Journal of Artificial Intelligence Research 74 (2022) 851-886 Submitted 08/2021; published 06/2022

# On the Tractability of SHAP Explanations

**Guy Van den Broeck** GUYVDB@CS.UCLA.EDU
*University of California*
*404 Westwood Plaza, Los Angeles, CA 90095, USA*

**Anton Lykov** ALYKOV@CS.WASHINGTON.EDU
**Maximilian Schleich** SCHLEICH@CS.WASHINGTON.EDU
**Dan Suciu** SUCIU@CS.WASHINGTON.EDU
*University of Washington*
*185 E Stevens Way NE, Seattle, WA 98195, USA*

## Abstract

SHAP explanations are a popular feature-attribution mechanism for explainable AI. They use game-theoretic notions to measure the influence of individual features on the prediction of a machine learning model. Despite a lot of recent interest from both academia and industry, it is not known whether SHAP explanations of common machine learning models can be computed efficiently. In this paper, we establish the complexity of computing the SHAP explanation in three important settings. First, we consider fully-factorized data distributions, and show that the complexity of computing the SHAP explanation is the same as the complexity of computing the expected value of the model. This fully-factorized setting is often used to simplify the SHAP computation, yet our results show that the computation can be intractable for commonly used models such as logistic regression. Going beyond fully-factorized distributions, we show that computing SHAP explanations is already intractable for a very simple setting: computing SHAP explanations of trivial classifiers over naive Bayes distributions. Finally, we show that even computing SHAP over the empirical distribution is #P-hard.

## 1. Introduction

Machine learning is increasingly applied in high stakes decision making. As a consequence, there is growing demand for the ability to explain the prediction of machine learning models. One popular explanation technique is to compute feature-attribution scores, in particular using the Shapley values from cooperative game theory (Roth, 1988) as a principled aggregation measure to determine the influence of individual features on the prediction of the collective model. Shapley value based explanations have several desirable properties (Datta, Sen, and Zick, 2016), which is why they have attracted a lot of interest in academia as well as industry in recent years (see e.g., Gade et al. (2019)).

Štrumbelj and Kononenko (2014) show that Shapley values can be used to explain arbitrary machine learning models. Datta, Sen, and Zick (2016) use Shapley-value-based explanations as part of a broader framework for algorithmic transparency. Lundberg and Lee (2017) use Shapley values in a framework that unifies various explanation techniques, and they coined the term SHAP explanation. They show that the SHAP explanation is effective in explaining predictions in the medical domain; see Lundberg et al. (2020). More recently there has been a lot of work on the tradeoffs of variants of the original SHAP

explanations, e.g., Sundararajan and Najmi (2020), Kumar et al. (2020), Janzing, Minorics, and Bloebaum (2020), Merrick and Taly (2020), and Aas, Jullum, and Løland (2019).

Despite all of this interest, there is considerable confusion about the tractability of computing Shap explanations. The Shap explanations determine the influence of a given feature by systematically computing the expected value of the model given a subsets of the features. As a consequence, the complexity of computing Shap explanations depends on the predictive model as well as assumptions on the underlying data distribution. Lundberg et al. (2020) describe a polynomial-time algorithm for computing the Shap explanation over decision trees, but online discussions have pointed out that this algorithm is not correct as stated. We present a concrete example of this shortcoming in Section 2.3. In contrast, for fully-factorized distributions, Bertossi et al. (2020) prove that there are models for which computing the Shap explanation is #P-hard. A contemporaneous paper by Arenas et al. (2020) shows that computing the Shap explanation for tractable logical circuits over uniform and fully factorized binary data distributions is tractable. In general, the complexity of the Shap explanation is open.

In this paper we consider the original formulation of the Shap explanation by Lundberg and Lee (2017) and analyze its computational complexity under the following data distributions and model classes:

1. First, we consider fully-factorized distributions, which are the simplest possible data distribution. Fully-factorized distributions capture the assumption that the model's features are independent, which is a commonly used assumption to simplify the computation of the Shap explanations, see for example Lundberg and Lee (2017).

   For fully-factorized distributions and any prediction model, we show that the complexity of computing the Shap explanation is the same as the complexity of computing the expected value of the model.

   It follows that there are classes of models for which the computation is tractable (e.g., linear regression, decision trees, tractable circuits) while for other models, including commonly used ones such as logistic regression and neural nets with sigmoid activation functions, it is #P-hard.

2. Going beyond fully-factorized distributions, we show that computing Shap explanation becomes intractable already for the simplest probabilistic model that does not assume feature independence: naive Bayes. As a consequence, the complexity of computing Shap explanations on such data distributions is also intractable for many classes of models, including linear and logistic regression.

3. Finally we consider the empirical distribution, and prove that computing Shap explanations is #P-hard for this class of distributions. This result implies that the algorithm by Lundberg et al. (2020) cannot be fixed to compute the exact Shap explanations over decision trees in polynomial time.

## 2. Background and Problem Statement

Suppose our data is described by $n$ indexed features $\mathbf{X} = \{X_1, \ldots, X_n\}$. Each feature variable $X$ takes a value from a finite domain $\text{dom}(X)$. A data instance $\mathbf{x} = (x_1, \ldots, x_n)$

consists of values $x \in \mathrm{dom}(X)$ for every feature $X$. This instance space is denoted $\mathbf{x} \in \mathcal{X} = \mathrm{dom}(X_1) \times \cdots \times \mathrm{dom}(X_n)$. We are also given a learned function $F : \mathcal{X} \to \mathbb{R}$ that computes a prediction $F(\mathbf{x})$ on each instance $\mathbf{x}$. Throughout this paper we assume that the prediction $F(\mathbf{x})$ can be computed in polynomial time in $n$.

For a particular instance of prediction $F(\mathbf{x})$, the goal of *local explanations* is to clarify why the function $F$ gave its prediction on instance $\mathbf{x}$, usually by attributing credit to the features. We will focus on local explanation that are inspired by game-theoretic Shapley values (Datta, Sen, and Zick, 2016; Lundberg and Lee, 2017). Specifically, we will work with the SHAP explanations as defined by Lundberg and Lee (2017).

### 2.1 Shap Explanations

To produce SHAP explanations, one needs an additional ingredient: a probability distribution $\Pr(\mathbf{X})$ over the features, which we call the data distribution. We will use this distribution to reason about partial instances. Concretely, for a set of indices $S \subseteq [n] = \{1, \ldots, n\}$, we let $\mathbf{x}_S$ denote the restriction of complete instance $\mathbf{x}$ to those features $\mathbf{X}_S$ with indices in $S$. Abusing notation, we will also use $\mathbf{x}_S$ to denote the probabilistic event $\mathbf{X}_S = \mathbf{x}_S$.

Under this data distribution, it now becomes possible to ask for the *expected value* of the predictive function $F$. Clearly, for a complete data instance $\mathbf{x}$ we have that $\mathbf{E}[F|\mathbf{x}] = F(\mathbf{x})$, as there is no uncertainty about the features. However, for a partial instance $\mathbf{x}_S$, which does not assign values to the features outside of $\mathbf{X}_S$, we appeal to the data distribution Pr to compute the expectation of function $F$ as $\mathbf{E}_{\Pr}[F|\mathbf{x}_S] = \sum_{\mathbf{x} \in \mathcal{X}} F(\mathbf{x}) \Pr(\mathbf{x}|\mathbf{x}_S)$.

The SHAP explanation framework draws from Shapley values in cooperative game theory. Given a particular instance $\mathbf{x}$, it considers features $\mathbf{X}$ to be players in a coalition game: the game of making a prediction for $\mathbf{x}$. SHAP explanations are defined in terms of a set function $v_{F,\mathbf{x},\Pr} : 2^{\mathbf{X}} \to \mathbb{R}$. Its purpose is to evaluate the *"value"* of each coalition of players/features $\mathbf{X}_S \subseteq \mathbf{X}$ in making the prediction $F(\mathbf{x})$ under data distribution Pr. Concretely, following Lundberg and Lee (2017), this value function is the conditional expectation of function $F$:

$$\mathrm{v}_{F,\mathbf{x},\Pr}(\mathbf{X}_S) \stackrel{\text{def}}{=} \mathbf{E}_{\Pr}[F|\mathbf{x}_S]. \tag{1}$$

We will elide $F$, $\mathbf{x}$, and Pr when they are clear from context.

Our goal, however, is to assign credit to individual features. In the context of a coalition $\mathbf{X}_S$, the *contribution* of an individual feature $X \notin \mathbf{X}_S$ is given by

$$\mathrm{c}(X, \mathbf{X}_S) \stackrel{\text{def}}{=} \mathrm{v}(\mathbf{X}_S \cup \{X\}) - \mathrm{v}(\mathbf{X}_S). \tag{2}$$

where each term is implicitly w.r.t. the same $F$, $\mathbf{x}$, and Pr.

Finally, the SHAP explanation computes a score for each feature $X \in \mathbf{X}$ averaged over all possible contexts, and thus measures the influence feature $X$ has on the outcome. Let $\pi$ be a permutation on the set of features $\mathbf{X}$, i.e., $\pi$ fixes a total order on all features. Let $\pi^{<X}$ be the set of features that come before $X$ in the order $\pi$. The SHAP explanations are then defined as computing the following scores.

**Definition 1** (SHAP Score)**.** Fix an entity $\mathbf{x}$, a predictive function $F$, and a data distribution Pr. The SHAP explanation of a feature $X$ is the contribution of $X$ given the features $\pi^{<X}$,

averaged over all permutations $\pi$:

$$\textsc{Shap}(X) \stackrel{\text{def}}{=} \frac{1}{n!}\sum_{\pi} \mathrm{c}(X, \pi^{<X}). \tag{3}$$

We mention two simple properties of the Shap explanations here; for more discussion see Datta, Sen, and Zick (2016) and Lundberg et al. (2020). First, for the linear combination of functions $G(.) = \sum_k \lambda_k F_k(.)$, we have that

$$\textsc{Shap}_G(X) \ = \sum_k \lambda_k \textsc{Shap}_{F_k}(X). \tag{4}$$

Second, the sum of the Shap explanation of all features is related to the expected value of function $F$:

$$\sum_i \textsc{Shap}_F(X_i) \ = F(\mathbf{x}) - \mathbf{E}[F]. \tag{5}$$

### 2.2 Computational Problems

This paper studies the complexity of computing $\textsc{Shap}(X)$; a task we formally define next. We write $\mathtt{F}$ for a class of functions. We also write $\mathtt{PR}_n$ for a class of data distributions over $n$ features, and let $\mathtt{PR} = \bigcup_n \mathtt{PR}_n$. We assume that all parameters are rationals. Because Shap explanations are for an arbitrary fixed instance $\mathbf{x}$, we will simplify the notation throughout this paper by assuming it to be the instance $\mathbf{e} = (1, 1, \ldots, 1)$, and that each domain contains the value 1, which is without loss of generality.

**Definition 2** (Shap Computational Problems)**.** For each function class $\mathtt{F}$ and distribution class $\mathtt{PR}$, consider the following computational problems.

- The *functional* Shap *problem* $\mathtt{F\text{-}SHAP}(\mathtt{F}, \mathtt{PR})$: given a data distribution $\Pr \in \mathtt{PR}$ and a function $F \in \mathtt{F}$, compute $\textsc{Shap}(X_1), \ldots, \textsc{Shap}(X_n)$.

- The *decision* Shap *problem* $\mathtt{D\text{-}SHAP}(\mathtt{F}, \mathtt{PR})$: given a data distribution $\Pr \in \mathtt{PR}$, a function $F \in \mathtt{F}$, a feature $X \in \mathbf{X}$, and a threshold $t \in \mathbb{R}$, decide if $\textsc{Shap}(X) > t$.

We may also fix function $F$, and consider the problems $\mathtt{F\text{-}SHAP}(\{F\}, \mathtt{PR})$ or $\mathtt{D\text{-}SHAP}(\{F\}, \mathtt{PR})$, where the only input is data distribution $\Pr \in \mathtt{PR}$.

To establish the complexities of these problems, we use standard notions of reductions. A *polynomial time reduction* from a problem $\mathtt{A}$ to a problem $\mathtt{B}$, denoted by $\mathtt{A} \leq^P \mathtt{B}$, and also called a *Cook reduction*, is a polynomial-time algorithm for the problem $\mathtt{A}$ with access to an oracle for the problem $\mathtt{B}$. We write $\mathtt{A} \equiv^P \mathtt{B}$ when both $\mathtt{A} \leq^P \mathtt{B}$ and $\mathtt{B} \leq^P \mathtt{A}$.

In the remainder of this paper, we study the computational complexity of these problems for natural hypothesis classes $\mathtt{F}$ that are popular in machine learning, and common classes of data distributions $\mathtt{PR}$, including those most often used to compute Shap explanations.

**Algorithm 1** Algorithm to compute value function v from Lundberg, Erion, and Lee (2018)

**procedure** EXPVALUE(**x**, S, root = {left,right,feature,threshold,value})

  **procedure** G($n$)

    **if** $n$ is a leaf **then**

      **return** $n$.value

    **else**

      **if** $n$.feature $\subseteq S$ **then**

        **return if** $\mathbf{x}[n.\text{feature}] \leq n.\text{threshold}$ **then** $G(n.\text{left})$ **else** $G(n.\text{right})$

      **else**

        **return** $(G(n.\text{left}) \cdot \text{count}(n.\text{left}) + G(n.\text{right}) \cdot \text{count}(n.\text{right}))/\text{count}(n)$

  **return** $G(\text{root})$

### 2.3 Discussion on the TreeSHAP Algorithm

Lundberg, Erion, and Lee (2018) propose TreeSHAP, a variant of Shap explanations for tree-based machine learning models such as decision trees, random forests and gradient boosted trees. The authors claim that, for the case when both the model and probability distribution[1] are defined by a tree-based model, the algorithm can compute the exact Shap explanations in polynomial time. However, it has been pointed out in Github discussions[2] that the TreeSHAP algorithm does not compute the Shap explanation as defined in Section 2.1. In this section, we provide a concrete example of this shortcoming.

The main shortcoming of the TreeSHAP algorithm is captured by Algorithm 1. The authors claim that Algorithm 1 computes the conditional expectation $\mathbf{E}[F \mid \mathbf{x}_S]$, for a given set of features $S$ and tree-based model $F$. We first describe the algorithm and then show by example that this algorithm does not accurately compute the conditional expectation.

Algorithm 1 takes as input a feature vector $\mathbf{x}$, a set of features $S$, and the root node for a binary tree that represents the tree-based model. Each internal node $n$ defines a binary decision between a $feature$ and a constant $threshold$, and has $left$ and $right$ pointers to the left and right children of $n$. Leaf nodes define a $value$ which represents the prediction of the model. We use $count(n)$ as an auxiliary function that returns the count of the number of data samples that fall in the sub-tree that is rooted by node $n$.

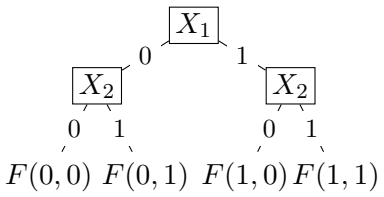
The algorithm proceeds recursively in a top-down traversal of the tree. For inner nodes, the algorithm follows the decision path for $\mathbf{x}$ if the split feature is in $S$, and takes the weighted average of both branches if the split feature is not in $S$. For leaf nodes, it returns the value of the node. The following simple example shows that the value returned by Algorithm 1 does not represent the conditional expectation $E[F \mid \mathbf{x}_S]$.

