST. The prediction is normal.

Table 4: Just-in-Time Defect Prediction comparison of FFA versus LIME and SHAP.

Approach	**openstack** ($\lvert\mathcal{F}\rvert = 13$)			**qt** ($\lvert\mathcal{F}\rvert = 16$)		
	Error	**Kendall's Tau**	**RBO**	**Error**	**Kendall's Tau**	**RBO**
LIME	4.84	0.05	0.55	5.63	-0.08	0.45
SHAP	5.08	0.00	0.53	5.22	-0.13	0.44

defect prediction has often been considered a black-box, lacking explainability for practitioners. To tackle this challenge, our proposed approach to generating FFA can be employed, as model-agnostic approaches cannot guarantee to provide accurate feature attribution (see above). We use logistic regression models of [64] based on large-scale open-source Openstack and Qt datasets provided by [57] commonly used for JIT defect prediction [64]. Monotonicity of logistic regression enables us to enumerate explanations using the approach of [56] and so to extract *exact FFA* for each instance *within a second*. Table 4 details the comparison of FFA, LIME and SHAP in terms of the three considered metrics. As with the outcomes presented in Table 1, Table 2, and Table 3, neither LIME nor SHAP align with formal feature attribution, though there are some similarities between them.

6 Limitations

Despite the rigorous guarantees provided by formal feature attribution and high-quality of the result explanations, the following limitations can be identified. First, our approach relies on formal reasoning and thus requires an ML model of interest to admit a representation in some fragments of first-order logic, and the corresponding reasoner to deal with it [53]. Second, the problem complexity impedes immediate and widespread use of FFA and signifies the need to develop effective methods of FFA approximation. Finally, though our experimental evidence suggests that FFA approximations quickly converge to the exact values of FFA, whether or not this holds in general remains an open question.

7 Conclusions

Most approaches to XAI are heuristic methods that are susceptible to unsoundness and out-of-distribution sampling. Formal approaches to XAI have so far concentrated on the problem of feature selection, detecting which features are important for justifying a classification decision, and not on feature attribution, where we can understand the weight of a feature in making such a decision. In this paper we define the first formal approach to feature attribution (FFA) we are aware of, using the proportion of abductive explanations in which a feature occurs to weight its importance. We show that we can compute FFA exactly for many classification problems, and when we cannot we can compute effective approximations. Existing heuristic approaches to feature attribution do not agree with FFA. Sometimes they markedly differ, for example, assigning no weight to a feature that appears in (a large number of) explanations, or assigning (large) non-zero weight to a feature that is irrelevant for the prediction. Overall, the paper argues that if we agree that FFA is a correct measure of feature attribution then we need to investigate methods that compute good FFA approximations quickly.

References

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

[2] L. Amgoud and J. Ben-Naim. Axiomatic foundations of explainability. In L. D. Raedt, editor, *IJCAI*, pages 636–642, 2022.

[3] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias. http://tiny.cc/dd7mjz, 2016.

[4] M. Arenas, D. Baez, P. Barceló, J. Pérez, and B. Subercaseaux. Foundations of symbolic languages for model interpretability. In *NeurIPS*, 2021.

[5] M. Arenas, P. Barceló, L. E. Bertossi, and M. Monet. The tractability of SHAP-score-based explanations for classification over deterministic and decomposable Boolean circuits. In *AAAI*, pages 6670–6678. AAAI Press, 2021.

[6] M. Arenas, P. Barceló, L. E. Bertossi, and M. Monet. On the complexity of SHAP-score-based explanations: Tractability via knowledge compilation and non-approximability results. *CoRR*, abs/2104.08015, 2021.

[7] M. Arenas, P. Barceló, M. A. R. Orth, and B. Subercaseaux. On computing probabilistic explanations for decision trees. In *NeurIPS*, 2022.

[8] G. Audemard, F. Koriche, and P. Marquis. On tractable XAI queries based on compiled representations. In *KR*, pages 838–849, 2020.

[9] G. Blanc, J. Lange, and L. Tan. Provably efficient, succinct, and precise explanations. In *NeurIPS*, 2021.

[10] R. Boumazouza, F. C. Alili, B. Mazure, and K. Tabia. ASTERYX: A model-Agnostic SaT-basEd appRoach for sYmbolic and score-based eXplanations. In *CIKM*, pages 120–129, 2021.

[11] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[12] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

[13] M. C. Cooper and J. Marques-Silva. Tractability of explaining classifier decisions. *Artif. Intell.*, 316:103841, 2023.

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

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

[16] L. Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[17] D. Dua and C. Graff. UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml.

[18] FairML. Auditing black-box predictive models. http://tiny.cc/6e7mjz, 2016.

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

[20] S. Friedler, C. Scheidegger, and S. Venkatasubramanian. On algorithmic fairness, discrimination and disparate impact. http://fairness.haverford.edu/, 2015.

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

[22] X. Huang and J. Marques-Silva. The inadequacy of Shapley values for explainability. *CoRR*, abs/2302.08160, 2023.

[23] X. Huang and J. Marques-Silva. From robustness to explainability and back again. *CoRR*, abs/2306.03048, 2023.

[24] X. Huang, Y. Izza, A. Ignatiev, M. C. Cooper, N. Asher, and J. Marques-Silva. Tractable explanations for d-DNNF classifiers. In *AAAI*, pages 5719–5728, 2022.

[25] X. Huang, M. C. Cooper, A. Morgado, J. Planes, and J. Marques-Silva. Feature necessity & relevancy in ML classifier explanations. In *TACAS (1)*, pages 167–186, 2023.

[26] X. Huang, Y. Izza, and J. Marques-Silva. Solving explainability queries with quantification: The case of feature relevancy. In *AAAI*, pages 4123–4131, 2023.

[27] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976. URL `https://doi.org/10.1016/0020-0190(76)90095-8`.

[28] A. Ignatiev. Towards trustable explainable AI. In *IJCAI*, pages 5154–5158, 2020.

[29] A. Ignatiev and J. Marques-Silva. SAT-based rigorous explanations for decision lists. In *SAT*, pages 251–269, 2021.

[30] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

[31] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. From contrastive to abductive explanations and back again. In *AI*IA*, pages 335–355, 2020.

[32] A. Ignatiev, Y. Izza, P. J. Stuckey, and J. Marques-Silva. Using MaxSAT for efficient explanations of tree ensembles. In *AAAI*, pages 3776–3785, 2022.

[33] Y. Izza and J. Marques-Silva. On explaining random forests with SAT. In *IJCAI*, July 2021.

[34] Y. Izza, A. Ignatiev, and J. Marques-Silva. On tackling explanation redundancy in decision trees. *J. Artif. Intell. Res.*, 75:261–321, 2022. URL `https://doi.org/10.1613/jair.1.13575`.

[35] Y. Izza, X. Huang, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. On computing probabilistic abductive explanations. *International Journal of Approximate Reasoning*, 159, 2023.

[36] Y. Izza, A. Ignatiev, P. J. Stuckey, and J. Marques-Silva. Delivering inflated explanations. *CoRR*, abs/2306.15272, 2023.

[37] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[38] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. A Large-Scale Empirical Study of Just-In-Time Quality Assurance. *IEEE Transactions on Software Engineering (TSE)*, 39(6):757–773, 2013.

[39] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[40] S. Kim, T. Zimmermann, E. J. Whitehead Jr, and A. Zeller. Predicting Faults from Cached History. In *ICSE*, pages 489–498, 2007.

[41] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207, 1996.

[42] H. Lakkaraju and O. Bastani. "How do I fool you?": Manipulating user trust via misleading black box explanations. In *AIES*, pages 79–85, 2020.

[43] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[44] M. H. Liffiton and A. Malik. Enumerating infeasibility: Finding multiple MUSes quickly. In *CPAIOR*, pages 160–175, 2013.

[45] M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2):223–250, 2016.

[46] D. Lin, C. Tantithamthavorn, and A. E. Hassan. The impact of data merging on the interpretation of cross-project just-in-time defect models. *IEEE Transactions on Software Engineering*, 2021.

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

[48] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017.

[49] E. L. Malfa, R. Michelmore, A. M. Zbrzezny, N. Paoletti, and M. Kwiatkowska. On guaranteed optimal robust explanations for NLP models. In *IJCAI*, pages 2658–2665, 2021.

[50] J. Marques-Silva. Logic-based explainability in machine learning. *CoRR*, abs/2211.00541, 2022.

[51] J. Marques-Silva. Logic-based explainability in machine learning. In *Reasoning Web*, pages 24–104, 2022.

[52] J. Marques-Silva. Disproving XAI myths with formal methods - initial results. *CoRR*, abs/2306.01744, 2023.

[53] J. Marques-Silva and A. Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350. AAAI Press, 2022.

[54] J. Marques-Silva and A. Ignatiev. No silver bullet: Interpretable ml models must be explained. *Frontiers in Artificial Intelligence*, 6:1–15, 2023.

[55] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explaining naive Bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*, 2020.

[56] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explanations for monotonic classifiers. In *ICML*, pages 7469–7479, 2021.

[57] S. McIntosh and Y. Kamei. Are fix-inducing changes a moving target? A longitudinal case study of Just-in-Time defect prediction. *IEEE Transactions on Software Engineering (TSE)*, pages 412–428, 2017.

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

[59] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[60] C. Molnar. *Interpretable Machine Learning*. Leanpub, 2020. `http://tiny.cc/6c76tz`.

[61] R. S. Olson, W. G. L. Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, 10(1): 36:1–36:13, 2017.

[62] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

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

[64] C. Pornprasit, C. Tantithamthavorn, J. Jiarpakdee, M. Fu, and P. Thongtanunam. PyExplainer: Explaining the predictions of Just-In-Time defect models. In *ASE*, pages 407–418, 2021.

[65] A. Previti and J. Marques-Silva. Partial MUS enumeration. In *AAAI*. AAAI Press, 2013.

[66] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[67] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144, 2016.

[68] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535, 2018.

[69] R. L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.

[70] C. 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.

[71] P. Schmidt and A. D. Witte. Predicting recidivism in North Carolina, 1978 and 1980. *Inter-University Consortium for Political and Social Research*, 1988.

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

[73] A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining Bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018.

[74] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In *AIES*, pages 180–186, 2020.

[75] D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. In *NeurIPS*, pages 9391–9404, 2021.

[76] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR (Poster)*, 2014.

[77] S. Wäldchen, J. MacDonald, S. Hauch, and G. Kutyniok. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.*, 70:351–387, 2021.

[78] W. Webber, A. Moffat, and J. Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

[79] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni. MedMNIST v2-a large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Scientific Data*, 10(1):41, 2023.

[80] J. Yu, A. Ignatiev, P. J. Stuckey, N. Narodytska, and J. Marques-Silva. Eliminating the impossible, whatever remains must be true. In *AAAI*, pages 4123–4131, 2023.

Appendices

A Exact Weighted Formal Feature Attribution

In this appendix, we once again limit our analysis to instances where we can calculate the *exact* WFFA values for the instance of interest by enumerating all AXp's. Also, the settings used in Section 5 are applied here, i.e. we take the absolute values of feature attibution assignned by LIME and SHAP, and normalize them within the range of $[0, 1]$. Just like in the main text of the paper, we then compare these approaches with normalized WFFA values in terms of errors, Kendall's Tau [39] and rank-biased overlap (RBO) [78].

A.1 Tabular Data

A comparison of WFFA, LIME and SHAP on an instance of the Compas dataset [3] is exemplified in Figure 7. We can observe the patterns similar to those depicted in Figure 3. The feature that WFFA considers most important is "Asian" while this viewpoint is shared by LIME but disputed by SHAP. However, neither LIME nor SHAP fully align with WFFA, although there is evident similarity between them. As with FFA, these observations can be generalized to the other instances of Compas, as discussed below.

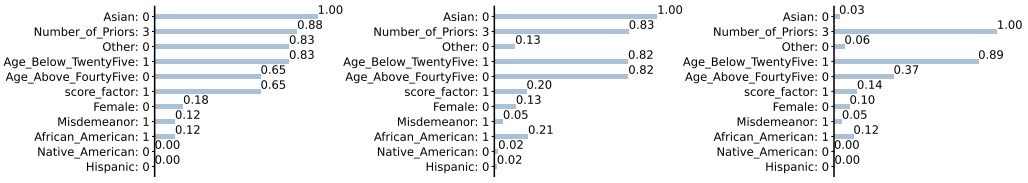
Table 5 presents a comparison of WFFA against LIME, and SHAP on the 11 selected tabular datasets as in Table 1, demonstrating similarities in the findings observed for WFFA and FFA for these datasets. The average runtime for generating the exact WFFA in a dataset varies between 0.18 and 1.89 seconds while the average number of AXp's per instance to explain and so to compute exact WFFA in a dataset ranges from 1.40 to 33.33. Both LIME and SHAP process each image in less than one second. LIME exhibits errors ranging from 1.37 to 4.96 across these datasets while SHAP shows similar errors spanning from 1.36 to 4.67. Besides errors, LIME and SHAP yield comparable outcomes in terms of the two ranking comparison metrics. The values of Kendall's Tau for LIME span from -0.35 to 0.25, whereas the values for SHAP are between -0.38 and 0.31. Regarding RBO values, LIME (resp. SHAP) demonstrates values ranging from 0.38 to 0.69 (resp. 0.43 to 0.67). Overall and consistent with the FFA findings shown earlier in Table 1, Table 5 indicates that both LIME and SHAP fail to achieve close enough agreement with WFFA.

A.2 10×10 Digits

Table 6 provides a comprehensive comparison of approximate WFFA against feature attribution reported by LIME and SHAP with respect to the exact WFFA values, conducted on the downscaled MNIST digists and PneumoniaMNIST images, where exhaustive AXp enumeration is feasible. The values of feature attribution generated by LIME, SHAP, and approximate WFFA_* for the three selected 10×10 images are shown in Figure 11, Figure 12, and Figure 13. Over time, the number of features included in the AXp's increases, and the weighted attribution of each feature changes converging to the exact WFFA. The results shown in Figure 8, Figure 9, and Figure 10 align with the main finding for FFA approximation shown earlier. Furthermore, the results shown in Table 6 are also consistent with FFA observations in Table 2. Both LIME and SHAP can process each image within a runtime of less than one second. The average runtime and average number of AXp's generated for 10×10 MNIST 1 vs 3 (resp. 1 vs 7) are 14264.78s and 15781.87 (resp. 6834.61s and 4028.27), while the values in 10×10 PneumoniaMNIST are 8656.18s and 8802.87, respectively. Similarly to the results in Table 2, Table 6 indicates that our approximation yields small errors. Even after 10 seconds, it outperforms both LIME and SHAP, and the errors continue to decrease as we compute more AXp's. Once again, the results of the orderings demonstrate that after 10 seconds, the ordering of WFFA_* approaches closer to the exact WFFA compared to both LIME and SHAP and converges to the exact WFFA ordering with the growth of the number AXp's enumerated. As can also be seen, LIME exhibits a substantial distance from the *exact* WFFA ordering.

A.3 Summary

The findings of this section again indicate that we can confidently obtain valuable approximations of the exact WFFA values without the need to exhaustively enumerate all AXp's for a given data



(a) WFFA (b) LIME (c) SHAP

Figure 7: Explanations for an instance of Compas $\mathbf{v} = \{\#\text{Priors} = 3, \text{Score\_factor} = 1, \text{Age\_Above\_FourtyFive} = 0, \text{Age\_Below\_TwentyFive} = 1, \text{African\_American} = 1, \text{Asian} = 0, \text{Hispanic} = 0, \text{Native\_American} = 0, \text{Other} = 0, \text{Female} = 0, \text{Misdemeanor} = 1\}$ predicted as Two_yr_Recidivism = true.

