# Recent advances in formal explainability

Xuanxiang Huang


# THÈSE

**En vue de l'obtention du**

## DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :** *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

---

**Présentée et soutenue le** *Le 14 décembre 2023* **par :**

**Xuanxiang HUANG**

**Recent Advances in Formal Explainability**

---

**JURY**

| | | |
|---|---|---|
| Claire Pagetti | ONERA Toulouse | Présidente |
| Saddek Bensalem | Université Grenoble Alpes | Rapporteur |
| Felip Manyà | CSIC—IIIA | Rapporteur |
| Jérôme Lang | CNRS Paris—LAMSADE | Examinateur |
| Francesco Ricca | Università della Calabria | Examinateur |
| João Marques-Silva | CNRS Toulouse—IRIT | Directeur de thèse |
| Nicholas Asher | CNRS Toulouse—IRIT | Co-directeur de thèse |

---

**École doctorale et spécialité :**
*MITT : Informatique*

**Unité de Recherche :**
*Institut de Recherche en Informatique de Toulouse (IRIT)*

**Directeur(s) de Thèse :**
*João Marques-Silva* et *Nicholas Asher*

**Rapporteurs :**
*Saddek Bensalem* et *Felip Manyà*

## Acknowledgments

I want to express my sincere gratitude to my supervisor, Prof. João Marques-Silva, for his support and guidance during my PhD studies. He has been dedicated to advancing the development of formal explainability in the past few years, and I feel deeply grateful to have completed my PhD under his guidance. I am also thankful to my co-advisor, Prof. Nicholas Asher, for his valuable advice and encouragement throughout this journey. I want to extend my gratitude to Dr. Yacine Izza for his frequent discussions and guidance as I started my PhD studies.

Furthermore, I am grateful to Prof. Martin C. Cooper, Dr. Alexey Ignatiev, Dr. António Morgado, and Associate Professor Jordi Planes for the great research collaborations we had. Their expertise and contributions have enriched the scope and impact of my work, and I am thankful for the opportunity to collaborate with such distinguished scholars.

Tremendous thanks to all my friends and colleagues, including Zhenyu Bai, Hao Hu, Arnaud Lequen, Xinghan Liu, and Léo Saulières, for the great moments we shared. I would also like to extend special thanks to Xinghan Liu, our frequent discussions on study and research questions have been immensely beneficial to my academic growth.

I am grateful to ANITI for providing funding for this doctoral research. Additionally, I thank Université de Toulouse, Université Paul Sabatier, and IRIT for providing the resources, facilities, and opportunities that have allowed me to pursue this research. I want to express my gratitude to the members of my thesis committee for their valuable feedback and insightful suggestions. I acknowledge the countless researchers, scholars, and authors whose work has inspired my research. Their contributions to the field have been instrumental in shaping my understanding and approach.

Last, I want to thank my parents for their constant love and understanding. Their trust in me has always been a driving force behind this endeavor.

Xuanxiang Huang
Toulouse, December 2023

## Abstract

In the past decade, monumental breakthroughs in Artificial Intelligence (AI), particularly in Machine Learning (ML), have shaped various fields. The widespread integration of complex ML models into various aspects of daily life, including healthcare, finance, and transportation, has created an urgent demand for transparency and accountability in ML systems. Unfortunately, understanding the rationale behind the decisions made by the most advanced ML models is challenging for humans. This lack of transparency can lead to several critical issues, including bias and unfair outcomes, jeopardized safety in safety-critical applications, and regulatory non-compliance. In response to these challenges, the field of eXplainable AI (XAI) has emerged as a crucial research domain. XAI aims to bridge the gap between the inner workings of AI/ML systems and human understanding to establish trustworthy AI. Its significance is underscored by guidelines, recommendations, and regulations from influential bodies (e.g. European Union, UNESCO). XAI offers several benefits, including enhancing trust in AI systems, mitigating biases, improving safety in autonomous vehicles, among others.

However, most of XAI approaches that have gained the most attention are commonly known as model-agnostic methods (e.g. LIME, SHAP). More importantly, model-agnostic XAI methods offer no guarantees of rigor and may produce logically unsound explanations. The limitations inherent in these non-formal XAI approaches pose a substantial challenge to the dependability of model-agnostic explanations, particularly in contexts classified as high-risk or safety-critical.

As an alternative, there is a growing trend in the application of automated reasoning techniques for explaining and verifying ML models, broadly known as formal XAI. This approach is logic-based and model-specific, designed to deliver formal explanations. These formal explanations are characterized by their rigor and provability, distinguishing them from non-formal XAI methods. The thesis delves into formal XAI methods, contributing to the development of formal explainability and offering insights into the evolving landscape of XAI research. The thesis also addresses various aspects of formal explanations for machine learning classifiers. Firstly, the thesis identifies the conditions enabling the computation of formal explanations in polynomial time for a class of tractable graph models (e.g. decision trees and d-DNNF circuits). It also provides practical and efficient methods for enumerating these explanations. Secondly, the thesis offers practical solutions for transforming decision trees into explained decision sets, enhancing their explainability. Thirdly, the thesis investigates the computational complexity of specific explainability queries across various classifiers (e.g. random forests), accompanied by practical and efficient problem-solving approaches. Lastly, the thesis compares SHAP scores with formal explanations and reveals some issues associated with SHAP scores in the field of explainability.


**Keywords: explainable AI, formal explainability, automated reasoning**

## Résumé

Au cours de la dernière décennie, des avancées monumentales dans le domaine de l'Intelligence Artificielle (IA), en particulier dans l'Apprentissage Automatique (Machine Learning, ML), ont façonné divers domaines. L'intégration généralisée de modèles ML complexes dans divers aspects de la vie quotidienne, notamment dans les domaines de la santé, de la finance et des transports, a créé une demande urgente de transparence et de responsabilité dans les systèmes de ML. Malheureusement, comprendre la justification des décisions prises par les modèles de ML les plus avancés est un défi pour les humains. Ce manque de transparence peut entraîner plusieurs problèmes critiques, notamment des biais et des résultats injustes, une mise en danger de la sécurité dans des applications critiques, et une non-conformité réglementaire.

En réponse à ces défis, le domaine de l'Intelligence Artificielle Explicable (eXplainable AI, XAI) a émergé en tant que domaine de recherche crucial. Le XAI vise à combler l'écart entre le fonctionnement interne des systèmes d'IA/ML et la compréhension humaine pour établir une IA digne de confiance. Son importance est soulignée par les directives, recommandations et réglementations de grandes organisations (par exemple, l'Union Européenne, l'UNESCO). Le XAI offre plusieurs avantages, notamment l'amélioration de la confiance dans les systèmes d'IA, la réduction des biais, l'amélioration de la sécurité dans les véhicules autonomes, entre autres.

Cependant, la plupart des approches de XAI qui ont attiré le plus d'attention sont communément appelées méthodes agnostiques au modèle (par exemple, LIME, SHAP). Plus important encore, les méthodes de XAI agnostiques au modèle ne garantissent pas de rigueur et peuvent produire des explications illogiques. Les limitations inhérentes à ces approches de XAI non formelles posent un défi substantiel à la fiabilité des explications agnostiques au modèle, en particulier dans des contextes classés comme à haut risque ou critiques pour la sécurité.

En alternative, une tendance croissante se dessine dans l'application de techniques de raisonnement automatisé pour expliquer et vérifier les modèles ML, largement connue sous le nom d'XAI formel. Cette approche est basée sur la logique et spécifique au modèle, conçue pour fournir des explications formelles. Ces explications formelles se caractérisent par leur rigueur et leur démontrabilité, les distinguant des méthodes de XAI non formelles. La thèse se penche sur les méthodes d'XAI formel, contribuant au développement de l'explicabilité formelle et offrant des aperçus sur l'évolution de la recherche en XAI. La thèse aborde également divers aspects des explications formelles pour les classificateurs d'apprentissage automatique. Premièrement, la thèse identifie les conditions permettant le calcul d'explications formelles en temps polynomial pour une classe de modèles graphiques gérables (par exemple, les arbres de décision et les circuits d-DNNF). Elle fournit également des méthodes pratiques et efficaces pour énumérer ces explications. Deuxièmement, la thèse propose des solutions pratiques pour transformer les arbres de décision en ensembles de décisions expliqués, améliorant leur explicabilité.

Troisièmement, la thèse examine la complexité informatique des requêtes spécifiques en matière d'explicabilité au sein de différents classificateurs (par exemple, les forêts aléatoires), accompagnée d'approches pratiques et efficaces pour résoudre ces problèmes. Enfin, la thèse compare les scores de SHAP aux explications formelles et met en lumière certaines problématiques liées aux scores de SHAP dans le domaine de l'explicabilité.

**Mots-clés: IA explicable, explicabilité formelle, raisonnement automatisé**

# Contents

# Notations

## Symbols

| | |
|---|---|
| $\top$ | True |
| $\bot$ | False |
| $\wedge$ | Conjunction, AND |
| $\vee$ | Disjunction, OR |
| $\neg$ | Negation, NOT |
| $\rightarrow$ | Material implication |
| $\models$ | Entailment |
| $X = \{x_1, \ldots, x_n\}$ | Propositional atoms |
| $\tau$ | Term (conjunction of literals) |
| $\omega$ | Clause (disjunction of literals) |
| $\varphi, \phi, \psi, \ldots$ | Logic formula |
| $\mathcal{F} = \{1, \ldots, m\}$ | Set of features |
| $\mathbb{D}_i$ | Domain of feature $i$ |
| $\mathbb{B} = \{0, 1\}$ | Boolean domain |
| $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_m$ | Feature space |
| $x_i$ | Variable associated with feature $i$ |
| $\mathcal{K} = \{c_1, \ldots, c_K\}$ | Set of classes |
| $\mathcal{M}$ | Classifier |
| $\kappa : \mathbb{F} \rightarrow \mathcal{K}$ | Classification function of classifier $\mathcal{M}$ |
| $\mathbf{x} \in \mathbb{F}$ | An arbitrary point in the feature space |
| $\mathbf{v} \in \mathbb{F}$ | A specific point in the feature space |
| $(\mathbf{v}, c)$ | An instance, where $\mathbf{v} \in \mathbb{F}$ and $\kappa(\mathbf{v}) = c$ |
| $\Pr(\cdot)$ | Probability of an event or a point |
| $\mathbf{E}[\cdot]$ | Expected value |
| $\mathcal{T} = (V, E)$ | Tree, with nodes $V$ and edges $E$ |
| $\mathcal{G} = (V, E)$ | Graph, with nodes $V$ and edges $E$ |
| $S = \{s_1, \ldots, s_m\}$ | Feature selectors |
| $\mathcal{X}$ | Weak AXp candidate |
| $\mathcal{Y}$ | Weak CXp candidate |

## Abbreviations

| | |
|---|---|
| iff | if and only if |
| DAG | Directed Acyclic Graph |
| NNF | Negation Normal Form |
| DNNF | Decomposable Negation Normal Form |
| d-DNNF | Deterministic Decomposable Negation Normal Form |
| sd-DNNF | Smooth Deterministic Decomposable Negation Normal Form |
| CNF | Conjunctive Normal Form |
| DNF | Disjunctive Normal Form |
| BDD | Binary Decision Diagram |
| BP | Branching Program |
| FBDD | Free Binary Decision Diagram |
| ROBP | Read-Once Branching Program |
| OBDD | Ordered Binary Decision Diagram |
| SDD | Sentential Decision Diagram |
| PI | Prime Implicants |
| OMDD | Ordered Multi-valued Decision Diagram |
| **CO** | Polytime consistency check |
| **VA** | Polytime validity check |
| **CT** | Polytime model counting |
| **CD** | Polytime conditioning |
| **¬C** | Polytime negation |
| DT | Decision Tree |
| DG | Decision Graph |
| RF | Random Forest |

Chapter 1

# Introduction

## 1.1 Successes of Machine Learning

Over the last decade, there has been a broad consensus that the breakthroughs witnessed in the fields of Artificial Intelligence (AI), particularly in Machine Learning (ML), have been truly monumental [41, 228]. These remarkable achievements, driven by advancements in algorithms, increased computing power, and the availability of vast datasets, have transformed various domains.

In the realm of language technology, a standout achievement has emerged with the development of Large Language Models (LLMs) [66, 210, 58, 107], exemplified by groundbreaking models like ChatGPT. These models represent a big leap in natural language understanding and generation capabilities, showing their versatility across various applications, including chatbots, and content generation. Similarly, in the area of image generation, the emergence of applications such as StableDiffussion and Midjourney [87, 302, 307, 378] have made a profound impact, notably raising the standards of image generation in domains such as computer graphics and artistic design. Meanwhile, the field of reinforcement learning has reached remarkable milestones, exemplified by the exceptional performance of AI agents like AlphaGo [268, 326, 327], which defeated world champions in complex games like Go and chess, showcasing AI's potential in strategic decision-making.

The societal importance of the advances in ML is also demonstrated by the numerous efforts by major companies, specifically dedicated to the general areas of AI/ML. Concrete examples include Amazon, Facebook, Google, Microsoft, among many others. Moreover, the rise of AI for Science (AI4Science) [4, 379], situated at the intersection of AI technologies and scientific disciplines, has emerged as a powerful approach to expedite scientific research and discovery. It harnesses AI's computational capabilities to analyze intricate scientific data, propose hypotheses, and facilitate breakthroughs in fields such as genomics [121, 354], climate science [206, 102], and materials science [61].

These achievements have not only heightened awareness of the transformative potential of ML but have also underscored the critical importance of dependable and trustworthy AI [355, 230, 212, 315, 253, 258]. Although these technologies offer boundless potential, ethical considerations, transparency, and responsible deployment are paramount to guarantee their positive impact on society.

## 1.2 Why eXplainable AI (XAI)?

The remarkable advancements in ML have further accelerated the integration of complex ML models into our daily lives, influencing decisions in diverse domains such as medical treatments, healthcare [283, 251, 295, 296, 310], law [289, 220], finance [308, 143, 25], and transportation [1, 278, 156]. Complex ML models are often viewed as "black boxes" [244, 362, 374, 65] because their internal workings are not readily understandable by humans. As complex ML models continue to gain widespread adoption, an increasing demand for transparency and accountability [294, 345, 227, 152] in its decision-making processes has arisen. Users, regulators, and stakeholders need to understand why an AI system made a particular recommendation or decision. The absence of such transparency and accountability can give rise to several critical issues:

- Bias and Fairness: AI systems may make unfair or biased decisions [16, 164]. Understanding how and why these biases occur is essential for addressing them and ensuring fairness in AI systems.
- Robustness and Safety: In safety-critical applications such as self-driving cars, understanding how AI systems navigate unforeseen scenarios is paramount. The lack of transparency poses significant challenges in ensuring the robustness and safety of AI systems, particularly in situations where AI model operation may be a potential factor contributing to events with catastrophic consequences [103, 323].
- Regulatory Compliance: In various industries, including finance and healthcare, regulations mandate that decisions be explainable and auditable. Non-compliance can result in legal and ethical consequences [204, 312, 76].

In response to the current state of affairs and driven by recent regulations and recommendations [114, 160, 280], along with existing proposals for AI/ML system regulation [112, 337, 344, 291], there is an urgent need to build trust in the operation of AI/ML systems. This demand has spurred rapid growth in the research domain of eXplainable AI (XAI). XAI can be defined as the process of bridging the gap between the inner workings of AI/ML systems and human understanding, all with the goal of establishing trustworthy AI [253].

**The significance of XAI.** Recent guidelines and regulations from influential entities (e.g. UNESCO, the OECD, the European Union (EU)) underscore the importance of both trustworthy AI and XAI [114, 90, 159, 160, 113, 112, 277, 32, 31, 280, 337, 253]. Here are some key reasons for the significance of XAI:

- Enhancing Trust: XAI provides insights into how AI models arrive at their decisions, allowing users to trust the system's recommendations. This is crucial in healthcare, where doctors need to trust AI-assisted diagnoses, and in finance, where investors rely on AI-driven investment advice.

- Mitigating Bias: By explaining why an AI system made a particular decision, XAI helps identify and rectify biases.
- Improving Safety: In autonomous vehicles, XAI can assist human operators in comprehending the rationale behind a driving system's actions. This understanding is essential for safety-critical scenarios, such as avoiding accidents.
- Aiding Compliance: XAI facilitates regulatory compliance by providing transparent documentation of AI decision-making processes. This is vital in sectors subject to strict regulatory oversight, such as pharmaceuticals and finance.

**Methods of XAI.** Various methods are employed in XAI to enhance the trustworthiness and transparency of ML systems. XAI methods can be classified according to various criteria.

- Intrinsic vs. post hoc: Intrinsic methods focuses on designing ML models in a way that they are inherently interpretable from the beginning. Models with simple structures are often considered intrinsically interpretable. Examples include decision trees, decision rules, and linear regression models [363, 304]. In contrast, post hoc methods [299, 247, 300] involves applying interpretation techniques to a ML model after it has been trained. These methods can be applied to complex and inherently non-interpretable models, such as ensemble methods, neural networks.
- Model-agnostic vs. model-specific: Model-agnostic methods [299, 247, 300] can be used with any ML model. These methods are typically applied after the model has completed its training phase, making them post hoc in nature. Model-agnostic methods often operate by analyzing feature input and output relationships, without relying on access to the model's internal details such as weights or structural information. It is undeniable that model-agnostic methods represent the mainstream in the field of XAI. In contrast, model-specific methods are tailored to particular model classes and are not broadly applicable. Model-specific methods can provide deeper insights into certain model types, but they are limited to those specific models. There is a growing trend in utilizing formal methods in the verification of ML systems [315], where logic-based explainability plays a key role [178, 257, 95, 253, 258].
- Global vs. local: Global methods [18, 225] focus on understanding the overall behaviour and decision-making process of the ML model across the entire feature space. In contrast, local methods [299, 332, 333, 101, 247, 300, 253, 258] offer insights into individual predictions, providing trust at the level of single data points.

Most of the XAI methods examined in this thesis can be categorized as post-hoc and local methods. For a more comprehensive understanding of these methods, interested readers are referred to the reference [15, 269, 294, 95, 358, 161, 178, 253, 258].

## 1.3 Why Formal Explainability?

**Model-agnostic explanations are unsound.** The XAI approaches that have gained the most attention are commonly known as model-agnostic methods [148, 299, 247, 300]. However, it is unfortunate that many of the widely adopted XAI methods introduced in recent years suffer from several issues. These include the generation of unsound explanations [189, 276, 178, 257] and challenges related to out-of-distribution sampling [330, 224, 369, 368].

The limitations of these non-formal XAI approaches present a significant challenge to the reliability of model-agnostic explanations, especially in settings categorized as high-risk or safety-critical [305, 304, 303, 345, 289, 88, 162, 253]. Depending on such unsound explanations could potentially result in catastrophic consequences in such scenarios. Besides unsoundness, other limitations of model-agnostic explanations have been reported [62, 108, 218, 201, 173, 174, 175].

**Formal explanations.** As an alternative, there is a growing trend in the application of automated reasoning techniques for explaining and verifying ML models. This approach is generally referred to as formal XAI, characterized by logic-based and model-specific XAI methods [178, 257, 95, 253]. Explanations generated using these methods are termed *formal explanations* or *logic-based explanations*. There are two main approaches for computing formal explanations. The first one is based on exploiting knowledge compilation techniques [320, 321, 72, 319, 96, 97, 94], whereas the second one builds on exploiting abductive reasoning [189, 187, 178, 255, 182, 257, 183, 168, 192, 169, 170, 167, 198, 256, 373, 322, 276]. It should be noted that a number of other works also propose formal approaches for computing explanations [12, 237, 292, 293, 357].

**Compilation-based computation of formal explanations.** This approach compiles the decision function of a given ML model into a tractable circuit [320]. The obtained circuit is tractable and replicates the input-output behavior of the ML models. The tractability, i.e. efficient representation and manipulation, of the resulting circuits solely depends on the queries and transformations [99, 341] they can supports. Once the target tractable circuit is obtained, this method can be highly effective. By examining the resulting circuit, formal explanations for any instance become easily accessible [320]. However, in practice, the process of compilation is worst-case exponential in both time and space [72, 178].

**Abduction-based computation of formal explanations.** This approach does not rely on the explicit transformation of the classification function associated with the ML model into tractable circuits. Therefore, it is not limited by the size of the representation. However, a defect is that it requires the computation of an explanation for each instance [178]. In practice, and in many cases, the utilization of abductive reasoning entails the development of a logic-based representation for the given ML model [178]. (Recent work introduced a method that utilizes various

robustness tools for computing formal explanations [172], which does not explicitly require the development of a logic representation of the ML models.) The efficiency of such approaches depends not only on the optimization of the encoding but also on the performance of automated reasoning tools, such as SMT (Satisfiability Modulo Theories), SAT (Boolean Satisfiability), and QBF (Quantified Boolean Formula) reasoners [45].

**Intrinsically interpretable models need to be explained.** Intrinsically interpretable models [269, 304, 37, 235, 67, 314], as the name suggests, are inherently capable of providing explanations without the need for additional computation. Examples of such interpretable models include decision trees [288], decision lists [301], and decision sets [263, 264, 223]. Recent research has shown that formal explanations for decision tree classifiers can be considerably more concise than explanations provided by interpretable classifiers [193, 194], with explanations in the case of decision trees corresponding to the tree paths. Additionally, the interpretability of decision lists and sets is also problematic [258]. These findings suggests that even interpretable ML models need to be explained.

**Goal of the thesis.** Considering the ongoing rapid growth of the XAI field, this thesis aims to present recent advancements in formal explainability, shedding light on this evolving domain, and providing insights into potential directions for future research.

## 1.4 Structure of the Thesis

As an emerging field, formal explainability requires knowledge of propositional logic and machine learning. We review these fundamental concepts and definitions in Chapter 2.

The next four chapters introduce the recent advance in formal explainability. Chapters 3 and 4 focus on the computation and enumeration of formal explanations for tractable decision graphs and tractable boolean circuits. In these two chapters, we present conditions enabling the computation of formal explanations in polynomial time for these classifiers. Furthermore, we propose a practically efficient solution for enumerating formal explanations. These two chapters are based on the following publications:

- Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, João Marques-Silva. **On Efficiently Explaining Graph-Based Classifiers.** In Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning Special Session on KR and Machine Learning, KR-2021.
- Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, Martin C. Cooper, Nicholas Asher and João Marques-Silva. **Tractable Explanations for d-DNNF Classifiers.** In Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI-2022.

Chapter 5 presents an application of formal explainability, that is, constructing explained decision sets from decision trees. This chapter is based on the following published paper:

- Xuanxiang Huang, João Marques-Silva. **From Decision Trees to Explained Decision Sets.** In Proceedings of 26th European Conference on Artificial Intelligence, ECAI 2023.

Chapter 6 is concerned with solving explainability queries. In this chapter, two specific explainability queries: *feature necessity* and *feature relevancy* are studied. Besides, we prove the computational complexity of these problems with respect to a wide range of classifiers. Additionally, we propose algorithms for their solution in practice. The work in this chapter is based on the following publications:

- Xuanxiang Huang, Yacine Izza, and João Marques-Silva. **Solving Explainability Queries with Quantification: The Case of Feature Relevancy.** In Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI-2023.
- Xuanxiang Huang, Martin C. Cooper, Antonio Morgado, Jordi Planes and João Marques-Silva. **Feature Necessity & Relevancy in ML Classifier Explanations.** In Proceedings of Tools and Algorithms for the Construction and Analysis of Systems: 29th International Conference, TACAS 2023.

Chapter 7 conducts a comparative study between SHAP scores and formal explanations, highlighting potential concerns with Shapley values in the context of explainability. In this chapter, we illustrate, both theoretically and empirically, that utilizing SHAP scores for explainability will yield misleading information about the relative importance of features for predictions. The work in this chapter is based on the following papers:

- Xuanxiang Huang and João Marques-Silva. **The Inadequacy of Shapley Values for Explainability.** arXiv 2023.
- Xuanxiang Huang and João Marques-Silva. **A Refutation of Shapley Values for Explainability.** arXiv 2023.
- Xuanxiang Huang and João Marques-Silva. **Refutation of Shapley Values for XAI – Additional Evidence.** arXiv 2023.
- Xuanxiang Huang and João Marques-Silva. **On the Failings of Shapley Values for Explainability.** International Journal of Approximate Reasoning 2024.

Chapter 8 summarizes the thesis and provides an outlook on future research.

Chapter 2

# Background

This chapter provides an overview of the fundamental knowledge underpinning the work presented in this thesis. We describe most concepts informally and refer to the literature for formal definitions.

This chapter comprises three sections. In Section 2.1, we present the basic concepts and definitions of propositional logic. We also introduce fundamental notions of boolean functions and their graph representation, which are commonly referred to as boolean circuits. Additionally, we present various queries and transformations defined on boolean circuits. Next, our focus shifts to inconsistent formulas, where we present two essential concepts. Following that, we provide a brief introduction to quantified boolean formulas, which represent a generalization of propositional logic, along with an overview of complexity classes.

In Section 2.2, we present machine learning models aiming for classification problems. More specifically, we will focus on tractable graph models, such as decision trees, decision graphs, and decision diagrams. Additionally, we will introduce rule-based systems like decision lists and decision sets. Furthermore, we will discuss random forest classifiers, and monotonic classifiers.

In Section 2.3, we present a brief history of eXplainable AI, focusing on well-known model-agnostic methods. However, model-agnostic methods are susceptible to a range of critical issues, including the unsoundness of explanations. As a result, this section will shed more light on an emerging area of research: formal eXplainable AI, which aims to address these limitations and focus on ensuring the rigor of explanations.

## 2.1 Logic Foundations

This section reviews standard notions in propositional logic, boolean circuits, and other related topics. For a more comprehensive introduction, we refer the readers to the relevant literature [60, 39, 351, 85, 40, 44, 284, 91, 99, 165, 232, 231].

### 2.1.1 Propositional Formulas

**Definition 1** (Proposition Formulas)**.** *Propositional formulas* $\varphi, \phi, \ldots$ are built from *atoms* $x_1, x_2, \ldots$, the *unary connective* $\neg$, the *binary connectives* $\wedge$, $\vee$, and *parentheses* (,). The logic operators $\neg$, $\wedge$ and $\vee$ are read "NOT", "AND" and "OR", respectively. Formulas are defined recursively as follows:

1. Any atom $x_i$ is a formula.

2. If $x_i$ is a formula, then $\neg x_i$ is also a formula.
3. If $x_i$, $x_j$ are formulas, then $(x_i \wedge x_j)$ is also a formula.
4. If $x_i$, $x_j$ are formulas, then $(x_i \vee x_j)$ is also a formula.

Other often used logic operators include $\rightarrow$ (implication) and $\leftrightarrow$ (equivalence), where $(\varphi \rightarrow \phi)$ stands for $(\neg\varphi \vee \phi)$, and $(\varphi \leftrightarrow \phi)$ stands for $((\varphi \rightarrow \phi) \wedge (\phi \rightarrow \varphi))$. Besides, a subformula of a formula $\varphi$ is any substring of $\varphi$ which is a formula.

**Definition 2** (Truth Assignment)**.** A *truth assignment* is a map $\mu : \{atoms\} \rightarrow \{\top, \bot\}$, where $\{\top, \bot\}$ represents $\{\mathbf{true}, \mathbf{false}\}$. ($\{1, 0\}$ will also be employed to denote $\{\mathbf{true}, \mathbf{false}\}$). A truth assignment $\mu$ can be extended to assign either $\top$ or $\bot$ to any formula, as follows:

1. $(\neg\varphi)^\mu = \top$ iff $\varphi^\mu = \bot$.
2. $(\varphi \wedge \phi)^\mu = \top$ iff $\varphi^\mu = \top$ and $\phi^\mu = \top$.
3. $(\varphi \vee \phi)^\mu = \top$ iff $\varphi^\mu = \top$ or $\phi^\mu = \top$.

**Definition 3** (Satisfiable, Unsatisfiable)**.** A *truth assignment* $\mu$ *satisfies* $\varphi$ iff $\varphi^\mu = \top$. $\mu$ *satisfies* a set of $\Delta$ of formulas iff $\mu$ *satisfies* $\varphi$ for all $\varphi \in \Delta$. The set $\Delta$ is *satisfiable* iff some *truth assignment* $\mu$ *satisfies* $\Delta$; otherwise $\Delta$ is *unsatisfiable*.

**Definition 4** (Entailment)**.** $\Delta \models \varphi$ (i.e. $\varphi$ is a logical consequence of $\Delta$) iff for every *truth assignment* $\mu$, if $\mu$ *satisfies* $\Delta$, then $\mu$ *satisfies* $\varphi$.

Two formulas $\varphi$ and $\phi$ are considered equivalent (denoted $\varphi \equiv \phi$) if they entail each other, that is, $\varphi \models \phi$ and $\phi \models \varphi$.

**Definition 5** (Valid Formula)**.** A formula $\varphi$ is *valid* iff $\models \varphi$ (i.e. $\varphi^\mu = \top$ for all $\mu$). A *valid* propositional formula is called a *tautology*.

**Definition 6** (Literal)**.** A literal $l$ is either the positive occurrence of an atom $x_i$, i.e. $l = x_i$ or the negative occurrence of an atom $x_i$, i.e. $l = \neg x_i$.

**Definition 7** (Clause)**.** A clause $\omega$ is a disjunction of a set of literals.

**Definition 8** (Term)**.** A term $\tau$ is a conjunction of a set of literals.

**Definition 9** (CNF)**.** Given a set of clauses $\{\omega_1, \ldots, \omega_m\}$, $\omega_1 \wedge \cdots \wedge \omega_m$ is a *conjunctive normal form* (CNF) formula.

**Definition 10** (DNF)**.** Given a set of terms $\{\tau_1, \ldots, \tau_m\}$, $\tau_1 \vee \cdots \vee \tau_m$ is a *disjunctive normal form* (DNF) formula.

**Definition 11** (Prime Implicants)**.** Given a propositional formula $\varphi$ and a term $\tau$. $\tau$ is an *implicant* of $\varphi$ if $\tau \models \varphi$. $\tau$ is a *prime implicant* of $\varphi$ if 1) $\tau$ is an *implicant* of $\varphi$, and 2) for any term $\tau'$ such that $\tau \models \tau'$ but $\tau \not\equiv \tau'$, then $\tau' \not\models \varphi$.

**Definition 12** (Prime Implicates)**.** Given a propositional formula $\varphi$ and a clause $\omega$. $\omega$ is an *implicate* of $\varphi$ if $\varphi \models \omega$. $\omega$ is a *prime implicate* of $\varphi$ if 1) $\omega$ is an *implicate* of $\varphi$, and 2) for any $\omega'$ such that $\omega' \models \omega$ but $\omega' \not\equiv \omega$, then $\varphi \not\models \omega'$.

### 2.1.2 Boolean Functions & Boolean Circuits

**Definition 13** (Boolean Functions [85])**.** Let $n$ be a positive integer and let $\mathbb{B} = \{0, 1\}$. A boolean function with $n$ variables is a function from $\mathbb{B}^n$ to $\mathbb{B}$, where $\mathbb{B}^n$ denotes the $n$-fold cartesian product of $\mathbb{B}$. A point $\mathbf{x} = (x_1, x_2 \ldots, x_n) \in \mathbb{B}^n$ is a *true point* (respectively, *false point*) of the boolean function $f$ if $f(\mathbf{x}) = 1$ (respectively, $f(\mathbf{x}) = 0$).

Building on earlier work [91, 99, 165, 59], we introduce boolean circuits represented as negation normal form (NNF) and some tractable boolean circuits that are strictly less succinct than NNF.

**Definition 14** (NNF)**.** The language *negation normal form* (NNF) is the set of all directed acyclic graphs (DAGs), where each leaf node is labeled with either $\top$, $\bot$, $x_i$ or $\neg x_i$, for $x_i \in X$, and has no child nodes. Each internal node is labeled with either $\wedge$ or $\vee$ and has at least two child nodes.

**Definition 15** (DNNF)**.** The language *decomposable* NNF (DNNF) is the set of all NNFs, where for every node labeled with $\wedge$, no atoms are shared between its child nodes.

**Definition 16** (d-DNNF)**.** The language *deterministic* DNNF (d-DNNF) is the set of all DNNFs, where for every node labeled with $\vee$, each pair of its child nodes is inconsistent.

**Definition 17** (sd-DNNF)**.** The language *smooth* d-DNNF (sd-DNNF) is the set of all d-DNNFs, where for every node labeled with $\vee$, all its child nodes are defined on the same set of atoms.

**Definition 18** (Shannon expansion, Boole's expansion [317, 53])**.** Let $f : \mathbb{B}^n \to \mathbb{B}$ be a boolean function defined on $X = \{x_1, \ldots, x_n\}$. Let $x_i \in X$, the function $f$ can be decomposed as follows:

$$f = (x_i \wedge f|_{x_i=1}) \vee (\neg x_i \wedge f|_{x_i=0}) \ [1] \tag{2.1}$$

where $f|_{x_i=1}$ ($f|_{x_i=0}$) denotes the *cofactor* [55] of $f$ with respect to $x_i = 1$ ($x_i = 0$).

**Definition 19** (BDD [5, 59, 351, 137], BP)**.** A *Binary decision diagram* (BDD) is a DAG with two types of nodes: terminal nodes and non-terminal nodes. Each non-terminal node is labeled by a variable $x_i \in X$, and has two outgoing edges, one labeled by 0 and the other by 1. Each terminal node is labeled by a 1 or 0, and has no outgoing edges. BDDs are also known as Branching Programs (BPs).

**Definition 20** (FBDD, ROBP [351, 137])**.** A BDD is *read-once* if each variable is tested at most once on any path from the root node to a terminal node. A *read-once* BDD is also referred to as a *free* BDD (FBDD) or ROBP (Read-Once BP).

[1] which is also referred to as the *decision* property [59, 99].

**Definition 21** (OBDD)**.** A BDD is *ordered* (OBDD) [5, 59] if the features are tested in the same order on all paths.

**Definition 22** (SDD)**.** *Sentential decision diagrams* (SDDs) [93, 341, 71, 73] are a subset of the d-DNNF. SDDs are based on a boolean function decomposition, called $(X, Y)$*-partitions*. Let $f$ be a boolean function defined on two disjoint sets $X$ and $Y$, $f$ can be decomposed as:

$$f = [p_1(X) \wedge s_1(Y)] \vee \cdots \vee [p_n(X) \wedge s_n(Y)] \tag{2.2}$$

where sub-functions $p_i(X)$ are *mutually exclusive*, *exhaustive*, and *non-false*. Moreover, $p_i(X)$ are referred to as *primes* and $s_i(Y)$ are referred to as *subs*.

An SDD can be constructed from a propositional formula by recursively applying $(X, Y)$-partitions on primes and subs. Variables and constants are SDDs but cannot be further decomposed. The decomposition of a boolean function $f$ using the $(X, Y)$-partitions is governed by a *vtree*.

**Definition 23** (Vtree [93, 341, 71, 73])**.** A *vtree* is a full binary tree with its leaves labeled with variables. Each internal node of the *vtree* partitions a variable set into those appearing in its left subtree ($X$) and those appearing in its right subtree ($Y$).

### 2.1.3 Queries & Transformations

In this thesis, we focus on some specific queries and transformations supported by different circuits. The queries we consider are: 1) *polytime consistency check* (**CO**), 2) *polytime validity check* (**VA**), and 3) *polytime model counting* (**CT**). The transformations we consider are: 1) *polytime conditioning* (**CD**), and 2) *polytime negation* ($\neg$**C**).

Let **L** denote a subset of NNF, we adopt the standard definitions of these queries and transformations as described in the literature [91, 99, 165, 59].

**Definition 24** (**CO**, **VA**)**.** A propositional language **L** satisfies **CO** (**VA**) iff there exists a polynomial-time algorithm that can decide whether an arbitrary formula $\varphi$ from **L** is consistent (valid).

**Definition 25** (**CT**)**.** A propositional language **L** satisfies **CT** iff there exists a polynomial-time algorithm that can count the number of models for any formula $\varphi$ from **L**.

**Definition 26** (Conditioning)**.** Let $\varphi$ represent a propositional formula and let $\tau$ denote a consistent term ($\tau \not\models \bot$). The *conditioning* [99] of $\varphi$ on $\tau$, denoted as $\varphi|_\tau$, is the formula obtained by replacing each variable $x_i$ by $\top$ (resp. $\bot$) if $x_i$ (resp. $\neg x_i$) is a positive (resp. negative) literal of $\tau$.

**Definition 27** (**CD**)**.** A propositional language **L** satisfies **CD** iff there exists a polynomial-time algorithm that maps every formula $\varphi$ from **L** and every consistent term $\tau$ into a formula in **L** that is logically equivalent to $\varphi|_\tau$.

**Definition 28** (¬**C**)**.** A propositional language **L** satisfies ¬**C** iff there exists a polynomial-time algorithm that maps every formula $\varphi$ from **L** to a formula of **L** that is logically equivalent to $\neg\varphi$.

It is well-known that d-DNNFs, SDDs, FBDDs, OBDDs satisfy the queries **CO**, **VA**, **CT**. Additionally, SDDs, FBDDs, OBDDs satisfy transformations **CD** and ¬**C**. However, it should be noted that d-DNNFs do not satisfy ¬**C**. There are additional queries and transformations, the interested readers are referred to [99].

### 2.1.4 Hitting Sets

Given a collection $\Omega$ of sets from some finite domain $D$.

**Definition 29** (Hitting Sets)**.** A hitting set $H$ of $\Omega$ is $H \subseteq D$ such that $\forall S \in \Omega, H \cap S \neq \emptyset$

A hitting set is considered *minimal* or *irreducible* if removing any element from it results in the loss of the property of being a hitting set [232, 231]. The hypergraph transversal problem [42, 213, 111, 146] is equivalent to the hitting set problem.

### 2.1.5 Inconsistent formulas

For an unsatisfiable CNF formula $\varphi$, let $\mathcal{B}$ denote the set of clauses in $\varphi$.

**Definition 30** (MUS)**.** A subset $\mathcal{U} \subseteq \mathcal{B}$ is an *minimal unsatisfiable subset* (MUS) if $\mathcal{U}$ is unsatisfiable and $\forall\omega_i \in \mathcal{U}, \mathcal{U} \setminus \{\omega_i\}$ is satisfiable.

**Definition 31** (MCS)**.** A subset $\mathcal{C} \subseteq \mathcal{B}$ is an *minimal correction subset* (MCS) if $\mathcal{B} \setminus \mathcal{C}$ is satisfiable and $\forall\omega_i \in \mathcal{C}, \mathcal{B} \setminus \{\mathcal{U} \setminus \{\omega_i\}\}$ is unsatisfiable.

An MUS can be seen as a minimal explanation of the unsatisfiability of the formula $\varphi$ and cannot be made smaller without becoming satisfiable. An MCS can be seen as a minimal effort required to "correct" the unsatisfiability of the formula $\varphi$ and cannot be made smaller without becoming unsatisfiable [232, 231, 261].

Additionally, there is a (subset-)minimal hitting set (MHS) relationship between MUSes and MCSes. The MHS relationship between MUSes and MCSes was first established in the field of model-based diagnosis [297, 253, 258] and subsequently investigated for propositional formulas in clausal form [47, 253, 258].

### 2.1.6 Quantification Problems

A well-known generalization of propositional logic is quantified boolean formulas (QBFs), where two additional logic operators, namely $\forall$ and $\exists$ are used to quantify the possible values of variables. A *prenex* QBF is of the form,

$$Q_1 x_1 Q_2 x_2 \ldots Q_m x_m . \phi$$

where $Q_i \in \{\exists, \forall\}$ and $\phi$ is a propositional formula. Moreover, $\phi$ is referred to as the *matrix* and $Q_1x_1Q_2x_2 \ldots Q_mx_m$ as the *prefix*. The decision problem for QBF is PSPACE-complete [24]. In this thesis, our primary focus centers on quantified problems with two levels of quantifiers, concretely $\exists\forall$, which is a well-known $\Sigma_2^{\mathrm{P}}$-complete decision problem [24].

### 2.1.7 Complexity Classes

The thesis adopts standard notation and definitions when addressing the decision problem for propositional logic, specifically the Boolean Satisfiability (SAT) problem [45], which is well-known to be NP-complete [80]. The thesis addresses several well-known classes of decision problems, including P, NP, and $\Sigma_2^{\mathrm{P}}$. Interested readers are referred to a standard reference on computational complexity [24].

## 2.2 Classification Models in Machine Learning

In this section, we introduce some well-known machine learning classifiers studied in the thesis. For readers who are not familiar with Machine Learning, we recommend referring to [381, 382, 48, 155] for a comprehensive introduction.

### 2.2.1 Classification Problems

A classification problem is defined on a set of features $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ has a domain $\mathbb{D}_i$. $\mathbb{D}_i$ can be categorical or ordinal, with values that can be boolean, integer, or real-valued. $\mathcal{K}$ can also be categorical or ordinal. Feature space is defined as $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_m$. For boolean domains, $\mathbb{D}_i = \mathbb{B}$, $i = 1, \ldots, m$, and $\mathbb{F} = \mathbb{B}^m$. Moreover, the set of variables associated with features is $X = \{x_1, \ldots, x_m\}$. The notation $\mathbf{x} = (x_1, \ldots, x_m)$ denotes an arbitrary point in feature space, where each $x_i$ is a variable taking values from $\mathbb{D}_i$. The notation $\mathbf{v} = (v_1, \ldots, v_m)$ represents a specific point in feature space, where each $v_i$ is a constant representing one concrete value from $\mathbb{D}_i$. With respect to the set of classes $\mathcal{K}$, the size of $\mathcal{K}$ is assumed to be finite; no additional restrictions are imposed on $\mathcal{K}$. An ML classifier $\mathcal{M}$ is characterized by a (non-constant) *classification function* $\kappa$ that maps feature space $\mathbb{F}$ into the set of classes $\mathcal{K}$, i.e. $\kappa : \mathbb{F} \to \mathcal{K}$. An *instance* is a pair $(\mathbf{v}, c)$ representing a point $\mathbf{v} = (v_1, \ldots, v_m)$ in feature space, and the classifier's prediction, i.e. $\kappa(\mathbf{v}) = c$. Abusing notation, we will also use $\mathbf{x}_{a..b}$ to denote $x_a, \ldots, x_b$, and $\mathbf{v}_{a..b}$ to denote $v_a, \ldots, v_b$. In addition, for a subset $\mathcal{S}$ of $\mathcal{F}$, we will use $\mathbf{v}_{\mathcal{S}}$ to denote the partial point of $\mathbf{v}$, which represents the restriction of the complete point $\mathbf{v}$ to those features in $\mathcal{S}$. Finally, a classifier $\mathcal{M}$ is a tuple $(\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$.

### 2.2.2 Decision Trees, Diagrams & Graphs

A decision tree (DT) [288, 194, 197, 192] $\mathcal{T} = (V, E)$ is a DAG having at most one path between every pair of nodes. $\mathcal{T}$ has a root node, characterized by having no

incoming edges. All other nodes have one incoming edge. We consider univariate decision trees where each non-terminal node is associated with a single feature $i \in \mathcal{F}$, and each terminal node is associated with a value from $\mathcal{K}$. Each edge is labeled with a literal, relating a feature (associated with the edge's starting node) with some values (or range of values) from the feature's domain. We will consider literals to be of the form $x_i \in \mathbb{I}_i$, whereas $\mathbb{I}_i \subseteq \mathbb{D}_i$. The type of literals used to label the edges of a DT allows the representation of the DTs generated by a wide range of decision tree learners (e.g. [339]). Paths in the DT are represented as a sequence of numbers, e.g. $\mathcal{P} = \langle r_1, r_2, \ldots, r_n \rangle$, such that each pair $(r_j, r_{j+1})$ denotes an edge of $\mathcal{T}$. The set of paths of $\mathcal{T}$ is denoted by $\mathbb{P}$ (where the dependency on $\mathcal{T}$ is omitted for simplicity). For each path $\mathcal{P} \in \mathbb{P}$ from the root node to a terminal node, a feature can be tested more than once (in other words, it is not *read-once*). Besides, if we conjunct all the literals of a path $\mathcal{P}$, we will obtain a term $\tau$.

Given a path $\mathcal{P}_k$, the features tested in the non-terminal nodes of $\mathcal{P}_k$ are represented by $\Phi(\mathcal{P}_k)$. Also, for a path $\mathcal{P}_k$ of $\mathcal{T}$, and a set of features $\mathcal{X} \subseteq \Phi(\mathcal{P}_k) \subseteq \mathcal{F}$, $\Lambda(\mathcal{P}_k, \mathcal{X})$ denotes the set of literals associated with the features in $\mathcal{X}$ along path $\mathcal{P}_k$. The definition of $\Lambda$ accounts for situations where a feature is tested more than once, but we will not delve into that in this thesis. Concretely, for each feature $i \in \Phi(\mathcal{P}_k)$, we have literals $x_i \in \mathbb{I}_i$, where $\mathbb{I}_i \subseteq \mathbb{D}_i$ is the intersection of the sets in each of the literals of $\mathcal{P}_k$ on feature $i$. Finally, it is assumed that DTs are organized such that the computed classification function is total. (Evidently, DTs can be envisioned for which $\kappa$ is not total [194, page 270], or for which $\kappa$ is not a function, but it is instead a relation [194], e.g. when node domain splits do not form a partition.)

Decision graphs (DGs) [281] can be viewed as a generalization of DTs, in the sense that each non-terminal node can have more than one incoming edges, so a sub-graph can be shared by different nodes. Moreover, each non-terminal node can have more than two outgoing edges. Furthermore, the DTs as well as the DGs considered in this thesis are assumed to satisfy the following three restrictions:

**Assumption 1.** The literals associated with the outgoing edges of each non-terminal represents a partition of $\mathbb{I}_i$. Every path from the root node to a terminal node is not *inconsistent*. Each path consistent with some points in the feature space.

The first restriction means that for any $\mathbf{v} \in \mathbb{F}$ and an arbitrary non-terminal node labeled $x_i$, the value $v_i$ will activate exactly one of its outgoing edges. The second restriction means that $\tau \not\models \bot$ ($\tau$ is the conjunction all the literals of a path $R$). With these restrictions, any $\mathbf{v} \in \mathbb{F}$ is consistent with *exactly* one path in $\mathcal{T}$. By *consistent* we mean that the literals associated with the path are satisfied (or consistent) with the feature values in $\mathbf{v}$. A more in-depth analysis of explaining decision trees is available in [195, 194]. Moreover, FBDD lies the intersection of propositional languages and decision graphs. To show that FBDDs are subset of DGs: 1) Every FBDD contains only binary features, and each non-terminal node has two outgoing edges, one labeled by 0 and the other labeled by 1. So the literals associated with the outgoing edges of each non-terminal node represent a partition

of $\{0, 1\}$. 2) Every FBDD is read-once, so it is impossible to have an inconsistent path in FBDD. According to the definition of DG, any FBDD is a DG.

### 2.2.3 Decision Lists & Sets

Decision lists (DLs) [301, 182] and sets (DSs) [263, 264, 223, 190] represent families of classifiers based on rules. A rule is of the form: IF cond THEN class, i.e. if the condition cond is true given the values assigned to features, then class is predicted. When cond is true, we say that the rule *fires* [258]. A decision list is a set of ordered rules, of the form:

$$\begin{array}{llllll} \mathcal{R}_1: & \text{IF} & (\tau_1) & \text{THEN} & c_1 & \\ \mathcal{R}_2: & \text{ELSE IF} & (\tau_2) & \text{THEN} & c_2 & \\ \cdots & & & & & \\ \mathcal{R}_r: & \text{ELSE IF} & (\tau_r) & \text{THEN} & c_r & \\ [\mathcal{R}_{\text{DEF}}: & \text{ELSE} & & & c_{r+1}] & \end{array} \tag{2.3}$$

If it is known that the conditions cover the feature space, then the default rule is unnecessary, as it will never fire. For some of the examples, we will use a more compact notation for rules, of the form:

$$\tau \rightarrowtail c \tag{2.4}$$

This more compact representation has the same interpretation as before, i.e. if the condition $\tau$ is true, then the prediction is $c$. Boolean literals in the conditions of rules will be represented by variables, e.g. $x$, or their negations, e.g. $\neg x$ or $\overline{x}$. (For the more complex examples, we will opt for the more compact notation, i.e. $\overline{x}$.)

In contrast to DLs, a DS is a set of unordered rules. Decision sets exhibit a number of issues, which complicate their analysis. One issue is *overlap*, i.e. two or more rules predicting different classes firing on the same inputs [253, 258]. Another issue is *coverage* of feature space, i.e. no rule firing one some input [253, 258]. A solution to the problem of coverage is to add a default rule, which fires only when none of the other rules do. This condition also complicates reasoning about DSs.

### 2.2.4 Random Forest Classifiers

There are a variety of classifiers that aggregate decision trees. Well-known examples of tree ensembles [381, 180] include random forests [56], but also boosted trees [124]. Different types of tree ensembles are induced with different learning algorithms. Random Forests (RFs) with majority voting [56, 364, 136] are very popular and widely used tree ensemble ML models. Conceptually, an RF is collection of decision trees (DTs), where each tree $\mathcal{T}_i$ of the ensemble $\mathcal{M}$ is trained on a randomly selected subset of the training data so as the trees of the RF are not correlated. (In contrast to a single DT, RFs are less prone to over-fitting and so offer in general better accuracy on test data.) The predictions of a RF classifier are made by majority vote

of trees, that is each tree predicts for a class and the class with largest score is picked. (Note that other versions of RFs using probabilities or weights are implemented by different learning tools, e.g., scikit-learn [286], XGBoost [69], etc. However, and similarly to related work [198], this thesis considers the original proposal for RFs [56].)

### 2.2.5 Monotonic Classifiers

Monotonic classifiers find a number of important applications, and have been studied extensively in recent years [117, 365, 236, 328, 256]. Let $\preccurlyeq$ denote a *total order* on the set of classes $\mathcal{K}$. Concretely, we assume $c_1 \preccurlyeq c_2 \preccurlyeq \ldots c_K$. Furthermore, we assume that each domain $\mathbb{D}_i$ is ordered such that the value taken by feature $i$ is between a lower bound $\lambda(i)$ and an upper bound $\mu(i)$. Given $\mathbf{v}_1 = (v_{11}, \ldots, v_{1i}, \ldots, v_{1m})$ and $\mathbf{v}_2 = (v_{21}, \ldots, v_{2i}, \ldots, v_{2m})$, we say that $\mathbf{v}_1 \leq \mathbf{v}_2$ if, $\forall (i \in \mathcal{F}).(v_{1i} \leq v_{2i})$. Finally, a classifier is monotonic if whenever $\mathbf{v}_1 \leq \mathbf{v}_2$, then $\kappa(\mathbf{v}_1) \preccurlyeq \kappa(\mathbf{v}_2)$.

## 2.3 eXplainable Artificial Intelligence

Motivated by the widespread adoption of machine learning (ML) in a ever-increasing range of domains, eXplainable AI (XAI) is becoming critical, both to build trust, but also to validate ML models [148, 309, 269]. Techniques used in XAI to interpret the behavior of ML models can be classified according to various criteria. including whether they are *intrinsic* [363, 304] or *post-hoc* [299, 247, 300, 257], *model-agnostic* [299, 247, 300] or *model-specific* [315, 178, 257], and *global* [18, 225] or *local* [299, 300, 178, 257]. However, the thesis will exclusively concentrate on local methods.

**How does the ML model make predictions?** Global methods [18, 225] focus on understanding the overall behavior and decision-making process of the ML model across the entire feature space. These methods provide insights into the model's general tendencies, biases, and feature importance.

**Why does the ML model make such a prediction about this data point?** On the other hand, local methods offer insights into individual predictions, providing trust at the level of single data points. The local explanation problem can be defined as follows:

**Definition 32** (Local Explanation Problem)**.** A *local explanation problem* $\mathcal{E}$ is a tuple $(\mathcal{M}, (\mathbf{v}, c))$, where $\mathcal{M} = (\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$ is a classifier such that $\kappa(\mathbf{v}) = c$.

### 2.3.1 Model-Agnostic Methods for Local Explainability

Popular local methods of explainability can be broadly organized into two families: those based on *feature attribution* and those based on *feature selection*. Feature

selection methods identify *sets of features* (i.e. an explanation) relevant for a prediction, while feature attribution methods assign an *importance* to each feature. Most well-known solutions are model-agnostic, meaning that they can be applied to any black-box ML model without requiring access to the model's internal structure or parameters.

**Local Interpretable Model-agnostic Explanations.** Local Interpretable Model-agnostic Explanations (LIME) [299, 82, 269] is a popular model-agnostic feature attribution method used in the field of XAI. The key idea behind LIME is to approximate the behavior of a black-box ML model around a specific instance by creating a simpler model called a "surrogate model" or "local model." LIME generates local explanations by perturbing the input instance of interest and observing how the model's predictions change. It provides explanations in the form of feature importance scores, enabling users to understand how the model arrived at its decision for a particular instance.

In this approach, an explanation is defined as a model $g \in \varkappa$, where $\varkappa$ is a class of potentially interpretable models (e.g. decision trees). Since not every $g \in \varkappa$ may be sufficiently simple to be interpretable, we let $\Omega(g)$ denote a measure of the explanation's complexity (as opposed to interpretability) for $g \in \varkappa$. Additionally, we employ $\pi_{\mathbf{v}}(\mathbf{x})$ as a proximity measure between an instance $\mathbf{x}$ to $\mathbf{v}$, so as to define locality around $\mathbf{v}$. Finally, let $\mathcal{L}(\kappa, g, \pi_{\mathbf{v}})$ be a measure of how unfaithful $g$ is in approximating $\kappa$ in the locality defined by $\pi_{\mathbf{v}}$. The explanation produced by LIME is obtained by the following:

**Definition 33** (LIME [299, 82])**.** Given a classifier $\mathcal{M}$ over a set of features $\mathcal{F}$, which is associated with a classification function $\kappa$, and a data point $\mathbf{v} \in \mathbb{F}$, LIME is defined as

$$\xi(\mathbf{v}) = \text{argmin}_{g \in \varkappa} \mathcal{L}(\kappa, g, \pi_{\mathbf{v}}) + \Omega(g) \tag{2.5}$$

Here, we aim to minimize $\mathcal{L}(\kappa, g, \pi_{\mathbf{v}})$ while ensuring that $\Omega(g)$ remains suitably low, in order to attain explanations that encompass both interpretability and local fidelity.

**Shapley Values & SHapley Additive exPlanations.** Shapley values were first introduced by L. Shapley [318, 10] in the context of game theory. Given a cooperative game, Shapley values represent a way to distribute the worth of the game by each player. Shapley values have been extensively used for explaining the predictions of ML models, e.g. [332, 333, 101, 247, 245, 68, 262, 329, 349], among a vast number of recent examples. Given an instance, these methods treat each feature (and its associated value) as an individual player, and a numeric value is assigned to each feature that quantifies its *contribution* with respect to the prediction.

SHapley Additive exPlanations (SHAP) [247, 245, 269] is one such method that computes so-called *SHAP scores*, which instantiate Shapley values in the context of explainability. It is arguably among the most popular model-agnostic feature-attribution methods. SHAP calculates how each feature's inclusion or exclusion

from a model affects the predictions. It takes into account all possible feature combinations and computes the average contribution of each feature over all possible permutations. It is well-known that the exact computation of SHAP scores is computationally hard [21, 20, 22, 105, 106]. However, for restricted families of classifiers, the computation of SHAP scores is polynomial [21, 20, 22, 105, 106].

To produce SHAP scores, we need a probability distribution over the features. We denote the probability of a data point as $\Pr(\cdot)$. Let $\Upsilon : 2^{\mathcal{F}} \to 2^{\mathbb{F}}$ be defined by

$$\Upsilon(\mathcal{S}; \mathbf{v}) := \{\mathbf{x} \in \mathbb{F} \mid \wedge_{i \in \mathcal{S}} x_i = v_i\} \tag{2.6}$$

$\Upsilon(\mathcal{S}; \mathbf{v})$ denotes all the points in feature space that have in common with $\mathbf{v}$ the values of the features specified by $\mathcal{S}$. The *expected value* of a classification function $\kappa$ is denoted as $\mathbf{E}[\kappa]$. For a complete data point $\mathbf{v}$, we have $\mathbf{E}[\kappa|_{\mathbf{v}}] = \kappa(\mathbf{v})$. Furthermore, let $\mathbf{E}[\kappa|_{\mathbf{v}_\mathcal{S}}]$ denote the expected value of the boolean function $\kappa|_{\mathbf{v}_\mathcal{S}}$, which is defined as follow:

$$\mathbf{E}[\kappa|_{\mathbf{v}_\mathcal{S}}] := \sum_{\mathbf{x} \in \Upsilon(\mathcal{S}; \mathbf{v})} \kappa(\mathbf{x}) \cdot \Pr(\mathbf{x}|\mathbf{v}_\mathcal{S}) \tag{2.7}$$

Let $\phi : 2^{\mathcal{F}} \to \mathbb{R}$ be defined by

$$\phi(\mathcal{S}; \mathcal{M}, \mathbf{v}) := \mathbf{E}[\kappa|_{\mathbf{v}_\mathcal{S}}] \tag{2.8}$$

In the case of uniform distribution, we have:

$$\phi(\mathcal{S}; \mathcal{M}, \mathbf{v}) = \frac{1}{2^{|\mathcal{F} \setminus \mathcal{S}|}} \sum_{\mathbf{x} \in \Upsilon(\mathcal{S}; \mathbf{v})} \kappa(\mathbf{x}) \tag{2.9}$$

To simplify the notation, the following definitions are used,

$$\Delta(i, \mathcal{S}; \mathcal{M}, \mathbf{v}) := (\phi(\mathcal{S} \cup \{i\}; \mathcal{M}, \mathbf{v}) - \phi(\mathcal{S}; \mathcal{M}, \mathbf{v})) \tag{2.10}$$

$$\varsigma(\mathcal{S}; \mathcal{M}, \mathbf{v}) := \frac{|\mathcal{S}|!(|\mathcal{F}| - |\mathcal{S}| - 1)!}{|\mathcal{F}|!} \tag{2.11}$$

**Definition 34** (SHAP Score [247, 245, 106, 21, 22])**.** Given a classifier $\mathcal{M}$ over a set of features $\mathcal{F}$, a probability distribution $\Pr$, a data point $\mathbf{v} \in \mathbb{F}$, and a feature $i \in \mathcal{F}$, the SHAP score of feature $i$ on $\mathbf{v}$ with respect to $\mathcal{M}$, denoted as $\mathsf{SHAP} : \mathcal{F} \to \mathbb{R}$, is defined as

$$\mathsf{SHAP}(i; \mathcal{M}, \mathbf{v}) := \sum\nolimits_{\mathcal{S} \subseteq (\mathcal{F} \setminus \{i\})} \varsigma(\mathcal{S}; \mathcal{M}, \mathbf{v}) \times \Delta(i, \mathcal{S}; \mathcal{M}, \mathbf{v}) \tag{2.12}$$

It is important to note the sum of the SHAP scores of all features is related to the prediction of the given instance and the expected value of the classification function $\kappa$ [332, 333, 105, 106]:

$$\sum_{i \in \mathcal{F}} \mathsf{SHAP}(i; \mathcal{M}, \mathbf{v}) + \phi(\emptyset; \mathcal{M}, \mathbf{v}) = \kappa(\mathbf{v}) \tag{2.13}$$

**Anchors Explanations.** Anchor Explanations (Anchor) [300, 82, 269] is a popular model-agnostic feature-selection method that provides explanations for the predictions of complex ML models. The main idea behind Anchor is to identify small and easily understandable "anchor rules" that sufficiently explain a model's predictions for specific instances. These anchor rules are simple **IF-THEN** statements that capture the key features or conditions under which the model makes a particular prediction. Anchor explanations are concise and human-readable, providing users with insights into how the model makes its decisions for specific instances.

**Definition 35** (Anchors [300, 82])**.** Given a classification function $\kappa$, and a data point $\mathbf{v}$ being explained, $A$ is an *anchor* if

$$\mathbf{E}_{\mathcal{D}(\mathbf{x}|A)}[1_{\kappa(\mathbf{x})=\kappa(\mathbf{v})}] \geq \delta, A(\mathbf{v}) = 1 \tag{2.14}$$

wherein $\mathcal{D}(\cdot|A)$ denote the conditional distribution when the rule $A$ applies. $0 \leq \delta \leq 1$ specifies a precision threshold, only rules that achieve a local fidelity of at least $\delta$ are considered a valid result. And $1_{\kappa(\mathbf{x})=\kappa(\mathbf{v})}$ is the indicator function.

$A$ is a rule (set of predicates) acting on such an interpretable representation, such that $A(\mathbf{v})$ returns 1 if all its feature predicates correspond to $\mathbf{v}$'s feature values.

### 2.3.2 Limitations of Model-Agnostic Methods

Model-agnostic approaches disregard the intricacies of the ML model itself and, instead, focus on analyzing its input-output behavior. It's crucial to highlight that model-agnostic approaches are susceptible to several critical issues, including the generation of unsound explanations [189, 276, 178, 257] and out-of-distribution sampling [330, 224, 369, 368]. These issues exacerbate the problem of trust in AI. Relying on such unsound explanations could lead to catastrophic consequences in high-risk or safety-critical scenarios [305, 304, 303, 345, 289, 88, 162]. Besides unsoundness, other limitations of model-agnostic explanations have been reported [62, 330, 224, 108, 218, 201, 173, 174, 175].

### 2.3.3 Formal Explainability

In contrast with the model-agnostic approaches [299, 247, 300, 148] which offer no guarantees of rigor, recent work studied rigorous model-based approaches for explainability [178, 257, 253, 258, 194], known as formal XAI (FXAI) methods. FXAI aims to provide rigorous and provable explanations for ML models predictions. Current theoretical framework builds on two distinct formal explanations: abductive explanations and contrastive explanations.