**Example 1.** We consider the following dataset and decision tree model. The dataset has two binary variables $X_1$ and $X_2$, and each instance $(x_1, x_2)$ is weighted by the occur-

1. Lundberg, Erion, and Lee (2018) compute the probability distribution using tree-based models recursively as the weighted average of the frequencies for the output variable are returned by the left and right subtrees. The weight is given by the number of data samples that fall into the respective subtree.
2. See, for instance, `https://github.com/christophM/interpretable-ml-book/issues/142` (accessed on August 20, 2021).

rence count (i.e., the instance (0,0) occurs twice in the dataset). We want to compute $E[F(X_1, X_2)|X_2 = 0]$, where $F(X_1, X_2)$ is the outcome of the decision tree.

| $X_1$ | $X_2$ | # |
|---|---|---|
| 0 | 0 | 2 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 2 |




The correct value is:

$$E[F(X_1, X_2) \mid X_2 = 0] = 2/3 \cdot F(0,0) + 1/3 \cdot F(1,0)$$

This is because there are three items with $X_2 = 0$, and their probabilities are 2/3 and 1/3.

Algorithm 1, however, returns:

$$G(1) = 1/2 \cdot F(0,0) + 1/2 \cdot F(1,0),$$

and thus does not compute $E[F(X_1, X_2) \mid X_2 = 0]$.

The algorithm does not accurately compute the conditional expectation $E[F \mid \mathbf{x}_S]$, because it does not normalize the expectation by the probability of the condition. Without this normalization, Lundberg, Erion, and Lee (2018) are able to compute all expectations required by Shap in one pass over the tree-based model. When we consider the normalization, however, the expectations depend on the values of $\mathbf{x}$ and it is non-obvious if Shap can still be computed efficiently. We address this question in the next sections.

## 3. Shap over Fully-Factorized Distributions

We start our study of the complexity of Shap by considering the simplest probability distribution: a fully-factorized distribution, where all features are independent.

There are both practical and computational reasons why it makes sense to assume a fully-factorized data distribution when computing Shap explanations. First, functions $F$ are often the product of a supervised learning algorithm that does not have access to a generative model of the data – it is purely discriminative. Hence, it is convenient to make the practical assumption that the data distribution is fully factorized, and therefore easy to estimate. Second, fully-factorized distributions are highly tractable; for example they make it easy to compute expectations of linear regression functions (Khosravi et al., 2019b) and other hard inference tasks (Vergari et al., 2020).

Lundberg and Lee (2017) indeed observe that computing the Shap-explanation on an arbitrary data distribution is challenging and consider using fully-factorized distributions (Sec. 4, Eq. 11). Other prior work on computing explanations also use fully-factorized distributions of features, e.g., Datta, Sen, and Zick (2016); Štrumbelj and Kononenko (2014). As we will show, the Shap explanation can be computed efficiently for several popular classifiers when the distribution is fully factorized. Yet, such simple data distributions are not a guarantee for tractability: computing Shap scores will be intractable for some other common classifiers.

### 3.1 Equivalence to Computing Expectations

Before studying various function classes, we prove a key result that connects the complexity of Shap explanations to the complexity of computing expectations.

Let $\texttt{IND}_n$ be the class of fully-factorized probability distributions over $n$ discrete and independent random variables $X_1, \ldots, X_n$. That is, for every instance $(x_1, \ldots, x_n) \in \mathcal{X}$, we have that $\Pr(X_1 = x_1, \ldots, X_n = x_n) = \prod_i \Pr(X_i = x_i)$. Let $\texttt{IND} \stackrel{\text{def}}{=} \bigcup_{n \geq 0} \texttt{IND}_n$. We show that for every function class $\texttt{F}$, the complexity of $\texttt{F-SHAP}(\texttt{F}, \texttt{IND})$ is the same as that of the *fully-factorized expectation problem*.

**Definition 3** (Fully-Factorized Expectation Problem)**.** Let $\texttt{F}$ be a class of real-valued functions with discrete inputs. The *fully-factorized expectation problem* for $\texttt{F}$, denoted $\texttt{E}(\texttt{F})$, is the following: given a function $F \in \texttt{F}$ and a probability distribution $\Pr \in \texttt{IND}$, compute $\mathbf{E}_{\Pr}(F)$.

We know from Equation (5) that $\mathbf{E}[F] = F(\mathbf{x}) - \sum_{i=1,n} \textsc{Shap}_F(X_i)$, and thus for any function $F$ over $n$ features $\texttt{E}(\{F\}) \leq^P \texttt{F-SHAP}(\{F\}, \texttt{IND}_n)$. In this section we prove that the converse holds too:

**Theorem 2.** *For any function $F : \mathcal{X} \to \mathbb{R}$, we have that $\texttt{F-SHAP}(\{F\}, \texttt{IND}_n) \equiv^P \texttt{E}(\{F\})$.*

In other words, for *any* function $F$, the complexity of computing the Shap scores is the same as the complexity of computing the expected value $\mathbf{E}[F]$ under a fully-factorized data distribution. One direction of the proof is immediate: $\texttt{E}(\{F\}) \leq^P \texttt{F-SHAP}(\{F\}, \texttt{IND}_n)$ because, if we are given an oracle to compute $\textsc{Shap}_F(X_i)$ for every feature $X_i$, then we can obtain $\mathbf{E}[F]$ from Equation (5) (recall that we assumed that $F(\mathbf{x})$ is computable in polynomial time). The hard part of the proof is the opposite direction: we will show in Sec. 3.2 how to compute $\textsc{Shap}_F(X_i)$ given an oracle for computing $\mathbf{E}[F]$. Theorem 2 immediately extends to classes of functions $\texttt{F}$, and to any number of variables, and therefore implies that $\texttt{F-SHAP}(\texttt{F}, \texttt{IND}) \equiv^P \texttt{E}(\texttt{F})$.

Sections 3.3 and 3.4 will discuss the consequences of this result, by delineating which function classes support tractable Shap explanations, and which do not. The next section is devoted to proving our main technical result.

### 3.2 Proof of Theorem 2

We start with the special case when all features $X$ are binary: $\text{dom}(X) = \{0, 1\}$. We denote by $\texttt{INDB}_n$ the class of fully-factorized distributions over binary domains.

**Theorem 3.** *For any function $F : \{0,1\}^n \to \mathbb{R}$, we have that $\texttt{F-SHAP}(\{F\}, \texttt{INDB}_n) \equiv^P \texttt{E}(\{F\})$.*

*Proof.* We prove only $\texttt{F-SHAP}(F, \texttt{INDB}_n) \leq^P \texttt{E}(\{F\})$; the opposite direction follows immediately from Equation (5). We will assume w.l.o.g. that $F$ has $n + 1$ binary features $\mathbf{X}' = \{X_0\} \cup \mathbf{X}$ and show how to compute $\textsc{Shap}_F(X_0)$ using repeated calls to an oracle for computing $\mathbf{E}[F]$, i.e., the expectation of the same function $F$, but over fully-factorized distributions with different probabilities. The probability distribution $\Pr$ is given to us by $n + 1$ rational numbers, $p_i \stackrel{\text{def}}{=} \Pr(X_i = 1)$, $i = 0, n$; obviously, $\Pr(X_i = 0) = 1 - p_i$. Recall

that the instance whose outcome we want to explain is $\mathbf{e} = (1, \ldots, 1)$. Recall that for any set $S \subseteq [n]$ we write $\mathbf{e}_S$ for the event $\bigwedge_{i \in S}(X_i = 1)$. Then, we have that

$$\textsc{Shap}(X_0) = \sum_{k=0,n} \frac{k!(n-k)!}{(n+1)!} D_k, \quad \text{where} \tag{6}$$
$$D_k \stackrel{\text{def}}{=} \sum_{S \subseteq [n]: |S|=k} \left( \mathbf{E}\left[F | \mathbf{e}_{S \cup \{0\}}\right] - \mathbf{E}[F|\mathbf{e}_S] \right).$$

This follows from Equation (3) by the following argument. Obviously, quantity $n!$ becomes $(n+1)!$. If $|S| = k$, then there are exactly $k!(n-k)!$ permutations $\pi$ that place all elements in $S$ before $X_0$, and all elements not in $S$ after $X_0$. Finally, the expression $\mathbf{E}\left[F|\mathbf{e}_{S\cup\{0\}}\right] - \mathbf{E}[F|\mathbf{e}_S]$ is precisely the contribution $\mathrm{c}(X_0, \mathbf{X}_S)$ of the feature $X_0$ in the context $S$, as defined by Equation (2).

Let $F_0 \stackrel{\text{def}}{=} F[X_0 := 0]$ and $F_1 \stackrel{\text{def}}{=} F[X_0 := 1]$ (both are functions in $n$ binary features, $\mathbf{X} = \{X_1, \ldots, X_n\}$). Then the contribution of feature 0 becomes:

$$\mathbf{E}\left[F\middle|\mathbf{e}_{S\cup\{0\}}\right] = \mathbf{E}[F_1|\mathbf{e}_S]$$
$$\mathbf{E}[F|\mathbf{e}_S] = \mathbf{E}[F_0|\mathbf{e}_S] \cdot (1-p_0) + \mathbf{E}[F_1|\mathbf{e}_S] \cdot p_0$$

and therefore $D_k$ is given by:

$$D_k = (1-p_0) \sum_{S \subseteq [n]: |S|=k} (\mathbf{E}[F_1|\mathbf{e}_S] - \mathbf{E}[F_0|\mathbf{e}_S])$$

Recall that we defined $\mathrm{v}_{G,\mathbf{e},\Pr}(\mathbf{X}_S) = \mathbf{E}[G|\mathbf{e}_S]$ in Equation (1). Abusing notation, we write $v_{G,k}$ for the sum of these quantities over all sets $S$ of cardinality $k$:

$$v_{G,k} \stackrel{\text{def}}{=} \sum_{S \subseteq [n], |S|=k} \mathbf{E}[G|\mathbf{e}_S]. \tag{7}$$

In summary so far, we have expressed $\textsc{Shap}(X_0)$ in Equation (6), where $D_k$ is given by:

$$D_k = (1-p_0)(v_{F_1,k} - v_{F_0,k})$$

To compute $\textsc{Shap}(X_0)$, it suffices to compute $v_{G,k}$ where $k = 0, n$ and $G$ is one of $F_0$ or $F_1$.

We will prove the following claim.

**Claim 1.** Let $G$ be a function over $n$ binary variables. Then the $n+1$ quantities $v_{G,0}$ until $v_{G,n}$ can be computed in polynomial time, using $n+1$ calls to an oracle for $\mathtt{E}(\{G\})$.

Note that an oracle for $\mathtt{E}(\{F\})$ is also an oracle for both $\mathtt{E}(\{F_0\})$ and $\mathtt{E}(\{F_1\})$, by simply setting $\Pr(X_0 = 1) = 0$ or $\Pr(X_0 = 1) = 1$ respectively. Therefore, Claim 1 proves Theorem 3, by applying it once to $F_0$ and once to $F_1$ in order to derive all the quantities $v_{F_0,k}$ and $v_{F_1,k}$, thereby computing $D_k$, and finally computing $\textsc{Shap}_F(X_0)$ using Equation (6). It remains to prove Claim 1.

Fix a function $G$ over $n$ binary variables and let $v_k = v_{G,k}$. Let $p_j = \Pr(X_j = 1)$, for $j = 1, n$, define the distribution over which we need to compute $v_0, \ldots, v_n$. We will prove the following additional claim.

**Claim 2.** Given any real number $z > 0$, consider the distribution $\Pr_z(X_j) = p'_j \stackrel{\text{def}}{=} \frac{p_j + z}{1+z}$, for $j = 1, n$. Let $\mathbf{E}_z[G]$ denote $\mathbf{E}[G]$ under distribution $\Pr_z$. We then have that

$$\sum_{k=0,n} z^k \cdot v_k = (1+z)^n \cdot \mathbf{E}_z[G]. \tag{8}$$

Assuming Claim 2 holds, we prove Claim 1. The LHS of Equation (8) is a polynomial in $z$ with coefficients $v_k$, while the RHS gives us an oracle for computing the value of this polynomial for any input $z > 0$. Using this oracle, we need to compute the coefficients $v_k$ of the polynomial. This can be done in many ways, we briefly describe here one possibility. Choose any $n+1$ distinct values for $z$, use the oracle to compute the quantities $\mathbf{E}_{z_0}[G], \ldots, \mathbf{E}_{z_n}[G]$, and form a system of $n+1$ linear equations (8) with unknowns $v_0, \ldots, v_n$. Next, observe that its matrix is a non-singular Vandermonde matrix, hence the system has a unique solution which can be computed in polynomial time. It remains to prove Claim 2.

Because of independence, the probability of instance $\mathbf{x} \in \{0,1\}^n$ is $\Pr(\mathbf{x}) = \prod_{i:\mathbf{x}_i=1} p_i \cdot \prod_{i:\mathbf{x}_i=0}(1-p_i)$, where $\mathbf{x}_i$ looks up the value of feature $X_i$ in instance $\mathbf{x}$. Similarly, $\Pr_z(\mathbf{x}) = \prod_{i:\mathbf{x}_i=1} p'_i \cdot \prod_{i:\mathbf{x}_i=0}(1-p'_i)$. Using direct calculations we derive:

$$\Pr(\mathbf{x}) \prod_{i:\mathbf{x}_i=1} \left(1 + \frac{z}{p_i}\right) = (1+z)^n \cdot \Pr_z(\mathbf{x}) \tag{9}$$

Separately we also derive the following identity, using the fact that $\Pr(\mathbf{e}_S) = \prod_{i \in S} p_i$ by independence:

$$\mathbf{E}[G|\mathbf{e}_S] = \frac{1}{\prod_{i \in S} p_i} \sum_{\mathbf{x}:\mathbf{x}_S = \mathbf{e}_S} G(\mathbf{x}) \cdot \Pr(\mathbf{x}) \tag{10}$$

We are now in a position to prove Claim 2:

$$\begin{aligned}
\sum_{k=0,n} z^k \cdot v_k &= \sum_{k=0,n} z^k \sum_{S \subseteq [n]:|S|=k} \mathbf{E}[G|\mathbf{e}_S] \\
&= \sum_{S \subseteq [n]} z^{|S|} \cdot \mathbf{E}[G|\mathbf{e}_S] \\
&= \sum_{S \subseteq [n]} \frac{z^{|S|}}{\prod_{i \in S} p_i} \sum_{\mathbf{x}:\mathbf{x}_S = \mathbf{e}_S} G(\mathbf{x}) \cdot \Pr(\mathbf{x})
\end{aligned}$$

The last line follows from Equation (10). Next, we simply exchange the summations $\sum_S$ and $\sum_{\mathbf{x}}$, after which we apply the identity $\sum_{S \subseteq A} \prod_{i \in S} u_i = \prod_{i \in A}(1+u_i)$.