Table 5: LIME and SHAP versus WFFA on tabular data.

Dataset	**adult**	**appendicitis**	**australian**	**cars**	**compas**	**heart-statlog**	**hungarian**	**lending**	**liver-disorder**	**pima**	**recidivism**
$\lvert\mathcal{F}\rvert$	(12)	(7)	(14)	(8)	(11)	(13)	(13)	(9)	(6)	(8)	(15)
Approach						**Error**					
LIME	4.32	2.06	4.96	1.48	3.26	4.40	4.43	1.37	2.37	2.63	4.66
SHAP	4.29	1.87	4.31	1.36	2.63	3.61	4.00	1.43	2.25	2.91	4.67
						Kendall's Tau					
LIME	0.11	0.17	0.25	-0.08	-0.08	0.22	0.08	-0.35	-0.17	0.25	0.08
SHAP	0.07	0.23	0.31	-0.07	-0.07	0.22	0.26	-0.38	-0.16	0.15	0.16
						RBO					
LIME	0.53	0.65	0.48	0.64	0.56	0.56	0.40	0.59	0.65	0.69	0.38
SHAP	0.48	0.67	0.55	0.66	0.59	0.52	0.49	0.61	0.67	0.64	0.43

Table 6: Comparison on 10×10 Images of WFFA versus LIME, SHAP and WFFA approximations.

Dataset	**LIME**	**SHAP**	**WFFA$_{10}$**	**WFFA$_{30}$**	**WFFA$_{60}$**	**WFFA$_{120}$**	**WFFA$_{600}$**	**WFFA$_{1200}$**
$\lvert\mathcal{F}\rvert = 100$					**Error**			
10×10-mnist-1vs3	11.28	9.81	5.52	5.12	4.83	4.50	3.32	2.61
10×10-mnist-1vs7	12.46	8.11	4.07	3.47	2.83	2.38	1.34	0.97
10×10-pneumoniamnist	17.25	17.84	5.33	4.29	3.76	3.36	2.20	1.63
					Kendall's Tau			
10×10-mnist-1vs3	-0.14	0.48	0.53	0.60	0.64	0.67	0.75	0.81
10×10-mnist-1vs7	-0.33	0.47	0.58	0.65	0.73	0.79	0.86	0.90
10×10-pneumoniamnist	-0.02	0.24	0.67	0.74	0.80	0.81	0.90	0.92
					RBO			
10×10-mnist-1vs3	0.20	0.50	0.63	0.67	0.70	0.74	0.81	0.84
10×10-mnist-1vs7	0.19	0.58	0.73	0.77	0.81	0.86	0.90	0.91
10×10-pneumoniamnist	0.21	0.37	0.63	0.70	0.74	0.77	0.82	0.87

instance. It is worth noting that feature attribution determined by LIME and SHAP is quite inaccurate and does not provide meaningful insights to a human decision-maker, despite being computationally fast.

B Approximate Weighted Formal Feature Attribution

As argued in Section 3, the exact WFFA computation can be difficult in practice, due to the complexity of the problem. But as Table 6 indicates, our approach can yield decent WFFA approximations even with a short duration of collecting AXp's. Here we assess the fidelity of our approach in contrast to the approximate WFFA computed after a duration of 2 hours (7200s). WFFA$_*$ and the values of feature attribution generated by LIME and SHAP for the three considered 28×28 images are depicted in Figure 14, 15, and 16. As time progresses, the accumulated AXp's incorporate an increasing number of features, and as a result the value of weighted attribution for each feature can change. Table 7 details the comparison between LIME, SHAP, and the approximate WFFA. Both LIME and

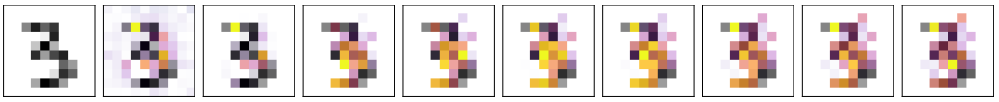


Figure 8: 10×10 MNIST 1 vs. 3. The prediction is 3.

Figure 9: 10×10 MNIST 1 vs. 7. The prediction is 7.

Table 7: Comparison on 28×28 Images of $\text{WFFA}_{7.2k}$ versus LIME, SHAP and WFFA approximations.

Dataset	**LIME**	**SHAP**	**WFFA_{10}**	**WFFA_{30}**	**WFFA_{120}**	**WFFA_{600}**	**WFFA_{1200}**	**WFFA_{3600}**
$\lvert\mathcal{F}\rvert = 784$					**Error**			
28,28-mnist-1,3	49.28	22.33	9.22	7.50	6.69	4.50	3.08	2.75
28,28-mnist-1,7	54.78	24.39	11.53	9.40	7.00	4.60	3.33	2.29
28,28-pneumoniamnist	62.88	31.46	8.17	7.74	5.67	4.85	3.75	3.08
					Kendall's Tau			
28,28-mnist-1,3	-0.80	0.42	0.49	0.64	0.70	0.81	0.86	0.88
28,28-mnist-1,7	-0.79	0.34	0.43	0.57	0.72	0.82	0.87	0.92
28,28-pneumoniamnist	-0.66	0.24	0.37	0.57	0.69	0.76	0.81	0.88
					RBO			
28,28-mnist-1,3	0.03	0.40	0.45	0.54	0.63	0.78	0.84	0.89
28,28-mnist-1,7	0.03	0.34	0.41	0.47	0.60	0.74	0.81	0.91
28,28-pneumoniamnist	0.03	0.23	0.30	0.35	0.43	0.59	0.65	0.81

Table 8: Just-in-time Defect Prediction comparison of WFFA versus LIME and SHAP.

Approach	**openstack** ($\lvert\mathcal{F}\rvert = 13$)			**qt** ($\lvert\mathcal{F}\rvert = 16$)		
	Error	**kendalltau**	**rbo**	**Error**	**kendalltau**	**rbo**
LIME	4.79	0.08	0.56	5.60	-0.07	0.45
SHAP	5.01	0.02	0.54	5.17	-0.11	0.44

SHAP require less than one second to process each image. The average results presented in Table 7 are consistent with those illustrated in Table 6 and the FFA results depicted in Table 2 and Table 3. Table 7 demonstrates that after only 10 seconds, our WFFA approximation outperforms both LIME and SHAP in terms of errors, Kendall's Tau, and RBO values. Additionally, after 10 seconds our approach produces weighted feature attributions, which is closer to WFFA_{7200} compared to both LIME and SHAP. This suggests that our approach effectively identifies the features that are genuinely relevant for the prediction, which is in stark contrast to LIME and SHAP.

C Application in Just-in-Time Defect Prediction

Modern software companies often engage in the rapid and frequent release of software products in short cycles. Because of the exponential growth of highly complex source code, such rapid-release software development presents significant challenges for under-resourced Software Quality Assurance (SQA) teams. Developers are unable to thoroughly ensure the highest quality of all newly developed code commits or pull requests within the limited time and resources available, due to the time-consuming and costly nature of various SQA activities, e.g. code review. To address this issue, a recent approach called Just-in-Time (JIT) defect prediction [38, 40, 46, 63] has been proposed. This approach aims to predict whether a commit will introduce software defects in the future such

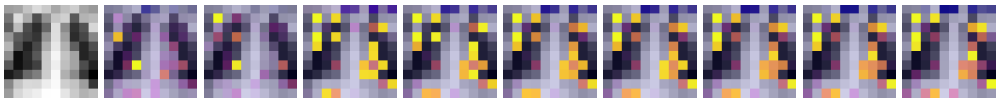


Figure 10: 10×10 PneumoniaMNIST. The prediction is pneumonia.

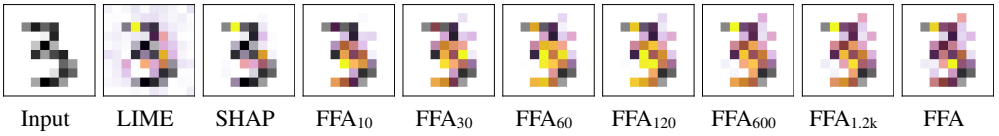


Figure 11: 10×10 MNIST 1 vs. 3. The prediction is 3.

that development teams can prioritize their limited SQA resources on the riskiest commits or pull requests.

However, the JIT defect prediction approach has frequently been criticized for being opaque and lacking explainability for practitioners. Model-agnostic explainability methods, e.g. LIME and SHAP, cannot guarantee accurate feature attribution, as discussed earlier in this appendix and Section 5). Experimental evidence presented in Section 5 demonstrates the usefulness of exact FFA in the context of JIT defect prediction. Given that our earlier observations above suggest that exact (resp. approximate) WFFA is consistent with exact (resp. approximate) FFA, we apply the computation of WFFA in the setting of JIT defection prediction and demonstrate that it can be also a viable approach to addressing practical explainability challenges.

In particular, where we use logistic regression models built on two widely-used large-scale open-source datasets, namely Openstack and Qt, which are commonly used in JIT defect prediction studies [64]. The property of monotonicity in logistic regression allows us to enumerate explanations efficiently, following the approach of [56]. By leveraging this method, we can extract the *exact WFFA* for each instance within one second. The comparison of WFFA, LIME, and SHAP in terms of the three selected metrics is provided in Table 8. These results are consistent with the FFA assessment presented in Table 4. Similar to the findings in Table 5, Table 6, and Table 7, both LIME and SHAP misalign with weighted formal feature attribution, although there are some similarities between them.

Figure 12: 10×10 MNIST 1 vs. 7. The prediction is 7.

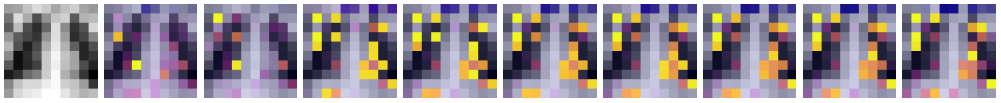


Figure 13: 10×10 PneumoniaMNIST. The prediction is pneumonia.

Figure 14: 28×28 MNIST 1 vs. 3. The prediction is digit 3.

Figure 15: 28×28 MNIST 1 vs. 7. The prediction is digit 7.

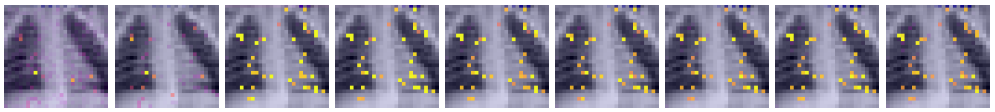


Figure 16: 28×28 PneumoniaMNIST. The prediction is normal.

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.

Anytime Approximate Formal Feature Attribution

Jinqiang Yu, Graham Farr, Alexey Ignatiev, Peter J. Stuckey

Department of Data Science and AI, Faculty of IT
Monash University, Melbourne, Victoria, Australia
{jinqiang.yu,graham.farr,alexey.ignatiev,peter.stuckey}@monash.edu

Abstract

Widespread use of artificial intelligence (AI) algorithms and machine learning (ML) models on the one hand and a number of crucial issues pertaining to them warrant the need for explainable artificial intelligence (XAI). A key explainability question is: given this decision was made, what are the input features which contributed to the decision? Although a range of XAI approaches exist to tackle this problem, most of them have significant limitations. Heuristic XAI approaches suffer from the lack of quality guarantees, and often try to approximate Shapley values, which is not the same as explaining which features contribute to a decision. A recent alternative is so-called formal feature attribution (FFA), which defines feature importance as the fraction of formal abductive explanations (AXp's) containing the given feature. This measures feature importance from the view of formally reasoning about the model's behavior. It is challenging to compute FFA using its definition because that involves counting AXp's, although one can approximate it. Based on these results, this paper makes several contributions. First, it gives compelling evidence that computing FFA is intractable, even if the set of contrastive formal explanations (CXp's) is provided, by proving that the problem is #P-hard. Second, by using the duality between AXp's and CXp's, it proposes an efficient heuristic to switch from CXp enumeration to AXp enumeration on-the-fly resulting in an adaptive explanation enumeration algorithm effectively approximating FFA in an anytime fashion. Finally, experimental results obtained on a range of widely used datasets demonstrate the effectiveness of the proposed FFA approximation approach in terms of the error of FFA approximation as well as the number of explanations computed and their diversity given a fixed time limit.

1 Introduction

The rise of the use of artificial intelligence (AI) and machine learning (ML) methods to help interpret data and make decisions has exposed a keen need for these algorithms to be able to explain their decisions/judgements. Lack of explanation of opaque and complex models leads to lack of trust, and allows the models to encapsulate unfairness, discrimination and other unwanted properties learnt from the data or through training.

For a classification problem a key explainability question is: "given a decision was made (a class was imputed to some data instance), what are the features that contributed to the decision?". A more complex question is: "given the decision was made, how important was each feature in making that decision?". There are many heuristic approaches to answering this question, mostly based on sampling around the instance (Ribeiro, Singh, and Guestrin 2016), and attempting to approximate Shapley values (Lundberg and Lee 2017). But there is strong evidence that Shapley values do not really compute the importance of a feature to a decision (Huang and Marques-Silva 2023b; Marques-Silva and Huang 2023).

Formal approaches to explainability are able compute formal *abductive explanations* (AXp's) for a decision, that is a minimal set of features which are enough to ensure the same decision will be made (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019; Marques-Silva and Ignatiev 2022). They can also compute formal *contrastive explanations* (CXp's), that is a minimal set of features, one of which must change in order to change the decision (Miller 2019; Ignatiev et al. 2020). A wealth of algorithms originating from the area of dealing with over-constrained systems (Bailey and Stuckey 2005; Liffiton and Sakallah 2008; Belov, Lynce, and Marques-Silva 2012; Marques-Silva et al. 2013; Liffiton et al. 2016) can be applied for the computation and enumeration of AXp's and CXp's (Marques-Silva and Ignatiev 2022). Here, enumeration of formal explanations builds on the use of the minimal hitting set duality between AXp's and CXp's (Reiter 1987; Liffiton et al. 2016; Ignatiev et al. 2020). Until recently there was no formal approach to ascribing importance to features.

A recent and attractive approach to formal feature attribution, called FFA (Yu, Ignatiev, and Stuckey 2023b), is simple. Compute all the abductive explanations for a decision, then the importance of a feature for the decision is simply the proportion of abductive explanations in which it appears. FFA is crisply defined, and easy to understand, but it is challenging to compute, as deciding if a feature has a non-zero attribution is at least as hard as deciding feature relevancy (Huang et al. 2023; Yu, Ignatiev, and Stuckey 2023b).

Yu *et al.* (Yu, Ignatiev, and Stuckey 2023b) show that FFA can be efficiently computed by making use of the hitting set duality between AXp's and CXp's. By trying to enumerate CXp's, a side effect of the algorithm is to discover AXp's. In fact, the algorithm will usually find many AXp's before finding the first CXp. The AXp's are guaranteed to be diverse, since they need to be broad in scope to ensure that the CXp is large enough to hit all AXp's that apply to the decision.

Using AXp's collected as a side effect of CXp enumeration is effective at the start of the enumeration. But as we find more and more AXp's as side effects we eventually get to a point where many more CXp's are generated than AXp's. Experimentation shows that if we wish to enumerate all AXp's then indeed we should not rely on the side effect behavior, but simply enumerate AXp's directly. This leads to a quandary: to get fast accurate approximations of FFA we wish to enumerate CXp's and generate AXp's as a side effect. But to compute the final correct FFA we wish to compute all AXp's, and we are better off directly enumerating AXp's.

