

Algorithmic fairness verification with graphical models

Bishwamittra Ghosh, Debabrota Basu, Kuldeep S. Meel

▶ **To cite this version:**

Bishwamittra Ghosh, Debabrota Basu, Kuldeep S. Meel. Algorithmic fairness verification with graphical models. AAAI-2022 - 36th AAAI Conference on Artificial Intelligence, Feb 2022, Virtual, United States. hal-03770361

HAL Id: hal-03770361
https://hal.science/hal-03770361

Submitted on 6 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial| 4.0 International License

arXiv:2109.09447v2 [cs.LG] 1 Jun 2022

Algorithmic Fairness Verification with Graphical Models*

Bishwamittra Ghosh
School of Computing
National University of Singapore
Singapore

Debabrota Basu
Équipe Scool, Univ. Lille, Inria, UMR 9189 - CRIStAL, CNRS
Centrale Lille, France

Kuldeep S. Meel
School of Computing
National University of Singapore
Singapore

Abstract

In recent years, machine learning (ML) algorithms have been deployed in safety-critical and high-stake decision-making, where the *fairness* of algorithms is of paramount importance. Fairness in ML centers on detecting bias towards certain demographic populations induced by an ML classifier and proposes algorithmic solutions to mitigate the bias with respect to different fairness definitions. To this end, several *fairness verifiers* have been proposed that compute the bias in the prediction of an ML classifier—essentially beyond a finite dataset—given the probability distribution of input features. In the context of verifying *linear classifiers*, existing fairness verifiers are limited by *accuracy* due to imprecise modeling of correlations among features and *scalability* due to restrictive formulations of the classifiers as SSAT/SMT formulas or by sampling.

In this paper, we propose an efficient fairness verifier, called FVGM, that encodes the correlations among features as a Bayesian network. In contrast to existing verifiers, FVGM proposes a *stochastic subset-sum* based approach for verifying linear classifiers. Experimentally, we show that FVGM leads to an *accurate* and *scalable* assessment for more diverse families of fairness-enhancing algorithms, fairness attacks, and group/causal fairness metrics than the state-of-the-art fairness verifiers. We also demonstrate that FVGM facilitates the computation of fairness influence functions as a stepping stone to detect the source of bias induced by subsets of features.

1. Introduction

The significant improvement of machine learning (ML) over the decades has led to a host of applications of ML in high-stake decision-making such as college admission (Martinez Neda et al., 2021), hiring of employees (Ajunwa et al., 2016), and recidivism prediction (Tollenaar and Van der Heijden, 2013; Dressel and Farid, 2018). ML algorithms often have an accuracy-centric learning objective, which may cause them to be biased towards certain part of the dataset belonging to a certain economically or socially sensitive groups (Landy et al., 1978; Zliobaite, 2015; Berk, 2019). The following example illustrates a plausible case of unfairness induced by ML algorithms.

*. Source code: https://github.com/meelgroup/justicia

Example 1 *Following (Ghosh et al., 2021, Example 1.), let us consider an ML problem where the classifier decides the eligibility of an individual for health insurance given their income and fitness (Figure 1). Here, the sensitive feature 'age' (A) follows a Bernoulli distribution, and income (I) and fitness (F) follow Gaussian distributions. We generate 1000 samples from these distributions and use them to train a Support Vector Machine (SVM) classifier. The decision boundary of this classifier is* $9.37I + 9.75F - 0.34A \geq 9.4$*, where* $A = 1$ *denotes the sensitive group 'age* $\geq$ *40'. This classifier selects an individual above and below* 40 *years of age with probabilities* 0.24 *and* 0.86*, respectively. This illustrates a disparate treatment of individuals of two age groups by the SVM classifier.*

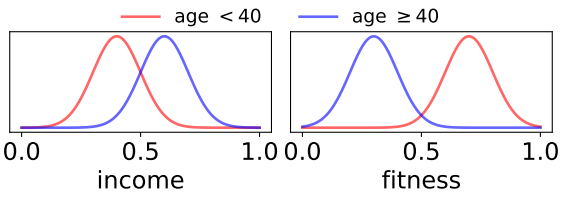


Figure 1: Age-dependent distributions of income and fitness in Example 1.

In order to identify and mitigate the bias of ML classifiers, different fairness definitions and fairness algorithms have been proposed (Hardt et al., 2016; Kusner et al., 2017; Mehrabi et al., 2019). In this paper, we focus on two families of fairness definitions: *group* and *causal* fairness. Group fairness metrics, such as disparate impact and equalized odds constrain the probability of the positive prediction of the classifier to be (almost) equal among different sensitive groups (Dwork et al., 2012; Feldman et al., 2015). On the other hand, causal fairness metrics assess the difference in positive predictions if every feature in the causal relation remains identical except the sensitive feature (Nabi and Shpitser, 2018; Zhang and Bareinboim, 2018). The early works on *fairness verification* focused on measuring fairness metrics of a classifier for a given dataset (Bellamy et al., 2018). Naturally, such techniques were limited in enhancing confidence of users for wide deployment. Consequently, recent verifiers seek to achieve verification beyond finite dataset and in turn focus on the probability distribution of features (Albarghouthi et al., 2017; Bastani et al., 2019; Ghosh et al., 2021). More specifically, the input to the verifier is a classifier and the probability distribution of features, and the output is an estimate of fairness metrics that the classifier obtains given the distribution. For Example 1, a fairness verifier takes the SVM classifier and the distribution of features I, F, A as an input and outputs the probability of positive prediction of the classifier for different sensitive groups.

In order to solve the fairness-verification problem, existing works have proposed two principled approaches. Firstly, Ghosh et al. (2021) and Albarghouthi et al. (2017) propose formal methods that reduce the problem into a solution of an SSAT or an SMT formula respectively. Secondly, Bastani et al. (2019) propose a sampling approach that relies on extensively enumerating the conditional probabilities of prediction given different sensitive features and thus, incurs high computational cost. Additionally, existing works assume feature independence of non-sensitive features and consider correlated features within a limited scope, such as conditional probabilities of non-sensitive features w.r.t. sensitive

features and ignore correlations among non-sensitive features. As a result, the *scalability* and *accuracy* of existing verifiers remains a major challenge.

