On Computing Relevant Features for Explaining NBCs*

Yacine Izza[1], Joao Marques-Silva[2]

[1]CREATE, National University of Singapore, 1 CREATE Way, 138602, Singapore
[2]IRIT, CNRS, 118 Route de Narbonne, 31062 Toulouse, France

Abstract

Despite the progress observed with model-agnostic explainable AI (XAI), it is the case that model-agnostic XAI can produce incorrect explanations. One alternative are the so-called formal approaches to XAI, that include *abductive* explanations. Unfortunately, abductive explanations also exhibit important drawbacks, the most visible of which is arguably their size. The computation of relevant features serves to trade off probabilistic precision for the number of features in an explanation. However, even for very simple classifiers, the complexity of computing sets of relevant features is prohibitive. This paper investigates the computation of relevant sets for Naive Bayes Classifiers (NBCs), and shows that, in practice, these are easy to compute. Furthermore, the experiments confirm that succinct sets of relevant features can be obtained with NBCs.

Keywords

Naive Bayes, Explainability, Dynamic Programming

1. Introduction

The advances in Machine Learning (ML) in recent years motivate an ever increasing range of practical applications of Artificial Intelligence (AI) systems. In some domains, the use of AI systems is premised on the availability of mechanisms for explaining the often opaque operation of ML models. Some uses of ML models are deemed *high-risk* given the impact that their operation can have on people [2]. (Other authors refer to *high-stakes* applications [3].) For high-risk AI systems, a critical requirement is rigor, either when reasoning about these systems, or when explaining their predictions.

Recent years have witnessed a growing interest in eXplainable AI (XAI) [4, 5, 6, 7, 8, 9]. The best-known XAI approaches can be broadly categorized as model-agnostic methods, that include for example LIME [10], SHAP [11] and Anchor [12], and intrinsic interpretability [3, 8], for which the explanation is represented by the actual (interpretable) ML model. Intrinsic interpretability may not represent a viable option in some uses of AI systems. On the other hand, model-agnostic methods, although locally accurate, can produce explanations that are unsound [13], in addition to displaying several other drawbacks [14, 15, 16, 17]. Unsound explanations are hopeless whenever rigor is a key requirement; thus, model-agnostic explanations ought not be used in high-risk settings. Indeed, it has been reported [13] that an explanation X can be consistent with different predicted classes. For example, for a bank loan application, X might be consistent with an approved loan application, but also with a declined loan application. An explanation that is consistent with both a declined and an approved loan applications offers no insight to why one of the loan applications was declined. There have been recent efforts on rigorous XAI approaches [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40], most of which are based on feature attribution, namely the computation of so-called abductive explanations (AXp's). However, these efforts have mostly focused on the scalability of computing rigorous explanations, with more recent work investigating input distributions [34]. Nevertheless, another important limitation of rigorous XAI approaches is the often unwieldy size of explanations. Recent work studied probabilistic explanations, as a mechanism to reduce the size of rigorous explanations [41, 42]. Probabilistic explanations have extended model-agnostic approaches [41], and so can suffer from unsoundness. Alternatively, more rigorous approaches to computing probabilistic explanations have been shown to be computationally hard, concretely hard for NP^{PP}, and so most likely beyond the reach of modern automated reasoners.

This paper builds on recent work [42] on rigorous probabilistic explanations, and investigates their practical scalability. However, instead of considering classifiers represented as boolean circuits (as in [42]), the paper specifically considers the family of naive Bayes classifiers (NBCs). Earlier work showed that rigorous explanations of NBCs, concretely AXp's, are computed in polynomial time, and that their enumeration is achieved with polynomial delay [43]. Unfortunately, the size of explanations was not investigated in this earlier work. This paper studies probabilistic explanations for the concrete case of NBCs. For the case of categorical features, the paper re-

ENIGMA-23, September 03–04, 2023, Rhodes, Greece
* A longer version of this paper has been published at Int. J. Approx. Reason. Vol 159 (2023) [1].
izza@comp.nus.edu.sg (Y. Izza); joao.marques-silva@irit.fr (J. Marques-Silva)
0000-0002-7774-1945 (Y. Izza); 0000-0002-6632-3086 (J. Marques-Silva)

lates probabilistic explanations of NBCs with the problem of counting the models of (restricted forms) of integer programming constraints, and proposes a dynamic programming based, pseudo-polynomial algorithm for computing approximate (or locally-minimal) explanations. Such approximate explanations offer important theoretical guarantees: i) approximate explanations are not larger than some rigorous explanation; ii) approximate explanations are not smaller than some rigorous probabilistic explanation; and iii) approximate explanations offer strong probabilistic guarantees on their precision. More importantly, the experimental results demonstrate that succinct explanations, with sizes that can be deemed within the grasp of human decision makers [44], can be very efficiently computed with most often a small decrease in the precision of the explanation.

The paper is organized as follows. Section 2 introduces the definitions and notation used throughout the paper. Section 3 summarizes the computation of explanations for NBCs proposed in earlier work [43]. Section 4 details the approach proposed in this paper for computing locally-minimal probabilistic AXp's. Section 5 presents experimental results confirming that precise short locally-minimal AXp's can be efficiently computed. Section 6 concludes the paper.

2. Preliminaries

2.1. Classification problems

This paper considers classification problems, which are defined on a set of features (or attributes) $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_1, c_2, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ takes values from a domain $\mathbb{D}_i$. In general, domains can be categorical or ordinal, with values that can be boolean, integer or real-valued but in this paper we restrict $\mathcal{K} = \{0, 1\}$, i.e. binary classifiers, and all features are categorical. (Throughout the paper, we also use the notations $\ominus$ and $\oplus$ to denote, resp. class 0 and class 1.) Feature space is defined as $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_m$; $|\mathbb{F}|$ represents the total number of points in $\mathbb{F}$ if none of the features is real-valued. For boolean domains, $\mathbb{D}_i = \{0, 1\} = \mathbb{B}$, $i = 1, \ldots, m$, and $\mathbb{F} = \mathbb{B}^m$. 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 set of variables associated with features is $X = \{x_1, \ldots, x_m\}$. Moreover, 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 $\mathbb{D}_i$. An ML classifier $\mathbb{M}$ is characterized by a (non-constant) *classification function* κ that maps feature space $\mathbb{F}$ into the set of classes $\mathcal{K}$, i.e. $\kappa : \mathbb{F} \to \mathcal{K}$. An *instance* (or observation) denotes a pair $(\mathbf{v}, c)$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$, with $c = \kappa(\mathbf{v})$. (We also use the term *instance* to refer to $\mathbf{v}$, leaving c implicit.)

2.2. Formal explanations

We now define formal explanations. In contrast with the well-known model-agnostic approaches to XAI [10, 11, 12, 5], formal explanations are model-precise, i.e. their definition reflects the model's computed function. Prime implicant (PI) explanations [18] 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 (AXp) [19]. Formally, given $\mathbf{v} = (v_1, \ldots, v_m) \in \mathbb{F}$ with $\kappa(\mathbf{v}) = c$, an AXp is any minimal subset $\mathcal{X} \subseteq \mathcal{F}$ such that,

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

i.e. the features in $\mathcal{X}$ are sufficient for the prediction when these take the values dictated by $\mathbf{v}$, and $\mathcal{X}$ is irreducible. Also, a non-minimal set such that (1) holds is a WeakAXp. AXp's can be viewed as answering a 'Why?' question, i.e. why is some prediction made given some point in feature space. Contrastive explanations [45] offer a different view of explanations, but these are beyond the scope of the paper.