**Abductive Explanations (AXps).** Prime implicant (PI) explanations [320] denote a minimal set of literals (relating a feature value $x_i$ and a constant $v_i \in \mathbb{D}_i$) that are sufficient for the prediction. PI-explanations are related with abduction,

and so are also referred to as abductive explanations (AXps) [187][2]. Moreover, abductive explanations are an example of explainability by feature selection, i.e. the selection of a subset of features as the explanation. Formally, given a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ where $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$ with $\kappa(\mathbf{v}) = c$, a set of features $\mathcal{X} \subseteq \mathcal{F}$ is a *weak abductive explanation* (or weak AXp) if the following predicate holds true[3]:

$$\mathsf{WAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c) := \forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge_{i \in \mathcal{X}} (x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) = c) \tag{2.15}$$

Moreover, a set of features $\mathcal{X} \subseteq \mathcal{F}$ is an *abductive explanation* (or AXp) if the following predicate holds true:

$$\begin{array}{rcl} \mathsf{AXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c) & := & \mathsf{WAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c) \land \\ & & \forall(\mathcal{X}' \subsetneq \mathcal{X}). \neg \mathsf{WAXp}(\mathcal{X}'; \mathbb{F}, \kappa, \mathbf{v}, c) \end{array} \tag{2.16}$$

Clearly, an AXp is any weak AXp that is subset-minimal (or irreducible). It is straightforward to observe that the definition of predicate $\mathsf{WAXp}$ is monotone, and so a AXp can instead be defined as follows:

$$\begin{array}{rcl} \mathsf{AXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c) & := & \mathsf{WAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c) \land \\ & & \forall(j \in \mathcal{X}). \neg \mathsf{WAXp}(\mathcal{X} \setminus \{j\}; \mathbb{F}, \kappa, \mathbf{v}, c) \end{array} \tag{2.17}$$

This alternative equivalent definition of abductive explanation is at the core of most algorithms for computing one AXp [187, 188, 276, 255, 193, 256, 198, 250, 182]. It is apparent that Formulas (2.15), (2.16), and (2.17) can be viewed as representing a (logic) *rule* of the form:

$$\textbf{IF} \ \land_{i \in \mathcal{X}} (x_i = v_i) \ \textbf{THEN} \ \kappa(\mathbf{x}) = c \tag{2.18}$$

Unless otherwise noted, this interpretation of explanations will be assumed throughout the thesis.

AXps can be interpreted as answering a "**Why?**" question, i.e. why is some prediction made given some point in feature space? The answer to this question is a (minimal, or irreducible) set of the features, which is sufficient (or entails) the prediction.

[2] PI-explanations were first proposed in the context of boolean classifiers based on restricted bayesian networks [320]. Independent work [187] studied PI-explanations in the case of for more general classification functions, i.e. not necessarily boolean, and related instead explanations with abduction. This thesis follows the formalizations used in more recent work [255, 198, 182, 256, 81, 180, 257].

[3] Each predicate associated with a given concept will be noted in sans-serif letterform. When referring to the same concept in the text, the same acronym will be used, but in standard letterform. For example, the predicate name $\mathsf{AXp}$ will be used in logic statements, and the acronym AXp will be used throughout the text.

**Contrastive Explanations (CXps).** Similarly to the case of AXps, one can define (weak) contrastive explanations (CXps) [266, 185][4]. $\mathcal{Y} \subseteq \mathcal{F}$ is a weak CXp for a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ if,

$$\mathsf{WCXp}(\mathcal{Y}; \mathbb{F}, \kappa, \mathbf{v}, c) := \exists(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i)\right] \land (\kappa(\mathbf{x}) \neq c) \tag{2.19}$$

Thus, given a local explanation problem $\mathcal{E}$, a weak CXp is a set of features which, if allowed to take any value from their domain, then there is an assignment to the features that changes the prediction to a class other than $c$. Furthermore, a set $\mathcal{Y} \subseteq \mathcal{F}$ is a CXp if, besides being a weak CXp, it is also subset-minimal, i.e.

$$\begin{array}{rcl} \mathsf{CXp}(\mathcal{Y}; \mathbb{F}, \kappa, \mathbf{v}, c) & := & \mathsf{WCXp}(\mathcal{Y}; \mathbb{F}, \kappa, \mathbf{v}, c) \land \\ & & \forall(\mathcal{Y}' \subsetneq \mathcal{Y}).\neg\mathsf{WCXp}(\mathcal{Y}'; \mathbb{F}, \kappa, \mathbf{v}, c) \end{array} \tag{2.20}$$

Similarly to the case of AXps, it is straightforward to observe that the definition of predicate $\mathsf{WCXp}$ is monotone, and so a CXp can instead be defined as follows:

$$\begin{array}{rcl} \mathsf{CXp}(\mathcal{Y}; \mathbb{F}, \kappa, \mathbf{v}, c) & := & \mathsf{WCXp}(\mathcal{Y}; \mathbb{F}, \kappa, \mathbf{v}, c) \land \\ & & \forall(j \in \mathcal{Y}).\neg\mathsf{WCXp}(\mathcal{Y} \setminus \{j\}; \mathbb{F}, \kappa, \mathbf{v}, c) \end{array} \tag{2.21}$$

As with the case of AXps, this alternative equivalent definition of constraints explanation is at the core of most algorithms for computing one CXp [193, 256, 198, 250, 182].

A CXp can be viewed as a possible answer to a "**Why Not?**" question, i.e. why isn't the classifier's prediction a class other than $c$? (Clearly, the definition can be adapted to the case when we seek a concrete change of class.) A different perspective for a contrastive explanation is as the answer to a "**How?**" question, i.e. how to change the features so as to change the prediction. In recent literature this alternative view has been investigated under the name *actionable recourse* [338, 343, 209, 208].

Throughout the thesis, we will omit the parameterization associated with each predicate, and so, we will use the notation $\mathsf{AXp}(\mathcal{X})$ instead of $\mathsf{AXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c)$, when the parameters are clear from the context. The same convention applies to $\mathsf{CXp}(\cdot)$.

**Duality Between AXps and CXps.** Given the definitions of AXp and CXp, and building on Reiter's seminal work [297], recent work [186, 185] proved the following duality between minimal hitting sets:

**Proposition 1** (MHS duality between AXps and CXps [186, 185])**.** Given a local explanation problem $\mathcal{E}$, $\mathcal{X}$ is an AXp iff $\mathcal{X}$ is a minimal hitting set of the set of CXps. $\mathcal{Y}$ is a CXp iff $\mathcal{Y}$ is a minimal hitting set of the set of AXps.

[4] Contrastive explanations are related with counterfactual explanations [238]. For simple literal-based explanation functions contrastive correspond counterfactual explanations. However, this is not the case for more complex explanation functions.

Chapter 3

# Formal Explanations for Tractable Decision Graphs

This chapter shows that for a wide range of classifiers, globally referred to as decision graphs, and which include decision trees and binary decision diagrams, but also their multi-valued variants, there exist polynomial-time algorithms for computing one abductive explanation. In addition, this chapter also proposes a polynomial-time algorithm for computing one contrastive explanation. These novel algorithms build on explanation graphs (XpG's). XpG's denote a graph representation that enables both theoretical and practically efficient computation of explanations for decision graphs.

Furthermore, this chapter proposes a practically efficient solution for the enumeration of explanations. For the concrete case of decision trees, this chapter shows that the set of all contrastive explanations can be enumerated in polynomial time. Finally, the experimental results validate the practical applicability of the algorithms proposed in this chapter on a wide range of publicly available benchmarks.

## 3.1 Introduction

The emerging societal impact of Machine Learning (ML) and its foreseen deployment in safety critical applications, puts additional demands on approaches for verifying and explaining ML models [353]. The vast majority of approaches for explainability in ML (often referred to as eXplainable AI (XAI) [149]) are heuristic, offering no formal guarantees of soundness, with well-known examples including tools like LIME, SHAP or Anchors [299, 247, 300]. (Recent surveys [148] cover a wider range of heuristic methods.) Moreover, recent work has shed light on the important practical limitations of heuristic XAI approaches [276, 189, 62, 330, 224, 108, 178].

In contrast, formal approaches to XAI have been proposed in recent years [320, 187, 321, 188, 96, 29, 26] (albeit it can be related to past work on logic-based explanations (e.g. [316, 116, 287])). The most widely studied form of explanation consists in the identification of prime implicants (PI) of the decision function associated with an ML classifier, being referred to as PI-explanations, and are also referred to as abductive explanations(AXps) [187]. Although AXps offer important formal guarantees, e.g. they represent minimal sufficient reasons for a prediction, they do have their own drawbacks. First, in most settings, finding one AXp is NP-hard, and in some settings scalability is an issue [320, 187]. Second, users have little control on the size of computed AXps (and it is well-known the difficulty that humans have in

grasping complex concepts). Third, there can be many AXps, and it is often unclear which ones are preferred. Fourth, in practice users may often prefer high-level explanations, in contrast with feature-based, low-level explanations. Despite these drawbacks, it is plain that AXps offer a sound basis upon which one can expect to develop theoretically sound and practically effective approaches for computing explanations. For example, more recent work has demonstrated the tractability of AXps for some ML models [193, 29, 255, 256, 196], in some cases allowing for polynomial delay enumeration [255]. Also, recent work [178, 198, 182, 196] showed that, even for ML models for which computing an AXp is NP-hard, scalability may not be an obstacle.

Moreover, it was recently shown that finding explanations can be crucial even for ML models that are generally deemed interpretable[1]. One such example are decision trees [193]. Decision trees (DTs) are not only among the most widely used ML models, but are also generally regarded as interpretable [57, 123, 298, 272, 309, 269, 266, 148, 304, 358, 325]. However, recent work [193] has shown that paths in DTs may contain literals that are irrelevant for identifying minimal sufficient reasons for a prediction, and that the number of redundant literals can grow asymptotically as large as the number of features. Furthermore, it was also shown [193] that AXps for DTs can be computed in polynomial time. Moreover, independent work showed that finding a smallest explanation is hard for NP [35], thus hinting at the need to finding AXps in the case of DTs.

This chapter complements this earlier work with several novel results. First, the chapter considers AXps and CXps [266, 185], which will be jointly referred to as explanations (XPs). Second, the chapter shows that XPs can be computed in polynomial time for a much larger class of classifiers, which will be conjointly referred to as *decision graphs* [281, 215][2]. For that, the chapter introduces a new graph representation, namely the *explanation graph*, and shows that for any classifier (and instance) that can be reduced to an explanation graph, XPs can be computed in polynomial time. (For example, multi-valued variants of decision trees, graphs or diagrams can be reduced to explanation graphs.) The chapter also shows that the MARCO algorithm for enumerating MUSes/MCSes [231] can be adapted to the enumeration of XPs, yielding a solution that is very efficient in practice. For the case of DTs, the chapter proves that the set of all CXps can be computed in polynomial time. In turn, this result offers an alternative approach for the enumeration of AXps, e.g. based on hitting set dualization [297, 232].

The chapter is organized as follows. Section 3.2 relates the chapter's contributions with earlier work. Section 3.3 studies explanation graphs (XpG's), and shows how XpG's can be used for computing explanations. Afterwards, Section 3.4 describes algorithms computing one XP (either AXp or CXp) of XpG's, and a

[1] Interpretability is regarded a subjective concept, with no accepted rigorous definition [234]. In this chapter, we equate interpretability with explanation succinctness.

[2] The term *decision graph* is also used in the context of Bayesian Networks [202, 92], and more recently in explainability [320, 321]. However, and to the best of our knowledge, the term "decision graph" was first proposed in the early 90s [281] to enable more compact representation of DTs.

MARCO-like algorithm for the enumeration of XPs. Section 3.4 also proves that for DTs, the set of all CXps can be computed in polynomial time. Section 3.5 discusses experimental results of explaining DTs and reduced ordered binary decision diagrams, including AXps, CXps and their enumeration. Finally, the chapter concludes in Section 3.6.

## 3.2 Related Work

This chapter can be related with recent work on bayesian classifiers and decision graphs [320, 321, 96], but also tractable boolean circuits from the knowledge compilation (KC) map [29, 26]. In addition, we build on the recent results on the interpretability and the need for explainability of DTs [193]. The algorithms described in some of the previous work [320, 321, 96] cover AXps (and also minimum cardinality explanations, which we do not consider), but do not consider contrastive explanations. The focus of this earlier work is on ordered decision diagrams, and the proposed algorithms operate on binary features. Furthermore, the proposed algorithms are based on the compilation to some canonical representation (referred to as an ODD). If the goal is to find a few explanations, the algorithms described in this chapter are essentially guaranteed to scale in practice, whereas compilation to a canonical representation is less likely to scale (e.g. see [255]). Similarly, other recent work [29] investigates tractable boolean circuits from the knowledge compilation map, which consider binary features. In addition, the tractable classifiers considered in [29] for AXps do not intersect those studied in this chapter. In a companion work, [26] prove that for several XAI queries proposed in [29], including AXp extraction, there exist polynomial algorithms for the case of DTs. In [35], the focus is on the complexity of *smallest* AXps, and the results prove its tractability for FBDDs, which generalize OBDDs and DTs. Lastly, [105, 106, 21, 20] show that computing SHAP explanations [247] is tractable for the KC languages d-DNNFs, including FBDDs, SDDs OBDDs and DTs [99].

## Running Examples

Throughout this chapter, we will use the following DT and OMDD as our running examples.

**Example 1.** For the DT in Figure 3.1, $\mathcal{F} = \{1, 2, 3, 4\}$, denoting respectively Age ($\in \{\text{W}, \text{T}, \text{O}\}$), Income ($\in \{\text{L}, \text{M}, \text{H}\}$), Student ($\in \{\text{N}, \text{Y}\}$) and Credit Rating ($\in \{\text{P}, \text{F}, \text{E}\}$). The prediction is the type of hardware bought, with N denoting No Hardware, T denoting a Tablet and L denoting a Laptop. For Age, W, T and O denote, respectively, Age $< 30$ (tWenties or younger), $30 \leq$ Age $< 40$ (Thirties) and $40 \leq$ Age (forties or Older). For Income, L, M, H denote, respectively, (L)ow, (M)edium, and (H)igh. For Student, N denotes not a student and Y denotes a student. Finally, for Credit Rating, P, F and E denote, respectively, (P)oor, (F)air

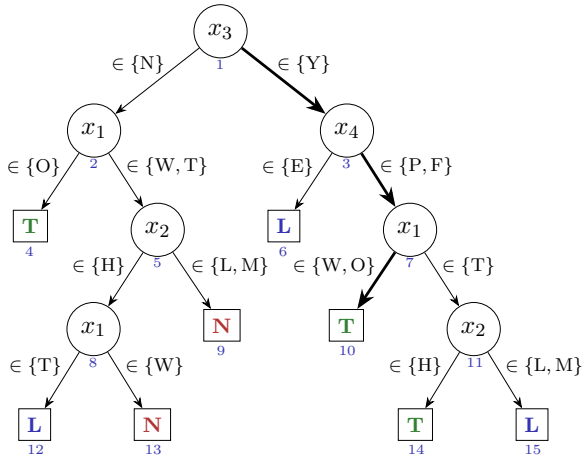



Figure 3.1: Example DT, $\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P})$ and $\kappa(\mathbf{v}) = \mathbf{T}$

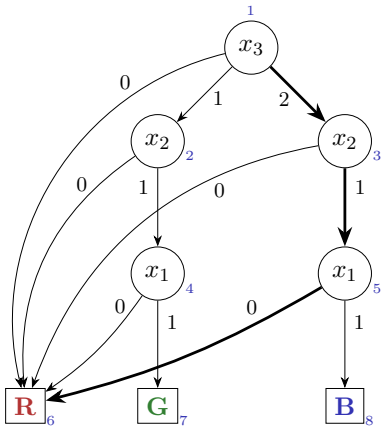



Figure 3.2: Example OMDD, $\mathbf{v} = (0, 1, 2)$ and $\kappa(\mathbf{v}) = \mathbf{R}$

and (E)xcellent. For the instance $\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P})$, with prediction T (i.e. Tablet), the consistent path is shown highlighted.

**Example 2.** For the OMDD in Figure 3.2, $\mathcal{F} = \{1, 2, 3\}$, with $D_1 = D_2 = \{0, 1\}$, $D_3 = \{0, 1, 2\}$. The prediction is one of three classes $\mathcal{K} = \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$. For the instance $\mathbf{v} = (0, 1, 2)$, with prediction R, the consistent path is shown highlighted.

## 3.3 Explanation Graphs

A difficulty with reasoning about explanations for DTs, DGs, OBDDs and OMDDs (and also in the case of other examples of ML models), is the multitude of cases that one needs to consider. For the concrete case of OBDDs, features are restricted to be boolean. However, for DTs and DGs, features can be boolean, categorical, integer

or real. Moreover, for OMDDs, features can be boolean, categorical or integer. Also, it is often the case that $|\mathcal{K}| > 2$. Explanation graphs (XpG) are a graph representation that abstracts away all the details that are effectively unnecessary for computing AXps or CXps. In turn, this facilitates the construction of unified explanation procedures.

**Definition 36.** An Explanation Graph (XpG) is a 5-tuple $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$, where:

1. $G_{\mathcal{D}} = (V_{\mathcal{D}}, E_{\mathcal{D}})$ is a labeled DAG, such that:
   - $V_{\mathcal{D}} = T_{\mathcal{D}} \cup N_{\mathcal{D}}$ is the set of nodes, partitioned into the terminal nodes $T_{\mathcal{D}}$ (with $\deg^+(q) = 0$, $q \in T_{\mathcal{D}}$) and the non-terminal nodes $N_{\mathcal{D}}$ (with $\deg^+(p) > 0$, $p \in N_{\mathcal{D}}$);
   - $E_{\mathcal{D}} \subseteq V_{\mathcal{D}} \times V_{\mathcal{D}}$ is the set of (directed) edges.
   - $G_{\mathcal{D}}$ is such that there is a single node with indegree equal to 0, i.e. the root (or source) node.
2. $S = \{s_1, \ldots, s_m\}$ is a set of variables;
3. $\upsilon : N_{\mathcal{D}} \to S$ is a total function mapping each non-terminal node to one variable in $S$.
4. $\alpha_V : V_{\mathcal{D}} \to \{0, 1\}$ labels nodes with one of two values.
   ($\alpha_V$ is required to be defined only for terminal nodes.)
5. $\alpha_E : E_{\mathcal{D}} \to \{0, 1\}$ labels edges with one of two values.

In addition, an XpG $\mathcal{D}$ must respect the following properties:

i. For each non-terminal node, there is at most one outgoing edge labeled 1; all other outgoing edges are labeled 0.
ii. There is exactly one terminal node $t \in T$ labeled 1 that can be reached from the root node with (at least) one path of edges labeled 1.

We refer to a *tree XpG* when the DAG associated with the XpG is a tree. Given a DG $\mathcal{G}$ and an instance $(\mathbf{v}, c)$, the (unique) mapping to an XpG is obtained as follows:

1. The same DAG is used.
2. Terminal nodes labeled $c$ in $\mathcal{G}$ are labeled 1 in $\mathcal{D}$. Terminal nodes labeled $c' \neq c$ in $\mathcal{G}$ are labeled 0 in $\mathcal{D}$.
3. A non-terminal node associated with feature $i$ in $\mathcal{G}$ is associated with $s_i$ in $\mathcal{D}$.
4. Any edge labeled with a literal that is consistent with $\mathbf{v}$ in $\mathcal{G}$ is labeled 1 in $\mathcal{D}$. Any edge labeled with a literal that is not consistent with $\mathbf{v}$ in $\mathcal{G}$ is labeled 0 in $\mathcal{D}$.

Since we can represent DTs, OBDDs or OMDDs with DGs, then the construction above ensures that we can also create XpG's for any of these classifiers.

The following examples illustrate the construction of XpG's for the chapter's two running examples.

**Example 3.** For the DT of Example 1 (shown in Figure 3.1, given the instance $(\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P}), \mathrm{T})$, and letting $S = (s_1, s_2, s_3, s_4)$, with each $s_i$ associated with feature $i$, the resulting XpG is shown in Figure 3.3.

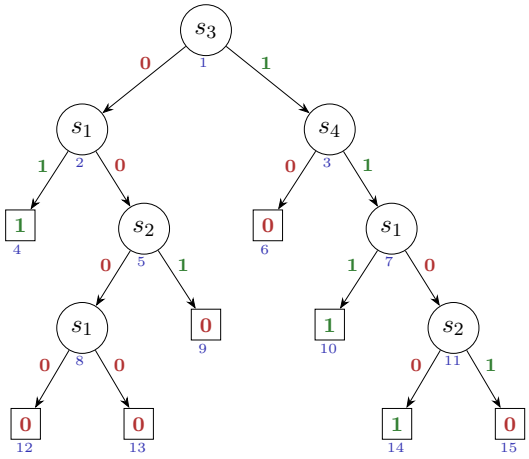



Figure 3.3: XpG for the DT in Figure 3.1, given $\mathbf{v} = (\mathrm{O}, \mathrm{L}, \mathrm{Y}, \mathrm{P})$

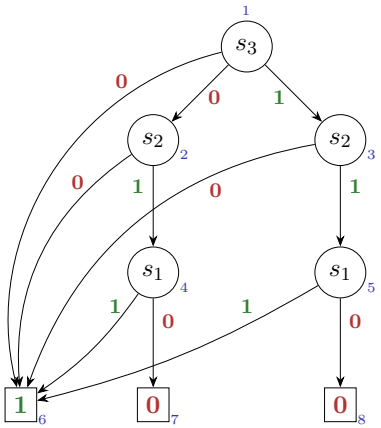



Figure 3.4: XpG for the OMDD of Figure 3.2, given $\mathbf{v} = (0, 1, 2)$

**Example 4.** For the OMDD of Example 2 (shown in Figure 3.2), given the instance $((0, 1, 2), \mathrm{R})$, and letting $S = (s_1, s_2, s_3)$, with each $s_i$ associated with feature $i$, the resulting XpG is shown in Figure 3.4.

**Evaluation of XpG's.** Given an XpG $\mathcal{D}$, let $\mathbb{S} = \{0, 1\}^m$, i.e. the set of possible assignments to the variables in $S$. The evaluation function of the XpG, $\sigma_{\mathcal{D}} : \mathbb{S} \to \{0, 1\}$, is based on the auxiliary *activation* function $\varepsilon : \mathbb{S} \times V_{\mathcal{D}} \to \{0, 1\}$. Moreover, for a point $\mathbf{s} \in \mathbb{S}$, $\sigma_{\mathcal{D}}$ and $\varepsilon$ are defined as follows:

1. If $r$ is the root node of $G_{\mathcal{D}}$, then $\varepsilon(\mathbf{s}, r) = 1$.
2. Let $p \in \mathsf{parent}(r)$ (i.e. a node can have multiple parents) and let $s_i = \upsilon(p)$.

$\varepsilon(\mathbf{s}, r) = 1$ iff $\varepsilon(\mathbf{s}, p) = 1$ and either $\alpha_E(p, r) = 1$ or $s_i = 0$, i.e.

$$\varepsilon(\mathbf{s}, r) \equiv \bigvee_{\substack{p \in \mathsf{parent}(r) \\ \wedge \neg\alpha_E(p,r)}} (\varepsilon(\mathbf{s}, p) \wedge \neg s_i) \bigvee_{\substack{p \in \mathsf{parent}(r) \\ \wedge \alpha_E(p,r)}} \varepsilon(\mathbf{s}, p) \tag{3.1}$$

3. $\sigma_{\mathcal{D}}(\mathbf{s}) = 1$ iff for every terminal node $t \in T_{\mathcal{D}}$, with $\alpha_V(t) = 0$, it is also the case that $\varepsilon(\mathbf{s}, t) = 0$, i.e.

$$\sigma_{\mathcal{D}}(\mathbf{s}) \equiv \bigwedge\nolimits_{t \in T_{\mathcal{D}} \wedge \neg\alpha_V(t)} \neg\varepsilon(\mathbf{s}, t) \tag{3.2}$$

Observe that terminal nodes labeled 1 are irrelevant for defining the evaluation function. Their existence is implicit (i.e. at least one terminal node with label 1 must exist and be reachable from the root when all the $s_i$ variables take value 1), but the evaluation of $\sigma_{\mathcal{D}}$ is oblivious to their existence. Furthermore, and as noted above, we must have $\sigma_{\mathcal{D}}(1, \ldots, 1) = 1$. If the graph has some terminal node labeled 0, then $\sigma_{\mathcal{D}}(0, \ldots, 0) = 0$.

**Example 5.** For the DT of Figure 3.1, and given the XpG of Figure 3.3, the evaluation function is defined as follows:

$$\sigma_{\mathcal{D}}(\mathbf{s}) \leftrightarrow \left( \bigwedge\nolimits_{r \in \{6,9,12,13,15\}} \neg\varepsilon(\mathbf{s}, r) \right)$$

with,

$$\begin{array}{l} [\varepsilon(\mathbf{s}, 1) \leftrightarrow 1] \wedge [\varepsilon(\mathbf{s}, 2) \leftrightarrow \varepsilon(\mathbf{s}, 1) \wedge \neg s_3] \wedge \\ [\varepsilon(\mathbf{s}, 3) \leftrightarrow \varepsilon(\mathbf{s}, 1)] \wedge [\varepsilon(\mathbf{s}, 5) \leftrightarrow \varepsilon(\mathbf{s}, 2) \wedge \neg s_1] \wedge \\ [\varepsilon(\mathbf{s}, 6) \leftrightarrow \varepsilon(\mathbf{s}, 3) \wedge \neg s_4] \wedge [\varepsilon(\mathbf{s}, 7) \leftrightarrow \varepsilon(\mathbf{s}, 3)] \wedge \\ [\varepsilon(\mathbf{s}, 8) \leftrightarrow \varepsilon(\mathbf{s}, 5) \wedge \neg s_2] \wedge [\varepsilon(\mathbf{s}, 9) \leftrightarrow \varepsilon(\mathbf{s}, 5)] \wedge \\ [\varepsilon(\mathbf{s}, 11) \leftrightarrow \varepsilon(\mathbf{s}, 7) \wedge \neg s_1] \wedge [\varepsilon(\mathbf{s}, 12) \leftrightarrow \varepsilon(\mathbf{s}, 8) \wedge \neg s_1] \wedge \\ [\varepsilon(\mathbf{s}, 13) \leftrightarrow \varepsilon(\mathbf{s}, 8) \wedge \neg s_1] \wedge [\varepsilon(\mathbf{s}, 15) \leftrightarrow \varepsilon(\mathbf{s}, 11)] \end{array}$$

(where, for simplicity and for reducing the number of parenthesis, the operator $\wedge$ has precedence over the operator $\leftrightarrow$.) Observe that $\sigma_{\mathcal{D}}(1, 1, 1, 1) = 1$ and $\sigma_{\mathcal{D}}(0, 0, 0, 0) = 0$.

**Example 6.** For the OMDD of Figure 3.2, and given the XpG of Figure 3.4, the evaluation function is defined as follows:

$$\sigma_{\mathcal{D}}(\mathbf{s}) \leftrightarrow \left( \bigwedge\nolimits_{r \in \{7,8\}} \neg\varepsilon(\mathbf{s}, r) \right)$$

with,

$$\begin{array}{l} [\varepsilon(\mathbf{s}, 1) \leftrightarrow 1] \wedge [\varepsilon(\mathbf{s}, 2) \leftrightarrow \varepsilon(\mathbf{s}, 1) \wedge \neg s_3] \wedge \\ [\varepsilon(\mathbf{s}, 3) \leftrightarrow \varepsilon(\mathbf{s}, 1)] \wedge [\varepsilon(\mathbf{s}, 4) \leftrightarrow \varepsilon(\mathbf{s}, 2)] \wedge \\ [\varepsilon(\mathbf{s}, 5) \leftrightarrow \varepsilon(\mathbf{s}, 3)] \wedge [\varepsilon(\mathbf{s}, 7) \leftrightarrow \varepsilon(\mathbf{s}, 4) \wedge \neg s_1] \wedge \\ [\varepsilon(\mathbf{s}, 8) \leftrightarrow \varepsilon(\mathbf{s}, 5) \wedge \neg s_1] \end{array}$$

Again, we have $\sigma_{\mathcal{D}}(1, 1, 1, 1) = 1$ and $\sigma_{\mathcal{D}}(0, 0, 0, 0) = 0$.

**Properties of XpG's.** The definition of $\sigma_{\mathcal{D}}$ is such that the evaluation function is monotone (where we define $0 \preceq 1$, $\mathbf{s}_1 \preceq \mathbf{s}_2$ if for all $i$, $s_{1,i} \preceq s_{2,i}$, and for

monotonicity we require $\mathbf{s}_1 \preceq \mathbf{s}_2 \rightarrow \sigma_{\mathcal{D}}(\mathbf{s}_1) \preceq \sigma_{\mathcal{D}}(\mathbf{s}_2)$.

**Proposition 2.** Given an XpG $\mathcal{D}$, $\sigma_{\mathcal{D}}$ is monotone.

*Proof.* Observe that $\varepsilon$ is monotone (and negative) on $\mathbf{s} \in \mathbb{S}$, and $\sigma_{\mathcal{D}}$ is monotone (and negative) on $\varepsilon$. Hence, $\sigma_{\mathcal{D}}$ is monotone (and positive) on $\mathbf{s}$. □

Given the definition of $\sigma_{\mathcal{D}}$, any PI will consist of a conjunction of positive literals [86]. Furthermore, we can view an XpG as a classifier, mapping features $\{1, \ldots, m\}$ (each feature $i$ associated with a variable $s_i \in S$) into $\{0, 1\}$, with instance $((1, \ldots, 1), 1)$. As a result, we can compute the AXps and CXps of an XpG $\mathcal{D}$ (given the instance $((1, \ldots, 1), 1)$).

**Example 7.** Observe that by setting $s_2 = s_3 = 0$, we still guarantee that $\sigma_{\mathcal{D}}(1, 0, 0, 1) = 1$. However, setting either $s_1 = 0$ or $s_4 = 0$, will cause $\sigma_{\mathcal{D}}$ to change value. Hence, one AXp for the XpG is $\{1, 4\}$. With respect to the original instance $((\text{O}, \text{L}, \text{Y}, \text{P}), \text{T})$, selecting $\{1, 4\}$ indicates that $(x_1 = \text{O}) \land (x_4 = \text{P})$ (i.e. Age in the forties or Older and a Credit Rating of Poor) suffices for the prediction of T.

**Example 8.** With respect to Example 6, we can observe that $s_2$ is not used for defining $\sigma_{\mathcal{D}}$. Hence, it can be set to 0. Also, as long as $s_1 = 1$, the prediction will remain unchanged. Thus, we can also set $s_3$ to 0. As a result, one AXp is $\{1\}$. With respect to the original instance $((x_1, x_2, x_3), c) = ((0, 1, 2), \text{R})$, selecting $\{1\}$ indicates that $x_1 = 0$ suffices for the prediction of R.

As suggested by the previous discussion and examples, we have the following result.

**Proposition 3.** There is a one-to-one mapping between AXps and CXps of $\sigma_{\mathcal{D}}$ and the AXps and CXps of the original classification problem (and instance) from which the XpG $\mathcal{D}$ is obtained.

*Proof.* The construction of the XpG from a DG ensures that for any node in the XpG, if $\varepsilon(\mathbf{s}, r) = 1$, then there exists some assignment to the features corresponding to unset variables, such that there is one consistent path in the DG from the root to $r$. Thus, if for some pick of unset variables, we have that $\varepsilon(\mathbf{s}, q) = 1$, for some $q \in T_{\mathcal{D}}$ with $\alpha_V(q) = 0$, then that guarantees that in the DG there is an assignment to the features associated with the unset variables, such that a prediction other than $c$ is obtained. □

## 3.4 Computing Explanations

It is well-known that prime implicants of monotone functions can be computed in polynomial time (e.g. [141, 142]). Moreover, whereas there are algorithms for finding one PI of a monotone function in polynomial time, there is evidence that enumeration of PIs cannot be achieved with polynomial delay [150].

Nevertheless, and given the fact that $\sigma_{\mathcal{D}}$ is defined on a DAG, this chapter proposes dedicated algorithms for computing one AXp and one CXp which build on iterative graph traversals. Furthermore, the MARCO algorithm [231] is adapted to exploit the algorithms for computing one AXp and one CXp, in the process ensuring that AXps/CXps can be enumerated with exactly one SAT oracle call per each computed explanation. (A recent work on explaining monotonic classifiers [256] proposes a poly-time algorithm to compute one AXp (resp. CXp) and a practically efficient algorithm for the iterative enumeration of XPs.)

### 3.4.1 Finding One Explanation

Different polynomial-time algorithms can be envisioned for finding one prime implicant of an XpG (and also of a monotone function). For the concrete case of $\sigma_{\mathcal{D}}$, we consider the well-known deletion-based algorithm [70], which iteratively removes literals from the implicant, and checks the value of $\sigma_{\mathcal{D}}$ using the DAG representation. (It is also plain that we could consider instead the algorithms QuickXplain [205] or Progression [259], or any other algorithm for finding a minimal set subject to a monotone predicate [260].)

As highlighted in the running examples, if $\sigma_{\mathcal{D}}(\mathbf{u}) = 1$, for some $\mathbf{u} \in \mathbb{S}$, then in the original classifier this means the prediction remains unchanged. The only way we have to change the prediction is to allow some features to take some other value from their domain. As a result, we equate $s_i = 1$ with declaring the original feature as *fixed*, whereas we equate $s_i = 0$ with declaring the original feature as *free*. By changing some $s_i$ from 1 to 0, we are allowing some of the features to take one value from their domains. If we manage to change the value of the evaluation function to 0, this means that in the original classifier there exists a pick of values to the free features which allows the prediction to change to some class other than $c$. As a result, the algorithms proposed in this section are solely based on finding a subset maximal set of features declared free (respectively, fixed), which is sufficient for the prediction not to change (respectively, to change).

To enable the integration of the algorithms, the basic algorithms for finding one XP are organized such that one XP is computed given a starting seed.

**Checking path to node with label 0.** All algorithms are based on graph traversals, which check whether a prediction of 0 can be reached given a set of value picks for the variables in $S$. Besides, the existence of a path to nodes with label 0 implies that there is a weak CXp thus, the predicate $\mathsf{WAXp}(\cdot)$ and $\mathsf{WCXp}(\cdot)$ can be defined as follows:

$$\begin{aligned} \mathsf{WCXp}(Z) &:= \mathsf{pathToZero}(\mathcal{D}, Z) \\ \mathsf{WAXp}(Z) &:= \neg\, \mathsf{WCXp}(S \setminus Z) \end{aligned} \tag{3.3}$$

This graph traversal algorithm is simple to envision, and is shown in Algorithm 1. As can be observed, the algorithm returns 1 if a terminal labeled 0 can be reached. Otherwise, it returns 0. Variables in set $Z$ serve to ignore the values of outgoing

**Algorithm 1** Check existence of path to 0-labeled terminal

**Input**: XpG $\mathcal{D} = (G_{\mathcal{D}}, S, v, \alpha_V, \alpha_E)$; Reference set: $Z \subseteq S$

```
1:  procedure pathToZero(D, Z)
2:      Q ← init(root(G_D))
3:      while not empty(Q) do
4:          (Q, p) ← dequeue(Q)
5:          if isTerminal(G_D, p) then
6:              if α_V(p) = 0 then
7:                  return true
8:          else
9:              s_i ← v(p)
10:             for all q ∈ children(G_D, p) do
11:                 if s_i ∈ Z or α_E(p, q) = 1 then
12:                     Q ← enqueue(Q, q)
13:     return false
```

edges of a node if the variable is in $Z$. The algorithm has a linear run time on the XpG's size (i.e. $|V_{\mathcal{D}}| + |E_{\mathcal{D}}|$).

**Algorithm 2** Extraction of one AXp given seed $\mathcal{X}$

**Input**: XpG $\mathcal{D} = (G_{\mathcal{D}}, S, v, \alpha_V, \alpha_E)$; Seed set: $\mathcal{X} \subseteq S$
**Output**: AXp $\mathcal{X}$

```
1:  procedure findAXp(D, X)
2:      for all s_i ∈ X do
3:          if WAXp(X \ {s_i}) then
4:              X ← X \ {s_i}
5:      return X
```

**Extraction of one AXp and one CXp given seed.** Given a seed set $\mathcal{X} \subseteq S$ of fixed variables, and so a set $\mathcal{Y} = S \setminus \mathcal{X}$ of free variables (which are *guaranteed* to be kept free), Algorithm 2 drops variables from $\mathcal{X}$ (i.e. makes variables free, and so allows the original features to take one of the values in their domains). Since $\sigma_{\mathcal{D}}$ is monotone, the deletion-based algorithm is guaranteed to find a subset-minimal set of fixed variables such that the XpG evaluates to 1.

Similarly, given a seed set $\mathcal{Y} \subseteq S$ of free variables, and so a set $\mathcal{X} = S \setminus \mathcal{Y}$ of fixed variables (which are *guaranteed* to be kept fixed), Algorithm 3 drops variables from $\mathcal{Y}$ (i.e. makes variables fixed, and so forces the original features to take the value specified by the instance).

### 3.4.2 Enumeration of Explanations

As indicated earlier in this section, we use a MARCO-like [231] algorithm for enumerating XPs of an XpG (see Algorithm 4). (An in-depth analysis of MARCO is

**Algorithm 3** Extraction of one CXp given seed $\mathcal{Y}$

**Input**: XpG $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$; Seed set: $\mathcal{Y} \subseteq S$
**Output**: CXp $\mathcal{Y}$

```
1: procedure findCXp(D, Y)
2:     for all s_i ∈ Y do
3:         if WCXp(Y \ {s_i}) then
4:             Y ← Y \ {s_i}
5:     return Y
```

**Algorithm 4** Enumeration of AXps and CXps

**Input**: XpG $\mathcal{D} = (G_{\mathcal{D}}, S, \upsilon, \alpha_V, \alpha_E)$

```
1: procedure Enumerate(D)
2:     H ← ∅                                   ▷ H defined on set S
3:     repeat
4:         (outc, r) ← SAT(H)
5:         if outc = true then
6:             X ← {s_i ∈ S | r_i = 1}
7:             Y ← {s_i ∈ S | r_i = 0}
8:             if not WCXp(Y) then
9:                 X ← findAXp(D, X)
10:                reportAXp(X)
11:                H ← H ∪ {(∨_{s_i∈X} ¬s_i)}
12:            else
13:                Y ← findCXp(D, Y)
14:                reportCXp(Y)
15:                H ← H ∪ {(∨_{s_i∈Y} s_i)}
16:    until outc = false
```

included in earlier work [231].) Algorithm 4 exploits hitting set duality between AXps and CXps [185], and represents the sets to hit (resp. block) as a set of positive (resp. negative) clauses $\mathcal{H}$, defined on a set of variables $S$. The algorithm iteratively calls a SAT oracle on $\mathcal{H}$ while the formula is satisfiable. Given a model, which splits $S$ into variables assigned value 1 (i.e. fixed) and variables assigned value 0 (i.e. free), we check if the model enables the prediction to change (i.e. we check the existence of a path to a terminal node labeled 0, with $\mathcal{Y}$ as the reference set). If no such path exists, then we extract one AXp, using $\mathcal{X}$ as the seed. Otherwise, we extract one CXp, using $\mathcal{Y}$ as the seed. The resulting XP is then used to block future assignments to the variables in $S$ from repeating XPs.

**Enumerating CXps for DTs.** The purpose of this section is to show that, if the XpG is a tree (e.g. in the case of a DT), then the number of CXps is polynomial on the size of the XpG. Furthermore, it is shown that the set of all CXps can be computed in polynomial time. This result has a number of consequences, some of

which are discussed in Chapter 6. For the concrete case of enumeration of XPs of tree XpG's, since we can enumerate all CXps in polynomial time, then we can exploit the well-known results of Fredman&Khachiyan [122] to prove that enumeration of AXps can be obtained in quasi-polynomial time. The key observation is that, since we can enumerate all the CXps in polynomial time, then we can construct the associated hypergraph, thus respecting the conditions of Fredman&Khachiyan's algorithms [122, 214].

**Proposition 4.** For a tree XpG, the number of CXps is polynomial on the size of the XpG, and can be enumerated in polynomial time.

*Proof.* To change the prediction, we must make a path to a prediction $c' \in \mathcal{K} \setminus \{c\}$ consistent. In a tree, the number of paths (connecting the root to a terminal) associated with a prediction in $c' \in \mathcal{K} \setminus \{c\}$ is linear on the size of the tree. Observe that each path yielding a prediction other that $c$ contributes at most one CXp, because the consistency of the path (in order to predict a class other than $c$) requires that all the inconsistent literals be allowed to take some consistent value. We can thus conclude that the number of CXps is linear on the size of a tree XpG. The algorithm for listing the CXps exploits the previous remarks, but takes into consideration that some paths may contribute candidate CXps that are supersets of others (and so not actual CXps); these must be filtered out. □

## 3.5 Experimental Results

This section presents the experiments carried out to assess the practical effectiveness of the proposed algorithms. The assessment is performed on the computation of AXp and CXp for two case studies of DGs: OBDDs and DTs.

**Experimental setup.** The experiments consider a selection of datasets that are publicly available and originate from UCI Machine Learning Repository [109], Penn Machine Learning Benchmarks [282] and openML [342]. These benchmarks are organized into two categories: the first category contains binary classification datasets with fully binary features, and counts 11 datasets; the second category comprises binary and multidimensional classification datasets with categorical and/or ordinal (i.e. integer or real-valued) features, and counts 34 datasets. Hence, the total number of considered datasets is 45. The subset of the binary datasets is considered for generating OBDDs, while the remaining selected datasets are used for learning DTs.

To learn OBDDs, we first train DLs on the given binary datasets and then compile the obtained DLs into OBDDs using the approach proposed in [275]. DLs are learned using Orange3 [104], the order of rules is determined by Orange3 and the last rule is the default rule. The compilation to OBDDs is performed using BuDDy [233]. For training DTs, we use the learning tool IAI [43, 177], which provides shallow DTs that are highly accurate. To achieve high accuracy in the

Table 3.1: Listing all XPs (AXp's and CXp's) for OBDDs. Columns $m$ and #TI report, respectively, the number of features, and the number of tested instances, in the dataset. (Note that for a dataset containing more than 1000 instances, 30% of its instances, randomly selected, are used to be explained. Moreover, duplicate rows in the datasets are filtered.) Column XPs reports the average number of total explanations (AXp's and CXp's). Sub-Columns #N and %A show, respectively, total number of nodes and test accuracy of an OBDD. Sub-columns M and avg of column AXp (resp., CXp ) show, respectively, the maximum and average number of explanations. The average length of an explanation (AXp/CXp) is given as %L. Sub-columns M and avg of column RunTime reports, respectively, the maximal and average time in second to list all the explanations for all tested instances.

| Dataset | $m$ | #TI | OBDD | | XPs | AXp | | | CXp | | | Runtime | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #N | %A | avg | M | avg | %L | M | avg | %L | M | avg |
| corral | 6 | 64 | 6 | 100 | 4 | 4 | 2 | 34 | 4 | 2 | 22 | 0.002 | 0.001 |
| dbworld-bodies | 4702 | 62 | 7 | 92 | 4 | 2 | 1 | 1 | 3 | 2 | 1 | 0.002 | 0.001 |
| dbworld-bodies-s | 3721 | 62 | 6 | 84 | 3 | 3 | 1 | 1 | 4 | 2 | 1 | 0.001 | 0.001 |
| dbworld-subjects | 242 | 63 | 14 | 84 | 5 | 2 | 1 | 2 | 5 | 4 | 1 | 0.003 | 0.001 |
| dbworld-subjects-s | 229 | 63 | 18 | 84 | 6 | 3 | 1 | 2 | 5 | 4 | 1 | 0.007 | 0.003 |
| mofn_3_7_10 | 10 | 251 | 21 | 98 | 11 | 33 | 5 | 34 | 33 | 7 | 23 | 0.022 | 0.005 |
| mux6 | 6 | 64 | 9 | 100 | 5 | 4 | 2 | 51 | 4 | 3 | 24 | 0.004 | 0.002 |
| parity5+5 | 10 | 222 | 71 | 80 | 8 | 11 | 2 | 59 | 15 | 6 | 14 | 0.015 | 0.006 |
| spect | 22 | 93 | 284 | 87 | 11 | 24 | 4 | 22 | 36 | 7 | 14 | 0.074 | 0.019 |
| threeOf9 | 9 | 205 | 33 | 95 | 8 | 16 | 3 | 39 | 18 | 5 | 21 | 0.017 | 0.004 |
| xd6 | 9 | 325 | 11 | 100 | 7 | 18 | 4 | 34 | 27 | 3 | 18 | 0.010 | 0.002 |

DTs, the maximum depth is tuned to 6 while the remaining parameters are kept in their default set up. (Note that the test accuracy achieved for the trained classifiers, both OBDDs and DTs, is always greater than 75%).

All the proposed algorithms are implemented in Python, in the `XpG` package[3]. The PySAT package [184] is used to instrument incremental SAT oracle calls in XP enumeration (see Algorithm 4) and the `dd` [4] package, implemented in Python and Cython, is used to integrate BuDDy, which is implemented in C. The experiments are performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Big Sur.

**Results.** Table 3.1 summarizes the obtained results of explaining OBDDs. As can be observed, the maximum running time to enumerate XPs is less than 0.074 sec for all tested XpG's in any OBDD and does not exceed 0.02 sec on average. In terms of the number of XPs, the total number of AXps and CXps per instance is relatively small. Thus the overall cost of the SAT oracle calls made for XP enumeration is negligible. In addition, these observations apply even for large OBDDs, e.g. OBDD learned from the *spect* dataset has 284 nodes and results in 11 XPs on average.

[3] `https://github.com/yizza91/xpg`
[4] `https://github.com/tulip-control/dd`

Table 3.2: Listing all XPs (AXp's and CXp's) for DTs. Sub-Columns #D, #N and %A report, respectively, tree's max depth, total number of nodes and test accuracy of a DT. The remaining columns hold the same meaning as described in the caption of Table 3.1.

| Dataset | $m$ | #TI | DT | | | XPs | AXp | | | CXp | | | Runtime | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | D | #N | %A | avg | M | avg | %L | M | avg | %L | M | avg |
| adult | 12 | 1766 | 6 | 83 | 78 | 8 | 11 | 2 | 41 | 12 | 5 | 13 | 0.010 | 0.003 |
| agaricus-lepiota | 22 | 2437 | 6 | 37 | 100 | 6 | 6 | 3 | 17 | 7 | 4 | 7 | 0.006 | 0.002 |
| anneal | 38 | 886 | 6 | 29 | 99 | 9 | 8 | 3 | 14 | 10 | 6 | 5 | 0.015 | 0.005 |
| bank | 19 | 10837 | 6 | 113 | 88 | 18 | 38 | 9 | 33 | 21 | 9 | 12 | 0.032 | 0.008 |
| cancer | 9 | 449 | 6 | 37 | 87 | 7 | 8 | 3 | 39 | 7 | 4 | 21 | 0.006 | 0.003 |
| car | 6 | 519 | 6 | 43 | 96 | 4 | 4 | 2 | 39 | 6 | 2 | 24 | 0.004 | 0.001 |
| chess | 36 | 959 | 6 | 33 | 97 | 7 | 10 | 3 | 12 | 10 | 5 | 5 | 0.012 | 0.003 |
| churn | 20 | 1500 | 6 | 21 | 75 | 2 | 1 | 1 | 5 | 1 | 1 | 5 | 0.002 | 0.001 |
| colic | 22 | 357 | 6 | 55 | 81 | 11 | 18 | 5 | 23 | 10 | 6 | 8 | 0.011 | 0.004 |
| collins | 23 | 485 | 6 | 29 | 75 | 4 | 1 | 1 | 11 | 4 | 3 | 5 | 0.002 | 0.001 |
| dermatology | 34 | 366 | 6 | 33 | 90 | 7 | 6 | 2 | 14 | 11 | 5 | 4 | 0.007 | 0.003 |
| divorce | 54 | 150 | 5 | 15 | 90 | 6 | 8 | 3 | 7 | 4 | 3 | 3 | 0.010 | 0.005 |
| dna | 180 | 901 | 6 | 61 | 90 | 10 | 28 | 4 | 3 | 12 | 5 | 2 | 0.097 | 0.036 |
| hayes-roth | 4 | 84 | 6 | 23 | 78 | 3 | 3 | 1 | 54 | 3 | 2 | 27 | 0.001 | 0.001 |
| hepatitis | 19 | 155 | 5 | 17 | 77 | 6 | 10 | 3 | 18 | 5 | 3 | 10 | 0.004 | 0.002 |
| house-votes-84 | 16 | 298 | 6 | 49 | 91 | 9 | 30 | 5 | 25 | 10 | 4 | 13 | 0.016 | 0.003 |
| iris | 4 | 149 | 5 | 23 | 90 | 5 | 3 | 2 | 58 | 4 | 3 | 39 | 0.003 | 0.001 |
| irish | 5 | 470 | 4 | 13 | 97 | 3 | 2 | 1 | 33 | 2 | 2 | 23 | 0.001 | 0.001 |
| kr-vs-kp | 36 | 959 | 6 | 49 | 96 | 7 | 23 | 4 | 12 | 8 | 4 | 5 | 0.014 | 0.003 |
| lymphography | 18 | 148 | 6 | 61 | 76 | 11 | 15 | 5 | 28 | 12 | 6 | 10 | 0.009 | 0.004 |
| promoters | 58 | 106 | 6 | 17 | 86 | 4 | 6 | 2 | 6 | 5 | 2 | 3 | 0.008 | 0.004 |
| monk1 | 6 | 124 | 4 | 17 | 100 | 3 | 2 | 1 | 38 | 3 | 2 | 18 | 0.002 | 0.001 |
| monk2 | 6 | 169 | 6 | 67 | 82 | 6 | 7 | 2 | 65 | 9 | 5 | 23 | 0.005 | 0.002 |
| monk3 | 6 | 122 | 6 | 35 | 80 | 4 | 6 | 2 | 45 | 4 | 3 | 23 | 0.004 | 0.001 |
| mouse | 5 | 57 | 3 | 9 | 83 | 3 | 4 | 1 | 41 | 3 | 2 | 25 | 0.001 | 0.001 |
| mushroom | 22 | 2438 | 6 | 39 | 100 | 6 | 5 | 2 | 18 | 7 | 4 | 7 | 0.007 | 0.002 |
| new-thyroid | 5 | 215 | 3 | 11 | 95 | 4 | 2 | 1 | 54 | 3 | 3 | 21 | 0.001 | 0.001 |
| pendigits | 16 | 3298 | 6 | 121 | 88 | 8 | 12 | 2 | 37 | 13 | 6 | 9 | 0.011 | 0.003 |
| seismic-bumps | 18 | 774 | 6 | 37 | 89 | 7 | 12 | 4 | 17 | 7 | 4 | 11 | 0.009 | 0.004 |
| shuttle | 9 | 17400 | 6 | 63 | 99 | 4 | 4 | 1 | 34 | 6 | 3 | 13 | 0.005 | 0.002 |
| soybean | 35 | 622 | 6 | 63 | 88 | 7 | 4 | 1 | 15 | 7 | 5 | 4 | 0.012 | 0.005 |
| spambase | 57 | 1262 | 6 | 63 | 75 | 10 | 22 | 3 | 11 | 15 | 7 | 3 | 0.019 | 0.005 |
| tic-tac-toe | 9 | 958 | 6 | 69 | 93 | 9 | 13 | 4 | 51 | 12 | 6 | 20 | 0.009 | 0.003 |
| zoo | 16 | 59 | 6 | 23 | 91 | 5 | 2 | 1 | 24 | 6 | 4 | 9 | 0.003 | 0.002 |

Similar observations can be made with respect to explanation enumeration for DTs, the results of which are detailed in Table 3.2. Exhaustive enumeration of XPs for a XpG built from a DT takes only a few milliseconds. Indeed, the largest average runtime (obtained for the *dna* dataset) is 0.036 sec. Furthermore and as can be observed, the average length %L of an XP is in general relatively small, compared to the total number of features of the corresponding dataset. Also, the total number of XPs per instance is on average less than 11 and never exceeds 18.

Although the DGs considered in the experiments can be viewed as relatively

small and shallow (albeit this only reflects the required complexity given the public datasets available), the run time of the enumerator depends essentially on solving a relatively simple CNF formula ($\mathcal{H}$) which grows linearly with the number of XPs. (The run time of the actual extractors in negligible.) This suggests that the proposed algorithms will scale for significantly larger DGs, characterized also by a larger total number of XPs.

## 3.6 Summary

This chapter introduces explanation graphs, which allow several classes of graph-based classifiers to be explained with the same algorithms. These algorithms allow for a single AXp or a single CXp to be computed in polynomial time, and enumeration of explanations to be achieved with a single call to a SAT oracle per computed explanation. This chapter also relates the evaluation of explanation graphs with monotone functions. In addition, this chapter proves that for decision trees, computing all contrastive explanations. The experimental results demonstrate the practical effectiveness of the ideas proposed in the chapter.

Chapter 4

# Formal Explanations for Tractable Boolean Circuits

Tractable boolean circuits find a growing number of practical uses, including in constraint programming, diagnosis, and machine learning, among others. One concrete example is the utilization of tractable boolean circuits as classifiers. These circuits can be compiled from complex machine learning models, such as Bayesian Networks. As a result, a natural question arises: How can we explain the predictions made by these circuits?

This chapter shows that for classifiers represented with some of the best-known tractable boolean circuits, different kinds of explanations can be computed in polynomial time. These tractable boolean circuits include deterministic decomposable negation normal form (d-DNNF) and any other tractable boolean circuit that is strictly less succinct than d-DNNF. Furthermore, this chapter also examines the conditions under which the polynomial time computation of explanations can be extended to boolean circuits that are more succinct than d-DNNF. The experimental results validate the practical applicability of the algorithms proposed in this chapter.

## 4.1 Introduction

The growing use of machine learning (ML) models in practical applications raises a number of concerns related with fairness, robustness, but also explainability [234, 353, 271]. Recent years have witnessed a number of works on computing explanations for the predictions made by ML models[1]. Approaches to computing explanations can be broadly categorized as heuristic [299, 247, 300], which offer no formal guarantees of rigor, and non-heuristic [320, 187, 96, 29], which in contrast offer strong guarantees of rigor. Non-heuristic explanation approaches can be further categorized into compilation-based [320, 321, 96] and oracle-based [187, 250].

Compilation-based approaches [320, 321] compile the decision function associated with an ML classifier into some propositional language (among those covered by the knowledge compilation map [99]). As a result, more recent work studied such propositional languages from the perspective of explainability, with the purpose of understanding the complexity of computing explanations [29, 35, 26], but also with

[1] There is a fast growing body of work on the explainability of ML models. Example references include [148, 309, 266, 265, 17, 267, 358, 274].

the goal of identifying examples of queries of interest [29, 26]. Furthermore, although recent work [29, 35] analyzed the complexity of explainability queries for classifiers represented with different propositional languages, it is also the case that it is unknown which propositional languages allow the expressible functions to be explained efficiently, and which do not. On the one hand, [29] proposes conditions not met by most propositional languages. On the other hand [35] studies restricted cases of propositional languages, but focusing on smallest AXps. Also, since one key motivation for the use of propositional languages is the efficiency of reasoning, namely with respect to specific queries and transformations [99], a natural question is whether similar results can be obtained in the setting of explainability.

This chapter studies the computational complexity of computing AXps [320] and CXps [266] for classifiers represented with well-known propositional languages. Concretely, the chapter shows that for any propositional language that implements in polynomial time the queries of consistency (**CO**) and validity (**VA**), and the transformation of conditioning (**CD**), then one AXp or one CXp can be computed in polynomial time. This requirement is strictly less stringent than another one proposed in earlier work [29], thus proving that explanations can be computed in polynomial time for a larger range of propositional languages. Concretely, the chapter shows that for classifiers represented with several propositional languages, that include d-DNNF, one AXp or one CXp can be computed in polynomial time. The result immediately generalizes to propositional languages less succinct than d-DNNF, e.g. OBDD, SDD, to name a few. Moreover, for the concrete case of SDDs, the chapter shows that practical optimizations lead to clear performance gains. Besides computing one explanation, one is often interested in obtaining multiple explanations, thus allowing a decision maker to get a better understanding of the reasons supporting a decision. As a result, the chapter also outlines a MARCO-like [231] algorithm for the enumeration of both AXps and CXps. Furthermore, the chapter studies the computational complexity of explaining generalizations of decision sets [223], and proposes conditions under which explanations can be computed in polynomial time.

The chapter is organized as follows. Section 4.2 relates the chapter's contributions with earlier work. Section 4.3 shows that for several classes of propositional languages, one AXp and one CXp can be computed in polynomial time. In addition, Section 4.3 also shows how to enumerate explanations requiring one NP oracle call (in fact a SAT reasoner call) for each computed explanation. Section 4.4 investigates a number of generalized classifiers, which can be built from propositional languages used as building blocks. Section 4.5 assesses the computation of explanations of d-DNNFs and SDDs in practical settings. Section 4.6 concludes the chapter.

## 4.2 Related Work

Although recent years have witnessed a growing interest in finding explanations of machine learning (ML) models [234, 148, 353, 271], explanations have been stud-

ied from different perspectives and in different branches of AI at least since the 80s [316, 116, 287], including more recently in constraint programming [14, 52, 133]. In the case of ML models, non-heuristic explanations have been studied in recent years [320, 187, 321, 276, 188, 189, 96, 179, 178, 29, 255, 35, 185, 193, 347, 198, 250, 182, 81, 169, 26, 257, 180, 322]. Some of these earlier works studied explanations for classifiers represented with propositional languages, namely those covered by the knowledge compilation map [320, 321, 96, 29, 35, 169, 26]. However, results on the efficient computation of explanations for classifiers represented with propositional languages are scarce. For example, [320, 321, 96] propose compilation algorithms (which are worst-case exponential) to generate the AXps from OBDDs. Concretely, a classifier is compiled into an OBDD, which is then compiled into an OBDD representing the AXps of the original classifier. Moreover, [29] proves that, in the context of multi-class classification, if a propositional language satisfies **CD**, **FO**, and **IM**, then one AXp can be computed in polynomial time. Our results in Section 4.4 consider multi-class classification with multiple classifiers. [35] studies the computation complexity of computing smallest AXps. Explanation enumeration based on the MARCO algorithm [231] was investigated in recent work (e.g. [256]). The main difference in Algorithm 7 is the explicit use of transformation and queries from the knowledge compilation map. Perhaps more importantly, the computation of AXps and CXps for a classifier represented as a d-DNNF circuit is fairly orthogonal to earlier work on the computation of explanations for propagators operating on d-DNNF circuits [135]. Indeed, in the case of propagators, the d-DNNF encodes valid assignments to a constraint, and explanations are always computed against a valuation of 1 of the d-DNNF, i.e. the allowed assignments to the constraint.

## Running Examples

Throughout this chapter, we will use the following d-DNNF as our running examples.

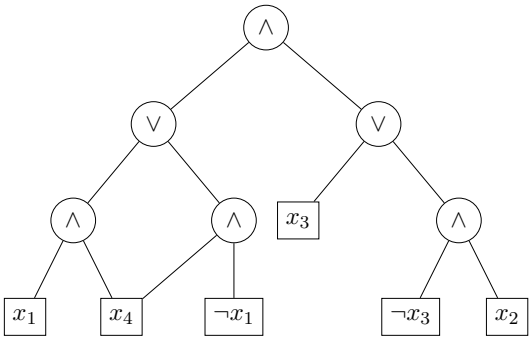



Figure 4.1: Example d-DNNF circuit, $(\mathbf{v}, c) = ((0, 0, 0, 0), 0)$

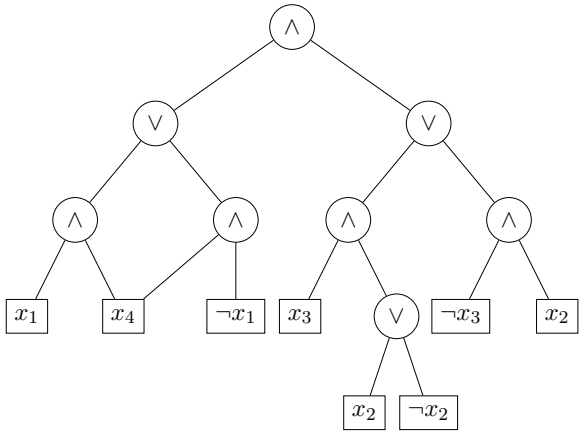



Figure 4.2: The corresponding *smooth* d-DNNF circuit of Figure 4.1

**Example 9.** Let $\mathcal{F} = \{1, 2, 3, 4\}$ and $X = \{x_1, x_2, x_3, x_4\}$. The d-DNNF circuits in Figures 4.1 and 4.3 represent the boolean function $\kappa((x_1, x_2, x_3, x_4)) = ((x_1 \land x_4) \lor (\neg x_1 \land x_4)) \land (x_3 \lor (\neg x_3 \land x_2))$. Moreover, this chapter considers the concrete instance $(\mathbf{v}, c) = ((0, 0, 0, 0), 0)$.

## 4.3 Explanations for d-DNNF

There is a tight connection between the definitions of AXp and CXp (see (2.16) and (2.20)) and the queries **VA**, **CO** and the transformation **CD**. Indeed, for (2.16) and (2.20), **CD** can serve to impose that the values of some features ($i$, represented by variable $x_i$) are fixed to some value $v_i$. In addition, **VA** (resp. **CO**) is used to decide (2.16), after conditioning, when $c = 1$ (resp. $c = 0$). Similarly, **VA** (resp. **CO**) is used to decide (2.20), again after conditioning, when $c = 1$ (resp. $c = 0$). Thus, the predicate $\mathsf{WAXp}(\cdot)$ and $\mathsf{WCXp}(\cdot)$ can be defined as follows:

$$\begin{aligned} \mathsf{WAXp}(\mathcal{S}) &:= [c = \top \land \mathsf{isValid}(\kappa|_{\mathbf{v}_\mathcal{S}})] \lor [c = \bot \land \neg\mathsf{isConsistent}(\kappa|_{\mathbf{v}_\mathcal{S}})] \\ \mathsf{WCXp}(\mathcal{F} \setminus \mathcal{S}) &:= [c = \top \land \neg\mathsf{isValid}(\kappa|_{\mathbf{v}_\mathcal{S}})] \lor [c = \bot \land \mathsf{isConsistent}(\kappa|_{\mathbf{v}_\mathcal{S}})] \end{aligned} \tag{4.1}$$

This means for languages respecting **VA**, **CO** and **CD**, one AXp and one CXp can be computed in polynomial time. This is detailed in the rest of this section.

### 4.3.1 Finding One Explanation

**Finding one AXp.** We start by detailing an algorithm to find one AXp. We identify any $\mathcal{X} \subseteq \{1, \ldots, m\}$ with its corresponding bit-vector $\mathbf{s} = (s_1, \ldots, s_m)$ where $s_i = 1 \Leftrightarrow i \in \mathcal{X}$. Given vectors $\mathbf{x}, \mathbf{v}, \mathbf{s}$, we can construct the vector $\mathbf{x}^{\mathbf{s},\mathbf{v}}$ (in which $\mathbf{s}$ is a selector between the two vectors $\mathbf{x}$ and $\mathbf{v}$) such that

$$x_i^{\mathbf{s},\mathbf{v}} = (x_i \land \overline{s_i}) \lor (v_i \land s_i) \tag{4.2}$$

**Algorithm 5** Finding one AXp given starting seed $\mathcal{X}$

**Input**: Classification function $\kappa$; Seed Set $\mathcal{X} \subseteq \mathcal{F}$; Instance $(\mathbf{v}, c)$
**Output**: AXp $\mathcal{X}$

1: **procedure** findAXp($\kappa, \mathcal{X}, \mathbf{v}, c$)
2: **for all** $i \in \mathcal{X}$ **do**
3: **if** WAXp($\mathcal{X} \setminus \{i\}$) **then**
4: $\mathcal{X} \leftarrow \mathcal{X} \setminus \{i\}$
5: **return** $\mathcal{X}$

To find an AXp, Algorithm 5 is used. For now, seed $\mathcal{X}$ is set to $\mathcal{F}$. Algorithm 5 is a general greedy algorithm that is well-known and used in a wide range of settings, e.g. minimal unsatisfiable core extraction in CSPs [70, 33], but which is also present in the seminal work of Valiant [340]. The novelty is the use of the same algorithm for finding AXps (and also CXps) of propositional languages that respect concrete transformations and queries of the knowledge compilation map. Possible alternatives would include the QuickXplain [205] or the Progression [259] algorithms, among other options [260].)