In this work, we seek to remedy the aforementioned situation. As a first step, we focus on *linear classifiers*, which has attracted significant attention from researchers in the context of fair algorithms (Pleiss et al., 2017; Zafar et al., 2017; Dressel and Farid, 2018; John et al., 2020). At this point, it is worth highlighting that our empirical evaluation demonstrates that the existing techniques fail to scale beyond small examples or provide highly inaccurate estimates for comparatively *small* linear classifiers.

Our Contributions. In this paper, we propose a fairness verification framework, namely FVGM (**F**airness **V**erification with **G**raphical **M**odels), for accurately and efficiently verifying linear classifiers. FVGM proposes a novel *stochastic subset-sum* encoding for linear classifiers with an efficient pseudo-polynomial solution using dynamic programming. To address feature-correlations, FVGM considers a graphical model, particularly a Bayesian Network that represents conditional dependence (and independence) among features in the form of a Directed Acyclic Graph (DAG). Experimentally, FVGM is more accurate and scalable than existing fairness verifiers; FVGM can verify group and causal fairness metrics for multiple fairness algorithms. We also demonstrate two novel applications of FVGM as a fairness verifier: (a) detecting fairness attacks, and (b) computing Fairness Influence Functions (FIF) of features as a mean of identifying (un)fairness contribution of a subset of features.

2. Background

In this section, we define different fairness metrics proposed for classification. Following that, we state basics of stochastic subset sum and Bayesian networks that are the main components of our methodology.

Fairness in ML. We consider[1] a dataset $\mathbf{D}$ as a collection of triples $(\mathbf{X}, \mathbf{A}, Y)$ generated from an underlying distribution $\mathcal{D}$. $\mathbf{X} \triangleq \{X_1, \ldots, X_{m_1}\}$ are non-sensitive features whereas $\mathbf{A} \triangleq \{A_1, \ldots, A_{m_2}\}$ are categorical sensitive features. $Y \in \{0, 1\}$ is the binary label (or class) of $(\mathbf{X}, \mathbf{A})$. Each non-sensitive feature X_i is sampled from a continuous probability distribution $\mathcal{X}_i$, and each sensitive feature $A_j \in \{0, \ldots, N_j\}$ is sampled from a discrete probability distribution $\mathcal{A}_j$. We use $(\mathbf{x}, \mathbf{a})$ to denote the feature-values of $(\mathbf{X}, \mathbf{A})$. For sensitive features, a valuation vector $\mathbf{a} = [a_1, .., a_{m_2}]$ is called a *compound sensitive group*. For example, consider $\mathbf{A}$ = {race, sex} where race $\in$ {Asian, Color, White} and sex $\in$ {female, male}. Thus $\mathbf{a}$ = [Asian, female] is a compound sensitive group. We represent a binary classifier trained on the dataset $\mathbf{D}$ as $\mathcal{M} : (\mathbf{X}, \mathbf{A}) \rightarrow \hat{Y}$. Here, $\hat{Y} \in \{0, 1\}$ is the predicted class of $(\mathbf{X}, \mathbf{A})$.

We now discuss different fairness metrics in the literature based on the prediction of a classifier (Feldman et al., 2015; Hardt et al., 2016; Nabi and Shpitser, 2018). In this paper, FVGM verifies two families of fairness metrics: group fairness (first three in the following) and path-specific causal fairness.

1. We represent sets/vectors by bold letters, and the corresponding distributions by calligraphic letters. We express random variables in uppercase, and an assignment of a random variable in lowercase.

1. *Disparate Impact* (DI): A classifier satisfies $(1-\epsilon)$-disparate impact if for $\epsilon \in [0,1]$, $\min_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}] \geq (1-\epsilon)\max_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}]$.

2. *Statistical Parity* (SP): A classifier satisfies ϵ-statistical parity if for $\epsilon \in [0,1]$, $\max_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}] - \min_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}] \leq \epsilon$.

3. *Equalized Odds* (EO): A classifier satisfies ϵ-equalized odds if for $\epsilon \in [0,1]$, $\max_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}, Y=0] - \min_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}, Y=0] \leq \epsilon$, and $\max_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}, Y=1] - \min_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}, Y=1] \leq \epsilon$.

4. *Path-specific Causal Fairness* (PCF): Let $\mathbf{a}_{\max} \triangleq \arg\max_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}]$. We consider mediator features $\mathbf{Z} \subseteq \mathbf{X}$ sampled from the conditional distribution $\mathcal{Z}_{|\mathbf{A}=\mathbf{a}_{\max}}$. This emulates the fact that mediator variables have the same sensitive features $\mathbf{a}_{\max}$. For $\epsilon \in [0,1]$, path-specific causal fairness is defined as $\max_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}, \mathbf{Z}] - \min_{\mathbf{a}} \Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}, \mathbf{Z}] \leq \epsilon$.

For all of the above metrics, lower value of ϵ indicates higher fairness demonstrated by the classifier $\mathcal{M}$. Following the observation of (Ghosh et al., 2021), computing all of the aforementioned fairness metrics is equivalent to computing the maximum and minimum of the *probability of positive prediction* of the classifier, denoted as $\Pr[\hat{Y}=1|\mathbf{A}=\mathbf{a}]$, for all compound sensitive groups $\mathbf{a}$ from $\mathbf{A}$. Thus, in Section 3, we focus on computing the maximum and minimum probability of positive prediction of the classifier and then extend it to assess corresponding fairness metrics. We call the group for which the probability of positive prediction is maximum (minimum) as the *most (least) favored group* of the classifier.

Stochastic Subset Sum Problem (S3P). Let $\mathbf{B} \triangleq \{B_i\}_{i=1}^{|\mathbf{B}|}$ be a set of Boolean variables and $w_i \in \mathbb{Z}$ be the weight of B_i. Given a constraint of the form $\sum_{i=1}^{|\mathbf{B}|} w_i B_i = \tau$, for a constant threshold $\tau \in \mathbb{Z}$, the subset-sum problem seeks to compute an assignment $\mathbf{b} \in \{0,1\}^{|\mathbf{B}|}$ such that the constraint evaluates to true when $\mathbf{B}$ is substituted with $\mathbf{b}$. Subset sum problem is known to be a NP-complete problem and well-studied in theoretical computer science (Kleinberg and Tardos, 2006). The *counting* version of the subset-sum problem counts all $\mathbf{b}$'s for which the above constraint holds; and this problem belongs to the complexity class #P. In this paper, we consider the counting problem for the constraint $\sum_{i=1}^{|\mathbf{B}|} w_i B_i \geq \tau$ where variables B_i's are stochastic. More precisely, we define a *stochastic subset-sum* problem, namely S3P, that computes $\Pr[\sum_{i=1}^{|\mathbf{B}|} w_i B_i \geq \tau]$. Details of S3P are in Section 3.1.