2.3. δ-relevant sets

δ-relevant sets were proposed in more recent work [42] as a generalized formalization of explanations. δ-relevant sets can be viewed as *probabilistic* PIs, with AXp's representing a special case of δ-relevant sets where $\delta = 1$, i.e. probabilistic PIs that are actual PIs. We briefly overview the definitions related with relevant sets. The assumptions regarding the probabilities of logical propositions are those made in earlier work [42]. Let $\Pr_{\mathbf{x}}(A(\mathbf{x}))$ denote the probability of some proposition A defined on the vector of variables $\mathbf{x} = (x_1, \ldots, x_m)$, i.e.

$$\begin{aligned} \Pr_{\mathbf{x}}(A(\mathbf{x})) &= \frac{|\{\mathbf{x} \in \mathbb{F} : A(\mathbf{x}) = 1\}|}{|\{\mathbf{x} \in \mathbb{F}\}|} \\ \Pr_{\mathbf{x}}(A(\mathbf{x}) \,|\, B(\mathbf{x})) &= \frac{|\{\mathbf{x} \in \mathbb{F} : A(\mathbf{x}) = 1 \land B(\mathbf{x}) = 1\}|}{|\{\mathbf{x} \in \mathbb{F} : B(\mathbf{x}) = 1\}|} \end{aligned} \quad (2)$$

(Similar to earlier work, it is assumed that the features are independent and uniformly distributed [42]. Moreover, the definitions above can be adapted in case some of the features are real-valued. As noted earlier, the paper studies only categorical features.)

Definition 2.1 (δ-relevant set [42])**.** Consider $\kappa : \mathbb{B}^m \to \mathcal{K} = \mathbb{B}$, $\mathbf{v} \in \mathbb{B}^m$, $\kappa(\mathbf{v}) = c \in \mathbb{B}$, and $\delta \in [0, 1]$. $\mathcal{S} \subseteq \mathcal{F}$ is a δ-relevant set for κ and $\mathbf{v}$ if,

$$\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}) \geq \delta \quad (3)$$

(where the restriction of $\mathbf{x}$ to the variables with indices in $\mathcal{S}$ is represented by $\mathbf{x}_{\mathcal{S}} = (x_i)_{i \in \mathcal{S}}$).

(Observe that $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})$ is often referred to as the *precision* of $\mathcal{S}$ [12, 21].) Thus, a δ-relevant set represents a set of features which, if fixed to some predefined value (taken from a reference vector $\mathbf{v}$), ensures that the probability of the prediction being the same as the one for $\mathbf{v}$ is no less than δ.

Definition 2.2 (Min-δ-relevant set)**.** Given κ, $\mathbf{v} \in \mathbb{B}^m$, and $\delta \in [0, 1]$, find the smallest k, such that there exists $\mathcal{S} \subseteq \mathcal{F}$, with $|\mathcal{S}| = k$, and $\mathcal{S}$ is a δ-relevant set for κ and $\mathbf{v}$.

With the goal of proving the computational complexity of finding a minimum-size set of features that is a δ-relevant set, earlier work [42] restricted the definition to the case where κ is represented as a boolean circuit.

(Boolean circuits were restricted to propositional formulas defined using the operators $\vee$, $\wedge$ and $\neg$, and using a set of variables representing the inputs; this explains the choice of *inputs* over *sets* in earlier work [42].)

2.4. Naive Bayes Classifiers (NBCs)

NBC [46] is a Bayesian Network model [47] characterized by strong conditional independence assumptions among the features. Given some observation $\mathbf{x} \in \mathbb{F}$, the predicted class is given by:

$$\kappa(\mathbf{x}) = \mathsf{argmax}_{c \in \mathcal{K}} \left(\Pr(c|\mathbf{x})\right) \tag{4}$$

Using the Bayes theorem, $\Pr(c|\mathbf{x})$ can be computed as follows: $\Pr(c|\mathbf{x}) = {}^{\Pr(c,\mathbf{x})}/_{\Pr(\mathbf{x})}$. In practice, we compute only the numerator of the fraction, since the denominator $\Pr(\mathbf{x})$ is constant for every $c \in \mathcal{K}$. Moreover, given the conditional mutual independency of the features, we have:

$$\Pr(c, \mathbf{x}) = \Pr(c) \times \prod\nolimits_i \Pr(x_i|c)$$

Furthermore, it is also common in practice to apply logarithmic transformations on probabilities of $\Pr(c, \mathbf{x})$, thus getting:

$$\log \Pr(c, \mathbf{x}) = \log \Pr(c) + \sum\nolimits_i \log \Pr(x_i|c)$$

Therefore, (4) can be rewritten as follows:

$$\kappa(\mathbf{x}) = \mathsf{argmax}_{c \in \mathcal{K}} \left(\log \Pr(c) + \sum\nolimits_i \log \Pr(x_i|c)\right) \tag{5}$$

For simplicity, and following the notations used in [43], we use lPr to denote the logarithmic probabilities, thus getting:

$$\kappa(\mathbf{x}) = \mathsf{argmax}_{c \in \mathcal{K}} \left(\mathrm{lPr}(c) + \sum\nolimits_i \mathrm{lPr}(x_i|c)\right) \tag{6}$$

(Note that also for simplicity, it is common in practice to add a sufficiently large positive threshold T to each probability and then use only positive values.)

Running Example. Consider the NBC depicted graphically in Figure 1 [1]. The features are the discrete random variables R_1, R_2, R_3, R_4 and R_5. Each R_i can take values $\mathbf{t}$ or $\mathbf{f}$ denoting, respectively, whether a listener likes or not that radio station. Random variable G denotes an age class, which can take values Y and O, denoting young and older listeners, respectively. The target class $\oplus$ denotes the prediction *yes* (i.e. the listener likes the radio station) and $\ominus$ denotes the prediction *no* (i.e. the listerner does not like the radio station). Thus, $\mathcal{K} = \{\ominus, \oplus\}$. Let us consider $\mathbf{v} = (R_1, R_2, R_3, R_4, R_5) = (\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$. We associate r_i to each literal ($R_i = \mathbf{t}$) and $\neg r_i$ to literals ($R_i = \mathbf{f}$). Using (6), we get the values shown in Figure 2. (Note that to use positive values, we added $T = +4$ to each $\mathrm{lPr}(\cdot)$.) As can be concluded, the classifier will predict $\oplus$.

3. Explaining NBCs in Polynomial Time

This section overviews the approach proposed in [43] for computing AXp's for binary NBCs. The general idea is to reduce the NBC problem into an Extended Linear Classifier (XLC) and then explain the resulting XLC. Our purpose is to devise a new approach that builds on XLC formulation to compute δ-relevant sets for NBCs. Hence, it is useful to recall first the translation of NBCs into XLCs and AXp's extraction from XLCs.

3.1. Extended Linear Classifiers

We consider an XLC with categorical features. (Recall that the paper considers NBCs with binary classes and categorical data.) Each feature $i \in \mathcal{F}$ has $x_i \in \{1, \ldots, d_i\}$, (i.e. $\mathbb{D}_i = \{1, \ldots, d_i\}$). Let,

$$\nu(\mathbf{x}) \triangleq w_0 + \sum\nolimits_{i \in \mathcal{F}} \sigma(x_i, v_i^1, v_i^2, \ldots, v_i^{d_i}) \tag{7}$$

σ is a selector function that picks the value v_i^r iff x_i takes value r. Moreover, let us define the decision function, $\kappa(\mathbf{x}) = \oplus$ if $\nu(\mathbf{x}) > 0$ and $\kappa(\mathbf{x}) = \ominus$ if $\nu(\mathbf{x}) \leq 0$.