In this paper, we develop an *anytime* approach to computing approximate FFA, by starting with CXp enumeration, and then dynamically switching to AXp enumeration when the rate of AXp discovery by CXp enumeration drops. In doing so, we are able to quickly get accurate approximations, but also arrive to the full set of AXp's quicker than pure CXp enumeration. As direct CXp enumeration is feasible to do without the need to resort to the hitting set duality (Ignatiev et al. 2020; Marques-Silva and Ignatiev 2022), one may want to estimate FFA by first enumerating CXp's. The second contribution of this paper is to investigate this alternative approach and to show that even if a(n) (in)complete set of CXp's is given, determining FFA is computationally expensive being #P-hard even if all CXp's are of size two.

2 Preliminaries

Here we introduce the notation and background on formal XAI in order to define formal feature attribution (FFA).

2.1 Classification Problems

We assume classification problems classify data instances into classes $\mathcal{K}$ where $|\mathcal{K}| = k \geq 2$. We are given a set of m features $\mathcal{F}$, where the value of feature $i \in \mathcal{F}$ comes from a domain $\mathbb{D}_i$, which may be Boolean, (bounded) integer or (bounded) real. The *complete feature space* is defined by $\mathbb{F} \triangleq \prod_{i=1}^{m} \mathbb{D}_i$.

A *data point* in feature space is denoted $\mathbf{v} = (v_1, \ldots, v_m)$ where $v_i \in \mathbb{D}_i, 1 \leq i \leq m$. An *instance* of the classification problem is a pair of feature vector and its corresponding class, i.e. $(\mathbf{v}, c)$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$.

We use the notation $\mathbf{x} = (x_1, \ldots, x_m)$ to represent an arbitrary point in feature space, where each x_i will take a value from $\mathbb{D}_i$.

A *classifier* is a total function from feature space to class: $\kappa : \mathbb{F} \to \mathcal{K}$. Many approaches exist to define classifiers including decision sets (Clark and Boswell 1991; Lakkaraju, Bach, and Leskovec 2016), decision lists (Rivest 1987), decision trees (Hyafil and Rivest 1976), random forests (Friedman 2001), boosted trees (Chen and Guestrin 2016), and neural nets (Nair and Hinton 2010; Hubara et al. 2016).

2.2 Formal Explainability

Given a data point $\mathbf{v}$, classifier κ classifies it as class $\kappa(\mathbf{v})$. A *post hoc explanation* of the behavior of κ on data point $\mathbf{v}$ tries to explain the behavior of κ on this instance. We consider two forms of formal explanation answering *why* and *why not* (or *how*) questions.

An *abductive explanation* (AXp) is a minimal set of features $\mathcal{X}$ such that any data point sharing the same feature values with $\mathbf{v}$ on these features is guaranteed to be assigned the same class by $c = \kappa(\mathbf{v})$ (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019). Formally, $\mathcal{X}$ is a subset-minimal set of features such that:

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

A dual concept of *contrastive explanations* (CXp's) helps us understand *how* to reach another prediction (Miller 2019; Ignatiev et al. 2020; Marques-Silva and Ignatiev 2022). A *contrastive explanation* (CXp) for the classification of data point $\mathbf{v}$ as class $c = \kappa(\mathbf{v})$ is a minimal set of features such that at least one must change in order that κ will return a different class. Formally, a CXp is a subset minimal set of features $\mathcal{Y}$ such that

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

Interestingly, the set $\mathbb{A}$ of all AXp's $\mathcal{X}$ explaining classification $\kappa(\mathbf{v}) = c$ and the set $\mathbb{C}$ of all CXp's $\mathcal{Y}$ explaining the same classification enjoy a *minimal hitting set duality* (Ignatiev et al. 2020). That is $\mathbb{A} = \text{MHS}(\mathbb{C})$ and is $\mathbb{C} = \text{MHS}(\mathbb{A})$ where $\text{MHS}(S)$ returns the minimal hitting sets of S, that is the minimal sets that share an element with each subset in S. More formally, $\text{HS}(S) = \{t \subseteq (\cup S) \mid \forall s \in S,\ t \cap s \neq \emptyset\})$ and $\text{mins}(S) = \{s \in S \mid \forall t \subsetneq s,\ t \notin S\}$ returns the subset minimal elements of a set of sets, and $\text{MHS}(S) = \text{mins}(\text{HS}(S))$. This property can be made use of in computing or enumerating AXp's and/or CXp's.

A growing body of recent work on formal explanations is represented (but not limited) by (Marques-Silva et al. 2020, 2021; Izza and Marques-Silva 2021; Ignatiev and Marques-Silva 2021; Arenas et al. 2021; Wäldchen et al. 2021; Darwiche and Marquis 2021; Malfa et al. 2021; Boumazouza et al. 2021; Blanc, Lange, and Tan 2021; Izza, Ignatiev, and Marques-Silva 2022; Gorji and Rubin 2022; Ignatiev et al. 2022; Huang et al. 2022; Marques-Silva and Ignatiev 2022; Amgoud and Ben-Naim 2022; Ferreira et al. 2022; Marques-Silva 2022a; Arenas et al. 2022; Marques-Silva 2022b; Yu et al. 2023; Huang, Izza, and Marques-Silva 2023; Cooper and Marques-Silva 2023; Marques-Silva and Ignatiev 2023; Izza et al. 2023a,b; Marques-Silva 2023; Marques-Silva and Huang 2023; Yu, Ignatiev, and Stuckey 2023a; Huang and Marques-Silva 2023a; Cooper and Marques-Silva 2023; Biradar et al. 2023).

2.3 Formal Feature Attribution

Given the definition of AXp's above, we can now illustrate the *formal feature attribution* (FFA) function by Yu *et al* (Yu, Ignatiev, and Stuckey 2023b).[1] Denoted as $\text{ffa}_\kappa(i, (\mathbf{v}, c))$, it returns for a classification $\kappa(\mathbf{v}) = c$ how important feature

[1] Measuring feature importance from the perspective of formal explainability was independently studied in (Biradar et al. 2023).

Algorithm 1: Anytime Explanation Enumeration

```
1: procedure XpEnum(κ, v, c)
2:     (𝔸, ℂ) ← (∅, ∅)
3:     while resources available do
4:         𝒴 ← MinimalHS(𝔸, ℂ)
5:         if 𝒴 = ⊥ then break
6:         if ∃(x ∈ 𝔽). ⋀_{i∉𝒴}(x_i = v_i) ∧ (κ(x) ≠ c) then
7:             ℂ ← ℂ ∪ {𝒴}
8:         else
9:             𝒳 ← ExtractAXp(ℱ \ 𝒴, κ, v, c)
10:            𝔸 ← 𝔸 ∪ {𝒳}
    return 𝔸, ℂ
```

$i \in \mathcal{F}$ is in making this classification, defined as the proportion of AXp's for the classification $\mathbb{A}_\kappa(\mathbf{v}, c)$, which include feature i, i.e.

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

2.4 Computing FFA

Yu *et al* (Yu, Ignatiev, and Stuckey 2023b) 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 $\text{MinimalHS}(\mathbb{A}, \mathbb{C})$. 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.

2.5 Graph-Related Notation

The paper uses some (undirected) graph-theoretic concepts. A graph is defined as a tuple, $G = (V, E)$, where V is a finite set of vertices and E is a finite set of unordered pairs of vertices. For simplicity, uv denotes an edge $\{u, v\}$ of E. Given a graph $G = (V, E)$, a *vertex cover* $X \subseteq V$ is such that for each $uv \in E$, $\{u, v\} \cap X \neq \emptyset$. A *minimal* vertex cover is a vertex cover that is minimal wrt. set inclusion.

2.6 The Complexity of Counting

The class #P consists of functions that count accepting computations of polynomial-time nondeterministic Turing machines (Valiant 1979). A problem is *#P-hard* if every problem in #P is polynomial-time Turing reducible to it; if it also belongs to #P then it is *#P-complete*.

#P-hardness is usually regarded as stronger evidence of intractability than NP-hardness or indeed hardness for any level of the Polynomial Hierarchy.

3 Approximate Formal Feature Attribution

Facing the need to compute (exact or approximate) FFA values, one may think of a possibility to first enumerate CXp's and then apply the minimal hitting set duality between AXp's and CXp's to determine FFA, without explicitly computing $\mathbb{A} = \text{MHS}(\mathbb{C})$. This looks plausible given that CXp enumeration can be done directly, without the need to enumerate AXp's (Ignatiev et al. 2020). However, as Section 3.1 argues, computing FFA given a set of CXp's turns out to be computationally difficult, being (roughly) at least as hard as counting the minimal hitting sets $\text{MHS}(\mathbb{C})$. Hence, Section 3.2 approaches the problem from a different angle by efficient exploitation of the eMUS- or MARCO-like setup (Previti and Marques-Silva 2013; Liffiton and Malik 2013; Liffiton et al. 2016; Ignatiev et al. 2020) and making the algorithm *switch* from CXp enumeration to AXp enumeration on the fly.

3.1 Duality-Based Approximation is Hard

We show that determining $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ from $\mathbb{C}$ is #P-hard even when all CXp's have size two. In that special case, the CXp's may be treated as the edges of a graph, which we denote by $G(\mathcal{F}, \kappa, \mathbf{v}, c)$, with vertex set $\mathcal{F}$. The minimal hitting set duality between the CXp's and AXp's then implies that the AXp's $\mathcal{X} \in \text{MHS}(\mathbb{C})$ are precisely the minimal vertex covers of $G(\mathcal{F}, \kappa, \mathbf{v}, c)$. It is known that determining the number of minimal vertex covers in a graph is #P-complete (even for bipartite graphs); this is implicit in (Provan and Ball 1983), as noted for example in (Vadhan 2001, p. 400).

When all CXp's have size 2, the formal feature attribution $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ is just the proportion of minimal vertex covers of $G(\mathcal{F}, \kappa, \mathbf{v}, c)$ that contain the vertex i, i.e. the vertex of $G(\mathcal{F}, \kappa, \mathbf{v}, c)$ that represents the feature $i \in \mathcal{F}$. To help express this in graph-theoretic language, write $\#\text{mvc}(G)$ for the number of minimal vertex covers of G. Write $\#\text{mvc}(G, v)$ and $\#\text{mvc}(G, \neg v)$ for the numbers of minimal vertex covers of G that *do* and *do not* contain vertex $v \in V(G)$, respectively. Define

$$\text{ffa}(G, v) := \frac{\#\text{mvc}(G, v)}{\#\text{mvc}(G)}. \quad (4)$$

Then

$$\text{ffa}_\kappa(i, (\mathbf{v}, c)) = \text{ffa}(G(\mathcal{F}, \kappa, \mathbf{v}, c), i).$$

Observe that $\#\text{mvc}(G) = \#\text{mvc}(G, v) + \#\text{mvc}(G, \neg v)$. Then we may rewrite (4) as

$$\text{ffa}(G, v) = \frac{\#\text{mvc}(G, v)}{\#\text{mvc}(G, v) + \#\text{mvc}(G, \neg v)}. \quad (5)$$

Theorem 1. *Determining* $\text{ffa}(G, v)$ *is #P-hard.*

Proof. We give a polynomial-time Turing reduction from the #P-complete problem of counting minimal vertex covers to the problem of determining ffa for a node in a graph.

Suppose we have an oracle that, when given a graph and a vertex, returns the ffa of the vertex in one time-step.

Let G be a graph for which we want to count the minimal vertex covers. Let v be a non-isolated vertex of G. (If none

exists, the problem is trivial.) Put

$$\begin{aligned} x &= \#\text{mvc}(G, \neg v), \\ y &= \#\text{mvc}(G, v), \end{aligned}$$

so that $\#\text{mvc}(G) = x + y$. It is routine to show that $x, y > 0$. Initially, x and y are unknown. Our reduction will use an ffa-oracle to gain enough information to determine x and y. We will then obtain $\#\text{mvc}(G) = x + y$.

First, query the ffa-oracle with G and vertex v. It returns

$$p := \frac{y}{x+y},$$

by (5). We can then recover the ratio $x/y = p^{-1} - 1$.

Then we construct a new graph $G_v^{[2]}$ from G as follows. Take two disjoint copies G_1 and G_2 of G. Let v_1 be the copy of vertex v in G_1. For every $w \in V(G_2)$, add an edge $v_1 w$ between v_1 and w. We query the ffa-oracle with $G_v^{[2]}$ and vertex v_1. Let $q = \text{ffa}(G_v^{[2]}, v_1)$ be the value it returns.

If a minimal vertex cover X of $G_v^{[2]}$ contains v_1 then all the edges from v_1 to G_2 are covered. The restriction of X to G_1 must be a minimal vertex cover of G_1 that contains v_1, and the number of these is just $\#\text{mvc}(G, v) = y$. The restriction of X to G_2 must just be a vertex cover of G_2, without any further restriction, and the number of these is just $\#\text{mvc}(G) = x + y$. These two restrictions of X can be chosen independently to give all possibilities for X. So

$$\#\text{mvc}(G_v^{[2]}, v_1) = y(x + y).$$

If a minimal vertex cover X of $G_v^{[2]}$ does not contain v_1 then the edges $v_1 w$, $w \in V(G_2)$, are not covered by v_1. So each $w \in V(G_2)$ must be in X, which serves to cover not only those edges but also all edges in G_2. The restriction of X to G_1 must just be a vertex cover of G_1 that does not contain v_1, and there are $\#\text{mvc}(G, \neg v) = x$ of these. Again, the two restrictions of X are independent. So

$$\#\text{mvc}(G_v^{[2]}, \neg v_1) = x.$$

Therefore

$$q = \frac{y(x+y)}{x + y(x+y)},$$

by (5) (applied this time to $G_v^{[2]}$), so

$$x + y = \frac{x/y}{q^{-1} - 1} = \frac{p^{-1} - 1}{q^{-1} - 1}.$$

We can therefore compute $x + y$ from the values p and q returned by our two oracle calls. We therefore obtain $\#\text{mvc}(G)$. The entire computation is polynomial time. □

Corollary 1. *Determining* $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ *from the set of CXp's is #P-hard, even if all CXp's have size 2.*

3.2 Switching from CXp to AXp Enumeration

As discussed in Section 2, (Yu, Ignatiev, and Stuckey 2023b) proposed to apply implicit hitting set enumeration for approximating FFA thanks to the duality between AXp's and CXp's. The approach builds on the use of the MARCO algorithm (Previti and Marques-Silva 2013; Liffiton and Malik 2013; Liffiton et al. 2016) in the anytime fashion, i.e. collects the sets of AXp's and CXp's and stops upon reaching a given resource limit. As MARCO can be set to target enumerating either AXp's or CXp's depending on user's preferences, the dual explanations will be collected by the algorithm as a *side effect*. Quite surprisingly, the findings of (Yu, Ignatiev, and Stuckey 2023b) show that for the purposes of *practical* FFA approximation it is beneficial to target CXp enumeration and get AXp's by duality. An explanation offered for this by (Yu, Ignatiev, and Stuckey 2023b) is that MARCO has to collect a large number of dual explanations before the minimal hitting sets it gets may realistically be the target explanations.

Our practical observations confirm the above. Also note that the AXp's enumerated by MARCO need to be *diverse* if we want to quickly get good FFA approximations. Due to the *incremental* operation of the minimal hitting set enumeration algorithms, this is hard to achieve if we *target* AXp enumeration. But if we aim for CXp's then we can extract diverse AXp's by duality, which helps us get reasonable FFA approximations quickly converging to the exact FFA values.