Bayesian Network. In general, a Probabilistic Graphical Model (Koller and Friedman, 2009), and specifically a *Bayesian network* (Pearl, 1985; Chavira and Darwiche, 2008), encodes the dependencies and conditional independence between a set of random variables. In this paper, we leverage an access to a Bayesian network on $\mathbf{X} \cup \mathbf{A}$ that represents the joint distribution on them. A Bayesian network is denoted by a pair (G, θ), where $G \triangleq (\mathbf{V}, \mathbf{E})$ is a DAG (Directed Acyclic Graph), and θ is a set of parameters encoding the conditional probabilities induced by the joint distribution under investigation. Each vertex $V_i \in \mathbf{V}$ corresponds to a random variable. Edges $\mathbf{E} \in \mathbf{V} \times \mathbf{V}$ imply conditional dependencies among variables. For each variable $V_i \in \mathbf{V}$, let $\mathrm{Pa}(V_i) \subseteq \mathbf{V} \setminus \{V_i\}$ denote the set of parents of V_i. Given $\mathrm{Pa}(V_i)$ and parameters θ, V_i is independent of its other non-descendant variables in

G. Thus, for the assignment v_i of V_i and $\mathbf{u}$ of $\text{Pa}(V_i)$, the aforementioned semantics of a Bayesian network encodes the joint distribution of $\mathbf{V}$ as:

$$\Pr[V_1 = v_1, \ldots, V_{|\mathbf{V}|} = v_{|\mathbf{V}|}] = \prod_{i=1}^{|\mathbf{V}|} \Pr[V_i = v_i | \text{Pa}(V_i) = \mathbf{u}; \theta]. \tag{1}$$

3. FVGM: Fairness Verification with Graphical Models

In this section, we present FVGM, a fairness verification framework for linear classifiers that accounts for correlated features represented as a graphical model. The core idea of verifying fairness of a classifier is to compute the probability of positive prediction of the classifier with respect to all compound sensitive groups. To this end, FVGM solves a stochastic subset sum problem, S3P, that is equivalent to computing the probability of positive prediction of the classifier for the most (and the least) favored sensitive group. In this section, we first define S3P and present an efficient dynamic programming solution for S3P. We then extend S3P to consider correlated features as input. Finally, we conclude by discussing fairness verification based on the solution of S3P.

Problem Formulation. Given a linear classifier $\mathcal{M} : (\mathbf{X}, \mathbf{A}) \to \hat{Y}$ and a probability distribution $\mathcal{D}$ of $\mathbf{X} \cup \mathbf{A}$, our objective is to compute $\max_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$ and $\min_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$ with respect to $\mathcal{D}$. In this study, we express a linear classifier $\mathcal{M}$ as

$$\hat{Y} = \mathbb{1}\Big[\sum_i w_{X_i} X_i + \sum_j w_{A_j} A_j \geq \tau\Big].$$

Here, w denotes the weight (or coefficients) of a feature, τ denotes the bias or the offset parameter of the classifier, and $\mathbb{1}$ is an indicator function. Hence, the prediction $\hat{Y} = 1$ if and only if the inner inequality holds. Thus, computing the maximum (resp. minimum) probability of positive prediction is equivalent to finding out the assignment of A_j's for which the probability of satisfying the inner inequality is highest (resp. lowest). We reduce this computation into an instance of S3P. To perform this reduction, we assume weights w and bias τ as integers, and features $\mathbf{X} \cup \mathbf{A}$ as Boolean. In Sec. 3.5, we relax these assumptions and extend to the practical settings.

3.1 S3P: Stochastic Subset Sum Problem

Now, we formally describe the specification and semantics of S3P. S3P operates on a set of Boolean variables $\mathbf{B} = \{B_i\}_{i=1}^n \in \{0,1\}^n$, where $w_i \in \mathbb{Z}$ is the weight of B_i, and $n \triangleq |\mathbf{B}|$. Given a constant threshold $\tau \in \mathbb{Z}$, S3P computes the *probability* of a subset of $\mathbf{B}$ with sum of weights of non-zero variables to be at least τ. Formally,

$$S(\mathbf{B}, \tau) \triangleq \Pr\Big[\sum_i w_i B_i \geq \tau\Big] \in [0, 1].$$

Aligning with terminologies in stochastic satisfiability (SSAT) (Littman et al., 2001), we categorize the variables $\mathbf{B}$ into two types: (i) *chance variables* that are stochastic and have an associated probability of being assigned to 1 and (ii) *choice variables* that we optimize

while computing $S(\mathbf{B}, \tau)$. To specify the category of variables, we consider a *quantifier* $q_i \in \{\mathfrak{R}^{p_i}, \exists, \forall\}$ for each B_i. Elaborately, $\mathfrak{R}^p$ is a *random quantifier* corresponding to a chance variable $B \in \mathbf{B}$, where $p \triangleq \Pr[B = 1]$. In contrast, $\exists$ is an *existential quantifier* corresponding to a choice variable B such that a Boolean assignment of B *maximizes* $S(\mathbf{B}, \tau)$. Finally, $\forall$ is an *universal quantifier* for a choice variable B that fixes an assignment to B that *minimizes* $S(\mathbf{B}, \tau)$.

Now, we formally present the semantics of $S(\mathbf{B}, \tau)$ provided that each variable B_i has weight w_i and quantifier q_i. Let $\mathbf{B}[2:n] \triangleq \{B_j\}_{j=2}^{n}$ be the subset of $\mathbf{B}$ without the first variable B_1. Then $S(\mathbf{B}, \tau)$ is recursively defined as:

$$S(\mathbf{B}, \tau) = \begin{cases} \mathbb{1}[\tau \leq 0], \text{ if } \mathbf{B} = \emptyset \\ S(\mathbf{B}[2:n], \tau - \max\{w_1, 0\}), \text{ if } q_1 = \exists \\ S(\mathbf{B}[2:n], \tau - \min\{w_1, 0\}), \text{ if } q_1 = \forall \\ p_1 \times S(\mathbf{B}[2:n], \tau - w_1) + \\ (1 - p_1) \times S(\mathbf{B}[2:n], \tau), \text{ if } q_1 = \mathfrak{R}^{p_1} \end{cases} \tag{2}$$

Observe that when $\mathbf{B}$ is empty, S is computed as 1 if $\tau \leq 0$, and $S = 0$ otherwise. For existential and universal quantifiers, we compute S based on the weight. Specifically, if $q_1 = \exists$, we decrement the threshold τ by the maximum between w_1 and 0. For example, if $w_1 > 0$, B_1 is assigned 1, and assigned 0 otherwise. Therefore, by solving for an existential variable, we maximize S. In contrast, when if $q_1 = \forall$, we fix an assignment of B_1 that minimizes S by choosing between the minimum of w_1 and 0. Finally, for random quantifiers, we decompose the computation of S into two sub-problems: one sub-problem where $B_1 = 1$ and the updated threshold becomes $\tau - w_1$ and another sub-problem where $B_1 = 0$ and the updated threshold remains the same. Herein, we compute S as the expected output of both sub-problems.