The reduction of a binary NBC, with categorical features, to an XLC is completed by setting: $w_0 \triangleq \mathrm{lPr}(\oplus) - \mathrm{lPr}(\ominus)$, $v_i^1 \triangleq \mathrm{lPr}(x_i = 1|\oplus) - \mathrm{lPr}(x_i = 1|\ominus)$, $v_i^2 \triangleq \mathrm{lPr}(x_i = 2|\oplus) - \mathrm{lPr}(x_i = 2|\ominus)$, $\ldots$, $v_i^{d_i} \triangleq \mathrm{lPr}(x_i = d_i|\oplus) - \mathrm{lPr}(x_i = d_i|\ominus)$. Hence, the argmax in (6) is replaced with inequality to get the following:

$$\mathrm{lPr}(\oplus) - \mathrm{lPr}(\ominus) + \sum\nolimits_{i=1}^{m} \sum\nolimits_{k=1}^{k=d_j} (\mathrm{lPr}(x_i = k|\oplus) - \mathrm{lPr}(x_i = k|\ominus))(x_i = k) > 0 \tag{8}$$

[1] This example of an NBC is adapted from [43], which is initially reported in [48, Ch.10].

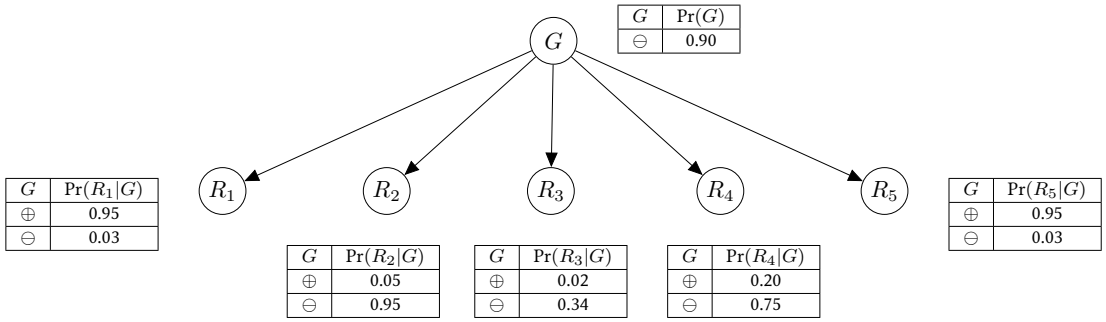

Figure 1: Running example.

	Pr(⊕)	Pr(r_1|⊕)	Pr($\neg r_2$|⊕)	Pr($\neg r_3$|⊕)	Pr($\neg r_4$|⊕)	Pr(r_5|⊕)	lPr(⊕|**v**)
Pr(·)	0.10	0.95	0.95	0.98	0.80	0.95	
lPr(·)	1.70	3.95	3.95	3.98	3.78	3.95	21.31

(a) Computing lPr(⊕|**v**)

	Pr(⊖)	Pr(r_1|⊖)	Pr($\neg r_2$|⊖)	Pr($\neg r_3$|⊖)	Pr($\neg r_4$|⊖)	Pr(r_5|⊖)	lPr(⊖|**v**)
Pr(·)	0.90	0.03	0.05	0.66	0.25	0.03	
lPr(·)	3.89	0.49	1.00	3.58	2.61	0.49	12.06

(b) Computing lPr(⊖|**v**)

Figure 2: Deciding prediction for $\mathbf{v} = (\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

Example 1. Figure 3a shows the resulting XLC formulation for the example in Figure 2. We also let $\mathbf{f}$ be associated with value 1 and $\mathbf{t}$ be associated with value 2, and $d_i = 2$.

3.2. Explaining XLCs

We now describe how AXp's can be computed for XLCs. For a given instance $\mathbf{x} = \mathbf{a}$, define a *constant* slack (or gap) value Γ given by,

$$\Gamma \triangleq \nu(\mathbf{a}) = \sum\nolimits_{i \in \mathcal{F}} \sigma(a_i, v_i^1, v_i^2, \ldots, v_i^{d_i}) \tag{9}$$

Computing an AXp corresponds to finding a subset-minimal set of literals $\mathcal{S} \subseteq \mathcal{F}$ such that (1) holds, or alternatively,

$$\forall(\mathbf{x} \in \mathbb{F}).\, \bigwedge\nolimits_{i \in \mathcal{S}} (x_i = a_i) \rightarrow (\nu(\mathbf{x}) > 0) \tag{10}$$

under the assumption that $\nu(\mathbf{a}) > 0$. Thus, the purpose is to find the *smallest* slack that can be achieved by allowing the feature not in $\mathcal{S}$ to take any value (i.e. *universal/free* features), given that the literals in $\mathcal{S}$ are fixed by $\mathbf{a}$ (i.e. $\bigwedge_{i \in \mathcal{S}} (x_i = a_i)$).

Let v_i^{ω} denote the *smallest* (or *worst-case*) value associated with x_i. Then, by letting every x_i take *any* value, the *worst-case* value of $\nu(\mathbf{e})$ is,

$$\Gamma^{\omega} = w_0 + \sum\nolimits_{i \in \mathcal{F}} v_i^{\omega} \tag{11}$$

Moreover, from (9), we have: $\Gamma = w_0 + \sum_{i \in \mathcal{F}} v_i^{a_i}$. The expression above can be rewritten as follows,

$$\begin{array}{rcl} \Gamma^{\omega} & = & w_0 + \sum_{i \in \mathcal{F}} v_i^{a_i} - \sum_{i \in \mathcal{F}} (v_i^{a_i} - v_i^{\omega}) \\ & = & \Gamma - \sum_{i \in \mathcal{F}} \delta_i = -\Phi \end{array} \tag{12}$$

w_0	v_1^1	v_1^2	v_2^1	v_2^2	v_3^1	v_3^2	v_4^1	v_4^2	v_5^1	v_5^2
-2.19	-2.97	3.46	2.95	-2.95	0.4	-2.83	1.17	-1.32	-2.97	3.46

(a) Example reduction of NBC to XLC (Example 1)

Γ	δ_1	δ_5	δ_2	δ_3	δ_4	Φ
9.25	6.43	6.43	5.90	3.23	2.49	15.23

(b) Computing δ_j's for the XLC (Example 2)

Figure 3: Values used in the running example (Example 1 and Example 2)

where $\delta_i \triangleq v_i^{a_i} - v_i^{\omega}$, and $\Phi \triangleq \sum_{i\in\mathcal{F}} \delta_i - \Gamma = -\Gamma^{\omega}$. Recall the goal is to find a subset-minimal set $\mathcal{S}$ such that the prediction is still c (whatever the values of the other features):

$$w_0 + \sum\nolimits_{i\in\mathcal{S}} v_i^{a_i} + \sum\nolimits_{i\notin\mathcal{S}} v_i^{\omega} = -\Phi + \sum\nolimits_{i\in\mathcal{S}} \delta_i > 0 \tag{13}$$

In turn, (13) can be represented as the following knapsack problem [49]:

$$\begin{array}{ll} \min & \sum_{i=1}^{m} p_i \\ \text{such that} & \sum_{i=1}^{m} \delta_i p_i > \Phi \\ & p_i \in \{0,1\} \end{array} \tag{14}$$

where the variables p_i assigned value 1 denote the indices included in $\mathcal{S}$. Note that, the fact that the coefficients in the cost function are all equal to 1 makes the problem solvable in log-linear time.

Example 2. Figure 3b shows the values used for computing explanations for the example in Figure 2. For this example, the sorted δ_j's become $\langle \delta_1, \delta_5, \delta_2, \delta_4, \delta_3 \rangle$. By picking δ_1, δ_2 and δ_5, we ensure that the prediction is $\oplus$, independently of the values assigned to features 3 and 4. Thus $\{1, 2, 5\}$ is an AXp for the NBC shown in Figure 1, with the input instance $(v_1, v_2, v_3, v_4, v_5) = (\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$. (It is easy to observe that $\kappa((\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})) = \kappa((\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})) = \kappa((\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})) = \kappa((\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})) = \kappa((\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})) = \kappa((\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})) = \oplus$.)