*(continuing)*

$$\begin{aligned}
&= \sum_{\mathbf{x} \in \{0,1\}^n} G(\mathbf{x}) \cdot \Pr(\mathbf{x}) \sum_{S:\mathbf{x}_S = \mathbf{e}_S} \frac{z^{|S|}}{\prod_{i \in S} p_i} \\
&= \sum_{\mathbf{x} \in \{0,1\}^n} G(\mathbf{x}) \cdot \Pr(\mathbf{x}) \prod_{i:\mathbf{x}_i=1} \left(1 + \frac{z}{p_i}\right) \\
&= (1+z)^n \sum_{\mathbf{x} \in \{0,1\}^n} G(\mathbf{x}) \cdot \Pr_z(\mathbf{x}) = (1+z)^n \cdot \mathbf{E}_z[G].
\end{aligned}$$

The final line uses Equation (9). This completes the proof of Claim 2 as well as Theorem 3. □

Next, we generalize this result from binary features to arbitrary discrete features. Fix a function with $n$ inputs, $F : \mathcal{X}(\stackrel{\text{def}}{=} \prod_i \text{dom}(X_i)) \to \mathbb{R}$, where each domain is an arbitrary finite set, $\text{dom}(X_i) = \{1, 2, \ldots, m_i\}$; we assume w.l.o.g. that $m_i > 1$. A fully factorized probability space $\Pr \in \texttt{IND}_n$ is defined by numbers $p_{ij} \in [0, 1]$, $i = 1, n$, $j = 1, m_i$, such that, for all $i$, $\sum_j p_{ij} = 1$. Given $F$ and Pr over the domain $\prod_i \text{dom}(X_i)$, we define their *projections*, $F_\pi, \Pr_\pi$ over the binary domain $\{0, 1\}^n$ as follows. For any instance $\mathbf{x} \in \{0, 1\}^n$, let $T(\mathbf{x})$ denote the event asserting that $X_j = 1$ iff $\mathbf{x}_j = 1$. Formally,

$$T(\mathbf{x}) \stackrel{\text{def}}{=} \bigwedge_{j:\mathbf{x}_j=1} (X_j = 1) \wedge \bigwedge_{j:\mathbf{x}_j=0} (X_j \neq 1). \tag{11}$$

Then, the projections are defined as follows: $\forall \mathbf{x} \in \{0, 1\}^n$,

$$\Pr_\pi(\mathbf{x}) \stackrel{\text{def}}{=} \Pr(T(\mathbf{x})) \qquad\qquad F_\pi(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{E}[F \mid T(\mathbf{x})] \tag{12}$$

Notice that $F_\pi$ depends both on $F$ and on the probability distribution Pr. Intuitively, the projections only distinguishes between $X_j = 1$ and $X_j \neq 1$, for example:

$$\begin{aligned} F_\pi(1, 0, 0) =& \mathbf{E}[F | (X_1 = 1, X_2 \neq 1, X_3 \neq 1)] \\ \Pr_\pi(1, 0, 0) =& \Pr(X_1 = 1, X_2 \neq 1, X_3 \neq 1) \end{aligned}$$

**Proposition 4.** *Let $F : \mathcal{X} \to \mathbb{R}$ be a function with $n$ input features, and $\Pr \in \texttt{IND}_n$ a fully factorized distribution over $\mathcal{X}$. Then the following hold: (1) For any feature $X_j$, $\text{Shap}_{F,\Pr}(X_j) = \text{Shap}_{F_\pi,\Pr_\pi}(X_j)$; in particular, $\texttt{E}(\{F\}) \leq^P \texttt{E}(\{F_\pi\})$. And (2) $\texttt{E}(\{F_\pi\}) \leq^P \texttt{E}(\{F\})$.*

Item (1) states that the Shap-score of $F$ computed over the probability space Pr is the same as that of its projection $F_\pi$ (which depends on Pr) over the projected probability space $\Pr_\pi$. Item (2) says that, for any probability space over $\{0, 1\}^n$ (not necessarily $\Pr_\pi$), we can compute $\mathbf{E}[F_\pi]$ in polynomial time given access to an oracle for computing $\mathbf{E}[F]$. Before we prove the proposition, we show how it to use it to complete the proof of Theorem 2, by showing that $\texttt{F-SHAP}(\{F\}, \texttt{IND}_n) \leq^P \texttt{E}(\{F\})$. We are given a function $F$ over domain $\mathcal{X}$, and have access to an oracle for computing $\mathbf{E}[F]$ over any fully factorized probability distribution on $\mathcal{X}$. Given probability space $\Pr \in \texttt{IND}_n$, our goal is to compute $\text{Shap}_{F,\Pr}(X_j)$. By item (1) of Proposition 4 it suffices to compute $\text{Shap}_{F_\pi,\Pr_\pi}(X_j)$. By Theorem 3, we can compute the latter given access to an oracle for computing $\mathbf{E}[F_\pi]$, over an arbitrary fully factorized probability distribution on $\{0, 1\}^n$. Finally, by item (2) of the proposition, we can compute $\mathbf{E}[F_\pi]$ given the oracle for computing $\mathbf{E}[F]$.

In the rest of the section we prove Proposition 4.

*Proof.* We start with item (1). Recall that $\text{dom}(X_i) = \{1, 2, 3, \ldots, m_i\}$. We denote by $p_{i1}, p_{i2}, \ldots, p_{im_i}$ their probabilities, thus $\sum_{j=1,m_i} p_{ij} = 1$. By definition, the projected distribution is: $\Pr_\pi(X_i = 1) \stackrel{\text{def}}{=} p_{i1}$, and $\Pr_\pi(X_i = 0) = 1 - p_{i1}$. We denote by $\mathbf{E}_\pi$ be the corresponding expectation. Our goal is to prove $\text{Shap}_{F,\Pr}(X_j) = \text{Shap}_{F_\pi,\Pr_\pi}(X_j)$.

Let $\mathbf{e}_S$ again denote the event $\bigwedge_{i\in S}(X_i = 1)$. Note that, by construction, for any set $S$, $\Pr(\mathbf{e}_S) = \Pr_\pi(\mathbf{e}_S)$. Given the definition of $T(\mathbf{x})$ in Equation (11), for any instance $\mathbf{x} \in \{0,1\}^n$,

$$\Pr(T(\mathbf{x})) = \prod_{i:\mathbf{x}_i=1} p_{i1} \cdot \prod_{i:\mathbf{x}_i=0} (p_{i2} + p_{i3} + \cdots) = \Pr_\pi(\mathbf{x}).$$

There are $2^n$ disjoint events $T(\mathbf{x})$, which partition the space $\mathcal{X}$. Therefore, for every set $S$:

$$\begin{aligned}\mathbf{E}[F \wedge \mathbf{e}_S] &= \sum_{\mathbf{x}:\forall i\in S,\mathbf{x}_i=1} \mathbf{E}[F|T(\mathbf{x})] \Pr(T(\mathbf{x})) \\ &= \sum_{\mathbf{x}:\forall i\in S,\mathbf{x}_i=1} F_\pi(\mathbf{x}) \Pr_\pi(\mathbf{x}) \\ &= \mathbf{E}_\pi[F_\pi \wedge \mathbf{e}_S]\end{aligned}$$

This implies that $\mathbf{E}[F|\mathbf{e}_S] = \mathbf{E}_\pi[F_\pi|\mathbf{e}_S]$ for any set $S$, and $\textsc{Shap}_{F,\Pr}(X_j) = \textsc{Shap}_{F_\pi,\Pr_\pi}(X_j)$ for all $j$ follows from Equation (6).

We now prove item (2). As before, we are given $F, \Pr$ over the domain $\mathcal{X}$, which in turn define $\Pr_\pi$ and $F_\pi$ in Equations (12) and (12). In addition, we are also given an oracle for computing $\mathbf{E}'[F]$ over an arbitrary distribution $\Pr'$. We need to compute $\mathbf{E}[F_\pi]$, over some arbitrary distribution on $\{0,1\}^n$, denote it by $\Pr'_\pi$ (not to be confused with $\Pr_\pi$). To compute $\mathbf{E}[F_\pi]$, we will construct a distribution $\Pr'$ over $\mathcal{X}$, use the oracle to compute the expectation $\mathbf{E}'[F]$ over $\Pr'$, then show to use this answer to compute $\mathbf{E}[F_\pi]$.

To define $\Pr'$, we need some notations. Recall that $p_{ij}$ denotes the probability $\Pr(X_i = j)$. Denote $q_i \stackrel{\text{def}}{=} \Pr'_\pi(X_i{=}1)$, thus $\Pr'_\pi(X_i{=}0) = 1 - q_i$, and further denote:

$$w_i \stackrel{\text{def}}{=} \frac{1-q_i}{q_i} \qquad Z \stackrel{\text{def}}{=} \prod_{i=1,n} q_i \qquad W \stackrel{\text{def}}{=} \prod_{i=1,n} \left(1 + \sum_{j=2,m_i} \frac{p_{ij} w_i}{1 - p_{i1}}\right)$$

With these notations, we define $\Pr'(X_i = j) \stackrel{\text{def}}{=} p'_{ij}$, where:

$$\begin{aligned} p'_{i1} &\stackrel{\text{def}}{=} \frac{1}{W} && i{=}1, n \\ p'_{ij} &\stackrel{\text{def}}{=} \frac{p_{ij} w_i}{W(1-p_{i1})} && i{=}1, n; j = 2, m_i \end{aligned}$$

One can check that the numbers $p'_{ij}$ indeed define a probability space on $\mathcal{X}$, in other words $p'_{ij} \in [0,1]$ and, for all $i = 1, n$: $\sum_{j=1,m_j} p'_{ij} = 1$. We denote by $\Pr'$ the probability space that they define, and denote by $\mathbf{E}'[F]$ the expectation of $F$ in this space. We claim:

**Claim 3.** $\mathbf{E}[F_\pi] = Z \cdot W \cdot \mathbf{E}'[F]$

The claim immediately proves item (2) of Proposition 4: to compute $\mathbf{E}[F_\pi]$ over the probability distribution $\Pr'_\pi$, we compute the distribution $\Pr'$ and invoke the oracle to compute $\mathbf{E}'[F]$, then multiply with the quantities $Z$ and $W$, both of which are computable in polynomial time. It remains to prove the claim.

We start with some observations and notations. We denote an outcome in $\mathcal{X} = \prod_{i=1,n} \mathrm{dom}(X_i)$ as $\tau$, and view it as either a vector $(\tau_1, \tau_2, \ldots)$ or a function with domain

$\{1, 2, \ldots, n\}$, where $\tau(i) \in \mathrm{dom}(X_i)$. Then, the expectation of some function $G : \mathcal{X} \to \mathbb{R}$ can be written as $\mathbf{E}[G] = \sum_{\tau \in \mathcal{X}} G(\tau)\Pr(\tau) = \sum_{\tau \in \mathcal{X}} G(\tau) \prod_i p_{i\tau_i}$. Furthermore, $\tau^{-1}(1)$ denotes the set $\{i \mid \tau(i) = 1\}$. We these notations, we compute the projection $F_\pi$, defined Equation (12), explicitly in terms of the probabilities $p_{ij}$. For $\mathbf{x} \in \{0,1\}^n$:

$$\begin{aligned} F_\pi[\mathbf{x}] = \ & \mathbf{E}[F|T(\mathbf{x})] = \frac{\mathbf{E}[F \cdot T(\mathbf{x})]}{\Pr(T(\mathbf{x}))} = \frac{\sum_{\tau \in \mathcal{X}: \mathbf{x}^{-1}(1) = \tau^{-1}(1)} F(\tau) \cdot \prod_i p_{i\tau_i}}{\prod_{i:\mathbf{x}_i=1} p_{i1} \cdot \prod_{i:\mathbf{x}_i \neq 1}(1 - p_{i1})} \\ = & \sum_{\tau \in \mathcal{X}: \mathbf{x}^{-1}(1) = \tau^{-1}(1)} F(\tau) \cdot \prod_{i:\tau_i \neq 1} \frac{p_{i\tau_i}}{1 - p_{i1}}. \end{aligned}$$

We now prove the claim by applying directly the definition of $\mathbf{E}[F_\pi]$:

$$\begin{aligned} \mathbf{E}[F_\pi] = & \sum_{\theta \in \{0,1\}^n} F_\pi(\theta) \prod_{i:\theta_i=1} q_i \prod_{i:\theta_i=0} (1 - q_i) = Z \cdot \sum_{\theta \in \{0,1\}^n} F_\pi(\theta) \prod_{i:\theta_i=0} w_i \\ = & Z \cdot \sum_{\theta \in \{0,1\}^n} \left( \sum_{\tau \in \mathcal{X}: \theta^{-1}(1) = \tau^{-1}(1)} F(\tau) \prod_{i:\tau_i \neq 1} \frac{p_{i\tau_i}}{1 - p_{i1}} \right) \prod_{i:\theta_i=0} w_i \\ = & Z \cdot \sum_{\tau \in \mathcal{X}} F(\tau) \prod_{i:\tau_i \neq 1} \frac{p_{i\tau_i} w_i}{1 - p_{i1}} = Z \cdot W \cdot \sum_{\tau \in \mathcal{X}} F(\tau) \prod_i p'_{i\tau_i} = Z \cdot W \cdot \mathbf{E}'[F] \end{aligned}$$

We explain the transition from line 2 to line 3. In the summation in line 2, the assignment $\theta \in \{0,1\}^n$ is used in two places: to restrict the range of $\tau \in \mathcal{X}$ in the second sum, and in the condition $\theta_i = 0$. The condition $\theta^{-1}(1) = \tau^{-1}(1)$ says $\theta_i = 1$ iff $\tau_i = 1$ and therefore $\theta_i = 0$ iff $\tau_i \neq 1$. Thus, we can replace the condition $\theta_i = 0$ with $\tau_i \neq 1$. Furthermore, the assignment $\tau \in \mathcal{X}$ uniquely defines $\theta$, hence $\theta$ can be dropped from the summation. In line 3 we simply used the definition of $p'_{ij}$ introduced earlier. This completes the proof of the claim, and of Proposition 4. □

### 3.3 Tractable Function Classes

Given the polynomial-time equivalence between computing Shap explanations and computing expectations under fully-factorized distributions, a natural next question is: which real-world hypothesis classes in machine learning support efficient computation of Shap scores?

**Corollary 5.** *For the following function classes* `F`*, computing* Shap *scores* `F-SHAP(F, IND)` *is in polynomial time in the size of the representations of function* $F \in$ `F` *and fully-factorized distribution* $\Pr \in$ `IND`*.*

1. *Linear regression models*
2. *Decision and regression trees*
3. *Random forests or additive tree ensembles*
4. *Factorization machines, regression circuits*

5. *Boolean functions in d-DNNF, binary decision diagrams*

6. *Bounded-treewidth Boolean functions in CNF*

These are all consequences of Theorem 2, and the fact that computing fully-factorized expectations $\texttt{E}(\texttt{F})$ for these function classes $\texttt{F}$ is in polynomial time. Concretely, we have the following observations about fully-factorized expectations:

1. Expectations of linear regression functions are efficiently computed by mean imputation (Khosravi et al., 2019b). The tractability of Shap on linear regression models is well known. In fact, Štrumbelj and Kononenko (2014) provide a closed-form formula for this case.

2. Paths from root to leaf in a decision or regression tree are mutually exclusive. Their expected value is therefore the sum of expected values of each path, which are tractable to compute within $\texttt{IND}$; see Khosravi et al. (2020).