Nevertheless, our experiments with the setup of (Yu, Ignatiev, and Stuckey 2023b) suggest that AXp enumeration in fact tends to finish much earlier than CXp enumeration despite "losing" at the beginning. This makes one wonder what to opt for if good and quickly converging FFA approximation is required: AXp enumeration or CXp enumeration. On the one hand, the latter quickly gives a large set of diverse AXp's and good initial FFA approximations. On the other hand, complete AXp enumeration finishes much faster, i.e. exact FFA is better to compute by targeting AXp's.

Motivated by this, we propose to set up MARCO in a way that it starts with CXp enumeration and then seamlessly switches to AXp enumeration using two simple heuristic criteria. It should be first noted that to make efficient switching in the target explanations, we employ pure SAT-based hitting set enumerator, where an incremental SAT solver is called multiple times aiming for minimal or maximal models (Giunchiglia and Maratea 2006), depending on the phase. This allows us to keep all the explanations found so far without ever restarting the hitting set enumerator.

As we observe that AXp's are normally larger than CXp's, both criteria for switching the target build on the use of the average *size* of the last w AXp's and the last w CXp's enumerated in the most recent iterations of the MARCO algorithm. (Recall that our MARCO setup aims for subset-minimal explanations rather than cardinality-minimal explanations, i.e. neither target nor dual explanations being enumerated are cardinality-minimal.) This can be seen as inspecting "sliding windows" of size w for both AXp's and CXp's. In particular, assume that the sets of the last w AXp's and CXp's are denoted as $\mathbb{A}^w$ and $\mathbb{C}^w$, respectively. First, switching can be done as soon as we observe that CXp's on average are *much* smaller than AXp's, i.e. when

$$\frac{\sum_{\mathcal{X} \in \mathbb{A}^w} |\mathcal{X}|}{\sum_{\mathcal{Y} \in \mathbb{C}^w} |\mathcal{Y}|} \geq \alpha, \tag{6}$$

where $\alpha \in \mathbb{R}$ is a predefined numeric parameter. The ra-

Algorithm 2: Adaptive Explanation Enumeration

```
1: procedure AdaptiveXpEnum(κ, v, c, w, α, ε)
2:     (𝔼0, 𝔼1) ← (∅, ∅)        ▷ CXp's and AXp's to collect
3:     ρ ← 0   ▷ Target phase of enumerator, initially CXp
4:     while true do
5:         μ ← MinimalHS(𝔼1−ρ, 𝔼ρ, ρ)
6:         if μ = ⊥ then break
7:         if IsTargetXp(μ, κ, v, c) then
8:             𝔼ρ ← 𝔼ρ ∪ {μ}        ▷ Collect target expl. μ
9:         else
10:            ν ← ExtractDualXp(F \ μ, κ, v, c)
11:            𝔼1−ρ ← 𝔼1−ρ ∪ {ν}   ▷ Collect dual expl. ν
12:        if IsSwitchNeeded(𝔼ρ, 𝔼1−ρ, w, α, ε) then
13:            ρ ← 1 − ρ      ▷ Flip phase of MinimalHS
    return 𝔼1, 𝔼0              ▷ Result AXp's and CXp's
```

tionale for this heuristic is as follows. Recall that extraction of a subset-minimal dual explanation is done by deciding the validity of the corresponding predicate, either (1) or (2), while iteratively removing features from the candidate feature set (see Section 2.4). As such, if the vast majority of CXp's is small, their extraction leads to the lion's share of the decision oracle calls being *satisfiable*. On the contrary, extracting large AXp's as dual explanations leads to most of the oracle calls proving *unsatisfiability*. Hence, we prefer to deal with cheap satisfiable calls rather than expensive unsatisfiable ones. Second, we can switch when the average CXp size "stabilizes". Here, let us denote a new CXp being just computed as $\mathcal{Y}_{\text{new}}$. Then the second criterion can be examined by checking if the following holds:

$$\left| |\mathcal{Y}_{\text{new}}| - \frac{\sum_{\mathcal{Y} \in \mathbb{C}^w} |\mathcal{Y}|}{w} \right| \leq \varepsilon, \tag{7}$$

with $\varepsilon \in \mathbb{R}$ being another numeric parameter. This condition is meant to signify the point when the set of AXp's we have already accumulated is diverse enough for all the CXp's to be of roughly equal size, which is crucial for good FFA approximations. Overall, the switching can be performed when either of the two conditions (6)–(7) is satisfied.

Algorithm 2 shows the pseudo-code of the adaptive explanation enumeration algorithm. Additionally to the classifier's representation κ, instance $\mathbf{v}$ to explain and its class c, it receives 3 numeric parameters: window size $w \in \mathbb{N}$ and switching-related constants $\alpha, \varepsilon \in \mathbb{R}$. The set of CXp's (resp. AXp's) is represented by $\mathbb{E}_0$ (resp. $\mathbb{E}_1$) while the target phase of the hitting set enumerator is denoted by $\rho \in \{0, 1\}$, i.e. at each iteration Algorithm 2 aims for $\mathbb{E}_\rho$ explanations. As initially $\rho = 0$, the algorithm targets CXp enumeration. Each of its iterations starts by computing a minimal hitting set μ of the set $\mathbb{E}_{1-\rho}$ subject to $\mathbb{E}_\rho$ (see line 5), i.e. we want μ to be a hitting set of $\mathbb{E}_{1-\rho}$ different from all the target explanations in $\mathbb{E}_\rho$ found so far. If no hitting set exists, the process stops as we have enumerated all target explanations. Otherwise, each new μ is checked for being a target explanation, which is done by invoking a reasoning oracle to decide the validity either of (1) if we target AXp's, or of (2) if we target CXp's. If the test is positive, the algorithm records the new explanation μ in $\mathbb{E}_\rho$. Otherwise, using the standard apparatus of formal explanations, it extracts a subset-minimal dual explanation ν from the complementary set $\mathcal{F} \setminus \mu$, which is then recorded in $\mathbb{E}_{1-\rho}$. Each iteration is additionally augmented with a check whether we should switch to the opposite phase $1 - \rho$ of the enumeration. This is done in line 12 by testing whether at least one of the conditions (6)–(7) is satisfied.

Remark. Flipping enumeration phase ρ can be seamlessly done because we apply pure SAT-based hitting enumeration (Giunchiglia and Maratea 2006) where both $\mathbb{E}_\rho$ and $\mathbb{E}_{1-\rho}$ are represented as sets of *negative* and *positive* blocking clauses, respectively. As such, by instructing the SAT solver to opt for minimal or maximal models,[2] we can flip from computing hitting sets of $\mathbb{E}_{1-\rho}$ subject to $\mathbb{E}_\rho$ to computing hitting sets of $\mathbb{E}_\rho$ subject to $\mathbb{E}_{1-\rho}$, and vice versa. Importantly, this can be done while incrementally keeping the internal state of the SAT solver, i.e. no learnt information gets lost after the phase switch. Also, note that although the algorithm allows us to apply phase switching multiple times, our practical implementation switches *once* because AXp enumeration normally gets done much earlier than CXp enumeration, i.e. there is no point in switching back.

4 Experimental Results

We evaluate our approach to FFA approximation for gradient boosted trees (BTs) on various data using a range of metrics.

4.1 Experimental Setup

The experiments were done on an Intel Xeon 8260 CPU running Ubuntu 20.04.2 LTS, with the 8GByte memory limit.

Prototype Implementation. The proposed approach was prototyped as a set of Python scripts, building on the approach of (Yu, Ignatiev, and Stuckey 2023b). The implementation can be found in the supplementary materials. The proposed approach is referred to as MARCO-S, where the MARCO algorithm switches from CXp to AXp enumeration based on the conditions (6)–(7). For this, "sliding windows" of size $w = 50$ are used; $\varepsilon = 2$ in (6) to signify the extent by which the size of AXp's should be larger than the size of CXp's, while $\alpha = 1$ in (7) denoting the stability of the average CXp size.

Datasets and Machine Learning Models. The experiments include five well-known image and text datasets. We use the widely used *MNIST* (Deng 2012; Paszke et al. 2019) dataset, which features hand-written digits from 0 to 9, with two concrete binary classification problems created: 1 vs. 3 and 1 vs. 7. Also, we consider the image dataset *PneumoniaMNIST* (Yang et al. 2023) differentiating normal X-ray cases from pneumonia. Since extracting *exact* FFA values for aforementioned image datasets turns out to be hard (Yu, Ignatiev, and Stuckey 2023b), we perform a size reduction, downscaling these images from $28 \times 28 \times 1$ to $10 \times 10 \times 1$. Additionally, 2 text datasets are considered in the experiments:

[2] In SAT solving, a *minimal* model is a satisfying assignment that respects subset-minimality wrt. the set of positive literals, i.e. where none of the 1's can be replaced by a 0 such that the result is still a satisfying assignment (Biere et al. 2021). *Maximal* models can be defined similarly wrt. subset-minimality of negative literals.

Sarcasm (Misra and Arora 2023; Misra and Grover 2021) and *Disaster* (Howard et al. 2019). The *Sarcasm* dataset contains news headlines collected from websites, along with binary labels indicating whether each headline is sarcastic or non-sarcastic. The *Disaster* dataset consists of the contents of tweets with labels about whether a user announces a real disaster or not. The 5 considered datasets are randomly divided into 80% training and 20% test data. To evaluate the performance of the proposed approach, 15 test instances in each test set are randomly selected. Therefore, the total number of instances used in the experiments is 75. In the experiments, gradient boosted trees (BTs) are trained by XGBoost (Chen and Guestrin 2016), where each BT consists of 25 to 40 trees of depth 3 to 5 per class.[3]

Competitors and Metrics. We compare the proposed approach (MARCO-S) against the original MARCO algorithms targeting AXp (MARCO-A) or CXp (MARCO-C) enumeration. We evaluate the FFA generated by these approaches by comparing it to the *exact* FFA through a variety of metrics, including errors, Kendall's Tau (Kendall 1938), rank-biased overlap (RBO) (Webber, Moffat, and Zobel 2010), and Kullback–Leibler (KL) divergence (Kullback and Leibler 1951). The *error* is quantified using Manhattan distance, i.e. the sum of absolute differences across all features in an instance. The comparison of feature ranking is assessed by Kendall's Tau and RBO, while feature distributions are compared by KL divergence.[4] Kendall's Tau and RBO produces values within the range of $[-1, 1]$ and $[0, 1]$, respectively, where higher values in both metrics indicate stronger agreement or similarity between the approximate FFA and the exact FFA. KL-divergence ranges from 0 to ∞, with the value approaching 0 reflecting better alignment between approximate FFA distribution and the exact FFA distribution. Note that if a feature in the exact FFA distribution holds a non-zero probability but is assigned a zero probability in the approximate one, the KL value becomes ∞. Finally, we also compare the efficiency of generating AXp's in the aforementioned approaches.

4.2 Integrated Results

This section compares the proposed approach against the original MARCO algorithms for both AXp enumeration and CXp enumeration within the examined datasets. Figures 1 to 5 illustrate the results of approximate FFA in terms of the five aforementioned metrics, namely, errors, Kendall's Tau, RBO, KL divergence, and the number of AXp's. These results are obtained by averaging values across all instances. Note that since KL-divergence is ∞ when there exists a feature in the exact FFA distribution that holds a non-zero probability but is assigned a zero probability in the approximate one, to address this issue we assign 0.5 as the KL-divergence value instead of ∞ in this case.[5] The average runtime to extract exact FFA is 3255.30s (from 2.15s to 29191.42s), 19311.87s (from 9.39s to 55951.57s), and 3509.50s (from 9.26s to 30881.55s) for MARCO-A, MARCO-C, and MARCO-S, respectively. Since the runtime required to get exact FFA varies, we normalized the runtime in each instance into [0, 1], where the longest time across three compared approaches in each instance is normalized to 1. Furthermore, we normalized the number of AXp's in each instance into the interval of [0, 1]. Finally, errors are also normalized into [0, 1] for each instance. Detailed experimental results can be found in the supplementary material.

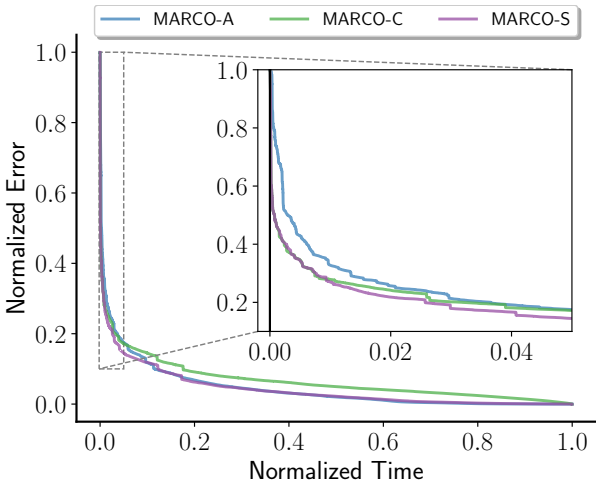


Figure 1: FFA error over time.

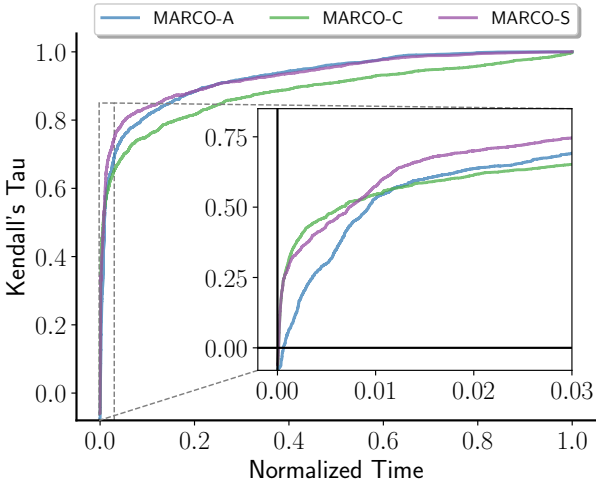


Figure 2: Kendall's Tau over time.

Errors. Figure 1 displays the average errors of approximate FFA across all instances over time. In the early pe-

[3]Test accuracy for *MNIST*, *PneumoniaMNIST*, *Sarcasm*, and *Disaster* datasets is 0.99, 0.83, 0.69, and 0.74, respectively.

[4]Kendall's Tau is a correlation coefficient metric that evaluates the ordinal association between two ranked lists, providing a similarity measure for the order of values, while RBO quantifies similarity between two ranked lists, considering both the order and depth of the overlap. KL-divergence measures the statistical distance between two probability distributions.

[5]According the experimental results we obtained, the maximum of non-infinity KL-divergence values is not greater than 0.5.

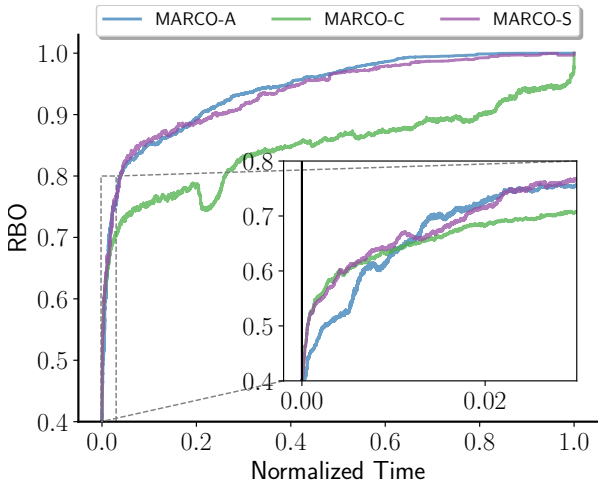


Figure 3: RBO over time.