Remark 1 *$S(\mathbf{B}, \tau)$ does not depend on the order of $\mathbf{B}$.*

Computing Minimum and Maximum probability of positive prediction of Linear Classifiers Using S3P. For computing $\max_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$ of a linear classifier, we set existential quantifiers $\exists$ to sensitive features A_j, randomized quantifiers $\mathfrak{R}$ to non-sensitive features X_i and construct a set $\mathbf{B} = \mathbf{A} \cup \mathbf{X}$. The coefficients w_{A_j} and w_{X_i} of the classifier become weights of $\mathbf{B}$. Also, we get $n = m_1 + m_2$. For non-sensitive variables X_i, which are chance variables, we derive their marginal probability $p_i = \Pr[X_i = 1]$ from the distribution $\mathcal{D}$. According to semantic of S3P, setting $\exists$ quantifiers on $\mathbf{A}$ computes the maximum value of $S(\mathbf{B}, \tau)$ that equalizes the maximum probability of positive prediction of the classifier. In this case, the *inferred* assignment of $\mathbf{A}$ implies the most favored group $\mathbf{a}_{\max} = \arg\max_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$. In contrast, to compute the minimum probability of positive prediction, we instead assign variable A_j a universal quantifier while keeping random quantifiers over X_i, and infer the least favored group $\mathbf{a}_{\min} = \arg\min_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$.

3.2 A Dynamic Programming Solution for S3P

We propose a dynamic programming approach (Pisinger, 1999; Woeginger and Yu, 1992) to solve S3P as the problem has overlapping sub-problem properties. For example, $S(\mathbf{B}, \tau)$

can be solved by solving $S(\mathbf{B}[2:n], \tau')$, where the updated threshold τ', called the *residual threshold*, depends on the original threshold τ and the assignment of B_1 as shown in Eq. (2). Building on this observation, we propose the recursion and terminating condition leading to our dynamic programming algorithm.

Recursion. We consider a function $\mathsf{dp}(i, \tau)$ that solves the sub-problem $S(\mathbf{B}[i:n], \tau)$, for $i \in \{1, \ldots, n\}$. The semantics of $S(\mathbf{B}, \tau)$ in Eq. (2) induces the recursive definition of $\mathsf{dp}(i, \tau)$ as:

$$\mathsf{dp}(i,\tau) = \begin{cases} \mathsf{dp}(i+1, \tau - \max\{w_i, 0\}), \text{ if } q_i = \exists \\ \mathsf{dp}(i+1, \tau - \min\{w_i, 0\}), \text{ if } q_i = \forall \\ p_i \times \mathsf{dp}(i+1, \tau - w_i) + \\ (1-p_i) \times \mathsf{dp}(i+1, \tau), \text{ if } q_i = \mathfrak{R}^{p_i} \end{cases} \tag{3}$$

Eq. (3) shows that $S(\mathbf{B}, \tau)$ can be solved by instantiating $\mathbf{dp}(1, \tau)$, which includes all the variables in $\mathbf{B}$.

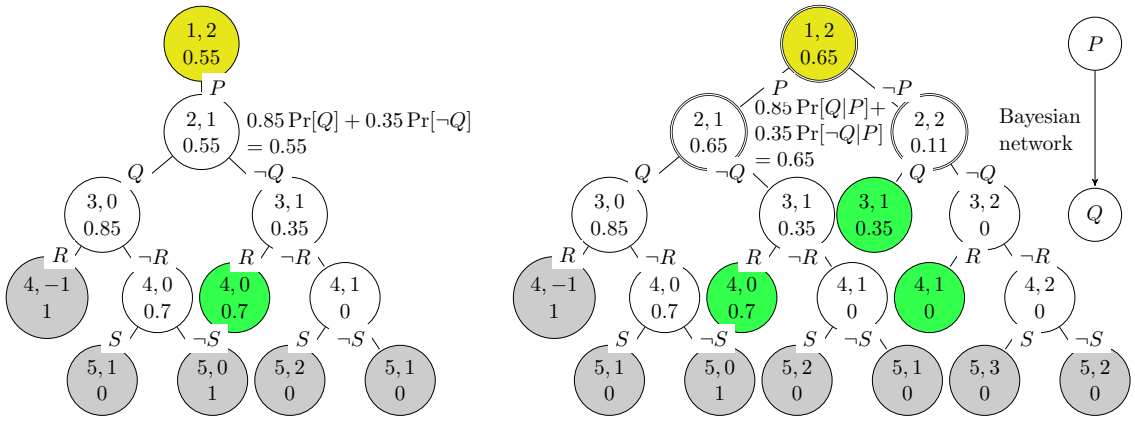
Terminating Condition. Let w_{neg}, w_{pos}, and w_{all} be the sum of negative, positive, and all weights of $\mathbf{B}$, respectively. We observe that $w_{neg} \leq w_{all} \leq w_{pos}$. Thus, for any i, if the residual threshold $\tau \leq w_{neg}$, there is always a subset of $\mathbf{B}[i:n]$ with sum of weights at least τ. Conversely, when $\tau > w_{pos}$, there is no subset of $\mathbf{B}[i:n]$ with sum of weights at least τ. We leverage this bound and tighten the terminating conditions of $\mathsf{dp}(i, \tau)$ in Eq. (4).

$$\mathsf{dp}(i,\tau) = \begin{cases} 1 \text{ if } \tau \leq w_{neg} \\ 0 \text{ if } \tau > w_{pos} \\ \mathbb{1}[\tau \leq 0] \text{ if } i = n+1 \end{cases} \tag{4}$$

Eq. (3) and (4) together define our dynamic programming algorithm. While deploying the algorithm, we store $\mathsf{dp}(i, \tau)$ in memory to avoid repetitive computations. This allows us to achieve a pseudo-polynomial algorithm (Lemma 2) instead of a naïve exponential algorithm enumerating all possible assignments. In particular, the time complexity is pseudo-polynomial for chance (random) variables and linear for choice (existential and universal) variables.