Considering $\mathbf{s}$ and $\mathbf{v}$ as constants, when $c = 1$, $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$ is valid iff $\mathcal{X}$ is a weak AXp of $\kappa(\mathbf{v}) = c$. Furthermore, when $c = 0$, $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$ is inconsistent iff $\mathcal{X}$ is a weak AXp of $\kappa(\mathbf{v}) = c$. We therefore have the following proposition.

**Proposition 5.** For a classifier implemented with some propositional language $\mathbf{L}$, finding one AXp is polynomial-time provided the following three operations can be performed in polynomial time:

1. construction of $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$ from $\kappa$, $\mathbf{s}$ and $\mathbf{v}$.
2. testing validity of $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$.
3. testing consistency of $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$.

**Corollary 1.** Finding one AXp of a decision taken by a d-DNNF is polynomial-time.

*Proof.* It is sufficient to show that d-DNNFs satisfy **VA**, **CO** and **CD**. To transform a d-DNNF calculating $\kappa(\mathbf{v})$ into a d-DNNF calculating $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$, we need to replace each leaf labelled $x_i$ by a leaf labelled $(x_i \wedge \overline{s_i}) \vee (v_i \wedge s_i)$ and each leaf labelled $\overline{x_i}$ by a leaf labelled $(\overline{x_i} \wedge \overline{s_i}) \vee (\overline{v_i} \wedge s_i)$. Note that $\mathbf{s}$ and $\mathbf{v}$ are constants during this construction. Thus, we simplify these formulas to obtain either a literal or a constant according to the different cases:

- $s_i = 0$: label $(x_i \wedge \overline{s_i}) \vee (v_i \wedge s_i)$ is $x_i$ and label $(\overline{x_i} \wedge \overline{s_i}) \vee (\overline{v_i} \wedge s_i)$ is $\overline{x_i}$. In other words, the label of the leaf node is unchanged.
- $s_i = 1$: label $(x_i \wedge \overline{s_i}) \vee (v_i \wedge s_i)$ is the (constant) value of $v_i$ and label $(\overline{x_i} \wedge \overline{s_i}) \vee (\overline{v_i} \wedge s_i)$ is the (constant) value of $\overline{v_i}$.

Indeed, this is just conditioning, i.e. fixing a subset of the variables $x_i$, given by the set $\mathcal{S}$, to $v_i$. □

Figure 4.3 illustrates the proposed transformation for part of the d-DNNF of Figure 4.1.

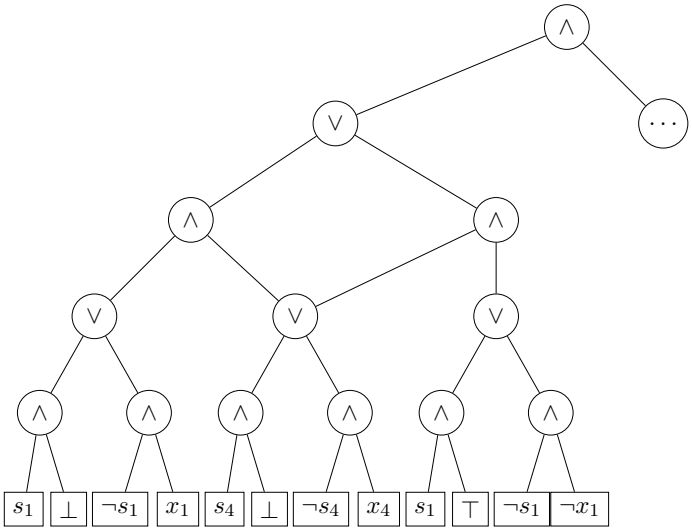



Figure 4.3: Partial Modified d-DNNF for $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$, with $\mathbf{v} = (0,0,0,0)$

| $i$ | $\mathbf{s}$ | $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$ | Justification | Decision |
|---|---|---|---|---|
| 1 | $(0,1,1,1)$ | 0 | $s_4 = 1$: left branch takes value 0, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 0$ | Drop 1 |
| 2 | $(0,0,1,1)$ | 0 | $s_4 = 1$: left branch takes value 0, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 0$ | Drop 2 |
| 3 | $(0,0,0,1)$ | 0 | $s_4 = 1$: left branch takes value 0, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 0$ | Drop 3 |
| 4 | $(0,0,0,0)$ | 1 | Simply set $\mathbf{x} = (1,1,1,1)$, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 1$ | Keep 4 |

Table 4.1: Example of finding one AXp

**Example 10.** The operation of the algorithm for computing one AXp is illustrated for the modified d-DNNF shown in Figure 4.3 for the instance $(\mathbf{v}, c) = ((0,0,0,0), 0)$. By inspection, we can observe that the value computed by the d-DNNF will be 0 as long as $s_4 = 1$, i.e. as long as 4 is part of the weak AXp. If removed from the weak AXp, one can find an assignment to $\mathbf{x}$, which sets $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 1$. The computed AXp is $\mathcal{X} = \{4\}$.

**Finding one CXp.** To compute one CXp, (2.20) is used. In this case, we identify any $\mathcal{Y} \subseteq \{1,\ldots,m\}$ with its corresponding bit-vector $\mathbf{s}$ where $s_i = 1 \Leftrightarrow i \notin \mathcal{Y}$ ($\mathcal{Y} = \mathcal{F} \setminus \mathcal{X}$). Moreover, we adapt the approach used for computing one AXp, as shown in Algorithm 6. As in the case of Algorithm 5, seed $\mathcal{Y}$ is set to $\mathcal{F}$, for now. $\kappa$ is assumed not to be constant, and so a CXp can always be computed.

**Proposition 6.** For a classifier implemented with some propositional language $\mathbf{L}$, finding one CXp is polynomial-time provided the operations of Proposition 5 can be performed in polynomial time.

**Corollary 2.** Finding one AXp/CXp of a decision taken by a classifier is polynomial-

**Algorithm 6** Finding one CXp given starting seed $\mathcal{Y}$

**Input**: Classification function $\kappa$; Seed Set $\mathcal{Y} \subseteq \mathcal{F}$; Instance $(\mathbf{v}, c)$
**Output**: CXp $\mathcal{Y}$

1: **procedure** findCXp($\kappa, \mathcal{Y}, \mathbf{v}, c$)
2: **for all** $i \in \mathcal{Y}$ **do**
3: **if** WCXp($\mathcal{F} \setminus (\mathcal{Y} \setminus \{i\})$) **then**
4: $\mathcal{Y} \leftarrow \mathcal{Y} \setminus \{i\}$
5: **return** $\mathcal{Y}$

| $i$ | $\mathbf{s}$ | $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}})$ | Justification | Decision |
|---|---|---|---|---|
| 1 | $(1,0,0,0)$ | 1 | Pick $\mathbf{x} = (0,1,1,1)$, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 1$ | Drop 1 |
| 2 | $(1,1,0,0)$ | 1 | Pick $\mathbf{x} = (0,0,1,1)$, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 1$ | Drop 2 |
| 3 | $(1,1,1,0)$ | 0 | $s_3 = 1$: right branch takes value 0, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 0$ | Keep 3 |
| 4 | $(1,1,0,1)$ | 0 | $s_4 = 1$: left branch takes value 0, and so $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 0$ | Keep 4 |

Table 4.2: Example of finding one CXp

time if the classifier is given in one of the following languages: d-DNNF [99], SDD [93], OBDD [99], PI [99], IP [99], renH-C [118], AFF [118], dFSD [279], and EADT [216].

*Proof.* It suffices to observe that the languages listed above satisfy the conditions of Propositions Propositions 5 and 6. □

**Example 11.** The operation of the algorithm for computing one CXp is illustrated for the modified d-DNNF shown in Figure 4.3 for the instance $(\mathbf{v}, c) = ((0,0,0,0), 0)$. By inspection, we can observe that the value computed by the d-DNNF can be changed to 1 as long as $s_3 = 0 \land s_4 = 0$, i.e. as long as $\{3,4\}$ are part of the weak CXp. If removed from the weak CXp, one no longer can find an assignment to $\mathbf{x}$ that sets $\kappa(\mathbf{x}^{\mathbf{s},\mathbf{v}}) = 1$. Thus, the computed CXp is $\mathcal{Y} = \{3,4\}$.

### 4.3.2 Enumeration of Explanations

Finally, we outline a MARCO-like algorithm [231] for on-demand enumeration of AXps and CXps. For that, we use Algorithm 5 and Algorithm 6, but now allow for some initial set of features (i.e. a seed) to be specified. The seed is used for computing the next AXp or CXp, and it is picked such that repetition of explanations is disallowed. As argued below, the algorithm's organization ensures that computed explanations are not repeated. Moreover, since the algorithms for computing one AXp or one CXp run in polynomial time, then the enumeration algorithm is guaranteed to require exactly one call to an NP oracle for each computed explanation, in addition to procedures that run in polynomial time.

The main building blocks of the enumeration algorithm are: (1) finding one AXp given a seed (see Algorithm 5); (2) finding one CXp given a seed (see Algorithm 6);

**Algorithm 7** Enumeration algorithm

**Input**: Classification function $\kappa$; Feature Set $\mathcal{F}$; Instance $(\mathbf{v}, c)$

1: **procedure** Enumerate($\mathcal{F}, \kappa, \mathbf{v}, c$)
2: $\mathcal{H} \leftarrow \emptyset$ ▷ $\mathcal{H}$ defined on set $R = \{r_1, \ldots, r_m\}$
3: **repeat**
4: $(\mathsf{outc}, \mathbf{r}) \leftarrow \mathsf{SAT}(\mathcal{H})$
5: **if** outc = **true then**
6: $\mathcal{X} \leftarrow \{i \in \mathcal{F} \,|\, r_i = 1\}$
7: $\mathcal{Y} \leftarrow \{i \in \mathcal{F} \,|\, r_i = 0\}$
8: **if** $\mathsf{WAXp}(\mathcal{X})$ **then**
9: $\mathcal{X} \leftarrow \mathsf{findAXp}(\kappa, \mathcal{X}, \mathbf{v}, c)$
10: $\mathsf{reportAXp}(\mathcal{X})$
11: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{i \in \mathcal{X}} \neg r_i)\}$
12: **else**
13: $\mathcal{Y} \leftarrow \mathsf{findCXp}(\kappa, \mathcal{Y}, \mathbf{v}, c)$
14: $\mathsf{reportCXp}(\mathcal{Y})$
15: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{i \in \mathcal{Y}} r_i)\}$
16: **until** outc = **false**

and (3) a top-level algorithm that ensures that previously computed explanations are not repeated (see Algorithm 7). The top level-algorithm invokes a SAT oracle to identify the seed which will determine whether a fresh AXp or CXp will be computed in the next iteration. As argued earlier, the algorithms for computing one AXp and one CXp use one transformation, specifically conditioning and two queries, namely consistency and validity. In the case of computing one AXp, if the prediction is $\top$, we need to check validity, i.e. for any point in the conditional feature space, the prediction is also $\top$. In contrast, if the prediction is $\bot$, then we need to check that consistency does not hold, i.e. for any point in the conditional feature space, the prediction is also $\bot$. In the case of computing one CXp, we need to change the tests that are executed, since we seek to change the value of the prediction. It should be noted that, by changing the conditioning operation, different propositional languages can be explained. Finally, Algorithm 7 shows the proposed approach for enumerating AXps and CXps, which adapts the basic MARCO algorithm for enumerating minimal unsatisfiable cores [231]. From the definitions, we can see that for any $\mathcal{X} \subseteq \mathcal{F}$, either $\mathcal{X}$ is a weak AXp or $\mathcal{Y} = \mathcal{F} \setminus \mathcal{X}$ is a weak CXp. Every set $\mathcal{X}$ calculated at line 6 of Algorithm 7 has the property that it is not a superset of any previously found AXp and that $\mathcal{Y}$ (calculated at line 7) is not a superset of any previously found CXp.

**Example 12.** Table 4.3 summarizes the main steps of enumerating the AXps and CXps of the running example (see Figure 4.1). It is easy to confirm that after four explanations are computed, $\mathcal{H}$ becomes inconsistent, and so the algorithm terminates. Also, one can confirm the hitting set duality between AXps and CXps [185].

| $\mathcal{H}$ | $\mathsf{SAT}(\mathcal{H})$ | $\mathbf{p}$ | AXp(1), CXp(0)? | $\mathcal{S}$ | AXp | CXp | Block |
|---|---|---|---|---|---|---|---|
| $\emptyset$ | 1 | $(1,1,1,1)$ | 1 | $\{1,2,3,4\}$ | $\{4\}$ | — | $b_1 = (\neg p_4)$ |
| $\{b_1\}$ | 1 | $(1,1,1,0)$ | 1 | $\{1,2,3\}$ | $\{2,3\}$ | — | $b_2 = (\neg p_2 \vee \neg p_3)$ |
| $\{b_1, b_2\}$ | 1 | $(1,0,1,0)$ | 0 | $\{1,3\}$ | — | $\{2,4\}$ | $b_3 = (p_2 \vee p_4)$ |
| $\{b_1, b_2, b_3\}$ | 1 | $(1,1,0,0)$ | 0 | $\{1,2\}$ | — | $\{3,4\}$ | $b_4 = (p_3 \vee p_4)$ |
| $\{b_1, b_2, b_3, b_4\}$ | 0 | — | — | — | — | — | — |

Table 4.3: Example of AXp/CXp enumeration, using Algorithm 7

## 4.4 Generalizations

This section generalizes earlier results by considering multi-class classification, i.e. the set of classes is now $\mathcal{K} = \{c_1, \ldots, c_K\}$.

**Explanations for generalized decision functions.** First, we consider that each class $c_j \in \mathcal{K}$ is associated with a total function $\kappa_j : \mathbb{F} \to \{0,1\}$, such that the class $c_j$ is picked iff $\kappa_j(\mathbf{v}) = 1$. For example, *decision sets* [223] represent one such example of multi-class classification, where each function $\kappa_j$ is represented by a DNF, and a *default rule* is used to pick some class for the points $\mathbf{v}$ in feature space for which all $\kappa_j(\mathbf{v}) = 0$. Moreover, decision sets may exhibit *overlap* [190], i.e. the existence of points $\mathbf{v}$ in feature space such that there exist $j_1 \neq j_2$ and $\kappa_{j_1}(\mathbf{v}) = \kappa_{j_2}(\mathbf{v}) = 1$. In practice, the existence of overlap can be addressed by randomly picking one of the classes for which $\kappa_j(\mathbf{v}) = 1$. Alternatively, DS learning can ensure that overlap is non-existing [190].

Here, we consider generalized versions of DSes, by removing the restriction that each class is computed with a DNF. Hence, a *generalized decision function* (GDF) is such that each function $\kappa_j$ is allowed to be an *arbitrary* boolean function. Furthermore, the following two properties of GDFs are considered:

**Definition 37.** A GDF is *binding* if,

$$\forall(\mathbf{x} \in \mathbb{F}). \bigvee\nolimits_{1 \leq j \leq K} \kappa_j(\mathbf{x}) \tag{4.3}$$

Thus, a binding GDF requires no default rule, since for any point $\mathbf{x}$ in feature space, there is at least one $\kappa_j$ such that $\kappa_j(\mathbf{x})$ holds.

**Definition 38.** A GDF is *non-overlapping* if,

$$\forall(\mathbf{x} \in \mathbb{F}). \bigwedge\nolimits_{1 \leq j_1, j_2 \leq K, j_1 \neq j_2} (\neg\kappa_{j_1}(\mathbf{x}) \vee \neg\kappa_{j_2}(\mathbf{x})) \tag{4.4}$$

Thus, a binding, non-overlapping GDF computes a total multi-class classification function. Furthermore, we can establish conditions for a GDF to be binding and non-overlapping:

**Proposition 7.** A GDF is binding and non-overlapping iff the following formula is inconsistent:

$$\exists(\mathbf{x} \in \mathbb{F}).\kappa_1(\mathbf{x}) + \ldots + \kappa_K(\mathbf{x}) \neq 1 \tag{4.5}$$

*Proof.* Given Definition 37 and Definition 38,

1. Clearly, there exists a point $\mathbf{v} \in \mathbb{F}$ such that $\kappa_1(\mathbf{v}) + \ldots + \kappa_K(\mathbf{v}) = 0$ iff the GDF is non-binding;
2. Clearly, there exists $\mathbf{v} \in \mathbb{F}$ such that $\kappa_1(\mathbf{v}) + \ldots + \kappa_K(\mathbf{v}) \geq 2$ iff the GDF is overlapping.

Thus, the result follows. □

**Remark 1.** For a GDF where each function is represented by a boolean circuit, deciding whether a GDF is binding and non-overlapping is in coNP. In practice, checking whether a GDF is binding and non-overlapping can be decided with a call to an NP oracle.

**Proposition 8.** For a binding and non-overlapping GDF, such that each classification function is represented by a sentence of a propositional language satisfying **CO** and **CD**, then one AXp or one CXp can be computed in polynomial time. Furthermore, enumeration of AXps/CXps can be achieved with one call to an NP oracle per computed explanation.

*Proof.* For computing one AXp of class $c_p$, one can iteratively check consistency on the remaining literals of the other functions $q \neq p$. Conditioning is used to reflect, in the classifiers, the choices made, i.e. which literals are included or not in the AXp. For a CXp a similar approach can be used. For enumeration, we can once again exploit a MARCO-like algorithm. □

**Corollary 3.** For a binding non-overlapping GDF, where each $\kappa_j$ is represented by a DNNF, one AXp and one CXp can be computed in polynomial time. Furthermore, enumeration of AXps/CXps can be achieved with one call to an NP oracle per computed explanation.

Thus, for GDFs that are both binding and non-overlapping, even if each function is represented by the fairly succinct DNNF, one can still compute AXps and CXps efficiently. As clarified by Proposition 8, **VA** is unnecessary to find an AXp/CXp; for any GDF implemented with propositional languages satisfying **CO** and **CD**, an AXp/CXp can be computed in polynomial time. In addition, a MARCO-like algorithm [231] can be used for enumerating AXps and CXps. The results above can be generalized to the case of multi-valued classification, where binarization (one-hot-encoding) can serve for representing multi-valued (non-continuous) features.

## 4.5 Experimental Results

In this section, we present the experiments carried out to assess the practical effectiveness of the proposed approach. The assessment is performed on the computation of AXps and CXps for d-DNNFs and SDDs.

Table 4.4: Listing all AXp's/CXp's for d-DNNFs and SDDs. (Observe that the dataset names with an asterisk, i.e. DATANAME*, represent those that originally contained categorical data, and which we binarized using the one hot encoding (OHE). As a result, for these binarized datasets, computed explanations are defined with respect to the binarized (OHE) features and not with respect to the original features.) Columns $m$ and #TI report, resp. the number of features and the number of tested samples (instances) in the dataset. (The number of tested samples #S represents 10% of the data, selected randomly.) Sub-Column %A reports the (test) accuracy of the model and #ND (resp. #NS) shows the total number of nodes in the compiled d-DNNF (resp. SDD). Column XPs reports the average number of total explanations (AXp's and CXp's). Sub-columns M and avg of column AXp (resp. CXp) show, resp., the maximum and average number of explanations. The average length (in percentage of $m$) of an AXp/CXp is given as %L. Sub-columns M and avg of column d-DNNF (resp. SDD) report, resp., maximum and average runtime (in seconds) to list all XPs of all tested instances. Finally, the average runtimes to compute Anchor explanations for the d-DNNFs is shown in the last column.

| Dataset | $m$ | #TI | Model | | | XPs | AXp | | | CXp | | | d-DNNF | | SDD | | Anchor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | %A | #ND | #NS | avg | M | avg | %L | M | avg | %L | M | avg | M | avg | avg |
| chess* | 38 | 320 | 99 | 88 | 54 | 7 | 9 | 2 | 12 | 12 | 4 | 5 | 0.053 | 0.013 | 0.007 | 0.002 | 3.07 |
| compas* | 46 | 617 | 67 | 300 | 149 | 22 | 34 | 11 | 15 | 25 | 11 | 5 | 0.914 | 0.309 | 0.127 | 0.037 | 5.63 |
| corral | 6 | 16 | 100 | 35 | 12 | 4 | 2 | 1 | 34 | 4 | 2 | 22 | 0.002 | 0.001 | 0.000 | 0.000 | 0.41 |
| kr_vs_kp* | 38 | 320 | 98 | 89 | 60 | 7 | 9 | 2 | 13 | 11 | 5 | 5 | 0.043 | 0.012 | 0.007 | 0.002 | 3.15 |
| Mammographic | 13 | 96 | 79 | 129 | 51 | 13 | 12 | 6 | 41 | 12 | 7 | 16 | 0.083 | 0.040 | 0.004 | 0.002 | 1.38 |
| mofn_3_7_10 | 10 | 132 | 97 | 107 | 47 | 11 | 19 | 4 | 33 | 27 | 6 | 23 | 0.065 | 0.013 | 0.002 | 0.001 | 1.10 |
| monk1* | 15 | 56 | 98 | 93 | 40 | 6 | 4 | 2 | 27 | 9 | 4 | 11 | 0.022 | 0.009 | 0.001 | 0.001 | 3.77 |
| monk2* | 15 | 17 | 64 | 156 | 100 | 15 | 24 | 7 | 45 | 12 | 8 | 12 | 0.161 | 0.058 | 0.016 | 0.006 | 7.93 |
| monk3* | 15 | 55 | 99 | 17 | 9 | 4 | 4 | 2 | 14 | 3 | 2 | 9 | 0.001 | 0.001 | 0.000 | 0.000 | 0.54 |
| mux6 | 6 | 13 | 92 | 58 | 23 | 5 | 4 | 2 | 54 | 5 | 3 | 22 | 0.005 | 0.002 | 0.000 | 0.000 | 0.56 |
| parity5+5 | 10 | 112 | 86 | 427 | 101 | 10 | 8 | 3 | 67 | 15 | 8 | 15 | 0.175 | 0.056 | 0.003 | 0.001 | 18.64 |
| postoperative* | 22 | 9 | 66 | 84 | 55 | 11 | 6 | 4 | 24 | 9 | 7 | 8 | 0.022 | 0.017 | 0.004 | 0.003 | 4.01 |
| primary-tumor* | 17 | 34 | 64 | 117 | 75 | 11 | 15 | 5 | 25 | 11 | 6 | 13 | 0.091 | 0.031 | 0.009 | 0.004 | 2.35 |
| promoters* | 334 | 11 | 81 | 38 | 24 | 5 | 4 | 2 | 1 | 5 | 3 | 1 | 0.005 | 0.003 | 0.001 | 0.001 | 57.72 |
| spect | 22 | 27 | 77 | 114 | 61 | 10 | 14 | 5 | 18 | 12 | 5 | 10 | 0.080 | 0.025 | 0.004 | 0.002 | 1.41 |
| threeOf9 | 9 | 51 | 100 | 87 | 36 | 7 | 12 | 3 | 36 | 9 | 4 | 18 | 0.022 | 0.007 | 0.001 | 0.000 | 1.10 |
| tic_tac_toe* | 27 | 96 | 94 | 179 | 109 | 15 | 33 | 7 | 25 | 17 | 8 | 8 | 0.384 | 0.096 | 0.142 | 0.034 | 7.38 |
| vote* | 48 | 43 | 94 | 38 | 28 | 4 | 5 | 1 | 5 | 6 | 2 | 4 | 0.006 | 0.002 | 0.001 | 0.000 | 1.91 |
| xd6 | 9 | 97 | 99 | 88 | 32 | 7 | 18 | 4 | 34 | 26 | 3 | 20 | 0.031 | 0.009 | 0.003 | 0.001 | 1.51 |

**Experimental setup.** The experiments consider a selection of 19 binary classification datasets that are publicly available and originate from the Penn Machine Learning Benchmarks [282] and the UCI Machine Learning Repository [109]. 8 datasets are fully binary and the remaining 11 datasets comprise categorical/binary features. Then, categorical features are binarized using the well-known one-hot-encoding. We note that the explanations are computed with respect to the new (one-hot-encoded) features, and not with respect to the original features. To learn d-DNNFs (resp. SDDs), we first train Read-Once Decision Tree (RODT) models on the given datasets using Orange3 [104] and then compile the obtained RODTs into d-DNNFs (resp. SDDs). A RODT is a FBDD whose underlying graph is a tree [35, 351]. The compilation of RODTs to d-DNNFs can be easily done by direct mapping, since RODT is a special case of FBDDs, and FBDDs is a subset of d-DNNFs [99]. To compile SDDs, we use the PySDD package[2], which is implemented in Python and Cython. PySDD wraps the well-known SDD package[3] which offers canonical SDDs[4]. Employing canonical SDDs allows performing consistency and validity checking in a constant time (If the canonical SDD is inconsistent (resp. valid) then it is a single node labeled with $\bot$ (resp. $\top$) [93]), so in practice it may improve the efficiency of explaining SDD classifiers. Moreover, all presented algorithms are implemented in Python, in the `Xddnnf` package [5]. In addition, the PySAT toolkit [184] is used to instrument incremental SAT oracle calls to enumerate AXp/CXp. As a baseline comparison, we also include in this evaluation an heuristic explainer Anchor [300] to assess the runtime performance of our approach. Lastly, we run the experiments on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Big Sur.

**Results.** Table 4.4 summarizes the obtained results of explaining d-DNNFs and SDDs. (Note that, for each dataset, the compiled d-DNNF and SDD represent the same decision function of the learned RODT. Hence, the computed explanations are the same as well. The size of the d-DNNF is on average twice as large as the corresponding SDD. Also note that compilation time is not included in the runtimes shown in the table, since these are not directly related with the computation of explanations.) Performance-wise, the maximum runtime to enumerate XPs is less than 1.0 sec for all the d-DNNFs, and less than 0.2 sec for all the SDDs. On average, total enumeration of XPs takes at most 0.4 sec for d-DNNFs; and for SDDs at most 0.05 sec. Thus the overall cost of the SAT oracle calls performed by Algorithm 7 is negligible. Given the results, one can conclude that the SAT calls do not constitute a bottleneck for enumerating the AXps/CXps of the classifiers represented as d-DNNFs or SDDs. However, in settings where the total number of explanations is much larger (i.e. exponentially large on the number of features), the cost of SAT

[2] `https://github.com/wannesm/PySDD`

[3] `http://reasoning.cs.ucla.edu/sdd`

[4] Since PySDD offers canonical SDDs, the **CD** transformation is not implemented in worst-case polynomial time [341]. However, in practice, this was never an issue in our experiments.

[5] `https://github.com/XuanxiangHuang/Xddnnf`

calls could become dominant. Apart from the runtime, one observation is that the total number of AXps and CXps per instance is relatively small. Moreover, if compared with the number of features, the average length of an AXp/CXp is also relatively small. Despite that runtimes reported in Table 4.4 are small for AXp and CXp, one might not argue that the explanation problems studied in this chapter are fairly easy. In fact, as can be observed Anchor's runtimes can exceed the running times of the d-DNNF non-heuristic explainer by several orders of magnitude.

To conclude, for the concrete case of classifiers that can be represented efficiently using d-DNNF and SDD, the experimental results confirm that, if a classifier can be represented with a propositional language that implements **CO** and **VA** as well **CD**, then the computation and enumeration of explanations is not only practical, but substantially more efficient than alternative heuristic approaches. Regarding the limitations of proposed approach, these are the same as for all compilation-based methods: the off-line compilation phase may theoretically be very expensive in time and space. This limitation has not prevented compilation being used in large-scale industrial applications.

## 4.6 Summary

This chapter proves that for any classifier that can be represented with a d-DNNF, both one AXp and one CXp can be computed in polynomial time on the size of the d-DNNF. Furthermore, the chapter shows that enumeration of AXps and CXps can be implemented with one NP oracle call per explanation. The experimental evidence confirms that for small numbers of explanations, the cost of enumeration is negligible. In addition, the chapter proposes conditions for generalized decision functions to be explained in polynomial time. Concretely, the chapter develops conditions which allow generalized decision functions represented with DNNFs to be explainable in polynomial time. Finally, the experimental results validate the scalability of the polynomial time algorithms and, more importantly, the scalability of oracle-based enumeration.

Chapter 5

# From Decision Trees to Explained Decision Sets

Recent work demonstrated that path explanation redundancy is ubiquitous in decision trees, i.e. most often paths in decision trees include literals that are redundant for explaining a prediction. The implication of this result is that decision trees must be explained. Nevertheless, there are applications of decision trees where running an explanation algorithm is impractical. For example, in settings that are time or power constrained, running software algorithms for explaining predictions would be undesirable.

Although the explanations for paths in decision trees do not generally represent themselves a decision tree, this chapter shows that one can construct a decision set from some of the decision tree explanations, such that the decision set is not only explained, but it also exhibits a number of properties that are critical for replacing the original decision tree.

## 5.1 Introduction

Recent years witness groundbreaking advances in machine learning (ML) [41]. However, these advances raise concerns about whether the operation of complex ML models can be understood and trusted by human decision makers. Such concerns are at the core of ongoing efforts on understanding the operation of ML models, e.g. stability of predictions and rationale for predictions. Moreover, the ongoing efforts towards understanding the rationale of predictions broadly represent the burgeoning field of eXplainable Artificial Intelligence (XAI). XAI is characterized by a number of different approaches for tackling the problem of explaining ML models [148]. One important approach is referred to as *intrinsic interpretability* [269], where so-called interpretable models are used, and where the model is itself the explanation.

Decision trees (DTs) have long been deemed interpretable [57], and are at the core of proposals for the use of interpretable models, especially in high-risk applications of ML [304, 305]. Explanations in DTs are apparently very easy to derive, in that the literals in the path consistent with the input represent the explanation. Unfortunately, recent work demonstrated that paths in DTs can be arbitrarily redundant when compared with logically sufficient (abductive) explanations for a prediction [194]. The main consequence of these recent results is that, similarly to other ML models, decision trees must be explained. (It should be noted that this consequence hinges on the assumption that explanation succinctness matters.

However, succinctness *must* always matter when it comes to explainability, since otherwise one could argue that the input to the ML model would suffice as an explanation.) Redundancy has also been observed in other so-called interpretable models, including decision lists [258].

There exist very efficient polynomial-time algorithms for computing abductive explanations in DTs [194]. However, one immediate question is whether one can remove explanation redundancy from paths, so that decision makers have immediate access to the actual explanations. (Also, in some settings, the iterated computation of explanations might be unrealistic, due to constraints on available resources.) Unfortunately, the removal of redundancy breaks the structure of DTs. In addition, it is known that the family of DTs that do not exhibit explanation redundancy is very restricted [194].

Since mapping a DT to an explanation-irredundant DT is unachievable, this chapter proposes a different solution. Concretely, this chapter proposes an algorithm for mapping a DT into a decision set (DS), but such that the resulting DS exhibits a number of critical properties, which ensures it operates as a DT. Since DSs are unordered, they can display a number of fundamental issues. Firstly, DSs can exhibit overlap, and thus may not even compute a classification function. Secondly, for DSs that exhibit no overlap, the classification function may not be total, i.e. for some inputs there is no prediction. In this case, the use of a default rule requires special handing, so that the default rule is only used when no other rule applies. Lastly, DSs require being explained, and explanations for DSs are harder to compute than for DTs [182, 26]. Furthermore, this chapter indirectly proposes a practical solution to the abstract goal of *intrinsic interpretability* [304, 269, 306], where the classifier is itself the explanation. Indeed, the algorithm proposed in this chapter offers a solution to deliver a classifier where the explanation can be extracted by manual inspection from the classifier. The experimental results validate the scalability of the proposed algorithm, and offer comprenhensive evidence to the quality of the obtained DSs, with a key metric being the total number of literals used for explaining the DT paths.

A generalization of (exact) abductive explanations are probabilistic (abductive) explanations [347, 35, 192], which aim at providing decision makers with shorter explanations (which are easier to grasp) and which, albeit not as rigorous, still offer strong probabilistic guarantees of rigor. As an additional contribution, and in the case of probabilistic explanations, this chapter shows that the properties of the DSs obtained from DTs no longer hold. As a result, this chapter outlines a simple, albeit less compact, solution that can be employed in the case of probabilistic explanations.

This chapter is organized as follows. Section 5.2 briefly overviews related work. Section 5.3 introduces the notion of path abductive explanations, which represents a restriction of abductive explanations. Section 5.4 introduces the notion of probabilistic abductive explanations, which serves as a generalization of abductive explanations. Section 5.5 develops the algorithm for mapping a DT into an (explained) DS, proves the key properties of the resulting DS, and investigates the limitations of the algorithm in the case of probabilistic explanations. Section 5.6 presents

experimental results that confirm the properties of DSs obtained from explaining DTs. The results confirm the explained DSs offer significantly more compact (and subset-minimal) explanations than the explanations obtained from the paths in the original DTs. Section 5.7 concludes the chapter.

## 5.2 Related Work

**Decision trees.** As indicated earlier, it is generally assumed that DTs compute total functions, but this may not always be the case [194, page 270]. Without exception, tree induction algorithms guarantee that the resulting DT computes a classification function. However, it is possible to force a greedy tree induction algorithm to generate a DT that does not compute a total function. Nevertheless, deciding whether a DT computes a total function can be formulated as a decision problem, and answered with an automated reasoner. In the rest of the chapter, we assume that such checking has been performed, and so DTs are assumed to compute a total function.

Despite the recent interest in computing explanations for DTs [193, 35, 169, 26, 194, 23], there seems to be no simple way to remove the redundancy from DT paths. Observe that the set of explanations associated with paths in a DT most often does not represent a DT, and attempts at constructing a DT from such explanations would necessarily re-introduce redundancy.

**Decision sets & lists.** Decision sets and lists have been studing since the 1960s [263], with extensive work in the 80s and 90s [264, 74, 77, 78, 79, 129]. The learning DLs and DSs is still an ongoing theme of research [130, 131, 223, 176, 190]. As noted above, DSs exhibit a number of limitations, the most important of which being overlap between rules predicting different classes. There exist solutions which guarantee that overlap is non-existing [190], but the computed classification function is either not total, or require the use of a default rule with a dedicated semantics.

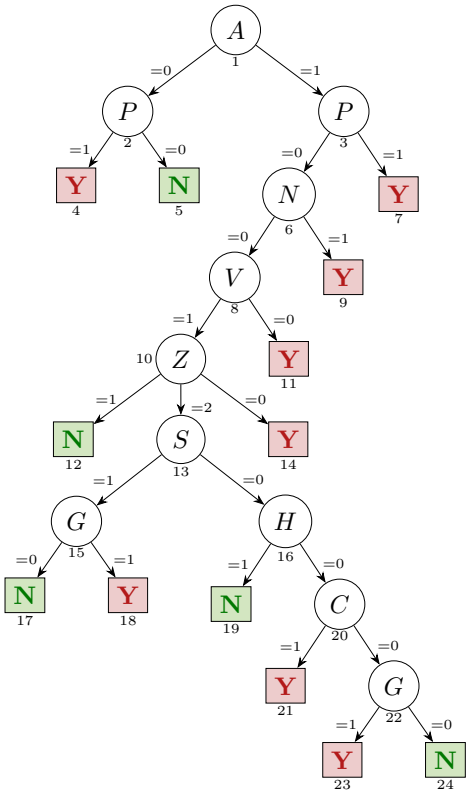
The use of a default rule with a dedicated semantics complicates interpretability or approaches for computing explanations. The alternative solution, i.e. allowing for the classification function not to be total, is also not desirable. To the best of our knowledge, there is no solution for learning a decision set that produces DSs that exhibit no overlap, require no default rule, and which offer explanations by inspection (i.e. guarantee that there is no need for computing explanations).

Despite being considered interpretable models, DTs, DLs and DSs have been shown to require the computation of explanations [182, 194, 258], most often because of explanation redundancy.

## Running examples

Throughout this chapter, we will use the following two running examples.

**Example 13.** The first running example is the DT of Figure 5.1, which is adapted from [229]. The DT serves to diagnose the most severe case of meningitis, Meningococcal Disease (MD), without invasive tests. Clearly, $\mathcal{F} = \{1, \ldots, 9\}$, $\mathcal{K} = \{\mathbf{Y}, \mathbf{N}\}$, $\mathbb{D}_i = \{0, 1\}$ for $i = \{1, 2, 3, 4, 6, 7, 8, 9\}$, and $\mathbb{D}_5 = \{0, 1, 2\}$. (Observe that `Age` is ordinal (integer or real), but we only test whether the value is greater than 5.) Moreover, we will consider the instance $(\mathbf{v}, c) = ((A = 1, P = 0, N = 0, V = 1, Z = 0, S = 0, H = 0, C = 0, G = 1), \mathbf{Y})$. The paths predicting $\mathbf{Y}$ are numbered $\mathcal{P}_1, \ldots, \mathcal{P}_8$. The paths predicting $\mathbf{N}$ are numbered $\mathcal{Q}_1, \ldots, \mathcal{Q}_5$. The paths and their numberings are obtained from an in-order traversal of the tree. For example $\mathcal{P}_1 = \langle 1, 2, 4\rangle$, $\mathcal{Q}_1 = \langle 1, 2, 5\rangle$, $\mathcal{Q}_3 = \langle 1, 3, 6, 8, 10, 13, 15, 17\rangle$, $\mathcal{Q}_5 = \langle 1, 3, 6, 8, 10, 13, 16, 20, 22, 24\rangle$, $\mathcal{P}_5 = \langle 1, 3, 6, 8, 10, 14\rangle$, and $\mathcal{P}_7 = \langle 1, 3, 6, 9\rangle$. For $\mathcal{P}_5$, $\Phi(\mathcal{P}_5) = \{1, 2, 3, 4, 5\}$, where the mapping of features is as shown in Figure 5.1, i.e. feature 1 is $A$, feature 2 is $P$, and so on. In addition, and also for $\mathcal{P}_5$, $\Lambda(\mathcal{P}_5, \{1, 4, 5\}) = \{(A), (\overline{V}), (Z = 0)\}$.



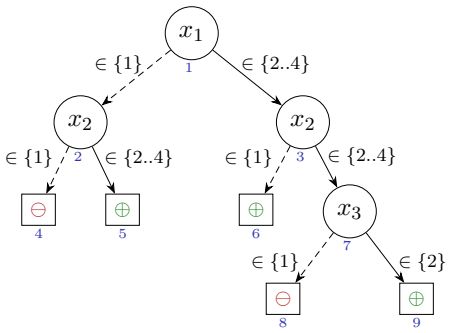

(a) Decision tree

| # | Name | Definition |
|---|---|---|
| 1 | $A$ | Age > 5? |
| 2 | $P$ | Petechiae? |
| 3 | $N$ | Stiff Neck? |
| 4 | $V$ | Vomiting? |
| 5 | $Z$ | Zone=0, 1, 2? |
| 6 | $S$ | Seizures? |
| 7 | $G$ | Gender? |
| 8 | $H$ | Headache? |
| 9 | $C$ | Coma? |

(b) Features' meaning

Figure 5.1: First example DT, adapted from [229]

**Example 14.** The second running example is shown in Figure 5.2. This second running example will be used to illustrate the computation of probabilistic abductive explanations. The figure also shows the truth table for the DT, and for each row in the table, the number of points in feature space consistent with that row.




(a) Decision tree

| $x_1$ | $x_2$ | $x_3$ | $\kappa(\mathbf{x})$ | Path | $\lvert\cdot\rvert$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | $\ominus$ | $\mathcal{Q}_1 = \langle 1, 2, 4\rangle$ | 1 |
| 1 | 1 | 2 | $\ominus$ | $\mathcal{Q}_1$ | 1 |
| 1 | 2..4 | 1 | $\oplus$ | $\mathcal{P}_1 = \langle 1, 2, 5\rangle$ | 3 |
| 1 | 2..4 | 2 | $\oplus$ | $\mathcal{P}_1$ | 3 |
| 2..4 | 1 | 1 | $\oplus$ | $\mathcal{P}_2 = \langle 1, 3, 6\rangle$ | 3 |
| 2..4 | 1 | 2 | $\oplus$ | $\mathcal{P}_2 = \langle 1, 3, 6\rangle$ | 3 |
| 2..4 | 2..4 | 1 | $\ominus$ | $\mathcal{Q}_2 = \langle 1, 3, 7, 8\rangle$ | 9 |
| 2..4 | 2..4 | 2 | $\oplus$ | $\mathcal{P}_3 = \langle 1, 3, 7, 9\rangle$ | 9 |

(b) Truth table

Figure 5.2: Second example DT, adapted from [192]

## 5.3 Path Abductive Explanations

In this chapter we use the restriction of AXp's to the case when features must be taken from the path. Such AXp's are referred to as *path AXp's* [194]. Throughout the chapter, path AXp's are AXp's, but where the features that can be included in the explanation must exist in the path.

**Example 15.** For the running example (see Figure 5.1), and the instance $((A = 1, P = 0, N = 0, V = 1, Z = 0, S = 0, H = 0, C = 0, G = 1), \mathbf{Y})$, this point corresponds to the path $\langle 1, 3, 6, 8, 10, 14\rangle$. We can show that one AXp is $\{A, Z\}$ (technically, we should write $\{1, 5\}$).

To offer a more detailed insight into the process of computing this path AXp, one possible computation is summarized next. While the we will argue that the order of features $\{S, H, C, G\}$ does not matter, the remaining features are analyzed in order $\langle A, P, N, V, Z\rangle$. (Depending on the explanation problem, the order features may or may not matter.)

1. As the features in $\{S, H, C, G\}$ do not appear in the path, we can assign any value to these features. As a result, during the computation of this path's AXp's, these features are not taken into consideration.

2. Let feature $A$ take any possible value from its domain. In this case, we can find a point $(A = 0, P = 0, N = 0, V = 1, Z = 0, S = 0, H = 0, C = 0, G = 1)$ that makes the path $\langle 1, 2, 5\rangle$ consistent, which predicts a different class $\mathbf{N}$. Thus, this violates the definition of path AXp. Hence, feature $A$ must be *fixed* to the value 1.

3. Let feature $P$ take any possible value from its domain. In this case, we cannot find a point in the feature space that makes consistent some path predicting the different class $\mathbf{N}$. As a result, feature $P$ can be declared *free*, allowing it to take any value from its domain.

4. For the same reason, features in $\{N, V\}$ can also be freed, i.e. no path predicting the different class $\mathbf{N}$ can be made consistent when these features are

allowed to take any values from their domains.

5. Finally, let feature $Z$ take any possible value from its domain. In this case, we can find a point $(A = 1, P = 0, N = 0, V = 1, Z = 1, S = 0, H = 0, C = 0, G = 1)$ that makes the path $\langle 1, 3, 6, 8, 10, 12\rangle$ consistent, which predicts the different class $\mathbf{N}$. As before, this violates the definition of path AXp. Hence, feature $Z$ must be fixed to the value 0.

6. In summary, we obtain the path AXp $\{A, Z\}$.

Thus, we can confidently state the following rule, representing a sufficient condition for predicting $\mathbf{Y}$,

$$\text{IF} \quad (\text{Age} > 5) \land (\text{Zone} = 0) \quad \text{THEN} \quad \kappa(\cdot) = \mathbf{Y}$$

Using the more compact notation proposed earlier, we could also write, $A \land Z = 0 \rightarrowtail \mathbf{Y}$. The use of explanations allows identifying possible model learning issues with the example DT; this is further discussed elsewhere [253].

## 5.4 Probabilistic Abductive Explanations

Building on earlier work [347, 23, 192], we define *weak probabilistic* AXp (or weak PAXp) $\mathcal{X} \subseteq \mathcal{F}$ as a set of fixed features for which the probability of predicting the correct class $c$, for points consistent with the values of $\mathcal{X}$ in $\mathbf{v}$, exceeds $\delta > 0$, with $c = \kappa(\mathbf{v})$. Thus, $\mathcal{X} \subseteq \mathcal{F}$ is a weak PAXp if the following predicate holds true,

$$\begin{aligned} &\mathsf{WeakPAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c, \delta) \\ &\quad := \Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}}) \geq \delta \\ &\quad := \frac{|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \land (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|}{|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|} \geq \delta \end{aligned} \tag{5.1}$$

(Where the restriction of $\mathbf{x}$ to the variables with indices in $\mathcal{X}$ is represented by $\mathbf{x}_{\mathcal{X}} = (x_i)_{i \in \mathcal{X}}$. Concretely, the notation $\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}}$ represents the constraint $\land_{i \in \mathcal{X}} x_i = v_i$.) The condition above means that the fraction of the number of points predicting the target class and consistent with the fixed features (represented by $\mathcal{X}$), given the total number of points in feature space consistent with the fixed features, must exceed $\delta$. We can adapt (2.16) to define a PAXp given the definition of WPAXp. Since the definition of weak PAXp (see (5.1)) is non-monotonic, then the computation of PAXp's cannot be simplified [192] using (2.17).

For DTs with categorical features, and for each pick of fixed features, one can compute the conditional probability in polynomial-time [192]. For the purposes of this chapter, we will use the truth table of Figure 5.2.

**Example 16.** For the DT of Figure 5.2, let us consider the instance $(\mathbf{v}, c) = ((1, 1, 1), \ominus)$ and $\mathcal{X} = \{1, 2\}$. Then, the number of points where $x_1 = 1 \land x_2 = 1$ is 2. Moreover, for all those points, $\kappa(\cdot) = \ominus$. Thus, $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \ominus \,|\, \mathbf{x}_{\{1,2\}} = \mathbf{v}_{\{1,2\}}) = 1$.

Moreover, when $(\mathbf{v}, c) = ((2,2,2), \oplus)$ and $\mathcal{X} = \{3\}$, the number of points where $x_3 = 2$ is 16 and, among these, the number of points for which $\kappa(\cdot) = \oplus$ is 15. Thus, $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \oplus \,|\, \mathbf{x}_{\{3\}} = \mathbf{v}_{\{3\}}) = {}^{15}/_{16}$. Finally, with $\delta = 0.9$, it is the case that $\{1,2\}$ is a WPAXp for (any instance of) $\mathcal{Q}_1$, and $\{3\}$ is a WPAXp for (any instance of) $\mathcal{P}_3$. It is simple to show that both WPAXp's are PAXp's [192].

## 5.5 From Decision Trees to (Explained) Decision Sets

To the best of our knowledge, there is no simple way to remove redundancy from a DT such that some DT can be reconstructed. As a result, one solution is to consider removing redundancy from a DT such that a different ML model is obtained. However, one key requirement for such ML model is that it must allow for explanations to be easily extracted, i.e. no algorithm is to be executed. The next section shows one basic approach to obtain such an ML model. Afterwards, we discuss extensions to the basic approach, their limitations, and alternative solutions.

### 5.5.1 Mapping a DT into a DS

This section develops an algorithm which, given a DT computing a total classification function, creates a DS with the following key properties:

1. The DS does not include a default rule;
2. The DS does not exhibit overlap (i.e. it computes a classification function);
3. The DS computes a total classification function; and
4. Each rule is a path AXp of the original DT.

Given the above properties, the DSs obtained with the algorithm described below will be referred to as *explained decision sets*.

The above properties are critically important, since the created DS does not exhibit *any* of the issues that are problematic for existing implementations of DSs. Furthermore, for any point in feature space, if some rule $\mathcal{R}_j$ fires, then the condition of the rule represents a path AXp of the original DT, i.e. there is *no* need for computing AXp's.

Algorithm 8 represents the proposed solution for constructing a DS starting from a DT. As can be observed, for each path in the DT, the algorithm computes one AXp. This AXp is then used for constructing a decision rule, using the literals obtained from the literals included in the AXp. In the end, duplicate rules are removed. The algorithm used for computing one path AXp, i.e. FindPathAXp, can be any of the algorithms proposed in earlier work [193, 194].

**Example 17.** Table 5.1 summarizes the execution of Algorithm 8 on the example DT of Figure 5.1. Each row lists: (a) the list of path nodes; (b) the features in the explanation when the path represents the explanation; (c) the condition of the

**Algorithm 8** Converting DT to DS

**Input**: Decision Tree $\mathcal{M}$ with classification function $\kappa$

**function** DT2DS($\mathcal{M}$)
  $\mathcal{S} \leftarrow \emptyset$ ▷ $\mathcal{S}$: DS to be constructed
  $\mathbb{P} \leftarrow \mathsf{AllPaths}(\mathcal{M})$ ▷ $\mathbb{P}$: set of all paths in $\mathcal{M}$
  **while** $\mathbb{P} \neq \emptyset$ **do**
    $\mathcal{P}_k \leftarrow \mathsf{PickPath}(\mathbb{P})$ ▷ $\mathcal{P}_k$: some path not yet explained
    $\mathbb{P} \leftarrow \mathbb{P} \setminus \{\mathcal{P}_k\}$
    $\mathcal{X} \leftarrow \mathsf{FindPathAXp}(\mathcal{P}_k)$ ▷ E.g. algorithms from [194]
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{IF} \quad \wedge_{l \in \Lambda(\mathcal{P}_k, \mathcal{X})} l \quad \text{THEN} \quad \kappa(\cdot) = c\}$
  $\mathcal{S} \leftarrow \mathsf{RemoveDuplicateRules}(\mathcal{S})$
  **return** $\mathcal{S}$

$$
\begin{aligned}
&\mathcal{R}_{01}\text{: IF } [P] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathcal{R}_{02}\text{: IF } [\overline{A} \land \overline{P}] \text{THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathcal{R}_{03}\text{: IF } [\overline{P} \land \overline{N} \land V \land Z = 1] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathcal{R}_{04}\text{: IF } [\overline{P} \land \overline{N} \land V \land Z = 2 \land S \land \overline{G}] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathcal{R}_{05}\text{: IF } [A \land Z = 2 \land S \land G] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathcal{R}_{06}\text{: IF } [\overline{P} \land \overline{N} \land V \land Z = 2 \land \overline{S} \land H] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathcal{R}_{07}\text{: IF } [A \land Z = 2 \land \overline{S} \land \overline{H} \land C] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathcal{R}_{08}\text{: IF } [A \land Z = 2 \land \overline{H} \land G] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathcal{R}_{09}\text{: IF } [\overline{P} \land \overline{N} \land V \land Z = 2 \land \overline{C} \land \overline{G}] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathcal{R}_{10}\text{: IF } [A \land Z = 0] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathcal{R}_{11}\text{: IF } [A \land \overline{V}] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathcal{R}_{12}\text{: IF } [A \land N] \text{ THEN } \kappa(\cdot) = \mathbf{Y}
\end{aligned}
\tag{5.2}
$$

rule that would be obtained in such a situation; (d) the features in some path AXp; and (e) the condition of the rule obtained from the features in the AXp. In total, the DT has 13 paths, and so Table 5.1 summarizes the algorithm's execution for each of the 13 paths. As can be observed, the last row creates a rule which is a duplicate, and so it will not be added to the DS. As a result, the DS consists of 12 rules (i.e. obtained from the first 12 executions of the algorithm's main loop). These are shown in (5.2) below. The order of the rules can be any, since the result is a decision set. The order shown in (5.2) is taken from the order in which paths are analyzed in Table 5.1, with duplicate rules removed.

As will be demonstrated in the next section, each rule is itself an abductive explanation (of the original DT), the DS computes a function (i.e. it exhibits *no* overlap), and the computed function is total. More importantly, as an be observed in Table 5.1, while the path literals for the DT total 75 literals, the explanations obtained from the DS total 44 literals, representing a reduction of more than 40% on the total number of literals used in explanations.

**Extensions.** The proposed algorithm leaves some flexibility on how to compute each AXp. One solution is to compute one subset-minimal AXp, since there are polynomial-time algorithms in the case of DTs [193, 169, 194]. Alternatively, one can consider computing cardinality-minimal AXp's, thus obtaining shortest explanations for each path. It is well-known that computing one cardinality-minimal AXp is NP-hard [35], but with a decision problem in NP. Thus, given proposed Horn encodings [194], a cardinality-minimal AXp can be computed by solving Horn Maximum Satisfiability (MaxSAT).

There has been recent work on inferring and exploiting constraints on the inputs when computing abductive explanations [145, 373]. It is also immediate to account for constraints on the inputs when computing explanations. Thus, Algorithm 8 can be used to produce an explained DS that takes input constraints into account.

Finally, we should observe that for each path in the DT there can be more than one path AXp. Indeed, in the worst-case, the number of path AXp's can be exponential on the number of features. As a result, the proposed algorithm can be adapted to allow for the (restricted) navigation of the space of AXp's for each path, thus enabling a human decision maker to select which AXp to associate with each path. It should be noted that in the case of the DT from Figure 5.1, and given the AXp's computed in Table 5.1, none of the paths exhibits more than one AXp.

**Algorithm's complexity.** The complexity of Algorithm 8 is linear on the complexity of computing abductive explanations. For plain subset-minimal AXp's, Algorithm 8 runs in polynomial-time, since there exist polynomial-time algorithms for computing one AXp [193, 194].

When the DS is to be constructed from cardinality-minimal AXp's, Algorithm 8 computes one such explanation a number of times that is linear with the nodes in the DT. However, computing one smallest AXp in the case of DTs is NP-hard [35]. Moreover, the algorithms for computing cardinality-minimal AXp's will require at most a logarithmic number of calls to an NP oracle in the worst-case. In the case of cardinality-minimal AXp's for DTs, a DS is obtained solving Horn MaxSAT a number of times that grows with the number of (terminal) nodes in the DT. Finally, navigation of the space of AXp's will also impact complexity, depending on how many AXp's are to be enumerated.

### 5.5.2 Properties of Explained Decision Sets

As suggested in earlier sections, we now prove the key properties of the DS created with Algorithm 8. First, we prove that each rule condition in the DS maps to a path AXp of the original DT. Then, we prove that Algorithm 8 creates a DS that computes a (classification) function, i.e. there is no overlap between rules with different predictions. Finally, we prove that the DS computes a total function, i.e. for any point in feature space, at least one rule fires.

**Proposition 9.** The conditions of each rule in the DS represent a path AXp of the original DT.

*Proof.* This result is an immediate consequence of Algorithm 8, since each rule in the DS is obtained from a path AXp of the original DT. □

As a consequence of Proposition 9, each rule corresponds to some path(s) of the original DT. Furthermore, and under the hypothesis that the original DT computes a total function, we can prove the following results. The first result ensures that the DS computes a function (i.e. there is no overlap). The second results ensures that the computed function is total (i.e. there is a prediction for every point in feature space).

**Proposition 10.** The constructed DS is non-overlapping.

*Proof.* Suppose the constructed DS is overlapping, which means there exist at least two rules with non-contradicting *conditions* that predict two *different classes*. In such a case, there would exist a point in the feature space that is consistent with two paths of the original DT leading to two different classes, which contradicts with the hypothesis that the DT computes a total function. Hence, DS is non-overlapping. □

**Proposition 11.** The constructed DS is total.

*Proof.* If the constructed DS is not total, it implies the existence of a point in the feature space that is not consistent with any rule in the DS. As a result, this point is also not consistent with any path of the original DT, which contradicts the hypothesis that the original DT is total. Thus, the constructed DS is total. □

### 5.5.3 Limitations & Solutions

The basic algorithm proposed in the previous section (see Algorithm 8) allows mapping DTs to (explained) DSs when an path AXp is associated with each path. This section investigates limitations of the proposed algorithm and outlines possible solutions.

**Probabilistic explanations.** There has been recent work on computing rigorous probabilistic explanations [347, 196, 197, 199, 23, 192]. Similarly to computing AXp's, Algorithm 8 could be instrumented to compute probabilistic AXp's (PAXp's) or locally minimal PAXp's (LmPAXp's) [192]. Thus, a DS would be constructed using different notions of probabilistic explanations instead of plain abductive explanations. Unfortunately, in this case the resulting DS would not respect the properties established in Section 5.5.2, in that overlap is no longer guaranteed not to exist. The following example illustrates the issue of overlap that PAXp's can induce.

**Example 18.** We use the example DT in Figure 5.2, and the WPAXp's studied in Example 16 to convey the issues raised by probabilistic explanations.

Let $\delta = 0.9$. For the path $\langle 1, 2, 4\rangle$, the (only) PAXp is $\{1, 2\}$, which would yield the rule $x_1 \in \{1\} \land x_2 \in \{1\} \rightarrowtail \ominus$. Moreover, for the path $\langle 1, 3, 7, 9\rangle$ a PAXp is

$\{3\}$, which would yield the rule $x_3 \in \{2\} \rightarrowtail \oplus$. It is plain to conclude that there exists overlap between the two rules.

**DT annotation.** In the case of probabilistic explanations, there is a simple, but less compact, approach to pre-compute the explanations of each path. The solution is to annotate the terminal nodes of DTs with the computed explanations. In the case of probabilistic explanations, it suffices to compute a (Lm)PAXp for each path, and then annotate the terminal node of the path with that explanation. It is plain that the DT guarantees the non-existence of overlap. Moreover, annotating the terminal nodes will have no effect on whether the DT computes a total function.

There are downsides to this solution, which Algorithm 8 addresses. First, DT annotation yields a less compact representation of both the ML model and a possible universe of explanations. Second, a human decision maker will be expected to be able to relate computed explanations with the paths the explanations are associated with.

## 5.6 Experiments

This section presents experimental results that evaluate the practical efficiency of the proposed approach for mapping a Decision Tree into a Decision Set. It is important to emphasize that the experiment did not consider computing cardinality-minimal AXp's for extracting rules and constructing DS.

**Experimental setup.** The evaluation comprises 44 datasets that are originate from Penn ML Benchmarks [282]. The datasets used in the chapter consist of features with either categorical or ordinal domains (i.e. integer or real-valued). The number of features ranges from 6 to 240, while the number of classes varies from 2 to 26, with an average of 47 features and 5 classes. Each dataset is divided into training and testing sets, with 80% of the data used for training and 20% used for testing. DTs are learned using Orange3 [104], the maximum depth set to 9 while the minimal test accuracy set to 70%. It is worth noting that Orange3 is capable of handling features with categorical or ordinal domains. Furthermore, a prototype of the proposed algorithm was implemented in Python and is publicly available in the repository [1] Finally, the experiments were performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Monterey.

**Results.** Table 5.2 summarizes the results of mapping DTs to DS. For the learned DTs, 34 out of the 44 DTs have depth more than 7, 33 out of the 44 DTs achieve a test accuracy of over 80%. Moreover, the number of nodes in the learned DTs ranges from 13 to 689, with 22 out of 44 models having more than 100 nodes. Besides, the number of tree paths for DTs varies from 7 to 345, with an average of 74 paths.

[1] https://github.com/XuanxiangHuang/dtree2dset

The total number of literals in tree paths varies from 26 to 3021, with an average of 566 literals.

Through the transformation of DTs into DSes, we have successfully reduced the number of rules and literals required for decision-making. The number of rules for the DSes ranged from 6 to 259, with an average of 66 rules. Besides, the total number of literals in the rules varied from 12 to 1915, with an average of 423 literals. On average, roughly 10% tree paths are redundant, and roughly 25% literals are redundant.

More specifically, for the *adult*, *car_evaluation*, *coil2000*, *connect_4*, and *corral* datasets, we observe that 43.4%, 22%, 25.4%, 24.9%, and 57.1%, respectively, of the tree paths are redundant. Although the tree paths for the *cancer* [2], *promoters* [3], *sonar*, *spectf*, and *wdbc* datasets are not redundant, a non-negligible ratio of redundant literals exists within the tree paths. Specifically, for these datasets, the maximal ratio of redundant literals in the tree paths is 50%, 50%, 42.9%, 66.7%, and 50%, respectively, while the average ratios of redundant literals in the tree paths are 26%, 16.7%, 19.4%, 41.1%, and 25.2%, respectively. An additional observation is that in 34 out of the 44 DTs, the maximal ratio of redundant literals in the tree paths is at least 40%, and in some cases, can exceed 70% (e.g., datasets *ionosphere* and *ring*). Additionally, in 24 out of 44 DTs, the average ratio of redundant literals in the tree paths is at least 20%. However, there are indeed some DTs where the average ratio of redundant literals in the tree paths is small, such as *authorship* [4] and *dermatology*.

Finally, the table shows that the runtime for mapping DTs into DSes is negligible, as indicated in the last column. This can be attributed to the algorithm used, which leverages a polynomial-time method for extracting one path AXp from each tree path.

## 5.7 Summary

This chapter demonstrates that (non-explained) decision trees can be mapped onto (explained) decision sets, such that the obtained decision sets exhibit all the key properties of decision trees. The chapter proposes an algorithm that constructs an explained decision set starting from a decision tree. In contrast with other algorithms for constructing decision sets proposed in the recent past [223, 252, 190, 140, 138, 370, 139, 371, 181], the algorithm proposed in this chapter ensures that the resulting decision sets compute a total function, such that the condition of each rule is the explanation for the prediction when the rule fires. Given the existing proposals for intrinsic interpretability [304, 269, 306], the algorithm proposed in this chapter offers a solution to deliver a classifier where the explanation is extracted, by inspection, from the classifier. The experiments demonstrate not only the scalability

[2] breast_cancer_wisconsin
[3] molecular_biology_promoters
[4] analcatdata_authorship

of the proposed algorithm, but also the significantly tighter representations that explainable decision sets achieve.

Future work will investigate the impact on the size of the explained DT of computing smallest explanations for each path in the DT. Similarly, additional heuristics for selecting the explanations to consider for each path in the DT will be investigated. For example, one could give preference to picking explanations that match already picked explanations, so as to minimize the total number of rules in the DS.