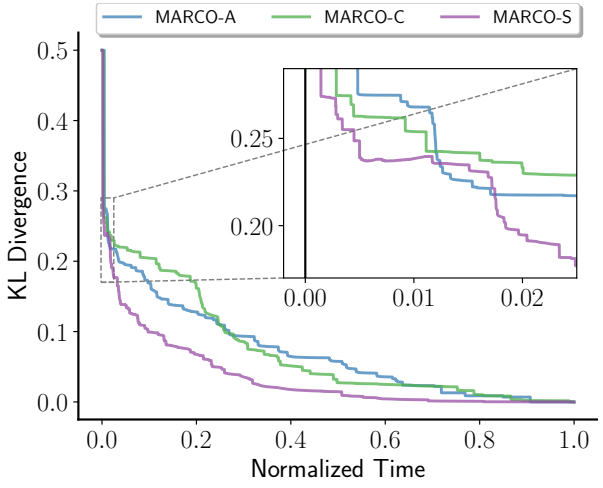


Figure 4: KL divergence over time.

riod, MARCO-C obtained more accurate approximate FFA in terms of errors compared with MARCO-A, while beyond the 0.04 time fraction, the latter surpasses the former and eventually achieves 0 error faster, which also indicates that MARCO-A requires less time to acquire the exact FFA. Motivated by the above observation, the proposed approach aims at replicating the "best of two worlds" during the FFA approximation process. Observe that MARCO-S commences with the MARCO algorithm targeting CXp's and so replicates the superior behavior of MARCO-C during the initial stage. Over time, MARCO-S triggers a switch criterion and transitions to targeting AXp's, thus emulating the behavior of the better competitor beyond the early stage, i.e. MARCO-A. Finally, MARCO-S acquires FFA with 0 error (i.e. exact FFA) as efficiently as MARCO-A.

Feature Ranking. The results of Kendall's Tau and RBO are depicted in Figures 2–3. Initially, MARCO-C outperforms MARCO-A in terms of both feature ranking metrics.

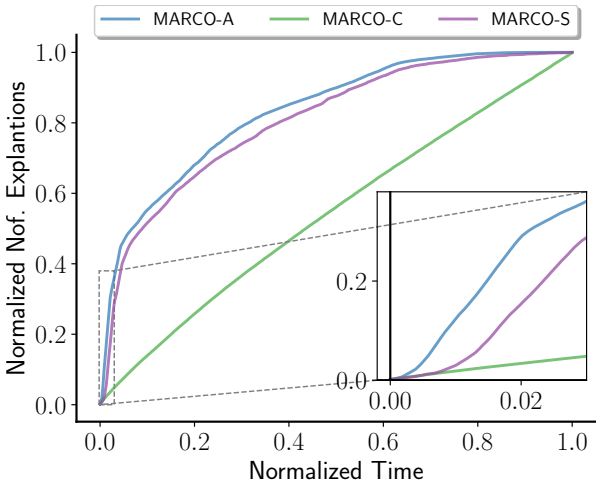


Figure 5: Number of explanations over time.

As time progresses, MARCO-A starts to surpass MARCO-C since 0.01 time fraction until the point of acquiring the exact FFA. Figures 2–3 demonstrate that initially MARCO-S manages to keep close to the better performing MARCO-C. When MARCO-A starts dominating, MARCO-S switches the target phase from CXp's to AXp's, replicating the superior performance displayed by MARCO-A.

Distribution. Figure 4 depicts the average results of KL divergence over time. Similar to feature ranking, MARCO-C is initially capable of computing an FFA distribution closer to the exact FFA distribution. Beyond the initial stage, MARCO-A exhibits the ability to generate closer FFA distribution. Once again, MARCO-S replicates the superior behavior between MARCO-A and MARCO-C in most of time. During the initial stage, MARCO-S reproduces the behavior of MARCO-C, and switch to target AXp's directly when the switch criterion is met. Surprisingly, MARCO-S *outperforms both competitors* throughout the entire time interval.

Number of AXp's. The average results of the normalized number of AXp's are illustrated in Figure 5. MARCO-A generates AXp's faster and finishes earlier than MARCO-C. Observe that the proposed approach MARCO-S is able to avoid the inferior performance between MARCO-A and MARCO-C throughout the process. Initially, MARCO-S replicates the behavior of MARCO-C and then switches to target AXp's to replicate the performance of MARCO-A.

Summary. MARCO-S can replicate the behavior of the superior competitor for most of the computation duration, leading to efficient and good approximation of FFA. As illustrated by Figures 1–4 in terms of FFA errors, Kendall's Tau, RBO, and KL divergence, starting from CXp enumeration and switching to AXp enumeration based on the criteria (6)–(7) successfully replicates the behavior of the winning configuration of MARCO, thus getting close of their *virtual best solver*. Although in terms of the number of AXp's shown in Figure 5 MARCO-A consistently outperforms MARCO-C, those AXp's are not diverse enough to

allow MARCO-A to beat MARCO-C in other relevant metrics. This is alleviated by MARCO-S, which manages to get enough diverse AXp's initially and then switches to target AXp's to catch up with the performance of MARCO-A.

5 Conclusions

Formal feature attribution (FFA) defines a crisp and easily understood notion of feature importance to a decision. Unfortunately for many classifiers and datasets it is challenging to compute exactly. As our paper demonstrates, it remains hard even if the set of CXp's is provided. Hence there is a need for *anytime* approaches to compute FFA. Surprisingly, using CXp enumeration to generate AXp's leads to fast good approximations of FFA, but in the longer term it is worse than simply enumerating AXp's. This paper shows how to combine the approaches by diligently switching the phase of enumeration, without losing information computed in the underlying MARCO enumeration algorithm. This gives a highly practical approach to computing FFA.

The proposed mechanism can be adapted to a multitude of other problems, e.g. in the domain of over-constrained systems, where one wants to collect a *diverse* set of minimal unsatisfiable subsets (MUSes) as the same minimal hitting set duality exists between MUSes and minimal correction sets (MCSes) (Birnbaum and Lozinskii 2003; Reiter 1987).

References

Amgoud, L.; and Ben-Naim, J. 2022. Axiomatic Foundations of Explainability. In Raedt, L. D., ed., *IJCAI*, 636–642.

Arenas, M.; Baez, D.; Barceló, P.; Pérez, J.; and Subercaseaux, B. 2021. Foundations of Symbolic Languages for Model Interpretability. In *NeurIPS*.

Arenas, M.; Barceló, P.; Orth, M. A. R.; and Subercaseaux, B. 2022. On Computing Probabilistic Explanations for Decision Trees. In *NeurIPS*.

Bailey, J.; and Stuckey, P. J. 2005. Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. In *PADL*, 174–186.

Belov, A.; Lynce, I.; and Marques-Silva, J. 2012. Towards efficient MUS extraction. *AI Commun.*, 25(2): 97–116.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability*.

Biradar, G.; Izza, Y.; Lobo, E.; Viswanathan, V.; and Zick, Y. 2023. Axiomatic Aggregations of Abductive Explanations. *CoRR*, abs/2310.03131.

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

Blanc, G.; Lange, J.; and Tan, L. 2021. Provably efficient, succinct, and precise explanations. In *NeurIPS*.

Boumazouza, R.; Alili, F. C.; Mazure, B.; and Tabia, K. 2021. ASTERYX: A model-Agnostic SaT-basEd appRoach for sYmbolic and score-based eXplanations. In *CIKM*, 120–129.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD*, 785–794.

Clark, P.; and Boswell, R. 1991. Rule Induction with CN2: Some Recent Improvements. In *EWSL*, 151–163.

Cooper, M. C.; and Marques-Silva, J. 2023. Tractability of explaining classifier decisions. *Artif. Intell.*, 316: 103841.

Darwiche, A.; and Marquis, P. 2021. On Quantifying Literals in Boolean Logic and Its Applications to Explainable AI. *J. Artif. Intell. Res.*, 72: 285–328.

Deng, L. 2012. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Ferreira, J.; de Sousa Ribeiro, M.; Gonçalves, R.; and Leite, J. 2022. Looking Inside the Black-Box: Logic-based Explanations for Neural Networks. In *KR*, 432–442.

Friedman, J. H. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5): 1189–1232.

Giunchiglia, E.; and Maratea, M. 2006. Solving Optimization Problems with DLL. In *ECAI*, 377–381.

Gorji, N.; and Rubin, S. 2022. Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints. In *AAAI*, 5660–5667.

Howard, A.; Devrishi; Culliton, P.; and Guo, Y. 2019. Natural Language Processing with Disaster Tweets.

Huang, X.; Cooper, M. C.; Morgado, A.; Planes, J.; and Marques-Silva, J. 2023. Feature Necessity & Relevancy in ML Classifier Explanations. In *TACAS (1)*, 167–186.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2022. Tractable Explanations for d-DNNF Classifiers. In *AAAI*, 5719–5728.

Huang, X.; Izza, Y.; and Marques-Silva, J. 2023. Solving Explainability Queries with Quantification: The Case of Feature Relevancy. In *AAAI*, 4123–4131.

Huang, X.; and Marques-Silva, J. 2023a. From Decision Trees to Explained Decision Sets. In *ECAI*, 1100–1108.

Huang, X.; and Marques-Silva, J. 2023b. The Inadequacy of Shapley Values for Explainability. *CoRR*, abs/2302.08160.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks. In *NIPS*, 4107–4115.

Hyafil, L.; and Rivest, R. L. 1976. Constructing Optimal Binary Decision Trees is NP-Complete. *Inf. Process. Lett.*, 5(1): 15–17.

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using MaxSAT for Efficient Explanations of Tree Ensembles. In *AAAI*, 3776–3785.

Ignatiev, A.; and Marques-Silva, J. 2021. SAT-Based Rigorous Explanations for Decision Lists. In *SAT*, 251–269.

Ignatiev, A.; Narodytska, N.; Asher, N.; and Marques-Silva, J. 2020. From Contrastive to Abductive Explanations and Back Again. In *AI*IA*, 335–355.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-Based Explanations for Machine Learning Models. In *AAAI*, 1511–1519.

Izza, Y.; Huang, X.; Ignatiev, A.; Narodytska, N.; Cooper, M. C.; and Marques-Silva, J. 2023a. On Computing Probabilistic Abductive Explanations. *International Journal of Approximate Reasoning*, 159.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2022. On Tackling Explanation Redundancy in Decision Trees. *J. Artif. Intell. Res.*, 75: 261–321.

Izza, Y.; Ignatiev, A.; Stuckey, P. J.; and Marques-Silva, J. 2023b. Delivering Inflated Explanations. *CoRR*, abs/2306.15272.

Izza, Y.; and Marques-Silva, J. 2021. On Explaining Random Forests with SAT. In *IJCAI*.

Kendall, M. G. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2): 81–93.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *KDD*, 1675–1684. ACM.

Liffiton, M.; and Malik, A. 2013. Enumerating Infeasibility: Finding Multiple MUSes Quickly. In *CPAIOR*, 160–175.

Liffiton, M. H.; Previti, A.; Malik, A.; and Marques-Silva, J. 2016. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2): 223–250.

Liffiton, M. H.; and Sakallah, K. A. 2008. Algorithms for Computing Minimal Unsatisfiable Subsets of Constraints. *J. Autom. Reasoning*, 40(1): 1–33.

Lundberg, S. M.; and Lee, S. 2017. A Unified Approach to Interpreting Model Predictions. In *NeurIPS*, 4765–4774.

Malfa, E. L.; Michelmore, R.; Zbrzezny, A. M.; Paoletti, N.; and Kwiatkowska, M. 2021. On Guaranteed Optimal Robust Explanations for NLP Models. In *IJCAI*, 2658–2665.

Marques-Silva, J. 2022a. Logic-Based Explainability in Machine Learning. In *Reasoning Web*, 24–104.

Marques-Silva, J. 2022b. Logic-Based Explainability in Machine Learning. *CoRR*, abs/2211.00541.

Marques-Silva, J. 2023. Disproving XAI Myths with Formal Methods - Initial Results. *CoRR*, abs/2306.01744.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2020. Explaining Naive Bayes and Other Linear Classifiers with Polynomial Time and Delay. In *NeurIPS*.

Marques-Silva, J.; Gerspacher, T.; Cooper, M. C.; Ignatiev, A.; and Narodytska, N. 2021. Explanations for Monotonic Classifiers. In *ICML*, 7469–7479.

Marques-Silva, J.; Heras, F.; Janota, M.; Previti, A.; and Belov, A. 2013. On Computing Minimal Correction Subsets. In *IJCAI*, 615–622.

Marques-Silva, J.; and Huang, X. 2023. Explainability is NOT a Game. *CoRR*, abs/2307.07514.

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *AAAI*, 12342–12350.

Marques-Silva, J.; and Ignatiev, A. 2023. No Silver Bullet: Interpretable ML Models Must Be Explained. *Frontiers in Artificial Intelligence*, 6: 1–15.

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

Misra, R.; and Arora, P. 2023. Sarcasm Detection using News Headlines Dataset. *AI Open*, 4: 13–18.

Misra, R.; and Grover, J. 2021. *Sculpting Data for ML: The first act of Machine Learning*. ISBN 9798585463570.

Nair, V.; and Hinton, G. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*, 807–814.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 8024–8035.

Previti, A.; and Marques-Silva, J. 2013. Partial MUS Enumeration. In *AAAI*. AAAI Press.

Provan, J. S.; and Ball, M. O. 1983. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4): 777–788.

Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.*, 32(1): 57–95.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*, 1135–1144.

Rivest, R. L. 1987. Learning Decision Lists. *Mach. Learn.*, 2(3): 229–246.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A Symbolic Approach to Explaining Bayesian Network Classifiers. In *IJCAI*, 5103–5111.

Vadhan, S. 2001. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31(2): 398–427.

Valiant, L. G. 1979. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2): 189–201.

Wäldchen, S.; MacDonald, J.; Hauch, S.; and Kutyniok, G. 2021. The Computational Complexity of Understanding Binary Classifier Decisions. *J. Artif. Intell. Res.*, 70: 351–387.

Webber, W.; Moffat, A.; and Zobel, J. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4): 1–38.

Yang, J.; Shi, R.; Wei, D.; Liu, Z.; Zhao, L.; Ke, B.; Pfister, H.; and Ni, B. 2023. MedMNIST v2-A large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Scientific Data*, 10(1): 41.

Yu, J.; Ignatiev, A.; and Stuckey, P. J. 2023a. From Formal Boosted Tree Explanations to Interpretable Rule Sets. In *CP*, volume 280 of *LIPIcs*, 38:1–38:21.

Yu, J.; Ignatiev, A.; and Stuckey, P. J. 2023b. On Formal Feature Attribution and Its Approximation. *CoRR*, abs/2307.03380.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; Narodytska, N.; and Marques-Silva, J. 2023. Eliminating The Impossible, Whatever Remains Must Be True. In *AAAI*, 4123–4131.