Lemma 2 *Let n' be the number of existential and universal variables in $\mathbf{B}$. Let $w_\exists = \sum_{B_i \in \mathbf{B} | q_i = \exists} \max\{w_i, 0\}$ and $w_\forall = \sum_{B_i \in \mathbf{B} | q_i = \forall} \min\{w_i, 0\}$ be the considered sum of weights of existential and universal variables, respectively. We can exactly solve* S3P *using dynamic programming with time complexity $\mathcal{O}((n-n')(\tau + |w_{neg}| - w_\exists - w_\forall) + n')$. The space complexity is $\mathcal{O}((n-n')(\tau + |w_{neg}| - w_\exists - w_\forall))$.*

A Heuristic for Faster Computation. We propose two improvements for a faster computation of the dynamic programming solution. Firstly, we observe that in Eq. (3), existential/universal variables are deterministically assigned based on their weights. Hence, we reorder $\mathbf{B}$ such that existential/universal variables appear earlier in $\mathbf{B}$ than random variables. This allows us to avoid unnecessary repeated exploration of existential/universal variables in dp. Moreover, according to the remark in Section 1, reordering $\mathbf{B}$ still produces



(a) Known marginal probabilities.

(b) Probabilities computed with a Bayesian network.

Figure 2: Search tree representation of S3P for computing the maximum probability of positive prediction of the classifier on variables $\mathbf{B} = \{P, Q, R, S\}$ with weights $\{1, 1, 1, -1\}$ and threshold $\tau = 2$. Each node is labeled by (i, τ'), where i is the index of $\mathbf{B}$ and τ' is the residual threshold. The tree is explored using Depth-First Search (DFS) starting with left child. Within a node, the value in the bottom denotes $\mathsf{dp}(i, \tau')$ that is solved recursively based on sub-problems $\mathsf{dp}(i+1, \cdot)$ in child nodes. Yellow nodes denote *existential* variables and all other nodes are *random* variables. Additionally, a green node denotes a collision, in which case a previously computed dp solution is returned. Leaf nodes (gray) are computed based on terminating conditions in Eq. 4. In Figure 2b, nodes with double circles, such as $\{(1, 2), (2, 1), (2, 2)\}$, are enumerated exponentially to compute conditional probabilities from the Bayesian network.

the same exact solution of S3P. Secondly, to reach the terminating condition of $\mathsf{dp}(i, \tau)$ more frequently, we sort $\mathbf{B}$ based on their weights—more specifically, within each cluster of random, existential, and universal variables. In particular, if $\tau \leq 0.5(w_{pos} - w_{neg})$, τ is closer to w_{pos} than w_{neg}. Hence, we sort each cluster in descending order of weights. Otherwise, we sort in ascending order. We illustrate our dynamic programming approach in Example 2.

Example 2 *We consider a linear classifier $P + Q + R - S \geq 2$. Herein, P is a Boolean sensitive feature, and Q, R, S are Boolean non-sensitive features with $\Pr[Q] = 0.4, \Pr[R] = 0.5$, and $\Pr[S] = 0.3$. To compute the maximum probability of positive prediction of the classifier, we impose an existential quantifier on P and randomized quantifiers on others. This leads us to the observation that $P = 1$ is the optimal assignment as $w_P = 1 > 0$. We now require to compute $\Pr[Q + R - S \geq 1]$, which by dynamic programming, is computed as 0.55. The solution is visualized as a search tree in Figure 2a, where we observe that storing the solution of sub-problems in the memory avoids repetitive computation, such as exploring the node $(4, 0)$. Similarly, the minimum probability of positive prediction of the classifier is 0.14 (not shown in Figure 2a) where we impose a universal quantifier on P to obtain $P = 0$ as the optimal assignment.*

3.3 S3P with Correlated Variables

In S3P presented in Section 3.1, we consider all Boolean variables to be probabilistically independent. This independence assumption often leads to an *inaccurate estimate* of the probability of positive prediction of the classifier because both sensitive and non-sensitive features can be correlated in practical fairness problems. Therefore, we extend S3P to include correlations among variables.

We consider a Bayesian network $\mathsf{BN} = (G, \theta)$ to represent correlated variables, where $G \triangleq (\mathbf{V}, \mathbf{E})$, $\mathbf{V} \subseteq \mathbf{B}$, $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$, and θ is the parameter of the network. In BN, we constrain that there is no conditional probability of choice (i.e., existential and universal) variables as we optimize their assignment in S3P. Choice variables, however, can impose conditions on chance (i.e., random) variables. In practice, we achieve this by allowing no incoming edge on choice variables while learning BN (ref. Section 4).

For a chance variable $B_i \in \mathbf{V}$, let $\mathrm{Pa}(B_i)$ denote its parents. According to Eq. (1), for an assignment $\mathbf{u}$ of $\mathrm{Pa}(B_i)$, BN ensures B_i to be independent of other non-descendant variables in $\mathbf{V}$. Hence, in the recursion of Eq. (3), we substitute p_i with $\Pr[B_i = 1|\mathrm{Pa}(B_i) = \mathbf{u}]$. In order to explicate the dependence on $\mathbf{u}$, we denote the expected solution of $S(\mathbf{B}[i:n], \tau)$ as $\mathsf{dp}(i, \tau, \mathbf{u})$, which for $B_i \in \mathbf{V}$ is modified as follows:

$$\begin{aligned}\mathsf{dp}(i,\tau,\mathbf{u}) = \Pr[B_i = 0|\mathrm{Pa}(B_i) = \mathbf{u}]\mathsf{dp}(i+1,\tau,\mathbf{u}\cup\{0\}) \\ + \Pr[B_i = 1|\mathrm{Pa}(B_i) = \mathbf{u}]\mathsf{dp}(i+1,\tau-w_i,\mathbf{u}\cup\{1\}).\end{aligned}$$

Since $\mathsf{dp}(i, \tau, \mathbf{u})$ involves $\mathbf{u}$, we initially perform a topological sort of $\mathbf{V}$ to know the assignment of parents before computing dp on the child. Moreover, there are $2^{|\mathrm{Pa}(B_i)|}$ assignments of $\mathrm{Pa}(B_i)$, and we compute $\mathsf{dp}(i, \tau, \mathbf{u})$ for $\mathbf{u} \in \{0, 1\}^{|\mathrm{Pa}(B_i)|}$ to incorporate all conditional probabilities into S3P. For this enumeration, we do not store $\mathsf{dp}(i, \tau, \mathbf{u})$ in memory. However, for $B_i \notin \mathbf{V}$ that does not appear in the network, we instead compute $\mathsf{dp}(i, \tau)$ and store it in memory as in Section 3.2, because B_i is not correlated with other variables. Lemma 3 presents the complexity of solving S3P with correlated variables, wherein unlike Lemma 2, the complexity differentiates based on variables in $\mathbf{V}$ (exponential) and $\mathbf{B} \setminus \mathbf{V}$ (pseudo-polynomial).