Table 5.1: Summary of the execution of Algorithm 8 for the DT in Figure 5.1. The rule extracted from each AXp is added to the DS, with duplicates removed.

| Path | Path Xp | Rule condition from path | Path AXp | Rule condition from path AXp |
|---|---|---|---|---|
| $\langle 1,2,4\rangle$ | $\{1,2\}$ | $\overline{A}\wedge P$ | $\{2\}$ | $P$ |
| $\langle 1,2,5\rangle$ | $\{1,2\}$ | $\overline{A}\wedge\overline{P}$ | $\{1,2\}$ | $\overline{A}\wedge\overline{P}$ |
| $\langle 1,3,6,8,10,12\rangle$ | $\{1,2,3,4,5\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=1$ | $\{2,3,4,5\}$ | $\overline{P}\wedge\overline{N}\wedge V\wedge Z=1$ |
| $\langle 1,3,6,8,10,13,15,17\rangle$ | $\{1,2,3,4,5,6,7\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge S\wedge\overline{G}$ | $\{2,3,4,5,6,7\}$ | $\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge S\wedge\overline{G}$ |
| $\langle 1,3,6,8,10,13,15,18\rangle$ | $\{1,2,3,4,5,6,7\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge S\wedge G$ | $\{1,5,6,7\}$ | $A\wedge Z=2\wedge S\wedge G$ |
| $\langle 1,3,6,8,10,13,16,19\rangle$ | $\{1,2,3,4,5,6,8\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge\overline{S}\wedge H$ | $\{2,3,4,5,6,8\}$ | $\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge\overline{S}\wedge H$ |
| $\langle 1,3,6,8,10,13,16,20,21\rangle$ | $\{1,2,3,4,5,6,8,9\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge\overline{S}\wedge\overline{H}\wedge C$ | $\{1,5,6,8,9\}$ | $A\wedge Z=2\wedge\overline{S}\wedge\overline{H}\wedge C$ |
| $\langle 1,3,6,8,10,13,16,20,22,23\rangle$ | $\{1,2,3,4,5,6,7,8,9\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge\overline{S}\wedge\overline{H}\wedge\overline{C}\wedge G$ | $\{1,5,7,8\}$ | $A\wedge Z=2\wedge\overline{H}\wedge G$ |
| $\langle 1,3,6,8,10,13,16,20,22,24\rangle$ | $\{1,2,3,4,5,6,7,8,9\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge\overline{S}\wedge\overline{H}\wedge\overline{C}\wedge\overline{G}$ | $\{2,3,4,5,7,9\}$ | $\overline{P}\wedge\overline{N}\wedge V\wedge Z=2\wedge\overline{C}\wedge\overline{G}$ |
| $\langle 1,3,6,8,10,14\rangle$ | $\{1,2,3,4,5\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge V\wedge Z=0$ | $\{1,5\}$ | $A\wedge Z=0$ |
| $\langle 1,3,6,8,11\rangle$ | $\{1,2,3,4\}$ | $A\wedge\overline{P}\wedge\overline{N}\wedge\overline{V}$ | $\{1,4\}$ | $A\wedge\overline{V}$ |
| $\langle 1,3,6,9\rangle$ | $\{1,2,3\}$ | $A\wedge\overline{P}\wedge N$ | $\{1,3\}$ | $A\wedge N$ |
| $\langle 1,3,7\rangle$ | $\{1,2\}$ | $A\wedge P$ | $\{2\}$ | $P$ |

Table 5.2: The table shows statistics for datasets, decision trees and the resulting decision sets. The table includes the number of features ($m$) and the number of classes ($K$) for each dataset. For each DT, the table reports the tree depth (D), the number of nodes (#N), test accuracy (A%), the number of paths ($|\mathbb{P}|$), and $|\Lambda(\mathbb{P})|$, which is the total number of literals in all tree paths. For the resulting DS, the table reports the number of rules ($|\mathcal{S}|$) and $|\Lambda(\mathcal{S})|$, which is the total number of literals in all rules. Moreover, the table reports the maximum (max R%) and average (avg. R%) path redundancy ratio, which refers to the proportion of literals that can be removed from a decision path to convert it into a decision rule over the total number of literals included in that path. The last column reports the runtime (in seconds) for converting DT into DS.

| Dataset | $m$ | $K$ | DT | | | | | DS | | | | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | D | #N | A% | $\|\mathbb{P}\|$ | $\|\Lambda(\mathbb{P})\|$ | $\|\mathcal{S}\|$ | $\|\Lambda(\mathcal{S})\|$ | max R% | avg. R% | |
| adult | 14 | 2 | 9 | 151 | 86.0 | 76 | 639 | 43 | 212 | 62.5 | 37.8 | 0.21 |
| authorship | 70 | 4 | 5 | 25 | 95.3 | 13 | 51 | 13 | 49 | 20.0 | 3.1 | 0.02 |
| ann_thyroid | 21 | 3 | 8 | 25 | 99.7 | 13 | 66 | 11 | 33 | 60.0 | 33.2 | 0.02 |
| cancer | 30 | 2 | 6 | 13 | 92.1 | 7 | 27 | 7 | 18 | 50.0 | 26.0 | 0.01 |
| car_evaluation | 21 | 4 | 9 | 117 | 96.2 | 59 | 478 | 46 | 299 | 50.0 | 17.8 | 0.13 |
| chess | 36 | 2 | 9 | 43 | 99.4 | 22 | 155 | 21 | 94 | 66.7 | 32.5 | 0.05 |
| churn | 20 | 2 | 9 | 93 | 92.8 | 47 | 303 | 43 | 178 | 60.0 | 34.8 | 0.09 |
| coil2000 | 85 | 2 | 9 | 117 | 93.9 | 59 | 467 | 44 | 254 | 66.7 | 23.9 | 0.18 |
| connect_4 | 42 | 3 | 9 | 689 | 72.4 | 345 | 3021 | 259 | 1915 | 44.4 | 15.0 | 1.55 |
| corral | 6 | 2 | 5 | 27 | 100.0 | 14 | 56 | 6 | 12 | 60.0 | 46.1 | 0.01 |
| dermatology | 34 | 6 | 7 | 17 | 95.9 | 9 | 42 | 9 | 40 | 14.3 | 3.2 | 0.02 |
| dna | 180 | 3 | 9 | 149 | 92.5 | 75 | 563 | 73 | 468 | 50.0 | 14.9 | 0.22 |
| ionosphere | 34 | 2 | 7 | 29 | 93.0 | 15 | 73 | 13 | 38 | 71.4 | 36.3 | 0.03 |
| kr_vs_kp | 36 | 2 | 9 | 39 | 98.9 | 20 | 135 | 20 | 89 | 66.7 | 31.0 | 0.05 |
| letter | 16 | 26 | 9 | 499 | 73.9 | 250 | 2122 | 229 | 1514 | 50.0 | 21.6 | 0.87 |
| mfeat_factors | 216 | 10 | 9 | 155 | 84.8 | 78 | 579 | 78 | 544 | 25.0 | 5.5 | 0.21 |
| mfeat_fourier | 76 | 10 | 9 | 217 | 75.0 | 109 | 826 | 107 | 692 | 44.4 | 13.5 | 0.30 |
| mfeat_karhunen | 64 | 10 | 9 | 253 | 77.2 | 127 | 980 | 125 | 816 | 42.9 | 14.8 | 0.34 |
| mfeat_pixel | 240 | 10 | 9 | 185 | 87.0 | 93 | 658 | 93 | 637 | 22.2 | 2.9 | 0.26 |
| mfeat_zernike | 47 | 10 | 9 | 293 | 74.2 | 147 | 1187 | 145 | 1076 | 33.3 | 7.7 | 0.46 |
| mofn_3_7_10 | 10 | 2 | 7 | 93 | 97.4 | 47 | 294 | 46 | 174 | 57.1 | 38.9 | 0.17 |
| promoters | 57 | 2 | 4 | 15 | 72.7 | 8 | 26 | 8 | 21 | 50.0 | 16.7 | 0.01 |
| movement_libras | 90 | 15 | 9 | 127 | 73.6 | 64 | 414 | 64 | 401 | 28.6 | 2.9 | 0.16 |
| mux6 | 6 | 2 | 6 | 55 | 100.0 | 28 | 141 | 15 | 46 | 50.0 | 34.7 | 0.04 |
| optdigits | 64 | 10 | 9 | 353 | 89.3 | 177 | 1433 | 177 | 1367 | 25.0 | 4.3 | 0.61 |
| pendigits | 16 | 10 | 9 | 235 | 94.0 | 118 | 903 | 115 | 776 | 44.4 | 10.6 | 0.33 |
| ring | 20 | 2 | 9 | 107 | 83.6 | 54 | 394 | 48 | 219 | 77.8 | 37.1 | 0.32 |
| satimage | 36 | 6 | 9 | 333 | 86.2 | 167 | 1351 | 154 | 980 | 66.7 | 20.9 | 0.47 |
| sonar | 60 | 2 | 7 | 27 | 78.6 | 14 | 68 | 14 | 52 | 42.9 | 19.4 | 0.02 |
| soybean | 35 | 18 | 9 | 79 | 84.4 | 40 | 260 | 38 | 212 | 33.3 | 10.6 | 0.07 |
| spambase | 57 | 2 | 9 | 141 | 92.0 | 71 | 509 | 68 | 379 | 62.5 | 20.1 | 0.19 |
| spect | 22 | 2 | 9 | 77 | 79.6 | 39 | 268 | 34 | 170 | 66.7 | 23.2 | 0.09 |
| spectf | 44 | 2 | 9 | 29 | 85.7 | 15 | 91 | 15 | 53 | 66.7 | 41.1 | 0.04 |
| splice | 60 | 3 | 9 | 91 | 89.2 | 46 | 319 | 46 | 257 | 50.0 | 20.1 | 0.12 |
| texture | 40 | 11 | 9 | 167 | 90.4 | 84 | 660 | 81 | 552 | 33.3 | 12.3 | 0.24 |
| threeOf9 | 9 | 2 | 8 | 63 | 100.0 | 32 | 185 | 23 | 75 | 57.1 | 34.6 | 0.06 |
| tic_tac_toe | 9 | 2 | 9 | 113 | 91.7 | 57 | 389 | 48 | 240 | 55.6 | 23.8 | 0.14 |
| tokyo1 | 44 | 2 | 9 | 27 | 94.3 | 14 | 91 | 12 | 46 | 44.4 | 30.9 | 0.03 |
| twonorm | 20 | 2 | 9 | 351 | 83.8 | 176 | 1409 | 171 | 1232 | 44.4 | 9.3 | 0.69 |
| vote | 16 | 2 | 8 | 19 | 93.1 | 10 | 50 | 10 | 35 | 40.0 | 25.0 | 0.02 |
| waveform_21 | 21 | 3 | 9 | 343 | 75.6 | 172 | 1375 | 154 | 1026 | 44.4 | 15.5 | 0.56 |
| waveform_40 | 40 | 3 | 9 | 391 | 75.0 | 196 | 1596 | 180 | 1244 | 44.4 | 14.3 | 0.70 |
| wdbc | 30 | 2 | 6 | 17 | 89.5 | 9 | 40 | 9 | 28 | 50.0 | 25.2 | 0.01 |
| xd6 | 9 | 2 | 9 | 79 | 100.0 | 40 | 243 | 30 | 90 | 66.7 | 44.2 | 0.10 |

Chapter 6

# Feature Necessity and Relevancy in Formal Explanations

In previous chapters, we studied the computation of formal explanations for a wide range of classifiers. However, there are several additional explainability queries that are of interest. Two concrete examples are feature necessity and relevancy. Feature necessity asks whether a feature must occur in all explanations of a given prediction. In contrast, feature relevancy asks whether a feature occurs in some explanation of a given prediction.

This chapter investigates both the computational complexity of these problems, but also algorithms for their solution in practice. In terms of algorithms for feature relevancy, this chapter studies algorithms for specific families of classifiers, but also general-purpose algorithms, which can be applied to families of classifiers used in most systems of AI and ML. The experimental results confirm that feature membership can be efficiently decided in practice, for a wide range of families of classifiers.

## 6.1 Introduction

The advances in ML over the years, and the fact that ML models are most often opaque, sparked the ongoing efforts on eXplainable artificial intelligence (XAI). Furthermore, the existing and expected uses of ML in high-risk applications of AI [112] motivate the need for explainability approaches that offer guarantees of rigor, and so can be trusted. Such need is underscored by the ample evidence of bias in ML models [249]. Unfortunately, the most visible XAI approaches [299, 247, 300] offer no guarantees of rigorous. For example, existing results have shown that such informal explanations can be consistent with points in feature space for which the prediction differs [189, 276, 178].

Pioneered by work on explaining boolean classifiers represented with restricted families of bayesian networks [320], there have been a stream of results on formal explainability, which are summarized in recent overviews include [257, 254, 253]. [1]

[1] Additional references include [320, 187, 321, 188, 189, 276, 357, 94, 178, 96, 179, 185, 29, 193, 255, 35, 256, 198, 250, 182, 81, 169, 26, 19, 49, 12, 347, 100, 196, 257, 168, 180, 145, 98, 28, 13, 27, 120, 239, 194, 197, 372, 171, 199, 23].

In addition to the problem of computing one explanation, recent work also studied a number of queries [29, 169, 26], which can be addressed in the context of formal explainability, and which find numerous applications.

By building on the relationship between explainability and logic-based abduction [125, 313, 110, 187], this chapter analyzes two concrete queries, namely feature necessity and relevancy. Given a local explanation problem comprising an ML classifier, an instance (i.e. point in feature space and associated prediction) and a target feature, the goal of feature necessity is to decide whether the target feature occurs in *all* explanations of the given instance. Under the same assumptions, the goal of feature relevancy is to decide whether a feature occurs in *some* explanation of the given instance. For example, and motivated by existing regulations and guidelines (e.g. [114, 115, 112]), the target feature can be a sensitive feature, e.g. age, gender, ethnic origin, etc., and the existence of an explanation that includes the target feature would represent a violation of such regulations.

Feature relevancy is also interesting from a theoretical standpoint, since the problem is in general complete for the second level of the polynomial hierarchy [169]. Thus, feature relevancy represents another practical example where efficient QBF (Quantified Boolean Formula) solvers are important.

This chapter covers both the computational complexity of feature necessity and relevancy, as well as practical algorithms for solving feature relevancy. For example, this chapter details QBF encodings of the feature relevancy decision problem, this chapter also proposes a general purpose algorithm based on the well-known paradigm of abstraction refinement [75]. As a side result, the QBF formulas resulting from modelling feature relevancy can serve as a new source of challenging problem instances for QBF solvers. The chapter also studies families of classifiers for which solving feature relevancy is substantially easier than the general cases. Finally, this chapter presents experimental results confirming that feature relevancy can be solved efficiently in practice for several families of classifiers.

The chapter is organized as follows. Section 6.2 defines the problems of feature necessity and relevancy, and studies the computational complexity of the problem, for a wide range of families of classifiers. Given the complexity results of Section 6.2, Section 6.3 studies general purpose solutions for feature relevancy, and highlights the concrete case of random forests as the example classifiers. Afterwards, Section 6.4 details algorithms for several other families of classifiers, for which more efficient solutions can be devised. These include decision trees, diagrams and graphs, boolean circuits, and monotonic classifiers. Section 6.5 evaluates the algorithms proposed in Sections 6.3 and 6.4 on representative datasets. Finally, Section 6.6 concludes this chapter.

## Running Examples

Throughout this chapter, we will use several running examples, which will serve to illustrate different claims and results.

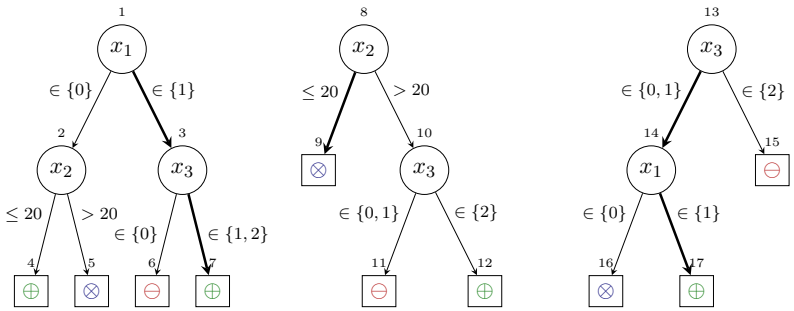



Figure 6.1: Random Forest Running Example.

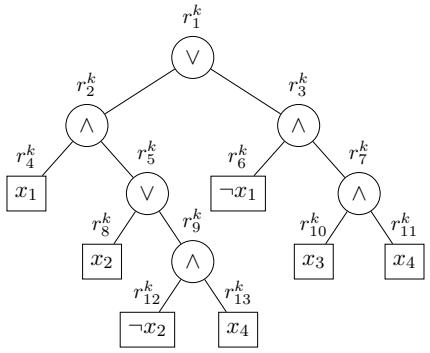



Figure 6.2: d-DNNF Running Example.

$\mathcal{F} = \{1, 2, 3, 4\}$
$\mathbb{D}_i = \{0, 1\}, i = 1, \ldots, 4$
$\mathcal{K} = \{0, 1\}$

(a) Definition of $\mathcal{F}, \mathbb{D}_i, \mathcal{K}$

$$\kappa(\mathbf{x}) = \begin{cases} 1 & \text{if } x_1 + x_2 + x_3 \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

(b) Definition of $\kappa$

Figure 6.3: Example of a monotonic classifier

**Example 19.** Figure 6.1 shows the running example of a RF classifier containing 3 decision trees $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$. The RF represents a classification function $\kappa$ defined on the set of features $\mathcal{F} = \{1, 2, 3\}$ and set of classes $\mathcal{K} = \{c_1, c_2, c_3\} = \{\ominus, \oplus, \otimes\}$. Moreover, the domain of the features are, respectively, $\mathbb{D}_1 = \{0, 1\}$, $\mathbb{D}_2 = [0, 50]$ and $\mathbb{D}_3 = \{0, 1, 2\}$. We consider the instance $(\mathbf{v}, c) = ((1, 10, 1), \oplus)$, the highlighted edges indicate the prediction of each tree.

**Example 20.** Figure 6.2 shows the running example of a d-DNNF classifier representing a boolean function $\kappa(x_1, x_2, x_3, x_4) = (x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_4)$. Consider the instance $(\mathbf{v}, c) = ((1, 0, 1, 1), 1)$.

**Example 21.** The monotonic classifier we consider is shown in Figure 6.3. It has four boolean features and two classes. The instance that is considered throughout this chapter is $(\mathbf{v}, c) = ((1, 1, 1, 1), 1)$.

## 6.2 Feature Necessity & Relevancy: Theory

This section starts by defining feature necessity and relevancy. The next step is to investigate the complexity of feature necessity. The rest of the section is then dedicated to establishing membership and hardness results for feature relevancy.

### 6.2.1 Defining Necessity, Relevancy & Irrelevancy

Given a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$. The sets of AXps and CXps are defined as follows:

$$\begin{aligned} \mathbb{A}(\mathcal{E}) &= \{\mathcal{X} \subseteq \mathcal{F} \,|\, \mathsf{AXp}(\mathcal{X})\} \\ \mathbb{C}(\mathcal{E}) &= \{\mathcal{Y} \subseteq \mathcal{F} \,|\, \mathsf{CXp}(\mathcal{Y})\} \end{aligned} \tag{6.1}$$

Moreover, let $F_{\mathbb{A}}(\mathcal{E}) = \cup_{\mathcal{X} \in \mathbb{A}(\mathcal{E})} \mathcal{X}$ and $F_{\mathbb{C}}(\mathcal{E}) = \cup_{\mathcal{Y} \in \mathbb{C}(\mathcal{E})} \mathcal{Y}$. $F_{\mathbb{A}}(\mathcal{E})$ aggregates the features occurring in any AXp, whereas $F_{\mathbb{C}}(\mathcal{E})$ aggregates the features occurring in any CXp. In addition, minimal hitting set duality between AXps and CXps (see Proposition 1) yields the following result.

**Proposition 12.** $F_{\mathbb{A}}(\mathcal{E}) = F_{\mathbb{C}}(\mathcal{E})$.

By examining the occurrences of features in formal explanations, we can define relevant features and necessary features.

**Definition 39.** Let $t \in \mathcal{F}$ be a target feature, then:

1. $t$ is necessary (for AXps), or AXp-necessary, if $t \in \cap_{\mathcal{X} \in \mathbb{A}} \mathcal{X}$;
2. $t$ is relevant (for AXps), or AXp-relevant, if $t \in F_{\mathbb{A}}(\mathcal{E})$;
3. $t$ is irrelevant (for AXps), or AXp-irrelevant, if $t \notin F_{\mathbb{A}}(\mathcal{E})$.

Similarly, we can define necessity, relevancy and irrelevancy for CXps.

**Definition 40.** Let $t \in \mathcal{F}$ be a target feature, then:

1. $t$ is necessary (for CXps), or CXp-necessary, if $t \in \cap_{\mathcal{Y} \in \mathbb{C}} \mathcal{Y}$;
2. $t$ is relevant (for CXps), or CXp-relevant, if $t \in F_{\mathbb{C}}(\mathcal{E})$;
3. $t$ is irrelevant (for CXps), of CXp-irrelevant, if $t \notin F_{\mathbb{C}}(\mathcal{E})$.

Furthermore, it should be noted that feature irrelevancy is a fairly demanding condition in that, a feature $t$ is irrelevant if it is not included in any subset-minimal set of features that is sufficient for the prediction.

**Example 22.** We consider the d-DNNF classifier example from Figure 6.2, and the instance presented in Example 20, i.e. $(\mathbf{v}, c) = ((1, 0, 1, 1), 1)$. The AXps are $\{1, 2\}, \{2, 3\}$, and the CXp are $\{1, 3\}$ and $\{2\}$. It is straightforward to see that 2 is AXp-necessary, and $1, 2, 3$ are AXp-relevant features for the instance. In contrast, features $1, 2, 3$ are CXp-relevant features. Finally, feature 4 is AXp/CXp-irrelevant for this instance.

**Example 23.** We consider the monotonic classifier in Example 21, and the corresponding instance $(\mathbf{v}, c) = ((1, 1, 1, 1), 1)$. In this case, the set of AXps is the

same as the set of CXps. These correspond to all the combinations of two features disregarding feature 4, that is, the AXps/CXps are $\{1,2\}$, $\{1,3\}$, $\{2,3\}$. Since the set of AXps is the same as the set of CXps, then the set of AXp-necessary features, and the set of CXp-necesary features are the same. As such, in the remainder of the example we will refer only to a feature as necessary to refer to both an AXp-necessary feature and a CXp-necesary feature. Similarly, to relevant and irrelevant features.

As can be seen, no feature belongs to the intersection of all the explanations, that means that there is no necessary feature. On the other hand, features 1, 2, and 3 belong to the union of all explanations, which means that 1,2 and 3 are the relevant features. Finally, feature 4 does not belong to any explanation, and is therefore an irrelevant feature.

Throughout the remainder of the chapter, the problem of deciding feature necessity is represented by the acronym FNP, and the problem of deciding feature relevancy is represented by the acronym FRP. The use of the prefixes (AXp/CXp) will be used when necessary. Furthermore, the following results can be proved.

**Proposition 13.** Given a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, a feature $t \in \mathcal{F}$ is AXp-relevant iff $t$ is CXp-relevant.

*Proof.* This result follows from Proposition 1, since a feature is included in some AXp iff it is included in some CXp. □

As a result, when studying feature relevancy, we can ignore the prefix(es) and simply state whether a feature is relevant (instead of AXp-relevant or CXp-relevant). Furthermore, minimal hitting set duality between AXps and CXps allows to prove the following results:

**Proposition 14.** Given a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, a feature $t \in \mathcal{F}$ is AXp-necessary iff $\{t\}$ is a CXp.

*Proof.* If $t$ is AXp-necessary, then it must be included in all AXps. It follows that $\mathcal{F} \setminus \{t\}$ is not a weak AXp, and hence $\{t\}$ is a CXp. If $\{t\}$ is a CXp, then it must hit (i.e. be included in) all AXps, and so $t$ is AXp-necessary. □

**Proposition 15.** Given a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, a feature $t \in \mathcal{F}$ is CXp-necessary iff $\{t\}$ is an AXp.

*Proof.* By duality (see Proposition 1), we can use the proof of Proposition 14, but replacing AXps with CXps and vice-versa. □

### 6.2.2 Complexity Results for Feature Necessity

We start this section with a lemma which is used in the proof of the next proposition.

**Lemma 1.** There exists an AXp that does not include a target feature $t \in \mathcal{F}$ iff $\mathsf{WAXp}(\mathcal{F} \setminus \{t\})$ holds.

*Proof.* Suppose there is some AXp $\mathcal{Z}$ that does not include $t$. Then, it must be the case that any superset of $\mathcal{Z}$ is a weak AXp. Thus, it must be true for the set $\mathcal{F} \setminus \{t\}$. Hence, if such AXp $\mathcal{Z}$ exists, then $\mathcal{F} \setminus \{t\}$ must be a weak AXp. Suppose that $\mathsf{WAXp}(\mathcal{F} \setminus \{t\})$ holds. Then there must exist at least one subset minimal set $\mathcal{Z} \subseteq \mathcal{F} \setminus \{t\}$ that is an AXp, and such a set does not include $t$. □

**Proposition 16.** If deciding $\mathsf{WAXp}(\mathcal{X})$ is in the complexity class $\mathfrak{C}$, then FNP is in the complexity class co-$\mathfrak{C}$.

*Proof.* From the previous Lemma 1, we can decide feature necessity by a single call to $\mathsf{WAXp}(\mathcal{F} \setminus \{t\})$. Since positive instances of FNP are negative instances of $\mathsf{WAXp}(\mathcal{F} \setminus \{t\})$, we conclude that FNP belongs to co-$\mathfrak{C}$. □

Given the known polynomial complexity of deciding whether a set is a weak AXp for several families of classifiers [257], we then have the following result:

**Corollary 4.** For DTs, XpG's, d-DNNF classifiers and monotonic classifiers, FNP is in P.

### 6.2.3 Feature Relevancy: Membership Results

This section proves a number of membership results for FRP. These will be complemented in the next section with hardness results.

**Proposition 17.** FRP for DTs is in P.

*Proof.* From Proposition 4, we know that enumeration of all CXps can be achieved in polynomial time. Hence, we can simply run the algorithm outlined in the proof of Proposition 4, obtain $F_{\mathbb{C}}(\mathcal{E})$ and decide whether the target feature $t$ is included in $F_{\mathbb{C}}(\mathcal{E})$. □

Moreover, for several families of classifiers, deciding FRP is actually in NP, and so in these cases FRP can (in theory) be decided with one NP oracle call.

**Proposition 18.** If deciding $\mathsf{WAXp}(\mathcal{X})$ is in P, then FRP is in NP.

*Proof.* Let $t \in \mathcal{F}$ be a target feature, and let $\mathcal{X} \subseteq \mathcal{F}$ be some guessed set of features, with $t \in \mathcal{X}$. To decide whether $\mathcal{X}$ is an AXp, we need to check that $\mathsf{WAXp}(\mathcal{X})$ holds, which runs in polynomial time. Then, we must also check that, for all $i \in \mathcal{X}$, $\mathsf{WAXp}(\mathcal{X} \setminus \{i\})$ does not hold, again in polynomial time. Hence, deciding feature relevancy is in NP. □

**Corollary 5.** For DGs, FBDDs, XpGs, d-DNNF classifiers and monotonic classifiers, FRP is in NP.

**Proposition 19.** If deciding $\mathsf{WAXp}(\mathcal{X})$ is in NP, then FRP is in $\Sigma_2^{\mathrm{P}}$.

*Proof.* Let $t \in \mathcal{F}$ be a target feature, and let $\mathcal{X} \subseteq \mathcal{F}$ be some guessed set of features, with $t \in \mathcal{X}$. Following the same arguments presented in Proposition 18, we need to check that $\mathsf{WAXp}(\mathcal{X})$ holds, additionally, we must ensure that, for all $i \in \mathcal{X}$, $\mathsf{WAXp}(\mathcal{X} \setminus \{i\})$ does not hold. By hypothesis, deciding whether $\mathcal{X}$ is an AXp is in NP, this means we can check whether $\mathcal{X}$ is an AXp containing the target feature $t$ in polynomial time, given access to an NP oracle deciding $\mathsf{WAXp}(\mathcal{X})$. Thus, deciding feature relevancy is in $\Sigma_2^{\mathrm{P}}$. □

**Corollary 6.** For DNFs, DLs, DSs, and RFs, FRP is in $\Sigma_2^{\mathrm{P}}$.

### 6.2.4 Feature Relevancy: Hardness Results

We now investigate the hardness of FRP.

**Proposition 20.** FRP for monotonic classifiers is NP-hard.

*Proof.* We say that a CNF is trivially satisfiable if some literal occurs in all clauses. Clearly, SAT restricted to nontrivial CNFs is still NP-complete. Let $\Phi$ be a not trivially satisfiable CNF on variables $x_1, \ldots, x_k$. Let $N = 2k$. Let $\tilde{\Phi}$ be identical to $\Phi$ except that each occurrence of a negative literal $x_i$ $(1 \leq i \leq k)$ is replaced by $x_{i+k}$. Thus $\tilde{\Phi}$ is a CNF on $N$ variables each of which occurs only positively. Define the boolean classifier $\kappa$ (on $N+1$ features) by $\kappa(x_0, x_1, \ldots, x_N) = 1$ iff $x_i = x_{i+k} = 1$ for some $i \in \{1, \ldots, k\}$ or $x_0 \land \tilde{\Phi}(x_1, \ldots, x_N) = 1$. To show that $\kappa$ is monotonic we need to show that $\mathbf{a} \leq \mathbf{b} \Rightarrow \kappa(\mathbf{a}) \leq \kappa(\mathbf{b})$. This follows by examining the two cases in which $\kappa(\mathbf{a}) = 1$: if $a_i = a_{i+k} = 1 \land \mathbf{a} \leq \mathbf{b}$, then $b_i = b_{i+k} = 1$, whereas, if $a_0 \land \tilde{\Phi}(a_1, \ldots, a_N) = 1$ and $\mathbf{a} \leq \mathbf{b}$, then $b_0 \land \tilde{\Phi}(b_1, \ldots, b_N) = 1$ (by positivity of $\tilde{\Phi}$), so in both cases $\kappa(\mathbf{b}) = 1 \geq \kappa(\mathbf{a})$.

Clearly $\kappa(\mathbf{1}_{N+1}) = 1$. There are $k$ obvious AXps of this prediction, namely $\{i, i+k\}$ $(1 \leq i \leq k)$. These are minimal by the assumption that $\Phi$ is not trivially satisfiable. This means that no other AXp contains both $i$ and $i+k$ for any $i \in \{1, \ldots, k\}$. Suppose that $\Phi(\mathbf{u}) = 1$. Let $\mathcal{X}_u$ be $\{0\} \cup \{i \mid 1 \leq i \leq k \land u_i = 1\} \cup \{i+k \mid 1 \leq i \leq k \land u_i = 0\}$. Then $\mathcal{X}_u$ is a weak AXp of the prediction $\kappa(1) = 1$. Furthermore $\mathcal{X}_u$ does not contain any of the AXp's $\{i, i+k\}$. Therefore some subset of $\mathcal{X}_u$ is an AXp and clearly this subset must contain feature 0. Thus if $\Phi$ is satisfiable, then there is an AXp which contains 0.

We now show that the converse also holds. If $\mathcal{X}$ is an AXp of $\kappa(\mathbf{1}_{N+1}) = 1$ containing 0, then it cannot also contain any of the pairs $i, i+k$ $(1 \leq i \leq k)$, otherwise we could delete 0 and still have an AXp. We will show that this implies that we can build a satisfying assignment $\mathbf{u}$ for $\Phi$. Consider first $\mathbf{v} = (v_0, \ldots, v_N)$ defined by $v_i = 1$ if $i \in \mathcal{X}$ $(0 \leq i \leq N)$ and $v_{i+k} = 1$ if neither $i$ nor $i+k$ belongs to $\mathcal{X}$ $(1 \leq i \leq k)$, and $v_i = 0$ otherwise $(1 \leq i \leq N)$. Then $\kappa(\mathbf{v}) = 1$ by definition of an AXp, since $\mathbf{v}$ agrees with the vector 1 on all features in $\mathcal{X}$. We can also note that $v_0 = 1$ since $0 \in \mathcal{X}$. Since $\mathcal{X}$ does not contain $i$ and $i+k$ $(1 \leq i \leq k)$, it follows that $v_i \neq v_{i+k}$. Now let $u_i = 1$ iff $i \in \mathcal{X} \land 1 \leq i \leq k$. It is easy to verify that $\Phi(\mathbf{u}) = \tilde{\Phi}(\mathbf{v}) = \kappa(\mathbf{v}) = 1$.

Thus, determining whether $\kappa(\mathbf{1}_{N+1}) = 1$ has an AXp containing the feature 0 is equivalent to testing the satisfiability of $\Phi$. It follows that FRP is NP-hard for monotonic classifiers by this polynomial reduction from SAT. □

**Example 24.** Given a CNF $\psi := (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_3)$, To reduce $\Phi$ to a monotonic classifier $\kappa$, we replace $\neg x_1$ with $x_4$, $\neg x_2$ with $x_5$ and $\neg x_3$ with $x_6$, and derive a monotonic CNF $\tilde{\Phi} := (x_1 \vee x_2) \wedge (x_4 \vee x_6)$. We then introduce the target variable $x_0$ and construct the monotonic classifier $\kappa := (x_0 \wedge \tilde{\Phi}) \vee [(x_1 \wedge x_4) \vee (x_3 \wedge x_6)]$, we omit $(x_2 \wedge x_5)$ since there is no $\neg x_2$ in $\psi$. Pick an instance $\mathbf{v} = (v_0 = 1, v_1 = 1, v_2 = 1, v_3 = 1, v_4 = 1, v_5 = 1, v_6 = 1)$. It can be verified that $\{1, 4\}$ is an AXp since for any point $\mathbf{u}$ such that $(u_1 = 1, u_4 = 1)$, we have $\kappa(\mathbf{u}) = 1$. It can also be verified that $\{0, 1, 6\}$ containing the target feature 0 is an AXp, and more importantly, this AXp corresponds to true points $(x_1 = 1, x_3 = 0)$ of $\Phi$ which confirm that $\Phi$ is satisfiable. Finally, pick a true point $(x_1 = 0, x_2 = 1, x_3 = 0)$ of $\Phi$, we can construct a weak AXp $\{0, 2, 4, 6\}$ of $\kappa$ such that the target feature 0 cannot be removed. One can extract either AXp $\{0, 2, 4\}$ or $\{0, 2, 6\}$ from $\{0, 2, 4, 6\}$.

**Proposition 21.** FRP for FBDD classifiers is NP-hard.

*Proof.* Let $\psi$ be a CNF formula defined on a variable set $X = \{x_1, \ldots, x_m\}$ and with clauses $\{\omega_1, \ldots, \omega_n\}$. We aim to construct an FBDD classifier $\mathcal{M}$ (representing a classification function $\kappa$) based on $\psi$ and a target variable in polynomial time, such that: $\psi$ is SAT iff for $\kappa$ there is an AXp containing this target variable.

For any literal $l_j \in \omega_i$, replace $l_j$ with $l_j^i$. Let $\psi' = \{\omega_1', \ldots, \omega_n'\}$ denote the resulting CNF formula defined on the new variables $\{x_1^1, \ldots, x_m^1, \ldots x_1^n, \ldots, x_m^n\}$. For each original variable $x_j$, let $I_j^+$ and $I_j^-$ denote the indices of clauses containing literal $x_j$ and $\neg x_j$, respectively. So if $i \in I_j^+$, then $x_j^i \in \omega_i'$, if $i \in I_j^-$, then $\neg x_j^i \in \omega_i'$. To build an FBDD $D$ from $\psi'$: 1) build an FBDD $D_i$ for each $\omega_i'$; 2) replace the terminal node 1 of $D_i$ with the root node of $D_{i+1}$; $D$ is read-once because each variable $x_j^i$ occurs only once in $\psi'$.

Satisfying a literal $x_j^i \in \omega_i'$ means $x_j = 1$, while satisfying a literal $\neg x_j^k \in \omega_k'$ means $x_j = 0$. If both $x_j^i$ and $\neg x_j^k$ are satisfied, then it means we pick inconsistent values for the variable $x_j$, which is unacceptable. Let us define $\phi$ to capture inconsistent values for any variable $x_j$:

$$\phi := \bigvee\nolimits_{1 \leq j \leq m} \left( \left( \bigvee\nolimits_{i \in I_j^+} x_j^i \right) \wedge \left( \bigvee\nolimits_{k \in I_j^-} \neg x_j^k \right) \right) \tag{6.2}$$

If $I_j^+ = \emptyset$, then let $\left( \bigvee_{i \in I_j^+} x_j^i \right) = 0$. If $I_j^- = \emptyset$, then let $\left( \bigvee_{k \in I_j^-} \neg x_j^k \right) = 0$. Any true point of $\phi$ means we pick inconsistent values for some variable $x_j$, so it represents an unacceptable point of $\psi$. To avoid such inconsistency, one needs to at least falsify either $\bigvee_{i \in I_j^+} x_j^i$ or $\bigvee_{k \in I_j^-} \neg x_j^k$ for each variable $x_j$. To build an FBDD $G$ from $\phi$: 1) build FBDDs $G_j^+$ and $G_j^-$ for $\bigvee_{i \in I_j^+} x_j^i$ and $\bigvee_{k \in I_j^-} \neg x_j^k$, respectively; 2) replace the terminal node 1 of $G_j^+$ with the root node of $G_j^-$, let $G_j$ denote the resulting

FBDD; 3) replace the terminal 0 of $G_j$ with the root node of $G_{j+1}$; $G$ is read-once because each variable $x_j^i$ occurs only once in $\phi$.

Create a root node labeled $x_0^0$, link its 1-edge to the root of $D$, 0-edge to the root of $G$. The resulting graph $\mathcal{M}$ is an FBDD representing $\kappa := (x_0^0 \wedge \psi') \vee (\neg x_0^0 \wedge \phi)$, $\kappa$ is a boolean classifier defined on $\{x_0^0, x_1^1, \ldots, x_m^n\}$ and $x_0^0$ is the target variable. The number of nodes of $\mathcal{M}$ is $O(n \times m)$. Let $\mathcal{I} = \{(0,0), (1,1), \ldots (n,m)\}$ denote the set of variable indices, for variable $x_j^i$, $(i,j) \in \mathcal{I}$.

Pick an instance $\mathbf{v} = \{v_0^0, \ldots, v_j^i, \ldots\}$ satisfying every literal of $\psi'$ (i.e. $v_j^i = 1$ and $v_j^k = 0$ for $x_j^i, \neg x_j^k \in \psi'$) and such that $v_0^0 = 1$, then $\psi'(\mathbf{v}) = 1$, and so $\kappa(\mathbf{v}) = 1$. Suppose $\mathcal{X} \subseteq \mathcal{I}$ is an AXp of $\mathbf{v}$: 1) If $\{(i,j), (k,j)\} \subseteq \mathcal{X}$ for some variable $x_j$, where $i \in I_j^+$ and $k \in I_j^-$, then for any point $\mathbf{u}$ of $\kappa$ such that $u_j^i = v_j^i$ for any $(i,j) \in \mathcal{X}$, we have $\kappa(\mathbf{u}) = 1$ and $\phi(\mathbf{u}) = 1$. Moreover, if $\mathbf{u}$ sets $u_0^0 = 1$, then $\kappa(\mathbf{u}) = 1$ implies $\psi'(\mathbf{u}) = 1$, else if $\mathbf{u}$ sets $u_0^0 = 0$, then $\kappa(\mathbf{u}) = 1$ because of $\phi(\mathbf{u}) = 1$. $\kappa(\mathbf{u}) = 1$ regardless the value of $u_0^0$, so $(0,0) \notin \mathcal{X}$. 2) If $\{(i,j), (k,j)\} \not\subseteq \mathcal{X}$ for any variable $x_j$, where $i \in I_j^+$ and $k \in I_j^-$, then for some point $\mathbf{u}$ of $\kappa$ such that $u_j^i = v_j^i$ for any $(i,j) \in \mathcal{X}$, we have $\phi(\mathbf{u}) \neq 1$, in this case $\kappa(\mathbf{u}) = 1$ implies $\psi'(\mathbf{u}) = 1$, besides, any such $\mathbf{u}$ must set $u_0^0 = 1$, so $(0,0) \in \mathcal{X}$.

If case 2) occurs, then $\psi$ is satisfiable. (a satisfying assignment is $x_j = 1$ iff $\exists i \in I_j^+$ s.t. $(i,j) \in \mathcal{X}$). If case 2) never occurs, then $\psi$ is unsatisfiable. It follows that FRP is NP-hard for FBDD classifiers by this polynomial reduction from SAT. □

**Corollary 7.** FRP for d-DNNF and DG classifiers is NP-hard.

*Proof.* The language FBDD is a subset of d-DNNF, so the hardness of FRP for FBDD implies the hardness of FRP for d-DNNF. Additionally, the language FBDD is a subset of DG, so the hardness of FRP for FBDD implies the hardness of FRP for DG. □

**Example 25.** Given a CNF $\psi := (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$. To reduce $\psi$ to a FBDD classifier $\kappa$, we first construct a FBDD $\psi' := (x_1^1 \vee x_2^1 \vee x_3^1) \wedge (\neg x_1^2 \vee \neg x_2^2) \wedge (\neg x_1^3 \vee x_2^3 \vee \neg x_3^3)$. We then build the constraint $\phi := [(x_1^1) \wedge (\neg x_1^2 \vee \neg x_1^3)] \vee [(x_2^1 \vee x_2^3) \wedge (\neg x_2^2)] \vee [(x_3^1) \wedge (\neg x_3^3)]$ and transform it into a FBDD. Then we introduce the target variable $x_0^0$ and construct our FBDD classifier $\kappa := (x_0^0 \wedge \psi') \vee (\neg x_0^0 \wedge \phi)$ (See the Figure 6.4). Pick an instance $\mathbf{v} = (v_0^0 = 1, v_1^1 = 1, v_2^1 = 1, v_3^1 = 1, v_1^2 = 0, v_2^2 = 0, v_1^3 = 0, v_2^3 = 1, v_3^3 = 0)$. It can be verified that $\{(1,1), (3,1)\}$ is an AXp as it represent a unacceptable point of $\psi$. Let $\mathcal{X} = \{(0,0), (1,1), (2,2), (3,3)\}$, then it can be checked for any point $\mathbf{u}$ such that $(u_0^0 = 1, u_1^1 = 1, u_2^2 = 0, u_3^3 = 0)$, we have $\kappa(\mathbf{u}) = 1$, so $\mathcal{X}$ is an AXp, moreover, it represents the true point $(x_1 = 1, x_2 = 0, x_3 = 0)$ of $\psi$. Finally, choose a true point $(0,1,1)$ of $\psi$, we can construct a weak AXp $\{(0,0), (1,2), (1,3), (2,1), (3,1)\}$ where feature $(0,0)$ cannot be removed. In addition, it contains two AXps, namely, $\{(0,0), (1,2), (2,1), (3,1)\}$ and $\{(0,0), (1,3), (2,1), (3,1)\}$.

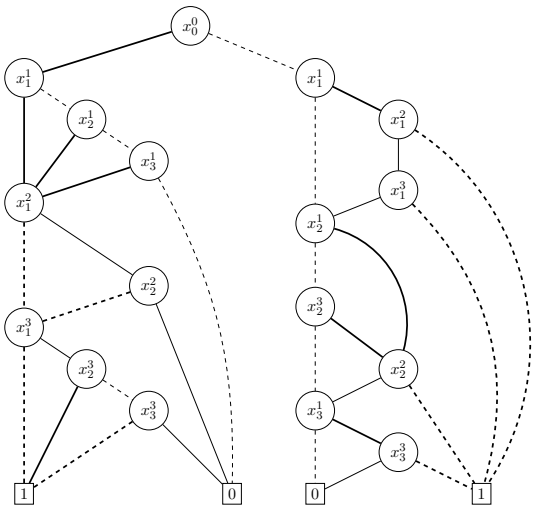



Figure 6.4: FBDD classifier $\kappa$. The left subgraph of the root node is the FBDD representing the modified CNF $\psi'$. The right subgraph of the root node is the FBDD representing the constraint $\phi$. Edges corresponding to value 0 (resp. 1) are indicated by dashed lines (resp. solid lines). Non-terminals are represented as circle nodes, terminal nodes are represented as boxes. The bold edges are consistent with the given instance.

### 6.2.5 Summary of Results & Perspective

Table 6.1 summarizes the computational complexity results for FRP for a number of well-known families of ML classifiers, most of which were proved in the previous sections. Furthermore, the results from Table 6.1 will now be used for deriving algorithms for all the families of classifiers shown in the table.

The presentation of algorithms is organized into two main groups. The first group, described in Section 6.3, exploits membership in $\Sigma_2^{\mathrm{P}}$ to propose two different algorithms. The first one builds on the QBF formulation. In this case, a logic encoding is required to guarantee that the prediction remains unchanged. The second algorithm exploits abstraction refinement, and does not require a dedicated logic encoding for each families of classifiers. The second group of algorithms, described in Section 6.4, devises one dedicated algorithm for each family of classifiers.

## 6.3 Feature Relevancy: General Purpose Algorithms

This section proposes two classes of algorithms for deciding feature relevancy in the case of an *arbitrary* ML classifier. The first class of algorithms builds on the QBF formulation. The second class of algorithms exploits the paradigm of abstraction refinement, which has been used in a wide range of practical settings.

In the rest of this section, we assume a given classifier $\mathcal{M}$, and a logic encoding that allows to decide the following predicates $\mathsf{WAXp}(\mathcal{X})$. To concretize the two

| Classifier | Membership | Hardness | Reference |
|---|---|---|---|
| DT | P | P | Proposition 17 |
| DG | NP | NP | Corollaries 5 and 7 |
| XpG | NP | NP | Corollaries 5 and 7 |
| FBDD | NP | NP | Corollary 5 and Proposition 21 |
| d-DNNF | NP | NP | Corollaries 5 and 7 |
| Monotonic | NP | NP | Corollary 5 and Proposition 20 |

Table 6.1: Summary of membership & hardness results

main approaches proposed in this section, we will consider RFs as the target family of classifiers.

### 6.3.1 QBF Encodings

We outline the main ideas for developing QBF encodings for deciding feature relevancy. Afterwards, we show how these ideas can be applied in the case of a concrete family of classifiers. We can prove that deciding whether feature $t \in \mathcal{F}$ is included in some explanations corresponds to deciding the following QBF statement:

$$\exists(\mathcal{X} \subseteq \mathcal{F}).(t \in \mathcal{X}) \land \mathsf{WAXp}(\mathcal{X}) \land [\land_{j \in \mathcal{X}} \neg \mathsf{WAXp}(\mathcal{X} \setminus \{j\})] \tag{6.3}$$

The proof of Propositions 18 and 19 offers a solution for solving feature relevancy in the case that computing AXps. However, the practical solutions hinted by the proof reveal inefficiencies, namely testing in the worst case (and in polynomial time) the predicate $\mathsf{WAXp}$ a total of $m + 1$ times, with $m = |\mathcal{F}|$. This is also the case with monotonic classifiers.

Below, we propose a different proof argument, which involves far fewer tests of $\mathsf{WAXp}$. This reduction in the number of runs of $\mathsf{WAXp}$ can have important practical impact, including the algorithms proposed in Section 6.3. The proposed approach hinges on the following result:

**Proposition 22.** Let $\mathcal{X} \subseteq \mathcal{F}$ represent a pick of the features, such that, $\mathsf{WAXp}(\mathcal{X})$ holds and $\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$ does not hold. Then, for any AXp $\mathcal{Z} \subseteq \mathcal{X} \subseteq \mathcal{F}$, it must be the case that $t \in \mathcal{Z}$.

*Proof.* Let $\mathcal{Z} \subseteq \mathcal{F}$ by any AXp such that $\mathcal{Z} \subseteq \mathcal{X}$ but $t \notin \mathcal{Z}$. Clearly, by definition $\mathsf{WAXp}(\mathcal{Z})$ must hold. As the predicate $\mathsf{WAXp}(\mathcal{X})$ is monotonic (i.e. if $\mathsf{WAXp}(\mathcal{X})$ holds for $\mathcal{X} \subseteq \mathcal{F}$, then $\mathsf{WAXp}(\mathcal{X}')$ holds for any $\mathcal{X} \subseteq \mathcal{X}' \subseteq \mathcal{F}$), it is also the case that $\mathsf{WAXp}(\mathcal{Z}')$ must hold, with $\mathcal{Z}' = \mathcal{Z} \cup (\mathcal{X} \setminus (\mathcal{Z} \cup \{t\}))$, since $\mathcal{Z} \subseteq \mathcal{Z}' \subseteq \mathcal{F}$. However, by hypothesis, $\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$ does not hold; a contradiction. □

When compared with Propositions 18 and 19, Proposition 22 offers a simpler test to decide whether $t$ is included in some AXp, in that it suffices to guess a set

$\mathcal{X}$ which is a weak AXp, and such that removing $t$ will cause $\mathcal{X} \setminus \{t\}$ not to be a weak AXp. An apparent drawback of this simpler test to decide AXp membership is that the guessed set $\mathcal{X}$ may *not* itself represent an AXp. However, since *any* AXp contained in $\mathcal{X}$ must include $t$, we can then extract an AXp starting from the set $\mathcal{X}$ with existing techniques [198, 169, 168].

As a consequence of Proposition 22, a more compact QBF encoding can be devised:

$$\exists(\mathcal{X} \subseteq \mathcal{F}).(t \in \mathcal{X}) \land \mathsf{WAXp}(\mathcal{X}) \land \neg\mathsf{WAXp}(\mathcal{X} \setminus \{t\}) \tag{6.4}$$

and can be further expanded as:

$$\begin{array}{l} \exists(\mathcal{X} \subseteq \mathcal{F}).(t \in \mathcal{X}) \land \\ \left[\forall(\mathbf{x} \in \mathbb{F}).\left(\bigwedge\nolimits_{i\in\mathcal{X}}(x_i = v_i)\right) \rightarrow (\kappa(\mathbf{x}) = c)\right] \land \\ \left[\exists(\mathbf{x} \in \mathbb{F}).\left(\bigwedge\nolimits_{i\in\mathcal{X}\setminus\{t\}}(x_i = v_i)\right) \land (\kappa(\mathbf{x}) \neq c)\right] \end{array} \tag{6.5}$$

It should be noted that the QBF encoding for (6.5) uses only two levels of quantifiers (i.e. $\exists\forall$). However, one must introduce another level of quantification to account for the auxiliary variables used for representing the matrix in clausal form. We denote this QBF encoding as the $\exists\forall$ encoding.

Furthermore, it is important to note that there is indeed a pure 2QBF encoding for feature relevancy. To achieve this, it suffices to 1) negate (6.4), and 2) decide whether the resulting formula is *false*. The resulting adjusted formula is as follows:

$$\forall(\mathcal{X} \subseteq \mathcal{F}).((t \in \mathcal{X}) \land \mathsf{WAXp}(\mathcal{X})) \rightarrow \mathsf{WAXp}(\mathcal{X} \setminus \{t\}) \tag{6.6}$$

and can be further expanded as:

$$\begin{array}{l} \forall(\mathcal{X} \subseteq \mathcal{F}).(t \notin \mathcal{X}) \lor \\ \left[\forall(\mathbf{x} \in \mathbb{F}).\left(\bigwedge\nolimits_{i\in\mathcal{X}\setminus\{t\}}(x_i = v_i)\right) \rightarrow (\kappa(\mathbf{x}) = c)\right] \lor \\ \left[\exists(\mathbf{x} \in \mathbb{F}).\left(\bigwedge\nolimits_{i\in\mathcal{X}}(x_i = v_i)\right) \land (\kappa(\mathbf{x}) \neq c)\right] \end{array} \tag{6.7}$$

In this case, the existentially quantified auxiliary variables, used for converting the matrix to clausal form, do not change the number of levels of quantification. For this resulting adjusted formula, we are now checking whether *there is no AXp containing the target feature* $t$. When the answer is *No*, it confirms the existence of an AXp $\mathcal{X}$ such that $t \in \mathcal{X}$. We denote this alternative QBF encoding as the $\forall\exists$ encoding.

### 6.3.2 Case Study: Random Forest classifiers

We overview an existing propositional encoding for computing AXps of RFs. Then, we build on this encoding to devise a 2QBF encoding. It should be noted that the general-purpose algorithm for deciding feature relevancy described in the next

section is also built on this propositional encoding.

**Propositional encoding for RFs.** We start by detailing how to encode the classification function $\kappa$ of an RF $\mathcal{M}$. This section exploits the propositional encoding proposed recently for computing AXps of RFs [198][2]. The encoding comprises: 1) the structure of an RF $\mathcal{M}$, and 2) the majority votes. Moreover, we make the following assumptions:

**Assumption 2.** Each $\mathbb{D}_i$ has $n_i$ distinct values or disjoint intervals. Values/intervals are ordered (from 1 to $n_i$).

To present the encoding of $\mathcal{M} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, we introduce some auxiliary boolean variables and predicates:

1. $q_{i,j}$, $1 \leq i \leq m, 1 \leq j \leq n_i$. $q_{i,j} = 1$ if feature $i$ assigned with the $j$-th value/interval from its domain $\mathbb{D}_i$.
2. $p_{i,j}$, $1 \leq i \leq n, 1 \leq j \leq K$, $p_{i,j} = 1$ if tree $\mathcal{T}_i$ predicts the class $j$.
3. $\mathsf{Class}(\mathcal{P})$ denotes the class (i.e. the label of a terminal node) of a path $\mathcal{P}$.
4. $\mathsf{L}(\mathcal{P})$ denotes the set of literals of a path $\mathcal{P}$.
5. $\mathsf{Votes}(c)$ denotes the number of trees picking the class $c \in \mathcal{K}$.

To encode the structure of an RF $\mathcal{M}$, one needs to encode each $\mathcal{T}_i$. The encoding of a tree is achieved by encoding all its paths. The set of paths $\mathbb{P}_i$ of $\mathcal{T}_i$ is encoded as follows:

$$\bigwedge\nolimits_{\mathcal{P} \in \mathbb{P}_i} \left( \bigwedge\nolimits_{l \in \mathsf{L}(\mathcal{P})} l \rightarrow \mathsf{Class}(\mathcal{P}) \right) \tag{6.8}$$

Each feature $i$ is assigned with exactly one value:

$$\bigwedge\nolimits_{1 \leq i \leq m} \sum\nolimits_{j=1}^{n_i} q_{i,j} = 1 \tag{6.9}$$

Each tree $\mathcal{T}_i$ predicts exactly one class:

$$\bigwedge\nolimits_{1 \leq i \leq n} \sum\nolimits_{j=1}^{K} p_{i,j} = 1 \tag{6.10}$$

Both constraints in (6.9), (6.10) represent a cardinality constraint $\mathsf{EqualsOne}()$. Next, we detail how to use cardinality constraints ($\mathsf{AtLeastK}()$) to encode the majority votes. Suppose w.l.o.g. $\mathcal{K} = \{c_{j_1}, c_{j_2}, c_{j_3}\}$ such that $j_1 < j_2 < j_3$, and the prediction of the given instance is $c_{j_2}$. If $(\mathsf{Votes}(c_{j_1}) < \mathsf{Votes}(c_{j_2})) \land (\mathsf{Votes}(c_{j_2}) \geq \mathsf{Votes}(c_{j_3}))$ then the prediction of $\mathcal{M}$ remains unchanged. Otherwise, if $(\mathsf{Votes}(c_{j_1}) \geq \mathsf{Votes}(c_{j_2})) \lor (\mathsf{Votes}(c_{j_2}) < \mathsf{Votes}(c_{j_3}))$ then the prediction of $\mathcal{M}$ changes.

The case where the prediction of $\mathcal{M}$ remains unchanged can be encoded with the following constraints:

$$\sum\nolimits_{i=1}^{n} p_{i,j_2} + \sum\nolimits_{i=1}^{n} \neg p_{i,j_1} \geq 1 + n \tag{6.11}$$

[2]One possible alternative was proposed in more recent work [54]. However, this encoding is less optimized and so it does not scale as well in practice. Other representations of RFs [72, 29, 285] are not applicable in this context.

$$\sum_{i=1}^{n} p_{i,j_2} + \sum_{i=1}^{n} \neg p_{i,j_3} \geq n \tag{6.12}$$

These require the use of $K-1$ cardinality constraints, each comparing the $\mathsf{Votes}(c_{j_2})$ with the votes of some other class.

The case where the prediction of $\mathcal{M}$ changes can be encoded with the following constraints:

$$\sum_{i=1}^{n} p_{i,j_1} + \sum_{i=1}^{n} \neg p_{i,j_2} \geq n \tag{6.13}$$

$$\sum_{i=1}^{n} p_{i,j_3} + \sum_{i=1}^{n} \neg p_{i,j_2} \geq 1 + n \tag{6.14}$$

These require only 2 (instead of $K-1$) cardinality constraints [198].

**$\exists\forall$ QBF encoding.** This 2QBF formulation comprises two predicates. To encode these two predicates, one needs to distinguish the set of picked features $\mathcal{X}$ and the set of unpicked features $\mathcal{F} \setminus \mathcal{X}$. For any feature $i \in \mathcal{X}$, its value is fixed, which means that there is exactly one $q_{i,j} = 1$. For any feature $i \notin \mathcal{X}$, its value is not fixed, which means that any legal combination of the $q_{i,j}$ is allowed, excluding illegal combinations since they violate the constraint (6.9). Given (6.5), we use two copies ($\mathcal{M}^0$ and $\mathcal{M}^t$) of the same RF $\mathcal{M}$ to encode the problem. $\mathcal{M}^0$ encodes $\mathsf{WAXp}(\mathcal{X})$ (i.e. the prediction of $\mathcal{M}$ remains unchanged, $[\kappa^0(\mathbf{x}) = c]$), $\mathcal{M}^t$ encodes $\neg\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$, or equivalently $\mathsf{WCXp}(\mathcal{F} \setminus (\mathcal{X} \setminus \{t\}))$ (i.e. the prediction of $\mathcal{M}$ changes, $[\kappa^t(\mathbf{x}) \neq c]$). To present the constraints included in this QBF encoding, we need to introduce additional auxiliary boolean variables:

1. $s_i$, $1 \leq i \leq m$. $s_i$ is a selector such that $s_i = 1$ iff feature $i$ is included in $\mathcal{X}$. Moreover, $s_i = 1$ also means that feature $i$ must be fixed to its given value $v_i$, while $s_i = 0$ means that feature $i$ can take any value from its domain.
2. $\mathbf{w}_i$, $1 \leq i \leq m$. $\mathbf{w}_i$ is a set of boolean variables (a bit vector) for $\mathbb{D}_i$ such that $|\mathbf{w}_i| = \log_2 n_i$. Since values/intervals of $\mathbb{D}_i$ are ordered, each value/interval has an index (from 1 to $n_i$) that can be represented by an assignment of $\mathbf{w}_i$. Let $g : \mathbb{B}^{|\mathbf{w}_i|} \to \mathbb{N}$ be a function mapping binary numbers to the indices of values/intervals of $\mathbb{D}_i$. The space of $\mathbf{w}_i$ is usually larger than $\mathbb{D}_i$, but due to constraint (6.9) that we always pick a value from $\mathbb{D}_i$, we leave some $g(\mathbf{w}_i)$ undefined. More importantly, $\mathbf{w}_i$ is activated if $i \notin \mathcal{X}$ (i.e. $s_i = 0$), and $\mathbf{w}_i$ is deactivated if $i \in \mathcal{X}$ (i.e. $s_i = 1$).

Suppose, for the given instance $\mathbf{v} = (v_1, \ldots, v_m)$, that each value $v_i$ corresponds to the first literal $z_{i,1}$ of its domain $\mathbb{D}_i$, so the instance is represented as $\mathbf{v} = (z_{1,1}, \ldots, z_{m,1})$. Let $\Omega(\kappa^0)$ (resp. $\Omega(\kappa^t)$) denote the set of variables of the encoding of $\mathcal{M}^0$ (resp. $\mathcal{M}^t$).

The QBF encoding based on (6.5) (quantifiers and constraints) is as follows:

1. $\exists(s_1, \ldots, s_m)$
2. $\forall(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$
3. $\exists(\Omega(\kappa^0) \cup \Omega(\kappa^t))$
4. $\bigwedge_{1 \leq i \leq m}(s_i \to q^0_{i,1})$

5. $\bigwedge_{1\leq i\leq m, 1\leq j\leq n_i}(\neg s_i \land (g(\mathbf{w}_i) = j-1) \rightarrow q^0_{i,j})$ [3]
6. $\Big[\kappa^0(q^0_{1,1},\ldots,q^0_{1,n_1},\ldots,q^0_{m,1},\ldots,q^0_{m,n_m}) = c\Big]$
7. $\bigwedge_{1\leq i\leq m, i\neq t}(s_i \rightarrow q^t_{i,1})$
8. $\Big[\kappa^t(q^t_{1,1},\ldots,q^t_{1,n_1},\ldots,q^t_{m,1},\ldots,q^t_{m,n_m}) \neq c\Big]$
9. $s_t$

The first existential quantifier picks a weak AXp candidate $\mathcal{X}$. The universal quantifier considers all possible values of $\mathbb{F}$. The second existential quantifier assigns values to the remaining variables. Line 4 states, for any feature $i$ of $\mathcal{M}^0$, if $i \in \mathcal{X}$, then it is fixed to the given value. Line 5 states, for any feature $i$ of $\mathcal{M}^0$, if $i \notin \mathcal{X}$, then if the value represented by the $\mathbf{w}_i$ equals $j-1$ then feature $i$ is assigned the $j$-th value/interval. Line 6 is the propositional encoding of $\mathcal{M}^0$ comprising constraints (6.8) to (6.12). Line 7 states that, for any feature $i$ (except the target feature $t$) of $\mathcal{M}^t$, if $i \in \mathcal{X}$, then it is fixed to the given value. Line 8 is the propositional encoding of $\mathcal{M}^t$ comprising constraints (6.8) to (6.10), and constraints (6.13) to (6.14). Line 9 states that target feature $t$ is included in $\mathcal{X}$.

**$\forall\exists$ QBF encoding.** Next, we detail the alternative 2QBF encoding based on (6.7), we use $\mathcal{M}^0$ to encode $\neg\mathsf{WAXp}(\mathcal{X})$, and $\mathcal{M}^t$ to encode $\mathsf{WAXp}(\mathcal{X}\setminus\{t\})$:

1. $\forall(s_1,\ldots,s_m,\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_m)$
2. $\exists(\Omega(\kappa^0)\cup\Omega(\kappa^t))$
3. $\bigwedge_{1\leq i\leq m}(s_i \rightarrow q^0_{i,1})$
4. $\sigma^0 \leftrightarrow \Big[\kappa^0(q^0_{1,1},\ldots,q^0_{1,n_1},\ldots,q^0_{m,1},\ldots,q^0_{m,n_m}) \neq c\Big]$
5. $\bigwedge_{1\leq i\leq m, i\neq t}(s_i \rightarrow q^t_{i,1})$
6. $\bigwedge_{1\leq i\leq m, i\neq t, 1\leq j\leq n_i}(\neg s_i \land (g(\mathbf{w}_i) = j-1) \rightarrow q^t_{i,j})$
7. $\bigwedge_{1\leq j\leq n_t}((g(\mathbf{w}_t) = j-1) \rightarrow q^t_{t,j})$
8. $\sigma^t \leftrightarrow \Big[\kappa^t(q^t_{1,1},\ldots,q^t_{1,n_1},\ldots,q^t_{m,1},\ldots,q^t_{m,n_m}) = c\Big]$
9. $\neg s_t \lor \sigma^0 \lor \sigma^t$

The universal quantifier picks all possible subsets of $\mathcal{F}$ as well as all possible values of $\mathbb{F}$. The existential quantifier assigns values to the remaining variables. Line 4 is the propositional encoding of $\mathcal{M}^0$, and we associate it with a variable $\sigma^0$. Line 7 states that, for the target feature $t$ of $\mathcal{M}^t$, it is always not fixed. Line 8 is the propositional encoding of $\mathcal{M}^t$, and we also associate it with a variable $\sigma^t$. Line 9 states that if feature $t$ is picked and $\mathcal{X}$ is a weak AXp then $\mathcal{X}\setminus\{t\}$ is still a weak AXp, which means there is no explanation containing target feature $t$. If this is not the case, then there is an AXp containing $t$.