3. Additive mixtures of trees, as obtained through bagging or boosting, are tractable, by the linearity of expectation.

4. Factorization machines extend linear regression models with feature interaction terms and factorize the parameters of the higher-order terms (Rendle, 2010). Their expectations remain easy to compute. Regression circuits are a graph-based generalization of linear regression. Khosravi et al. (2019a) provide an algorithm to efficiently take their expectation w.r.t. a probabilistic circuit distribution, which is trivial to construct for the fully-factorized case.

The remaining tractable cases are Boolean functions. Computing fully-factorized expectations of Boolean functions is widely known as the *weighted model counting* task (WMC) (Sang, Beame, and Kautz, 2005; Chavira and Darwiche, 2008). WMC has been extensively studied both in the theory and the AI communities, and the precise complexity of $\texttt{E}(\texttt{F})$ is known for many families of Boolean functions $\texttt{F}$. These results immediately carry over to the $\texttt{F-SHAP}(\texttt{F}, \texttt{IND})$ problem through Theorem 2:

5. Expectations can be computed in time linear in the size of various circuit representations, called d-DNNF, which includes binary decision diagrams (OBDD, FBDD) and SDDs (Bryant, 1986; Darwiche and Marquis, 2002).[3]

6. Bounded-treewidth CNFs are efficiently compiled into OBDD circuits (Ferrara, Pan, and Vardi, 2005), and thus enjoy tractable expectations.

To conclude this section, the reader may wonder about the algorithmic complexity of solving $\texttt{F-SHAP}(\texttt{F}, \texttt{IND})$ with an oracle for $\texttt{E}(\texttt{F})$ under the reduction in Section 3.2. Briefly, we require a linear number of calls to the oracle, as well as time in $O(n^3)$ for solving a system of linear equations. Hence, for those classes, such as d-DNNF circuits, where expectations are linear in the size of the (circuit) representation of $F$, computing $\texttt{F-SHAP}(\texttt{F}, \texttt{IND})$ is also linear in the representation size and polynomial in $n$.

3. In contemporaneous work, Arenas et al. (2020) also show that the Shap explanation is tractable for d-DNNFs, but for the more restricted class of uniform data distributions.

### 3.4 Intractable Function Classes

The polynomial-time equivalence of Theorem 2 also implies that computing Shap scores must be intractable whenever computing fully-factorized expectations is intractable. This section reviews some of those function classes $\mathtt{F}$, including some for which the computational hardness of $\mathtt{E}(\mathtt{F})$ is well known. We begin, however, with a more surprising result.

Logistic regression is one of the simplest and most widely used machine learning models, yet it is conspicuously absent from Corollary 5. We prove that computing the expectation of a logistic regression model is #P-hard, even under a uniform data distribution, which is of independent interest.

A *logistic regression* model is a parameterized function $F(\boldsymbol{x}) \stackrel{\text{def}}{=} \sigma(\boldsymbol{w} \cdot \boldsymbol{x})$, where $\boldsymbol{w} = (w_0, w_1, \ldots, w_n)$ is a vector of weights, $\sigma(z) = 1/(1+e^{-z})$ is the logistic function, $\boldsymbol{x} \stackrel{\text{def}}{=} (1, x_1, x_2, \ldots, x_n)$, and $\boldsymbol{w} \cdot \boldsymbol{x} \stackrel{\text{def}}{=} \sum_{i=0,n} w_i x_i$ is the dot product. Note that we define the logistic regression function to output probabilities, not data labels. Let $\mathtt{LOGIT}_n$ denote the class of logistic regression functions with $n$ variables, and $\mathtt{LOGIT} = \bigcup_n \mathtt{LOGIT}_n$. We prove the following:

**Theorem 6.** *Computing the expectation of a logistic regression model w.r.t. a uniform data distribution is #P-hard.*

*Proof.* The proof is by reduction from counting solutions to the number partitioning problem. The *number partitioning* problem, NUMPAR, is the following: given $n$ natural numbers $k_1, \ldots, k_n$, decide whether there exists a subset $S \subseteq [n]$ that partitions the numbers into two sets with equal sums: $\sum_{i \in S} k_i = \sum_{i \notin S} k_i$. NUMPAR is known to be NP-complete. The corresponding counting problem, in notation #NUMPAR, asks for the number of sets $S$ such that $\sum_{i \in S} k_i = \sum_{i \notin S} k_i$, and is #P-hard.

We show that we can solve the #NUMPAR problem using an oracle for $\mathbf{E}_{\mathbf{U}}[F]$, where $F$ is a logistic regression function and $\mathbf{U}$ is the uniform probability distribution. This implies that computing the expectation of a logistic regression function is #P-hard.

Fix an instance of NUMPAR, $k_1, \ldots, k_n$, and assume w.l.o.g. that the sum of the numbers $k_i$ is even, $\sum_i k_i = 2c$ for some natural number $c$. Let

$$P \stackrel{\text{def}}{=} \{S \mid S \subseteq [n], \sum_{i \in S} k_i = c\} \tag{13}$$

For each set $S \subseteq [n]$, denote by $\bar{S}$ its complement. Obviously, $S \in P$ iff $\bar{S} \in P$, therefore $|P|$ is an even number.

We next describe an algorithm that computes $|P|$ using an oracle for $\mathbf{E}_{\mathbf{U}}[F]$, where $F$ is a logistic regression function and $\mathbf{U}$ is the uniform probability distribution. Let $m$ be a natural number large enough, to be chosen later, and define the following weights:

$$w_0 \stackrel{\text{def}}{=} -\frac{m}{2} - mc \qquad\qquad w_i \stackrel{\text{def}}{=} mk_i \quad \forall i = 1, n$$

Let $\boldsymbol{w} = (w_1, \ldots, w_n)$, then $F(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \sigma(w_0 + \boldsymbol{w} \cdot \boldsymbol{x})$ is the logistic regression function defined by the weights $w_0, \ldots, w_n$.

**Claim 4.** Let $\varepsilon \stackrel{\text{def}}{=} 1/2^{n+3}$. If $m$ satisfies both $2\sigma(-m/2) \leq \varepsilon$ and $1 - \sigma(m/2) \leq \varepsilon$, then:

$$|P| = \left\lceil 2^n - \frac{2^{n+1}\mathbf{E}[F]}{1-\varepsilon} \right\rceil$$

The claim immediately proves the theorem: in order to solve the #NUMPAR problem, compute $\mathbf{E}[F]$ and then use the formula above to derive $|P|$. To prove the claim, for each set $S \subseteq [n]$ denote by:

$$\text{weight}(S) \stackrel{\text{def}}{=} -\frac{m}{2} - mc + m(\sum_{i \in S} k_i)$$

Let $\mathbf{U}$ denote the uniform probability distribution over the domain $\{0,1\}^n$. Then,

$$\begin{aligned}
\mathbf{E}_{\mathbf{U}}[F] &= \frac{1}{2^n} \sum_{\mathbf{x}} \sigma(w_0 + \boldsymbol{w} \cdot \mathbf{x}) \\
&= \frac{1}{2^n} \sum_{\mathbf{x}} \sigma(-\frac{m}{2} - mc + m(\sum_{i \in [n]} k_i \mathbf{x}_i)) \\
&= \frac{1}{2^n} \sum_{\mathbf{x}} \sigma(-\frac{m}{2} - mc + m(\sum_{i:\mathbf{x}_i=1} k_i)) \\
&= \frac{1}{2^n} \sum_{S \subseteq [n]} \sigma(\text{weight}(S)) \\
&= \frac{1}{2^{n+1}} \sum_{S \subseteq [n]} (\sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})))
\end{aligned}$$

If $S$ is a solution to the number partitioning problem ($S \in P$), then:

$$\sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) = 2\sigma(-m/2)$$

Otherwise, one of $\text{weight}(S)$, $\text{weight}(\bar{S})$ is $\geq m/2$ and the other is $\leq -3m/2$ and therefore:

$$\sigma(m/2) \leq \sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) \leq 1 + \sigma(-3m/2)$$

Since $\varepsilon = 1/2^{n+3}$, and $m$ satisfies both $2\sigma(-m/2) \leq \varepsilon$ and $1 - \sigma(m/2) \leq \varepsilon$, we have:

$$\begin{aligned}
S \in P: &\qquad 0 \leq \sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) \leq \varepsilon \\
S \notin P: &\qquad 1 - \varepsilon \leq \sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) \leq 1 + \varepsilon
\end{aligned}$$

This implies:

$$\begin{aligned}
\frac{2^n - |P|}{2^{n+1}}(1-\varepsilon) \leq \mathbf{E}[F] \leq \frac{|P|}{2^{n+1}}\varepsilon + \frac{2^n - |P|}{2^{n+1}}(1+\varepsilon) \\
|P| \geq 2^n - \frac{2^{n+1}\mathbf{E}[F]}{1-\varepsilon} \\
|P| \leq 2^n(1+\varepsilon) - 2^{n+1}\mathbf{E}[F]
\end{aligned}$$

Thus, we have a lower and an upper bound for $|P|$. Since $\mathbf{E}[F] \leq 1$, the difference between the two bounds is $< 1$ and there exists at most one integer number between them, hence $|P|$ is equal to the ceiling of the lower bound (and also to the floor of the upper bound), proving the claim. □

Because the uniform distribution is contained in IND, and following Theorem 2, we immediately obtain:

**Corollary 7.** *The computational problems* E(LOGIT) *and* F-SHAP(LOGIT, IND) *are both #P-hard.*

We are now ready to list general function classes for which computing the Shap explanation is #P-hard.

**Corollary 8.** *For the following function classes* F*, computing* Shap *scores* F-SHAP(F, IND) *is #P-hard in the size of the representations of function* $F \in$ F *and fully-factorized distribution* $\Pr \in$ IND.

1. *Logistic regression models (Corollary 7)*
2. *Neural networks with sigmoid activation functions*
3. *Naive Bayes classifiers, logistic circuits*
4. *Boolean functions in CNF or DNF*

Our intractability results stem from these observations:

2. Each neuron is a logistic regression model, and therefore this class subsumes LOGIT.
3. The conditional distribution used by a naive Bayes classifier is known to be equivalent to a logistic regression model (Ng and Jordan, 2002). Logistic circuits are a graph-based classification model that subsumes logistic regression (Liang and Van den Broeck, 2019).
4. For general CNFs and DNFs, weighted model counting, and therefore E(F) is #P-hard. This is true even for very restricted classes, such as monotone 2CNF and 2DNF functions, and Horn clause logic (Wei and Selman, 2005).

## 4. Beyond Fully-Factorized Distributions

Features in real-world data distributions are not independent. In order to capture more realistic assumptions about the data when computing Shap scores, one needs a more intricate probabilistic model. In this section we prove that the complexity of computing the Shap-explanation quickly becomes intractable, even over the simplest probabilistic models, namely naive Bayes models. To make computing the Shap-explanation as easy as possible, we will assume that the function $F$ simply outputs the value of one feature. We show that even in this case, even for function classes that are tractable under fully-factorized distributions, computing Shap explanations becomes computationally hard.

Let $\texttt{NBN}_n$ denote the family of naive Bayes networks over $n{+}1$ variables $\mathbf{X} = \{X_0, \ldots, X_n\}$ with binary domains, where $X_0$ is a parent of all features:

$$\Pr(\mathbf{X}) = \Pr(X_0) \cdot \prod_{i=1,n} \Pr(X_i|X_0).$$

As usual, the class $\texttt{NBN} \stackrel{\text{def}}{=} \bigcup_{n\geq 0} \texttt{NBN}_n$. We write $X_0$ for the function $F$ that returns the value of feature $X_0$; that is, $F(\mathbf{x}) = \mathbf{x}_0$. We prove the following.

**Theorem 9.** *The decision problem* D-SHAP($\{X_0\}$, NBN) *is NP-hard.*

*Proof.* We use a reduction from the number partitioning problem, similar to the proof of Corollary 7. We note that the subset sum problem was also used to prove related hardness results, e.g., for proving hardness of the Shapely value in network games Elkind et al. (2008).

As before we assume w.l.o.g. that the sum of the numbers $k_i$ is even, $\sum_i k_i = 2c$ for some natural number $c$. Let $m$ be a large natural number to be defined later. We reduce the NUMPAR problem to the D-SHAP($\{X_0\}$, NBN) problem. The Naive Bayes network NBN consists of $n+1$ binary random variables $X_0, \ldots, X_n$. Let $X_i, \bar{X}_i$ denote the events $X_i = 1$ and respectively $X_i = 0$. We define the following probabilities of the NBN:

$$\frac{\Pr(X_0)}{\Pr(\bar{X}_0)} = e^{-\frac{m}{2} - mc} \qquad \frac{\Pr(X_i|X_0)}{\Pr(X_i|\bar{X}_0)} = e^{mk_i}$$

The probabilities $\Pr(\bar{X}_0)$ and $\Pr(X_i|\bar{X}_0)$ can be chosen arbitrarily (with the obvious constraints $\Pr(\bar{X}_0) \leq e^{\frac{m}{2}+mc}$ and $\Pr(X_i|\bar{X}_0) \leq e^{-mk_i}$). As required, our classifier is $F(X_0, \ldots, X_n) \stackrel{\text{def}}{=} X_0$. Let $a_k \stackrel{\text{def}}{=} \frac{k!(n-k)!}{(n+1)!}$ and let $\varepsilon > 0$ be any number such that $\varepsilon \leq a_k$ for all $k = 0, 1, \ldots, n$. We prove:

**Claim 5.** Let $\varepsilon$ be the value defined above. If $m$ satisfies both $2\sigma(-m/2) \leq \varepsilon$ and $1 - \sigma(m/2) \leq \varepsilon$, then NUMPAR has a solution iff $\textsc{Shap}_F(X_0) \geq 1/2(1+\varepsilon)$.