Lemma 3 *Let $\mathbf{V} \subseteq \mathbf{B}$ be the set of vertices in the Bayesian network and n'' be the number of existential and universal variables in $\mathbf{B} \setminus \mathbf{V}$. Let $w'_\exists = \sum_{B_i \in \mathbf{B}\setminus\mathbf{V}|q_i=\exists} \max\{w_i, 0\}$ and $w'_\forall = \sum_{B_i \in \mathbf{B}\setminus\mathbf{V}|q_i=\forall} \min\{w_i, 0\}$ be the sum of considered weights of existential and universal variables, respectively that only appear in $\mathbf{B} \setminus \mathbf{V}$. To exactly compute S3P with correlated variables in the dynamic programming approach, time complexity is $\mathcal{O}(2^{|\mathbf{V}|}+(n-n''-|\mathbf{V}|)(\tau+|w_{neg}|-w'_\exists-w'_\forall)+n'')$ and space complexity is $\mathcal{O}((n-n''-|\mathbf{V}|)(\tau+|w_{neg}|-w'_\exists-w'_\forall))$.*

A Heuristic for Faster Computation. We observe that to encode conditional probabilities, we enumerate all assignments of variables in $\mathbf{V}$. For computing the probability of positive prediction of a linear classifier with correlated features, we consider a heuristic to sort variables in $\mathbf{B} = \mathbf{A} \cup \mathbf{X}$. Let $\mathbf{V} \subseteq \mathbf{B}$ be the set of vertices in the network and $\mathbf{V}^c = \mathbf{B} \setminus \mathbf{V}$. In this heuristic, we sort sensitive variables $\mathbf{A}$ by positioning $\mathbf{A} \cap \mathbf{V}$ in the beginning followed by $\mathbf{A} \cap \mathbf{V}^c$. Then we order the variables $\mathbf{B}$ such that variables in $\mathbf{A}$

precedes those in $\mathbf{X} \cap \mathbf{V}$, and the variables in $\mathbf{X} \cap \mathbf{V}^c$ follows the ones in $\mathbf{X} \cap \mathbf{V}$. This sorting allows to avoid repetitive enumeration of variables in $\mathbf{V} \subseteq \mathbf{B}$ as they are placed earlier in $\mathbf{B}$.

Example 3 *We extend Example 2 with a Bayesian Network* (G, θ) *with* $\mathbf{V} = \{P, Q\}$ *and* $\mathbf{E} = \{(P, Q)\}$. *Parameters* θ *imply conditional probabilities* $\Pr[Q|P] = 0.6$ *and* $\Pr[Q|\neg P] = 0.3$. *In Figure 2b, we enumerate all assignment of* P *and* Q *to incorporate all conditional probabilities of* Q *given* P. *We, however, observe that the dynamic programming solution in Section 3.2 still prunes search space for variables that do not appear in* $\mathbf{V}$, *such as* $\{R, S\}$. *Hence following the calculation in Figure 2b, we obtain the maximum probability of positive prediction of the classifier as* 0.65 *for* $P = 1$. *The minimum probability of positive prediction (not shown) is similarly calculated as* 0.11 *for* $P = 0$.

3.4 Fairness Verification with Computed Probability of Positive Prediction

Given a classifier $\mathcal{M}$, a distribution $\mathcal{D}$, and a fairness metric f, verifying whether a classifier is ϵ-fair ($\epsilon \in [0, 1]$) is equivalent to computing $\mathbb{1}[f(\mathcal{M}|\mathcal{D}) \leq \epsilon]$. We now compute $f(\mathcal{M}|\mathcal{D})$ based on the maximum probability of positive prediction $\max_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}]$ and the minimum probability of positive prediction $\min_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}]$ of a classifier.

For measuring fairness metric SP, we compute the difference $\max_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}] - \min_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}]$. We, however, deploy FVGM twice while measuring EO: one for the distribution $\mathcal{D}$ conditioned on $Y = 1$ and another for $Y = 0$. In each case, we compute $\max_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, Y = y] - \min_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, Y = y]$ for $y \in \{0, 1\}$ and take the maximum difference as the value of EO. For measuring causal metric PCF, we compute $\max_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, \mathbf{Z}]$ and $\min_{\mathbf{a}} \Pr[\hat{Y} = 1, \mathbf{Z}|\mathbf{A} = \mathbf{a}, \mathbf{Z}]$ conditioned on mediator features $\mathbf{Z}$ and take their difference. To measure DI, we compute the ratio $\max_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}] / \min_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}]$. In contrast to other fairness metrics, DI closer to 1 indicates higher fairness level. Thus, we verify whether a classifier achieves $(1 - \epsilon)$-DI by checking $\mathbb{1}[DI(\mathcal{M}|\mathcal{D}) \geq 1 - \epsilon]$.

3.5 Extension to Practical Settings

For verifying linear classifiers with real-valued features and coefficients, we preprocess them so that FVGM can be invoked. Let $X_c \in \mathbb{R}$ be a continuous real-valued feature with coefficient $w_c \in \mathbb{R}$ in the classifier. We discretize X_c to a set $\mathbf{B}_c$ of k Boolean variables using binning-based discretization and assign a Boolean variable to each bin. Hence, $B_i \in \mathbf{B}_c$ becomes 1, when X_c belongs to the i^{th} bin. Let μ_i denote the mean of feature-values within i^{th} bin. We then set the coefficient of B_i as $w_c \mu_i$. By law of large numbers, $X_c \approx \sum_i \mu_i B_i$ for infinitely many bins (Grimmett and Stirzaker, 2020). Finally, we multiply the coefficients of discretized variables by $l \in \mathbb{N} \setminus \{0\}$ and round to an integer. Accuracy of the preprocessing step relies on the number of bins k and the multiplier l. Therefore, we fine-tune both k and l by comparing the processed classifier with the initial classifier on a validation dataset.