**Example 26.** Given the RF in Figure 6.1 and the instance $\mathbf{v} = (1, 10, 1)$, suppose the target feature is 3. For feature 1, define variables $\{q_{1,1}, q_{1,2}\}$ such that $q_{1,1} = 1$ iff $x_1 = 0$, $q_{1,2} = 1$ iff $x_1 = 1$. For feature 2, define variables $\{q_{2,1}, q_{2,2}\}$ such that $q_{2,1} = 1$ iff $x_2 \leq 20$ and $q_{2,2} = 1$ iff $x_2 > 20$. And for feature 3, define variables $\{q_{3,1}, q_{3,2}, q_{3,3}\}$ such that $q_{3,1} = 1$ iff $x_3 = 0$, $q_{3,2} = 1$ iff $x_3 = 1$ and $q_{3,3} = 1$ iff

[3] -1 serves as offset since indices range from 1 to $n_i$ but binary numbers range from 0 to $n_i - 1$.

$x_3 = 2$. Moreover, define bit vector $\mathbf{w}_1 = \{w_{1,0}\}$ mapping $\mathbf{w}_1 = 0$ to $q_{1,1}$ and $\mathbf{w}_1 = 1$ to $q_{1,2}$. Bit vector $\mathbf{w}_2 = \{w_{2,0}\}$ mapping $\mathbf{w}_2 = 0$ to $q_{2,1}$ and $\mathbf{w}_2 = 1$ to $q_{2,2}$. Bit vector $\mathbf{w}_3 = \{w_{3,0}, w_{3,1}\}$ mapping $\mathbf{w}_3 = (0,0)$ to $q_{3,1}$, $\mathbf{w}_3 = (0,1)$ to $q_{3,2}$, and $\mathbf{w}_3 = (1,0)$ to $q_{3,3}$. The given instance $\mathbf{v} = (1,10,1)$ is translated to $(q_{1,2}, q_{2,1}, q_{3,2})$ and the QBF encoding based on (6.5) is as follows:

1. $\exists(s_1, s_2, s_3).\forall(w_{1,0}, w_{2,0}, w_{3,0}, w_{3,1}).\exists(\Omega(\kappa^0) \cup \Omega(\kappa^t))$.
2. $(s_1 \rightarrow q^0_{1,2}) \land (s_2 \rightarrow q^0_{2,1}) \land (s_3 \rightarrow q^0_{3,2})$
3. $(\neg s_1 \land \neg w_{1,0} \rightarrow q^0_{1,1}) \land (\neg s_1 \land w_{1,0} \rightarrow q^0_{1,2}) \land$
   $(\neg s_2 \land \neg w_{2,0} \rightarrow q^0_{2,1}) \land (\neg s_1 \land w_{2,0} \rightarrow q^0_{2,2}) \land$
   $(\neg s_3 \land \neg w_{3,0} \land \neg w_{3,1} \rightarrow q^0_{3,1}) \land$
   $(\neg s_3 \land \neg w_{3,0} \land w_{3,1} \rightarrow q^0_{3,2}) \land (\neg s_3 \land w_{3,0} \land \neg w_{3,1} \rightarrow q^0_{3,3})$
4. $\left[\kappa^0(q^0_{1,1}, q^0_{1,2}, q^0_{2,1}, q^0_{2,2}, q^0_{3,1}, q^0_{3,2}, q^0_{3,3}) = c\right]$
5. $(s_1 \rightarrow q^t_{1,2}) \land (s_2 \rightarrow q^t_{2,1})$
6. $\left[\kappa^t(q^t_{1,1}, q^t_{1,2}, q^t_{2,1}, q^t_{2,2}, q^t_{3,1}, q^t_{3,2}, q^t_{3,3}) \neq c\right]$
7. $(s_3)$

Solving these QBF formulas, we will obtain an AXp $\{1,3\}$ containing the target feature.

### 6.3.3 Abstraction Refinement

This section details a general-purpose algorithm for feature relevancy, that solely requires testing whether a set of features $\mathcal{X} \subseteq \mathcal{F}$ is (or is not) a weak AXp, i.e. it just requires the ability to decide (2.15). The novel algorithm iteratively refines an over-approximation (or abstraction) of all the subsets $\mathcal{S}$ of $\mathcal{F}$ such that: i) $\mathcal{S}$ is a weak AXp, and ii) any AXp included in $\mathcal{S}$ also includes the target feature $t$. Formally, the set of subsets of $\mathcal{F}$ that we are interested in is defined as follows:

$$\mathbb{H} \triangleq \{\mathcal{S} \subseteq \mathcal{F} \,|\, \mathsf{WAXp}(\mathcal{S}) \land \forall(\mathcal{X} \subseteq \mathcal{S}).\, [\mathsf{AXp}(\mathcal{X}) \rightarrow (t \in \mathcal{X})]\} \tag{6.15}$$

Evidently, $\mathcal{X} \in \mathbb{H}$ iff $\mathcal{X}$ respects the conditions of Proposition 22.

The proposed algorithm iteratively refines the over-approximation of set $\mathbb{H}$ until one can decide with certainty whether $t$ is included in some AXp. The refinement step involves exploiting counterexamples as these are identified.[4] In practice, it will in general be impractical to manipulate such over-approximation of set $\mathbb{H}$ directly. As a result, we use a propositional formula (in fact a CNF formula) $\mathcal{H}$, such that the models of $\mathcal{H}$ encode the subsets of features about which we have yet to decide whether each of those subsets only contains AXps that include $t$. (Formula $\mathcal{H}$ is defined on a set of feature selectors $S = \{s_1, \ldots, s_m\}$, where assigning $s_i = 1$ denotes that feature $i$ is included in a given set $\mathcal{X}$.) The algorithm then iteratively refines the over-approximation by filtering out sets of sets that have been shown not to be included in $\mathbb{H}$, i.e. the so-called counterexamples.

[4] The approach is referred to as counterexample-guided abstraction refinement (CEGAR), since the use of counterexamples in abstraction refinement can be related with earlier work (with the same name) for model checking of software and hardware systems [75].

**Algorithm 9** Deciding feature relevancy for an arbitrary classifier

**Input**: Instance $\mathbf{v}$, Target Feature $t$; Features $\mathcal{F}$, Classifier $\kappa$

1: **function** FRPCGR($\mathbf{v}, t; \mathcal{F}, \kappa$)
2: $\quad\mathcal{H} \leftarrow \emptyset$ ▷ $\mathcal{H}$ overapproximates $\mathbb{H}$
3: $\quad$**repeat**
4: $\quad\quad$(outc, $\mathbf{s}$) $\leftarrow$ SAT($\mathcal{H}, s_t$) ▷ Pick candidate weak AXp containing $t$
5: $\quad\quad$**if** outc = **true then**
6: $\quad\quad\quad\mathcal{X} \leftarrow \{i \in \mathcal{F} \,|\, s_i = 1\}$ ▷ Set $\mathcal{X}$: candidate weak AXp, $t \in \mathcal{X}$
7: $\quad\quad\quad\mathcal{Y} \leftarrow \{i \in \mathcal{F} \,|\, s_i = 0\}$ ▷ $\mathcal{Y}$ represents $\mathcal{F} \setminus \mathcal{X}$
8: $\quad\quad\quad$**if** $\neg$WAXp($\mathcal{X}$) **then** ▷ Is $\mathcal{X}$ not a weak AXp?
9: $\quad\quad\quad\quad\mathcal{H} \leftarrow \mathcal{H} \cup$ newPosCl($\mathcal{Y}; t, \kappa$) ▷ $\mathcal{X}$ *not* weak AXp; block set
10: $\quad\quad\quad$**else** ▷ $\mathcal{X}$ is a weak AXp
11: $\quad\quad\quad\quad$**if** $\neg$WAXp($\mathcal{X} \setminus \{t\}$) **then** ▷ $\mathcal{X} \setminus \{t\}$ not a weak AXp?
12: $\quad\quad\quad\quad\quad$reportWeakAXp($\mathcal{X}$) ▷ $t$ is included in any AXp $\mathcal{Z} \subseteq \mathcal{X}$
13: $\quad\quad\quad\quad\quad$**return true**
14: $\quad\quad\quad\quad\mathcal{H} \leftarrow \mathcal{H} \cup$ newNegCl($\mathcal{X}; t, \kappa$) ▷ $t$ unneeded; block set
15: $\quad$**until** outc = **false**
16: $\quad$**return false** ▷ If $\mathcal{H}$ becomes inconsistent, then no AXp contains $t$

Algorithm 9 summarizes the proposed approach[5]. Algorithms 10 and 11 provide supporting functions. We now detail the key aspects of Algorithm 9. The algorithm iteratively uses an NP oracle (in fact a SAT solver) to pick (or *guess*) a subset $\mathcal{X}$ of $\mathcal{F}$, such that any previously picked set is not repeated. Since we are interested in feature $t$, we enforce that $t \in \mathcal{X}$. (This step is shown in lines 4 to 7.) Given a set $\mathcal{X}$ of picked features, that includes the target feature $t$, we can check the conditions of Proposition 22, namely:

1. $\mathcal{X}$ is a weak AXp; and
2. $\mathcal{X} \setminus \{t\}$ is not a weak AXp.

If the two conditions above hold, then we know that $\mathcal{X}$ belongs to set $\mathbb{H}$. Furthermore, $\mathcal{X}$ represents a witness that there must exist some AXp that contains $t$, and we know how to compute such an AXp by starting from $\mathcal{X}$. If the picked set $\mathcal{X}$ is not a weak AXp, then we can safely remove it from further consideration. This is achieved by enforcing that at least one of the non-picked elements is picked in the future. Why? Because we want to find a set that is at least a weak AXp, and the set we picked is not one. (As can be observed $\mathcal{H}$ is updated with a positive clause that captures this constraint, as shown in line 9.) After adding the new clause, the algorithm repeats the loop. Otherwise, the picked set $\mathcal{X}$ is a weak AXp (and so the first condition above holds). As a result, we now need to check whether removing $t$ makes $\mathcal{X} \setminus \{t\}$ not to be a weak AXp. If $\mathcal{X} \setminus \{t\}$ is not a weak AXp, then we know that any weak AXp included in $\mathcal{X}$ must include $t$, and this also applies to any (subset-minimal weak) AXp. In this case, the algorithm reports $\mathcal{X}$ as a weak AXp that is *guaranteed* to be included in $\mathbb{H}$. (This is shown in line 12.) It should

[5]The algorithms are parametrized with the arguments shown after the semi-colon.

**Algorithm 10** Create new positive clause (example)

**Input**: Set $\mathcal{Y}$; $t$, $\kappa$, ...

1: **function** newPosCl($\mathcal{Y}; t, \kappa, \ldots$)
2: **for all** $i \in \mathcal{Y}$ **do**
3: **if** $\neg$WAXp($\mathcal{F} \setminus (\mathcal{Y} \setminus \{i\})$) **then**
4: $\mathcal{Y} \leftarrow \mathcal{Y} \setminus \{i\}$
5: $\omega \leftarrow (\vee_{i \in \mathcal{Y}} s_i)$
6: **return** $\omega$

**Algorithm 11** Create new negative clause (example)

**Input**: Set $\mathcal{X}$; $t$, $\kappa$, ...

1: **function** newNegCl($\mathcal{X}; t, \kappa, \ldots$)
2: **for all** $i \in \mathcal{X} \setminus \{t\}$ **do**
3: **if** WAXp($\mathcal{X} \setminus \{t, i\}$) **then**
4: $\mathcal{X} \leftarrow \mathcal{X} \setminus \{i\}$
5: $\omega \leftarrow (\vee_{i \in \mathcal{X} \setminus \{t\}} \neg s_i)$
6: **return** $\omega$

be noted that $\mathcal{X}$ is not necessarily an AXp. However, by Proposition 22, $\mathcal{X}$ is guaranteed to be a weak AXp such that *any* of the AXps contained in $\mathcal{X}$ *must* include feature $t$. Furthermore, we can extract an AXp from a weak AXp with a polynomial number of calls to an oracle that decides (2.15), and in this case we are guaranteed to always pick one that includes $t$. Finally, the last case is when allowing $t$ to take any value does not cause the prediction to change. This means we picked a set $\mathcal{X}$ that is a weak AXp, but not all AXps in $\mathcal{X}$ include the target feature $t$ (again due to Proposition 22). As a result, we must prevent the same weak AXp from being re-picked. This is achieved by requiring that at least one of the picked features not to be picked again in the future. (This is shown in line 14. As can be observed, $\mathcal{H}$ is updated with a negative clause that captures this constraint.)

With respect to the clauses that are added to $\mathcal{H}$ at each step, as shown in Algorithms 10 and 11, one can envision optimizations (shown in lines 2 to 4 in both algorithms) that heuristically aim at removing features from the given sets, and so produce shorter (and so logically stronger) clauses. The insight is that any feature, which can be deemed irrelevant for the condition used for constructing the clause, can be safely removed from the set. For the experiments, we opted to use the simplest approach for constructing the clauses, and so opting to reduce the number of classification queries. Nevertheless, simple optimizations are easy to implement. For example, with respect to the last case (i.e. adding a negative clause in line 14), $\mathcal{X} \setminus \{t\}$ must be a weak AXp. From (2.15), this test requires deciding the satisfiability of $\wedge_{i \in \mathcal{X} \setminus \{t\}} (x_i = v_i) \wedge (\kappa(\mathbf{x}) \neq c)$, and getting an unsatisfiability result. Hence, a simple refinement of $\mathcal{X}$ is given by the unsatisfiable core yielded by the satisfiability test.

Given the above discussion, we can conclude that the proposed algorithm is

sound, complete and terminating for deciding feature relevancy for arbitrary classifiers.

**Proposition 23.** For a classifier $\mathcal{M}$, defined on set of features $\mathcal{F}$, with $\kappa$ mapping $\mathbb{F}$ to $\mathcal{K}$, and an instance $(\mathbf{v}, c)$, $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, and a target feature $t \in \mathcal{F}$, Algorithm 9 returns a set $\mathcal{X} \subseteq \mathcal{F}$ iff $\mathcal{X}$ is a weak AXp for $(\mathbf{v}, c)$, with the property that any AXp $\mathcal{Z} \subseteq \mathcal{X}$ is such that $t \in \mathcal{Z}$.

*Proof.* A set $\mathcal{P}$ respects set $\mathbb{H}$ if $\mathcal{S}$ is a weak AXp, and any of its subsets $\mathcal{X}$ that is an AXp is such that $t \in \mathcal{X}$.

1. Algorithm 9 is terminating. At each step, the algorithm adds a clause that guarantees that a picked assignment is not repeated. In total, $2^{|\mathcal{F}|}$ assignments can be made to the $s_i$ variables. Hence, the main loop of Algorithm 9 executes at most $2^{|\mathcal{F}|}$ times.
2. Algorithm 9 is sound. Given the conditions used to report a picked $\mathcal{P}$, then by Proposition 22 we know that this picked set respects set $\mathbb{H}$, and so any AXp contained in $\mathcal{P}$ will include $t$.
3. Algorithm 9 is complete. It is plain that each clause added to $\mathcal{H}$ blocks only sets that ought not be included in $\mathbb{H}$. The SAT solver will enumerate assignments (i.e. and so a picked set) while that set is not yet blocked by clauses added to $\mathcal{H}$. If there exists a set $\mathcal{P}$ that respects $\mathcal{H}$, then it will eventually be picked.

□

## 6.4 Feature Relevancy: Classifier-Specific Algorithms

This section investigates the solution of deciding feature relevancy for specific families of classifiers, including d-DNNF circuits and monotonic classifiers. Additionally, it is worth noting that for DTs, as proven in Proposition 17, deciding feature relevancy is in P.

### 6.4.1 Case Study: d-DNNF Circuits

This section details a propositional encodings that decide feature relevancy for d-DNNFs. The propositional encoding follow the approach described in the proof of Proposition 22, and comprise two copies ($\mathcal{M}^0$ and $\mathcal{M}^t$) of the same d-DNNF classifier $\mathcal{M}$, $\mathcal{M}^0$ encodes $\mathsf{WAXp}(\mathcal{X})$ (i.e. the prediction of $\kappa$ remains unchanged), $\mathcal{M}^t$ encodes $\neg\mathsf{WAXp}(\mathcal{X} \setminus \{t\})$ (i.e. the prediction of $\kappa$ changes). The encoding is polynomial in the size of classifier's representation. The encoding is applicable to the case $\kappa(\mathbf{x}) = 0$. The case $\kappa(\mathbf{x}) = 1$ can be transformed to $\neg\kappa(\mathbf{x}) = 0$, so we assume both d-DNNF $\mathcal{M}$ and its negation $\neg\mathcal{M}$ are given. To present the propositional encoding, we define some auxiliary boolean variables and predicates:

1. $r_j^k$ $(1 \leq j \leq |\mathcal{M}|, 0 \leq k \leq m)$, $r_j^k$ is the indicator of node $j$ in the $k$-th replica, such that $r_j^k = 1$ if the sub-d-DNNF rooted at node $j$ in $k$-th replica is consistent. Let the root node of d-DNNF be indexed by 1.

Table 6.2: SAT encoding for deciding whether there is a weak AXp including feature $t$, where $0 \leq k \leq m, 1 \leq i \leq m, 1 \leq j \leq |\mathcal{M}|$

| Conditions | Constraints | Fml # |
|---|---|---|
| $\mathsf{Ter}(j), \mathsf{Feat}(j,i), \mathsf{Sat}(\mathsf{Lit}(j), v_i)$ | $r_j^k$ | (6.2.1) |
| $\mathsf{Ter}(j), \mathsf{Feat}(j,i), \neg\mathsf{Sat}(\mathsf{Lit}(j), v_i), i = k$ | $r_j^k$ | (6.2.2) |
| $\mathsf{Ter}(j), \mathsf{Feat}(j,i), \neg\mathsf{Sat}(\mathsf{Lit}(j), v_i), i \neq k$ | $r_j^k \leftrightarrow \neg s_i$ | (6.2.3) |
| $\neg\mathsf{Ter}(j), \mathsf{Oper}(j) = \vee$ | $r_j^k \leftrightarrow \bigvee_{l \in \mathsf{children}(j)} r_l^k$ | (6.2.4) |
| $\neg\mathsf{Ter}(j), \mathsf{Oper}(j) = \wedge$ | $r_j^k \leftrightarrow \bigwedge_{l \in \mathsf{children}(j)} r_l^k$ | (6.2.5) |
| $\kappa(\mathbf{v}) = 0$ | $\neg r_1^0$ | (6.2.6) |
| $\kappa(\mathbf{v}) = 0$ | $s_i \leftrightarrow r_1^i$ | (6.2.7) |
| | $s_t$ | (6.2.8) |

2. $\mathsf{Ter}(j) = 1$ if the node $j$ is a terminal node, otherwise node $j$ is a non-terminal node.
3. $\mathsf{Feat}(j,i) = 1$ if the terminal node $j$ is labeled with feature $i$.
4. $\mathsf{Sat}(\mathsf{Lit}(j), v_i) = 1$ if for terminal node $j$, the literal on feature $i$ is satisfied by the value $v_i$.

**Propositional encoding.** The idea behind this encoding is checking consistency of the d-DNNF classifier. It is applicable to the case $\kappa(\mathbf{x}) = 0$. The case $\kappa(\mathbf{x}) = 1$ can be transformed to the $\neg\kappa(\mathbf{x}) = 0$, so we assume both d-DNNF $\mathcal{M}$ and its negation $\neg\mathcal{M}$ are given. This encoding is summarized in Table 6.2. As literals are terminal d-DNNFs, the values of the selector variables only affect the values of the indicator variables of terminal nodes. Constraint (6.2.1) states that for any terminal node $j$ whose literal is consistent with the given instance, its indicator $r_j^k$ is always consistent regardless of the value of $s_i$. On the contrary, constraint (6.2.3) states that for any terminal node $j$ whose literal is inconsistent with the given instance, its indicator $r_j^k$ is consistent iff feature $i$ is not picked, in other words, feature $i$ can take any value. Because replica $k$ ($k > 0$) is used to check the necessity of including feature $k$ in $\mathcal{X}$, we assume the value of the local copy of selector $s_k$ is 0 in replica $k$. In this case, as defined in constraint (6.2.2), even though terminal node $j$ labeled feature $k$ has a literal that is inconsistent with the given instance, its indicator $r_j^k$ is consistent. Constraint (6.2.4) defines the indicator for an arbitrary $\vee$ node $j$. Constraint (6.2.5) defines the indicator for an arbitrary $\wedge$ node $j$. Together, these constraints declare how the consistency is propagated through the entire d-DNNF. Constraint (6.2.6) states that the prediction of the d-DNNF classifier $\mathcal{M}$ remains 0 since the selected features form a weak AXp. Constraint (6.2.7) states that if feature $i$ is selected, then removing it will change the prediction of $\mathcal{M}$. Finally,

constraint (6.2.8) indicates that feature $t$ must be included in $\mathcal{X}$.

**Example 27.** Given the d-DNNF classifier of Figure 6.2 and the instance $(\mathbf{v}_1, c_1) = ((0,1,0,0),0)$, suppose that the target feature is 3. We have selectors $\mathbf{s} = \{s_1, s_2, s_3, s_4\}$, and the encoding is as follows:

1. $(r_1^0 \leftrightarrow r_2^0 \vee r_3^0) \wedge (r_2^0 \leftrightarrow r_4^0 \wedge r_5^0) \wedge (r_3^0 \leftrightarrow r_6^0 \wedge r_7^0) \wedge (r_5^0 \leftrightarrow r_8^0 \vee r_9^0) \wedge (r_7^0 \leftrightarrow r_{10}^0 \wedge r_{11}^0) \wedge (r_9^0 \leftrightarrow r_{12}^0 \wedge r_{13}^0) \wedge (r_4^0 \leftrightarrow \neg s_1) \wedge (r_6^0 \leftrightarrow 1) \wedge (r_8^0 \leftrightarrow 1) \wedge (r_{10}^0 \leftrightarrow \neg s_3) \wedge (r_{11}^0 \leftrightarrow \neg s_4) \wedge (r_{12}^0 \leftrightarrow \neg s_2) \wedge (r_{13}^0 \leftrightarrow \neg s_4) \wedge (\neg r_1^0) \wedge (s_3)$
2. $(r_1^3 \leftrightarrow r_2^3 \vee r_3^3) \wedge (r_2^3 \leftrightarrow r_4^3 \wedge r_5^3) \wedge (r_3^3 \leftrightarrow r_6^3 \wedge r_7^3) \wedge (r_5^3 \leftrightarrow r_8^3 \vee r_9^3) \wedge (r_7^3 \leftrightarrow r_{10}^3 \wedge r_{11}^3) \wedge (r_9^3 \leftrightarrow r_{12}^3 \wedge r_{13}^3) \wedge (r_4^3 \leftrightarrow \neg s_1) \wedge (r_6^3 \leftrightarrow 1) \wedge (r_8^3 \leftrightarrow 1) \wedge (r_{10}^3 \leftrightarrow 1) \wedge (r_{11}^3 \leftrightarrow \neg s_4) \wedge (r_{12}^3 \leftrightarrow \neg s_2) \wedge (r_{13}^3 \leftrightarrow \neg s_4) \wedge (s_3 \leftrightarrow r_1^3)$

Given the AXps listed in Example 20, by solving these formulas we will either obtain $\{1,3\}$ or $\{1,4\}$ as the AXp.

### 6.4.2 Case Study: Monotonic Classifiers

This section adapts the previous proposed general-purpose algorithm to decide feature relevancy in the case of monotonic classifiers. No assumption is made regarding the actual implementation of the monotonic classifier.

Given a monotonic classifier $\mathcal{M}$ and an instance $\mathbf{v} = (v_1, \ldots, v_m)$, let $\mathbf{v}_L = (v_{L_1}, \ldots, v_{L_N})$ be the lower bound of $\mathbf{v}$ and $\mathbf{v}_U = (v_{U_1}, \ldots, v_{U_N})$ be the upper bound of $\mathbf{v}$, that is, $\mathbf{v}_L \leq \mathbf{v} \leq \mathbf{v}_U$. Define $\tilde{\mathbf{v}}_L$ ($\tilde{\mathbf{v}}_U$) as the updated lower (upper) bound with respect to a given $\mathcal{Z} \subseteq \mathcal{F}$.

$$\tilde{\mathbf{v}}_L := \bigwedge_{i \in \mathcal{Z}} (v_{L_i} = v_i) \bigwedge_{j \notin \mathcal{Z}} (v_{L_j} = \lambda(j)) \tag{6.16}$$

$$\tilde{\mathbf{v}}_U := \bigwedge_{i \in \mathcal{Z}} (v_{U_i} = v_i) \bigwedge_{j \notin \mathcal{Z}} (v_{U_j} = \mu(j)) \tag{6.17}$$

Features not in $\mathcal{Z}$ are deemed universal, we need to account for the range of possible values that these universal features can take. For that, we update lower and upper bounds on the predicted classes. For the features in $\mathcal{Z}$ we must use the values dictated by $\mathbf{v}$. Next, define $\mathsf{WAXp}$ predicates as follow:

$$\mathsf{WAXp}(\mathcal{X}) := [\kappa(\tilde{\mathbf{v}}_L) = \kappa(\tilde{\mathbf{v}}_U)] \tag{6.18}$$

So $\neg\mathsf{WAXp}(\mathcal{X}) := [\kappa(\tilde{\mathbf{v}}_L) \neq \kappa(\tilde{\mathbf{v}}_U)]$, also note that $\neg\mathsf{WAXp}(\mathcal{X})$ is equivalent to $\mathsf{WCXp}(\mathcal{F} \setminus \mathcal{X})$. Given a $\mathcal{X}$, if the lower and upper bounds do not differ, then $\mathcal{X}$ is a weak AXp. Otherwise, $\mathcal{F} \setminus \mathcal{X}$ is a weak CXp. Each execution of the predicate $\mathsf{WAXp}$ comprises at most two calls to the classification function $\kappa$. Besides, for different set $\mathcal{Z}$, one needs to obtain updated lower bound $\tilde{\mathbf{v}}_L$ and upper bound $\tilde{\mathbf{v}}_U$ before invoking the predicates $\mathsf{WAXp}(\cdot)$ in the algorithms Algorithm 9, Algorithm 10 and Algorithm 11.

**Example 28.** We consider the monotonic classifier of Figure 6.3, with instance $(\mathbf{v}, c) = ((1,1,1,1),1)$. Table 6.3 summarizes a possible execution of the algorithm

Table 6.3: Example algorithm execution for $t = 4$

| $\mathbf{s}$ | $\mathcal{X}$ | $\mathcal{Y}$ | $\kappa(\mathbf{v}_L)$ | $\kappa(\mathbf{v}_U)$ | Decision | New clause | Line |
|---|---|---|---|---|---|---|---|
| $(0,0,0,1)$ | $\{4\}$ | $\{1,2,3\}$ | 0 | 1 | New pos clause | $(s_1 \vee s_2 \vee s_3)$ | 9 |
| $(1,0,0,1)$ | $\{1,4\}$ | $\{2,3\}$ | 0 | 1 | New pos clause | $(s_2 \vee s_3)$ | 9 |
| $(1,1,0,1)$ | $\{1,2,4\}$ | $\{3\}$ | 1 | 1 | New neg clause | $(\neg s_1 \vee \neg s_2)$ | 14 |
| $(1,0,1,1)$ | $\{1,3,4\}$ | $\{2\}$ | 1 | 1 | New neg clause | $(\neg s_1 \vee \neg s_3)$ | 14 |
| $(0,1,1,1)$ | $\{2,3,4\}$ | $\{1\}$ | 1 | 1 | New pos clause | $(s_1)$ | 9 |
| — | — | — | – | – | $\mathcal{H}$ inconsistent | – | 16 |

Table 6.4: Example algorithm execution for $t = 1$

| $\mathbf{s}$ | $\mathcal{X}$ | $\mathcal{Y}$ | $\kappa(\mathbf{v}_L)$ | $\kappa(\mathbf{v}_U)$ | Decision | New clause | Line |
|---|---|---|---|---|---|---|---|
| $(1,0,0,0)$ | $\{1\}$ | $\{2,3,4\}$ | 0 | 1 | New pos clause | $(s_2 \vee s_3 \vee s_4)$ | 9 |
| $(1,1,0,0)$ | $\{1,2\}$ | $\{3,4\}$ | 1 | 1 | Weak AXp: $\{1,2\}$ | – | 12 |

when $t = 4$. Similarly, Table 6.4 summarizes a possible execution of the algorithm when $t = 1$. (As with the current implementation, and for both examples, the creation of clauses uses no optimizations.) In general, different executions will be determined by the models returned by the SAT solver.

## 6.5 Experimental Results

This section reports the experimental results of deciding feature relevancy for the classifiers studied in the earlier sections, namely: random forests, d-DNNF circuits, and monotonic classifiers. For each case study, the used benchmarks and the training/compilation procedure will be described, followed by a table summarising the results will be presented. At last, the results will be discussed. All the experiments were performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Monterey. Prototype implementations of the proposed approaches were implemented in Python. The PySAT toolkit [184][6] was employed to implement the propositional encoding as well as the positive and negative clauses of the CEGAR approach. Except the case of solving 2QBF instances, PySAT was configured to run the Glucose 4 [30][7] solver.

### 6.5.1 Case Study 1: Random Forests

The QBF solvers we used are DepQBF [242][8] and CAQE [290][9]. Moreover, we combined CAQE with preprocessor Bloqqer [46][10]. We ran both QBF solvers with their default configurations. Moreover, all presented algorithms are implemented in Python and are publicly available in the repository [11].

**Benchmarks & Training.** The 27 datasets are split into two sets of datasets: the first one contains 9 small datasets that have at most 16 features (the average number of features is 8.4), and used to compare the performances of the two methods; the second one contain 18 datasets, with an average number of 21.5 features, which mainly serves to assess the CEGAR approach. The RF models are trained with varying the maximum depth from 4 to 6 and the number of trees from 20 to 100, so that we obtain the most accurate models. (These numbers are in line with RFs used in practice.) As a result, small RFs (i.e. with a number of trees less or equal 30) form the first set and the large RFs (with 100 trees) constitute the second benchmark set. For each dataset, a suite of 200 samples randomly picked is tested or all input data if there are less than 200 rows in the dataset. Moreover, the candidate feature set in the query is picked randomly for each test. (Hence, for each dataset, we generate 200 feature relevancy queries.) The time limit for deciding one query was set to 1200 seconds, and we capped the time for finishing 200 queries by 5 hours.

**Results: QBF versus CEGAR.** Table 6.5 summarizes the comparison results of QBF and CEGAR. Unsurprisingly, we observe that the resulting QBF encodings are somewhat larger that the SAT encodings. Indeed, the QBF formulation encodes two copies of the RF, i.e. $\mathcal{M}^0$ and $\mathcal{M}^t$, whereas the CEGAR encodes only one, i.e. $\mathcal{M}^t$. In addition, we note that encoding $\mathcal{M}^0$ requires $(K-1)$ cardinality constraints, therefore for a multi-class problem the encoding size of QBF can be larger than the CEGAR SAT encoding. Table 6.5 also shows the average running times of both approaches for solving one feature relevancy query. (Note that the reported average running times are computed on successful tests, and so the tests that time out are omitted.) Clearly, the results show that CEGAR outperforms QBF on all datasets. More importantly, the running times for CEGAR are most often negligible and at least one order of magnitude smaller than running times for QBF. Furthermore, we observe that in some datasets, QBF solvers were unable to terminate for some tests (e.g. 2 timeouts (resp. 1) for CAQE (resp. DepQBF) with *crx* dataset (resp. *glass2* dataset)) or all tests (e.g. DepQBF with *crx* dataset).

[6] https://github.com/pysathq/pysat
[7] https://www.labri.fr/perso/lsimon/glucose/
[8] https://github.com/lonsing/depqbf
[9] https://github.com/ltentrup/caqe
[10] http://fmv.jku.at/bloqqer/
[11] https://github.com/XuanxiangHuang/frpRF-experiments

**Results: Alternative QBF encoding.** Table 6.6 summarizes the results of the QBF encoding based on Equation (6.7). Compared with the results reported in Table 6.5, this method produces much larger encoding, the size of the encoding is roughly 3 times as large as that reported in Table 6.5. In terms of runtime, when running DepQBF on these QBF examples, it failed to solve all QBF queries for datasets *crx*, *ecoli*, *glass2*, *house_votes_84* and *new_thyroid*. For the rest datasets, the average runtime is one order of magnitude larger than the average runtime of running DepQBF on the QBF examples reported in Table 6.5, this indicates that the encoding based on (6.5) is more efficient than the encoding based on (6.7). When running CAQE on these QBF examples, the average runtime (except *crx*), is approximately 3 times that of the runtime reported in Table 6.5.

**Results: CEGAR.** Since the main goal is to assess the scalability of the CEGAR method on large RFs (of sizes common in practical applications), instances obtained from RFs with 100 trees (as described earlier) were also considered. The number of nodes in these RFs ranges from 1426 to 10176. The results are shown in Table 6.7. As can be observed, the average running times of CEGAR to decide one feature relevancy query take from 0.1s to 6.9s (resp. 0.1s to 62s) for outputs "Yes" (resp. "No"). It should be underscored that CEGAR computes in general more counterexamples to solve a negative decision (i.e. answer No), as this can be confirmed from the results, where the number of calls to the SAT oracle are substantially larger for decisions answered No on the majority of datasets (e.g. 11 261 calls for No against 285 for Yes, for *kr_vs_kp* dataset). As a result, and with a few exceptions, the running times of feature relevancy tests of output No are larger than tests answered Yes. Also, we emphasize that in contrast to the QBF solvers, no timeouts were observed with the CEGAR method, for the results shown in both tables. Additional results of the QBF method based on (6.5) (since this encoding is more efficient) on the second set of RFs are detailed in Table 6.8; as can be observed, the QBF method times out on the vast majority of the instances.

As can be concluded from the results, the CEGAR-based algorithm is effective in practice and usable on large size RFs induced from realistic datasets. Furthermore, the results also indicate that CEGAR substantially outperforms the encoding to QBF, being able to solve a vast number of feature relevancy queries that QBF solvers are unable to.

### 6.5.2 Case Study 2: d-DNNF Circuits

We consider SDDs (note that SDDs are a subset of d-DNNFs) as our target classifier. SDDs support polynomial time negation, so given a SDD $\mathcal{M}$, one can obtain its negation $\neg\mathcal{M}$ efficiently. All presented algorithms are implemented in Python and are publicly available in the repository [12].

[12] https://github.com/XuanxiangHuang/frp-experiment

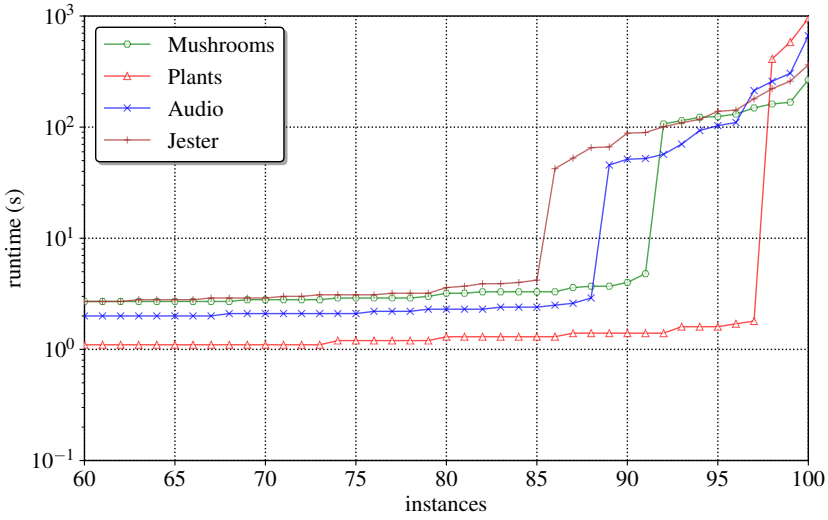



Figure 6.5: Running times of Audio, Jester, Mushrooms and Plants.

**Benchmarks & Compilation.** For SDDs, we selected 11 datasets from Density Estimation Benchmark Datasets[13]. [243, 151, 226]. 11 datasets were used to learn SDD via using LearnSDD[14] [38] (with parameter *maxEdges=20000*). The obtained SDDs were used as binary classifiers (albeit the selected circuits/datasets might not originally target classification tasks.) SDD models were loaded by using PySDD[15] package.

**Results of SDDs.** For each SDD classifier, 100 test instances were randomly generated. All tested instances have prediction 0. (We didn't pick instances predicted to class 1 as this requires the compilation of a new classifier which may have different size). Besides, for each instance, we randomly picked a feature appearing in the model. Hence for each SDD, we solved 100 queries. Note that for SDDs learned from LearnSDD, the reported number of features includes both original features and generated features (e.g. for *Audio* the original number of features is 100). Also note that PySDD offers canonical SDDs whose *conditioning* may take exponential time in the worst-case. Nevertheless, this worst-case behaviour was not observed in the experiments. Table 6.9 summarizes the obtained results of deciding feature relevancy on SDDs. It can be observed that the number of nodes of the tested SDD is in the range of 3704 and 9472, and the number of features of tested SDD is in the range of 183 and 513. Besides, the percentage of examples for which the answer is Y (i.e. target feature is in some AXp) ranges from 85% to 100%. Regarding the runtime, the largest running time for deciding one feature relevancy query can exceed 15 minutes. But the average running time to solve a query is less than 25 seconds, this highlights the scalability of the proposed encoding. However,

[13] https://github.com/UCLA-StarAI/Density-Estimation-Datasets
[14] https://github.com/ML-KULeuven/LearnSDD
[15] https://github.com/wannesm/PySDD

notice that for SDDs representing *Audio*, *Jester*, *Mushrooms* and *Plants*, the largest running time for deciding one feature relevancy query can exceed 3 minutes. As a result, we analyzed these results in greater detail. Figure 6.5 depicts a cactus plot showing the running time (in seconds) of deciding feature relevancy queries for these 4 datasets (note that the runtime axis is scaled logarithmically, and the instances axis starts from 60). As can be observed, for each of dataset, around 85 queries can be solved in a few seconds. This means that the running times of the method only exceeds a few seconds for a few concrete examples, and for a few of the datasets considered.

### 6.5.3 Case Study 3: Monotonic Classifiers

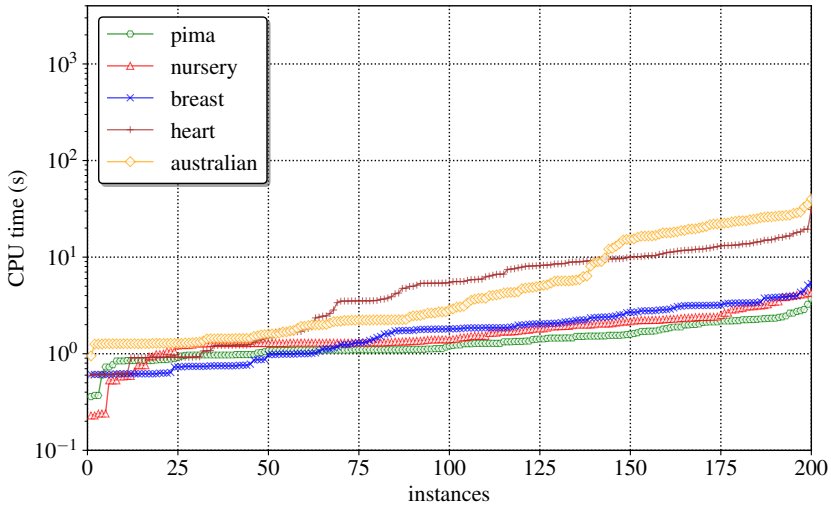



Figure 6.6: CPU time calling DLN's predict function.

We consider the Deep Lattice Network (DLN) [365] [16] as our target classifier. DLN is an architecture integrating linear embeddings, ensembles of lattices and calibrators. The trained networks are guaranteed to be monotonic with respect to a user-specified subset of the inputs. Since our approach is model-agnostic, it could also be used with other approaches for learning monotonic classifiers [352, 236] including Min-Max Network [324, 89] and COMET [328]. Moreover, all presented algorithms are implemented in Python and are publicly available in the repository [17].

**Benchmarks & Training.** We selected five publicly available datasets: *australian*, *breast_cancer*, *heart_c*, *nursery* [282] [18] and *pima* [9] [19]. In this case

[16] https://github.com/tensorflow/lattice
[17] https://github.com/XuanxiangHuang/frp-experiment
[18] https://epistasislab.github.io/pmlb/index.html
[19] https://sci2s.ugr.es/keel/dataset.php?cod=21

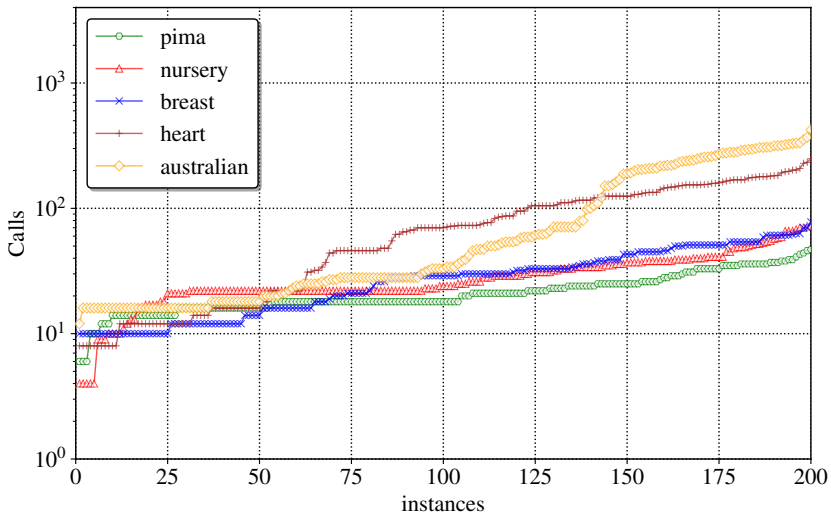



Figure 6.7: Number of calls to DLN's predict function.

study, we used the three-layer DLN architecture: Calibrators → Random Ensemble of Lattices → Linear Layer. All calibrators for all models used a fixed number of 20 keypoints. And the size of all lattices is set to 3. Table 6.10 shows the summary of trained DLN models. To evaluate the runtime performance (i.e. CPU time for predicting one datapoint) of the trained DLN, we ran trained DLNs on 10000 randomly picked datapoints from feature space, and report the maximum and average time (in seconds) for predicting one data point.

**Results.** Table 6.11 summarizes the experimental results. For each DLN model, we randomly picked 200 tested instances, and for each tested instance, we randomly pick a feature. Hence for each DLN, we solved 200 queries. The use of a SAT solver has a negligible contribution to the running time. Indeed, for all the examples shown, at least 97% of the running time is spent running the classifier. This should be unsurprising, since the number of the iterations of Algorithm 9 never exceeds a few hundred. (The fraction of a second reported in some cases should be divided by the number of calls to the SAT solver; hence the time spent in each call to the SAT solver is indeed negligible.) As can be observed, the percentage of examples for which the answer is Y (i.e. target feature is in some AXp and the algorithm returns **true**) ranges from 35% to 74%. There is no apparent correlation between the percentage of Y answers and the number of iterations. The large number of queries accounts for the number of times the DLN is queried by Algorithm 9, but it also accounts for the number of times the DLN is queried for extracting an AXp from set $\mathcal{X}$ (i.e. the witness) when the algorithm's answer is **true**. A loose upper bound on the number of queries to the classifier is $4 \times \text{NS} + 2 \times |\mathcal{F}|$, where NS is the number of SAT calls, and $|\mathcal{F}|$ is the number of features. Each iteration of Algorithm 9 can require at most 4 queries to the classifier. After reporting $\mathcal{X}$, at

most 2 queries per feature will be required to extract the AXp (see Section 2.3.3). As can be observed this loose upper bound is respected by the reported results.

Figure 6.6 depicts a cactus plot showing the accumulated CPU time (in seconds) calling DLN's *predict* function for deciding feature relevancy queries. Figure 6.7 depicts a cactus plot showing accumulated number of calls to DLN's *predict* function for deciding feature relevancy queries. It should be noted that the Y-axis is scaled logarithmically.

## 6.6 Summary

This chapter studies the problem of feature necessity and feature relevancy in logic-based explanations for several families of classifiers. The chapter starts by studying the complexity of both feature necessity and feature relevancy. Given the complexity gap between the two problems, most of the problem is dedicated to the feature relevancy. For the concrete case of feature relevancy, a number of membership and hardness results are proved. The chapter also devises a number of algorithms for deciding feature relevancy. This includes algorithms specific to each family of classifiers, but also general purpose algorithms, that can be used with any family of classifiers. The experimental results demonstrate that feature relevancy can be decided efficiently for a large range of families of classifiers. This observation also holds for the case of more complex classifiers, e.g. random forests.

A number of lines of research can be envisioned to continue this work. One line of work is to consider more sophisticated relevancy queries.

Table 6.5: Comparison of QBF and CEGAR methods. Columns $m$, $K$ report the characteristics of the dataset, namely number of features and classes, respectively. Sub-columns #N, A% in column RF report, resp., the number of nodes and the test accuracy of the trained models. Column Y% counts the number of feature relevancy queries answered 'Yes' (in percentage). Column QBF shows the average number of variables and clauses in the QBF encoding. Average runtimes for solving feature relevancy queries with QBF solvers are reported (in seconds) in columns DepQBF and CAQE, s.t. times of resulting queries answers Yes and No are reported separately. ("—" indicates that solver reached the fixed timeout for all tests; "*" indicates that timeouts are observed for some tests.) Column CNF shows the average number of variables and clauses of the SAT encoding generated by the CEGAR approach. The last column reports the average times (in seconds) of the CEGAR approach for solving one feature relevancy query, again times of answers Yes and No are reported separately.

| Dataset | $m$ | $K$ | RF | | Y% | QBF | | DepQBF | | CAQE | | CNF | | CEGAR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #N | A% | | vars | clauses | Yes | No | Yes | No | vars | clauses | Yes | No |
| crx | 15 | 2 | 522 | 81.8 | 90 | 2579 | 5260 | — | — | 12.1* | 36.70 | 1290 | 2719 | 0.03 | 0.05 |
| ecoli | 7 | 5 | 526 | 87.8 | 60 | 7448 | 12376 | 1.40 | 29.38 | 3.91 | 2.25 | 2917 | 5103 | 0.04 | 0.02 |
| glass2 | 9 | 2 | 348 | 84.8 | 87 | 2414 | 4744 | 18.5* | 48.38 | 1.26 | 1.73 | 1202 | 2440 | 0.01 | 0.01 |
| hayes_roth | 4 | 3 | 336 | 84.2 | 71 | 5036 | 7832 | 0.07 | 0.08 | 2.30 | 2.94 | 2555 | 4017 | 0.02 | 0.01 |
| votes_84 | 16 | 2 | 464 | 91.3 | 97 | 1266 | 2458 | 66.51 | 36.46 | 0.61 | 0.58 | 643 | 1256 | 0.01 | 0.05 |
| iris | 4 | 3 | 224 | 100 | 52 | 5079 | 8012 | 0.11 | 0.09 | 1.56 | 1.79 | 2564 | 4092 | 0.02 | 0.01 |
| mofn | 10 | 2 | 582 | 86.3 | 40 | 859 | 1851 | 0.03 | 0.04 | 0.22 | 0.28 | 440 | 931 | 0.01 | 0.01 |
| monk3 | 6 | 2 | 472 | 94.3 | 22 | 937 | 1868 | 0.02 | 0.03 | 0.16 | 0.20 | 473 | 937 | 0.01 | 0.00 |
| n_thyroid | 5 | 3 | 284 | 100 | 83 | 5722 | 9386 | 4.24 | 0.97 | 2.23 | 2.18 | 2884 | 4807 | 0.02 | 0.01 |

Table 6.6: Deciding feature relevancy results of alternative QBF encoding.

| Dataset | $m$ | $K$ | RF | | Y% | QBF | | DepQBF | | CAQE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #N | A% | | vars | clauses | Yes | No | Yes | No |
| crx | 15 | 2 | 522 | 81.8 | 90 | 7650 | 23670 | — | — | 7.85 | 93.39* |
| ecoli | 7 | 5 | 526 | 87.8 | 60 | 19680 | 54517 | — | — | 18.78 | 16.59 |
| glass2 | 9 | 2 | 348 | 84.8 | 87 | 7003 | 20988 | — | — | 4.44 | 10.89 |
| hayes_roth | 4 | 3 | 336 | 84.2 | 71 | 12846 | 34457 | 3.91 | 6.60 | 7.61 | 7.24 |
| votes_84 | 16 | 2 | 464 | 91.3 | 97 | 3645 | 11440 | — | — | 1.59 | 1.96 |
| iris | 4 | 3 | 224 | 100 | 52 | 13042 | 34969 | 27.93 | 40.92 | 7.02 | 6.70 |
| mofn | 10 | 2 | 582 | 86.3 | 40 | 2672 | 9397 | 2.72 | 36.35 | 0.84 | 0.89 |
| monk3 | 6 | 2 | 472 | 94.3 | 22 | 2778 | 9148 | 0.27 | 1.128 | 0.79 | 0.78 |
| n_thyroid | 5 | 3 | 284 | 100 | 83 | 15003 | 40979 | — | — | 8.77 | 9.67 |

Table 6.7: Assessing CEGAR approach on larger datasets and RFs trained with 100 trees. Column AXp reports the average size of computed AXps for queries answered Yes. Column Time reports average runtimes (in seconds) for solving one feature relevancy query. Column #SAT calls reports the average number of oracle guesses (counterexamples) performed in the CEGAR loop (i.e. number of iterations) to solve one feature relevancy query. The remaining columns have the same meaning as described in the caption of Table 6.5.

| Dataset | $m$ | $K$ | RF | | CNF | | Y% | AXp | Time | | #SAT calls | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | #N | A% | vars | clauses | | sz | Yes | No | Yes | No |
| agaricus_lepiota | 22 | 2 | 1866 | 99.2 | 3343 | 6310 | 89 | 10 | 0.2 | 4.7 | 52 | 2538 |
| allbp | 29 | 3 | 2492 | 96.5 | 16038 | 26452 | 47 | 4 | 2.6 | 4.3 | 65 | 261 |
| ann_thyroid | 21 | 3 | 2192 | 98.9 | 16802 | 27509 | 26 | 6 | 1.0 | 1.0 | 33 | 75 |
| appendicitis | 7 | 2 | 1426 | 90.9 | 4674 | 8736 | 97 | 4 | 0.1 | 0.1 | 4 | 20 |
| collins | 23 | 13 | 2890 | 86.6 | 24772 | 42186 | 95 | 12 | 3.4 | 0.4 | 38 | 16 |
| hypothyroid | 25 | 2 | 2034 | 95.9 | 4768 | 9347 | 53 | 4 | 0.4 | 1.3 | 32 | 324 |
| ionosphere | 34 | 2 | 1566 | 87.1 | 5922 | 12594 | 98 | 19 | 6.4 | 0.6 | 1272 | 232 |
| kr_vs_kp | 36 | 2 | 2268 | 94.2 | 2952 | 8102 | 71 | 11 | 0.6 | 20.5 | 285 | 11261 |
| magic | 10 | 2 | 2990 | 81.9 | 10631 | 22403 | 86 | 6 | 0.2 | 0.1 | 14 | 36 |
| mushroom | 22 | 2 | 2078 | 99.0 | 3374 | 6386 | 90 | 11 | 0.2 | 2.8 | 46 | 1375 |
| pendigits | 16 | 10 | 3098 | 85.0 | 22656 | 38420 | 99 | 10 | 1.6 | 1.5 | 18 | 70 |
| ring | 20 | 2 | 2458 | 84.1 | 9113 | 18815 | 68 | 15 | 0.2 | 0.5 | 20 | 130 |
| segmentation | 19 | 7 | 2288 | 92.8 | 20822 | 35114 | 91 | 9 | 1.6 | 4.5 | 45 | 290 |
| shuttle | 9 | 7 | 2618 | 99.8 | 19543 | 31942 | 78 | 4 | 0.9 | 0.9 | 14 | 31 |
| texture | 40 | 11 | 3040 | 81.4 | 27018 | 47325 | 97 | 23 | 6.9 | 62.0 | 210 | 5522 |
| twonorm | 20 | 2 | 3100 | 93.5 | 11729 | 24904 | 94 | 12 | 0.3 | 10.6 | 25 | 2606 |
| vowel | 13 | 11 | 10176 | 90.4 | 44530 | 88700 | 98 | 9 | 4.1 | 5.7 | 19 | 56 |
| waveform_21 | 21 | 3 | 3100 | 83.5 | 22446 | 39732 | 75 | 10 | 1.2 | 12.6 | 47 | 943 |

Table 6.8: QBF method solves queries for dataset *crx* and datasets in Table 6.5. The first row shows the timeout information for dataset *crx*. The rest show the timeout information for datasets in Table 6.7. #Test(TO) shows the number of feature relevancy queries tested in 5 hours, inside the parentheses is the number of timeout queries. If the query is solved, then its total and average time (in seconds) are reported in Column Yes Time and No Time. A '*' indicates out of time.

| Datasets | DepQBF | | | | | CAQE | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #Test(TO) | Yes Time | | No Time | | #Test(TO) | Yes Time | | No Time | |
| | | Total | Avg. | Total | Avg. | | Total | Avg. | Total | Avg. |
| crx | 21(14) | 614.33 | 122.87 | 68.43 | 68.43 | 200(2) | 2134.4 | 12.1 | 770.6 | 36.7 |
| agaricus_lepiota | 17(15) | 0.1 | 0.1 | 0.0 | 0.0 | 68(13) | 2138.7 | 42.8 | 36.4 | 9.1 |
| allbp | 17(15) | * | * | 0.0 | 0.0 | 17(15) | * | * | 0.0 | 0.0 |
| ann_thyroid | 23(14) | 1.9 | 0.6 | 731.5 | 146.3 | 49(13) | 1083.6 | 180.6 | 909.1 | 31.3 |
| appendicitis | 19(14) | 1095.0 | 273.7 | * | * | 27(13) | 1516.3 | 126.4 | 643.5 | 643.5 |
| collins | 16(15) | * | * | 0.0 | 0.0 | 16(14) | 1199.4 | 1199.4 | 0.0 | 0.0 |
| hypothyroid | 15(15) | * | * | * | * | 18(13) | 973.1 | 486.6 | 698.4 | 349.2 |
| ionosphere | 16(15) | 0.2 | 0.2 | * | * | 15(15) | * | * | * | * |
| kr_vs_kp | 21(14) | 814.9 | 163.0 | 0.0 | 0.0 | 29(13) | 1430.6 | 102.2 | 0.0 | 0.0 |
| magic | 22(14) | 638.2 | 91.2 | * | * | 20(13) | 1692.7 | 423.2 | 52.2 | 26.1 |
| mushroom | 18(14) | 1018.1 | 1018.1 | 0.0 | 0.0 | 37(12) | 3238.7 | 161.9 | 0.0 | 0.0 |
| pendigits | 16(15) | 1.8 | 1.8 | * | * | 17(14) | 647.9 | 323.9 | * | * |
| ring | 20(14) | 1103.3 | 220.7 | * | * | 72(14) | 707.6 | 14.7 | 573.6 | 57.4 |
| segmentation | 16(15) | 1.2 | 1.2 | * | * | 17(14) | 307.1 | 153.5 | * | * |
| shuttle | 16(15) | 1.2 | 1.2 | * | * | 24(12) | 2148.2 | 268.5 | 425.8 | 141.9 |
| texture | 15(15) | * | * | * | * | 15(15) | * | * | * | * |
| twonorm | 18(14) | 15.9 | 8.0 | 383.2 | 383.2 | 16(14) | * | * | 23.5 | 23.5 |
| vowel | 21(14) | 715.6 | 119.3 | * | * | 17(12) | 3500.2 | 875.1 | * | * |
| waveform_21 | 17(15) | 1.6 | 0.8 | * | * | 15(15) | * | * | * | * |

Table 6.9: Deciding feature relevancy for SDDs.

| Name | SDD | | Y% | CNF | | Runtime (s) | |
|---|---|---|---|---|---|---|---|
| | $m$ | #N | | vars | clauses | Max | Avg. |
| Accidents | 415 | 8863 | 97 | 26513 | 78276 | 56.4 | 3.5 |
| Audio | 272 | 7224 | 88 | 31148 | 100972 | 663.1 | 22.0 |
| DNA | 513 | 8570 | 91 | 29155 | 91288 | 86.3 | 11.0 |
| Jester | 254 | 7857 | 85 | 35998 | 121508 | 362.1 | 22.7 |
| KDD | 306 | 8109 | 99 | 26402 | 83480 | 111.2 | 2.8 |
| Mushrooms | 248 | 7096 | 91 | 23874 | 82112 | 266.3 | 15.8 |
| Netflix | 292 | 7039 | 94 | 25520 | 83324 | 105.7 | 4.2 |
| NLTCS | 183 | 6661 | 100 | 19817 | 58494 | 1.4 | 0.5 |
| Plants | 244 | 6724 | 97 | 25356 | 84782 | 950.7 | 20.6 |
| RCV-1 | 410 | 9472 | 90 | 33438 | 102500 | 153.6 | 11.2 |
| Retail | 341 | 3704 | 87 | 10601 | 28342 | 1.8 | 1.1 |

Table 6.10: Summary of DLN features

| DLN | Test Accuracy | #Parameters | predict($\mathbf{v}$) Time | |
|---|---|---|---|---|
| | | | Max | Avg. |
| australian (aus) | 88% | 775 | 0.74 | 0.08 |
| breast_cancer (b.c.) | 67% | 429 | 0.30 | 0.06 |
| heart_c | 73% | 755 | 0.67 | 0.08 |
| nursery | 76% | 415 | 0.32 | 0.06 |
| pima | 82% | 655 | 0.52 | 0.06 |

Table 6.11: Deciding feature relevancy queries for DLN. Sub-columns Max and Avg of column Runtime report, respectively, the maximum and average CPU time in seconds for deciding one feature relevancy query and extracting one AXp. Column SAT Time reports, maximum and average accumulated CPU time in seconds for SAT solver to decide one feature relevancy query. Column SAT Calls reports maximum and average accumulated number of calls to the SAT solver to decide one feature relevancy query. Column $\kappa(\mathbf{v})$ Time reports maximum and average accumulated CPU time in seconds calling DLN's predict function to decide one feature relevancy query. Column $\kappa(\mathbf{v})$ Calls reports the maximum and average accumulated number of calls to the DLN's predict function to decide one feature relevancy query.

| Name | Y% | Runtime | | SAT Time | | SAT Calls | | $\kappa(\mathbf{v})$ Time | | $\kappa(\mathbf{v})$ Calls | | $\frac{\kappa(\mathbf{v})\text{Time}}{\text{Runtime}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Max | Avg. | Max | Avg. | Max | Avg. | Max | Avg. | Max | Avg. | |
| aus | 61 | 40.4 | 8.31 | 0.02 | 0.01 | 291 | 65 | 40.0 | 8.15 | 424 | 98 | 97.8% |
| b.c. | 45 | 5.4 | 1.93 | 0.00 | 0.00 | 53 | 20 | 5.3 | 1.89 | 78 | 30 | 98.0% |
| heart_c | 35 | 31.5 | 6.67 | 0.02 | 0.00 | 171 | 54 | 31.1 | 6.52 | 249 | 80 | 97.7% |
| nursery | 45 | 4.3 | 1.77 | 0.00 | 0.00 | 31 | 13 | 4.3 | 1.75 | 73 | 30 | 98.6% |
| pima | 74 | 3.7 | 1.41 | 0.00 | 0.00 | 33 | 13 | 3.7 | 1.39 | 47 | 22 | 98.4% |

Chapter 7

# The Inadequacy of SHAP scores: Initial Results

Explainable Artificial Intelligence (XAI) is widely considered to be critical for building trust into the deployment of systems that integrate the use of machine learning (ML) models. For more than two decades Shapley values have been used as the theoretical underpinning for some methods of XAI, some of which now rank among the most widely used, including in high-risk domains.

This chapter proves that the use of the well-known SHAP scores for explainability can yield misleading information about the relative importance of features for predictions. This chapter identifies a number of ways in which misleading information can be conveyed to human decision makers, and proves that there exist classifiers which will yield such misleading information. Furthermore, this chapter offers empirical evidence that such theoretical limitations of SHAP scores are routinely observed in ML classifiers.

## 7.1 Introduction

Feature attribution is one of the most widely used approaches in machine learning (ML) explainability, begin implemented with a variety of different methods [332, 299, 309]. Moreover, the use of Shapley values [318] for feature attribution ranks among the most popular solutions [332, 333, 247, 84, 246], offering a widely accepted theoretical justification on how to assign importance to features in machine learning (ML) model predictions. One notable example is the tool SHAP [247], which endeavors to approximate the so-called SHAP scores—instantiations of Shapley values in the context of explainability. Despite the success of using SHAP scores for explainability, it is also the case that their exact computation is in general intractable [21, 22, 105, 106], with tractability results for some families of boolean circuits [21, 22]. Furthermore, the definition of SHAP scores (as well as its use in explainability) is purely axiomatic, i.e. there exists *no* formal proof that SHAP scores capture any specific properties related with explainability (even if defining such properties might prove elusive).

Feature selection represents a different alternative to feature attribution. The goal of feature selection is to select a set of features as representing the reason for a prediction, i.e. if the selected features take their assigned values, then the prediction cannot be changed. There are rigorous [253] and non-rigorous [300] approaches for selecting the features that explain a prediction. This chapter considers rigorous

(or model-precise) approaches for selecting such features. Furthermore, it should be plain that feature selection must aim for irredundancy, since otherwise it would suffice to report all features as the explanation. Given the universe of possible irreducible sets of feature selections that explain a prediction, the features that do not occur in *any* such set are deemed *irrelevant* [167] for a prediction; otherwise features that occur in one or more feature selections are deemed *relevant* [167].