The claim implies Theorem 9. To prove the claim, we express the Shap explanation using Eq. (6). Let $\mathbf{X}_S$ denote the event $\bigwedge_{i \in S}(X_i = 1)$. Then, we can write the Shap explanation as:

$$\textsc{Shap}_F(X_0) = \sum_{S \subseteq [n]} a_{|S|} \left( \mathbf{E}[F \mid \mathbf{X}_{S \cup \{0\}}] - \mathbf{E}[F \mid \mathbf{X}_S] \right)$$

Obviously, $\mathbf{E}[X_0 \mid \mathbf{X}_{S\cup\{0\}}] = 1$. In addition, we have $\sum_{S\subseteq[n]} a_{|S|} = 1$, because there are $\binom{n}{k}$ sets of size $k$, hence $\sum_{S\subseteq[n]} a_{|S|} = \sum_{k=0,n} \binom{n}{k} \cdot \frac{k!(n-k)!}{(n+1)!} = 1$. Therefore $\textsc{Shap}_F(X_0) = 1 - D$, where:

$$D \stackrel{\text{def}}{=} \sum_{S \subseteq [n]} a_{|S|} \cdot \mathbf{E}[X_0 \mid \mathbf{X}_S] \tag{14}$$

To compute $D$, we expand:

$$\begin{aligned}
\mathbf{E}[X_0|\mathbf{X}_S] = \Pr(X_0|\mathbf{X}_S) &= \frac{\Pr(X_0, \mathbf{X}_S)}{\Pr(\mathbf{X}_S)} \\
&= \frac{\prod_{i\in S} \Pr(X_i|X_0)\Pr(X_0)}{\prod_{i\in S} \Pr(X_i|X_0)\Pr(X_0) + \prod_{i\in S} \Pr(X_i|\bar{X}_0)\Pr(\bar{X}_0)} \\
&= \frac{1}{1 + \Pr(\bar{X}_0)/\Pr(X_0) \cdot \prod_{i\in S}(\Pr(X_i|\bar{X}_0)/\Pr(X_i|X_0))} \\
&= \sigma(\text{weight}(S))
\end{aligned}$$

where:

$$\sigma(x) \stackrel{\text{def}}{=} \frac{1}{1+e^{-x}} \qquad \text{weight}(S) \stackrel{\text{def}}{=} -\frac{m}{2} - mc + m(\sum_{i \in S} k_i)$$

We compute $D$ in Eq. (14) by grouping each set $S$ with its complement $\bar{S} \stackrel{\text{def}}{=} [n] - S$:

$$D = \frac{1}{2} \sum_{S \subseteq [n]} a_{|S|} \left(\sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S}))\right) \tag{15}$$

If $S$ is a solution to the number partitioning problem, then:

$$\sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) = 2\sigma(-m/2)$$

Otherwise, one of weight$(S)$, weight$(\bar{S})$ is $\geq m/2$ and the other is $\leq -3m/2$ and therefore:

$$\sigma(m/2) \leq \sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) \leq 1 + \sigma(-3m/2)$$

As in the proof of Theorem 6, we obtain:

$$\begin{aligned} S \in P: &\qquad 0 \leq \sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) \leq \varepsilon \\ S \notin P: &\qquad 1-\varepsilon \leq \sigma(\text{weight}(S)) + \sigma(\text{weight}(\bar{S})) \leq 1+\varepsilon \end{aligned}$$

Therefore, using the fact that $\sum_{S \subseteq [n]} a_{|S|} = 1$, we derive these bounds for the expression (15) for $D$:

- If the number partitioning problem has no solution, then $D \geq 1/2(1-\varepsilon)$, and $\text{Shap}_F(X_0) \leq 1/2(1+\varepsilon)$.
- Otherwise, let $S$ be any solution to the `NUMPAR` problem, and $k = |S|$, then:

$$\begin{aligned} D &\leq \left(\frac{1}{2}(1+\varepsilon) - a_k(1+\varepsilon)\right) + a_k\varepsilon \\ &\leq \frac{1}{2} - \left(a_k - \frac{\varepsilon}{2}\right) < \frac{1}{2} - \frac{\varepsilon}{2} \end{aligned}$$

  and therefore $\text{Shap}_F(X_0) > 1/2(1+\varepsilon)$.

□

This result is in sharp contrast with the complexity of the Shap score over fully-factorized distributions in Section 3. There, the complexity was dictated by the choice of function class `F`. Here, the function is as simple as possible, yet computing Shap is hard. This ruins any hope of achieving tractability by restricting the function, and this motivates us to restrict the probability distribution in the next section. This result is also surprising because it is efficient to compute marginal probabilities (such as the expectation of $X_0$) and conditional probabilities in naive Bayes distributions.

Theorem 9 immediately extends to a large class of probability distributions and functions. We say that $F$ *depends only on* $X_i$ if there exist two constants $c_0 \neq c_1$ such that $F(\mathbf{x}) = c_0$ when $\mathbf{x}_i = 0$ and $F(\mathbf{x}) = c_1$ when $\mathbf{x}_i = 1$. In other words, $F$ ignores all variables other than $X_i$, and *does* depend on $X_i$. We then have the following.

**Corollary 10.** *The problem* D-SHAP(F, PR) *is NP-hard, when* PR *is any of the following classes of distributions:*

1. *Naive Bayes, bounded-treewidth Bayesian networks*
2. *Bayesian networks Markov networks, Factor graphs*
3. *Decomposable probabilistic circuits*

*and when* F *is any class that contains some function $F$ that depends only on $X_0$, including the class of linear regression models and all the classes listed in Corollaries 5 and 8.*

This corollary follows from two simple observations.First, each of the classes of probability distributions listed in the corollary can represent a naive Bayes network over binary variables $\mathbf{X}$. For example, a Markov network will consists of $n$ factors $f_1(X_0, X_1), f_2(X_0, X_2), \ldots, f_n(X_0, X_n)$; similar arguments prove that all the other classes can represent naive Bayes, including tractable probabilistic circuits such as sum-product networks (Vergari et al., 2020).

Second, for each function that depends only on $X_0$, there exist two distinct constants $c_0 \neq c_1 \in \mathbb{R}$ such that $F(\mathbf{x}) = c_0$ when $\mathbf{x}_0 = 0$ and $F(\mathbf{x}) = c_1$ when $\mathbf{x}_0 = 1$. For example, if we consider the class of logistic regression functions $F(\mathbf{x}) = \sigma(\sum_i w_i \mathbf{x}_i)$, then we choose the weights $w_0 = 1$, $w_1 = \ldots = w_n = 0$ and we obtain $F(\mathbf{x}) = 1/2$ when $\mathbf{x}_0 = 0$ and $F(\mathbf{x}) = 1/(1 + e^{-1})$ when $\mathbf{x}_0 = 1$. Then, over the binary domain $\{0, 1\}$ the function is equivalent to $F(\mathbf{x}) = (c_1 - c_0)\mathbf{x}_0 + c_0$, and, therefore, by the linearity of the Shap explanation (Equation (4)) we have $\text{Shap}_F(X_0) = (c_1 - c_0) \cdot \text{Shap}_{X_0}(X_0)$ (because the Shap explanation of a constant function $c_0$ is 0) for which, by Theorem 9, the decision problem is NP-hard.

We end this section by proving that Theorem 9 continues to hold even if the prediction function $F$ is the value of some leaf node of a (bounded treewidth) Bayesian Network. In other words, the hardness of the Shap explanation is not tied to the function returning the root of the network, and applies to more general functions.

**Corollary 11.** *The* Shap *decision problem for Bayesian networks with latent variables is NP-hard, even if the function $F$ returns a single leaf variable of the network.*

*Proof.* (Sketch) We use a reduction from the NUMPAR problem, as in the proof of Theorem 9. We start by constructing the NBN with variables $X_0, X_1, \ldots, X_n$ (as for Theorem 9), then add two more variables $X_{n+1}, X_{n+2}$, and edges $X_0 \to X_{n+1} \to X_{n+2}$, and define the random variables $X_{n+1}, X_{n+2}$ to be identical to $X_0$, i.e. $X_0 = X_{n+1} = X_{n+2}$. The prediction function is $F = X_{n+2}$, i.e. it returns the feature $X_{n+2}$, and the variables $X_0, X_{n+1}$ are latent. Thus, the new BN is identical to the NBN, and, since both models have exactly the same number of non-latent variables, the Shap-explanation is the same. □

## 5. Shap on Empirical Distributions

In supervised learning one does not require a generative model of the data, instead, the model is trained on some concrete data set: the *training data*. When some probabilistic model is needed, then the training data itself is conveniently used as a probability model, called the *empirical distribution*. This distribution captures dependencies between features,

while its set of possible worlds is limited to those in the data set. For example, the intent of the KernelSHAP algorithm by Lundberg and Lee (2017) is to compute the Shap explanation on the empirical distribution. In another example, Aas, Jullum, and Løland (2019) extend KernelSHAP to work with dependent features, by estimating the conditional probabilities from the empirical distribution.

Compared to the data distributions considered in the previous sections, the empirical distribution has one key advantage: it has many fewer possible worlds with positive probability – this suggests increased tractability. Unfortunately, in this section, we prove that computing the Shap explanation over the empirical distribution is #P-hard in general.

To simplify the presentation, this section assumes that all features are binary: $\mathrm{dom}(X_j) = \{0, 1\}$. The probability distribution is given by a 0/1-matrix $\boldsymbol{d} = (x_{ij})_{i\in[m],j\in[n]}$, where each row $(x_{i1}, \ldots, x_{in})$ is an outcome with probability $1/m$. One can think of $\boldsymbol{d}$ as a dataset with $n$ features and $m$ data instances, where each row $(x_{i1}, \ldots, x_{in})$ is one data instance. Repeated rows are possible: if a row occurs $k$ times, then its probability is $k/m$. We denote by $\mathtt{X}$ the class of empirical distributions. The predictive function can be any function $F : \{0, 1\}^n \to \mathbb{R}$. As our data distribution is no longer strictly positive, we adopt the standard convention that $\mathbf{E}[F|\mathbf{X}_S = 1] = 0$ when $\Pr(\mathbf{X}_S = 1) = 0$.

Recall from Section 2.2 that, by convention, we compute the Shap-explanation w.r.t. instance $\mathbf{e} = (1, 1, \ldots, 1)$, which is without loss of generality. Somewhat surprisingly, the complexity of computing the Shap-explanation of a function $F$ over the empirical distribution given by a matrix $\boldsymbol{d}$ is related to the problem of computing the expectation of a certain CNF formula associated to $\boldsymbol{d}$.

**Definition 4.** The *positive, partitioned 2CNF formula*, PP2CNF, associated to a matrix $\boldsymbol{d} \in \{0, 1\}^{m\times n}$ is:

$$\Phi_{\boldsymbol{d}} \stackrel{\text{def}}{=} \bigwedge_{(i,j):x_{ij}=0} (U_i \vee V_j).$$

Thus, a PP2CNF formula is over $m + n$ variables $U_1, \ldots, U_m, V_1, \ldots, V_n$, and has only positive clauses. The matrix $\boldsymbol{d}$ dictates which clauses are present. A *quasy-symmetric probability distribution* is a fully factorized distribution over the $m + n$ variables for which there exists two numbers $p, q \in [0, 1]$ such that for every $i = 1, m$, $\Pr(U_i = 1) = p$ or $\Pr(U_i = 1) = 1$, and for every $j = 1, n$, $\Pr(V_j = 1) = q$ or $\Pr(V_j = 1)$. In other words, all variables $U_1, \ldots, U_m$ have the same probability $p$, or have probability 1, and similarly for the variables $V_1, \ldots, V_n$. We denote by $\mathtt{EQS(PP2CNF)}$ the expectation computation problem for PP2CNF over quasi-symmetric probability distributions. $\mathtt{EQS(PP2CNF)}$ is #P-hard, because computing $\mathbf{E}[\Phi_{\boldsymbol{d}}]$ under the uniform distribution (i.e. $\Pr(U_1 = 1) = \cdots = \Pr(V_n = 1) = 1/2$) is #P-hard Provan and Ball (1983). We prove:

**Theorem 12.** *Let* $\mathtt{X}$ *be the class of empirical distributions, and* $\mathtt{F}$ *be any class of function such that, for each* $i$*, it includes some function that depends only on* $X_i$*. Then, we have that* $\mathtt{F\text{-}SHAP(F, X)} \equiv^P \mathtt{EQS(PP2CNF)}$.

*As a consequence, the problem* $\mathtt{F\text{-}SHAP(F, X)}$ *is #P-hard in the size of the empirical distribution.*

The theorem is surprising, because the set of possible outcomes of an empirical distribution is small. This is unlike all the distributions discussed earlier, for example those mentioned in Corollary 10, which have $2^n$ possible outcomes, where $n$ is the number of features. In particular, given an empirical distribution $\boldsymbol{d}$, one can compute the expectation $\mathbf{E}[F]$ in polynomial time for any function $F$, by doing just one iteration over the data. Yet, computing the Shap explanation of $F$ is #P-hard.

Theorem 12 implies hardness of Shap explanations on the empirical distribution for a large class of functions.

**Corollary 13.** *The problem* $\texttt{F-SHAP}(\texttt{F}, \texttt{X})$ *is #P-hard, when* $\texttt{X}$ *is the class of empirical distributions, and* $\texttt{F}$ *is any class such that for each feature* $X_i$*, the class contains some function that depends only on* $X_i$*. This includes all the function classes listed in Corollaries 5 and 8.*

For instance, any class of Boolean function that contains the $n$ single-variable functions $F \stackrel{\text{def}}{=} X_i$, for $i = 1, n$, fall under this corollary. Section 4 showed an example of how the class of logistic regression functions fall under this corollary as well.

The proof of Theorem 12 follows from the following technical lemma, which is of independent interest:

**Lemma 14.** *We have that:*

1. *For every matrix* $\boldsymbol{d}$, $\texttt{F-SHAP}(\texttt{F}, \boldsymbol{d}) \leq^P \texttt{EQS}(\{\Phi_{\boldsymbol{d}}\})$.
2. $\texttt{EQS}(\texttt{PP2CNF}) \leq^P \texttt{F-SHAP}(\texttt{F}, \texttt{X})$.

The proof of the Lemma is given in Sections 5.1 and 5.2. The first item says that we can compute the Shap-explanation in polynomial time using an oracle for computing $\mathbf{E}[\Phi_{\boldsymbol{d}}]$ over quasi-symmetric distributions. The oracle is called only on the PP2CNF $\Phi_{\boldsymbol{d}}$ associated to the data $\boldsymbol{d}$, but may perform repeated calls, with different probabilities of the Boolean variables. This is somewhat surprising because the Shap explanation is over an empirical distribution, while $\mathbf{E}[\Phi_{\boldsymbol{d}}]$ is taken over a fully-factorized distribution; there is no connection between these two distributions. This item immediately implies $\texttt{F-SHAP}(\texttt{F}, \texttt{X}) \leq^P \texttt{EQS}(\texttt{PP2CNF})$, where $\texttt{X}$ is the class of empirical distributions $\boldsymbol{d}$, since the formula $\Phi_{\boldsymbol{d}}$ is in the class $\texttt{PP2CNF}$.

The second item says that a weak form of converse also holds. It states that we can compute in polynomial time the expectation $\mathbf{E}[\Phi]$ over a quasi-symmetric probability distributions by using an oracle for computing Shap explanations, over several matrices $\boldsymbol{d}$, but not necessarily restricted to the matrix associated to $\Phi$. Together, the two items of the lemma prove Theorem 12.

We end this section with a comment on the TreeSHAP algorithm in Lundberg et al. (2020), which is computed over a distribution defined by a tree-based model. Our result implies that the problem that TreeSHAP tries to solve is #P-hard. This follows immediately by observing that every empirical distribution $\boldsymbol{d}$ can be represented by a binary tree of size polynomial in the size of $\boldsymbol{d}$. The tree examines the attributes in the order $X_1, X_2, \ldots, X_n$, and each decision node for $X_i$ has two branches: $X_i = 0$ and $X_i = 1$. A branch that does not exists in the matrix $\boldsymbol{d}$ will end in a leaf with label 0. A complete branch that

corresponds to a row $x_{i1}, x_{i2}, \ldots, x_{in}$ in $\boldsymbol{d}$ ends in a leaf with label $1/m$ (or $k/m$ if that row occurs $k$ times in $\boldsymbol{d}$). The size of this tree is no larger than twice the size of the matrix (because of the extra dead end branches). This concludes our study of Shap explanations on the empirical distribution.

### 5.1 Proof of Lemma 14 (1)

Fix a PP2CNF $\Phi = \bigwedge(U_i \vee V_j)$. A *symmetric* probability space is defined by two numbers $p, q \in [0, 1]$ and consists of the fully-factorized distribution where $\Pr(U_1) = \Pr(U_2) = \cdots = p$ and $\Pr(V_1) = \Pr(V_2) = \cdots = q$. A *quasi-symmetric* probability space consists of two sets of indices $I, J$ and two numbers $p, q$ such that:

$$\Pr(U_i) = \begin{cases} p & \text{when } i \notin I \\ 1 & \text{when } i \in I \end{cases} \qquad \Pr(V_j) = \begin{cases} q & \text{when } j \notin J \\ 1 & \text{when } j \in J \end{cases}$$

In this and the following section we prove Lemma 14: computing the Shap-explanation over an empirical distribution is polynomial time equivalent to computing the expectation of PP2CNF formulas over a (quasi)-symmetric distribution. Provan and Ball (1983) proved that computing the expectation of a PP2CNF over uniform distributions is #P-hard in general. Since uniform distribution are symmetric (namely $p = q = 1/2$) it follows that computing Shap-explanations is #P-hard in general.