Appendices

A Detailed Experimental Results

This appendix compares the proposed approach (MARCO-S) against the original MARCO algorithms for targeting AXp's (MARCO-A) and CXp's (MARCO-C) in each considered dataset, namely *MNIST-1vs3*, *MNIST-1vs7*, *PneumoniaMNIST*, *Sarcasm*, and *Disaster*. Figures 6 to 10 depict the average results of the comparison between the approximate FFA and the exact FFA using 5 aforementioned metrics, namely, errors, Kendall's Tau, RBO, KL divergence, and the number of AXp's. The results are acquired by averaging the values across 15 selected instances in a dataset. The average runtime of the three approaches to acquire the exact FFA in each datadset is summarized in Table 1. Once again, as KL-divergence is ∞ when a feature in the exact FFA distribution holds a non-zero probability but is assigned a zero probability in the approximate one, we assign a value of 0.5 to KL-divergence in stead of ∞. Note that the maximum of non-infinity KL-divergence values is not greater than 0.5 in the experimental results we acquired. The same normalization technique is applied to runtime, errors, and the number of AXp's as described in Section 4.2. We normalized these three metrics across the three compared approaches in each instance into [0, 1].

Errors. The average errors of approximate FFA across selected instances in each dataset over time are shown in Figure 6. In the initial stage, compared with MARCO-A, MARCO-C achieves a more accurate approximation of FFA in terms of errors in *MNIST-1vs3*, *MNIST-1vs7*, *PneumoniaMNIST*, and *Disaster* datasets while beyond the early phase MARCO-A starts to outperform MARCO-C and eventually achieves 0 error faster in these datasets, indicating that MARCO-A is able to spend less time to obtain the exact FFA. However, MARCO-A consistently outperforms MARCO-C in the *Sarcasm* dataset. Inspired by these observations, the proposed approach targets replicating the "best of two world" in the process of approximating FFA. Observe that MARCO-S starts with the MARCO algorithm of CXp enumeration and thus emulates the better behavior displayed by MARCO-C during the initial phase, as shown in Figures 6a, 6b, 6c, and 6e. Over time, MARCO-S meets a switch criterion and transitions to targeting AXp enumeration, so replicating the behavior of the superior competitor beyond the initial stage. As for the *Sarcasm* dataset in Figure 6d, MARCO-S replicates the behavior of MARCO-C and then switches to target AXp's to replicate the performance of MARCO-A, avoiding the inferior performance between MARCO-A and MARCO-C. Finally, in all datasets MARCO-S is able to obtain FFA with 0 error (i.e. exact FFA) as efficient as MARCO-A.

Feature Ranking. Figures 7 and 8 illustrate the results of Kendall's Tau and RBO in each dataset. Observe that MARCO-C exhibits better performance than MARCO-A during the initial stage for both feature ranking metrics in *MNIST-1vs3*, *MNIST-1vs7*, *PneumoniaMNIST*, and *Disaster* datasets. Over time, MARCO-A gradually overtakes MARCO-C until reaching the point of obtaining the exact FFA in these datasets. Nevertheless, in the *Sarcasm* dataset, MARCO-A consistently displays the superior performance during the entire period. These figures demonstrate that MARCO-S is capable of maintaining close to the superior performance exhibited by MARCO-C during the initial phase in all datasets except for *Sarcasm*. When MARCO-A begins to outperform MARCO-C, MARCO-S transitions the target phase from CXp's to AXp's, replicating the superior performance demonstrated by MARCO-A. In the *Sarcasm* dataset, switching from CXp enumeration to AXp enumeration beyond the initial stage avoids reproducing the inferior performance between MARCO-A and MARCO-C in most of time.

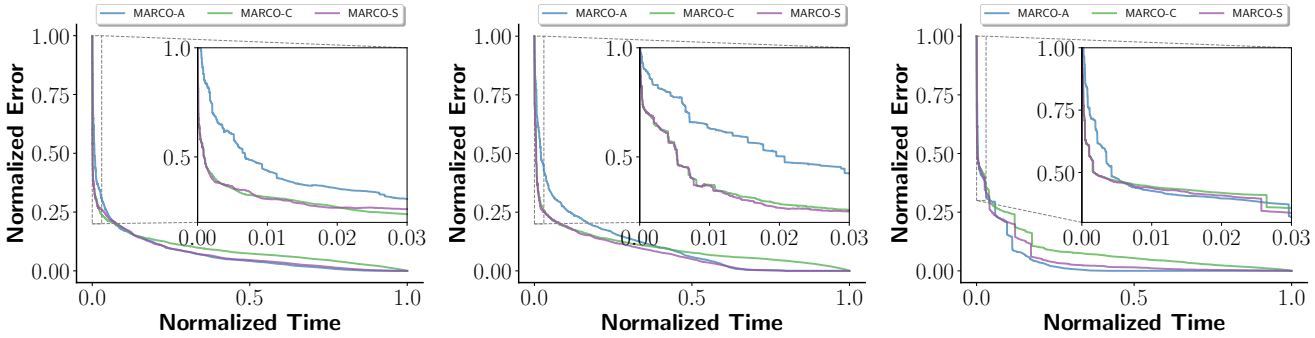
Distribution. The average results of KL divergence over time are depicted in Figure 9. MARCO-C is initially capable of generating an FFA distribution more similar to the exact FFA distribution in *MNIST-1vs3* and *MNIST-1vs7* datasets. Afterwards, MARCO-A exhibits the ability to compute FFA distribution more similar to the exact FFA attribution. However, MARCO-A consistently generate a closer FFA distribution than MARCO-C in the other datasets. Once again, MARCO-S emulates the superior behavior between MARCO-A and MARCO-C in most of time or avoids replicating the inferior performance for a long time due to the switch mechanism. MARCO-S initially reproduces the behavior of MARCO-C, and switches to target AXp's when meeting the switch criterion. Surprisingly, MARCO-S exhibits the best performance among the competitors in most of the entire time interval in *MNIST-1vs3* and *MNIST-1vs7*.

Number of AXp's. Figure 10 shows the normalized number of AXp's in each dataset. Observe that compared with MARCO-A, MARCO-C is capable of generating AXp's more efficiently during the early stage in *MNIST-1vs3* and *MNIST-1vs7* datasets, but MARCO-A starts to outperform MARCO-C as time progresses. In the other three datasets, MARCO-A achieves similar or better performance in the entire process. As demonstrated by Figure 10, the proposed approach MARCO-S is able to avoid the inferior performance between MARCO-A and MARCO-C for most of the duration. Initially, MARCO-S emulates the behavior of MARCO-C, and transitions to target AXp's to replicate the performance of MARCO-A afterwards, preventing the reproduction of inferior performance. Remarkably, in the *MNIST-1vs7* dataset, MARCO-S emerges as the best-performing approach for most of time.

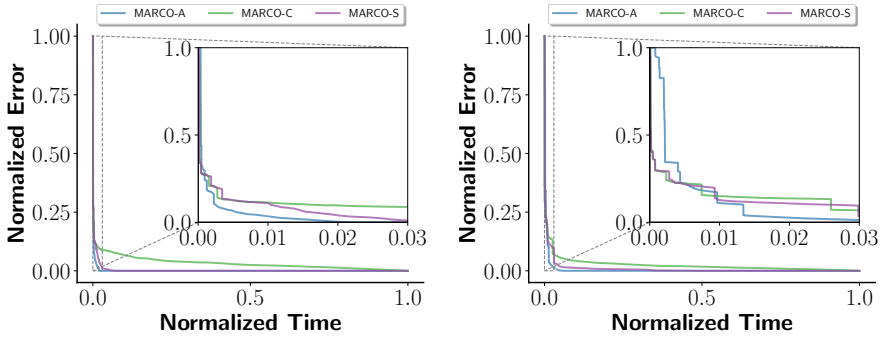
Summary. In alignment with the results presented in Section 4, MARCO-S is able to replicate the behavior of the superior competitor between MARCO-A and MARCO-C throughout most of the computation period, resulting in fast and good approximation of FFA. As depicted in Figures 6 to 9, which display the results of FFA errors, Kendall's Tau, RBO, and KL divergence, beginning with CXp enumeration and subsequently transitioning to AXp enumeration based on criteria 6–7 can reproduce the performance of the winning MARCO configuration, and therefore this approach enables the model to closely approach their virtual

Table 1: Average runtime(s) in each dataset.

Approach	MNIST-1vs3	MNIST-1vs7	Pneumoniamnist	Sarcasm	Disaster
MARCO-A	9350.79	2844.15	1972.41	669.91	1439.24
MARCO-C	14787.22	7412.40	8343.55	33391.29	32624.89
MARCO-S	9970.55	2959.15	2016.49	975.31	1626.01



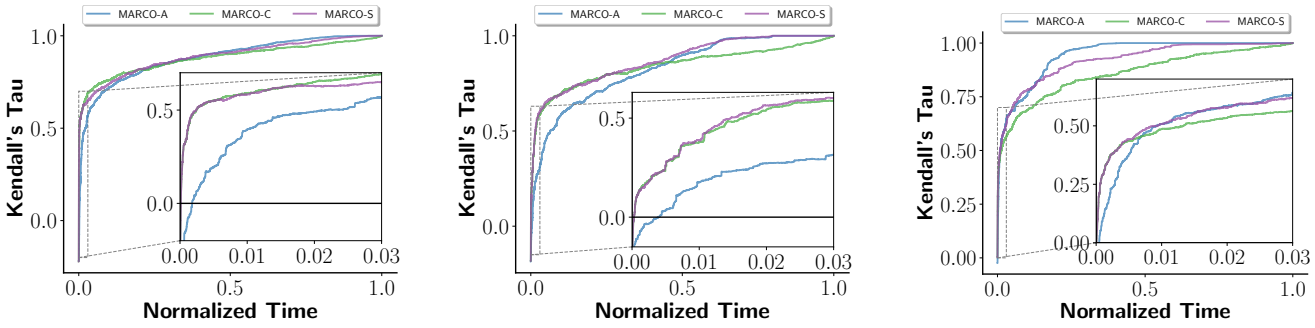
(a) MNIST-1vs3 (b) MNIST-1vs7 (c) PneumoniaMNIST



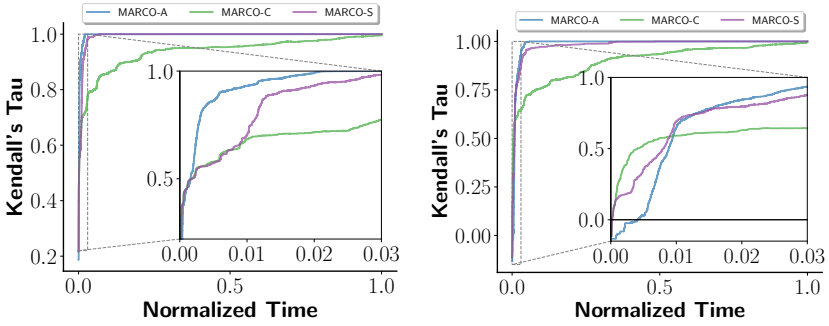
(d) Sarcasm (e) Disaster

Figure 6: FFA error over time in each dataset.

best solver. Although MARCO-A consistently exhibits better performance than MARCO-C in *PneumoniaMNIST*, *Sarcasm* and *Disaster* datasets in terms of the number of AXp's depicted in Figure 10, the lack of diversity among these AXp's prevents MARCO-A from outperforming MARCO-C in other relevant metrics. This diversity issue is mitigated by MARCO-S, which effectively addresses this by initially obtaining a sufficiently diverse set of AXp's and subsequently transitioning to targeting AXp's, thereby matching the performance of MARCO-A.

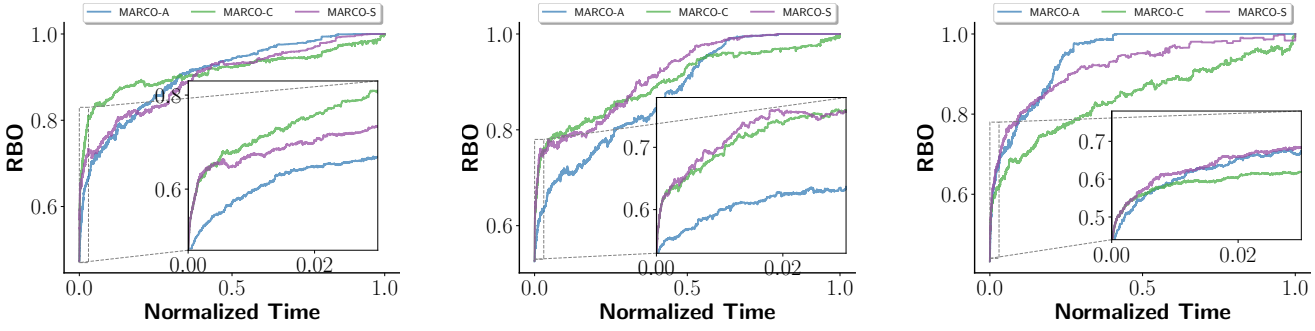


(a) MNIST-1vs3 (b) MNIST-1vs7 (c) PneumoniaMNIST

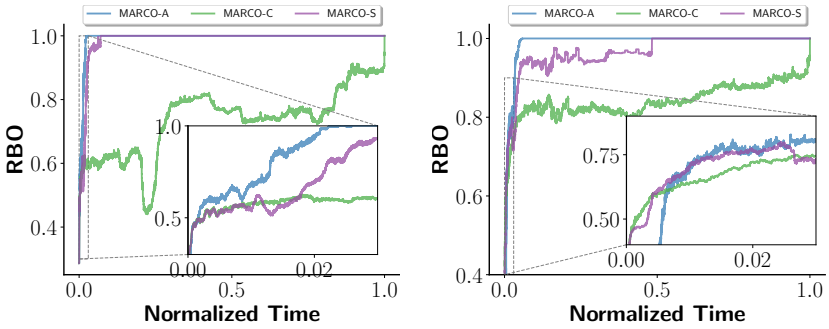


(d) Sarcasm (e) Disaster

Figure 7: Kendall's Tau over time in each dataset.

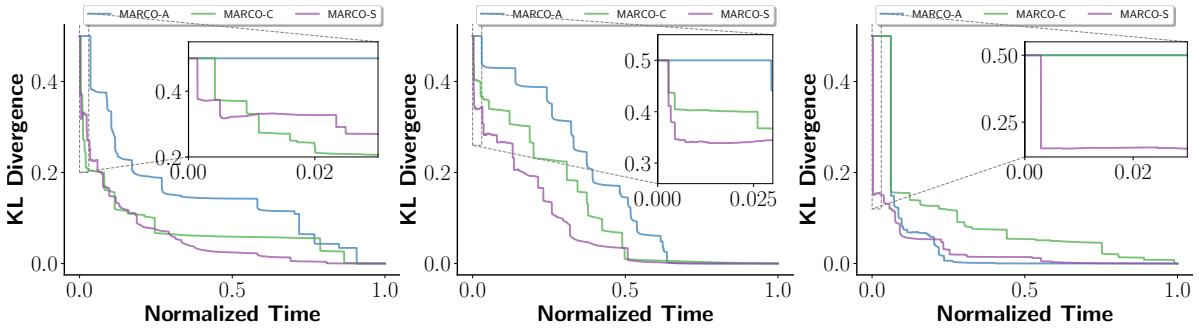


(a) MNIST-1vs3 (b) MNIST-1vs7 (c) PneumoniaMNIST

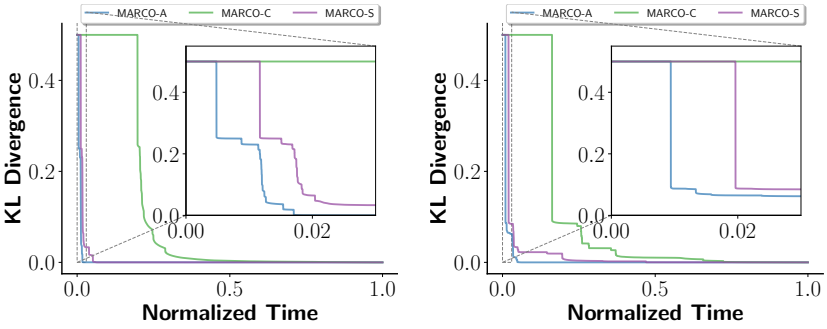


(d) Sarcasm (e) Disaster

Figure 8: RBO over time in each dataset.