4. δ-Relevant Sets for NBCs

This section investigates the computation of δ-relevant sets in the concrete case of NBCs.

Observe that Definition 2.2 imposes no restriction on the representation of the classifier that is assumed in earlier work [42], i.e. the logical representation of κ need not be a boolean circuit. As a result, we extend the definitions from earlier work [42], as detailed below.

4.1. Weak, Locally-Minimal & Smallest Probabilistic AXp's

A *weak probabilistic* AXp (WeakPAXp) is a set of fixed features for which the conditional probability of predicting the correct class c exceeds δ, given $c = \kappa(\mathbf{v})$. Thus, $\mathcal{S} \subseteq \mathcal{F}$ is a WeakPAXp if,

$$\begin{aligned} &\mathsf{WeakPAXp}(\mathcal{S}; \mathbb{F}, \kappa, \mathbf{v}, \delta) \\ &\quad := \Pr\nolimits_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}) \geq \delta \\ &\quad := \frac{|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \land (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})\}|}{|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})\}|} \geq \delta \end{aligned} \tag{15}$$

which means that the fraction of the number of models predicting the target class and consistent with the fixed features (represented by $\mathcal{S}$), given the total number of points in feature space consistent with the fixed features, must exceed δ. (The main difference to (3) is that features and classes are no longer required to be boolean. Also, the definition makes explicit the parameterizations assumed.) Moreover, a *probabilistic* AXp (PAXp) $\mathcal{X}$ is a WeakPAXp that is also subset-minimal,

$$\begin{aligned} &\mathsf{PAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, \delta) := \\ &\quad \mathsf{WeakPAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, \delta) \land \\ &\quad \forall(\mathcal{X}' \subsetneq \mathcal{X}).\neg\mathsf{WeakPAXp}(\mathcal{X}'; \mathbb{F}, \kappa, \mathbf{v}, \delta) \end{aligned} \tag{16}$$

Minimum-size PAXp's (MinPAXp, or smallest PAXp) generalize Min-δ-relevant sets in Definition 2.2. Furthermore, we define an *locally-minimal* probabilistic AXp (LmPAXp) $\mathcal{X}$ as a WeakPAXp such that the removal of any single feature i from $\mathcal{X}$ will falsify $\mathsf{WeakPAXp}(\mathcal{X} \setminus \{i\}; \mathbb{F}, \kappa, \mathbf{v}, \delta)$. Formally:

$$\begin{aligned} &\mathsf{LmPAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, \delta) := \\ &\quad \mathsf{WeakPAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, \delta) \land \\ &\quad \forall(\mathcal{X} \setminus \{i\}).\neg\mathsf{WeakPAXp}(\mathcal{X} \setminus \{i\}; \mathbb{F}, \kappa, \mathbf{v}, \delta) \end{aligned} \tag{17}$$

As stated earlier, the main purpose of this paper is to investigate the computation of LmPAXp explanations. The next section introduces a pseudo-polynomial time algorithm for computing LmPAXp's. Although, LmPAXp are not minimal subset/cardinality, our experiments show that the proposed approach computes (in pseudo-polynomial time) succinct [44] and highly precise approximate explanations.

4.2. Counting Models of XLCs

Earlier work [50, 51, 52, 53] proposed the use of dynamic programming (DP) for approximating the number of feasible solutions of the 0-1 knapsack constraint, i.e. the #knapsack problem. Here we propose an extension of the basic formulation, to allow counting feasible solutions of XLCs.

We are interested in the number of solutions of,

$$\sum\nolimits_{j\in\mathcal{F}} \sigma(x_j, v_j^1, v_j^2, \ldots, v_j^{d_j}) > -w_0 \tag{18}$$

where we assume all v_j^i to be integer-valued, and non-negative (e.g. this is what the translation from NBCs to XLCs yields). Moreover, (18) can be written as follows:

$$\sum\nolimits_{j\in\mathcal{F}} \sigma(x_j, -v_j^1, -v_j^2, \ldots, -v_j^{d_j}) < w_0 \tag{19}$$

which reveals the relationship with the Knapsack constraint.

For each j, let us sort the $-v_j^i$ in non-decreasing order, collapsing duplicates, and counting the number of duplicates, obtaining two sequences:

$$\langle w_j^1, \ldots, w_j^{d_j'} \rangle$$
$$\langle n_j^1, \ldots, n_j^{d_j'} \rangle$$

such that $w_j^1 < w_j^2 < \ldots < w_j^{d_j'}$ and each $n_j^i \geq 1$ gives the number of repetitions of weight w_j^i.

Counting. Let $C(k, r)$ denote the number of solutions of (19) when the subset of features considered is $\{1, \ldots, k\}$ and the sum of picked weights is at most r. To define the solution for the first k features, taking into account the solution for the first $k-1$ features, we must consider that the solution for r can be obtained due to *any* of the possible values of x_j. As a result, for an XLC the general recursive definition of $C(k, r)$ becomes,

$$C(k, r) = \sum_{i=1}^{d_k'} n_k^i \times C(k-1, r - w_k^i)$$

Moreover, $C(1, r)$ is given by,

$$C(1, r) = \begin{cases} 0 & \text{if } r < w_1^1 \\ n_1^1 & \text{if } w_1^1 \leq r < w_1^2 \\ n_1^1 + n_1^2 & \text{if } w_1^2 \leq r < w_1^3 \\ \ldots & \\ \sum_{i=1}^{d_1'} n_1^i & \text{if } w_1^{d_1'} \leq r \end{cases}$$

In addition, if $r < 0$, then $C(k, r) = 0$, for $k = 1, \ldots, m$. Finally, the dimensions of the $C(k, r)$ table are as follows:

1. The number of rows is m.
2. The (worst-case) number of columns is given by:

$$W' = \sum_{j \in \mathcal{F}} n_j^{d_j'} \times w_j^{d_j'} \tag{20}$$

W' represents the largest possible value, in theory. However, in practice, it suffices to set the number of columns to $W = w_0 + T$, which is often much smaller than W'.

Example 3. Consider the following problem. There are 4 features, $\mathcal{F} = \{1, 2, 3, 4\}$. Each feature j takes values in $\{1, 2, 3\}$, i.e. $x_j \in \{1, 2, 3\}$. The prediction should be 1 when the sum of the values of the x_j variables is no less than 8. We set $w_0 = -7$, and get the formulation,

$$\sum_{j \in \{1,2,3,4\}} \sigma(x_j, 1, 2, 3) > 7$$

where each x_j picks value in $\{1, 2, 3\}$. We translate to the extended knapsack formulation and obtain:

$$\sum_{j \in \{1,2,3,4\}} \sigma(x_j, -1, -2, -3) < -7$$

Table 1
DP table for Example 3

k \ r	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	2	3	3	3	3	3	3	–	–	–	–
2	0	0	1	3	6	8	9	9	9	–	–	–	–
3	0	0	0	1	4	10	17	23	16	–	–	–	–
4	0	0	0	0	1	5	15	31	50	–	–	–	–

We require the weights to be integer and non-negative, and so we sum to each w_j^k the complement of the most negative w_j^k plus 1. Therefore, we add +4 to each j and +16 to right-hand side of the inequality. Thus, we get

$$\sum_{j \in \{1,2,3,4\}} \sigma(x_j, 3, 2, 1) < 9$$

For this formulation, $x_j = 1$ picks value 3. (For example, we can pick two x_j with value 1, but not 3, as expected.)

In this case, the DP table size will be 4×12, even though we are interested in entry $C(4, 8)$. Table 1 shows DP table, and the number of solutions for the starting problem, i.e. there are 50 combinations of values whose sum is no less than 8.