4. Empirical Performance Analysis

In this section, we empirically evaluate the performance of FVGM. We first present the experimental setup and the objective of our experiments, followed by experimental results.

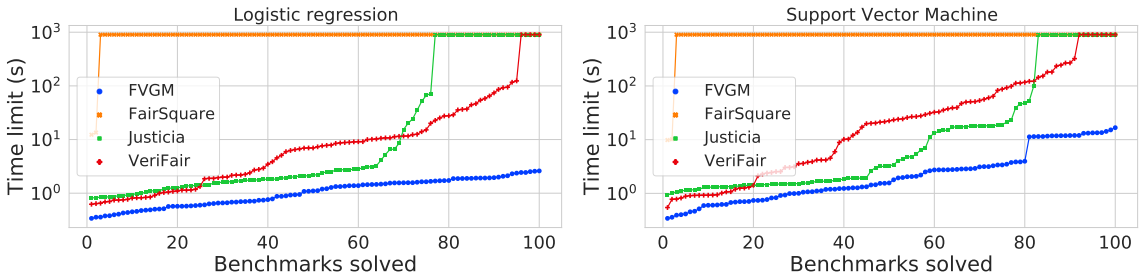


Figure 3: A cactus plot to present the scalability of different fairness verifiers. The number of solved benchmarks are on the X-axis and the required time is on the Y-axis; a point (x, y) implies that a verifier takes less than or equal to y seconds to compute fairness metrics of x many benchmarks. We consider 100 benchmarks generated from 5 real-world datasets using 5-fold cross-validation. In each fold, we consider $\{25, 50, 75, 100\}$ percent of non-sensitive features.

Experimental Setup. We implement a prototype of FVGM in Python (version 3.8). We deploy the Scikit-learn library for learning linear classifiers such as Logistic Regression (LR) and Support Vector Machine (SVM) with linear kernels. We perform five-fold cross-validation on a dataset. While the classifier is trained on continuous features, we discretize them to Boolean features to be invoked by FVGM. During discretization, we apply a gird-search to estimate the best bin-size within a maximum bin of 10. To convert the coefficients of features into integers, we employ another grid-search to choose the best multiplier within $\{1, 2, \ldots, 100\}$. For learning a Bayesian network on the converted Boolean data, we deploy the PGMPY library (Ankan and Panda, 2015). For network learning, we apply a Hill-climbing search algorithm that learns a DAG structure by optimizing K2 score (Koller and Friedman, 2009). For estimating parameters of the network, we use Maximum Likelihood Estimation (MLE) algorithm.

We compare FVGM with three existing fairness verifiers: Justicia (Ghosh et al., 2021), FairSquare (Albarghouthi et al., 2017), and VeriFair (Bastani et al., 2019).

4.1 Scalability Analysis

Benchmarks. We perform the scalability analysis on five real-world datasets studied in fair ML literature: UCI Adult, German-credit (Doshi-Velez and Kim, 2017), COMPAS (Angwin et al., 2016), Ricci (McGinley, 2010), and Titanic (`https://www.kaggle.com/c/titanic`). We consider 100 benchmarks generated from 5 real-world datasets and report the computation times (for DI and SP) of different verifiers.

Results. In Figure 3, we present the scalability results of different verifiers. First, we observe that FairSquare often times out (= 900 seconds) and can solve ≤ 5 benchmarks. This indicates that SMT-based reduction for linear classifiers cannot scale. Similarly, SSAT-based verifier Justicia that performs pseudo-Boolean to CNF translation for linear classifiers, times out for around 20 out of 100 benchmarks. Sampling-based framework, VeriFair, has comparatively better scalability than SMT/SSAT based frameworks and can solve more than

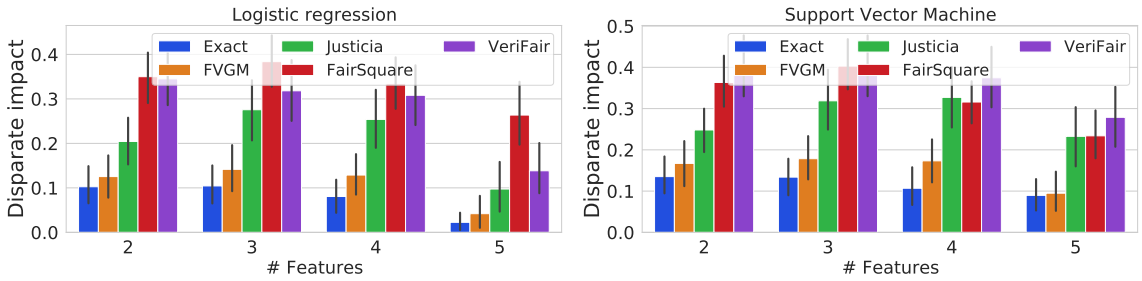


Figure 4: Comparing the average accuracy of different verifiers over 100 synthetic benchmarks while varying the number of features. FVGM yields the closest estimation of the analytically calculated *Exact* values of DI for LR and SVM classifiers.

90 benchmarks. Finally, FVGM achieves impressive scalability by solving all 100 benchmarks with 1 to 2 orders of magnitude runtime improvements than existing verifiers. Therefore, S3P-*based framework* FVGM *proves to be highly scalable in verifying fairness properties of linear classifiers than the state-of-the-art.*

4.2 Accuracy Analysis

Benchmark Generation. To perform accuracy analysis, we require the ground truth, which is not available for real-world instances. Therefore, we focus on generating synthetic benchmarks for analytically computing the ground truth of different fairness metrics, such as DI, from the known distribution of features. In each benchmark, we consider $n \in \{2, 3, 4, 5\}$ features including one Boolean sensitive feature, say A, generated from a Bernoulli distribution with mean 0.5. We generate non-sensitive features X_i from Gaussian distributions such that $\Pr[X_i|A = 1] \sim \mathcal{N}(\mu_i, \sigma^2)$ and $\Pr[X_i|A = 0] \sim \mathcal{N}(\mu_i', \sigma^2)$, where $\mu_i, \mu_i' \in [0, 1]$, $\sigma = 0.1$, and μ_i, μ_i' are chosen from a uniform distribution in $[0, 1]$. Finally, we create label $Y = \mathbb{1}[\sum_{i=1}^{n-1} X_i \geq 0.5 \sum_{i=1}^{n-1}(\mu_i + \mu_i')]$ such that Y does not directly depend on the sensitive feature. For each n, we generate 100 random benchmarks, learn LR and SVM classifiers on them, and compute DI[2] using different verifiers.