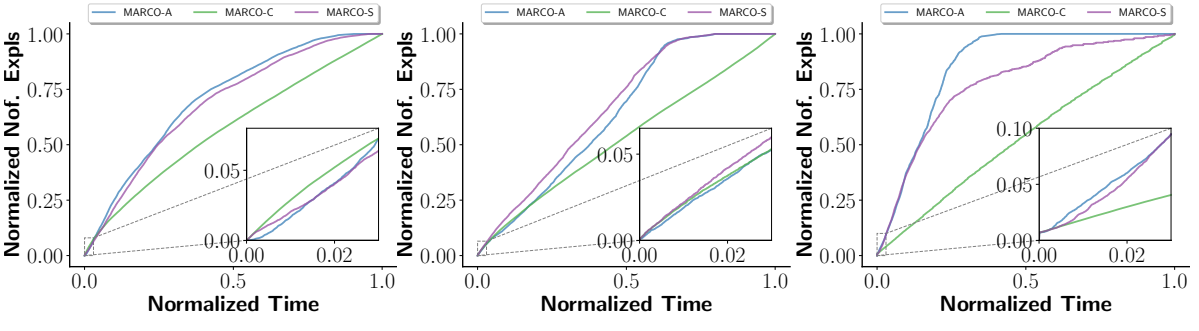


(a) MNIST-1vs3 (b) MNIST-1vs7 (c) PneumoniaMNIST

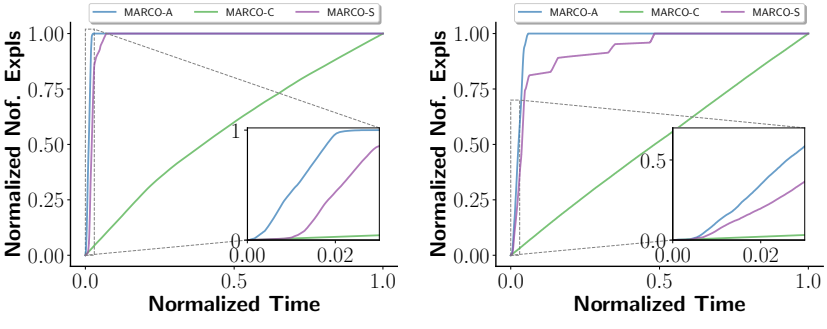


(d) Sarcasm (e) Disaster

Figure 9: KL-divergence over time in each dataset.



(a) MNIST-1vs3 (b) MNIST-1vs7 (c) PneumoniaMNIST



(d) Sarcasm (e) Disaster

Figure 10: Number of explanations over time in each dataset.

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

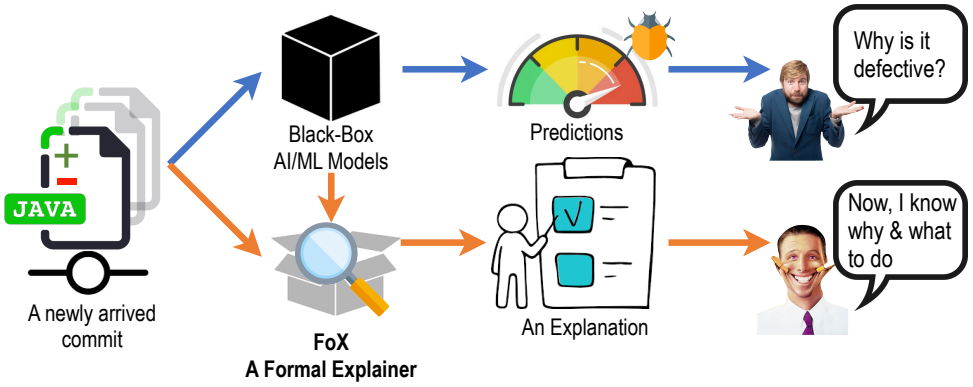


Fig. 1. A motivation on the need of Explainable AI for Just-In-Time defect prediction. Currently, practitioners and domain experts often ask many *why?* questions (e.g., why a commit is predicted as defective) and *how?* questions (e.g., how should developers mitigate the risk) [23].

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

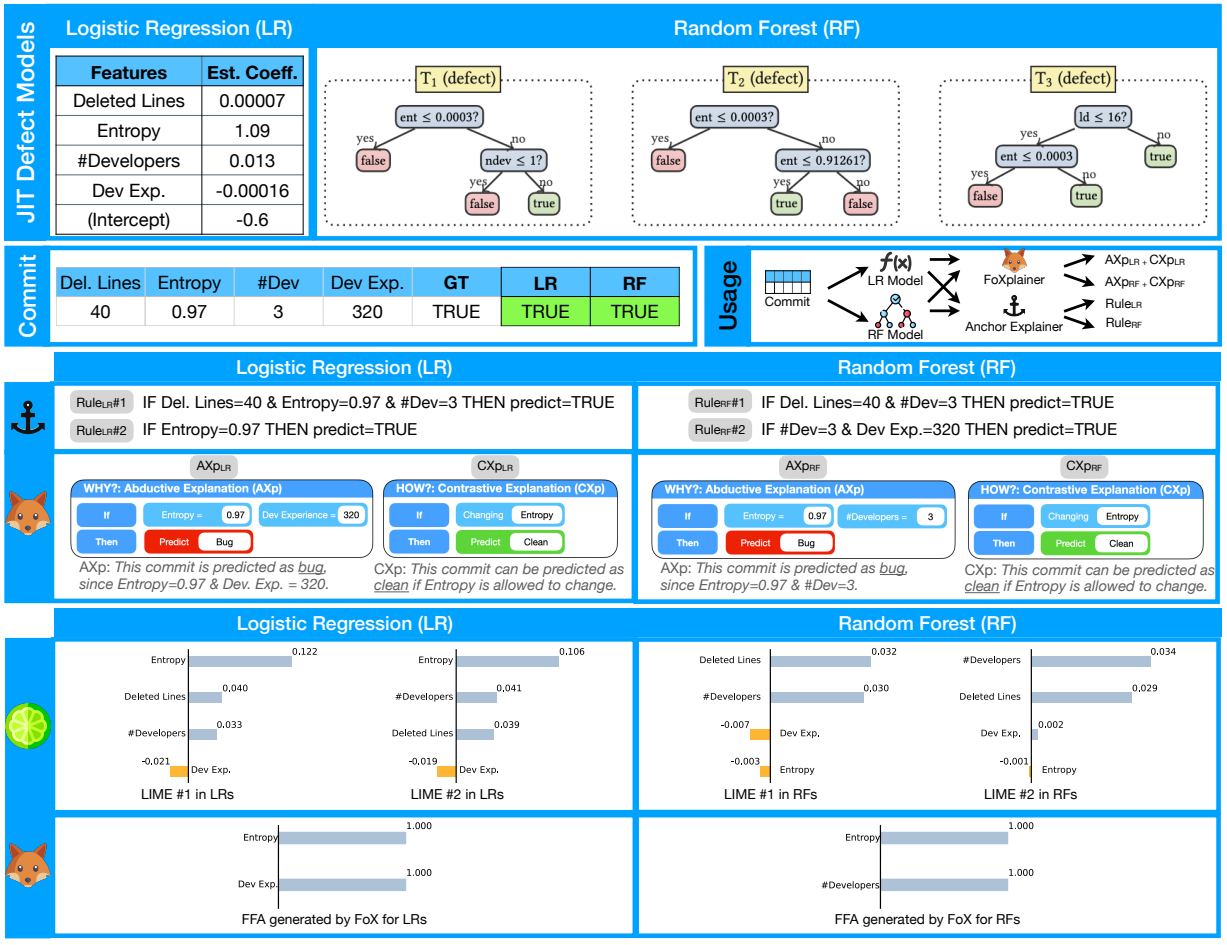


Fig. 2. An illustrative example of explanations generated by FoX, Anchor [46], and LIME [45] for Just-In-Time defect prediction. FFA refers to Formal Feature Attribution (see Definition 3).

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.

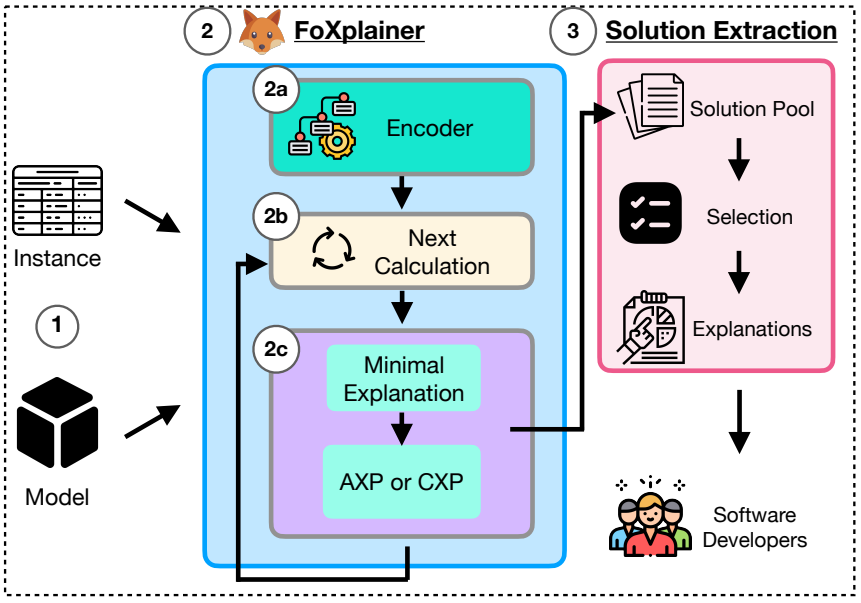


Fig. 3. An overview of FoX to extract explanations. Given an instance to be explained and an ML model, FoX consists of 3 main components: (1) formal encoding, (2) explanation enumeration, and (3) solution pool with the ability to select explanations given user preferences. FoX extracts two types of formal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

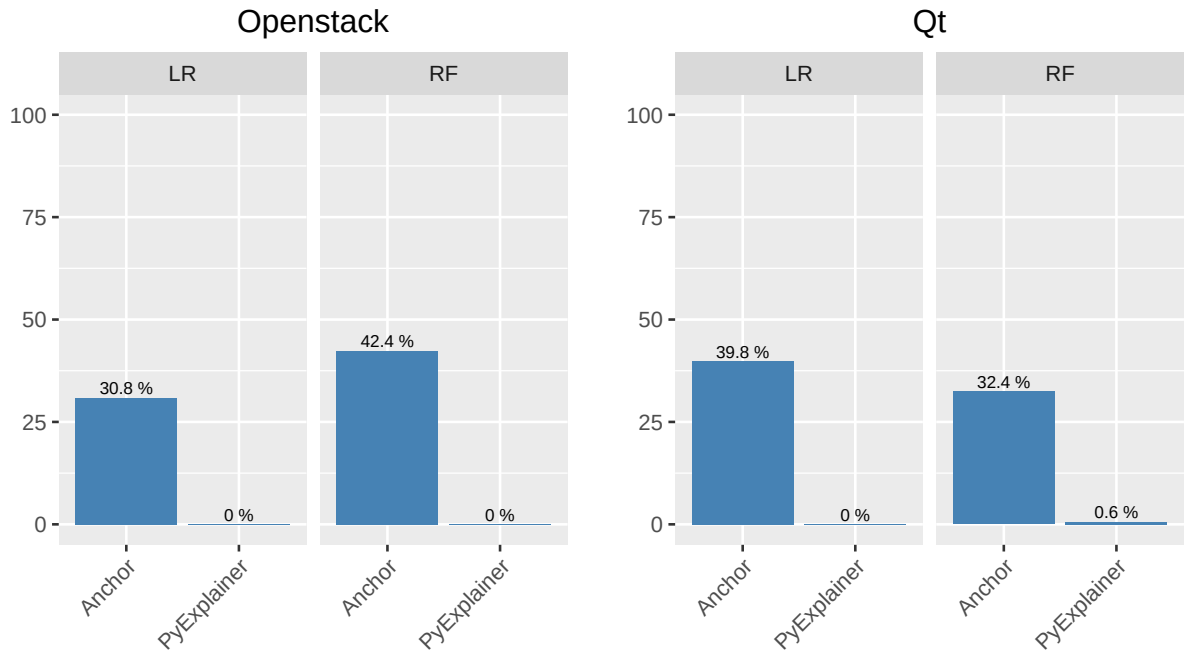


Fig. 6. (RQ2) Robustness of feature selection methods. FoX achieves perfect robustness (i.e., 100%) on both datasets and thus is not shown in the bar chart. (↗) Higher Robustness = Better.

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

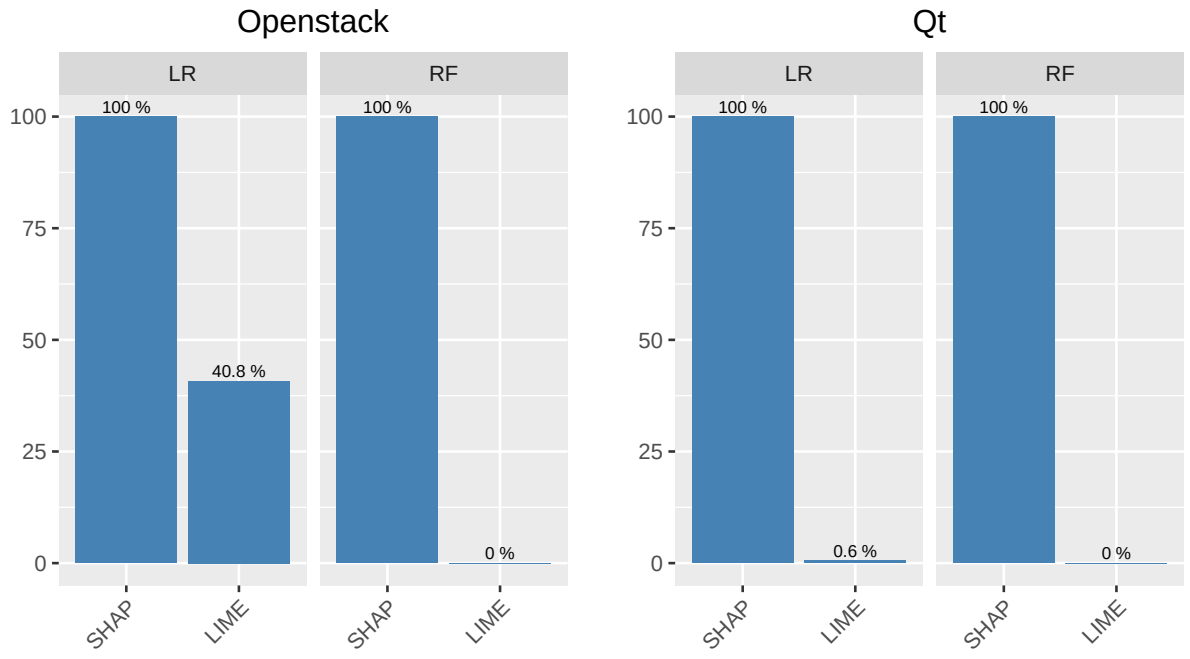


Fig. 7. (RQ2) Robustness of feature attribution methods. FoX achieves perfect robustness (i.e., 100%) on both datasets and thus is not shown in the bar chart. (↗) Higher Robustness = Better.