By default, the dynamic programming formulation assumes that features can take any value. However, the same formulation can be adapted when features take a given (fixed) value. Observe that this will be instrumental for computing LmPAXp's.

Consider that feature k is fixed to value l. Then, the formulation for $C(k, r)$ becomes:

$$C(k, r) = n_k^l \times C(k-1, r - w_k^l) = C(k-1, r - w_k^l)$$

Given that k is fixed, then it is the case that $n_k^l = 1$.

Example 4. For Example 3, assume that $x_2 = 1$ and $x_4 = 3$. Then, the constraint we want to satisfy is:

$$\sum_{j \in \{1,3\}} \sigma(x_j, 1, 2, 3) > 3$$

Following a similar transformation into knapsack formulation, we get

$$\sum_{j \in \{1,3\}} \sigma(x_j, 3, 2, 1) < 5$$

After updating the DP table, with fixing features 2 and 4, we get the DP table shown in Table 2. As a result, we can conclude that the number of solutions is 6.

The table $C(k, r)$ can be filled out in pseudo-polynomial time. The number of rows is m. The number of columns is W (see (20)). Moreover, the computation of each entry uses the values of at most m other entries. Thus, the total running time is: $\Theta(m^2 \times W)$.

Table 2
DP table for Example 4

k \ r	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	2	3	3	3	3	3	3	–	–	–	–
2	0	0	0	0	1	2	3	3	3	–	–	–	–
3	0	0	0	0	0	1	3	6	8	–	–	–	–
4	0	0	0	0	0	0	1	3	6	–	–	–	–

From XLCs to Positive Integer Knapsacks. To assess heuristic explainers, we consider NBCs, and use a standard transformation from probabilities to positive real values [54]. Afterwards, we convert the real values to integer values by scaling the numbers. However, to avoid building a very large DP table, we implement the following optimization. The number of decimal places of the probabilities is reduced while there is no decrease in the accuracy of the classifier both on training and on test data. In our experiments, we observed that there is no loss of accuracy if four decimal places are used, and that there is a negligible loss of accuracy with three decimal places.

Assessing explanation precision. Given a Naive Bayes classifier, expressed as an XLC, we can assess explanation accuracy in pseudo-polynomial time. Given an instance $\mathbf{v}$, a prediction $\kappa(\mathbf{v}) = \oplus$, and an approximate explanation $\mathbf{S}$, we can use the approach described in this section to count the number of instances consistent with the explanation for which the prediction remains unchanged (i.e. number of points $\mathbf{x} \in \mathbb{F}$ s.t. $(\kappa(\mathbf{x}) = \kappa(\mathbf{v}) \land (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}))$). Let this number be $n_{\oplus}$ (given the assumption that the prediction is $\oplus$). Let the number of instances with a different prediction $(\ominus \neq \kappa(\mathbf{v}))$[2] be $n_{\ominus}$. Hence, the conditional probability (2) can be defined, in the case of NBCs, as follow:

$$\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \oplus \,|\, \mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}) = \frac{n_{\oplus}}{|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})\}|}$$

Observe that the numerator $|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = \oplus \land (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})\}|$ is expressed by the number of models $n_{\oplus}$, i.e. the points $\mathbf{x}$ in feature space that are consistent with $\mathbf{v}$ given $\mathcal{S}$ and with prediction $\oplus$. Further, we have

$$\begin{aligned}\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \oplus \,|\, \mathbf{x}_{\mathcal{S}} = &\mathbf{v}_{\mathcal{S}}) = \\ &1 - \Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \ominus \,|\, \mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}}) \\ &= 1 - \frac{n_{\ominus}}{|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})\}|}\end{aligned}$$

where $n_{\ominus} = |\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = \ominus \land (\mathbf{x}_{\mathcal{S}} = \mathbf{v}_{\mathcal{S}})\}|$.

[2] As we are in binary setting, then $\ominus = \neg\oplus = \neg\kappa(\mathbf{v})$).

4.3. Computing LmPAXp's

Algorithm 1 depicts our method for computing LmPAXp's given a prediction function κ of an NBC, an input instance $\mathbf{v}$ and a threshold δ. The procedure LmPAXp is referred to as a deletion-based algorithm[3]; it starts from a set of features $\mathcal{S}$, e.g. initialized to $\mathcal{F}$ and then it iteratively drops features while the updated set $\mathcal{S}$ remains a WeakPAXp. The function isWeakPAXp implements the approach described in the previous section, which measures explanation precision by exploiting a pseudo-polynomial algorithm for model counting. Hence, it is implicit that the DP table is updated at each iteration of the loop in the LmPAXp procedure. More specifically, when a feature i is newly set universal, its associated cells $C(i, r)$ are recalculated such that $C(k, r) = \sum_{i=1}^{d'_k} n_k^i \times C(k-1, r-w_k^i)$; and when i is fixed, i.e. $i \in \mathcal{S}$, then $C(i, r) = C(i-1, r-v_i^j)$ where $v_i^j \triangleq \mathrm{lPr}(v_i = j|c) - \mathrm{lPr}(v_i = j|\neg c)$. Furthermore, we point out that in our experiment, $\mathcal{S}$ is initialized to an AXp $\mathcal{X}$ that we compute initially for all tested instances using the outlined (polynomial) algorithm in Section 3. It is easy to observe that features not belonging to $\mathcal{X}$ do not contribute in the decision of $\kappa(\mathbf{v})$ (i.e. their removal does not change the value of $n_{\ominus}$ that equals to zero) and thus can be set universal at the initialisation step, which allows us to improve the performance of Algorithm 1.

Moreover, we apply an heuristic order over $\mathcal{S}$ that aims to remove earlier less relevant features and thus to produce shorter approximate explanations. Typically, we order $\mathcal{S}$ following the increasing order of δ_i values, namely the reverse order applied to compute the AXp. Conducted preliminary experiments using a (naive heuristic) lexicographic order over the features show less succinct explanations.

Finally, notice that Algorithm 1 can be used to compute an AXp, i.e. an LmPAXp with $\delta = 1$. Nevertheless, the polynomial time algorithm for computing AXp's proposed in [43] remains a better choice to use in case of AXp's than Algorithm 1 which runs in pseudo polynomial time.

Example 5. Let us consider again the NBC of the running example (Example 1) and $\mathbf{v} = (\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$. The corresponding XLC is shown in Figure 3b (Example 2). Also, consider the AXp $\{1, 2, 5\}$ of $\mathbf{v}$ and $\delta = 0.85$. The resulting DP table for $\mathcal{S} = \{1, 2, 5\}$ is shown in Table 3. Note that for illustrating small tables, we set the number of decimal places to zero (greater number of decimal places, i.e. 1,2, etc, were tested and return the results). (Also, note that the DP table reports "–" if the cell is not calculated during the running of Algorithm 1.) Moreover,

[3] This sort of algorithm can be traced at least to the work of Valiant[55], but some authors [56] argue that it is also implicit in works from the 19th century [57].