In this section we prove item (1) of Lemma 14. Fix a 0/1-matrix $\boldsymbol{x}$ defining an empirical distribution, and let $F$ be a real-valued prediction function over these features. Let $\Phi_{\boldsymbol{x}}$ be the PP2CNF associated to $\boldsymbol{x}$ (see Definition 4). Will assume w.l.o.g. that $\boldsymbol{x}$ has $n + 1$ features (columns), denoted $X_0, X_1, \ldots, X_n$. The prediction function $F$ is any function $F : \{0, 1\}^{n+1} \to \mathbb{R}$. We prove:

**Proposition 15.** *One can compute* $\text{Shap}_F(X_0)$ *in polynomial time using an oracle for computing* $\mathbf{E}[\Phi_{\boldsymbol{x}}]$ *over quasi-symmetric distributions.*

Denote by $y_i \stackrel{\text{def}}{=} F(x_{i0}, x_{i1}, \ldots, x_{in})$, $i = 1, m$ the value of $F$ on the $i$'th row of the matrix $\boldsymbol{x}$. Since the only possible outcomes of the probability space are the $m$ rows of the matrix, the quantity $\text{Shap}_F(X_0)$ depends only on the vector $\boldsymbol{y} \stackrel{\text{def}}{=} (y_1, \ldots, y_m)$. Furthermore, by the linearity of the Shap explanation (Eq. (4)), it suffices to compute the Shap explanation in the case when $\boldsymbol{y}$ has a single value $= 1$ and all others are $= 0$. By permuting the rows of the matrix, we will assume w.l.o.g. that $y_1 = 1$, and $y_2 = y_3 = \cdots = y_m = 0$. In summary, denoting $F_1$ the function that is 1 on the first row of the matrix $\boldsymbol{x}$ and is 0 on all other rows, our task is to compute $\text{Shap}_{F_1}(X_0)$.

For that we use the following expression for Shap (see also Sec. 3):

$$\text{Shap}_{F_1}(X_0) = \sum_{k=0,n} \frac{k!(n-k)!}{(n+1)!} \Big( \sum_{S \subseteq [n]: |S|=k} \big( \mathbf{E}[F_1 | \mathbf{X}_{S \cup \{0\}} = 1] - \mathbf{E}[F_1 | \mathbf{X}_S = 1] \big) \Big) \tag{16}$$

We will only show how to compute the quantity

$$v_{F_1,k} = \sum_{S\subseteq[n]:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1] \tag{17}$$

using an oracle to $\mathbf{E}[\Phi_{\boldsymbol{x}}]$, because the quantity $\sum_{S:|S|=k} \mathbf{E}[F_1|\mathbf{X}_{S\cup\{0\}} = 1]$ is computed similarly, by restricting the matrix $\boldsymbol{x}$ to the rows $i$ where $x_{i0} = 1$. The PP2CNF $\Phi$ associated to this restricted matrix is obtained from $\Phi_{\boldsymbol{x}}$ as follows. Let $I \stackrel{\text{def}}{=} \{i \mid x_{i0} = 1\}$ be the set of rows of the matrix where the feature $X_0$ is 1. Then, we need to remove all clauses of the form $(U_i \vee V_j)$ for $i \in I$. This is equivalent to setting $U_i := 1$ in $\Phi_{\boldsymbol{x}}$. Therefore, we can compute the expectation of the restricted $\Phi$ by using our oracle for $\mathbf{E}[\Phi_{\boldsymbol{x}}]$, and running it over the probability space where we define $\Pr(U_i) \stackrel{\text{def}}{=} 1$ for all $i \in I$. Hence, it suffices to show only how to compute the expression (17). Notice that the quantity $v_{F_1,k}$ is the same as what we defined earlier in Eq. (7).

Column $X_0$ of the matrix is not used in expression (17), because the set $S$ ranges over subsets of $[n]$. Hence w.l.o.g. we can drop feature $X_0$ and denote by $\boldsymbol{x}$ (with some abuse) the matrix that only has the features $X_1, \ldots, X_n$. In other words, $\boldsymbol{x} \in \{0,1\}^{m\times n}$. The PP2CNF formula for the modified matrix is obtained from $\Phi_{\boldsymbol{x}}$ by setting $V_0 := 1$, hence we can compute its expectation by using our oracle for $\mathbf{E}[\Phi_{\boldsymbol{x}}]$.

We introduce the following quantities associated to the matrix $\boldsymbol{x} \in \{0,1\}^{m\times n}$:

- For all $S \subseteq [n]$, $\ell \le m, k \le n$, we define:

$$g(S) \stackrel{\text{def}}{=} \{i \mid \forall j \in S, x_{ij} = 1\} \tag{18}$$

$$a_{\ell k} \stackrel{\text{def}}{=} |\{S \mid |S| = k, |g(S)| = \ell\}| \tag{19}$$

- We define the sequence $v_k$, $k = 0, 1, \ldots, n$:

$$v_k \stackrel{\text{def}}{=} \sum_{l=1,m} \frac{a_{\ell k}}{\ell} \tag{20}$$

- We define the value $V$:

$$V \stackrel{\text{def}}{=} \sum_{k=0,n} \frac{k!(n-k)!}{(n+1)!} v_k \tag{21}$$

We prove that, under a certain condition, the value $v_k$ in Eq. (20) is equal to Eq. (17); this justifies the notation $v_k$, since it turns out to be the same as $v_{F_1,k}$ from Eq. (7).

**Definition 5.** Call the matrix $\boldsymbol{x}$ "good" if $\forall i, j$, $x_{1j} \ge x_{ij}$.

In other words, the matrix is "good" if the first row dominates all others. In general the matrix $\boldsymbol{x}$ need not be "good", however we can make it "good" by removing all columns where row 1 has a value 0. More precisely, let $J^{(1)} \stackrel{\text{def}}{=} \{j \mid x_{1j} = 1\}$ denote the non-zero

positions of the first row, and let $\boldsymbol{x}^{(1)}$ denote the sub-matrix of $\boldsymbol{x}$ consisting of the columns $J^{(1)}$. Obviously, $\boldsymbol{x}^{(1)}$ is "good", because its first row is $(1, 1, \ldots, 1)$. The following hold:

$$\begin{aligned}
&\text{If } S \subseteq J^{(1)}: && \mathbf{E}_{\boldsymbol{x}}[F_1|\mathbf{X}_S = 1] = \mathbf{E}_{\boldsymbol{x}^{(1)}}[F_1|\mathbf{X}_S = 1] \\
&\text{If } S \not\subseteq J^{(1)}: && \mathbf{E}_{\boldsymbol{x}}[F_1|\mathbf{X}_S = 1] = 0
\end{aligned}$$

(When $S \not\subseteq J^{(1)}$ then the quantity $\mathbf{E}_{\boldsymbol{x}^{(1)}}[F_1|\mathbf{X}_S = 1]$ is undefined). Therefore:

$$\sum_{S\subseteq[n]:|S|=k} \mathbf{E}_{\boldsymbol{x}}[F_1|\mathbf{X}_S = 1] = \sum_{S\subseteq J^{(1)}:|S|=k} \mathbf{E}_{\boldsymbol{x}^{(1)}}[F_1|\mathbf{X}_S = 1]$$

It follows that, in order to compute the values in Eq. (17), we can consider the matrix $\boldsymbol{x}^{(1)}$ instead of $\boldsymbol{x}$; its associated PP2CNF is obtained from $\Phi_{\boldsymbol{x}}$ by setting $V_j := 1$ for all $j \in [m] - J^{(1)}$, hence we can compute its expectation over a quasi-symmetric space by using our oracle for computing $\mathbf{E}[\Phi_{\boldsymbol{x}}]$ over quasi-symmetric spaces. To simplify the notation, we will still use the name $\boldsymbol{x}$ for the matrix instead of $\boldsymbol{x}^{(1)}$, and assume w.l.o.g. that the first row of the matrix $\boldsymbol{x}$ is $(1, 1, \ldots, 1)$.

We prove that, when $\boldsymbol{x}$ is "good", then $v_k$ is indeed the quantity Eq. (17) that we want to compute. This holds for any "good" matrix, not just matrices with $(1, 1, \ldots, 1)$ in the first row, and we need this more general result later in Sec. 5.2.

**Claim 6.** If the matrix $\boldsymbol{x}$ is "good", then, for any $k = 0, n$:

$$v_k = \sum_{S:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1]$$

*Proof.* Recall that $J^{(1)} \stackrel{\text{def}}{=} \{j \mid x_{1j} = 1\}$. Let $S \subseteq [n]$ be any set of columns. We consider two cases, depending on whether $S$ is a subset of $J^{(1)}$ or not:

$$\begin{aligned}
&S \subseteq J^{(1)}: && |g(S)| > 0 && \mathbf{E}[F_1|\mathbf{X}_S = 1] = \frac{1}{|g(S)|} \\
&S \not\subseteq J^{(1)}: && |g(S)| = 0 && \mathbf{E}[F_1|\mathbf{X}_S = 1] = 0
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\sum_{S\subseteq[n]:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1] = \sum_{S\subseteq J^{(1)}:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1] \\
&= \sum_{S\subseteq J^{(1)}:|S|=k} \frac{1}{|g(S)|} = \sum_{S:|S|=k,|g(S)|>0} \frac{1}{|g(S)|} = \sum_{\ell>0} \frac{a_{\ell k}}{\ell}
\end{aligned}$$

□

At this point we introduce two polynomials, $P$ and $Q$.

**Definition 6.** Fix an $m \times n$ matrix $\boldsymbol{x}$ with 0, 1-entries. The polynomials $P(u, v)$ and $Q(u, v)$ in real variables $u, v$ associated to the matrix $\boldsymbol{x}$ are the following:

$$P(u,v) \stackrel{\text{def}}{=} \sum_{S \subseteq [n]} u^{|g(S)|} v^{|S|}$$

$$Q(u,v) \stackrel{\text{def}}{=} \sum_{\substack{T \subseteq [m], S \subseteq [n]: \\ \forall (i,j) \in T \times S: \; x_{ij} = 1}} u^{|T|} v^{|S|}$$

The polynomials are defined by summing over exponentially many sets $S \subseteq [n]$, or pairs of sets $S \subseteq [n], T \subseteq [m]$. In the definition of $P$, we use the function $g(S)$ associated to the matrix $\boldsymbol{x}$, see Eq. (18). In the definition of $Q(u, v)$ we sum only those pairs $T, S$ where $\forall i \in T$, $\forall j \in S$, $x_{ij} = 1$. While their definition involves exponentially many terms, these polynomials have only $(m + 1)(n + 1)$ terms, because the degrees of the variables $u, v$ are $m$ and $n$ respectively. We claim that these terms are as follows:

**Claim 7.** The following identities hold:

$$P(u,v) = \sum_{\ell=0,m; k=0,n} a_{\ell k} u^{\ell} v^{k}$$

$$Q(u,v) = P(1+u, v)$$

*Proof.* The identity for $P(u, v)$ follows immediately from the definition of $a_{\ell k}$. We prove the identity for $Q$. From the definition of $g(S)$ in Eq. (18) we derive the following equivalence:

$$(\forall i \in T, \forall j \in S : x_{ij} = 1) \qquad \Leftrightarrow \qquad T \subseteq g(S)$$

Which implies:

$$Q(u,v) = \sum_{S \subseteq [n], T \subseteq g(S)} u^{|T|} v^{|S|}$$

and the claim follows from $\sum_{T \subseteq g(S)} u^{|T|} = (1 + u)^{|g(S)|}$. □

Thus, in order to compute the quantities $v_k$ for $k = 0, 1, \ldots, n$ it suffices to compute the coefficients $a_{\ell k}$ of the polynomial $P(u, v)$, and, for that, it suffices to compute the coefficients of the polynomial $Q(u, v)$. For that, we establish the following important connection between $\mathbf{E}[\Phi_{\boldsymbol{x}}]$ and the polynomial $Q(u, v)$. Fix $u, v > 0$ any two positive real values, and let $p \stackrel{\text{def}}{=} 1/(1 + u)$, $q \stackrel{\text{def}}{=} 1/(1 + v)$; notice that $p, q \in (0, 1)$. Consider the probability space over independent Boolean variables $U_1, \ldots, U_m, V_1, \ldots, V_n$ where $\forall i \in [m]$, $\Pr(U_i) = p$, and $\forall j \in [n]$, $\Pr(V_j) = q$. Then:

**Claim 8.** Given the notations above, the following identity holds:

$$\mathbf{E}[\Phi_{\boldsymbol{x}}] = \frac{1}{(1+u)^m (1+v)^n} Q(u,v) \tag{22}$$

*Proof.* A truth assignment for $\Phi_{\boldsymbol{x}}$ consists of two assignments, $\theta \in \{0,1\}^m$ for the variables $U_i$, and $\tau \in \{0,1\}^n$ for the variables $V_j$. Defining $T \stackrel{\text{def}}{=} \{i \mid \theta(U_i) = 0\}$ and $S \stackrel{\text{def}}{=} \{j \mid \tau(V_j) = 0\}$, we observe that $\Phi_{\boldsymbol{x}}[\theta, \tau] = \texttt{true}$ iff $\forall i \in T, \forall j \in S$, $x_{ij} = 1$, and therefore:

$$\begin{aligned}
\Pr(\Phi_{\boldsymbol{x}}) &= \sum_{\theta,\tau:\Phi[\theta,\tau]=1} \Pr(\theta)\Pr(\tau) \\
&= \sum_{\substack{T \subseteq [m], S \subseteq [n] \\ \forall (i,j) \in T \times S : x_{ij} = 1}} p^{m-|T|}(1-p)^{|T|} q^{n-|S|}(1-q)^{|S|} \\
&= p^m q^n Q((1-p)/p, (1-q)/q)
\end{aligned}$$

□

Finally, to prove Lemma 14 (1), it suffices to show how to use an oracle for $E[\Phi_{\boldsymbol{x}}]$ to compute the coefficients of the polynomial $Q(u, v)$. We denote by $b_{\ell k}$ these coefficients, in other words:

$$Q(u,v) = \sum_{\ell=0,m;k=0,n} b_{\ell k} u^\ell v^k \tag{23}$$

To compute the coefficients $b_{\ell k}$, we proceed as follows. Choose $m+1$ distinct values $u_0, u_1, \ldots, u_m > 0$, and choose $n+1$ distinct values $v_0, v_1, \ldots, v_n > 0$, and for all $i = 0, m$ and $j = 0, n$, use the oracle for $\mathbf{E}[\Phi_{\boldsymbol{x}}]$ to compute $Q(u_i, v_j)$ as per identity (22). This leads to a system of $(m+1)(n+1)$ equations whose unknowns are the coefficients $b_{\ell k}$ (see Eq. (23)) and whose coefficients are $u_i^\ell v_j^k$. The matrix $\boldsymbol{A}$ of this system of equations is an $[(m+1)(n+1)] \times [(m+1)(n+1)]$ matrix, whose rows are indexed by pairs $(i, j)$, and whose columns are indexed by pairs $(\ell, k)$:

$$A_{(ij),(\ell k)} = u_i^\ell v_j^k$$

We prove that this matrix is non-singular, and for that we observe that it is the Kronecker product of two Vandermonde matrices. Recall that the $t \times t$ Vandermonde matrix defined by $t$ numbers $z_1, \ldots, z_t$ is:

$$V(z_1, \ldots, z_t) = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ z_1 & z_2 & \ldots & z_t \\ z_1^2 & z_2^2 & \ldots & z_t^2 \\ & & \ldots & \\ z_1^{t-1} & z_2^{t-1} & \ldots & z_t^{t-1} \end{bmatrix}$$

It is known that $\det(V(z_1, \ldots, z_t)) = \prod_{1 \le i < j \le t}(z_j - z_i)$ and this is $\neq 0$ iff the values $z_1, \ldots, z_t$ are distinct. We observe that the matrix $\boldsymbol{A}$ is the Kronecker product of two Vandermonde matrices:

$$\boldsymbol{A} = V(u_0, u_1, \ldots, u_m) \otimes V(v_0, v_1, \ldots, v_n)$$

Since we have chosen $u_0, \ldots, u_m$ to be distinct, and similarly for $v_0, \ldots, v_n$, it follows that both Vandermonde matrices are non-singular, hence $\det(\boldsymbol{A}) \neq 0$. Thus, we can solve this linear system of equations in time $O\left(((m+1)(n+1))^3\right)$, and compute all coefficients $b_{\ell k}$.