Since both feature attribution and feature selection measure contributions of features to explanations, one would expect that the two approaches were related. However, this is not the case. We observed that SHAP scores could produce *misleading information* about features [173, 174, 175], in that irrelevant features (for feature selection) could be deemed more important (in terms of feature attribution) than relevant features (also for feature selection). Clearly, misleading information about the relative importance of features can easily induce human decision makers in error, by suggesting the *wrong* features as those to analyze in greater detail. Furthermore, situations where human decision makers can be misled are inadmissible in high-risk or safety-critical uses of ML. The existence in practice of those misleading issues with SHAP scores for explainability is evidently problematic for their use as the theoretical underpinning of feature attribution methods. This chapter proves that the identified misleading issues with SHAP scores for explainability exists for boolean functions with arbitrarily larger number of variables, and one can easily construct functions which exhibit the identified misleading issues. Empirically, this chapter studies a number of non-boolean classifiers, and shows that the conclusions of earlier work also apply to those non-boolean classifiers.

This chapter is organized as follows. In Section 7.2, we briefly review key concepts and introduce supplementary ones that are essential for this chapter. Section 7.3 relates the chapter's contributions with earlier work. Section 7.4 uncovers a number of possible issues that SHAP scores may exhibit, which would confirm the inadequacy of SHAP scores for explainability, and illustrates the existence of those issues in a number of motivating example boolean functions. Section 7.5 presents the chapter's main results, proving that all the issues with SHAP scores for explainability occur for boolean functions with arbitrarily larger number of variables. Section 7.6 summarizes identified issues with SHAP scores in a number of decision trees, d-DNNF circuits and OMDDs, all of which are either available in published works, or are learned from publicly available datasets. Section 7.7 discusses several potential threats to the validity of the results in this chapter. Section 7.8 concludes the chapter.

## 7.2 Preliminaries

The preliminary knowledge of this chapter consists of three parts. The first part, which covers *SHAP scores*, is elaborated on in Section 2.3. The second part, focusing on the concept of *feature relevancy* in formal XAI, can be found in Section 6.2. The third part establishes a connection between the CXp and *Adversarial Examples*.

We will use the predicate $\mathsf{Relevant}(i)$ to denote whether feature $i$ is relevant, and predicate $\mathsf{Irrelevant}(i)$ to denote whether feature $i$ is irrelevant.

Besides, for the purposes of this chapter, we consider solely a uniform input distribution, and so the dependency on the input distribution is not accounted for. A more general formulation is considered in related work [21, 20, 22, 106, 105]. However, assuming a uniform distribution suffices for the purposes of this chapter.

**How irrelevant are irrelevant features?** The fact that a feature is declared irrelevant for a local explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ is significant. Given the minimal hitting set duality between AXp and CXp, then an irrelevant features does not occur neither in any AXp, nor in any CXp. Furthermore, from the definition of AXp, each AXp for $\mathcal{E}$ can be represented as a logic rule. Let $\mathcal{R}$ denote the set of *all irreducible* logic rules which can be used to predict $c$, given the literals dictated by $\mathbf{v}$. Then, an irrelevant feature does not occur in *any* of those rules.

To further strengthen the above discussion, let us consider a (feature selection based) explanation $\mathcal{X} \subseteq \mathcal{F}$ such that $\mathsf{WAXp}(\mathcal{X})$ holds (i.e. (2.15) is true, and so $\mathcal{X}$ is sufficient for the prediction). Moreover, let $i \in \mathcal{F}$ be an irrelevant feature, such that $i \in \mathcal{X}$. Then, by definition of irrelevant feature, there *must* exist some $\mathcal{Z} \subseteq (\mathcal{X} \setminus \{i\})$, such that $\mathsf{WAXp}(\mathcal{Z})$ also holds (i.e. $\mathcal{Z}$ is *also* sufficient for the prediction). It is simple to understand why such set $\mathcal{Z}$ must exist. By definition of irrelevant feature, and because $i \in \mathcal{X}$, then $\mathcal{X}$ is not an AXp. However, there must exist an AXp $\mathcal{W} \subsetneq \mathcal{X}$ which, by definition of irrelevant feature, must not include $i$. Furthermore, and invoking Occam's razor[1], there is no reason to select $\mathcal{X}$ over $\mathcal{Z}$, and this remark applies to *any* set of features containing some irrelevant feature.

**Adversarial Examples vs (Ir)relevant Features.** Besides studying the relationship between SHAP scores and formal explanations, we also study their relationship with adversarial examples in ML models [144].

We use Hamming distance as a measure of distance between points in feature space. The Hamming distance is also referred to as the $l_0$ [2] measure of distance [163, 153], being defined as follows:

$$||\mathbf{x} - \mathbf{y}||_0 = \sum\nolimits_{i=1}^{m} \text{ITE}(x_i \neq y_i, 1, 0) \tag{7.1}$$

where $\text{ITE}(a, b, c)$ represents an IF-THEN-ELSE operator, with the semantics that the result is $b$ if the predicate $a$ is true, and it is $c$ if the predicate $a$ is false. Thus, the Hamming distance represents the number of different variables (or features) between two vectors $\mathbf{x}$ and $\mathbf{y}$.

Given a point $\mathbf{v}$ in feature space, an *adversarial example* (AE) is some other point $\mathbf{x}$ in feature space, such that the prediction on $\mathbf{x}$ differs from the prediction on $\mathbf{v}$, and such that $\mathbf{x}$ is close enough to $\mathbf{v}$, according to some measure of distance

[1] Here, we adopt a fairly standard definition of Occam's razor [50]: *given two explanations of the data, all other things being equal, the simpler explanation is preferable.*

[2] We also use $d_H(\cdot, \cdot)$ to denote the Hamming distance.

$l_p$. Formally,

$$\exists(\mathbf{x} \in \mathbb{F}).||\mathbf{x} - \mathbf{v}||_p \leq \epsilon \land (\kappa(\mathbf{x}) \neq \kappa(\mathbf{v})) \tag{7.2}$$

(in our case, we consider solely $p = 0$.) If (7.2) is true, then we say that the classifier has an $\epsilon$ adversarial example (or $\epsilon$-AE). Using $l_0$, then (7.2) states that, if we allow $\epsilon$ features to change value, then there exists some point in feature space (that differs from $\mathbf{v}$ in at most $\epsilon$ features) for which the prediction changes. The relationship between adversarial examples are explanations is well-known [188, 185].

**Proposition 24.** Given $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, $\mathcal{Y} \subseteq \mathcal{F}$ is a weak CXp iff $\mathcal{M}$ has an $\epsilon$-AE, with $l_0$ distance $\epsilon = |\mathcal{Y}|$.

*Proof.* If there exists one weak CXp $\mathcal{Y}$, this means that, if we allow the features in $\mathcal{Y}$ to change value, then the prediction changes values. Thus, if we allow $\epsilon = |\mathcal{Y}|$ features to change value, we are guaranteed to find an adversarial example. If there exists an $\epsilon$ adversarial example, then we construct $\mathcal{Y}$, with $|\mathcal{Y}| = \epsilon$ by picking the $\epsilon$ features that change their value with respect to $\mathbf{v}$ in the adversarial example. Thus, $\mathcal{Y}$ is a weak CXp. □

Furthermore, the following result relates adversarial examples with irrelevant features:

**Proposition 25.** If a classifier $\mathcal{M}$ on instance $(\mathbf{v}, c)$ has an adversarial example with $l_0$ distance $\delta$ that includes an irrelevant feature $j \in \mathcal{F}$, then there exists an adversarial example with $l_0$ distance $\delta - 1$ that does not include $j$.

*Proof.* From Proposition 24, we have that each $\epsilon$-AE maps to a weak CXp $\mathcal{Y}$ and vice-versa. So, if the $\delta$-AE includes and irrelevant feature $j$, then we can construct a weak CXp $\mathcal{W}$ that also includes feature $j$. However, since $j$ is irrelevant, then there must exist a (weak) CXp $\mathcal{Y}$, with $\mathcal{Y} = \mathcal{W} \setminus \{j\}$. Using again Proposition 24, it is the case that $\mathcal{Y}$ maps to some $\epsilon$-AE, with $\epsilon = \delta - 1$. □

Thus, irrelevant features are not included in subset- (or cardinality-) minimal adversarial examples.

## 7.3 Related Work

Shapley values for explainability is one of the hallmarks of feature attribution methods in XAI [332, 333, 101, 247, 68, 246, 262, 84, 127, 83, 126, 329, 203, 311, 349, 8, 51, 147, 7, 346]. Motivated by the success of Shapley values for explainability, there exists a burgeoning body of work on using Shapley values for explainability (e.g. [200, 367, 356, 191, 270, 34, 11, 376, 222, 6, 331, 375, 248, 348, 240, 241, 380, 132, 134, 166, 2]). Recent work studied the complexity of exactly computing SHAP scores [21, 22, 105, 106]. Finally, there have been proposals for the exact computation of SHAP scores in the case of circuit-based classifiers [21, 22]. Although there exist some differences in the proposals for the use of Shapley values for explainability, the basic formulation is the same and can be expressed as in Section 2.3.

A number of authors have reported pitfalls with the use of SHAP and Shapley values as a measure of feature importance [330, 366, 219, 334, 262, 128, 361, 273, 3, 350, 217, 64, 63]. However, these earlier works do not identify flaws with the use of SHAP scores in explainability, as we have identified in this chapter. Attempts at addressing those pitfalls include proposals to integrate SHAP scores with abductive explanations, as reported in recent work [221].

## Running Examples

Throughout this chapter, we will use the following boolean functions, which are represented by truth tables, as our running examples. The highlighted rows will serve as concrete examples throughout.

**Example 29.** We consider the example boolean functions of Figure 7.1. If the functions are represented by a truth table, then the SHAP scores can be computed in polynomial time on the size of the truth table, since the number of subsets considered in (2.12) is also polynomial on the size of the truth table [173]. (Observe that for each subset used in (2.12), we can use the truth table for computing the expected values in (2.8).) For example, for $\kappa_{I1}$ (see Figure 7.1a) and for the point in feature space $(0, 0, 1)$, one can compute the following SHAP scores: $\mathsf{SHAP}(1) = -0.417$, $\mathsf{SHAP}(2) = -0.042$, and $\mathsf{SHAP}(3) = 0.083$.

**Example 30.** Similar to the computation of SHAP scores, given a truth table representation of a function, and for a given instance, there is a polynomial-time algorithm for computing the AXp's [173]. For example, for function $\kappa_{I4}$ (see Figure 7.1d), and for the instance $((1, 1, 1, 1), 1)$, it can be observed that, if features 3 and 4 are allowed to take other values, the prediction remains at 1. Hence, $\{1, 2\}$ is a weak AXp, which is easy to conclude that it is also an AXp. When interpreted as a rule, the AXp would yield the rule:

$$\text{IF} \quad (x_1 = 1) \land (x_2 = 1) \quad \text{THEN} \quad \kappa(\mathbf{x}) = 1$$

In a similar way, if features 1 and 4 are allowed to take other values, the prediction remains at 0. Hence, $\{2, 3\}$ is another weak AXp (which can easily be shown to be an AXp). Furthermore, considering all other possible subsets of fixed features, allows us to conclude that there are no more AXp's.

## 7.4 Relating SHAP scores with Feature Relevancy

In this section, we list a number of issues that can be associated with SHAP scores. We consider relative feature importance, i.e. the ranking of features obtained by comparison of their absolute SHAP scores. Each issue captures a situation where SHAP scores provide misleading information about relative feature importance. By

| $x_1$ | $x_2$ | $x_3$ | $\kappa_{I1}(\mathbf{x})$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(a) Function $\kappa_{I1}$

| $x_1$ | $x_2$ | $x_3$ | $\kappa'_{I1}(\mathbf{x})$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

(b) Function $\kappa'_{I1}$

| $x_1$ | $x_2$ | $x_3$ | $\kappa_{I3}(\mathbf{x})$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(c) Function $\kappa_{I3}$

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $\kappa_{I4}(\mathbf{x})$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(d) Function $\kappa_{I4}$

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $\kappa'_{I4}(\mathbf{x})$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(e) Function $\kappa'_{I4}$

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $\kappa_{I5}(\mathbf{x})$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

(f) Function $\kappa_{I5}$

Figure 7.1: Example functions for the issues described in I1, I3, I4, and I5, respectively: $\kappa_{I1}$, $\kappa'_{I1}$, $\kappa_{I3}$, $\kappa_{I4}$, $\kappa'_{I4}$, and $\kappa_{I5}$.

*misleading*, this chapter signifies that SHAP scores either give undue high importance or undue low importance to some feature, when compared to some of the other feature(s), and given the relative importance of features.

**Issues with SHAP scores for explainability.** In this chapter, we consider the following main issues of SHAP scores for explainability:

**I1.** For a boolean classifier, with an instance $(\mathbf{v}, c)$, and feature $i$ such that,

$$\mathsf{Irrelevant}(i) \land (\mathsf{SHAP}(i) \neq 0)$$

Thus, an I1 issue is such that the feature is irrelevant, but its SHAP score is

non-zero.

**I2.** For a boolean classifier, with an instance $(\mathbf{v}, c)$ and features $i_1$ and $i_2$ such that,

$$\mathsf{Irrelevant}(i_1) \land \mathsf{Relevant}(i_2) \land (|\mathsf{SHAP}(i_1)| > |\mathsf{SHAP}(i_2)|)$$

Thus, an I2 issue is such that there is at least one irrelevant feature exhibiting a SHAP score larger (in absolute value) than the SHAP score of a relevant feature.

**I3.** For a boolean classifier, with instance $(\mathbf{v}, c)$, and feature $i$ such that,

$$\mathsf{Relevant}(i) \land (\mathsf{SHAP}(i) = 0)$$

Thus, an I3 issue is such that the feature is relevant, but its SHAP score is zero.

**I4.** For a boolean classifier, with instance $(\mathbf{v}, c)$, and features $i_1$ and $i_2$ such that,

$$[\mathsf{Irrelevant}(i_1) \land (\mathsf{SHAP}(i_1) \neq 0)] \land [\mathsf{Relevant}(i_2) \land (\mathsf{SHAP}(i_2) = 0)]$$

Thus, an I4 issue is such that there is at least one irrelevant feature with a non-zero SHAP score and a relevant feature with a SHAP score of 0.

**I5.** For a boolean classifier, with instance $(\mathbf{v}, c)$ and feature $i$ such that,

$$[\mathsf{Irrelevant}(i) \land \forall_{1 \leq j \leq m, j \neq i} (|\mathsf{SHAP}(j)| < |\mathsf{SHAP}(i)|)]$$

Thus, an I5 issue is such that there is one irrelevant feature exhibiting the highest SHAP score (in absolute value).

The issues above are all related with SHAP scores for explainability giving *misleading information* to a human decision maker, by assigning some importance to irrelevant features, by not assigning enough importance to relevant features, by assigning more importance to irrelevant features than to relevant features and, finally, by assigning the most importance to irrelevant features.

In the rest of the chapter we consider mostly I1, I3, I4 and I5 given that I5 implies I2.

**Examples exhibiting issues.** We study the example functions of Figure 7.1, which were derived from the main results of this chapter (see Section 7.5.2). These example functions will then be used to motivate the rationale for how those results are proved. In all cases, the reported SHAP scores are computed using the truth-table algorithm outlined in earlier work [173]. Similarly, the relevancy/irrelevancy claims of features use the truth-table algorithms outlined in earlier work [173].

**Example 31.** Figure 7.1a illustrates a boolean function that exhibits issue I1. By inspection, we can conclude that the function shown corresponds to $\kappa_{I1}(x_1, x_2, x_3) = (x_1 \land x_2 \land \neg x_3) \lor (x_1 \land x_3)$. Moreover, for the instance $((0, 0, 1), 0)$, Table 7.1 confirms that an issue I1 is identified.

Table 7.1: Examples of issues of SHAP scores for functions in Figure 7.1

| Case | Instance | Relevant | Irrelevant | SHAP's | Justification |
|---|---|---|---|---|---|
| I1 | $((0,0,1),0)$ | 1 | 2, 3 | $\mathsf{SHAP}(1) = -0.417$<br>$\mathsf{SHAP}(2) = -0.042$<br>$\mathsf{SHAP}(3) = 0.083$ | $\mathsf{Irrelevant}(3) \land \mathsf{SHAP}(3) > 0$ |
| I1 | $((1,1,1),0)$ | 1, 2 | 3 | $\mathsf{SHAP}(1) = -0.292$<br>$\mathsf{SHAP}(2) = -0.292$<br>$\mathsf{SHAP}(3) = -0.042$ | $\mathsf{Irrelevant}(3) \land \mathsf{SHAP}(3) < 0$ |
| I3 | $((1,1,1),1)$ | 1, 2, 3 | – | $\mathsf{SHAP}(1) = 0.125$<br>$\mathsf{SHAP}(2) = 0.375$<br>$\mathsf{SHAP}(3) = 0.000$ | $\mathsf{Relevant}(3) \land \mathsf{SHAP}(3) = 0$ |
| I4 | $((1,1,1,1),1)$ | 1, 2, 3 | 4 | $\mathsf{SHAP}(1) = 0.125$<br>$\mathsf{SHAP}(2) = 0.333$<br>$\mathsf{SHAP}(3) = 0.000$<br>$\mathsf{SHAP}(4) = -0.083$ | $\mathsf{Relevant}(3) \land \mathsf{SHAP}(3) = 0$<br>$\mathsf{Irrelevant}(4) \land \mathsf{SHAP}(4) < 0$ |
| I4 | $((1,1,1,1),1)$ | 1, 2, 3 | 4 | $\mathsf{SHAP}(1) = 0.042$<br>$\mathsf{SHAP}(2) = 0.292$<br>$\mathsf{SHAP}(3) = 0.000$<br>$\mathsf{SHAP}(4) = 0.042$ | $\mathsf{Relevant}(3) \land \mathsf{SHAP}(3) = 0$<br>$\mathsf{Irrelevant}(4) \land \mathsf{SHAP}(4) > 0$ |
| I5 | $((1,1,1,1),0)$ | 1, 2, 3 | 4 | $\mathsf{SHAP}(1) = -0.12$<br>$\mathsf{SHAP}(2) = -0.12$<br>$\mathsf{SHAP}(3) = -0.12$<br>$\mathsf{SHAP}(4) = 0.172$ | $\mathsf{Irrelevant}(4) \land$<br>$\forall(j \neq 4).\lvert\mathsf{SHAP}(j)\rvert < \mathsf{SHAP}(4)\rvert$ |

**Example 32.** Figure 7.1b illustrates a boolean function that exhibits issue I1. By inspection, we can conclude that the function shown corresponds to $\kappa'_{I1}(x_1, x_2, x_3) = (\neg x_1 \land \neg x_3) \lor (\neg x_2 \land \neg x_3) \lor (\neg x_1 \land x_2 \land x_3) \lor (x_1 \land \neg x_2 \land x_3)$. Moreover, for the instance $((1,1,1),0)$, Table 7.1 confirms that an issue I1 is identified.

**Example 33.** Figure 7.1c illustrates a boolean function that exhibits issue I3. By inspection, we can conclude that the function shown corresponds to $\kappa_{I3}(x_1, x_2, x_3) = (x_1 \land \neg x_3) \lor (x_2 \land x_3)$. Moreover, for the instance $((1,1,1),1)$, Table 7.1 confirms that an issue I3 is identified.

**Example 34.** Figure 7.1d illustrates a boolean function that exhibits issue I4. By inspection, we can conclude that the function shown corresponds to $\kappa_{I4}(x_1, x_2, x_3, x_4) = (x_1 \land \neg x_4) \lor (x_2 \land \neg x_4) \lor (x_1 \land \neg x_3 \land x_4) \lor (x_2 \land x_3 \land x_4)$. Moreover, for the instance $((1,1,1,1),1)$, Table 7.1 confirms that an issue I4 is identified.

**Example 35.** Figure 7.1e illustrates a boolean function that exhibits issue I4. By inspection, we can conclude that the function shown corresponds to $\kappa'_{I4}(x_1, x_2, x_3, x_4) = (x_1 \land \neg x_3 \land \neg x_4) \lor (x_2 \land x_3 \land \neg x_4) \lor (\neg x_1 \land \neg x_2 \land x_4) \lor (x_1 \land \neg x_3 \land x_4) \lor (x_2 \land x_3 \land x_4)$. Moreover, for the instance $((1,1,1,1),1)$, Table 7.1 confirms that an issue I4 is identified.

**Example 36.** Figure 7.1f illustrates a boolean function that exhibits issue I5. By inspection, we can conclude that the function shown corresponds to $\kappa_{I5}(x_1, x_2, x_3, x_4) = ((x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_3 \wedge \neg x_2) \vee (x_2 \wedge x_3 \wedge \neg x_1)) \wedge x_4$. Moreover, for the instance $((1, 1, 1, 1), 0)$, Table 7.1 confirms that an issue I5 is identified.

## 7.5 Issues with SHAP scores: Theory

This section proves that for arbitrary large numbers of variables, there exist boolean functions and instances for which the SHAP scores exhibit the issues detailed in Section 7.4. Throughout this section, let $m$ be the number of variables of the boolean functions we start from, and let $n$ denote the number of variables of the functions we will be constructing. In this case, we set $\mathcal{F} = \{1, \ldots, n\}$. Furthermore, for the sake of simplicity, we opt to introduce the new features as the last features (e.g., feature $n$). Besides, we opt to set the values of these additional features to 1 in the instance $(\mathbf{v}, c)$ that we intend to explain, that is, $v_n = 1$. This choice does not affect the proof's argument in any way.

We will outline a paradigm for computing SHAP scores for these additional features, then present a list of propositions exposing issues detailed in Section 7.4, each proposition will be followed by an example illustrating how to find boolean functions as listed in Figure 7.1. Furthermore, we will analyze a case study, as presented in Section 7.5.3, to illustrate the limitations of SHAP scores.

### 7.5.1 Proof Approaches

For a boolean function $\kappa$, we use $\kappa_0$ to denote the conditioning of the function $\kappa$ on $x_n = 0$ (i.e. $\kappa|_{x_n=0}$), and $\kappa_1$ to denote the conditioning of the function $\kappa$ on $x_n = 1$. Besides, we use $\kappa_{00}$ to denote the conditioning of the function $\kappa$ on $x_n = 0$ and $x_{n-1} = 0$ (i.e. $\kappa|_{x_n=0,x_{n-1}=0}$), $\kappa_{01}$ for the conditioning on $x_n = 0$ and $x_{n-1} = 1$, $\kappa_{10}$ for the conditioning on $x_n = 1$ and $x_{n-1} = 0$, and $\kappa_{11}$ for the conditioning on $x_n = 1$ and $x_{n-1} = 1$. In general, the following equations hold for $\mathbf{E}[\kappa]$ under a uniform input distribution:

$$\mathbf{E}[\kappa] = \frac{1}{2} \cdot \mathbf{E}[\kappa_0] + \frac{1}{2} \cdot \mathbf{E}[\kappa_1], \tag{7.3}$$

$$\mathbf{E}[\kappa] = \frac{1}{4} \cdot \mathbf{E}[\kappa_{00}] + \frac{1}{4} \cdot \mathbf{E}[\kappa_{01}] + \frac{1}{4} \cdot \mathbf{E}[\kappa_{10}] + \frac{1}{4} \cdot \mathbf{E}[\kappa_{11}]. \tag{7.4}$$

As stated previously, we assumed that the instance $(\mathbf{v}, c)$ to be explained satisfies either $v_n = 1$ when only one additional feature is considered, or $v_{n-1} = v_n = 1$ when two additional features are considered.

In the case where only feature $n$ is the additional feature. For feature $n$, consider

an arbitrary subset $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$, we can derive

$$\begin{aligned}&\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v})\\ &= \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n\}}}] - \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}}}]\\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_{\mathcal{S}}}]).\end{aligned} \tag{7.5}$$

In the case where feature $n$ and feature $n-1$ are additional features. For feature $n$, consider an arbitrary subset $\mathcal{S} \subseteq \mathcal{F} \setminus \{n-1, n\}$, we can derive

$$\begin{aligned}&\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v})\\ &= \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n\}}}] - \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}}}]\\ &= \frac{1}{4} \cdot (\mathbf{E}[\kappa_{10}|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_{00}|_{\mathbf{v}_{\mathcal{S}}}]) + \frac{1}{4} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_{01}|_{\mathbf{v}_{\mathcal{S}}}]).\end{aligned} \tag{7.6}$$

For feature $n$, consider an arbitrary subset $\mathcal{S} \subseteq \mathcal{F} \setminus \{n-1, n\}$ and feature $n-1$, we can derive

$$\begin{aligned}&\Delta(n, \mathcal{S} \cup \{n-1\}; \mathcal{M}, \mathbf{v})\\ &= \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n,n-1\}}}] - \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n-1\}}}]\\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_{01}|_{\mathbf{v}_{\mathcal{S}}}]).\end{aligned} \tag{7.7}$$

For feature $n-1$, consider an arbitrary subset $\mathcal{S} \subseteq \mathcal{F} \setminus \{n-1, n\}$, we can derive

$$\begin{aligned}&\Delta(n-1, \mathcal{S}; \mathcal{M}, \mathbf{v})\\ &= \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n-1\}}}] - \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}}}]\\ &= \frac{1}{4} \cdot (\mathbf{E}[\kappa_{01}|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_{00}|_{\mathbf{v}_{\mathcal{S}}}]) + \frac{1}{4} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_{10}|_{\mathbf{v}_{\mathcal{S}}}]).\end{aligned} \tag{7.8}$$

For feature $n-1$, consider an arbitrary subset $\mathcal{S} \subseteq \mathcal{F} \setminus \{n-1, n\}$ and feature $n$, we can derive

$$\begin{aligned}&\Delta(n-1, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v})\\ &= \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n,n-1\}}}] - \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{n\}}}]\\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_{\mathcal{S}}}] - \mathbf{E}[\kappa_{10}|_{\mathbf{v}_{\mathcal{S}}}]).\end{aligned} \tag{7.9}$$

By choosing different boolean functions for $\kappa_0$, $\kappa_1$, $\kappa_{00}$, $\kappa_{01}$, $\kappa_{10}$ and $\kappa_{11}$, we are able to construct boolean functions $\kappa$ exhibiting issues reported in Section 7.4.

**CXps, Hamming distance and Counterexamples.** Contrastive explanations are tightly related with counterexamples around the instance $(\mathbf{v}, c)$ we intend to explain. *Counterexamples* refer to any data point $\mathbf{v}'$ such that $\kappa(\mathbf{v}') \neq c$. Note that counterexamples differ from adversarial examples (see Equation (7.2)) because $\mathbf{v}'$ does not necessarily need to be close to $\mathbf{v}$. The size of a CXp measures the Hamming distance [153] between the point $\mathbf{v}$ and one of its counterexamples $\mathbf{v}'$. Let $d_H(\cdot, \cdot)$ denotes the Hamming distance [153] between two points in the feature

| $x_1$ | $x_2$ | $\kappa_1(\mathbf{x})$ | $\kappa_2(\mathbf{x})$ | $\kappa_3(\mathbf{x})$ |
|---|---|---|---|---|
| 0 | 0 | 2 | 3 | 0 |
| 0 | 1 | 2 | 2 | 0 |
| 0 | 2 | 3 | 0 | 3 |
| 1 | 0 | 0 | 1 | 2 |
| 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 0 | 3 |

(a) Three discrete functions $\kappa_1$, $\kappa_2$ and $\kappa_3$.

| $x_1$ | $x_2$ | $x_3$ | $\kappa(\mathbf{x})$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(b) A function $\kappa$.

Figure 7.2: Tabular representations for some example discrete classifiers demonstrating CXps, hamming distance and counterexamples.

space. When considering an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, where $\mathcal{M}$ is a discrete classifier [3], we can observe the following basic facts about CXps:

**Proposition 26.** For an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, if there exists a CXp with a size of $k$ then there is a counterexample $\mathbf{v}'$ in the feature space which has a Hamming distance of exactly $k$ from the point $\mathbf{v}$:

$$[\exists(\mathcal{Y} \in \mathbb{C}(\mathcal{E})).(|\mathcal{Y}| = k)] \rightarrow [\exists(\mathbf{v}' \in \mathbb{F}).(d_H(\mathbf{v}, \mathbf{v}') = k) \land (\kappa(\mathbf{v}') \neq c)] \tag{7.10}$$

**Remark 2.** The existence of a CXp $\mathcal{Y}$ with size $k$ only indicates the existence of a counterexample $\mathbf{v}'$ having a Hamming distance of $k$ from the data point $\mathbf{v}$. But we do not know how many counterexamples are covered by the CXp $\mathcal{Y}$.

**Remark 3.** For two discrete classifiers $\mathcal{M}_1$ and $\mathcal{M}_2$ defined on the same set of features. Let $\mathcal{E}_1 = (\mathcal{M}_1, (\mathbf{v}, c))$ and $\mathcal{E}_2 = (\mathcal{M}_2, (\mathbf{v}, c))$. If $\mathcal{Y} \in \mathbb{C}(\mathcal{E}_1) \cap \mathbb{C}(\mathcal{E}_2)$, then we can infer that $\exists(\mathbf{v}' \in \mathbb{F}).(d_H(\mathbf{v}, \mathbf{v}') = k) \land (\kappa_1(\mathbf{v}') \neq c)$ and $\exists(\mathbf{v}'' \in \mathbb{F}).(d_H(\mathbf{v}, \mathbf{v}'') = k) \land (\kappa_2(\mathbf{v}'') \neq c)$. It does not mean that $\mathbf{v}' = \mathbf{v}''$ or $\kappa_1(\mathbf{v}') = \kappa_2(\mathbf{v}'')$.

**Example 37.** Figure 7.2a shows three different discrete classifier $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ defined on the same set of features. For the instance $(\mathbf{v}, c) = ((1, 1), 1)$ and three explanation problems $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$, we have $\mathbb{C}(\mathcal{E}_1) = \mathbb{C}(\mathcal{E}_2) = \mathbb{C}(\mathcal{E}_3) = \{\{1\}, \{2\}\}$.

**Proposition 27.** For an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, if there is a counterexample $\mathbf{v}'$ in the feature space which has a Hamming distance of $k$ from the point $\mathbf{v}$, then there exists a CXp with a size of at most $k$:

$$[\exists(\mathbf{v}' \in \mathbb{F}).(d_H(\mathbf{v}, \mathbf{v}') = k) \land (\kappa(\mathbf{v}') \neq c)] \rightarrow [\exists(\mathcal{Y} \in \mathbb{C}(\mathcal{E})).(|\mathcal{Y}| \leq k)] \tag{7.11}$$

[3] If both the domains and the set of classes are ordinal (and discrete), then the classifier is referred to as *discrete*

*Proof.* The existence of a counterexample $\mathbf{v}'$ in the feature space which has a Hamming distance of $k$ from the point $\mathbf{v}$ implies that there exists a weak CXp with a size of $k$. Thus, there is an CXp with a size of at most $k$. □

**Corollary 8.** Given an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ such that $\forall \mathcal{Y} \in \mathbb{C}(\mathcal{E}).|\mathcal{Y}| < k$. For any counterexample $\mathbf{v}'$ such that $d_H(\mathbf{v}, \mathbf{v}') \geq k$, $\mathbf{v}'$ is not covered by any CXp.

**Example 38.** Figure 7.2b shows a classifier $\mathcal{M}$ and an explanation problem $\mathcal{E} = (\mathcal{M}, ((1,1,1),1))$ such that $\mathbb{C}(\mathcal{E}) = \{\{1\},\{3\}\}$. However, there is no CXp covering the counterexample $((0,0,1),0)$ and $((1,0,0),0)$. They are covered by the weak CXps $\{1,2\}$ and $\{2,3\}$ respectively.

Again, by choosing different boolean functions for $\kappa_0$, $\kappa_1$, $\kappa_{00}$, $\kappa_{01}$, $\kappa_{10}$ and $\kappa_{11}$, and taking into account the fundamental insights about about CXps, i.e. Propositions 26 and 27 and Corollary 8, we are able to construct boolean functions $\kappa$ exhibiting issues reported in Section 7.4.

### 7.5.2 Main Results for Boolean Classifiers

By utilizing Equation (7.5), Equation (7.6), Equation (7.7), Equation (7.8), Equation (7.9), Propositions 26 and 27 and Corollary 8, we present negative results pertaining to the SHAP scores of irrelevant features and relevant features. Specifically, we provide evidence that confirm the existence of the issues detailed in Section 7.4. When considering both feature $n$ and feature $n-1$, we will assume that the feature $n$ is irrelevant while the feature $n-1$ is relevant. In addition, for each result, we will illustration how to construct functions that exhibit these issues. We will use the example functions listed in Figure 7.1 as references.

In the following, we provide evidence that an irrelevant feature can have non-zero SHAP score. We will focus on instances $(\mathbf{v}, c)$ where $c = 0$. The case where $c = 1$ can be proven using similar techniques.

**Proposition 28.** For any $n \geq 3$, there exists boolean functions defined on $n$ variables and at least one instance $(\mathbf{v}, c)$ and an irrelevant feature $i \in \mathcal{F}$ such that $\mathsf{SHAP}(i) > 0$.

*Proof.* Let $\mathcal{M}$ be a classifier defined on the feature set $\mathcal{F}$ and characterized by the boolean function defined as follows:

$$\kappa(\mathbf{x}_{1..m}, x_n) := \begin{cases} \kappa_0(\mathbf{x}_{1..m}) & \text{if } x_n = 0 \\ \kappa_1(\mathbf{x}_{1..m}) & \text{if } x_n = 1 \end{cases} \tag{7.12}$$

The non-constant sub-functions $\kappa_0$ and $\kappa_1$ are defined on the feature set $\mathcal{F} \setminus \{n\}$, and satisfy the following conditions: 1) $\kappa_0 \models \kappa_1$ and 2) $\kappa_0 \neq \kappa_1$.

Choose a $n$-dimensional point $\mathbf{v}$ such that: 1) $v_n = 1$, and 2) $\kappa_0(\mathbf{v}) = \kappa_1(\mathbf{v}) = 0$, this means $\kappa(\mathbf{v}) = 0$. For any $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$, we have

$$\begin{aligned} &\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_\mathcal{S}}]). \end{aligned} \tag{7.13}$$

Given that $\kappa_0 \models \kappa_1$ but $\kappa_0 \neq \kappa_1$, it follows that for any points $\mathbf{x} \in \Upsilon(\mathcal{S}; \mathbf{v})$, if $\kappa_0(\mathbf{x}) = 1$ then $\kappa_1(\mathbf{x}) = 1$. In other words, if $\kappa_1(\mathbf{x}) = 0$ then $\kappa_0(\mathbf{x}) = 0$. Moreover, there are cases where the inequality holds $\mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_\mathcal{S}}] > 0$. Hence, $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) > 0$.

To prove that the feature $n$ is irrelevant, we assume the contrary that the feature $n$ is relevant, and $\mathcal{X}$, where $n \in \mathcal{X}$, is an AXp of the point $\mathbf{v}$. Based on the definition of AXp, we only include points $\mathbf{x}$ for which $\kappa_1(\mathbf{x}) = 0$ holds. However, as $\kappa_1(\mathbf{x}) = 0$ implies that $\kappa_0(\mathbf{x}) = 0$, $\mathcal{X} \setminus \{n\}$ will not include any points $\mathbf{x}$ such that either $\kappa_0(\mathbf{x}) = 1$ or $\kappa_1(\mathbf{x}) = 1$ holds. This means $\mathcal{X} \setminus \{n\}$ remains an AXp of the point $\mathbf{v}$, leading to a contradiction. Thus, feature $n$ is irrelevant. □

**Example 39.** Let $\kappa_0(x_1, x_2) = x_1 \land x_2$ and $\kappa_1(x_1, x_2) = x_1$. Clearly, we have $\kappa_0 \models \kappa_1$ and $\kappa_0 \neq \kappa_1$. Set $x_n = x_3$, and build $\kappa_{I1}(x_1, x_2, x_3) = (x_1 \land x_2 \land \neg x_3) \lor (x_1 \land x_3)$. Moreover, as shown in Example 31 and Table 7.1, it is the case that feature 3 is irrelevant but $\mathsf{SHAP}(3) > 0$.

**Proposition 29.** For any $n \geq 3$, there exists boolean functions defined on $n$ variables and at least one instance $(\mathbf{v}, c)$ and an irrelevant feature $i \in \mathcal{F}$ such that $\mathsf{SHAP}(i) < 0$.

*Proof.* Let $\mathcal{M}$ be a classifier defined on the feature set $\mathcal{F}$ and characterized by the boolean function defined as follows:

$$\kappa(\mathbf{x}_{1..m}, x_n) := \begin{cases} \kappa_1(\mathbf{x}_{1..m}) \lor f(\mathbf{x}_{1..m}) & \text{if } x_n = 0 \\ \kappa_1(\mathbf{x}_{1..m}) & \text{if } x_n = 1 \end{cases} \tag{7.14}$$

The non-constant sub-functions $\kappa_1$ and $f$ are defined on the feature set $\mathcal{F} \setminus \{n\}$, and satisfy the following conditions:

1. $\kappa_1 \neq \kappa_1 \lor f$ and $\kappa_1 \land f = 0$.
2. Both $\kappa_1$ and $\kappa_1 \lor f$ predict a specific point $\mathbf{v}_{1..m}$ to 0.
3. The set of CXps for both $\kappa_1$ and $\kappa_1 \lor f$ with respect to the point $\mathbf{v}_{1..m}$ are identical.

Choose this specific $m$-dimensional point $\mathbf{v}_{1..m}$ and extend it with $v_n = 1$. This

means $\kappa_0(\mathbf{v}) = \kappa_1(\mathbf{v}) = 0$, and therefore $\kappa(\mathbf{v}) = 0$. For any $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$, we have

$$\begin{aligned}
&\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\
&= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[(\kappa_1 \vee f)|_{\mathbf{v}_\mathcal{S}}]) \\
&= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[f|_{\mathbf{v}_\mathcal{S}}]) \\
&= \frac{1}{2} \cdot (-\mathbf{E}[f|_{\mathbf{v}_\mathcal{S}}]),
\end{aligned} \tag{7.15}$$

we can infer that $-\mathbf{E}[f|_{\mathbf{v}_\mathcal{S}}] < 0$ for some $\mathcal{S}$, which implies $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) < 0$.

To prove that the feature $n$ is irrelevant, we assume the contrary that the feature $n$ is relevant, and $\mathcal{X}$, where $n \in \mathcal{X}$, is an AXp of the point $\mathbf{v}$. Based on the definition of AXp, we only include points $\mathbf{x}$ for which $\kappa_1(\mathbf{x}) = 0$ holds. As $\kappa_1$ and $\kappa_1 \vee f$ share the same set of CXps, they have the same set of AXps. This means $\mathcal{X} \setminus \{n\}$ will not include any points $\mathbf{x}$ such that either $\kappa_1(\mathbf{x}) = 1$ or $(\kappa_1 \vee f)(\mathbf{x}) = 1$ holds. This means $\mathcal{X} \setminus \{n\}$ remains an AXp of the point $\mathbf{v}$, leading to a contradiction. Thus, feature $n$ is irrelevant. □

**Example 40.** Let $\kappa_1(x_1, x_2) = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$ and $f(x_1, x_2) = \neg x_1 \wedge \neg x_2$. Clearly $\kappa_1 \vee f = \neg x_1 \vee \neg x_2$, $\kappa_1 \neq \kappa_1 \vee f$ and $\kappa_1 \wedge f = 0$. Both $\kappa_1$ and $\kappa_1 \vee f$ predict the point $(1, 1)$ to 0. More importantly, the set of CXps for these two functions with respect to this point are identical $\{\{1\}, \{2\}\}$. Set $x_n = x_3$, and build $\kappa'_{I1}(x_1, x_2, x_3) = (\neg x_1 \wedge \neg x_3) \vee (\neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3)$. Moreover, as shown in Example 32 and Table 7.1, it is the case that feature 3 is irrelevant but $\mathsf{SHAP}(3) < 0$.

The above two propositions demonstrate that an irrelevant feature can have positive or negative SHAP scores, regardless of the class value of the given instance.

In the following, we provide evidence that a relevant feature can have zero SHAP score. We will focus on instances $(\mathbf{v}, c)$ where $c = 1$. The case where $c = 0$ can be proven using the same techniques.

**Proposition 30.** For any $n \geq 3$, there exists boolean functions defined on $n$ variables and at least one instance $(\mathbf{v}, c)$ and a relevant feature $i \in \mathcal{F}$ such that $\mathsf{SHAP}(i) = 0$.

*Proof.* Let $\mathcal{M}$ be a classifier defined on the feature set $\mathcal{F}$ and characterized by the boolean function defined as follows:

$$\kappa(\mathbf{x}_{1..m}, \mathbf{x}_{m+1..2m}, x_n) := \begin{cases} \kappa_0(\mathbf{x}_{1..m}) & \text{if } x_n = 0 \\ \kappa_1(\mathbf{x}_{m+1..2m}) & \text{if } x_n = 1 \end{cases} \tag{7.16}$$

The non-constant sub-functions $\kappa_0$ and $\kappa_1$ are defined on the feature sets $\mathcal{F}_0 = \{1, \ldots, m\}$ and $\mathcal{F}_1 = \{m+1, \ldots, 2m\}$, respectively. It is important to note that $\kappa_0$ is independent of $\kappa_1$ as $\mathcal{F}_0$ and $\mathcal{F}_1$ are disjoint. Moreover, $\kappa_0$ and $\kappa_1$ are identical

up to isomorphism. (For simplicity, we assume that the feature $i$ corresponds to the feature $m+i$ for all $i \in \{1,\ldots,m\}$.)

Choose a $n$-dimensional point $\mathbf{v}$ such that: 1) $v_n = 1$, 2) $v_i = v_{m+i}$ for any $1 \leq i \leq m$, and 3) $\kappa_0(\mathbf{v}) = \kappa_1(\mathbf{v}) = 1$. This means $\kappa(\mathbf{v}) = 1$. For any $\mathcal{S} \subset \mathcal{F} \setminus \{n\}$ such that $\mathcal{S} \neq \emptyset$, let $\{\mathcal{S}_0, \mathcal{S}_1\}$ be a partition of $\mathcal{S}$ such that $\mathcal{S}_0 \subseteq \mathcal{F}_0$ and $\mathcal{S}_1 \subseteq \mathcal{F}_1$, then

$$\begin{aligned} &\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}_1}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_{\mathcal{S}_0}}]). \end{aligned} \tag{7.17}$$

For any $\{\mathcal{S}_0, \mathcal{S}_1\}$, we can construct a unique new partition $\{\mathcal{S}_0', \mathcal{S}_1'\}$ by replacing any $i \in \mathcal{S}_0$ with $m+i$ and any $m+i \in \mathcal{S}_1$ with $i$. Let $\mathcal{S}' = \mathcal{S}_0' \cup \mathcal{S}_1'$, then we have

$$\begin{aligned} &\Delta(n, \mathcal{S}'; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}_0'}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_{\mathcal{S}_1'}}]). \end{aligned} \tag{7.18}$$

Besides, we have $\mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}_1}}] = \mathbf{E}[\kappa_0|_{\mathbf{v}_{\mathcal{S}_1'}}]$ and $\mathbf{E}[\kappa_0|_{\mathbf{v}_{\mathcal{S}_0}}] = \mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}_0'}}]$, which means

$$\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) = -\Delta(n, \mathcal{S}'; \mathcal{M}, \mathbf{v}), \tag{7.19}$$

note that $\varsigma(\mathcal{S}; \mathcal{M}, \mathbf{v}) = \varsigma(\mathcal{S}'; \mathcal{M}, \mathbf{v})$. Hence, for any $\mathcal{S} \subset \mathcal{F} \setminus \{n\}$ such that $\mathcal{S} \neq \emptyset$, there is a unique $\mathcal{S}'$ that can cancel its effect. Besides, if $\mathcal{S} = \emptyset$ or $\mathcal{S} = \mathcal{F} \setminus \{n\}$, then we have $\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) = 0$. We can derive that $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) = 0$. However, $n$ is a relevant feature. To prove this, it is evident that $\mathcal{F} \setminus \mathcal{F}_0$ represents a weak AXp. Moreover, $\mathcal{F} \setminus (\mathcal{F}_0 \cup \{n\})$ is not a weak AXp because allowing $x_n$ to take the value 0 will include points $\mathbf{x}$ such that $\kappa_0(\mathbf{x}) = 0$. Hence, there are AXps containing the feature $n$. □

**Example 41.** Let $\kappa_0(x_1) = x_1$ and $\kappa_1(x_2) = x_2$. Clearly $\kappa_0$ and $\kappa_1$ are defined on disjoint feature sets. Set $x_n = x_3$, and build $\kappa_{I3}(x_1, x_2, x_3) = (x_1 \land \neg x_3) \lor (x_2 \land x_3)$. Moreover, as shown in Example 33 and Table 7.1, it is the case that feature 3 is relevant but $\mathsf{SHAP}(3) = 0$.

Furthermore, we present evidence demonstrating the simultaneous occurrence of an irrelevant feature with a non-zero SHAP score and a relevant feature with a zero SHAP score. We will focus on instances $(\mathbf{v}, c)$ where $c = 1$. The case where $c = 0$ can be proven using similar techniques.

**Proposition 31.** For any $n \geq 4$, there exist boolean functions defined on $n$ variables, and at least one instance $(\mathbf{v}, c)$, for which there exists an irrelevant feature $i_1 \in \mathcal{F}$, such that $\mathsf{SHAP}(i_1) < 0$, and a relevant feature $i_2 \in \mathcal{F} \setminus \{i_1\}$, such that $\mathsf{SHAP}(i_2) = 0$.

*Proof.* Let $\mathcal{M}$ be a classifier defined on the feature set $\mathcal{F}$ and characterized by the

boolean function defined as follows:

$$\kappa(\mathbf{x}_{1..m}, \mathbf{x}_{m+1..2m}, x_{n-1}, x_n) := \begin{cases} \kappa_0(\mathbf{x}_{1..2m}) & \text{if } x_n = 0 \\ \kappa_{10}(\mathbf{x}_{1..m}) & \text{if } x_n = 1 \land x_{n-1} = 0 \\ \kappa_{11}(\mathbf{x}_{m+1..2m}) & \text{if } x_n = 1 \land x_{n-1} = 1 \end{cases} \tag{7.20}$$

The non-constant sub-functions $\kappa_0$, $\kappa_{10}$ and $\kappa_{11}$ are defined on the feature sets $\mathcal{F} \setminus \{n-1, n\}$, $\mathcal{F}_0 = \{1, \ldots, m\}$ and $\mathcal{F}_1 = \{m+1, \ldots, 2m\}$, respectively. It is worth noting that $\kappa_{10}$ is independent of $\kappa_{11}$ as $\mathcal{F}_0$ and $\mathcal{F}_1$ are disjoint. Also note that $\kappa_1 = (\neg x_{n-1} \land \kappa_{10}) \lor (x_{n-1} \land \kappa_{11})$. Moreover, $\kappa_0$, $\kappa_{10}$ and $\kappa_{11}$ satisfy the following conditions: 1) $\kappa_{10}$ and $\kappa_{11}$ are identical up to isomorphism, 2) $\kappa_1 \models \kappa_0$, and 3) $\kappa_0 \neq \kappa_1$. (For simplicity, we assume that the feature $i$ corresponds to the feature $m+i$ for all $i \in \{1, \ldots, m\}$.)

Choose a $n$-dimensional point $\mathbf{v}$ such that: 1) $v_n = 1$, 2) $v_i = v_{m+i}$ for any $1 \leq i \leq m$, and 3) $\kappa_{10}(\mathbf{v}) = \kappa_{11}(\mathbf{v}) = 1$. This implies $\kappa(\mathbf{v}) = 1$. For any $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$, we have

$$\begin{aligned} &\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_\mathcal{S}}]) \end{aligned} \tag{7.21}$$

Since $\kappa_1 \models \kappa_0$ but $\kappa_1 \neq \kappa_0$, we can apply the same reasoning as presented in the proof of Proposition 28 to deduce that $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) < 0$ even though feature $n$ is irrelevant. Next, we show that $\mathsf{SHAP}(n-1; \mathcal{M}, \mathbf{v}) = 0$ but the feature $n-1$ is relevant. For any $\mathcal{S} \subset \mathcal{F} \setminus \{n-1, n\}$ such that $\mathcal{S} \neq \emptyset$, we have

$$\begin{aligned} &\Delta(n-1, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\frac{1}{2} \cdot (\mathbf{E}[\kappa_{01}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{00}|_{\mathbf{v}_\mathcal{S}}]) + \frac{1}{2} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{10}|_{\mathbf{v}_\mathcal{S}}])) \\ &= \frac{1}{2} \cdot \frac{1}{2} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{10}|_{\mathbf{v}_\mathcal{S}}]) \end{aligned} \tag{7.22}$$

moreover, we have

$$\begin{aligned} &\Delta(n-1, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{10}|_{\mathbf{v}_\mathcal{S}}]) \end{aligned} \tag{7.23}$$

According to the proof of Proposition 30, for any $\mathcal{S}$ there is a unique $\mathcal{S}'$ such that $|\mathcal{S}| = |\mathcal{S}'|$ to cancel the effect of $\mathcal{S}$. Thus, $\mathsf{SHAP}(n-1; \mathcal{M}, \mathbf{v}) = 0$. Again, applying the same reasoning presented in the proof of Proposition 30, we can infer that $\mathcal{F} \setminus (\mathcal{F}_0 \cup \{n\})$ is a weak AXp but $\mathcal{F} \setminus (\mathcal{F}_0 \cup \{n-1, n\})$ is not a weak AXp. Thus, we can conclude that the feature $n-1$ is relevant. □

**Example 42.** Let $\kappa_0(x_1, x_2) = x_1 \lor x_2$, $\kappa_{10}(x_1) = x_1$ and $\kappa_{11}(x_2) = x_2$. Clearly, $\kappa_{10}$ and $\kappa_{11}$ are defined on disjoint feature sets. Set $x_{n-1} = x_3$, then $\kappa_1(x_1, x_2) = (\neg x_3 \land x_1) \lor (x_3 \land x_2)$ and we have $\kappa_1 \models \kappa_0$ and $\kappa_0 \neq \kappa_1$. Set $x_n = x_4$, and build $\kappa_{I4}(x_1, x_2, x_3, x_4) = (x_1 \land \neg x_4) \lor (x_2 \land \neg x_4) \lor (x_1 \land \neg x_3 \land x_4) \lor (x_2 \land x_3 \land x_4)$. Moreover, as shown in Example 34 and Table 7.1, it is the case that feature 3 is

relevant but $\mathsf{SHAP}(3) = 0$ and feature 4 is irrelevant but $\mathsf{SHAP}(4) < 0$.

**Proposition 32.** For any $n \geq 4$, there exist boolean functions defined on $n$ variables, and at least one instance $(\mathbf{v}, c)$, for which there exists an irrelevant feature $i_1 \in \mathcal{F}$, such that $\mathsf{SHAP}(i_1) > 0$, and a relevant feature $i_2 \in \mathcal{F} \setminus \{i_1\}$, such that $\mathsf{SHAP}(i_2) = 0$.

*Proof.* Let $\mathcal{M}$ be a classifier defined on the feature set $\mathcal{F}$ and characterized by the boolean function defined as follows:

$$\kappa(\mathbf{x}_{1..m}, \mathbf{x}_{m+1..2m}, x_{n-1}, x_n) := \begin{cases} \kappa_{00}(\mathbf{x}_{1..m}) & \text{if } x_n = 0 \land x_{n-1} = 0 \\ \kappa_{01}(\mathbf{x}_{m+1..2m}) & \text{if } x_n = 0 \land x_{n-1} = 1 \\ \kappa_0(\mathbf{x}_{1..2m, x_{n-1}}) \lor f(\mathbf{x}_{1..2m}) & \text{if } x_n = 1 \end{cases} \tag{7.24}$$

The non-constant sub-functions $\kappa_{00}$, $\kappa_{01}$ and $f$ are defined on the feature sets $\mathcal{F}_0 = \{1, \ldots, m\}$, $\mathcal{F}_1 = \{m+1, \ldots, 2m\}$, and $\mathcal{F} \setminus \{n-1, n\}$, respectively. It is worth noting that $\kappa_{00}$ is independent of $\kappa_{01}$ as $\mathcal{F}_0$ and $\mathcal{F}_1$ are disjoint. Also note that $\kappa_1 = \kappa_0 \lor f$. Moreover, $\kappa_{00}$, $\kappa_{01}$ and $f$ satisfy the following conditions:

1. $\kappa_{00}$ and $\kappa_{01}$ are identical up to isomorphism. (For simplicity, we assume that the feature $i$ corresponds to the feature $m + i$ for all $i \in \{1, \ldots, m\}$.)
2. $\kappa_0 \neq \kappa_0 \lor f$, $\kappa_{00} \land f = 0$ and $\kappa_{01} \land f = 0$.
3. Both $\kappa_0$ and $\kappa_0 \lor f$ predict a specific point $\mathbf{v}_{1..n-1}$ to 1, where $v_{n-1} = 1$, and $v_i = v_{m+i}$ for any $1 \leq i \leq m$.
4. The set of CXps for $\kappa_0$ and $\kappa_0 \lor f$ with respect to the point $\mathbf{v}_{1..n-1}$ are identical.

Choose this specific $n-1$-dimensional point $\mathbf{v}_{1..n-1}$ and extend it with $v_n = 1$, then $\kappa_0(\mathbf{v}) = \kappa_1(\mathbf{v}) = 1$ and $\kappa(\mathbf{v}) = 1$. For any $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$, we have

$$\begin{aligned} &\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\mathbf{E}[(\kappa_0 \lor f)|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_0|_{\mathbf{v}_\mathcal{S}}]) \\ &= \frac{1}{2} \cdot (\mathbf{E}[f|_{\mathbf{v}_\mathcal{S}}]), \end{aligned} \tag{7.25}$$

which implies $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) > 0$. As $\kappa_0$ and $\kappa_0 \lor f$ share the same set of CXps, they have the same set of AXps. By applying similar reasoning as presented in the proof of Proposition 29, we can conclude that feature $n$ is irrelevant. Next, we show that $\mathsf{SHAP}(n-1; \mathcal{M}, \mathbf{v}) = 0$ but the feature $n-1$ is relevant. For any $\mathcal{S} \subset \mathcal{F} \setminus \{n-1, n\}$ such that $\mathcal{S} \neq \emptyset$, we have

$$\begin{aligned} &\Delta(n-1, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot (\frac{1}{2} \cdot (\mathbf{E}[\kappa_{01}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{00}|_{\mathbf{v}_\mathcal{S}}]) + \frac{1}{2} \cdot (\mathbf{E}[\kappa_{11}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{10}|_{\mathbf{v}_\mathcal{S}}])) \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_{01}|_{\mathbf{v}_\mathcal{S}}] - \mathbf{E}[\kappa_{00}|_{\mathbf{v}_\mathcal{S}}]), \end{aligned} \tag{7.26}$$

besides, we have

$$\begin{aligned}&\Delta(n-1,\mathcal{S}\cup\{n\};\mathcal{M},\mathbf{v})\\&=\frac{1}{2}\cdot(\mathbf{E}[\kappa_{11}|_{\mathbf{v}_\mathcal{S}}]-\mathbf{E}[\kappa_{10}|_{\mathbf{v}_\mathcal{S}}])\\&=\frac{1}{2}\cdot(\mathbf{E}[\kappa_{01}|_{\mathbf{v}_\mathcal{S}}]-\mathbf{E}[\kappa_{00}|_{\mathbf{v}_\mathcal{S}}]),\end{aligned}\tag{7.27}$$

also note that $\Delta(n-1,\mathcal{S};\mathcal{M},\mathbf{v})=0$ when $\mathcal{S}=\emptyset$ or $\mathcal{S}=\mathcal{F}\setminus\{n-1\}$. By applying the same reasoning as presented in the proof of Proposition 31, we can conclude that $\mathsf{SHAP}(n-1;\mathcal{M},\mathbf{v})=0$ but the feature $n-1$ is relevant. □

**Example 43.** Let $\kappa_{00}(x_1)=x_1$, $\kappa_{01}(x_2)=x_2$ and $f(x_1,x_2)=\neg x_1\wedge\neg x_2$. Clearly, $\kappa_{00}$ and $\kappa_{01}$ are defined on disjoint feature sets. Furthermore, we have $\kappa_{00}\wedge f=0$ and $\kappa_{01}\wedge f=0$. Set $x_{n-1}=x_3$, we have $\kappa_0(x_1,x_2)=(\neg x_3\wedge x_1)\vee(x_3\wedge x_2)$. Moreover, we have $\kappa_0\neq\kappa_0\vee f$. Both $\kappa_0$ and $\kappa_1$ predict the point $(1,1,1)$ to 1, and the set of CXps of these two functions with respect to this point are the same $\{\{2\},\{1,3\}\}$. Set $x_n=x_4$, and build $\kappa'_{I4}(x_1,x_2,x_3,x_4)=(x_1\wedge\neg x_3\wedge\neg x_4)\vee(x_2\wedge x_3\wedge\neg x_4)\vee(\neg x_1\wedge\neg x_2\wedge x_4)\vee(x_1\wedge\neg x_3\wedge x_4)\vee(x_2\wedge x_3\wedge x_4)$. Moreover, as shown in Example 35 and Table 7.1, it is the case that feature 3 is relevant but $\mathsf{SHAP}(3)=0$ and feature 4 is irrelevant but $\mathsf{SHAP}(4)>0$.

In the following, we prove that irrelevant features can have highest SHAP score (in absolute value). We will focus on instances $(\mathbf{v},c)$ where $c=0$. The case where $c=1$ can be proven using similar techniques.

**Proposition 33.** For any $n\geq 4$, there exists boolean functions defined on $n$ variables, and at least one instance, for which there exists an irrelevant feature $i\in\mathcal{F}=\{1,\ldots,n\}$, such that $|\mathsf{SHAP}(i)|=\max\{|\mathsf{SHAP}(j)|\,|\,j\in\mathcal{F}\}$.

*Proof.* Let $\mathcal{M}$ be a classifier defined on the feature set $\mathcal{F}$ and characterized by the boolean function defined as follows:

$$\kappa(\mathbf{x}_{1..m},x_n):=\begin{cases}0 & \text{if } x_n=0\\ \kappa_1(\mathbf{x}_{1..m}) & \text{if } x_n=1\end{cases}\tag{7.28}$$

Its sub-function $\kappa_1$ is a non-constant boolean function defined on the feature set $\mathcal{F}\setminus\{n\}$, and satisfies the following conditions:

1. $\kappa_1$ predicts a specific point $\mathbf{v}_{1..m}$ to 0.
2. For any point $\mathbf{x}_{1..m}$ such that $||\mathbf{x}_{1..m}-\mathbf{v}_{1..m}||_0=1$, we have $\kappa_1(\mathbf{x}_{1..m})=1$.
3. $\kappa_1$ predicts all the other points to 0.

For example, $\kappa_1$ can be the function $\sum_{i=1}^{m}\neg x_i=1$, which predicts the point $\mathbf{1}_{1..m}$ to 0 and all points around this point with a Hamming distance of 1 to 1.

Select this specific $m$-dimensional point $\mathbf{v}_{1..m}$ such that $\kappa_1(\mathbf{v}_{1..m})=0$. Extend $\mathbf{v}_{1..m}$ with $v_n=1$, we have $\kappa(\mathbf{v})=0$. Applying the same reasoning presented in the

proof of Proposition 28, we can deduce that the feature $n$ is irrelevant. In addition, for $\kappa_1$ and any $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$, we have

$$\mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] = \frac{m - |\mathcal{S}|}{2^{m-|\mathcal{S}|}}. \tag{7.29}$$

For feature $n$ and an arbitrary $\mathcal{S} \subseteq \mathcal{F}$, we have

$$\begin{aligned} &\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \frac{1}{2} \cdot \mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] \\ &= \frac{1}{2} \cdot \frac{m - |\mathcal{S}|}{2^{m-|\mathcal{S}|}}, \end{aligned} \tag{7.30}$$

this means $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) > 0$. Besides, the unique minimal value of $\Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v})$ is 0 when $\mathcal{S} = \mathcal{F} \setminus \{n\}$.

We now focus on a feature $j \neq n$. Consider an arbitrary $\mathcal{S} \subseteq \mathcal{F} \setminus \{j, n\}$, we have

$$\begin{aligned} &\Delta(j, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}) \\ &= \mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}\cup\{j\}}}] - \mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}] \\ &= \frac{m - |\mathcal{S}| - 1}{2^{m-|\mathcal{S}|-1}} - \frac{m - |\mathcal{S}|}{2^{m-|\mathcal{S}|}} \\ &= \frac{m - |\mathcal{S}| - 2}{2^{m-|\mathcal{S}|}}. \end{aligned} \tag{7.31}$$

In this case, $\Delta(j, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}) = -\frac{1}{2}$ if $|\mathcal{S}| = m - 1$, which is its unique minimal value. $\Delta(j, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}) = 0$ if $|\mathcal{S}| = m - 2$, and $\Delta(j, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}) > 0$ if $|\mathcal{S}| < m - 2$. Besides, we have

$$\begin{aligned} &\Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}) \\ &= \mathbf{E}[\kappa|_{\mathbf{v}_{\mathcal{S}\cup\{j\}}}] - \mathbf{E}[\kappa|_{\mathbf{v}_\mathcal{S}}] \\ &= \frac{1}{2} \cdot (\mathbf{E}[\kappa_1|_{\mathbf{v}_{\mathcal{S}\cup\{j\}}}] - \mathbf{E}[\kappa_1|_{\mathbf{v}_\mathcal{S}}]) \\ &= \frac{1}{2} \cdot \frac{m - |\mathcal{S}| - 2}{2^{m-|\mathcal{S}|}}. \end{aligned} \tag{7.32}$$

In this case, $\Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}) = -\frac{1}{4}$ if $|\mathcal{S}| = m - 1$, which is its unique minimal value. $\Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}) = 0$ if $|\mathcal{S}| = m - 2$, and $\Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}) > 0$ if $|\mathcal{S}| < m - 2$.