Algorithm 1 Computing one LmPAXp

Input: Classifier κ, instance $\mathbf{v}$, threshold δ
Output: LmPAXp $\mathcal{S}$

procedure LmPAXp($\kappa, \mathbf{v}, \delta$)
  $\mathcal{S} \leftarrow \{1, \ldots, m\}$
  for $i \in \{1, \ldots, m\}$ **do**
    if isWeakPAXp($\mathcal{S} \setminus \{i\}, \kappa(\mathbf{x}) = c, \delta$) **then**
      $\mathcal{S} \leftarrow \mathcal{S} \setminus \{i\}$
  return $\mathcal{S}$

we convert the probabilities into positive integers, so we sum to each w_j^k the complement of the most negative w_j^k plus 1. The resulting weights are shown in Figure 4. Thus, we get $\sum_{i \in \{1,2,3,4,5\}} \sigma(x_i, w_i^1, w_i^2) < 17$. Observe that the number of models $n_{\oplus} = C(5, 16)$, and $C(5, 16)$ is calculated using $C(4, 16 - w_5^2) = C(4, 15)$, i.e. $C(4, 15) = C(5, 16)$ (feature 5 is fixed, so it is allowed to take only the value $w_5^2 = 1$). Next, $C(4, 15) = C(3, 15 - w_4^1) + C(3, 15 - w_4^2) = C(3, 12) + C(3, 14)$ (feature 4 is free, so it is allowed to take any value of $\{w_4^1, w_4^2\}$); the recursion ends when k=1, namely for $C(1, 5) = C(2, 6) = n_1^2 = 1$, $C(1, 7) = C(2, 7) = n_1^2 = 1$, $C(1, 8) = C(2, 8) = n_1^2 = 1$ and $C(1, 10) = C(2, 11) = n_1^2 = 1$ (feature 1 is fixed and takes value w_1^2). Next, Table 4 (resp. Table 5 and Table 6) report the resulting DP table for $\mathcal{S} = \{2, 5\}$ (resp. $\mathcal{S} = \{1, 5\}$ and $\mathcal{S} = \{1\}$). It is easy to confirm that after dropping feature 2, the precision of $\mathcal{S} = \{1, 5\}$ becomes 87.5%, i.e. $\frac{7}{8} = 0.875 > \delta$. Furthermore, observe that the resulting $\mathcal{S}$ when dropping feature 1 or 2 and 5, are not WeakPAXp's, namely, the precision of $\{2, 5\}$ is $\frac{6}{8} = 0.75 < \delta$ and the precision of $\{1\}$ is $\frac{9}{16} = 0.5625 < \delta$. In summary, Algorithm 1 starts with $\mathcal{S} = \{1, 2, 5\}$, then at iteration#1, feature 1 is tested and since $\{2, 5\}$ is not WeakPAXp then 1 is kept in $\mathcal{S}$; at iteration#2, feature 2 is tested and since $\{1, 5\}$ is a WeakPAXp, then $\mathcal{S}$ is updated (i.e. $\mathcal{S} = \{1, 5\}$); at iteration#3, feature 5 is tested and since $\{1\}$ is not a WeakPAXp, then 5 is saved in $\mathcal{S}$. As a result, the delivered LmPAXp is $\{1, 5\}$.

Let us underline that we could initialize $\mathcal{S}$ to $\mathcal{F}$, in which case the number of models would be 1. However, we opt instead to always start from an AXp. In the example, the AXp is $\{1, 2, 5\}$ which, because it is an AXp, the number of models must be 4 (i.e. 2^2, since two features are free).

For any proper subset of the AXp, with r free variables, it must be the case that the number of models is strictly less than 2^r. Otherwise, we would have an AXp as a proper subset of another AXp; but this would contradict the definition of AXp. The fact that the number of models is strictly less than 2^r is confirmed by the examples of subsets considered. It must also be the case that if $\mathcal{S}' \subseteq \mathcal{S}$, then the number of models of $\mathcal{S}'$ must not exceed the number of models of $\mathcal{S}$. So, we can argue that there is monotonicity in the number of models, but not on the precision.

W	w_1^1	w_1^2	w_2^1	w_2^2	w_3^1	w_3^2	w_4^1	w_4^2	w_5^1	w_5^2
16	7	1	1	6	3	6	1	3	7	1

Figure 4: #knapsack problem of Example 5

Properties of LmPAXp's. In addition to the comments above, and by carefully computing LmPAXp's, these can exhibit important properties. Let $\mathcal{X}$ denote an AXp. Then, for any LmPAXp $\mathcal{A}$ obtained using $\mathcal{X}$ as the seed, i.e. $\mathcal{A}$ is required to be a subset of $\mathcal{X}$, then we have the following properties:

1. $\mathcal{A} \subseteq \mathcal{X}$;
2. There exists a probabilistic abductive explanation $\mathcal{E}$ such that $\mathcal{E} \subseteq \mathcal{A}$; and
3. $\mathcal{A}$ is a δ-relevant set (see Definition 2.1).

Thus, an LmPAXp $\mathcal{A}$ can be made to be a superset of some PAXp, a subset of some AXp, and such that $\mathcal{A}$ exhibits the strong probabilistic properties of δ-relevant sets.

5. Experimental Results

This section evaluates the algorithm proposed for computing LmPAXp's. The evaluation aims at assessing not only the succinctness and precision of computed explanations but also the scalability of our solution.

5.1. Experimental setup

The benchmarks used in the experiments comprise publicly available and widely used datasets that originate from UCI ML Repository[4] and Penn ML Benchmarks[5]. The number of training data (resp. features) in the target datasets varies from 336 to 14113 (resp. 10 to 37) and on average is 3999.1 (resp. 20.0). All the NBCs are trained using the learning tool *scikit-learn*[6]. The data split for training and test data is set to 80% and 20%, respectively. Model accuracies are above 80% for the training accuracy and above 75% for the test accuracy.

A prototype implementation of the proposed approach for computing relevant sets is developed in Python. To compute AXp's, we use the Perl script implemented by [43].The prototype implementation was tested with varying the threshold $\delta \in \{0.90, 0.93, 0.95, 0.98\}$. When

[4]https://archive.ics.uci.edu
[5]https://epistasislab.github.io/pmlb/
[6]https://scikit-learn.org

Table 3
DP table for $\mathcal{S} = \{1, 2, 5\}$ (Example 5)

k \ r	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	—	—	—	—	1	—	1	1	—	1	—	—	—	—	—	—
2	0	—	—	—	—	—	1	—	1	1	—	1	—	—	—	—	—
3	0	—	—	—	—	—	—	—	—	—	—	—	2	—	2	—	—
4	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	4	—
5	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	4

Table 4
DP table for $\mathcal{S} = \{2, 5\}$ (Example 5)

k \ r	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	—	—	—	—	1	—	1	2	—	2	—	—	—	—	—	—
2	0	—	—	—	—	—	1	—	1	2	—	2	—	—	—	—	—
3	0	—	—	—	—	—	—	—	—	—	—	—	3	—	3	—	—
4	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	6	—
5	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	6

converting probabilities from real values to integer values, the selected number of decimal places is 3. (As outlined earlier, we observed that there is a negligible accuracy loss from using three decimal places.) In order to produce explanations of size admissible for the cognitive capacity of human decision makers [44], we selected three different target sizes for the explanations to compute: 9, 7 and 4, and we compute a LmPAXp for the input instance when its AXp $\mathcal{X}$ is larger than the target size (recall that $\mathcal{S}$ is initialized to $\mathcal{X}$); otherwise we consider the AXp is succinct and the explainer returns $\mathcal{X}$. For example, assume the target size is 7, an instance $\mathbf{v}_1$ with an AXp $\mathcal{S}_1$ of 5 features and an second instance $\mathbf{v}_2$ with an AXp $\mathcal{S}_2$ of 8 features, then for $\mathbf{v}_1$ the output will be $\mathcal{S}_1$ and for $\mathbf{v}_2$ the output will be a subset of $\mathcal{S}_2$.