**Putting It Together** We prove now Proposition 15. We are given a 0/1 matrix $\boldsymbol{x}$ with $n+1$ features $X_0, \ldots, X_n$ and $m$ rows. To compute $\text{SHAP}_F(X_0)$ we proceed as follows:

1. For each $i = 1, m$, compute $\text{SHAP}_{F_i}(X_0)$, where $F_i$ is the function defined as $= 1$ on row $i$ of the matrix, and $= 0$ on all other rows of the matrix. Return $\text{SHAP}_F(X_0) = \sum_{i=1,m} y_i \text{SHAP}_{F_i}(X_0)$, where $y_i \stackrel{\text{def}}{=} F(x_{i0}, x_{i1}, \ldots, x_{in})$ is the value of $F$ on the $i$'th row of the matrix.

2. To compute $\text{SHAP}_{F_i}(X_0)$, switch rows 1 and $i$ of the matrix, and compute $\text{SHAP}_{F_1}(X_0)$ on the modified matrix.

3. To compute $\text{SHAP}_{F_1}(X_0)$, compute both sums in Eq. (16).

4. To compute $\sum_{S\subseteq[n]:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1]$, perform steps (5) to (8) below.

5. Let $J^{(1)} = \{j \mid j \in [n], x_{1j} = 1\}$; notice that $0 \notin J^{(1)}$. Let $n^{(1)} = |J^{(1)}|$. Let $\Phi'$ denote the PP2CNF obtained from $\Phi_{\boldsymbol{x}}$ by setting $V_j := 1$ for all $j \notin J^{(1)}$. Thus, $\Phi'$ has $m + n^{(1)}$ variables: $U_i$ for $i \in [m]$, and $V_j$ for $j \in J^{(1)}$.

6. Choose distinct values $u_0, u_1, \ldots, u_m \in (0,1)$ and distinct values $v_0, v_1, \ldots, v_{n^{(1)}} \in (0,1)$. For each fixed combination $u_\alpha, v_\beta$, compute $Q(u_\alpha, v_\beta) = (1+u_\alpha)^m (1+v_\beta)^{n^{(1)}} \mathbf{E}[\Phi']$ (see Claim 8). The value $\mathbf{E}[\Phi']$ over the probability space where, for all $i, j$: $\Pr(U_i) = u_\alpha$, $\Pr(V_j) = v_\beta$: this can be done by computing $\mathbf{E}[\Phi_{\boldsymbol{x}}]$ over a quasi-symmetric space.

7. Using the $(m+1)(n^{(1)}+1)$ results from the previous step, form a system of Equations where the unknowns are the coefficients $b_{\ell k}$, $\ell = 0, m$, $k = 0, n^{(1)}$, of the polynomial $Q(u, v)$, see (23). Solve for the coefficients $b_{\ell k}$.

8. Compute the coefficients $a_{\ell k}$ of the polynomial $P(u, v) = Q(u-1, v)$, see Claim 7, then compute $v_k = \sum_\ell a_{\ell k}/\ell$. By Claim 6, $v_k = \sum_{S:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1]$, completing Step (4).

9. To compute $\sum_{S\subseteq[n]:|S|=k} \mathbf{E}[F_1|\mathbf{X}_{S\cup\{0\}} = 1]$, first set $U_i := 0$ for all rows $i$ where $x_{i0} = 0$, then repeat steps (5) to (8).

10. This completes Step (3), and we obtain $\text{SHAP}_{F_1}(X_0)$.

### 5.2 Proof of Lemma 14 (2)

Here we prove item (2) of Lemma 14: one can compute $\mathbf{E}[\Phi]$ over a quasi-symmetric probability space in polynomial time, given an oracle for SHAP on empirical distributions. If the probability space sets $\Pr(U_i) = 1$ for some variable, then we can simply replace $\Phi$ with $\Phi[U_i := 1]$, and similarly if $\Pr(V_j) = 1$. Hence, w.l.o.g., we can assume that the probability space is symmetric.

More precisely, we fix a PP2CNF formula $\Phi = \bigwedge(U_i \vee V_j)$, and let $p = \Pr(U_1) = \cdots = \Pr(U_m)$ and $q = \Pr(V_1) = \cdots = \Pr(V_n)$ define a symmetric probability space. Our task is to compute $\mathbf{E}[\Phi]$ over this space, given an oracle for computing SHAP-explanations on

empirical distributions. Throughout this section we will use the notations introduced in Sec. 5.1.

Let $\boldsymbol{x}$ the matrix associated to $\Phi$: $x_{ij} = 0$ iff $\Phi$ contains a clause $U_i \vee V_j$. We describe our algorithm for computing $\mathbf{E}(\Phi)$ in three steps.

**Step 1:** $\mathbf{E}[\Phi] \leq^P (v_0, v_1, \ldots, v_k)$. More precisely:, we claim that we can compute $\Pr(\Phi)$ using an oracle for computing the quantities $v_0, v_1, \ldots, v_n$ defined in Eq. (20). We have seen in Eq. (22) that $\mathbf{E}[\Phi] = \frac{1}{(1+u)^m(1+v)^n} Q(u, v)$ where $u = (1-p)/p$ and $v = (1-q)/q$. From Claim 7 we know that $Q(u, v) = P(1 + u, v)$, and the coefficients of $P(u, v)$ are the quantities $a_{\ell k}$ defined in Eq. (19). To complete Step 1, we will describe a polynomial time algorithm that computes the quantities $a_{\ell k}$ associated to our matrix $\boldsymbol{x}$, with access to an oracle for computing the quantities $v_0, \ldots, v_k$ associated to any matrix $\boldsymbol{x}'$.

Starting from the matrix $\boldsymbol{x}$, construct $m+1$ new matrices, denoted by $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m+1)}$, where, for each $\Gamma = 1, m+1$, $\boldsymbol{x}^{(\Gamma)}$ consists of the matrix $\boldsymbol{x}$ extended with $\Gamma$ rows consisting of $(1, 1, \ldots, 1)$. That is, the matrix $\boldsymbol{x}^{(\Gamma)}$ has $\Gamma + m$ rows, the first $\Gamma$ rows are $(1, \ldots, 1)$, and the remaining $m$ rows are those in $\boldsymbol{x}$. We run our oracle to compute the quantities $v_k$ on each matrix $\boldsymbol{x}^{(\Gamma)}$. We continue to use the notations $g(S), a_{\ell k}, v_k$ introduced in Equations (18), (19), (20) for the matrix $\boldsymbol{x}$, and add the superscript $(\Gamma)$ for the same quantities associated to the matrix $\boldsymbol{x}^{(\Gamma)}$. We observe:

$$\begin{aligned} g^{(\Gamma)} =& g(S) \cup \{\text{the } \Gamma \text{ new rows}\} \\ a^{(\Gamma)}_{\ell+\Gamma,k} =& a_{\ell k} \end{aligned}$$

and therefore:

$$\begin{aligned} v_k^{(1)} &= \frac{1}{1} a_{0k} + \frac{1}{2} a_{1k} + \cdots + \frac{1}{m+1} a_{mk} \\ v_k^{(2)} &= \frac{1}{2} a_{0k} + \frac{1}{3} a_{1k} + \cdots + \frac{1}{m+2} a_{mk} \\ &\cdots \\ v_k^{(m+1)} &= \frac{1}{m+2} a_{0k} + \frac{1}{m+3} a_{1k} + \cdots + \frac{1}{2m+2} a_{mk} \end{aligned}$$

By solving this system of equations, we compute the quantities $a_{\ell k}$ for $\ell = 0, m$. The matrix of this system is a special case of Cauchy's double alternant determinant:

$$\det \left[ \frac{1}{x_i + y_j} \right] = \frac{\prod_{1 \leq i < j \leq n} (x_i - x_j)(y_i - y_j)}{\prod_{i,j} (x_i + y_j)}$$

where $x_i = i$ and $y_j = j - 1$, and therefore the matrix of the system is non-singular.

We observe that all matrices $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m+1)}$ are "good" (see Definition 5), because their first row is $(1, \ldots, 1)$.

**Step 2:** Let $\boldsymbol{x}$ be a "good" matrix (Definition 5). Then: $(v_0, v_1, \ldots, v_n) \leq^P V$ ($V$ defined in Eq. (21)). In other words, given a matrix $\boldsymbol{x}$, we claim that we can compute the quantities $v_0, v_1, \ldots, v_n$ associated to $\boldsymbol{x}$ by Eq. (20) in polynomial time, given access to an oracle for computing the quantity $V$ associated to any matrix $\boldsymbol{x}'$. The algorithm proceeds

as follows. For each $\Delta = 0, 1, \ldots, n$, construct a new $m \times (2n)$ matrix $x^{(\Delta)}$ by extending $\boldsymbol{x}$ with $\Delta$ new columns set to 1 and $n - \Delta$ new columns set to 0. Thus, $x^{(\Delta)}$ is:

$$\begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} & 1 & 1 & \ldots & 1 & 0 & \ldots & 0 \\ x_{21} & x_{22} & \ldots & x_{2n} & 1 & 1 & \ldots & 1 & 0 & \ldots & 0 \\ & & \ldots & & & & & & & & \\ x_{m1} & x_{m2} & \ldots & x_{mn} & 1 & 1 & \ldots & 1 & 0 & \ldots & 0 \end{pmatrix}$$

Notice that $x^{(\Delta)}$ is "good", for any $\Delta$. We run the oracle on each matrix $x^{(\Delta)}$ to compute the quantity $V^{(\Delta)}$. We start by observing the following relationships between the parameters of the matrix $\boldsymbol{x}$ and those of the matrix $\boldsymbol{x}^{(\Delta)}$:

$$\begin{aligned} g^{(\Delta)}(S) =& g(\Delta \cap [n]) \\ a^{(\Delta)}_{\ell p} =& \sum_{k=0,\min(p,n)} \binom{\Delta}{p-k} a_{\ell k} \\ v^{(\Delta)}_{p} =& \sum_{k=0,\min(p,n)} \binom{\Delta}{p-k} v_k \end{aligned}$$

Notice that, when $p > n + \Delta$, then $v^{(\Delta)}_p = 0$. We use the oracle to compute the quantity $V^{(\Delta)}$, which is:

$$\begin{aligned} V^{(\Delta)} =& \sum_{p=0,2n} \frac{p!(2n-p)!}{(2n+1)!} v^{(\Delta)}_p \\ =& \frac{1}{2n+1} \sum_{p=0,n+\Delta} \frac{1}{\binom{2n}{p}} v^{\Delta}_p \\ =& \frac{1}{2n+1} \sum_{p=0,n+\Delta} \sum_{k=0,\min(p,n)} \frac{\binom{\Delta}{p-k}}{\binom{2n}{p}} v_k \\ =& \frac{1}{2n+1} \sum_{k=0,n} \sum_{p=k,k+\Delta} \frac{\binom{\Delta}{p-k}}{\binom{2n}{p}} v_k \\ =& \frac{1}{2n+1} \sum_{k=0,n} \left( \sum_{q=0,\Delta} \frac{\binom{\Delta}{q}}{\binom{2n}{k+q}} \right) v_k \\ \stackrel{\text{def}}{=}& \frac{1}{2n+1} \sum_{k=0,n} A_{\Delta,k} \cdot v_k \end{aligned}$$

Thus, after running the oracle on all matrices $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(n)}$, we obtain a system of $n + 1$ equations with $n + 1$ unknowns $v_0, v_1, \ldots, v_n$. It remains to prove that system's matrix,

$A_{\Delta,k}$, is non-singular matrix. Let us denote following matrices by:

$$A_{\Delta,k} \stackrel{\text{def}}{=} \sum_{q=0,\Delta} \frac{\binom{\Delta}{q}}{\binom{2n}{k+q}} \qquad \Delta = 0, n; k = 0, n;$$

$$B_{\Delta,q} \stackrel{\text{def}}{=} \binom{\Delta}{q} \qquad \Delta = 0, n; q = 0, n;$$

$$C_{q,k} \stackrel{\text{def}}{=} \frac{1}{\binom{2n}{k+q}} \qquad q = 0, n; k = 0, n;$$

It is immediate to verify that $\boldsymbol{A} = \boldsymbol{B} \cdot \boldsymbol{C}$, so it suffices to prove $\det(\boldsymbol{B}) \neq 0$, $\det(\boldsymbol{C}) \neq 0$. We start with $\boldsymbol{B}$, and for that consider the Vandermonde matrix $\boldsymbol{X} \stackrel{\text{def}}{=} V(x_0, x_1, \ldots, x_n)$, $X_{qt} \stackrel{\text{def}}{=} x_t^q$. Denoting $\boldsymbol{Y} \stackrel{\text{def}}{=} \boldsymbol{B} \cdot \boldsymbol{X}$, we have that

$$Y_{\Delta t} = \sum_{q=0,n} B_{\Delta,q} X_{q,t} = \sum_{q=0,n} \binom{\Delta}{q} x_t^q = (1 + x_t)^\Delta$$

is also a Vandermonde matrix $\boldsymbol{Y} = V(1+x_0, 1+x_1, \ldots, 1+x_n)$. We have $\det(\boldsymbol{Y}) \neq 0$ when $x_0, x_1, \ldots, x_n$ are distinct, proving that $\det(\boldsymbol{B}) \neq 0$.