Next, we prove $|\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v})| > |\mathsf{SHAP}(j; \mathcal{M}, \mathbf{v})|$ by showing $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) + \mathsf{SHAP}(j; \mathcal{M}, \mathbf{v}) > 0$ and $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) - \mathsf{SHAP}(j; \mathcal{M}, \mathbf{v}) > 0$. Note that $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) > 0$. Additionally, $\Delta(j, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}) < 0$ and $\Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}) < 0$ only when $|\mathcal{S}| =$

$m-1$. Compute the SHAP score for feature $n$:

$$\begin{aligned}
\mathsf{SHAP}(n;\mathcal{M},\mathbf{v}) &= \sum_{\mathcal{S}\subseteq\mathcal{F}\setminus\{n\}} \frac{|\mathcal{S}|!(m-|\mathcal{S}|)!}{(m+1)!}\cdot\Delta(n,\mathcal{S};\mathcal{M},\mathbf{v}) \\
&= \sum_{\mathcal{S}\subseteq\mathcal{F}\setminus\{n\}} \frac{|\mathcal{S}|!(m-|\mathcal{S}|)!}{(m+1)!}\cdot\frac{1}{2}\cdot\frac{m-|\mathcal{S}|}{2^{m-|\mathcal{S}|}} \\
&= \frac{1}{2}\cdot\frac{1}{m+1}\cdot\sum_{\mathcal{S}\subseteq\mathcal{F}\setminus\{n\}} \frac{|\mathcal{S}|!(m-|\mathcal{S}|)!}{m!}\cdot\frac{m-|\mathcal{S}|}{2^{m-|\mathcal{S}|}} \\
&= \frac{1}{2}\cdot\frac{1}{m+1}\cdot\sum_{0\leq|\mathcal{S}|\leq m} \frac{|\mathcal{S}|!(m-|\mathcal{S}|)!}{m!}\cdot\frac{m!}{|\mathcal{S}|!(m-|\mathcal{S}|)!}\cdot\frac{m-|\mathcal{S}|}{2^{m-|\mathcal{S}|}} \\
&= \frac{1}{2}\cdot\frac{1}{m+1}\cdot\sum_{k=1}^{m}\frac{k}{2^k} \\
&= \frac{1}{2}\cdot\frac{1}{m+1}\cdot\frac{2^{m+1}-m-2}{2^m} \\
&= \frac{1}{m+1}\cdot\frac{2^{m+1}-m-2}{2^{m+1}}.
\end{aligned} \tag{7.33}$$

Now we focus on a feature $j\neq n$. Consider the subset $\mathcal{S}=\mathcal{F}\setminus\{j,n\}$ where $|\mathcal{S}|=m-1$, we have

$$\begin{aligned}
&\frac{|\mathcal{S}\cup\{n\}|!(m-|\mathcal{S}\cup\{n\}|)!}{(m+1)!}\cdot\frac{m-|\mathcal{S}|-2}{2^{m-|\mathcal{S}|}} \\
&=-\frac{1}{2}\cdot\frac{1}{m+1},
\end{aligned} \tag{7.34}$$

moreover, we have

$$\begin{aligned}
&\frac{|\mathcal{S}|!(m-|\mathcal{S}|)!}{(m+1)!}\cdot\frac{1}{2}\cdot\frac{m-|\mathcal{S}|-2}{2^{m-|\mathcal{S}|}} \\
&=-\frac{1}{4}\cdot\frac{1}{m(m+1)}.
\end{aligned} \tag{7.35}$$

The sum of these three values is

$$\begin{aligned}
&\frac{1}{m+1}\cdot\frac{2^{m+1}-m-2}{2^{m+1}}-\frac{1}{2}\cdot\frac{1}{m+1}-\frac{1}{4}\cdot\frac{1}{m(m+1)} \\
&=\frac{1}{m+1}\cdot\left(\frac{(2^{m+1}-m-2)m}{m2^{m+1}}-\frac{m2^m}{m2^{m+1}}-\frac{2^{m-1}}{m2^{m+1}}\right) \\
&=\frac{1}{m(m+1)2^{m+1}}\cdot\left((m-\frac{1}{2})2^m-m^2-2m\right),
\end{aligned} \tag{7.36}$$

since $m\geq 3$, the sum of these three values is always greater than 0. Thus, we can conclude that $\mathsf{SHAP}(n;\mathcal{M},\mathbf{v})+\mathsf{SHAP}(j;\mathcal{M},\mathbf{v})>0$.

To show $\mathsf{SHAP}(n;\mathcal{M},\mathbf{v})-\mathsf{SHAP}(j;\mathcal{M},\mathbf{v})>0$, we focus on all $\mathcal{S}\subseteq\mathcal{F}\setminus\{n\}$ where $|\mathcal{S}|<m-2$. This is because, as previously stated, $\Delta(j,\mathcal{S}\cup\{n\};\mathcal{M},\mathbf{v})\leq 0$

and $\Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}) \leq 0$ if $|\mathcal{S}| \geq m - 2$.

Moreover, for all $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$ where $|\mathcal{S}| = k$ and $0 < k \leq m - 3$, we compute the following three quantities:

$$\begin{aligned} Q_1 &:= \sum_{\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}, |\mathcal{S}| = k} \Delta(n, \mathcal{S}; \mathcal{M}, \mathbf{v}), \\ Q_2 &:= \sum_{\mathcal{S} \subseteq \mathcal{F} \setminus \{j,n\}, |\mathcal{S}| = k-1} \Delta(j, \mathcal{S} \cup \{n\}; \mathcal{M}, \mathbf{v}), \\ Q_3 &:= \sum_{\mathcal{S} \subseteq \mathcal{F} \setminus \{j,n\}, |\mathcal{S}| = k} \Delta(j, \mathcal{S}; \mathcal{M}, \mathbf{v}), \end{aligned} \tag{7.37}$$

and show that $Q_1 - Q_2 - Q_3 > 0$. Note that $Q_1$, $Q_2$ and $Q_3$ share the same coefficient $\frac{k!(n-k-1)!}{n!}$. For feature $n$, we pick all possible $\mathcal{S} \subseteq \mathcal{F} \setminus \{n\}$ where $|\mathcal{S}| = k$, which implies $|\mathcal{S} \cup \{n\}| = k + 1$, then

$$Q_1 = \binom{m}{|\mathcal{S}|} \cdot \frac{1}{2} \cdot \frac{m - |\mathcal{S}|}{2^{m-|\mathcal{S}|}} = \binom{m}{k} \cdot \frac{1}{2} \cdot \frac{m-k}{2^{m-k}}. \tag{7.38}$$

For a feature $j \neq n$. We pick all possible $\mathcal{S} \subseteq \mathcal{F} \setminus \{j, n\}$ where $|\mathcal{S}| = k - 1$, which implies $|\mathcal{S} \cup \{j, n\}| = k + 1$, then

$$Q_2 = \binom{m-1}{|\mathcal{S}|} \cdot \frac{m - |\mathcal{S}| - 2}{2^{m-|\mathcal{S}|}} = \binom{m-1}{k-1} \cdot \frac{1}{2} \cdot \frac{m-k-1}{2^{m-k}}. \tag{7.39}$$

We pick all possible $\mathcal{S} \subseteq \mathcal{F} \setminus \{j, n\}$ where $|\mathcal{S}| = k$, which implies $|\mathcal{S} \cup \{j\}| = k + 1$, then

$$Q_3 = \binom{m-1}{|\mathcal{S}|} \cdot \frac{1}{2} \cdot \frac{m - |\mathcal{S}| - 2}{2^{m-|\mathcal{S}|}} = \binom{m-1}{k} \cdot \frac{1}{2} \cdot \frac{m-k-2}{2^{m-k}}. \tag{7.40}$$

Then we compute $Q_1 - Q_2 - Q_3$:

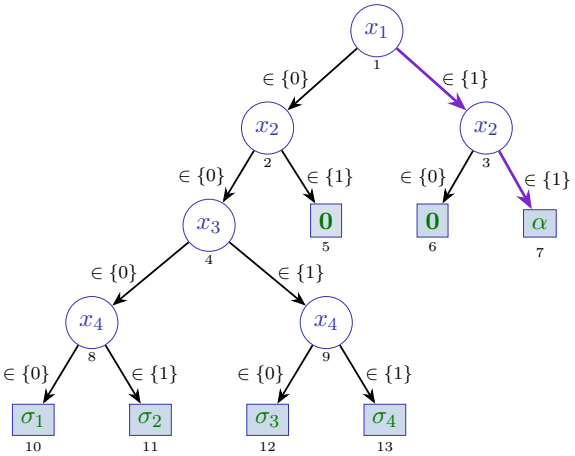
$$\begin{aligned} &\binom{m}{k} \cdot \frac{1}{2} \cdot \frac{m-k}{2^{m-k}} - \binom{m-1}{k-1} \cdot \frac{1}{2} \cdot \frac{m-k-1}{2^{m-k}} - \binom{m-1}{k} \cdot \frac{1}{2} \cdot \frac{m-k-2}{2^{m-k}} \\ &= \frac{1}{2} \cdot \frac{1}{2^{m-k}} \left[ \binom{m}{k}(m-k) - \binom{m-1}{k-1}(m-k-1) - \binom{m-1}{k}(m-k-2) \right] \\ &= \frac{1}{2} \cdot \frac{1}{2^{m-k}} \left[ \binom{m-1}{k-1} + 2\binom{m-1}{k} \right], \end{aligned} \tag{7.41}$$

this means that $\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v}) - \mathsf{SHAP}(j; \mathcal{M}, \mathbf{v}) > 0$. Hence, we can conclude that $|\mathsf{SHAP}(n; \mathcal{M}, \mathbf{v})| > |\mathsf{SHAP}(j; \mathcal{M}, \mathbf{v})|$. □

**Example 44.** Let $\kappa_1(x_1, x_2, x_3) = (x_1 \land x_2 \land \neg x_3) \lor (x_1 \land x_3 \land \neg x_2) \lor (x_2 \land x_3 \land \neg x_1)$ Clearly, $\kappa_1$ predicts the point $(1, 1, 1)$ to 0, and its set of CXp with respects to this point is $\{\{1\}, \{2\}, \{3\}\}$. Set $x_n = x_4$, and build $\kappa_{I5}(x_1, x_2, x_3, x_4) = ((x_1 \land x_2 \land$

| row # | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $\kappa_3$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | $\sigma_1$ |
| 2 | 0 | 0 | 0 | 1 | $\sigma_2$ |
| 3 | 0 | 0 | 1 | 0 | $\sigma_3$ |
| 4 | 0 | 0 | 1 | 1 | $\sigma_4$ |
| 5 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 1 | 0 |
| 7 | 0 | 1 | 1 | 0 | 0 |
| 8 | 0 | 1 | 1 | 1 | 0 |
| 9 | 1 | 0 | 0 | 0 | 0 |
| 10 | 1 | 0 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 0 | 0 |
| 12 | 1 | 0 | 1 | 1 | 0 |
| 13 | 1 | 1 | 0 | 0 | $\alpha$ |
| 14 | 1 | 1 | 0 | 1 | $\alpha$ |
| 15 | 1 | 1 | 1 | 0 | $\alpha$ |
| 16 | 1 | 1 | 1 | 1 | $\alpha$ |

(a) Tabular representation




(b) DT for $\kappa_3$

Figure 7.3: Example parameterized classifier $\kappa_3$. Both the TR and the DT represent the target parameterization

$\neg x_3) \vee (x_1 \wedge x_3 \wedge \neg x_2) \vee (x_2 \wedge x_3 \wedge \neg x_1)) \wedge x_4$. Moreover, as shown in Example 36 and Table 7.1, it is the case that the feature 4 is irrelevant but $|\mathsf{SHAP}(4)|$ is the highest.

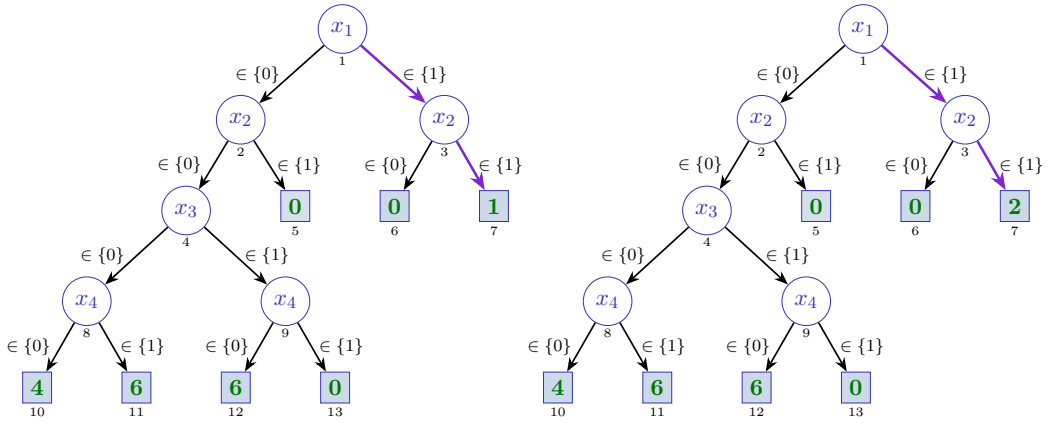
### 7.5.3 Case Study – Multi-Valued Classifier

This case study focuses on constructing arbitrary many examples of the occurrence of issue $\forall j \in \mathcal{F}.([\mathsf{Relevant}(j) \wedge (\mathsf{SHAP}(j) = 0)] \vee [\mathsf{Irrelevant}(j) \wedge (\mathsf{SHAP}(j) \neq 0)])$, which implies I1 to I5.

**Classifier & instance.** This case study is represented by the tabular representation (TR) shown in Figure 7.3a. By inspection, we conclude that $\mathcal{F} = \{1, 2, 3, 4\}$, $\mathbb{D}_i = \{0, 1\}$, $i \in \mathcal{F}$, and $\mathbb{F} = \{0, 1\}^4$. We also pick $\mathcal{K} = \{0, \alpha, \sigma_1, \ldots, \sigma_4\}$. We pick $\mathbf{v} = (1, 1, 1, 1)$, and set the target instance to be $((\mathbf{v}, c) = ((1, 1, 1, 1), \alpha)$, as highlighted in Figure 7.3a. And we impose $\alpha \neq \sigma_j, j = 1, \ldots, 4$, but also that $\alpha \neq 0$. Figure 7.3b shows a possible DT for the parameterized classifier.

Figure 7.4 shows two possible instantiations of the parameterized classifier. One example instantiation of the classifier is shown as the DT of Figure 7.4a, and it is obtained by setting $\alpha = 1$, $\sigma_1 = 4$, $\sigma_2 = \sigma_3 = 6$ and $\sigma_4 = 0$. Another example instantiation of the classifier is shown in Figure 7.4b, and it is obtained by setting $\alpha = 2$, $\sigma_1 = 4$, $\sigma_2 = \sigma_3 = 6$ and $\sigma_4 = 0$. In terms of the differences between the two DTs, only a single terminal node (i.e. terminal node 7) changes its predicted class.

**Feature influence in predicted class.** By (manual) inspection of the parameterized TR & DT shown in Figure 7.3, and the instantiations in Figure 7.4, it is plain




(a) DT for $\kappa_{3,a}$ (b) DT for $\kappa_{3,b}$

Figure 7.4: Two example instantiations for the classifier $\kappa_3$, with $\sigma_1 = 4$, $\sigma_2 = 6$, $\sigma_3 = 6$, $\sigma_4 = 0$ and either $\alpha = 1$ (case $\kappa_{3,a}$) or $\alpha = 2$ (case $\kappa_{3,b}$).

that, for any point in feature space, the prediction is $\alpha$ if and only if $x_1 = x_2 = 1$; otherwise the predicted class is other than $\alpha$. It is also plain that the predicted class $\alpha$ is determined *solely* by the values assigned to features 1 and 2, and it is *independent* of the values assigned to features 3 or 4. Observe also that these are properties of the classifier's function, and the DTs serve only to make the argument simpler to grasp. Evidently, the same arguments apply when our aim is to change the prediction to a class other than $\alpha$. Therefore, for the target point $(1, 1, 1, 1)$, and given this analysis of the influence of each feature on the predicted class $\alpha$, one should expect features 1 and 2 to play some role in answering the question *why* the predicted class is $\alpha$, but also in answering the question why not a predicted class other than $\alpha$, i.e. what to change to also change the prediction.

**Abductive & contrastive explanations.** For the instance $((1, 1, 1, 1), \alpha)$, and for *any* of the classifiers that can be instantiated from Figure 7.3a, the sets of AXps and CXps are:

$$\begin{aligned} \mathbb{A} &= \{\{1, 2\}\} \\ \mathbb{C} &= \{\{1\}, \{2\}\} \end{aligned} \tag{7.42}$$

As a result, it is clear that features 1 and 2 are relevant, and that features 3 and 4 are irrelevant, for the target instance.

**Adversarial examples.** We seek minimal $l_0$-distance adversarial examples. Recall that the target instance is $((1, 1, 1, 1), \alpha)$. Given the DTs shown in Figures 7.3b and 7.4, it is plain that to obtain a predicted class other than $\alpha = 1$, we must change the value of either feature 1 or 2. Hence, an $l_0$-minimal adversarial example must assign $x_1 = 0$ or $x_2 = 0$, but not both. Evidently, for a minimal AE, the features 3 or 4 *must* not be changed.

Similarly to the case of abductive and contrastive explanations, we conclude

| $\mathcal{S}$ | $\phi(\mathcal{S})$ | $\phi(\mathcal{S}\cup\{1\})$ | $\Delta(\mathcal{S})$ | $\varsigma(\mathcal{S})$ | $\varsigma(\mathcal{S})\times\Delta(\mathcal{S})$ |
|---|---|---|---|---|---|
| $\emptyset$ | $(\sum_{j=1}^{4}\sigma_j)/16+\alpha/4$ | $\alpha/2$ | $\alpha/4-(\sum_{j=1}^{4}\sigma_j)/16$ | $1/4$ | $\alpha/16-(\sum_{j=1}^{4}\sigma_j)/64$ |
| $\{2\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/12$ | $\alpha/24$ |
| $\{3\}$ | $(\sigma_3+\sigma_4)/8+\alpha/4$ | $\alpha/2$ | $\alpha/4-(\sigma_3+\sigma_4)/8$ | $1/12$ | $\alpha/48-(\sigma_3+\sigma_4)/96$ |
| $\{4\}$ | $(\sigma_2+\sigma_4)/8+\alpha/4$ | $\alpha/2$ | $\alpha/4-(\sigma_2+\sigma_4)/8$ | $1/12$ | $\alpha/48-(\sigma_2+\sigma_4)/96$ |
| $\{2,3\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/12$ | $\alpha/24$ |
| $\{2,4\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/12$ | $\alpha/24$ |
| $\{3,4\}$ | $\sigma_4/4+\alpha/4$ | $\alpha/2$ | $\alpha/4-\sigma_4/4$ | $1/12$ | $\alpha/48-\sigma_4/48$ |
| $\{2,3,4\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/4$ | $\alpha/8$ |
| | SHAP score for feature 1 | | $\mathsf{SHAP}(1)$ | $=$ | $3\alpha/8-(3\sigma_1+5\sigma_2+5\sigma_3+11\sigma_4)/192$ |

| $\mathcal{S}$ | $\phi(\mathcal{S})$ | $\phi(\mathcal{S}\cup\{2\})$ | $\Delta(\mathcal{S})$ | $\varsigma(\mathcal{S})$ | $\varsigma(\mathcal{S})\times\Delta(\mathcal{S})$ |
|---|---|---|---|---|---|
| $\emptyset$ | $(\sum_{j=1}^{4}\sigma_j)/16+\alpha/4$ | $\alpha/2$ | $\alpha/4-(\sum_{j=1}^{4}\sigma_j)/16$ | $1/4$ | $\alpha/16-(\sum_{j=1}^{4}\sigma_j)/64$ |
| $\{1\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/12$ | $\alpha/24$ |
| $\{3\}$ | $(\sigma_3+\sigma_4)/8+\alpha/4$ | $\alpha/2$ | $\alpha/4-(\sigma_3+\sigma_4)/8$ | $1/12$ | $\alpha/48-(\sigma_3+\sigma_4)/96$ |
| $\{4\}$ | $(\sigma_2+\sigma_4)/8+\alpha/4$ | $\alpha/2$ | $\alpha/4-(\sigma_2+\sigma_4)/8$ | $1/12$ | $\alpha/48-(\sigma_2+\sigma_4)/96$ |
| $\{1,3\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/12$ | $\alpha/24$ |
| $\{1,4\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/12$ | $\alpha/24$ |
| $\{3,4\}$ | $\sigma_4/4+\alpha/4$ | $\alpha/2$ | $\alpha/4-\sigma_4/4$ | $1/12$ | $\alpha/48-\sigma_4/48$ |
| $\{1,3,4\}$ | $\alpha/2$ | $\alpha$ | $\alpha/2$ | $1/4$ | $\alpha/8$ |
| | SHAP score for feature 2 | | $\mathsf{SHAP}(2)$ | $=$ | $3\alpha/8-(3\sigma_1+5\sigma_2+5\sigma_3+11\sigma_4)/192$ |

Table 7.2: Computation of SHAP scores for the example DT of $\kappa_3$ and instance $((1,1,1,1),\alpha)$, for features 1 and 2. For each feature $i$, the sets to consider are all the sets that do not include the feature. The average values are obtained by summing up the values of the classifier in the rows consistent with $\mathcal{S}$ and dividing by the total number of rows.

about the importance of features and 1 and 2, and about the unimportance of features 3 and 4. We also conclude that a more formal analysis yields the same conclusions we obtained by manual inspection of the DTs.

**SHAP scores.** For the target instance $((1,1,1,1),\alpha)$, we compute the SHAP scores for the parameterized classifier, and then use the obtained symbolic expressions to obtain the conditions to devise classifiers for which SHAP scores produce misleading information. The computation of SHAP scores is summarized in Tables 7.2 and 7.3.

Given the computated of the SHAP scores in Tables 7.2 and 7.3, we proceed to derive the conditions for a classifier, parameterized by Figure 7.3b, to have an issue $\forall j\in\mathcal{F}.([\mathsf{Relevant}(j)\wedge(\mathsf{SHAP}(j)=0)]\vee[\mathsf{Irrelevant}(j)\wedge(\mathsf{SHAP}(j)\neq 0)])$. Given that both features 1 and 2 are relevant, and features 3 and 4 are irrelevant, then we seek to obtain: $\mathsf{SHAP}(1)=0$, $\mathsf{SHAP}(2)=0$, $\mathsf{SHAP}(3)\neq 0$, and $\mathsf{SHAP}(4)\neq 0$.

| $\mathcal{S}$ | $\phi(\mathcal{S})$ | $\phi(\mathcal{S}\cup\{3\})$ | $\Delta(\mathcal{S})$ | $\varsigma(\mathcal{S})$ | $\varsigma(\mathcal{S})\times\Delta(\mathcal{S})$ |
|---|---|---|---|---|---|
| $\emptyset$ | $(\sum_{j=1}^{4}\sigma_j)/16+\alpha/4$ | $(\sigma_3+\sigma_4)/8+\alpha/4$ | $(-\sigma_1-\sigma_2+\sigma_3+\sigma_4)/16$ | $1/4$ | $(-\sigma_1-\sigma_2+\sigma_3+\sigma_4)/64$ |
| $\{1\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{2\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{4\}$ | $(\sigma_2+\sigma_4)/8+\alpha/4$ | $\sigma_4/4+\alpha/4$ | $-\sigma_2/8+\sigma_4/8$ | $1/12$ | $-\sigma_2/96+\sigma_4/96$ |
| $\{1,2\}$ | $\alpha$ | $\alpha$ | $0$ | $1/12$ | $0$ |
| $\{1,4\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{2,4\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{1,2,4\}$ | $\alpha$ | $\alpha$ | $0$ | $1/4$ | $0$ |
| | SHAP score for feature 3 | | $\mathsf{SHAP}(3)$ | $=$ | $(-3\sigma_1-5\sigma_2+3\sigma_3+5\sigma_4)/192$ |

| $\mathcal{S}$ | $\phi(\mathcal{S})$ | $\phi(\mathcal{S}\cup\{4\})$ | $\Delta(\mathcal{S})$ | $\varsigma(\mathcal{S})$ | $\varsigma(\mathcal{S})\times\Delta(\mathcal{S})$ |
|---|---|---|---|---|---|
| $\emptyset$ | $(\sum_{j=1}^{4}\sigma_j)/16+\alpha/4$ | $(\sigma_2+\sigma_4)/8+\alpha/4$ | $(-\sigma_1+\sigma_2-\sigma_3+\sigma_4)/16$ | $1/4$ | $(-\sigma_1+\sigma_2-\sigma_3+\sigma_4)/64$ |
| $\{1\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{2\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{3\}$ | $(\sigma_3+\sigma_4)/8+\alpha/4$ | $\sigma_4/4+\alpha/4$ | $-\sigma_3/8+\sigma_4/8$ | $1/12$ | $-\sigma_3/96+\sigma_4/96$ |
| $\{1,2\}$ | $\alpha$ | $\alpha$ | $0$ | $1/12$ | $0$ |
| $\{1,3\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{2,3\}$ | $\alpha/2$ | $\alpha/2$ | $0$ | $1/12$ | $0$ |
| $\{1,2,3\}$ | $\alpha$ | $\alpha$ | $0$ | $1/4$ | $0$ |
| | SHAP score for feature 4 | | $\mathsf{SHAP}(4)$ | $=$ | $(-3\sigma_1+3\sigma_2-5\sigma_3+5\sigma_4)/192$ |

Table 7.3: Computation of SHAP scores for the example DT of $\kappa_3$ and instance $((1,1,1,1),\alpha)$, for features 3 and 4.

Since $\mathsf{SHAP}(1)=\mathsf{SHAP}(2)$, we obtain the following constraints:

$$\alpha = (3\sigma_1+5\sigma_2+5\sigma_3+11\sigma_4)/72 \tag{7.43}$$

$$(-3\sigma_1-5\sigma_2+3\sigma_3+5\sigma_4)/192 \neq 0 \tag{7.44}$$

$$(-3\sigma_1+3\sigma_2-5\sigma_3+5\sigma_4)/192 \neq 0 \tag{7.45}$$

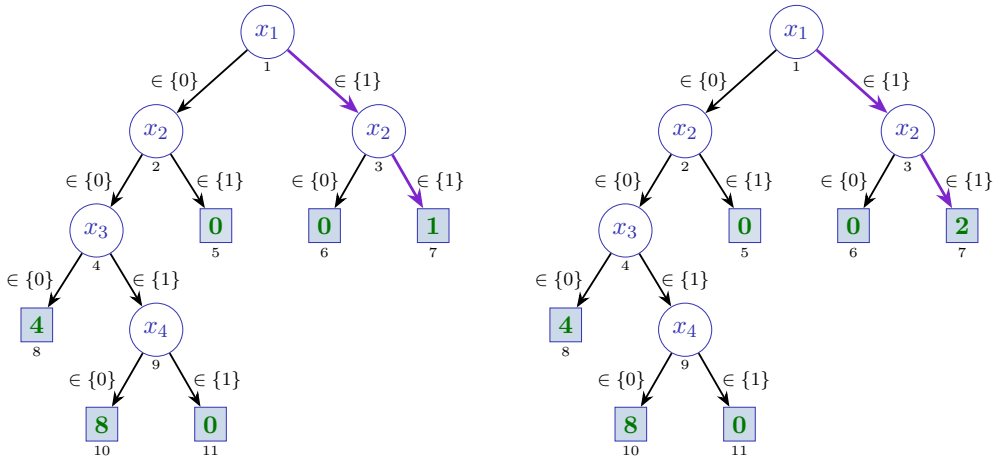
By plugging in the instantiated values in the above constraints, we can conclude that Figure 7.4a respects the conditions above, whereas Figure 7.4b does not. For the two instantiated DTs, the computed SHAP scores are shown in Table 7.4. As can be observed, despite the minor changes between the two DTs, the differences in computed SHAP scores are not only very significant, but also cause important differences in the obtained ranking of feature importance. (The ranking of features is by decreasing absolute SHAP score.)

Further analysis of the conditions to set $\mathsf{SHAP}(1)=\mathsf{SHAP}(2)=0$ yielded the case $\sigma_1=\sigma_2=4$, $\sigma_3=8$ and $\sigma_4=0$, with $\alpha=1$. This solution reduces the number of nodes in the template DT (see Figure 7.5a). By changing the value of $\alpha$ we are able to increase the importance of features 1 and 2. The resulting DTs are shown in Figure 7.5b.

The computed SHAP scores for the classifiers represented by the DTs in Figures 7.4 and 7.5 is shown in Table 7.4. The results confirm that, as expected, for the two DTs configured to cancel the SHAP scores of the key features 1 and 2

| Classifier | SHAP(1) | SHAP(2) | SHAP(3) | SHAP(4) | Rank |
|---|---|---|---|---|---|
| $\kappa_{3,a}$ | 0.000 | 0.000 | -0.125 | -0.125 | $\langle 3:4, 1:2\rangle$ |
| $\kappa_{3,b}$ | 0.375 | 0.375 | -0.125 | -0.125 | $\langle 1:2, 4:3\rangle$ |
| $\kappa_{3,c}$ | 0.000 | 0.000 | -0.042 | -0.208 | $\langle 4, 3, 1:2\rangle$ |
| $\kappa_{3,d}$ | 0.375 | 0.375 | -0.042 | -0.208 | $\langle 1:2, 4, 3\rangle$ |

Table 7.4: SHAP scores for the classifiers in Figures 7.4 and 7.5




(a) DT for $\kappa_{3,c}$ (b) DT for $\kappa_{3,d}$

Figure 7.5: Two more example instantiations for the classifier $\kappa_3$, with $\sigma_1 = \sigma_2 = 4$, $\sigma_3 = 8$, $\sigma_4 = 0$, and either $\alpha = 1$ (case $\kappa_{3,c}$), or $\alpha = 2$ (case $\kappa_{3,d}$).

(see Figures 7.4a and 7.5a) we get a SHAP score of 0. By only changing one value in the DTs, i.e. increasing $\alpha$ from 1 to 2, we get completely different SHAP scores, which now assign importance to features 1 and 2 (see Figures 7.4b and 7.5b).

**Assessment.** The conclusion to draw from this case study is that SHAP scores produce misleading information about relative feature importance. Indeed, the case study analyzed in this section reveals an issue $\forall j \in \mathcal{F}.([\mathsf{Relevant}(j) \land (\mathsf{SHAP}(j) = 0)] \lor [\mathsf{Irrelevant}(j) \land (\mathsf{SHAP}(j) \neq 0)])$. For $\kappa_{3,a}$, SHAP scores assign some importance to features 3 and 4, and *no* importance to features 1 and 2. The features that have any influence of the predicted class are exactly features 1 and 2, i.e. the features assigned no importance in terms of SHAP scores.

**Preliminary remarks.** It is straightforward to devise really simple classifiers for which the relative feature importance obtained with SHAP scores is evidently misleading. In the case study, and depending on the selection of picked classes, the most important feature can be forced to be assigned a SHAP score of 0, denoting no importance. We also showed that one can devise classifiers where *several* relevant features can be forced to be assigned no importance according to SHAP scores.

Also significant is that several of the classifiers shown are parameterized, and so this effectively indicates that there are *arbitrary many* instantiated classifiers that will yield the conclusions obtained with the case studies analyzed in this section.

One may ask why is it fairly straightforward to force SHAP scores to produce misleading information. The key reason is that the computation of SHAP scores accounts for all possible subsets of fixed features, and some of these subset subsets play no role in the prediction that is being explained. By suitably picking average values, we can force SHAP scores to completely misrepresent the relative importance of features. Evidently, if we can generate those examples, then similar situations are bound to occur in practice.

The next sections present additional evidence, demonstrating that similar results are obtained in a wide range of situations, including classifiers learned form publicly available datasets.

## 7.6 Issues with SHAP scores: Practice

This section demonstrates that the identified issues with SHAP scores as discussed in Section 7.4, are indeed exist in a number of decision trees, d-DNNF circuits and OMDDs. These models have either been described in published works or can be learned from publicly available datasets. The experiments were performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Ventura.

### 7.6.1 Examples of Decision Trees

This section studies two example DTs. However, in contrast with the classifiers studied earlier in this chapter, these DTs has been studied in earlier works [229, 382], and represent concrete use cases. The choice of DTs is motivated by their size, i.e. the DTs are not too small and so not trivial to analyze, and by the fact that it exhibits some of the issues with SHAP scores that have been studied in this chapter.

For the selected DT, we investigate whether there are instances exhibiting issue I2 $\exists i, j \in \mathcal{F}.[\mathsf{Irrelevant}(i) \land \mathsf{Relevant}(j) \land (|\mathsf{SHAP}(i)| > |\mathsf{SHAP}(j)|)]$, i.e. whether the rank of SHAP scores is misleading. This test was conducted for all possible instances in the feature space. The method for computing SHAP scores in the case of DTs is based on Equation (2.12), under the assumption of a uniform distribution of data points across the feature space. Computation of explanations is based on earlier work as well [169, 194].

**Example Decision Tree.** We consider two decision trees with discrete features and classes, discussed in the literature [229, Figure 9] and the other from [382, Figure 4.8]. The DTs are shown in Figures 7.6 and 7.7. For simplicity, the DTs use set notation for the literals, as proposed in recent work [194]. Table 7.5 shows the

Table 7.5: Mapping of original features for the DT from [229]. The original classes {MD, Non-MD} are mapped to $\{\mathbf{Y}, \mathbf{N}\}$.

| Feature | Acronym | Original Domain | Feature # | Mapped Domain |
|---|---|---|---|---|
| Age | $A$ | $\{A \leq 5, A > 5\}$ | 1 | $\{0, 1\}$ |
| Petechiae | $P$ | {no, yes} | 2 | $\{0, 1\}$ |
| Neck Stiffness | $N$ | {no, yes} | 3 | $\{0, 1\}$ |
| Vomiting | $V$ | {no, yes} | 4 | $\{0, 1\}$ |
| Zone | $Z$ | {rural, peri-urban, urban} | 5 | $\{0, 1, 2\}$ |
| Seizures | $S$ | {no, yes} | 6 | $\{0, 1\}$ |
| Headache | $H$ | {no, yes} | 7 | $\{0, 1\}$ |
| Comma | $C$ | {no, yes} | 8 | $\{0, 1\}$ |
| Gender | $G$ | {female, male} | 9 | $\{0, 1\}$ |

Table 7.6: Mapping of original features for the DT from [382]. The original classes {ripe, unripe} are mapped to {Y, N}.

| Feature | Acronym | Original Domain | Feature # | Mapped Domain |
|---|---|---|---|---|
| Texture | $T$ | {slightly blurry, clear, blurry} | 1 | $\{0, 1, 2\}$ |
| Root | $R$ | {curly, slightly curly, straight} | 2 | $\{0, 1, 2\}$ |
| Color | $C$ | {green, dark, light} | 3 | $\{0, 1, 2\}$ |
| Surface | $S$ | {hard, soft} | 4 | $\{0, 1\}$ |
| Sound | $O$ | {dull, muffled, crisp} | 5 | $\{0, 1, 2\}$ |
| Umbilicus | $U$ | {hollow, slightly hollow, flat} | 6 | $\{0, 1, 2\}$ |

feature domains of the DT in Figure 7.6, while Table 7.6 shows the feature domains of the DT in Figure 7.7.

**Summary of results.** For each instance, all AXps are enumerated. This serves to decide which features are relevant and which are irrelevant. Then we compute the SHAP scores for each feature and analyze whether the issue I2 $\exists i, j \in \mathcal{F}.[\mathsf{Irrelevant}(i) \land \mathsf{Relevant}(j) \land (|\mathsf{SHAP}(i)| > |\mathsf{SHAP}(j)|)]$ occurs. If an instance exhibits such an issue, we plot a pair of values $(v_i, v_j)$. More specifically, $v_i = \max\{|\mathsf{SHAP}(k)| \,|\, k \notin F_{\mathbb{A}(\mathcal{E})}\}$ and $v_j = \min\{|\mathsf{SHAP}(k)| \,|\, k \in F_{\mathbb{A}(\mathcal{E})}\}$. (Observe that this means that the relative order of feature importance will be misleading.) We then plot $v_i$ in yellow and $v_j$ in blue, these pairs of values are depicted in Figures 7.8 and 7.9. Another observation is the occurrence of issues with SHAP scores is non-negligible. For the DT in Figure 7.6, 151 out of 768 instances exhibit the aforementioned issue, i.e. 19.7% of the total. Moreover, for the DT in Figure 7.7, 82 out of 486 instances exhibit the same issue, i.e. 16.8% of the total.

Moreover, for the DT in Figure 7.6, we found that for the instance $((1, 0, 0, 0, 0, 0, 1, 1, 1), 1)$, there exist two AXps: $\{1, 5\}$ and $\{1, 4\}$ and the SHAP scores are: $\mathsf{SHAP}(1) = 0.3572$, $\mathsf{SHAP}(2) = -0.1428$, $\mathsf{SHAP}(3) = -0.0178$, $\mathsf{SHAP}(4) = 0.0449$, $\mathsf{SHAP}(5) = 0.0449$, $\mathsf{SHAP}(6) = -0.0029$, $\mathsf{SHAP}(7) = -0.002$, $\mathsf{SHAP}(8) = 0.0005$, $\mathsf{SHAP}(9) = 0.0005$. As can be concluded, for this instance, feature 2 is irrelevant and feature 4 and 5 are relevant. However, we have $|\mathsf{SHAP}(2)| > |\mathsf{SHAP}(4)|$ and

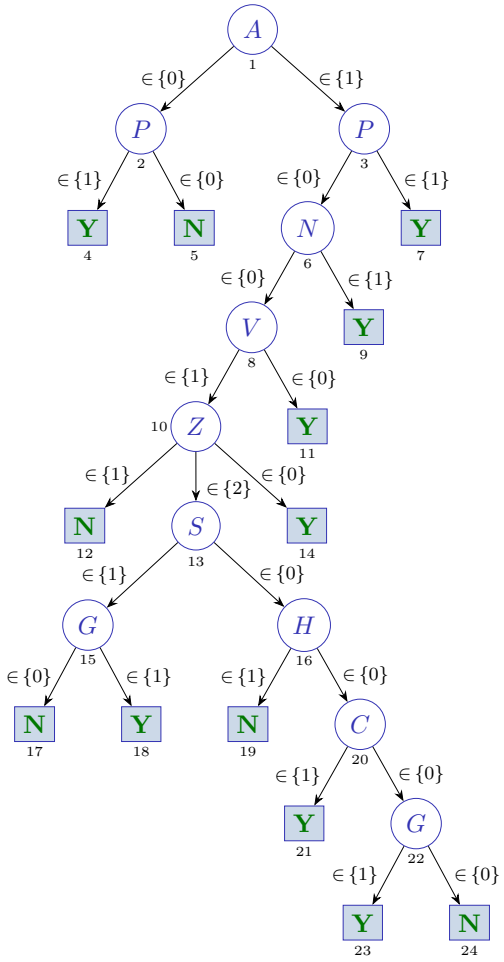



Figure 7.6: Example DT, adapted from [229, Figure 9]

$|\mathsf{SHAP}(2)| > |\mathsf{SHAP}(5)|$. Additionally, for the same DT, we found two instances such that relevant features assigned with a SHAP score of 0. Specifically, for the instance $((1,1,1,0,2,1,1,0,1),1)$, we can compute four AXps: $\{2\}$, $\{1,5,6,7\}$, $\{1,4\}$, and $\{1,3\}$. The SHAP scores are: $\mathsf{SHAP}(1) = 0.1172$, $\mathsf{SHAP}(2) = 0.1373$, $\mathsf{SHAP}(3) = 0.0123$, $\mathsf{SHAP}(4) = 0.0123$, $\mathsf{SHAP}(5) = 0$, $\mathsf{SHAP}(6) = 0.0016$, $\mathsf{SHAP}(7) = 0.0016$, $\mathsf{SHAP}(8) = -0.0003$, $\mathsf{SHAP}(9) = 0.0004$. Clearly, feature 5 is relevant but its SHAP score is 0. For another instance $((1,1,1,0,2,1,1,1,0),1)$, we can compute four AXps: $\{2\}$, $\{1,5,6,7\}$, $\{1,4\}$, and $\{1,3\}$. The SHAP scores are: $\mathsf{SHAP}(1) = 0.1172$, $\mathsf{SHAP}(2) = 0.1373$, $\mathsf{SHAP}(3) = 0.0123$, $\mathsf{SHAP}(4) = 0.0123$, $\mathsf{SHAP}(5) = 0$, $\mathsf{SHAP}(6) = 0.0016$, $\mathsf{SHAP}(7) = 0.0016$, $\mathsf{SHAP}(8) = 0.0004$, $\mathsf{SHAP}(9) = -0.0003$. Clearly, for the relevant feature 5, it has a SHAP score of 0.

### 7.6.2 Examples of d-DNNF circuits

In this section, we repeated the same experiment comparing SHAP scores and feature relevancy but for d-DNNF circuits. We consider six publicly available datasets

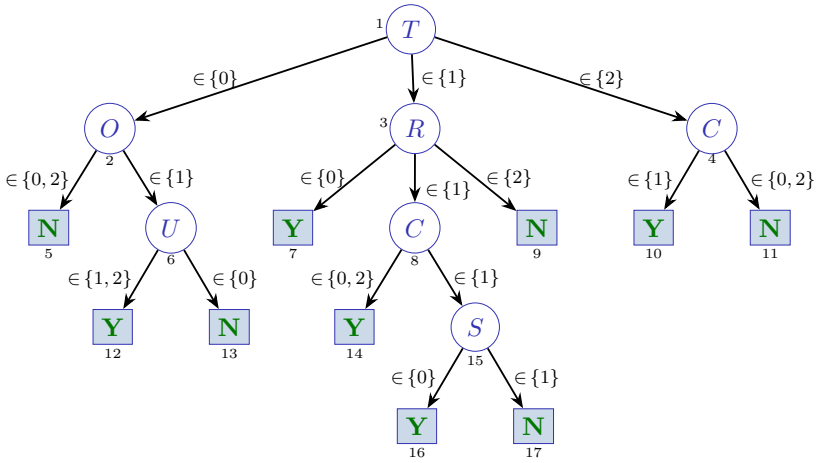



Figure 7.7: Example DT, adapted from [382, Figure 4.8]

Table 7.7: Description of d-DNNF circuits. # Nodes denotes the number of nodes in the circuit.

| Dataset | $\lvert\mathcal{F}\rvert$ | Feature Domain Sizes | $\lvert\mathcal{K}\rvert$ | # Nodes |
|---|---|---|---|---|
| corral | 6 | $\mathbb{B}^6$ | 2 | 69 |
| mofn_3_7_10 | 10 | $\mathbb{B}^{10}$ | 2 | 207 |
| mux6 | 6 | $\mathbb{B}^6$ | 2 | 106 |
| parity5+5 | 10 | $\mathbb{B}^{10}$ | 2 | 847 |
| threeOf9 | 9 | $\mathbb{B}^9$ | 2 | 167 |
| xd6 | 9 | $\mathbb{B}^9$ | 2 | 194 |

and analyze whether there are instances exhibit the issue I2 $\exists i, j \in \mathcal{F}.[\mathsf{Irrelevant}(i) \land \mathsf{Relevant}(j) \land (|\mathsf{SHAP}(i)| > |\mathsf{SHAP}(j)|)]$. Besides, for each dataset we test all possible instances in the feature space. These six datasets are from the Penn Machine Learning Benchmarks [282], with boolean features and boolean classes. To obtain d-DNNF circuits, we first trained Read-once Decision Tree (RODT) models on the given datasets using Orange3 [104] and then mapped the obtained RODTs into d-DNNFs. RODTs can be encoded in linear time as d-DNNF circuits [20, 22]. For computing SHAP scores, we assumed a uniform data distribution for each dataset. The algorithm for computing SHAP scores of d-DNNFs was proposed in [20, 22]. Computation of explanations is based on earlier work as well [169, 194].

**Description of the datasets.** Table 7.7 summarizes the characteristics of the six d-DNNFs used in the experiments.

**Summary of results.** For the case of d-DNNF circuits, we repeat the experiment conducted in Section 7.6.1 and plot their results. These results are depicted in

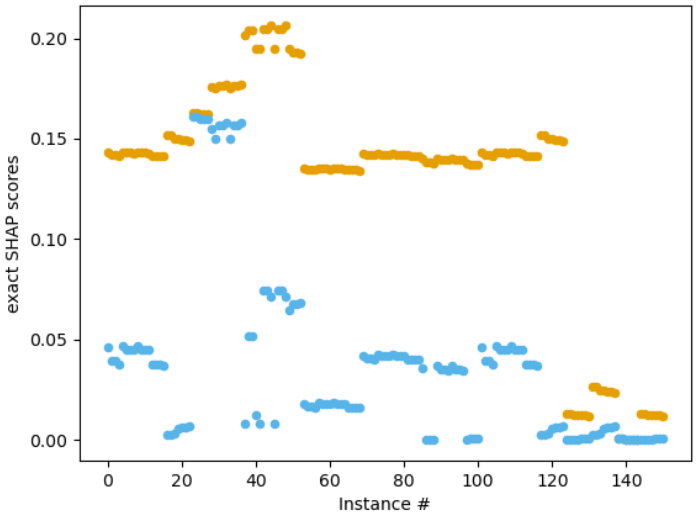



Figure 7.8: Plot showing whether there exist irrelevant features (dots in yellow) with higher scores than relevant features (dots in blue) in absolute value, for the DT in Figure 7.6.

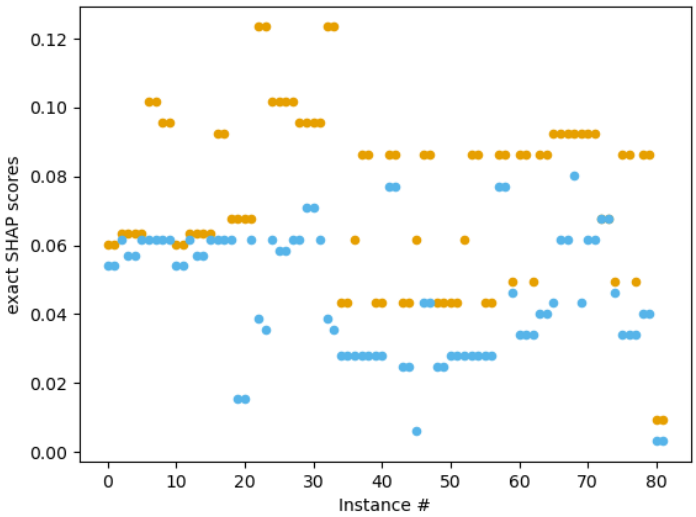
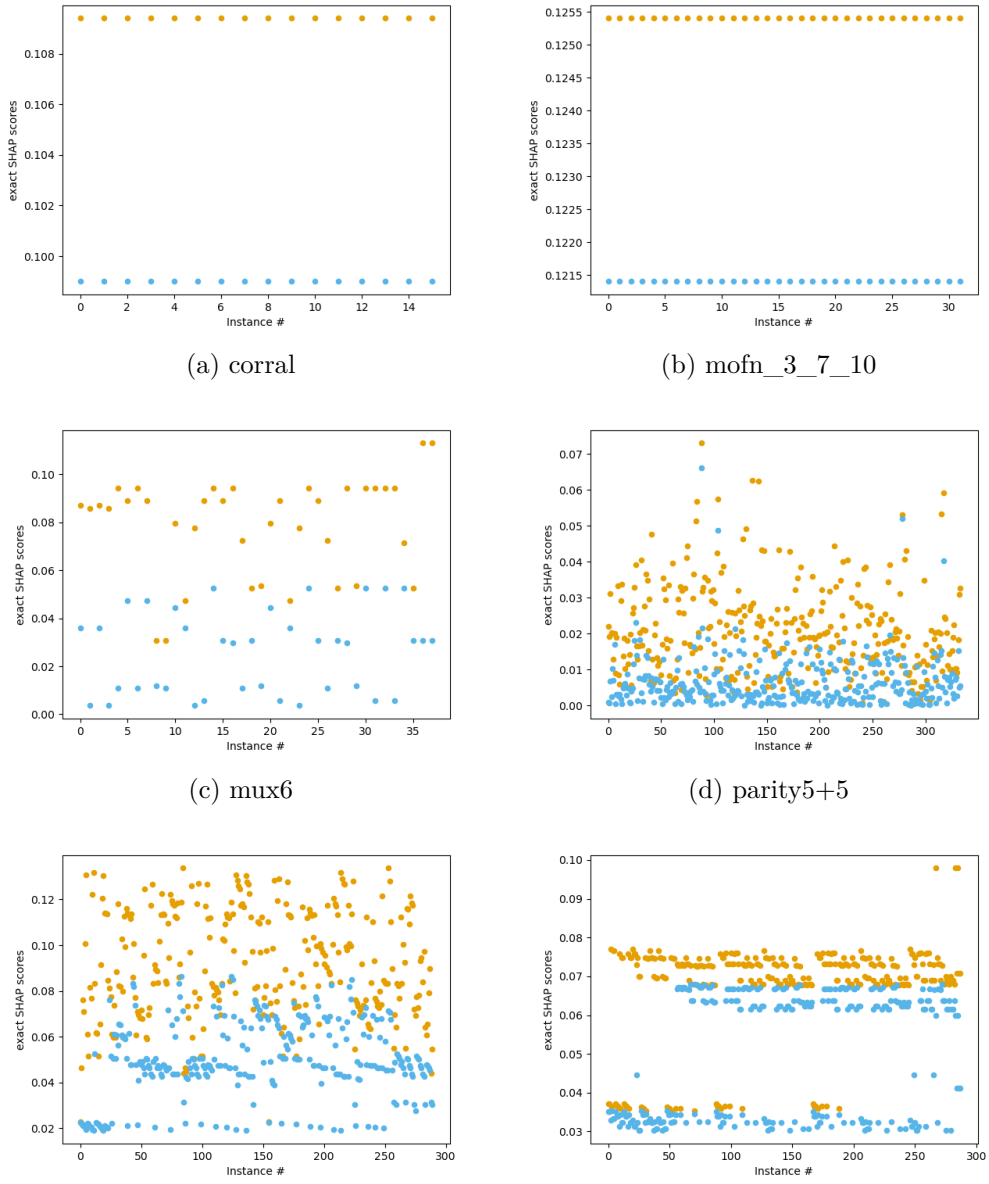



Figure 7.9: Plot showing whether there exist irrelevant features (dots in yellow) with higher scores than relevant features (dots in blue) in absolute value, for the DT in Figure 7.7.




(a) corral (b) mofn_3_7_10

(c) mux6 (d) parity5+5

(e) threeOf9 (f) xd6

Figure 7.10: Plot showing whether there exist irrelevant features (dots in yellow) with higher scores than relevant features (dots in blue) in absolute value.

Figure 7.10.

An observation is the occurrence of issues with SHAP scores is non-negligible. For the d-DNNF in Figure 7.10a, 16 out of 64 instances (i.e. 25%) exhibit the aforementioned issue. For the d-DNNF in Figure 7.10b, 32 out of 1024 instances (i.e. 3.1%) exhibit the same issue. For the d-DNNF in Figure 7.10c, 38 out of 64 instances (i.e. 59.4%) exhibit the same issue. For the d-DNNF in Figure 7.10d, 344 out of 1024 instances (i.e. 32.6%) exhibit the same issue. And for the d-DNNF in Figure 7.10e, 290 out of 512 instances (i.e. 56.6%) exhibit the same issue. And for the d-DNNF in Figure 7.10f, 288 out of 512 instances (i.e. 56.3%) exhibit the same issue.

### 7.6.3 Examples of Multi-Valued Decision Diagrams

In this section, we repeated the same experiment comparing SHAP scores and feature relevancy but for OMDDs. we consider five publicly available datasets and analyze whether there are instances exhibit the issue I2 $\exists i, j \in \mathcal{F}.[\mathsf{Irrelevant}(i) \land \mathsf{Relevant}(j) \land (|\mathsf{SHAP}(i)| > |\mathsf{SHAP}(j)|)]$. These five datasets are from the Penn Machine Learning Benchmarks [282], with discrete features and classes. For each dataset, we picked a consistent subset of samples (i.e. no two instances are contradictory) for building OMDDs. For example, for the dataset `postoperative_patient_data`, there are only 88 instances, and a consistent subset of samples include 66 instances. OMDDs were built heuristically using a publicly available package MEDDLY [4], which is implemented in C/C++. For computing SHAP scores, we assumed a uniform data distribution for each dataset. Besides, for each dataset we test randomly picked 200 instances or all instances if there are less than 200 rows in the dataset.

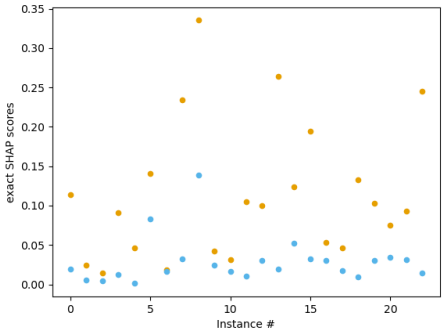
The method computing SHAP scores is based on Equation (2.12). However, it is known that OMDDs [279] are *deterministic* and *decomposable*. Moreover, they also supports the query *polytime model counting*, and the transformation *polytime conditioning* [207, 279]. This means the algorithm proposed in [20, 22] for computing SHAP scores of d-DNNFs can be extended to the case of OMDDs. Computation of explanations is based on earlier work as well [169, 194].

**Description of the datasets.** Table 7.8 summarizes the characteristics of the five OMDDs used in the experiments.

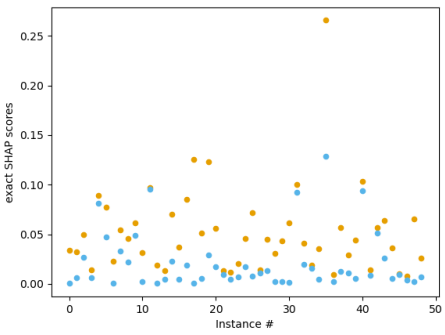
**Summary of results.** For the case of OMDDs, we repeat the experiment conducted in Section 7.6.1 and plot their results. These results are depicted in Figure 7.11.

An observation is the occurrence of issues with SHAP scores is non-negligible. For the OMDD in Figure 7.11a, 23 out of 200 instances (i.e. 11.5%) exhibit the aforementioned issue. For the OMDD in Figure 7.11b, 49 out of 200 instances (i.e.

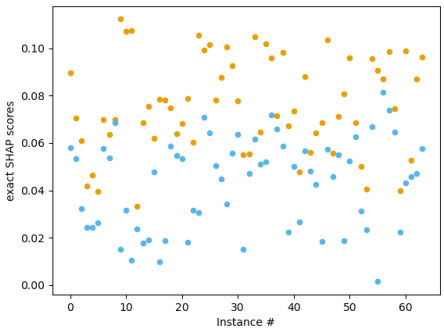
[4] `https://asminer.github.io/meddly/`



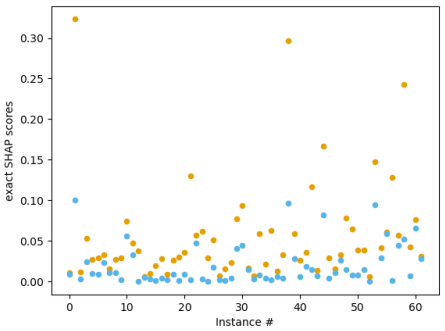

(a) car



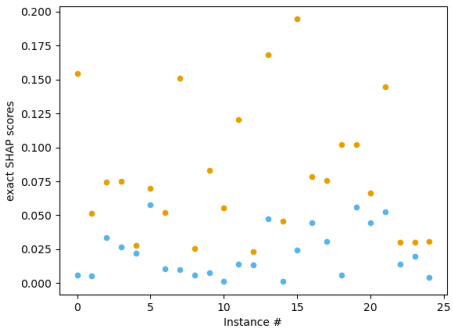

(b) monk1




(c) monk2




(d) monk3




(e) postoperative_patient_data

Figure 7.11: Plot showing whether there exist irrelevant features (dots in yellow) with higher scores than relevant features (dots in blue) in absolute value.

Table 7.8: Description of OMDD classifiers. # Nodes denotes the number of nodes in the OMDD.

| Dataset | $\lvert\mathcal{F}\rvert$ | Feature Domain Sizes | $\lvert\mathcal{K}\rvert$ | # Nodes |
|---|---|---|---|---|
| car | 6 | $4 \times 4 \times 4 \times 3 \times 3 \times 3$ | 4 | 248 |
| monk1 | 6 | $3 \times 3 \times 2 \times 3 \times 4 \times 2$ | 2 | 68 |
| monk2 | 6 | $3 \times 3 \times 2 \times 3 \times 4 \times 2$ | 2 | 70 |
| monk3 | 6 | $3 \times 3 \times 2 \times 3 \times 4 \times 2$ | 2 | 74 |
| post._patient | 8 | $3 \times 3 \times 2 \times 3 \times 2 \times 3 \times 3 \times 5$ | 2 | 109 |

24.5%) exhibit the same issue. For the OMDDs in Figures 7.11c and 7.11d, 64 out of 200 instances (i.e. 32%) exhibit the same issue. And for the OMDD in Figure 7.11e, 22 out of 66 instances (i.e. 33.3%) exhibit the same issue.

## 7.7 Discussion

This section addresses and rebuts a number of possible criticisms to the results presented in this chapter.

**Definition of (ir)relevant features.** Our definition of (ir)relevant features mirrors the one proposed and studied in logic-based abduction [110] since the early and mid 90s. (Logic-based abduction formalizes the concept of abduction, studied in logic and philosophy for more than a century [154].) Nevertheless, we explicitly consider subset-minimality for the definition of (abductive) explanation, whereas logic-based abduction contemplates other possible definitions [110]. For example, there are other definitions of (minimal) explanation which involve a user indicating some sort of preference among hypotheses (or features), that can involve some sort of prioritization or penalization [110]. Since SHAP scores are not defined in terms of user-specified preferences, this sort of preference-minimal explanations are inapplicable in our setting. In addition, another definition of explanations involves those that are cardinality-minimal [110]. The following is a straightforward observation.

**Proposition 34.** Any feature that is deemed irrelevant under a subset-minimal definition of explanation must also be an irrelevant feature under a cardinality-minimal definition of explanation.

Most of the examples in this chapter and earlier reports [173, 175, 174] already consider a single explanation which is necessarily cardinality-minimal. Hence, replacing a subset-minimal definition of explanation by a cardinality-minimal definition would not impact the implications of the results presented in this chapter and earlier reports [173, 175, 174] in terms of the inadequacy of SHAP scores for XAI.

Furthermore, the results presented in this chapter and earlier reports [173, 175, 174] demonstrate that SHAP scores for XAI do not correlate with the information obtained from adversarial examples.

**Definition of Shapley values for XAI.** Although this chapter and earlier reports [173, 175, 174] consider a well-established definition of Shapley values for XAI, specifically the one proposed in a number of well-known references [247, 21, 105, 106, 22], one possible criticism to our results is that there are other definitions of Shapley values besides the one being used. One example is the use of baselines [334, 201]. Our initial experiments suggest that the use of baselines is even more problematic than the original definitions of Shapley for XAI. Concretely, the percentages of detected issues for Boolean classifiers far exceed those reported in earlier work [173]. Future work will build on these initial experiments, and will document the issues that are also observed when using Shapley values for XAI based on baselines.

**Shapley values for XAI unrelated with formal explanations.** One additional criticism to the results in this chapter and earlier reports [173, 175, 174] is that the fact that Shapley values and SHAP scores for XAI do not capture feature relevancy is not problematic per se, and it might be the case that we could be talking about different and unrelated measures of feature importance, one provided by feature attribution and the other provided by feature selection. As shown in this chapter, we can construct classifiers with features that are of paramount importance for a prediction, but that are assigned a SHAP score of 0 (i.e. denoting no importance whatsoever for the prediction). Similarly, we can construct classifiers (actually the same classifier can be used!) with features that serve no purpose in terms of explanations, and that also serve no purpose in terms of creating adversarial examples, but which are assigned the largest absolute SHAP score. In such circumstances, it would be perplexing if there could exist some ascribed meaning to computed SHAP scores such that the information they convey would not be misleading for human decision makers.

## 7.8 Summary

For more than a decade Shapley values have represented one of the most visible approaches for feature attribution in explainability.

This chapter gives theoretical arguments as well as experimental results to the fact that SHAP scores for explainability can produce misleading information about the relative importance of features. This chapter distinguishes between the features that occur in one or more of the irreducible rule-based explanations, i.e. the *relevant features*, from those that do not occur in any irreducible rule-based explanation, i.e. the *irrelevant features*. This chapter proves that, for boolean functions with arbitrary number of variables, irrelevant features can be deemed more important, given their SHAP scores, than relevant features. Our results are also significant

in practical deployment of explainability solutions. Indeed, misleading information about relative feature importance can induce human decision makers in error, by persuading them to look at the wrong causes of predictions.

Furthermore, this chapter shows that the relative order of feature importance obtained with SHAP scores for XAI does not correlate with the features that can serve for producing $l_0$-minimal adversarial examples, i.e. those that are sufficiently close to the original instance. Thus, besides SHAP scores for XAI not being correlated with feature relevancy, it is also the case that SHAP scores for XAI do not relate with $l_0$-minimal adversarial examples.

The significance of our results should be framed in light of the rapid growth of practical uses of explainability methods based on SHAP scores, with one concrete example being the medical domain, of which [200, 367, 356, 191, 270, 34, 11, 376, 222, 6, 331, 375, 248, 348, 240, 241, 380, 166, 2] represent a fraction of the many existing examples. And given the results in this chapter, the use of SHAP scores as a measure of feature importance should be expected to mislead decision makers when assessing the features that impact some predictions.

Chapter 8

# Conclusions and Future Work

## Conclusions

The explainability of machine learning models has become a crucial area of study, given their expanding applications in various fields like healthcare, finance, and law. Ensuring the trustworthiness of machine learning systems relies on the ability to provide explanations for their decisions. In contrast to well-known non-formal explanation approaches, *formal explainability* has emerged as a promising research area. It aims to provide rigorous and provable explanations for machine learning model predictions.

This thesis provides an overview of recent advances in formal explainability. It contributes multiple theoretical results and practical efficient algorithms in formal explainability. The main contributions of this thesis can be summarized as follows:

- In Chapter 3, we proposed a novel framework called explanation graphs (XpG's), which enables the computation of formal explanations in polynomial time for tractable decision graph models. This includes decision trees, binary decision diagrams, and multi-valued decision diagrams. Furthermore, we proposed a practically efficient solution for the enumeration of explanations. Additionally, for the concrete case of decision trees, we showed that the set of all contrastive explanations can be enumerated in polynomial time.

- In Chapter 4, we identified conditions enabling the computation of formal explanations in polynomial time for classifiers represented as tractable boolean circuits. This includes well-known d-DNNF circuits and any other tractable boolean circuit that is strictly less succinct than d-DNNF. Furthermore, we also identified conditions under which the polynomial time computation of explanations can be extended to boolean circuits that are more succinct than d-DNNF.

- Chapter 5 introduced an application of formal explainability, we showed that one can construct a decision set from some of the decision tree explanations, such that the decision set is not only explained, but it also exhibits a number of properties that are critical for replacing the original decision tree.

- Apart from the computation of formal explanations, there are several additional explainability queries that are of interest. In Chapter 6, we studied two specific explainability queries: *feature necessity* and *feature relevancy*. These queries, in general, inquire whether a user-interested feature is included in

formal explanations and, if so, how it is included. We proved the computational complexity of these problems with respect to a wide range of classifiers. Additionally, we proposed algorithms for their solution in practice.

- One of the hallmarks of XAI are measures of relative feature importance, which are theoretically justified through the use of Shapley values. In Chapter 7, we illustrate, both theoretically and empirically, that utilizing SHAP scores for explainability will yield misleading information about the relative importance of features for predictions.

## Future Research

Even though this thesis introduces several advances in formal explainability, various questions remain to be addressed in future work. We give some directions for future investigation.