For each dataset, we run the explainer on 200 instances

Table 5
DP table for $\mathcal{S} = \{1, 5\}$ (Example 5)

k \ r	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	—	1	1	—	1	—	1	1	—	1	—	—	—	—	—	—
2	0	—	—	—	—	—	1	—	2	2	—	2	—	—	—	—	—
3	0	—	—	—	—	—	—	—	—	—	—	—	3	—	4	—	—
4	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	7	—
5	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	7

Table 6
DP table for $\mathcal{S} = \{1\}$ (Example 5)

k \ r	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	1	1	1	1	—	1	1	—	1	—	—	—	—	—	—
2	0	—	0	1	—	1	1	—	2	2	—	2	—	—	—	—	—
3	0	—	—	—	—	—	1	—	1	—	—	—	3	—	4	—	—
4	0	—	—	—	—	—	—	—	—	2	—	—	—	—	—	7	—
5	0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	9

Table 7
Assessing ApproxPAXp explanations for NBCs. Columns **m** and **#I** show, respectively, number of features and tested instances in the Dataset. Column **A%** reports in (%) the training accuracy of the classifier. Column δ reports in (%) the used value of the parameter δ. **LmPAXp**$_{\leq 9}$ and **LmPAXp**$_{\leq 7}$ denote, respectively, LmPAXp's of (target) length 9 and 7. Columns **Length** and **Precision** report, respectively, the average explanation length and the average explanation precision ($\pm$ denotes the standard deviation). **W%** shows in (%) the number of success/wins where the explanation size is less or equal than the target size. Finally, the average runtime to compute an explanation is shown (in seconds) in **Time**.

Dataset	(m	#I)	NBC	AXp	LmPAXp$_{\leq 9}$				LmPAXp$_{\leq 7}$			
			A%	Length	Length	Precision	W%	Time	Length	Precision	W%	Time
adult	(13	200)	81.37	6.8± 1.2	6.8± 1.1	99.99± 0.2	100	0.074	5.9± 1.0	98.87± 1.8	99	0.058
agaricus	(23	200)	95.41	10.3± 2.5	6.9± 3.1	97.62± 2.1	95	0.954	5.3± 3.2	96.59± 1.6	92	1.273
chess	(37	200)	88.34	12.1± 3.7	7.7± 3.8	98.51± 1.4	68	0.404	5.5± 4.4	97.90± 0.9	64	0.483
vote	(17	81)	89.66	5.3± 1.4	5.3± 1.4	100± 0.0	100	0.000	5.3± 1.3	99.93± 0.3	100	0.008
kr-vs-kp	(37	200)	88.07	12.2± 3.9	7.3± 3.9	98.29± 1.4	64	0.416	6.0± 4.3	97.89± 1.1	64	0.453
mushroom	(23	200)	95.51	10.7± 2.3	6.5± 2.6	97.35± 1.8	96	1.011	5.1± 2.5	96.52± 1.0	90	1.130
threeOf9	(10	103)	83.13	4.2± 0.4	4.2± 0.4	100± 0.0	100	0.000	4.2± 0.4	100± 0.0	100	0.000
xd6	(10	176)	81.36	4.5± 0.9	4.5± 0.8	100± 0.0	100	0.000	4.5± 0.8	100± 0.0	100	0.000
mamo	(14	53)	80.21	4.9± 0.8	4.9± 0.7	100± 0.0	100	0.000	4.9± 0.7	100± 0.0	100	0.000
tumor	(16	104)	83.21	5.3± 0.9	5.3± 0.8	100± 0.0	100	0.000	5.2± 0.6	99.83± 0.7	100	0.012

randomly picked from the test data or on all instances if there are less than 200.

The experiments are conducted on a MacBook Air with a 1.1GHz Quad-Core Intel Core i5 CPU with 16 GByte RAM running macOS Monterey.

5.2. Results

Table 7 summarizes the results of our experiments for $\delta = 0.95$ and target size 9 and 7. The complete results of the empirical evaluation are reported in [1]. (Note that the same observations are perceived on the results obtained with the remaining parameters, i.e. $\delta \in \{0.90, 0.93, 0.98\}$ and LmPAXp≤ 4.) As can be observed for all considered settings, the locally-minimal explanations are succinct, in particular the average sizes of the explanations are invariably lower than the target sizes. Moreover, theses explanations offer strong guarantees of precision, as their average precisions are strictly greater than δ with significant gaps (e.g. above 97%, in column LmPAXp$_{\leq 7}$, for datasets *adult*, *vote*, *threeOf9*, *xd6*, *mamo* and *tumor* and above 95% for *chess* and *kr-vs-kp*). An important observation from the results, is the gain of succinctness (explanation size) when comparing AXp's with LmPAXp's. In fact, for some datasets, the AXp's are too large (e.g. for *chess* and *kr-vs-kp* datasets, the average number of features in the AXp's is 12), exceeding the cognitive limits of human decision makers [44] (limited to 7 ± 2 features). To illustrate that, one can focus on the dataset *agaricus* or *mushroom* and see that for a target size equals to 7, the average length of the LmPAXp's (i.e. 5.3 and 5.1, resp.) is 2 times less than the average length of the AXp's (i.e. 10.3 and 10.7, resp.). Besides, the results show that $\delta = 0.95$ is a good probability threshold to guarantee highly precise and short approximate explanations.

Despite the complexity of the proposed approach being in pseudo polynomial, the results demonstrate that in practice the algorithm is effective and scales for large datasets. As can be seen, the runtimes are negligible for all datasets, never exceeding 2 seconds for the largest datasets (i.e. *agaricus* or *mushroom*) and the average is 0.33 seconds for all tested instances across all datasets and all settings. Furthermore, we point out that the implemented prototype was tested with 4 decimal places to assess further the scalability of the algorithm on larger DP tables, and the results show that computing LmPAXp's is still feasible, e.g. with *agaricus* the average runtime when the target size set to 7 is 10.08 seconds.

The table also reports the number of explanations being shorter than or of size equal to the target size over the total number of tested instances. We observe that for both settings LmPAXp$_{\leq 9}$ and LmPAXp$_{\leq 7}$ and for the majority of datasets and with a few exceptions the fraction is significantly high, e.g. varying for 96% to 100% for *adult* dataset. However, in our assessment we observed that for LmPAXp$_{\leq 4}$ despite the poor percentage of wins for some datasets, it is the case that the average lengths of computed explanations are close to 4 (see Table 13 in [1]).

Overall, the experiments demonstrate that our approach efficiently computes succinct and provably precise explanations for NBCs. The results also showcase empirically the advantage of the algorithm, i.e. in practice one may rely on the computation of LmPAXp's, which pays off in terms of (1) performance, (2) succinctness and (3) sufficiently high probabilistic guarantees of precision.

6. Conclusion

This paper builds on recent work on computing rigorous probabilistic explanations [42], and investigates the concrete case of NBCs. The paper proposes a pseudo-polynomial algorithm for computing the number of points in feature space predicting a specific class, and relates this problem with that of computing a rigorous probabilistic explanation. Furthermore, the paper proposes an algorithm for computing locally minimal probabilistic explanations, which offers strong guarantees in terms of precision. The experimental results confirm that short and precise probabilistic explanations can be efficiently computed in the case of NBCs.

Two lines of future work can be envisioned. One line is to investigate the complexity of explaining multi-class NBCs and extend the approach for computing locally minimal probabilistic explanations for multi-class Naive Bayes models. Furthermore, one might be interested in computing smallest probabilistic explanations instead of approximates. Hence, another line of research is to devise a logical (Satisfiability Modulo Theories, SMT) encoding for computing cardinality minimal probabilistic explanations.

Acknowledgments

This work was supported by the AI Interdisciplinary Institute ANITI, funded by the French program “Investing for the Future – PIA3” under Grant agreement no. ANR-19-PI3A-0004, and by the H2020-ICT38 project COALA “Cognitive Assisted agile manufacturing for a Labor force supported by trustworthy Artificial intelligence”, and by the National Research Foundation, Prime Minister’s Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

[1] Y. Izza, X. Huang, A. Ignatiev, N. Narodytska, M. C. Cooper, J. Marques-Silva, On computing probabilistic abductive explanations, Int. J. Approx. Reason. 159 (2023) 108939.

[2] EU, Artificial Intelligence Act, http://tiny.cc/ahcnuz, 2021.

[3] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (2019) 206–215.