Finally, we prove $\det(\boldsymbol{C}) \neq 0$. For that, we prove a slightly more general result. For any $N \geq 2n$, denote by $\boldsymbol{C}^{(n,N)}$ the following $(n+1) \times (n+1)$ matrix:

$$\boldsymbol{C}^{(n,N)} \stackrel{\text{def}}{=} \begin{pmatrix} \frac{1}{\binom{N}{0}} & \frac{1}{\binom{N}{1}} & \cdots & \frac{1}{\binom{N}{n}} \\ \frac{1}{\binom{N}{1}} & \frac{1}{\binom{N}{2}} & \cdots & \frac{1}{\binom{N}{n+1}} \\ & & \cdots & \\ \frac{1}{\binom{N}{n}} & \frac{1}{\binom{N}{n+1}} & \cdots & \frac{1}{\binom{N}{2n}} \end{pmatrix}$$

We will prove that $\det(\boldsymbol{C}^{(n,N)}) \neq 0$; our claim follows from the special case $N = 2n$. For the base case, $n = 0$, $\det(\boldsymbol{C}^{(0,N)}) = 1$ because $\boldsymbol{C}^{(0,N)}$ is a $1 \times 1$ matrix equal to $1/\binom{N}{0}$, hence $\det(\boldsymbol{C}^{(0,N)}) = 1$. To show the induction step, we will perform elementary column operations (which preserve the determinant) to make the last row of the resulting matrix consist of zeros, except for the last entry.

Consider an arbitrary row $i$, and two adjacent columns $j, j+1$ in that row:

$$\cdots \quad \frac{1}{\binom{N}{i+j}} \quad \frac{1}{\binom{N}{i+j+1}} \quad \cdots$$

We use the fact that $\binom{N}{i+j} = \binom{N}{i+j+1} \frac{i+j+1}{N-i-j}$ and rewrite the two adjacent elements as:

$$\cdots \quad \left( \frac{1}{\binom{N}{i+j+1}} \times \frac{N-i-j}{i+j+1} \right) \quad \frac{1}{\binom{N}{i+j+1}} \quad \cdots$$

Now, for each $j = 0, 1, 2, ..., n-1$, we subtract column $j+1$, multiplied by $\frac{N-(n+j)}{(n+j)+1}$, from column $j$. The last row becomes $0, 0, \ldots, 0, \frac{1}{\binom{N}{2n}}$, which means that $det(C^{(n,N)})$ is equal to $\frac{1}{\binom{N}{2n}}$ times the upper left $(n \times n)$ minor.

Now, we check what happens with element at $(i, j)$. After subtraction, it becomes

$$\frac{1}{\binom{N}{i+j+1}} \times \left( \frac{N-(i+j)}{(i+j)+1} - \frac{N-(n+j)}{(n+j)+1} \right)$$

This expression can be rewritten as:

$$\begin{aligned}
&\frac{1}{\binom{N}{(i+j)+1}} \times \left( \frac{N-(i+j)}{(i+j)+1} - \frac{N-(n+j)}{(n+j)+1} \right) \\
&= \frac{(N-i-j-1)!(i+j+1)!}{N!} \frac{(N+1)(n-i)}{(i+j+1)(n+j+1)} \\
&= \frac{(N-i-j-1)!(i+j)!}{(N-1)!N} \frac{(N+1)(n-i)}{(n+j+1)} \\
&= \frac{1}{\binom{N-1}{(i+j)}} \frac{(N+1)(n-i)}{N(n+j+1)}
\end{aligned}$$

Note that this expression holds with the whole $(n \times n)$ upper-left minor of $\boldsymbol{C}^{(n,N)}$: the element in the lower-right corner of the matrix remains $1/\binom{N}{2n}$. Observe that the $(i, j)$-th entry of this minor is precisely the $(i, j)$-entry of $C^{(n-1,N-1)}$, multiplied by $\frac{(N+1)(n-i)}{N(n+j+1)}$. Here $\frac{N+1}{N}$ is a global constant, $n-i$ is the same constant in the entire row $i$, and $\frac{1}{n+j+1}$ is the same constant in the entire column $j$. We factor out the global constant $\frac{N+1}{N}$, factor out $n-i$ from each row $i$, and factor out $\frac{1}{n+j+1}$ from each column $j$, and obtain the following recurrence:

$$\det(\boldsymbol{C}^{(n,N)}) = \frac{1}{\binom{N}{2n}} \left( \frac{N+1}{N} \right)^n \times \frac{\prod_{i=0}^{n-1}(n-i)}{\prod_{j=0}^{n-1}(n+j+1)} \times \det(\boldsymbol{C}^{(n-1,N-1)})$$

It follows by induction on $n$ that $\det(\boldsymbol{C}^{(n,N)}) \neq 0$.

**Step 3:** Let $\boldsymbol{x}$ be a "good" matrix (Definition 5). Then $V \leq^P \textsc{Shap}$. More precisely, we claim that we can compute the quantity $V$ associated to a matrix $\boldsymbol{x}$ as defined in Eq. (21) in polynomial time, by using an oracle for computing $\textsc{Shap}_{F_1}(X_j)$ over any matrix $\boldsymbol{x}'$.

We modify the matrix $\boldsymbol{x}$ as follows. We add a new attribute $X_0$ whose value is 1 only in the first row, and let $F_1 = X_0$ denote the function that returns the value of feature $X_0$. We show here the new matrix $\boldsymbol{x}'$, augmented with the values of the function $F_1$:

$$\left( \begin{array}{ccccc|c} X_0 & X_1 & X_2 & \dots & X_n & F_1 \\ \hline 1 & x_{11} & x_{12} & \dots & x_{1n} & 1 \\ 0 & x_{21} & x_{22} & \dots & x_{2n} & 0 \\ & \dots & \dots & \dots & \dots & \\ 0 & x_{m1} & x_{m2} & \dots & x_{mn} & 0 \end{array} \right)$$

We run our oracle to compute $\textsc{Shap}_{F_1}(X_0)$ over the matrix $\boldsymbol{x}'$. The value $\textsc{Shap}_{F_1}(X_0)$ is given by Eq. (16), but notice that the matrix $\boldsymbol{x}'$ has $n+1$ columns, while Eq. (16) is given

for a matrix with $n$ columns. Therefore, since $\mathbf{E}[F_1|X_{S\cup\{0\}}] = 1$ for any set $S$, we have:

$$\textsc{Shap}_{F_1}(X_0) = 1 - \sum_{k=0,n} \frac{k!(n-k)!}{(n+1)!}\mathbf{E}[F_1|\mathbf{X}_S = 1]$$

Since $\boldsymbol{x}$ is "good", so is the new matrix $\boldsymbol{x}'$ and, by Claim 6, for any $k = 0, n$

$$\sum_{S:|S|=k} \mathbf{E}[F_1|\mathbf{X}_S = 1] = v_k$$

This implies that we can use the value $\textsc{Shap}_{F_1}(X_0)$ returned by the oracle to compute the quantity:

$$\sum_{k=0,n} \frac{k!(n-k)!}{(n+1)!}\mathbf{E}[F_1|\mathbf{X}_S = 1] = \sum_{k=0,n} \frac{k!(n-k)!}{(n+1)!}v_k = V$$

which completes Step 3

**Putting It Together** Given a PP2CNF formula $\Phi = \bigwedge(U_i \vee V_j)$, and two probability values $p = \Pr(U_1) = \cdots = \Pr(U_m)$ and $q = \Pr(V_1) = \cdots = \Pr(V_n)$, to compute $\mathbf{E}[\Phi]$ we proceed as follows:

- Construct the 0,1-matrix associated to $\Phi$, denote it $\boldsymbol{x}$.
- Construct $m+1$ matrices $\boldsymbol{x}^{(\Gamma)}$, $\Gamma = 1, m+1$, by adding $\Gamma$ rows $(1, 1, \ldots, 1)$ at the beginning of the matrix.
- For each matrix $\boldsymbol{x}^{(\Gamma)}$, construct $n+1$ matrices $\boldsymbol{x}^{(\Gamma,\Delta)}$, $\Delta = 0, n$, by adding $n$ columns, of which the first $\Delta$ columns are 1, the others are 0.
- For each $\boldsymbol{x}^{(\Gamma,\Delta)}$, construct one new matrix $(\boldsymbol{x}^{(\Gamma,\Delta)})'$ by adding a column $(1, 0, 0, \ldots, 0)$. Call this new column $X_0$.
- Use the oracle to compute $\textsc{Shap}_{F_1}(X_0)$. From here, compute the value $V^{(\Gamma,\Delta)}$ associated with the matrix $\boldsymbol{x}^{(\Gamma,\Delta)}$.
- Using the values $V^{(\Gamma,0)}, V^{(\Gamma,1)}, \ldots, V^{(\Gamma,n)}$, compute the values $v_0^{(\Gamma)}, v_1^{(\Gamma)}, \ldots, v_n^{(\Gamma)}$ associated to the matrix $\boldsymbol{x}^{(\Gamma)}$.
- For each $k = 0, n$, use the values $v_k^{(1)}, v_k^{(2)}, \ldots, v_k^{(m+1)}$ to compute the coefficients $a_{0k}, a_{1k}, \ldots, a_{mk}$ associated to the matrix $\boldsymbol{x}$.
- At this point we have all coefficients $a_{\ell k}$ of the polynomial $P(u, v)$.
- Compute the coefficients $b_{\ell k}$ of the polynomial $Q(u, v) = P(1 + u, v)$.
- Finally, return $\mathbf{E}[\Phi] = \frac{p^m q^n}{(1-p)^m(1-q)^n} Q(\frac{1-p}{p}, \frac{1-q}{q})$.

This concludes the entire proof of Lemma 14.

## 6. Perspectives and Conclusions

We establish the complexity of computing the Shap explanation in three important settings. First, we consider fully-factorized data distributions and show that for any prediction model, the complexity of computing the Shap explanation is the same as the complexity of computing the expected value of the model. It follows that there are commonly used models, such as logistic regression, for which computing Shap explanations is intractable. Going beyond fully-factorized distributions, we show that computing Shap explanations is also intractable for simple functions and simple distributions – naive Bayes and empirical distributions.

The recent literature on Shap explanations predominantly studies tradeoffs of variants of the original Shap formulation, and relies on approximation algorithms to compute the explanations. These approximation algorithms, however, tend to make simplifying assumptions which can lead to counter-intuitive explanations, see e.g., Slack et al. (2020). We believe that more focus should be given to the computational complexity of Shap explanations. In particular, which classes of machine learning models can be explained efficiently using the Shap scores? Our results show that, under the assumption of fully-factorized data distributions, there are classes of models for which the Shap explanations can be computed in polynomial time. In future work, we plan to explore if there are classes of models for which the complexity of the Shap explanations is tractable under more complex data distributions, such as the ones defined by tractable probabilistic circuits Vergari et al. (2020) or tractable symmetric probability spaces (Van den Broeck, Meert, and Darwiche, 2014; Beame et al., 2015).

### Acknowledgements

This work is partially supported by NSF grants IIS-1907997, IIS-1954222 IIS-1943641, IIS-1956441, CCF-1837129, DARPA grant N66001-17-2-4032, a Sloan Fellowship, and gifts by Intel and Facebook research. Schleich is supported by a RelationalAI fellowship. The authors would like to thank YooJung Choi for valuable discussions on the proof of Theorem 6.

## References

Aas, K.; Jullum, M.; and Løland, A. 2019. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *arXiv preprint arXiv:1903.10464* .

Arenas, M.; Barceló, P.; Bertossi, L.; and Monet, M. 2020. The Tractability of SHAP-Score-Based Explanations over Deterministic and Decomposable Boolean Circuits. *arXiv preprint arXiv:2007.14045* .

Beame, P.; Van den Broeck, G.; Gribkoff, E.; and Suciu, D. 2015. Symmetric Weighted First-Order Model Counting. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, 313–328.

Bertossi, L.; Li, J.; Schleich, M.; Suciu, D.; and Vagena, Z. 2020. Causality-Based Explanation of Classification Outcomes. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, DEEM'20. New York, NY, USA: Association for Computing Machinery.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on* 100(8): 677–691.

Chavira, M.; and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7): 772–799.

Darwiche, A.; and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17: 229–264.

Datta, A.; Sen, S.; and Zick, Y. 2016. Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 598–617.

Elkind, E.; Goldberg, L. A.; Goldberg, P. W.; and Wooldridge, M. J. 2008. A tractable and expressive class of marginal contribution nets and its applications. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 2*, 1007–1014.

Ferrara, A.; Pan, G.; and Vardi, M. Y. 2005. Treewidth in verification: Local vs. global. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, 489–503. Springer.

Gade, K.; Geyik, S. C.; Kenthapadi, K.; Mithal, V.; and Taly, A. 2019. Explainable AI in Industry. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 3203–3204. New York, NY, USA: Association for Computing Machinery.

Janzing, D.; Minorics, L.; and Bloebaum, P. 2020. Feature relevance quantification in explainable AI: A causal problem. volume 108 of *Proceedings of Machine Learning Research*, 2907–2916. PMLR.

Khosravi, P.; Choi, Y.; Liang, Y.; Vergari, A.; and Van den Broeck, G. 2019a. On Tractable Computation of Expected Predictions. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*.

Khosravi, P.; Liang, Y.; Choi, Y.; and den Broeck, G. V. 2019b. What to Expect of Classifiers? Reasoning about Logistic Regression with Missing Features. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 2716–2724.

Khosravi, P.; Vergari, A.; Choi, Y.; Liang, Y.; and Van den Broeck, G. 2020. Handling Missing Data in Decision Trees: A Probabilistic Approach. In *The Art of Learning with Missing Values Workshop at ICML (Artemiss)*.

Kumar, I. E.; Venkatasubramanian, S.; Scheidegger, C.; and Friedler, S. 2020. Problems with Shapley-value-based explanations as feature importance measures. In *Proceedings of the 37th International Conference on Machine Learning*, Vienna, Austria, PMLR 119, 2020.

Liang, Y.; and Van den Broeck, G. 2019. Learning Logistic Circuits. In *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*.

Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; and Lee, S. 2020. From Local Explanations to Global Understanding with Explainable AI for Trees. *Nature Machine Intelligence* 2: 56–67.

Lundberg, S. M.; Erion, G. G.; and Lee, S.-I. 2018. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888* .

Lundberg, S. M.; and Lee, S. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in neural information processing systems (NIPS)*, 4765–4774.

Merrick, L.; and Taly, A. 2020. The Explanation Game: Explaining Machine Learning Models Using Shapley Values. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, 17–38. Springer.

Ng, A. Y.; and Jordan, M. I. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, 841–848.

Provan, J. S.; and Ball, M. O. 1983. The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected. *SIAM J. Comput.* 12(4): 777–788.

Rendle, S. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*, 995–1000. IEEE.

Roth, A. e. 1988. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge Univ. Press.

Sang, T.; Beame, P.; and Kautz, H. A. 2005. Performing Bayesian inference by weighted model counting. In *AAAI*, volume 5, 475–481.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*.

Štrumbelj, E.; and Kononenko, I. 2014. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems* 41(3): 647–665.

Sundararajan, M.; and Najmi, A. 2020. The many Shapley values for model explanation. In *Proceedings of the 37th International Conference on Machine Learning*, Vienna, Austria, PMLR 119, 2020.

Van den Broeck, G.; Meert, W.; and Darwiche, A. 2014. Skolemization for Weighted First-Order Model Counting. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*.

Vergari, A.; Choi, Y.; Peharz, R.; and Van den Broeck, G. 2020. Probabilistic Circuits: Representations, Inference, Learning and Applications. AAAI Tutorial.

Wei, W.; and Selman, B. 2005. A new approach to model counting. In *International Conference on Theory and Applications of Satisfiability Testing*, 324–339. Springer.