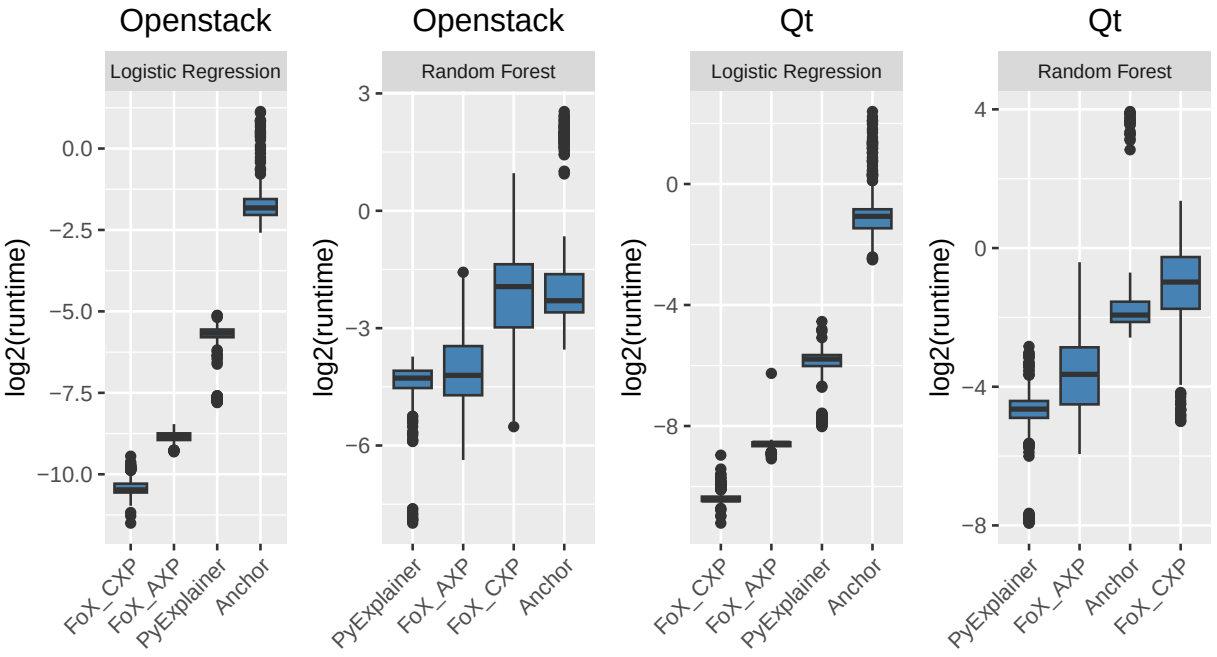


Fig. 8. (RQ3) Runtime of feature selection methods. (↘) Lower Runtime = Better.

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

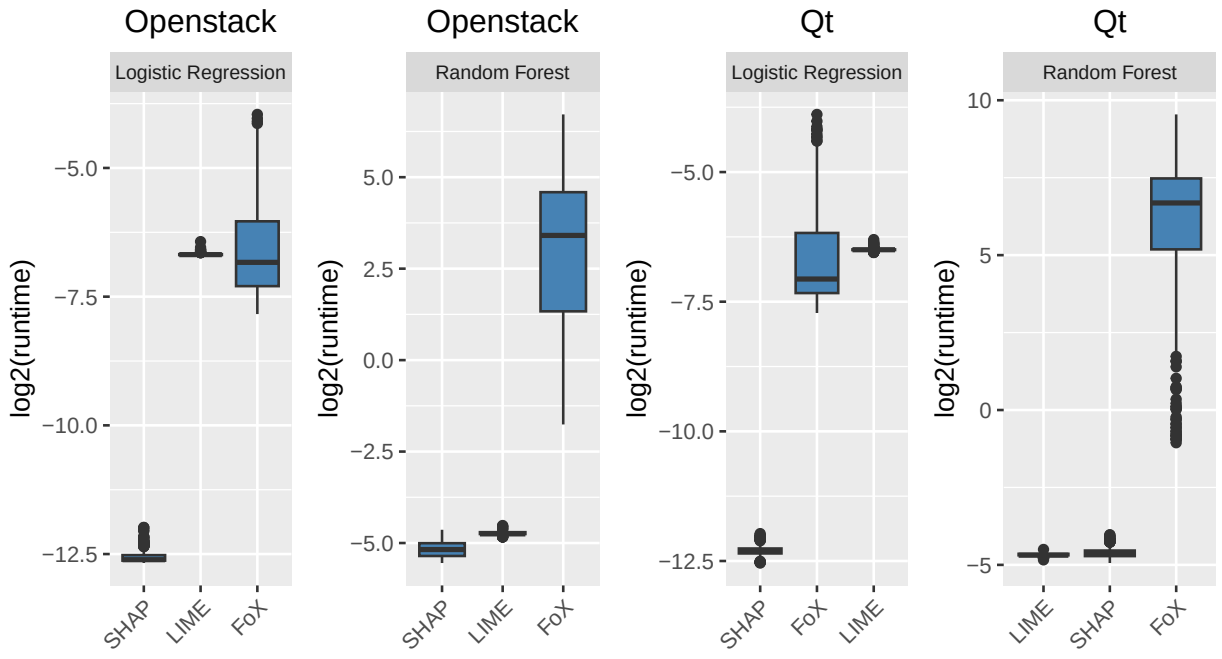


Fig. 9. (RQ3) Runtime of feature feature attribution methods. (↘) Lower Runtime = Better.

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

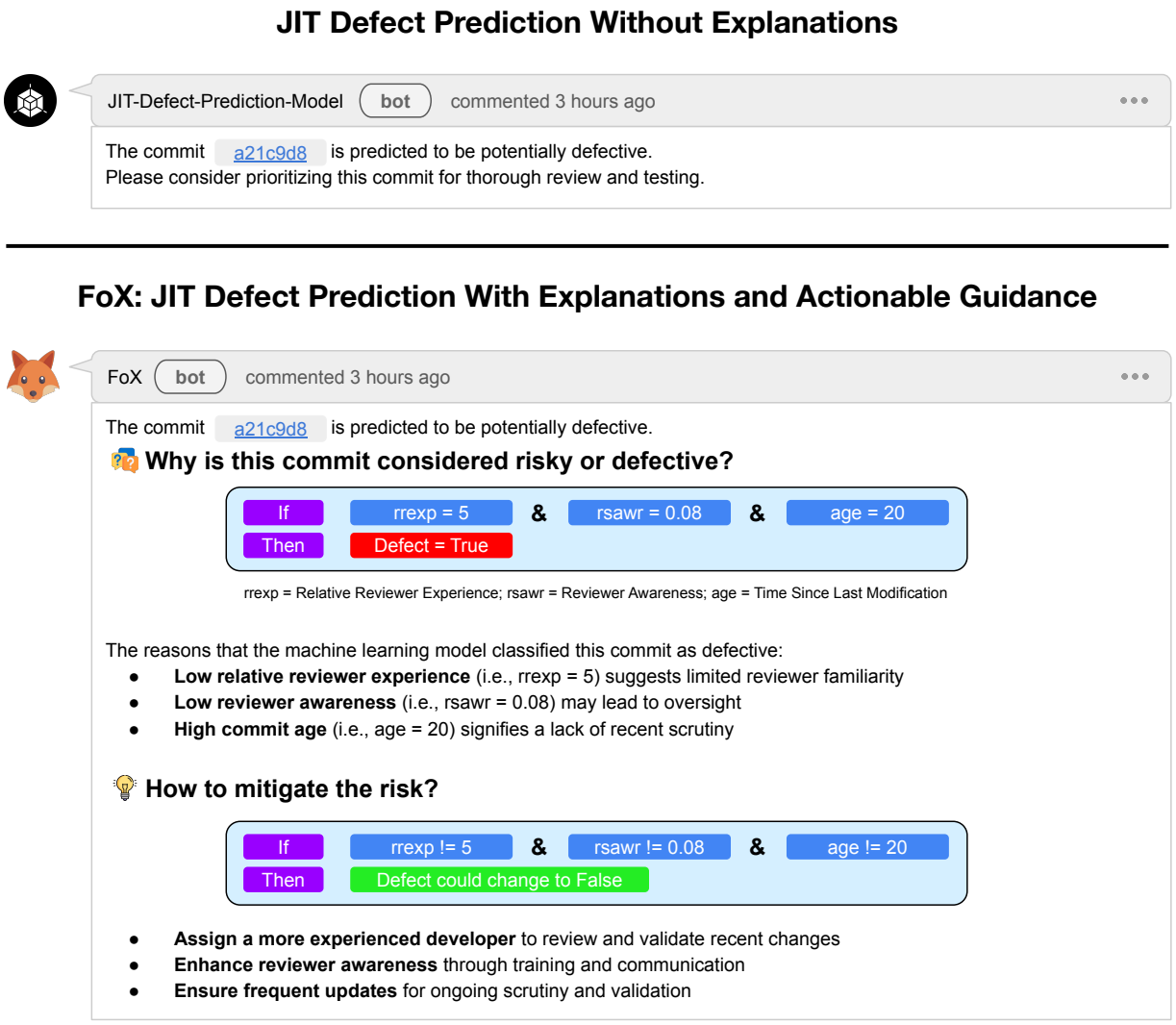


Fig. 10. The upper part shows the Just-In-Time (JIT) defect prediction without any explanations, as presented in Part II-A of our user study. The lower part shows FoX, which presents the JIT defect prediction with explanations and actionable guidance, as presented in Part II-B of our user study.

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

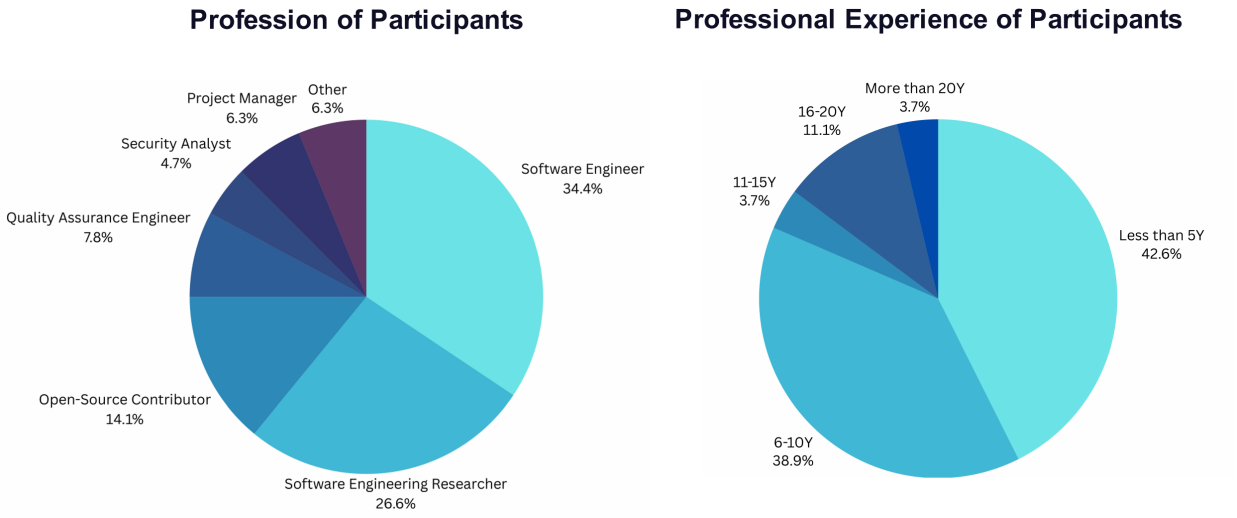


Fig. 11. The demographics of our survey participants in terms of their profession and professional experience.

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,*

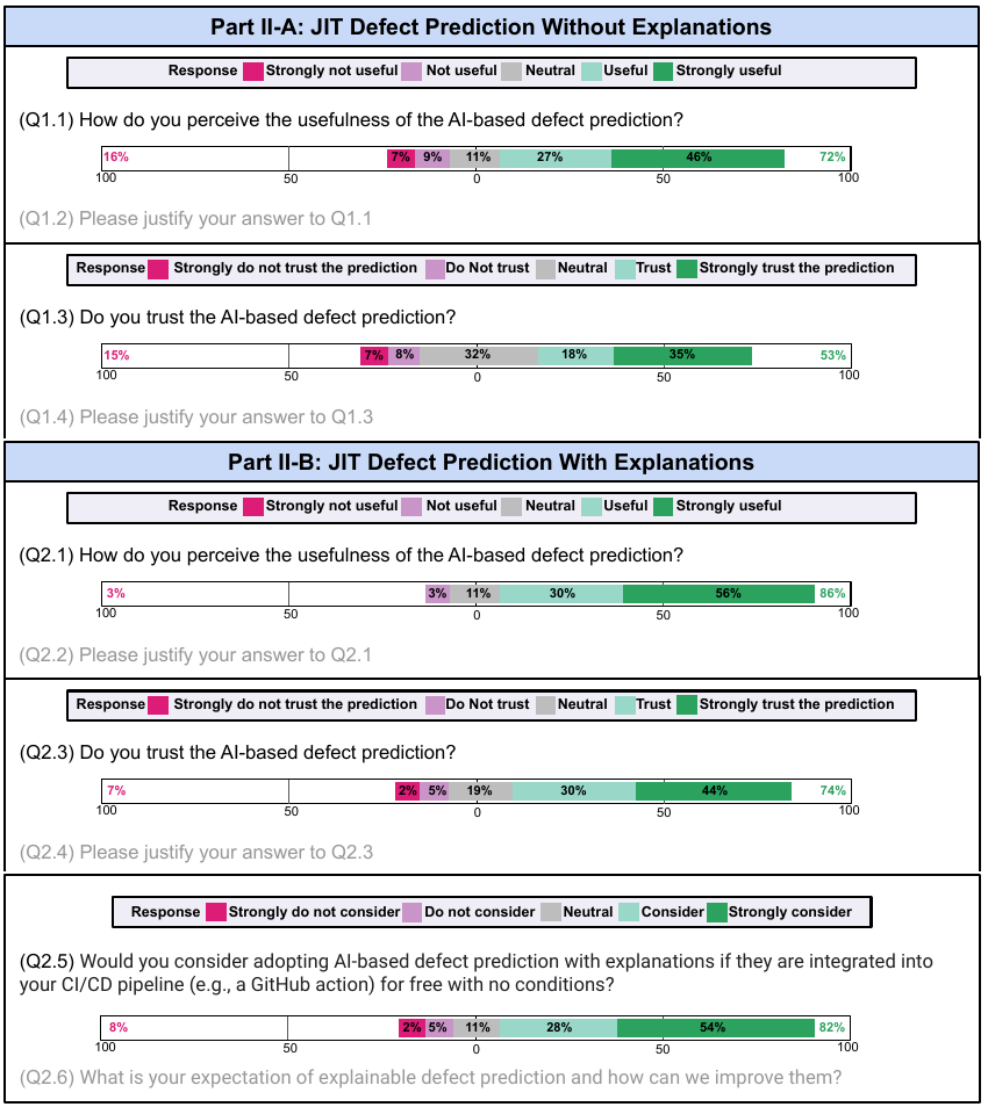


Fig. 12. (Survey Results) A summary of the survey questions (i.e., Part II-A: Q1.1-Q1.4 and Part II-B: Q2.1-2.6) and the results obtained from 54 participants.

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.