[4] G. Montavon, W. Samek, K. Müller, Methods for interpreting and understanding deep neural networks, Digit. Signal Process. 73 (2018) 1–15.

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

[6] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, K. Müller (Eds.), Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Springer, 2019.

[7] W. Samek, K. Müller, Towards explainable artificial intelligence, in: [6], 2019, pp. 5–22.

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

[9] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, K. Müller, Explaining deep neural networks and beyond: A review of methods and applications, Proc. IEEE 109 (2021) 247–278.

[10] M. T. Ribeiro, S. Singh, C. Guestrin, "why should I trust you?": Explaining the predictions of any classifier, in: KDD, 2016, pp. 1135–1144.

[11] S. M. Lundberg, S. Lee, A unified approach to interpreting model predictions, in: NeurIPS, 2017, pp. 4765–4774.

[12] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, in: AAAI, 2018, pp. 1527–1535.

[13] A. Ignatiev, Towards trustable explainable AI, in: IJCAI, 2020, pp. 5154–5158.

[14] O. Camburu, E. Giunchiglia, J. Foerster, T. Lukasiewicz, P. Blunsom, Can I trust the explainer? verifying post-hoc explanatory methods, CoRR abs/1910.02065 (2019).

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

[16] H. Lakkaraju, O. Bastani, "how do I fool you?": Manipulating user trust via misleading black box explanations, in: AIES, 2020, pp. 79–85.

[17] B. Dimanov, U. Bhatt, M. Jamnik, A. Weller, You shouldn’t trust me: Learning models which conceal unfairness from multiple explanation methods, in: ECAI, 2020, pp. 2473–2480.

[18] A. Shih, A. Choi, A. Darwiche, A symbolic approach to explaining bayesian network classifiers, in: IJCAI, 2018, pp. 5103–5111.

[19] A. Ignatiev, N. Narodytska, J. Marques-Silva, Abduction-based explanations for machine learning models, in: AAAI, 2019, pp. 1511–1519.

[20] A. Ignatiev, N. Narodytska, J. Marques-Silva, On relating explanations and adversarial examples, in: NeurIPS, 2019, pp. 15857–15867.

[21] N. Narodytska, A. A. Shrotri, K. S. Meel, A. Ignatiev, J. Marques-Silva, Assessing heuristic machine learning explanations with model counting, in: SAT, 2019, pp. 267–278.

[22] Y. Izza, A. Ignatiev, J. Marques-Silva, On explaining decision trees, CoRR abs/2010.11034

(2020). URL: https://arxiv.org/abs/2010.11034. arXiv:2010.11034.

[23] A. Ignatiev, N. Narodytska, N. Asher, J. Marques-Silva, From contrastive to abductive explanations and back again, in: AIxIA, 2020, pp. 335–355.

[24] A. Darwiche, A. Hirth, On the reasons behind decisions, in: ECAI, 2020, pp. 712–720.

[25] G. Audemard, F. Koriche, P. Marquis, On tractable XAI queries based on compiled representations, in: KR, 2020, pp. 838–849.

[26] Y. Izza, J. Marques-Silva, On explaining random forests with SAT, in: IJCAI, 2021, pp. 2584–2591.

[27] X. Huang, Y. Izza, A. Ignatiev, J. Marques-Silva, On efficiently explaining graph-based classifiers, in: KR, 2021, pp. 356–367.

[28] M. C. Cooper, J. Marques-Silva, On the tractability of explaining decisions of classifiers, in: L. D. Michel (Ed.), CP, 2021, pp. 21:1–21:18.

[29] A. Ignatiev, J. Marques-Silva, N. Narodytska, P. J. Stuckey, Reasoning-based learning of interpretable ML models, in: IJCAI, 2021, pp. 4458–4465.

[30] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, N. Narodytska, Explanations for monotonic classifiers, in: ICML, 2021, pp. 7469–7479.

[31] A. Ignatiev, J. Marques-Silva, SAT-based rigorous explanations for decision lists, in: SAT, 2021, pp. 251–269.

[32] E. L. Malfa, R. Michelmore, A. M. Zbrzezny, N. Paoletti, M. Kwiatkowska, On guaranteed optimal robust explanations for NLP models, in: IJCAI, 2021, pp. 2658–2665.

[33] R. Boumazouza, F. C. Alili, B. Mazure, K. Tabia, ASTERYX: A model-agnostic sat-based approach for symbolic and score-based explanations, in: CIKM, 2021, pp. 120–129.

[34] N. Gorji, S. Rubin, Sufficient reasons for classifier decisions in the presence of domain constraints, in: AAAI, 2022.

[35] X. Huang, Y. Izza, A. Ignatiev, M. C. Cooper, N. Asher, J. Marques-Silva, Tractable explanations for d-DNNF classifiers, in: AAAI, 2022.

[36] A. A. Shrotri, N. Narodytska, A. Ignatiev, K. Meel, J. Marques-Silva, M. Vardi, Constraint-driven explanations of black-box ML models, in: AAAI, 2022.

[37] A. Ignatiev, Y. Izza, P. Stuckey, J. Marques-Silva, Using MaxSAT for efficient explanations of tree ensembles, in: AAAI, 2022.

[38] J. Marques-Silva, A. Ignatiev, Delivering trustworthy AI through formal XAI, in: AAAI, 2022.

[39] Y. Izza, A. Ignatiev, J. Marques-Silva, On tackling explanation redundancy in decision trees, J. Artif. Intell. Res. 75 (2022) 261–321.

[40] J. Yu, A. Ignatiev, P. J. Stuckey, N. Narodytska, J. Marques-Silva, Eliminating the impossible, whatever remains must be true, CoRR (2022).

[41] E. Wang, P. Khosravi, G. V. den Broeck, Probabilistic Sufficient Explanations, in: IJCAI, 2021, pp. 3082–3088.

[42] S. Wäldchen, J. MacDonald, S. Hauch, G. Kutyniok, The computational complexity of understanding binary classifier decisions, J. Artif. Intell. Res. 70 (2021) 351–387.

[43] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, N. Narodytska, Explaining naive bayes and other linear classifiers with polynomial time and delay, in: NeurIPS, 2020.

[44] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information., Psychological review 63 (1956) 81–97.

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

[46] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, John Wiley & Sons, 1973.

[47] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (1997) 131–163. URL: https://doi.org/10.1023/A:1007465528199. doi:10.1023/A:1007465528199.

[48] D. Barber, Bayesian reasoning and machine learning, Cambridge University Press, 2012.

[49] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack problems, Springer, 2004.

[50] M. E. Dyer, Approximate counting by dynamic programming, in: STOC, 2003, pp. 693–699.

[51] P. Gopalan, A. R. Klivans, R. Meka, D. Stefankovic, S. S. Vempala, E. Vigoda, An FPTAS for #knapsack and related counting problems, in: FOCS, 2011, pp. 817–826.

[52] P. Gawrychowski, L. Markin, O. Weimann, A faster FPTAS for #knapsack, in: ICALP, 2018, pp. 64:1–64:13.

[53] R. Rizzi, A. I. Tomescu, Faster fptases for counting and random generation of knapsack solutions, Inf. Comput. 267 (2019) 135–144. URL: https://doi.org/10.1016/j.ic.2019.04.001. doi:10.1016/j.ic.2019.04.001.

[54] J. D. Park, Using weighted MAX-SAT engines to solve MPE, in: AAAI, 2002, pp. 682–687.

[55] L. G. Valiant, A theory of the learnable, Commun. ACM 27 (1984) 1134–1142.

[56] B. Juba, Learning abductive reasoning using random examples, in: AAAI, 2016, pp. 999–1007.

[57] J. S. Mill, A System of Logic, Ratiocinative and Inductive, volume 1, John W. Parker, 1843.