**Enhancing Scalability of Formal XAI Methods.** Due to the rapidly growing demand for deploying large-scale ML systems in various fields, the scalability of XAI methods has become a significant concern. However, formal explainability is hindered by poor scalability for some families of classifiers, the most significant being neural networks. As a result, there are concerns as to whether formal explainability might serve to complement other approaches in delivering trustworthy AI. Recent work [36] proposes a novel approach to approximate formal explanations by leveraging technologies for assessing the robustness of deep neural networks. (Analysis of robustness is motivated by the existence of adversarial examples in complex ML model [335].) Besides, recent work [172] addresses the limitation of scalability of formal explainability, and proposes novel algorithms for computing formal explanations. Motivated by these works, one future research direction will be to develop practical and efficient approaches for computing rigorous explanations for complex machine learning models by leveraging off-the-shelf robustness tools [211, 377, 360, 359, 336, 157, 158, 119].

**Comparative Studies Between SHAP scores and Formal Explanations.** Building upon the findings of Chapter 7, one future research direction will be to conduct a more extensive and detailed comparative analysis of similarities and differences between SHAP scores and formal explanations. Another direction of research is to develop a better understanding of the distributions of functions exhibiting one or more of the issues of SHAP scores. Furthermore, recent work [369, 368] proposes a way for applying formal XAI to the case of feature attribution, leveraging the enumeration of formal explanations. One direction of future research is to devise other measures of relative importance that might serve as alternatives to the use of SHAP scores.

# Bibliography


[1] R. Abduljabbar, H. Dia, S. Liyanage, and S. A. Bagloee. Applications of artificial intelligence in transport: An overview. *Sustainability*, 11(1):189, 2019. (Cited on page 2.)

[2] J. Adeoye, L.-W. Zheng, P. Thomson, S.-W. Choi, and Y.-X. Su. Explainable ensemble learning model improves identification of candidates for oral cancer screening. *Oral Oncology*, 136:106278, 2023. (Cited on pages 100 and 133.)

[3] D. Afchar, V. Guigue, and R. Hennequin. Towards rigorous interpretations: a formalisation of feature attribution. In *ICML*, pages 76–86, 2021. (Cited on page 101.)

[4] M. R. AI4Science and M. A. Quantum. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023. (Cited on page 1.)

[5] S. B. Akers. Binary decision diagrams. *IEEE Transactions on computers*, 27(06):509–516, 1978. (Cited on pages 9 and 10.)

[6] R. O. Alabi, A. Almangush, M. Elmusrati, I. Leivo, and A. A. Mäkitie. An interpretable machine learning prognostic system for risk stratification in oropharyngeal cancer. *International Journal of Medical Informatics*, 168:104896, 2022. (Cited on pages 100 and 133.)

[7] M. S. Alam and Y. Xie. Appley: Approximate Shapley value for model explainability in linear time. In *Big Data*, pages 95–100, 2022. (Cited on page 100.)

[8] E. Albini, J. Long, D. Dervovic, and D. Magazzeni. Counterfactual Shapley additive explanations. In *FAccT*, pages 1054–1070, 2022. (Cited on page 100.)

[9] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17(2-3):255–287, 2011. (Cited on page 89.)

[10] E. Algaba, V. Fragnelli, and J. Sánchez-Soriano. *Handbook of the Shapley value*. CRC Press, 2019. (Cited on page 16.)

[11] B. Alsinglawi, O. Alshari, M. Alorjani, O. Mubin, F. Alnajjar, M. Novoa, and O. Darwish. An explainable machine learning framework for lung cancer hospital length of stay prediction. *Scientific reports*, 12(1):1–10, 2022. (Cited on pages 100 and 133.)

[12] L. Amgoud. Non-monotonic explanation functions. In *ECSQARU*, pages 19–31, 2021. (Cited on pages 4 and 64.)

[13] L. Amgoud and J. Ben-Naim. Axiomatic foundations of explainability. In *IJCAI*, pages 636–642, 2022. (Cited on page 64.)

[14] J. Amilhastre, H. Fargier, and P. Marquis. Consistency restoration and explanations in dynamic CSPs application to configuration. *Artif. Intell.*, 135(1-2):199–234, 2002. (Cited on page 38.)

[15] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021. (Cited on page 3.)

[16] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. Machine bias. 2016. (Cited on page 2.)

[17] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling. Explainable agents and robots: Results from a systematic literature review. In *AAMAS*, pages 1078–1088, 2019. (Cited on page 36.)

[18] D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, 2020. (Cited on pages 3 and 15.)

[19] M. Arenas, D. Baez, P. Barceló, J. Pérez, and B. Subercaseaux. Foundations of symbolic languages for model interpretability. In *NeurIPS*, 2021. (Cited on page 64.)

[20] 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. (Cited on pages 17, 23, 99, 126 and 129.)

[21] 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, 2021. (Cited on pages 17, 23, 97, 99, 100 and 132.)

[22] 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. *J. Mach. Learn. Res.*, 24:63:1–63:58, 2023. (Cited on pages 17, 97, 99, 100, 126, 129 and 132.)

[23] M. Arenas, P. Barceló, M. Romero, and B. Subercaseaux. On computing probabilistic explanations for decision trees. In *NeurIPS*, 2022. (Cited on pages 51, 54, 58 and 64.)

[24] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. (Cited on page 12.)

[25] H. Arslanian and F. Fischer. *The future of finance: The impact of FinTech, AI, and crypto on financial services*. Springer, 2019. (Cited on page 2.)

[26] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis. On the computational intelligibility of boolean classifiers. In *KR*, pages 74–86, 2021. (Cited on pages 21, 23, 36, 37, 38, 50, 51, 64 and 65.)

[27] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis. On preferred abductive explanations for decision trees and random forests. In *IJCAI*, pages 643–650, 2022. (Cited on page 64.)

[28] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis. Trading complexity for sparsity in random forest explanations. In *AAAI*, pages 5461–5469, 2022. (Cited on page 64.)

[29] G. Audemard, F. Koriche, and P. Marquis. On tractable XAI queries based on compiled representations. In *KR*, pages 838–849, 2020. (Cited on pages 21, 22, 23, 36, 37, 38, 64, 65 and 76.)

[30] G. Audemard and L. Simon. On the glucose SAT solver. *Int. J. Artif. Intell. Tools*, 27(1):1840001:1–1840001:25, 2018. (Cited on page 85.)

[31] Australian Gov. Australia's AI action plan. tiny.cc/hy8juz, 2021. (Cited on page 2.)

[32] Australian Gov. Australias artificial intelligence ethics framework. `tiny.cc/ey8juz`, 2021. (Cited on page 2.)

[33] R. R. Bakker, F. Dikker, F. Tempelman, and P. M. Wognum. Diagnosing and solving over-determined constraint satisfaction problems. In *IJCAI*, pages 276–281, 1993. (Cited on page 40.)

[34] M. L. Baptista, K. Goebel, and E. M. Henriques. Relation between prognostics predictor evaluation metrics and local interpretability SHAP values. *Artificial Intelligence*, 306:103667, 2022. (Cited on pages 100 and 133.)

[35] P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux. Model interpretability through the lens of computational complexity. In *NeurIPS*, 2020. (Cited on pages 22, 23, 36, 37, 38, 47, 50, 51, 57 and 64.)

[36] S. Bassan and G. Katz. Towards formal xai: Formally approximate minimal explanations of neural networks. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 187–207. Springer, 2023. (Cited on page 135.)

[37] A. Behrouz, M. Lécuyer, C. Rudin, and M. Seltzer. Fast optimization of weighted sparse decision trees for use in optimal treatment regimes and optimal policy design. In *CEUR workshop proceedings*, volume 3318. NIH Public Access, 2022. (Cited on page 5.)

[38] J. Bekker, J. Davis, A. Choi, A. Darwiche, and G. V. den Broeck. Tractable learning for complex probability queries. In *NeurIPS*, pages 2242–2250, 2015. (Cited on page 88.)

[39] J. L. Bell and M. Machover. *A course in mathematical logic*. Elsevier, 1977. (Cited on page 7.)

[40] M. Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012. (Cited on page 7.)

[41] Y. Bengio, Y. LeCun, and G. E. Hinton. Deep learning for AI. *Commun. ACM*, 64(7):58–65, 2021. (Cited on pages 1 and 49.)

[42] C. Berge. *Hypergraphs: combinatorics of finite sets*. Elsevier, 1984. (Cited on page 11.)

[43] D. Bertsimas and J. Dunn. Optimal classification trees. *Mach. Learn.*, 106(7):1039–1082, 2017. (Cited on page 32.)

[44] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009. (Cited on page 7.)

[45] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021. (Cited on pages 5 and 12.)

[46] A. Biere, F. Lonsing, and M. Seidl. Blocked clause elimination for QBF. In *CADE*, pages 101–115, 2011. (Cited on page 86.)

[47] E. Birnbaum and E. L. Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46, 2003. (Cited on page 11.)

[48] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. (Cited on page 12.)

[49] G. Blanc, J. Lange, and L. Tan. Provably efficient, succinct, and precise explanations. In *NeurIPS*, 2021. (Cited on page 64.)

[50] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987. (Cited on page 99.)

[51] F. Bodria, S. Rinzivillo, D. Fadda, R. Guidotti, F. Giannotti, and D. Pedreschi. Explaining Black Box with Visual Exploration of Latent Space. In *EuroVis – Short Papers*, 2022. (Cited on page 100.)

[52] B. Bogaerts, E. Gamba, J. Claes, and T. Guns. Step-wise explanations of constraint satisfaction problems. In *ECAI*, pages 640–647, 2020. (Cited on page 38.)

[53] G. Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*, volume 2. Walton and Maberly, 1854. (Cited on page 9.)

[54] 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. (Cited on page 76.)

[55] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic minimization algorithms for VLSI synthesis*, volume 2. Springer Science & Business Media, 1984. (Cited on page 9.)

[56] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001. (Cited on pages 14 and 15.)

[57] L. Breiman. Statistical modeling: The two cultures. *Statistical science*, 16(3):199–231, 2001. (Cited on pages 22 and 49.)

[58] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. (Cited on page 1.)

[59] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986. (Cited on pages 9 and 10.)

[60] S. R. Buss. *Handbook of proof theory*. Elsevier, 1998. (Cited on page 7.)

[61] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018. (Cited on page 1.)

[62] O. Camburu, E. Giunchiglia, J. Foerster, T. Lukasiewicz, and P. Blunsom. Can I trust the explainer? verifying post-hoc explanatory methods. *CoRR*, abs/1910.02065, 2019. (Cited on pages 4, 18 and 21.)

[63] O.-M. Camburu, E. Giunchiglia, J. Foerster, T. Lukasiewicz, and P. Blunsom. The struggles of feature-based explanations: Shapley values vs. minimal sufficient subsets. *arXiv preprint arXiv:2009.11023*, 2020. (Cited on page 101.)

[64] T. W. Campbell, H. Roder, R. W. Georgantas III, and J. Roder. Exact Shapley values for local and model-true explanations of decision tree ensembles. *Machine Learning with Applications*, 9:100345, 2022. (Cited on page 101.)

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

[66] Y. Chang, X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109*, 2023. (Cited on page 1.)

[67] C. Chen, K. Lin, C. Rudin, Y. Shaposhnik, S. Wang, and T. Wang. A holistic approach to interpretability in financial lending: Models, visualizations, and summary-explanations. *Decision Support Systems*, 152:113647, 2022. (Cited on page 5.)

[68] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan. L-Shapley and C-Shapley: Efficient model interpretation for structured data. In *ICLR*, 2019. (Cited on pages 16 and 100.)

[69] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016. (Cited on page 15.)

[70] J. W. Chinneck and E. W. Dravnieks. Locating minimal infeasible constraint sets in linear programs. *INFORMS J. Comput.*, 3(2):157–168, 1991. (Cited on pages 29 and 40.)

[71] A. Choi and A. Darwiche. Sdd advanced-user manual version 2.0. *Automated Reasoning Group, UCLA*, 2018. (Cited on page 10.)

[72] A. Choi, A. Shih, A. Goyanka, and A. Darwiche. On symbolically encoding the behavior of random forests. *CoRR*, abs/2007.01493, 2020. (Cited on pages 4 and 76.)

[73] Y. Choi, A. Darwiche, and G. Van den Broeck. Optimal feature selection for decision robustness in bayesian networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017. (Cited on page 10.)

[74] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989. (Cited on page 51.)

[75] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003. (Cited on pages 65 and 79.)

[76] M. Coeckelbergh. *AI ethics*. Mit Press, 2020. (Cited on page 2.)

[77] W. W. Cohen. Efficient pruning methods for separate-and-conquer rule learning systems. In *IJCAI*, pages 988–994, 1993. (Cited on page 51.)

[78] W. W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995. (Cited on page 51.)

[79] W. W. Cohen and Y. Singer. A simple, fast, and effictive rule learner. In *AAAI*, pages 335–342, 1999. (Cited on page 51.)

[80] S. A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971. (Cited on page 12.)

[81] M. C. Cooper and J. Marques-Silva. On the tractability of explaining decisions of classifiers. In L. D. Michel, editor, *CP*, pages 21:1–21:18, 2021. (Cited on pages 19, 38 and 64.)

[82] M. T. Correia Ribeiro. *Model-Agnostic Explanations and Evaluation of Machine Learning*. PhD thesis, 2018. (Cited on pages 16 and 18.)

[83] I. Covert and S. Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *AISTATS*, pages 3457–3465, 2021. (Cited on page 100.)

[84] I. Covert, S. M. Lundberg, and S. Lee. Understanding global feature contributions with additive importance measures. In *NeurIPS*, 2020. (Cited on pages 97 and 100.)

[85] Y. Crama and P. L. Hammer. *Boolean functions: Theory, algorithms, and applications*. Cambridge University Press, 2011. (Cited on pages 7 and 9.)

[86] Y. Crama and P. L. Hammer. *Boolean Functions: Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2011. (Cited on page 28.)

[87] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. (Cited on page 1.)

[88] M. Daily, S. Medasani, R. Behringer, and M. Trivedi. Self-driving cars. *Computer*, 50(12):18–23, 2017. (Cited on pages 4 and 18.)

[89] H. Daniels and M. Velikova. Monotone and partially monotone neural networks. *IEEE Trans. Neural Networks*, 21(6):906–917, 2010. (Cited on page 89.)

[90] DARPA. DARPA explainable Artificial Intelligence (XAI) program. `https://www.darpa.mil/program/explainable-artificial-intelligence`, 2016. (Cited on page 2.)

[91] A. Darwiche. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Appl. Non Class. Logics*, 11(1-2):11–34, 2001. (Cited on pages 7, 9 and 10.)

[92] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009. (Cited on page 22.)

[93] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, pages 819–826, 2011. (Cited on pages 10, 42 and 47.)

[94] A. Darwiche. Three modern roles for logic in AI. In *PODS*, pages 229–243, 2020. (Cited on pages 4 and 64.)

[95] A. Darwiche. Logic for explainable AI. In *LICS*, pages 1–11, 2023. (Cited on pages 3 and 4.)

[96] A. Darwiche and A. Hirth. On the reasons behind decisions. In *ECAI*, pages 712–720, 2020. (Cited on pages 4, 21, 23, 36, 38 and 64.)

[97] A. Darwiche and A. Hirth. On the (complete) reasons behind decisions. *J. Log. Lang. Inf.*, 32(1):63–88, 2023. (Cited on page 4.)

[98] A. Darwiche and C. Ji. On the computation of necessary and sufficient explanations. In *AAAI*, pages 5582–5591, 2022. (Cited on page 64.)

[99] A. Darwiche and P. Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002. (Cited on pages 4, 7, 9, 10, 11, 23, 36, 37, 42 and 47.)

[100] A. Darwiche and P. Marquis. On quantifying literals in boolean logic and its applications to explainable AI. *J. Artif. Intell. Res.*, 2021. (Cited on page 64.)

[101] A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *IEEE S&P*, pages 598–617, 2016. (Cited on pages 3, 16 and 100.)

[102] C. O. de Burgh-Day and T. Leeuwenburg. Machine learning for numerical weather and climate modelling: a review. *Geoscientific Model Development*, 16(22):6433–6477, 2023. (Cited on page 1.)

[103] A. DeArman. The wild, wild west: A case study of self-driving vehicle testing in arizona. *Ariz. L. Rev.*, 61:983, 2019. (Cited on page 2.)

[104] J. Demšar, T. Curk, A. Erjavec, Črt Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013. (Cited on pages 32, 47, 59 and 126.)

[105] G. V. den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the tractability of SHAP explanations. In *AAAI*, pages 6505–6513, 2021. (Cited on pages 17, 23, 97, 99, 100 and 132.)

[106] G. V. den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the tractability of SHAP explanations. *J. Artif. Intell. Res.*, 74:851–886, 2022. (Cited on pages 17, 23, 97, 99, 100 and 132.)

[107] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. (Cited on page 1.)

[108] B. Dimanov, U. Bhatt, M. Jamnik, and A. Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *ECAI*, pages 2473–2480, 2020. (Cited on pages 4, 18 and 21.)

[109] D. Dua and C. Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2017. (Cited on pages 32 and 47.)

[110] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995. (Cited on pages 65 and 131.)

[111] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304, 1995. (Cited on page 11.)

[112] EU. Artificial Intelligence Act. `http://tiny.cc/ahcnuz`, 2021. (Cited on pages 2, 64 and 65.)

[113] EU. Coordinated plan on artificial intelligence – 2021 review. `https://bit.ly/3hJG2HF`, 2021. (Cited on page 2.)

[114] European Commission. General Data Protection Regulation. `http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=en`, 2016. Accessed: 2023-03-23. (Cited on pages 2 and 65.)

[115] European Commission's High-Level Expert Group on AI. Ethics guidelines for trustworthy AI. `https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai`, 2019. Accessed: 2023-03-23. (Cited on page 65.)

[116] M. A. Falappa, G. Kern-Isberner, and G. R. Simari. Explanations, belief revision and defeasible reasoning. *Artif. Intell.*, 141(1/2):1–28, 2002. (Cited on pages 21 and 38.)

[117] M. M. Fard, K. R. Canini, A. Cotter, J. Pfeifer, and M. R. Gupta. Fast and flexible monotonic functions with ensembles of lattices. In *NeurIPS*, pages 2919–2927, 2016. (Cited on page 15.)

[118] H. Fargier and P. Marquis. Extending the knowledge compilation map: Krom, horn, affine and beyond. In *AAAI*, pages 442–447, 2008. (Cited on page 42.)

[119] C. Ferrari, M. N. Mueller, N. Jovanović, and M. Vechev. Complete verification via multi-neuron relaxation guided branch-and-bound. In *International Conference on Learning Representations*, 2021. (Cited on page 135.)

[120] 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 432442, 2022. (Cited on page 64.)

[121] V. Fishman, M. Sindeeva, N. Chekanov, T. Shashkova, N. Ivanisenko, and O. Kardymon. Ai in genomics and epigenomics. In *Artificial Intelligence for Healthy Longevity*, pages 217–243. Springer, 2023. (Cited on page 1.)

[122] M. L. Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms*, 21(3):618–628, 1996. (Cited on page 32.)

[123] A. A. Freitas. Comprehensible classification models: a position paper. *SIGKDD Explorations*, 15(1):1–10, 2013. (Cited on page 22.)

[124] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. (Cited on page 14.)

[125] G. Friedrich, G. Gottlob, and W. Nejdl. Hypothesis classification, abductive diagnosis and therapy. In *ESE*, pages 69–78, 1990. (Cited on page 65.)

[126] C. Frye, D. de Mijolla, T. Begley, L. Cowton, M. Stanley, and I. Feige. Shapley explainability on the data manifold. In *ICLR*, 2021. (Cited on page 100.)

[127] C. Frye, C. Rowat, and I. Feige. Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability. In *NeurIPS*, 2020. (Cited on page 100.)

[128] D. V. Fryer, I. Strümke, and H. D. Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. *IEEE Access*, 9:144352–144360, 2021. (Cited on page 101.)

[129] J. Fürnkranz. Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 13(1):3–54, 1999. (Cited on page 51.)

[130] J. Fürnkranz, D. Gamberger, and N. Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, 2012. (Cited on page 51.)

[131] J. Fürnkranz and T. Kliegr. A brief overview of rule learning. In *RuleML*, pages 54–69, 2015. (Cited on page 51.)

[132] I. B. Galazzo, F. Cruciani, L. Brusini, A. M. A. Salih, P. Radeva, S. F. Storti, and G. Menegaz. Explainable artificial intelligence for magnetic resonance imaging aging brainprints: Grounds and challenges. *IEEE Signal Process. Mag.*, 39(2):99–116, 2022. (Cited on page 100.)

[133] E. Gamba, B. Bogaerts, and T. Guns. Efficiently explaining CSPs with unsatisfiable subset optimization. In *IJCAI*, pages 1381–1388, 2021. (Cited on page 38.)

[134] M. Gandolfi, I. B. Galazzo, R. G. Pavan, F. Cruciani, N. Valè, A. Picelli, S. F. Storti, N. Smania, and G. Menegaz. eXplainable AI allows predicting upper limb rehabilitation outcomes in sub-acute stroke patients. *IEEE J. Biomed. Health Informatics*, 27(1):263–273, 2023. (Cited on page 100.)

[135] G. Gange and P. J. Stuckey. Explaining propagators for s-dnnf circuits. In *CPAIOR*, pages 195–210, 2012. (Cited on page 38.)

[136] W. Gao and Z. Zhou. Towards convergence rate analysis of random forests for classification. In *NeurIPS*, 2020. (Cited on page 14.)

[137] J. Gergov and C. Meinel. *Efficient analysis and manipulation of OBDDs can be extended to read-once-only branching programs.* Univ. Trier, 1992. (Cited on page 9.)

[138] B. Ghosh, D. Malioutov, and K. S. Meel. Classification rules in relaxed logical form. In *ECAI*, pages 2489–2496, 2020. (Cited on page 60.)

[139] B. Ghosh, D. Malioutov, and K. S. Meel. Efficient learning of interpretable classification rules. *J. Artif. Intell. Res.*, 74:1823–1863, 2022. (Cited on page 60.)

[140] B. Ghosh and K. S. Meel. IMLI: an incremental framework for MaxSAT-based learning of interpretable classification rules. In *AIES*, pages 203–210, 2019. (Cited on page 60.)

[141] J. Goldsmith, M. Hagen, and M. Mundhenk. Complexity of DNF and isomorphism of monotone formulas. In *MFCS*, pages 410–421, 2005. (Cited on page 28.)

[142] J. Goldsmith, M. Hagen, and M. Mundhenk. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Inf. Comput.*, 206(6):760–775, 2008. (Cited on page 28.)

[143] J. W. Goodell, S. Kumar, W. M. Lim, and D. 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. (Cited on page 2.)

[144] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. (Cited on page 99.)

[145] N. Gorji and S. Rubin. Sufficient reasons for classifier decisions in the presence of domain constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5660–5667, 2022. (Cited on pages 57 and 64.)

[146] G. Gottlob and E. Malizia. Achieving new upper bounds for the hypergraph duality problem through logic. *SIAM J. Comput.*, 47(2):456–492, 2018. (Cited on page 11.)

[147] R. Guidotti, A. Monreale, S. Ruggieri, F. Naretto, F. Turini, D. Pedreschi, and F. Giannotti. Stable and actionable explanations of black-box models through factual and counterfactual rules. *Data Mining and Knowledge Discovery*, pages 1–38, 2022. (Cited on page 100.)

[148] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019. (Cited on pages 4, 15, 18, 21, 22, 36, 37 and 49.)

[149] D. Gunning and D. W. Aha. Darpa's explainable artificial intelligence (XAI) program. *AI Mag.*, 40(2):44–58, 2019. (Cited on page 21.)

[150] V. Gurvich and L. Khachiyan. On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions. *Discret. Appl. Math.*, 96-97:363–373, 1999. (Cited on page 28.)

[151] J. V. Haaren and J. Davis. Markov network structure learning: A randomized feature generation approach. In *AAAI*, 2012. (Cited on page 88.)

[152] B. Haibe-Kains, G. A. Adam, A. Hosny, F. Khodakarami, M. A. Q. C. M. S. B. 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, L. Waldron, B. Wang, C. McIntosh, A. Goldenberg, A. Kundaje, et al. Transparency and reproducibility in artificial intelligence. *Nature*, 586(7829):E14–E16, 2020. (Cited on page 2.)

[153] R. W. Hamming. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160, 1950. (Cited on pages 99 and 106.)

[154] C. Hartshorne and P. Weiss, editors. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931. (Cited on page 131.)

[155] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. (Cited on page 12.)

[156] A. Haydari and Y. Yılmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):11–32, 2020. (Cited on page 2.)

[157] P. Henriksen and A. Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *ECAI 2020*, pages 2513–2520. IOS Press, 2020. (Cited on page 135.)

[158] P. Henriksen and A. Lomuscio. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In *IJCAI*, pages 2549–2555, 2021. (Cited on page 135.)

[159] HLEG AI. Ethics guidelines for trustworthy AI. `https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai`, 2019. (Cited on page 2.)

[160] HLEG AI. Assessment list for trustworthy artificial intelligence (ALTAI) for self-assessment. `https://bit.ly/3jAeHds`, 2020. (Cited on page 2.)

[161] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek. Explainable AI methods - A brief overview. In *xxAI*, pages 13–38, 2020. (Cited on page 3.)

[162] J.-W. Hong, I. Cruz, and D. Williams. Ai, you can drive my car: How we evaluate human drivers vs. self-driving cars. *Computers in Human Behavior*, 125:106944, 2021. (Cited on pages 4 and 18.)

[163] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012. (Cited on page 99.)

[164] X. Hu, Y. Huang, B. Li, and T. Lu. Uncovering the source of machine bias. *arXiv preprint arXiv:2201.03092*, 2022. (Cited on page 2.)

[165] J. Huang and A. Darwiche. The language of search. *J. Artif. Intell. Res.*, 29:191–219, 2007. (Cited on pages 7, 9 and 10.)

[166] T. Huang, D. Le, L. Yuan, S. Xu, and X. Peng. Machine learning for prediction of in-hospital mortality in lung cancer patients admitted to intensive care unit. *Plos one*, 18(1):e0280606, 2023. (Cited on pages 100 and 133.)

[167] X. Huang, M. C. Cooper, A. Morgado, J. Planes, and J. Marques-Silva. Feature necessity & relevancy in ML classifier explanations. In *TACAS*, pages 167–186. Springer, 2023. (Cited on pages 4 and 98.)

[168] 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. (Cited on pages 4, 64 and 75.)

[169] X. Huang, Y. Izza, A. Ignatiev, and J. Marques-Silva. On efficiently explaining graph-based classifiers. In *KR*, pages 356–367, 2021. (Cited on pages 4, 38, 51, 57, 64, 65, 75, 123, 126 and 129.)

[170] X. Huang, Y. Izza, and J. Marques-Silva. Solving explainability queries with quantification: The case of feature relevancy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3996–4006, 2023. (Cited on page 4.)

[171] X. Huang and J. Marques-Silva. On deciding feature membership in explanations of SDD & related classifiers. *CoRR*, abs/2202.07553, 2022. (Cited on page 64.)

[172] X. Huang and J. Marques-Silva. From robustness to explainability and back again. *arXiv preprint arXiv:2306.03048*, 2023. (Cited on pages 5 and 135.)

[173] X. Huang and J. Marques-Silva. The inadequacy of shapley values for explainability. *CoRR*, abs/2302.08160, 2023. (Cited on pages 4, 18, 98, 101, 103, 131 and 132.)

[174] X. Huang and J. Marques-Silva. A refutation of shapley values for explainability. *arXiv preprint arXiv:2309.03041*, 2023. (Cited on pages 4, 18, 98, 131 and 132.)

[175] X. Huang and J. Marques-Silva. Refutation of shapley values for xai – additional evidence. 2023. (Cited on pages 4, 18, 98, 131 and 132.)

[176] E. Hüllermeier, J. Fürnkranz, E. L. Mencía, V. Nguyen, and M. Rapp. Rule-based multi-label classification: Challenges and opportunities. In *RuleML+RR*, pages 3–19, 2020. (Cited on page 51.)

[177] IAI. Interpretable AI. `https://www.interpretable.ai/`, 2020. (Cited on page 32.)

[178] A. Ignatiev. Towards trustable explainable AI. In *IJCAI*, pages 5154–5158, 2020. (Cited on pages 3, 4, 15, 18, 21, 22, 38 and 64.)

[179] A. Ignatiev, M. C. Cooper, M. Siala, E. Hebrard, and J. Marques-Silva. Towards formal fairness in machine learning. In *CP*, pages 846–867, 2020. (Cited on pages 38 and 64.)

[180] 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. (Cited on pages 14, 19, 38 and 64.)

[181] A. Ignatiev, E. Lam, P. J. Stuckey, and J. Marques-Silva. A scalable two stage approach to computing optimal decision sets. In *AAAI*, pages 3806–3814, 2021. (Cited on page 60.)

[182] A. Ignatiev and J. Marques-Silva. SAT-based rigorous explanations for decision lists. In *SAT*, pages 251–269, 2021. (Cited on pages 4, 14, 19, 20, 22, 38, 50, 51 and 64.)

[183] A. Ignatiev, J. Marques-Silva, N. Narodytska, and P. J. Stuckey. Reasoning-based learning of interpretable ML models. In *IJCAI*, pages 4458–4465, 2021. (Cited on page 4.)

[184] A. Ignatiev, A. Morgado, and J. Marques-Silva. PySAT: A python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018. (Cited on pages 33, 47 and 85.)

[185] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. From contrastive to abductive explanations and back again. In *AIxIA*, pages 335–355, 2020. (Cited on pages 20, 22, 31, 38, 43, 64 and 100.)

[186] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. On relating 'why?' and 'why not?' explanations. *CoRR*, abs/2012.11067, 2020. (Cited on page 20.)

[187] A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019. (Cited on pages 4, 19, 21, 36, 38, 64 and 65.)

[188] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On relating explanations and adversarial examples. In *NeurIPS*, pages 15857–15867, 2019. (Cited on pages 19, 21, 38, 64 and 100.)

[189] A. Ignatiev, N. Narodytska, and J. Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019. (Cited on pages 4, 18, 21, 38 and 64.)

[190] A. Ignatiev, F. Pereira, N. Narodytska, and J. Marques-Silva. A SAT-based approach to learn explainable decision sets. In *IJCAR*, pages 627–645, 2018. (Cited on pages 14, 44, 51 and 60.)

[191] T. Inoguchi, Y. Nohara, C. Nojiri, and N. Nakashima. Association of serum bilirubin levels with risk of cancer development and total death. *Scientific reports*, 11(1):1–12, 2021. (Cited on pages 100 and 133.)

[192] Y. Izza, X. Huang, A. Ignatiev, N. Narodytska, M. Cooper, and J. Marques-Silva. On computing probabilistic abductive explanations. *International Journal of Approximate Reasoning*, 159:108939, 2023. (Cited on pages 4, 12, 50, 53, 54, 55 and 58.)

[193] Y. Izza, A. Ignatiev, and J. Marques-Silva. On explaining decision trees. *CoRR*, abs/2010.11034, 2020. (Cited on pages 5, 19, 20, 22, 23, 38, 51, 55, 57 and 64.)

[194] Y. Izza, A. Ignatiev, and J. Marques-Silva. On tackling explanation redundancy in decision trees. *J. Artif. Intell. Res.*, 75:261–321, 2022. (Cited on pages 5, 12, 13, 18, 49, 50, 51, 53, 55, 56, 57, 64, 123, 126 and 129.)

[195] Y. Izza, A. Ignatiev, and J. Marques-Silva. On tackling explanation redundancy in decision trees. *CoRR*, abs/2205.09971, 2022. (Cited on page 13.)

[196] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. Efficient explanations with relevant sets. *CoRR*, abs/2106.00546, 2021. (Cited on pages 22, 58 and 64.)

[197] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. Provably precise, succinct and efficient explanations for decision trees. *CoRR*, abs/2205.09569, 2022. (Cited on pages 12, 58 and 64.)

[198] Y. Izza and J. Marques-Silva. On explaining random forests with SAT. In *IJCAI*, pages 2584–2591, 2021. (Cited on pages 4, 15, 19, 20, 22, 38, 64, 75, 76 and 77.)

[199] Y. Izza and J. Marques-Silva. On computing relevant features for explaining NBCs. *CoRR*, abs/2207.04748, 2022. (Cited on pages 58 and 64.)

[200] T. Jansen, G. Geleijnse, M. Van Maaren, M. P. Hendriks, A. Ten Teije, and A. Moncada-Torres. Machine learning explainability in breast cancer survival. In *Digital Personalized Health and Medicine*, pages 307–311. IOS Press, 2020. (Cited on pages 100 and 133.)

[201] D. Janzing, L. Minorics, and P. Blöbaum. Feature relevance quantification in explainable AI: A causal problem. In *AISTATS*, pages 2907–2916, 2020. (Cited on pages 4, 18 and 132.)

[202] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001. (Cited on page 22.)

[203] N. Jethani, M. Sudarshan, I. C. Covert, S. Lee, and R. Ranganath. FastSHAP: Real-time Shapley value estimation. In *ICLR*, 2022. (Cited on page 100.)

[204] A. Jobin, M. Ienca, and E. Vayena. The global landscape of ai ethics guidelines. *Nature machine intelligence*, 1(9):389–399, 2019. (Cited on page 2.)

[205] U. Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004. (Cited on pages 29 and 40.)

[206] C. Kadow, D. M. Hall, and U. Ulbrich. Artificial intelligence reconstructs missing climate information. *Nature Geoscience*, 13(6):408–413, 2020. (Cited on page 1.)

[207] T. Y.-k. Kam and R. K. Brayton. Multi-valued decision diagrams. Technical Report UCB/ERL M90/125, University of California Berkeley, December 1990. (Cited on page 129.)

[208] A. Karimi, G. Barthe, B. Schölkopf, and I. Valera. A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Comput. Surv.*, 55(5):95:1–95:29, 2023. (Cited on page 20.)

[209] A. Karimi, B. Schölkopf, and I. Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *FAccT*, pages 353–362, 2021. (Cited on page 20.)

[210] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023. (Cited on page 1.)

[211] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*, pages 443–452. Springer, 2019. (Cited on page 135.)

[212] D. Kaur, S. Uslu, K. J. Rittichier, and A. Durresi. Trustworthy artificial intelligence: a review. *ACM Computing Surveys (CSUR)*, 55(2):1–38, 2022. (Cited on page 1.)

[213] D. J. Kavvadias and E. C. Stavropoulos. An efficient algorithm for the transversal hypergraph generation. *J. Graph Algorithms Appl.*, 9(2):239–264, 2005. (Cited on page 11.)

[214] L. Khachiyan, E. Boros, K. M. Elbassioni, and V. Gurvich. An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discret. Appl. Math.*, 154(16):2350–2372, 2006. (Cited on page 32.)

[215] R. Kohavi. Bottom-up induction of oblivious read-once decision graphs: Strengths and limitations. In *AAAI*, pages 613–618, 1994. (Cited on page 22.)

[216] F. Koriche, J. Lagniez, P. Marquis, and S. Thomas. Knowledge compilation for model counting: Affine decision trees. In *IJCAI*, pages 947–953, 2013. (Cited on page 42.)

[217] I. Kumar, C. Scheidegger, S. Venkatasubramanian, and S. A. Friedler. Shapley residuals: Quantifying the limits of the Shapley value for explanations. In *NeurIPS*, pages 26598–26608, 2021. (Cited on page 101.)

[218] I. E. Kumar, S. Venkatasubramanian, C. Scheidegger, and S. Friedler. Problems with shapley-value-based explanations as feature importance measures. In *International Conference on Machine Learning*, pages 5491–5500. PMLR, 2020. (Cited on pages 4 and 18.)

[219] I. E. Kumar, S. Venkatasubramanian, C. Scheidegger, and S. A. Friedler. Problems with Shapley-value-based explanations as feature importance measures. In *ICML*, pages 5491–5500, 2020. (Cited on page 101.)

[220] J. Kuppala, K. K. Srinivas, P. Anudeep, R. S. Kumar, and P. H. Vardhini. Benefits of artificial intelligence in the legal system and law enforcement. In *2022 International Mobile and Embedded Technology Conference (MECON)*, pages 221–225. IEEE, 2022. (Cited on page 2.)

[221] C. Labreuche. Explanation of pseudo-boolean functions using cooperative game theory and prime implicants. In *SUM*, pages 295–308, 2022. (Cited on page 101.)

[222] C. Ladbury, R. Li, J. Shiao, J. Liu, M. Cristea, E. Han, T. Dellinger, S. Lee, E. Wang, C. Fisher, et al. Characterizing impact of positive lymph node number in endometrial cancer using machine-learning: A better prognostic indicator than figo staging? *Gynecologic Oncology*, 164(1):39–45, 2022. (Cited on pages 100 and 133.)

[223] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684, 2016. (Cited on pages 5, 14, 37, 44, 51 and 60.)

[224] H. Lakkaraju and O. Bastani. "how do I fool you?": Manipulating user trust via misleading black box explanations. In *AIES*, pages 79–85, 2020. (Cited on pages 4, 18 and 21.)

[225] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019. (Cited on pages 3 and 15.)

[226] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *AISTATS*, pages 29–37, 2011. (Cited on page 88.)

[227] S. Larsson and F. Heintz. Transparency in artificial intelligence. *Internet Policy Review*, 9(2), 2020. (Cited on page 2.)

[228] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. (Cited on page 1.)

[229] V. M. Lelis, E. Guzmán, and M. Belmonte. Non-invasive meningitis diagnosis using decision trees. *IEEE Access*, 8:18394–18407, 2020. (Cited on pages 52, 123, 124 and 125.)

[230] B. Li, P. Qi, B. Liu, S. Di, J. Liu, J. Pei, J. Yi, and B. Zhou. Trustworthy ai: From principles to practices. *ACM Computing Surveys*, 55(9):1–46, 2023. (Cited on page 1.)

[231] M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2):223–250, 2016. (Cited on pages 7, 11, 22, 29, 30, 31, 37, 38, 42, 43 and 45.)

[232] M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reason.*, 40(1):1–33, 2008. (Cited on pages 7, 11 and 22.)

[233] J. Lind-Nielsen. Buddy : A binary decision diagram package. 1999. `http://buddy.sourceforge.net`. (Cited on page 32.)

[234] Z. C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018. (Cited on pages 22, 36 and 37.)

[235] J. Liu, C. Zhong, B. Li, M. Seltzer, and C. Rudin. Fasterrisk: fast and accurate interpretable risk scores. *Advances in Neural Information Processing Systems*, 35:17760–17773, 2022. (Cited on page 5.)

[236] X. Liu, X. Han, N. Zhang, and Q. Liu. Certified monotonic neural networks. In *NeurIPS*, 2020. (Cited on pages 15 and 89.)

[237] X. Liu and E. Lorini. A logic for binary classifiers and their explanation. In *CLAR*, 2021. (Cited on page 4.)

[238] X. Liu and E. Lorini. A logic of "black box" classifier systems. *CoRR*, abs/2210.07161, 2022. (Cited on page 20.)

[239] X. Liu and E. Lorini. A logic of "black box" classifier systems. In *WoLLIC*, pages 158–174, 2022. (Cited on page 64.)

[240] Y. Liu, Z. Liu, X. Luo, and H. Zhao. Diagnosis of parkinson's disease based on SHAP value feature selection. *Biocybernetics and Biomedical Engineering*, 42(3):856–869, 2022. (Cited on pages 100 and 133.)

[241] H. W. Loh, C. P. Ooi, S. Seoni, P. D. Barua, F. Molinari, and U. R. Acharya. Application of explainable artificial intelligence for healthcare: A systematic review of the last decade (2011-2022). *Comput. Methods Programs Biomed.*, 226:107161, 2022. (Cited on pages 100 and 133.)

[242] F. Lonsing and U. Egly. DepQBF 6.0: A search-based QBF solver beyond traditional QCDCL. In *CADE*, pages 371–384, 2017. (Cited on page 86.)

[243] D. Lowd and J. Davis. Learning markov network structure with decision trees. In *ICDM*, pages 334–343, 2010. (Cited on page 88.)

[244] O. Loyola-Gonzalez. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE access*, 7:154096–154113, 2019. (Cited on page 2.)

[245] S. Lundberg. *Explainable Machine Learning for Science and Medicine*. PhD thesis, 2019. (Cited on pages 16 and 17.)

[246] S. M. Lundberg, G. G. Erion, H. Chen, A. J. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.*, 2(1):56–67, 2020. (Cited on pages 97 and 100.)

[247] S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017. (Cited on pages 3, 4, 15, 16, 17, 18, 21, 23, 36, 64, 97, 100 and 132.)

[248] M. Ma, R. Liu, C. Wen, W. Xu, Z. Xu, S. Wang, J. Wu, D. Pan, B. Zheng, G. Qin, et al. Predicting the molecular subtype of breast cancer and identifying interpretable imaging features using machine learning algorithms. *European Radiology*, pages 1–11, 2022. (Cited on pages 100 and 133.)

[249] K. Makortoff. 'risks posed by ai are real': Eu moves to beat the algorithms that ruin lives. https://www.theguardian.com/technology/2022/aug/07/ai-eu-moves-to-beat-the-algorithms-that-ruin-lives, 2022. Accessed: 2023-03-23. (Cited on page 64.)

[250] 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. (Cited on pages 19, 20, 36, 38 and 64.)

[251] P. Malik, M. Pathania, V. K. Rathaur, et al. Overview of artificial intelligence in medicine. *Journal of family medicine and primary care*, 8(7):2328, 2019. (Cited on page 2.)

[252] D. Malioutov and K. S. Meel. MLIC: A maxsat-based framework for learning interpretable classification rules. In *CP*, pages 312–327, 2018. (Cited on page 60.)

[253] J. Marques-Silva. Logic-based explainability in machine learning. In *Reasoning Web*, pages 24–104, 2022. (Cited on pages 1, 2, 3, 4, 11, 14, 18, 54, 64 and 97.)

[254] J. Marques-Silva. Logic-based explainability in machine learning. *CoRR*, abs/2211.00541, 2022. (Cited on page 64.)

[255] 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. (Cited on pages 4, 19, 22, 23, 38 and 64.)

[256] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. Explanations for monotonic classifiers. In *ICML*, pages 7469–7479, 2021. (Cited on pages 4, 15, 19, 20, 22, 29, 38 and 64.)

[257] J. Marques-Silva and A. Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350, 2022. (Cited on pages 3, 4, 15, 18, 19, 38, 64 and 69.)

[258] J. Marques-Silva and A. Ignatiev. No silver bullet: interpretable ml models must be explained. *Frontiers in Artificial Intelligence*, 6, 2023. (Cited on pages 1, 3, 5, 11, 14, 18, 50 and 51.)

[259] J. Marques-Silva, M. Janota, and A. Belov. Minimal sets over monotone predicates in boolean formulae. In *CAV*, pages 592–607, 2013. (Cited on pages 29 and 40.)

[260] J. Marques-Silva, M. Janota, and C. Mencía. Minimal sets on propositional formulae. problems and reductions. *Artif. Intell.*, 252:22–50, 2017. (Cited on pages 29 and 40.)

[261] J. Marques-Silva and C. Mencía. Reasoning about inconsistent formulas. In *IJCAI*, pages 4899–4906, 2020. (Cited on page 11.)

[262] L. Merrick and A. Taly. The explanation game: Explaining machine learning models using Shapley values. In *CDMAKE*, pages 17–38, 2020. (Cited on pages 16, 100 and 101.)

[263] R. S. Michalski. On the quasi-minimal solution of the general covering problem. In *ISIP*, pages 125–128, 1969. (Cited on pages 5, 14 and 51.)

[264] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *AAAI*, pages 1041–1047, 1986. (Cited on pages 5, 14 and 51.)

[265] T. Miller. "but why?" understanding explainable artificial intelligence. *XRDS*, 25(3):20–25, 2019. (Cited on page 36.)

[266] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019. (Cited on pages 20, 22, 36 and 37.)

[267] B. D. Mittelstadt, C. Russell, and S. Wachter. Explaining explanations in AI. In *FAT*, pages 279–288, 2019. (Cited on page 36.)

[268] 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):529–533, 2015. (Cited on page 1.)

[269] C. Molnar. *Interpretable Machine Learning*. Leanpub, 2020. `http://tiny.cc/6c76tz`. (Cited on pages 3, 5, 15, 16, 18, 22, 49, 50 and 60.)

[270] A. Moncada-Torres, M. C. van Maaren, M. P. Hendriks, S. Siesling, and G. Geleijnse. Explainable machine learning can outperform cox regression predictions and provide insights in breast cancer survival. *Scientific reports*, 11(1):6968, 2021. (Cited on pages 100 and 133.)

[271] D. Monroe. Deceiving AI. *Commun. ACM*, 64, 2021. (Cited on pages 36 and 37.)

[272] G. Montavon, W. Samek, and K. Müller. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, 73:1–15, 2018. (Cited on page 22.)

[273] R. K. Mothilal, D. Mahajan, C. Tan, and A. Sharma. Towards unifying feature attribution and counterfactual explanations: Different means to the same end. In *AIES*, pages 652–663, 2021. (Cited on page 101.)

[274] S. T. Mueller, R. R. Hoffman, W. J. Clancey, A. Emrey, and G. Klein. Explanation in human-AI systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI. *CoRR*, abs/1902.01876, 2019. (Cited on page 36.)

[275] N. Narodytska, L. Ryzhyk, I. Ganichev, and S. Sevinc. BDD-based algorithms for packet classification. In *FMCAD*, pages 64–68, 2019. (Cited on page 32.)

[276] N. Narodytska, A. A. Shrotri, K. S. Meel, A. Ignatiev, and J. Marques-Silva. Assessing heuristic machine learning explanations with model counting. In *SAT*, pages 267–278, 2019. (Cited on pages 4, 18, 19, 21, 38 and 64.)

[277] National Science and Technology Council (US). Select Committee on Artificial Intelligence. The national artificial intelligence research and development strategic plan: 2019 update. 2019. https://www.nitrd.gov/pubs/National-AI-RD-Strategy-2019.pdf. (Cited on page 2.)

[278] A. Nikitas, K. Michalakopoulou, E. T. Njoya, and D. Karampatzakis. Artificial intelligence, transport and the smart city: Definitions and dimensions of a new mobility era. *Sustainability*, 12(7):2789, 2020. (Cited on page 2.)

[279] A. Niveau, H. Fargier, and C. Pralet. Representing CSPs with set-labeled diagrams: A compilation map. In *GKR*, pages 137–171, 2011. (Cited on pages 42 and 129.)

[280] OECD. Recommendation of the council on artificial intelligence. https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449, 2021. (Cited on page 2.)

[281] J. J. Oliver. Decision graphs – an extension of decision trees. Technical Report 92/173, Monash University, 1992. (Cited on pages 13 and 22.)

[282] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(36):1–13, Dec 2017. (Cited on pages 32, 47, 59, 89, 126 and 129.)

[283] A. S. Panayides, A. Amini, N. D. Filipovic, A. Sharma, S. A. Tsaftaris, A. Young, D. Foran, N. Do, S. Golemati, T. Kurc, et al. Ai in medical imaging informatics: current challenges and future directions. *IEEE journal of biomedical and health informatics*, 24(7):1837–1857, 2020. (Cited on page 2.)

[284] C. H. Papadimitriou. Computational complexity. In *Encyclopedia of computer science*, pages 260–265. 2003. (Cited on page 7.)

[285] A. Parmentier and T. Vidal. Optimal counterfactual explanations in tree ensembles. In *ICML*, pages 8422–8431, 2021. (Cited on page 76.)

[286] F. Pedregosa and et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. (Cited on page 15.)

[287] R. P. Pérez and C. Uzcátegui. Preferences and explanations. *Artif. Intell.*, 149(1):1–30, 2003. (Cited on pages 21 and 38.)

[288] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. (Cited on pages 5 and 12.)

[289] S. Raaijmakers. Artificial intelligence for law enforcement: challenges and opportunities. *IEEE security & privacy*, 17(5):74–77, 2019. (Cited on pages 2, 4 and 18.)

[290] M. N. Rabe and L. Tentrup. CAQE: A certifying QBF solver. In *FMCAD*, pages 136–143, 2015. (Cited on page 86.)

[291] T. Rademacher. Artificial intelligence and law enforcement. *Regulating artificial intelligence*, pages 225–254, 2020. (Cited on page 2.)

[292] A. Rago, O. Cocarascu, C. Bechlivanidis, D. A. Lagnado, and F. Toni. Argumentative explanations for interactive recommendations. *Artif. Intell.*, 296:103506, 2021. (Cited on page 4.)

[293] A. Rago, O. Cocarascu, C. Bechlivanidis, and F. Toni. Argumentation as a framework for interactive explanations for recommendations. In *KR*, pages 805–815, 2020. (Cited on page 4.)

[294] A. Rai. Explainable ai: From black box to glass box. *Journal of the Academy of Marketing Science*, 48:137–141, 2020. (Cited on pages 2 and 3.)

[295] P. Rajpurkar, E. Chen, O. Banerjee, and E. J. Topol. Ai in health and medicine. *Nature medicine*, 28(1):31–38, 2022. (Cited on page 2.)

[296] S. Reddy, S. Allan, S. Coghlan, and P. Cooper. A governance model for the application of ai in health care. *Journal of the American Medical Informatics Association*, 27(3):491–497, 2020. (Cited on page 2.)

[297] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987. (Cited on pages 11, 20 and 22.)

[298] M. T. Ribeiro, S. Singh, and C. Guestrin. Model-agnostic interpretability of machine learning. *CoRR*, abs/1606.05386, 2016. (Cited on page 22.)

[299] 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. (Cited on pages 3, 4, 15, 16, 18, 21, 36, 64 and 97.)

[300] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535, 2018. (Cited on pages 3, 4, 15, 18, 21, 36, 47, 64 and 97.)

[301] R. L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987. (Cited on pages 5 and 14.)

[302] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. (Cited on page 1.)

[303] C. Rudin. Please stop explaining black box models for high stakes decisions. *CoRR*, abs/1811.10154, 2018. (Cited on pages 4 and 18.)

[304] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. (Cited on pages 3, 4, 5, 15, 18, 22, 49, 50 and 60.)

[305] C. Rudin. Why black box machine learning should be avoided for high-stakes decisions, in brief. *Nature Reviews Methods Primers*, 2(1):1–2, 2022. (Cited on pages 4, 18 and 49.)

[306] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022. (Cited on pages 50 and 60.)

[307] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023. (Cited on page 1.)

[308] L. Ryll, M. E. Barton, B. Z. Zhang, R. J. McWaters, E. Schizas, R. Hao, K. Bear, M. Preziuso, E. Seger, R. Wardrop, et al. Transforming paradigms: A global ai in financial services survey. 2020. (Cited on page 2.)

[309] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K. Müller, editors. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019. (Cited on pages 15, 22, 36 and 97.)

[310] D. Saraswat, P. Bhattacharya, A. Verma, V. K. Prasad, S. Tanwar, G. Sharma, P. N. Bokoro, and R. Sharma. Explainable ai for healthcare 5.0: opportunities and challenges. *IEEE Access*, 2022. (Cited on page 2.)

[311] M. Sarvmaili, R. Guidotti, A. Monreale, A. Soares, Z. Sadeghi, F. Giannotti, D. Pedreschi, and S. Matwin. A modularized framework for explaining black box classifiers for text data. In *CCAI*, 2022. (Cited on page 100.)

[312] D. Schiff, J. Biddle, J. Borenstein, and K. Laas. What's next for ai ethics, policy, and governance? a global overview. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 153–158, 2020. (Cited on page 2.)

[313] B. Selman and H. J. Levesque. Abductive and default reasoning: A computational core. In *AAAI*, pages 343–348, 1990. (Cited on page 65.)

[314] L. Semenova, C. Rudin, and R. 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. (Cited on page 5.)

[315] S. A. Seshia, D. Sadigh, and S. S. Sastry. Toward verified artificial intelligence. *Commun. ACM*, 65(7):46–55, 2022. (Cited on pages 1, 3 and 15.)

[316] M. Shanahan. Prediction is deduction but explanation is abduction. In *IJCAI*, pages 1055–1060, 1989. (Cited on pages 21 and 38.)

[317] C. E. Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12):713–723, 1938. (Cited on page 9.)

[318] L. S. Shapley. A value for $n$-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953. (Cited on pages 16 and 97.)

[319] W. Shi, A. Shih, A. Darwiche, and A. Choi. On tractable representations of binary neural networks. In *KR*, pages 882–892, 2020. (Cited on page 4.)

[320] A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018. (Cited on pages 4, 18, 19, 21, 22, 23, 36, 37, 38 and 64.)

[321] A. Shih, A. Choi, and A. Darwiche. Compiling bayesian network classifiers into decision graphs. In *AAAI*, pages 7966–7974, 2019. (Cited on pages 4, 21, 22, 23, 36, 38 and 64.)

[322] A. A. Shrotri, N. Narodytska, A. Ignatiev, K. S. Meel, J. Marques-Silva, and M. Y. Vardi. Constraint-driven explanations for black-box ml models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8304–8314, 2022. (Cited on pages 4 and 38.)

[323] F. Siddiqui and J. B. Merrill. 17 fatalities, 736 crashes: The shocking toll of teslas autopilot, 2023. (Cited on page 2.)

[324] J. Sill. Monotonic networks. In *NIPS*, pages 661–667, 1997. (Cited on page 89.)

[325] A. Silva, M. C. Gombolay, T. W. Killian, I. D. J. Jimenez, and S. Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *AISTATS*, pages 1855–1865, 2020. (Cited on page 22.)

[326] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. (Cited on page 1.)

[327] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. (Cited on page 1.)

[328] A. Sivaraman, G. Farnadi, T. D. Millstein, and G. V. den Broeck. Counterexample-guided learning of monotonic neural networks. In *NeurIPS*, 2020. (Cited on pages 15 and 89.)

[329] D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. In *NeurIPS*, pages 9391–9404, 2021. (Cited on pages 16 and 100.)

[330] 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. (Cited on pages 4, 18, 21 and 101.)

[331] A. Sorayaie Azar, S. Babaei Rikan, A. Naemi, J. Bagherzadeh Mohasefi, H. Pirnejad, M. Bagherzadeh Mohasefi, and U. K. Wiil. Application of machine learning techniques for predicting survival in ovarian cancer. *BMC Medical Informatics and Decision Making*, 22(1):345, 2022. (Cited on pages 100 and 133.)

[332] E. Strumbelj and I. Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, 2010. (Cited on pages 3, 16, 17, 97 and 100.)

[333] E. Strumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.*, 41(3):647–665, 2014. (Cited on pages 3, 16, 17, 97 and 100.)

[334] M. Sundararajan and A. Najmi. The many Shapley values for model explanation. In *ICML*, pages 9269–9278, 2020. (Cited on pages 101 and 132.)

[335] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. (Cited on page 135.)

[336] H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson. Nnv: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, pages 3–17. Springer, 2020. (Cited on page 135.)

[337] UNESCO. Draft recommendation on the ethics of artificial intelligence. https://unesdoc.unesco.org/ark:/48223/pf0000374266, June 2021. (Cited on page 2.)

[338] B. Ustun, A. Spangher, and Y. Liu. Actionable recourse in linear classification. In *FAT*, pages 10–19, 2019. (Cited on page 20.)

[339] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Mach. Learn.*, 29(1):5–44, 1997. (Cited on page 13.)

[340] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. (Cited on page 40.)

[341] G. Van den Broeck and A. Darwiche. On the role of canonicity in knowledge compilation. In *AAAI*, pages 1641–1648, 2015. (Cited on pages 4, 10 and 47.)

[342] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. (Cited on page 32.)

[343] S. Venkatasubramanian and M. Alfano. The philosophical basis of algorithmic recourse. In *FAT*, pages 284–293, 2020. (Cited on page 20.)

[344] K. N. Vokinger and U. Gasser. Regulating ai in medicine in the united states and europe. *Nature machine intelligence*, 3(9):738–739, 2021. (Cited on page 2.)

[345] W. J. von Eschenbach. Transparency and the black box problem: Why we do not trust ai. *Philosophy & Technology*, 34(4):1607–1622, 2021. (Cited on pages 2, 4 and 18.)

[346] V. Voukelatou, I. Miliou, F. Giannotti, and L. Pappalardo. Understanding peace through the world news. *EPJ Data Sci.*, 11(1):2, 2022. (Cited on page 100.)

[347] 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. (Cited on pages 38, 50, 54, 58 and 64.)

[348] Y. Wang, J. Lang, J. Z. Zuo, Y. Dong, Z. Hu, X. Xu, Y. Zhang, Q. Wang, L. Yang, S. T. Wong, et al. The radiomic-clinical model using the SHAP method for assessing the treatment response of whole-brain radiotherapy: a multicentric study. *European Radiology*, pages 1–11, 2022. (Cited on pages 100 and 133.)

[349] D. S. Watson. Rational Shapley values. In *FAccT*, pages 1083–1094, 2022. (Cited on pages 16 and 100.)

[350] D. S. Watson, L. Gultchin, A. Taly, and L. Floridi. Local explanations via necessity and sufficiency: unifying theory and practice. In *UAI*, volume 161, pages 1382–1392, 2021. (Cited on page 101.)

[351] I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000. (Cited on pages 7, 9 and 47.)

[352] A. Wehenkel and G. Louppe. Unconstrained monotonic neural networks. In *NeurIPS*, pages 1543–1553, 2019. (Cited on page 89.)

[353] D. S. Weld and G. Bansal. The challenge of crafting intelligible intelligence. *Commun. ACM*, 62(6):70–79, 2019. (Cited on pages 21, 36 and 37.)

[354] S. Whalen, J. Schreiber, W. S. Noble, and K. S. Pollard. Navigating the pitfalls of applying machine learning in genomics. *Nature Reviews Genetics*, 23(3):169–181, 2022. (Cited on page 1.)

[355] J. M. Wing. Trustworthy ai. *Communications of the ACM*, 64(10):64–71, 2021. (Cited on page 1.)

[356] E. Withnell, X. Zhang, K. Sun, and Y. Guo. Xomivae: an interpretable deep learning model for cancer classification using high-dimensional omics data. *Briefings in Bioinformatics*, 22(6):bbab315, 2021. (Cited on pages 100 and 133.)

[357] L. Wolf, T. Galanti, and T. Hazan. A formal approach to explainability. In *AIES*, pages 255–261, 2019. (Cited on pages 4 and 64.)

[358] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu. Explainable AI: A brief survey on history, research areas, approaches and challenges. In *NLPCC*, pages 563–574, 2019. (Cited on pages 3, 22 and 36.)

[359] K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020. (Cited on page 135.)

[360] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C.-J. Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *arXiv preprint arXiv:2011.13824*, 2020. (Cited on page 135.)

[361] T. Yan and A. D. Procaccia. If you like Shapley then you'll love the core. In *AAAI*, pages 5751–5759, 2021. (Cited on page 101.)

[362] G. Yang, Q. Ye, and J. 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. (Cited on page 2.)

[363] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *ICML*, volume 70, pages 3921–3930, 2017. (Cited on pages 3 and 15.)

[364] L. Yang, X. Wu, Y. Jiang, and Z. Zhou. Multi-label learning with deep forest. In *ECAI*, volume 325, pages 1634–1641, 2020. (Cited on page 14.)

[365] S. You, D. Ding, K. R. Canini, J. Pfeifer, and M. R. Gupta. Deep lattice networks and partial monotonic functions. In *NeurIPS*, pages 2981–2989, 2017. (Cited on pages 15 and 89.)

[366] K. Young, G. Booth, B. Simpson, R. Dutton, and S. Shrapnel. Deep neural network or dermatologist? *CoRR*, abs/1908.06612, 2019. (Cited on page 101.)

[367] D. Yu, Z. Liu, C. Su, Y. Han, X. Duan, R. Zhang, X. Liu, Y. Yang, and S. Xu. Copy number variation in plasma as a tool for lung cancer prediction using extreme gradient boosting (xgboost) classifier. *Thoracic cancer*, 11(1):95–102, 2020. (Cited on pages 100 and 133.)

[368] J. Yu, G. Farr, A. Ignatiev, and P. J. Stuckey. Anytime approximate formal feature attribution. *arXiv preprint arXiv:2312.06973*, 2023. (Cited on pages 4, 18 and 135.)

[369] J. Yu, A. Ignatiev, and P. J. Stuckey. On formal feature attribution and its approximation. *arXiv preprint arXiv:2307.03380*, 2023. (Cited on pages 4, 18 and 135.)

[370] J. Yu, A. Ignatiev, P. J. Stuckey, and P. L. Bodic. Computing optimal decision sets with SAT. In H. Simonis, editor, *CP*, pages 952–970, 2020. (Cited on page 60.)

[371] J. Yu, A. Ignatiev, P. J. Stuckey, and P. L. Bodic. Learning optimal decision sets and lists with SAT. *J. Artif. Intell. Res.*, 72:1251–1279, 2021. (Cited on page 60.)

[372] J. Yu, A. Ignatiev, P. J. Stuckey, N. Narodytska, and J. Marques-Silva. Eliminating the impossible, whatever remains must be true. *CoRR*, abs/2206.09551, 2022. (Cited on page 64.)

[373] J. Yu, A. Ignatiev, P. J. Stuckey, N. Narodytska, and J. 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. (Cited on pages 4 and 57.)

[374] R. Yu and G. S. Alì. What's inside the black box? ai challenges for lawyers and researchers. *Legal Information Management*, 19(1):2–13, 2019. (Cited on page 2.)

[375] R. Zarinshenas, C. Ladbury, H. McGee, D. Raz, L. Erhunmwunsee, R. Pathak, S. Glaser, R. Salgia, T. Williams, and A. Amini. Machine learning to refine prognostic and predictive nodal burden thresholds for post-operative radiotherapy in completely resected stage iii-n2 non-small cell lung cancer. *Radiotherapy and Oncology*, 173:10–18, 2022. (Cited on pages 100 and 133.)

[376] G. Zhang, Y. Shi, P. Yin, F. Liu, Y. Fang, X. Li, Q. Zhang, and Z. Zhang. A machine learning model based on ultrasound image features to assess the risk of sentinel lymph node metastasis in breast cancer patients: Applications of scikit-learn and SHAP. *Frontiers in Oncology*, 12, 2022. (Cited on pages 100 and 133.)

[377] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018. (Cited on page 135.)

[378] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. (Cited on page 1.)

[379] X. Zhang, L. Wang, J. Helwig, Y. Luo, C. Fu, Y. Xie, M. Liu, Y. Lin, Z. Xu, K. Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. *arXiv preprint arXiv:2307.08423*, 2023. (Cited on page 1.)

[380] Y. Zhang, Y. Weng, and J. Lund. Applications of explainable artificial intelligence in diagnosis and surgery. *Diagnostics*, 12(2):237, 2022. (Cited on pages 100 and 133.)

[381] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012. (Cited on pages 12 and 14.)

[382] Z.-H. Zhou. *Machine learning*. Springer Nature, 2021. (Cited on pages 12, 123, 124 and 126.)