Results. We assess the accuracy of the competing verifiers in estimating fairness metrics, specifically DI with LR and SVM classifiers. In Figure 4, FVGM computes DI closest to the *Exact* value for different number of features and both type of classifiers. In contrast, Justicia, FairSquare, and VeriFair measure DI far from the *Exact* because of ignoring correlations among the features. For example, for SVM classifier with $n = 5$ (right plot in Figure 4), *Exact* DI is 0.089 (average over 100 random benchmarks). Here, FVGM computes DI as

2. To analytically compute DI, let the coefficients of the classifier be w_i for X_i and w_A for A, and bias be τ. Since all non-sensitive features are from Gaussian distributions, we compute the probability of the predicted class $\Pr[\hat{Y}|A = 1] \sim \mathcal{N}(\sum_{i=1}^{n-1} w_i\mu_i, \sigma_{\hat{Y}}^2)$ and $\Pr[\hat{Y}|A = 0] \sim \mathcal{N}(\sum_{i=1}^{n-1} w_i\mu_i', \sigma_{\hat{Y}}^2)$ with $\sigma_{\hat{Y}}^2 = (\sum_{i=1}^{n-1} w_i^2)\sigma^2$. Hence, the probability of positive prediction of the classifier is $1 - \mathsf{CDF}_{\hat{Y}|A=1}(\tau - w_A)$ for $A = 1$ and $1 - \mathsf{CDF}_{\hat{Y}|A=0}(\tau)$ for $A = 0$, where CDF is the cumulative distribution function. Finally, we compute DI by taking the ratio of the minimum and the maximum of the probability of positive prediction of the classifier.

Dataset	**A**	Algo.	ΔDI	ΔPCF	ΔSP	ΔEO
Adult	race	RW	**0.53**	**-0.06**	**-0.06**	**-0.02**
		OP	**0.57**	**-0.07**	**-0.07**	**-0.02**
	sex	RW	**0.96**	**-0.16**	**-0.15**	**-0.08**
		OP	**0.43**	**-0.08**	**-0.08**	0.03
COMPAS	race	RW	**0.13**	**-0.07**	**-0.07**	**-0.06**
		OP	**0.15**	**-0.08**	**-0.08**	**-0.05**
	sex	RW	**0.1**	**-0.04**	**-0.04**	0.04
		OP	**0.09**	**-0.04**	**-0.04**	**-0.03**
German	age	RW	**0.52**	**-0.53**	**-0.52**	**-0.47**
		OP	**0.53**	**-0.53**	**-0.53**	**-0.51**
	sex	RW	−0.06	0.06	0.06	0.02
		OP	−0.12	0.12	0.12	0.07

Table 1: Verification of fairness algorithms using FVGM. **A** denotes sensitive features. RW and OP refer to reweighing and optimized-preprocessing algorithms. Numbers in bold refer to fairness improvement.

0.094, while all other verifiers compute DI as at least 0.233. Therefore, FVGM *is more accurate than existing verifiers as it explicitly considers correlations among features.*

5. Applications of FVGM

In this section, we apply FVGM for verifying fairness-enhancing algorithms and depreciation attacks. We also demonstrate that FVGM facilitates computation of fairness influence functions by enabling the detection of bias due to individual features.

Verifying Fairness-enhancing Algorithms. We deploy FVGM in verifying the effectiveness of fairness-enhancing algorithms designed to ameliorate bias. For example, fairness pre-processing algorithms can be validated by applying FVGM on the unprocessed and the processed data separately and comparing different fairness metrics. In Table 1, we report the effect of fairness algorithms w.r.t. four fairness metrics: disparate impact (DI), path-specific causal fairness (PCF), statistical parity (SP), and equalized odds (EO). Note that, fairness is improved if DI increases and the rest of the metrics decrease. For instance, in most instances for Adult dataset, reweighing (RW) (Kamiran and Calders, 2012) and optimized pre-processing (OP) (Calmon et al., 2017) algorithms are successful in improving fairness. The exceptional case is the unfairness regarding the sensitive feature 'sex', where OP algorithm fails in improving fairness metric EO. Thus, FVGM *verifies the enhancement and decrement in fairness by fairness-enhancing algorithms.*

Verifying Fairness Attacks. We apply FVGM in verifying a fairness poisoning-attack algorithm. This algorithm injects a small fraction of poisoned samples into the training data such that the classifier becomes relatively unfair (Solans et al., 2020). We apply this

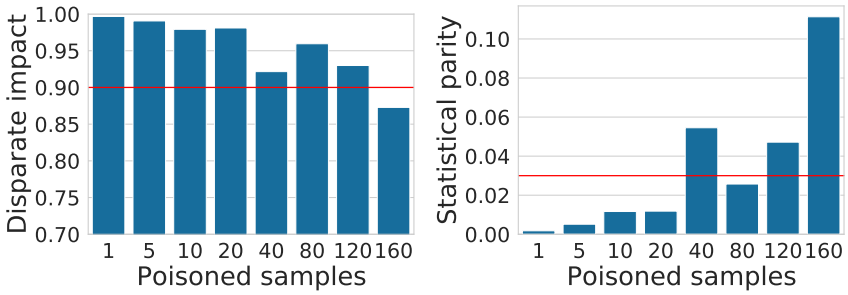


Figure 5: Verifying poisoning attack against fairness using FVGM. The red line denotes the safety margin of the ML model against the attack.

attack to add $\{1, 5, \ldots, 160\}$ poisoned samples and measure the corresponding disparate impact and statistical parity. In Figure 5, FVGM verifies that the disparate impact of the classifier decreases and statistical parity increases, i.e. *the classifier becomes more unfair, as the number of poisoned samples increases.* Therefore, FVGM shows the potential of being deployed in safety-critical applications to detect fairness attacks. For example, if we set 0.9 as an acceptable threshold of disparate impact, FVGM can raise an alarm once 160 poisoned samples are added.

Fairness Influence Function (FIF): Tracing Sources of Unfairness. Another application of FVGM as a fairness verifier is to quantify the effect of a subset of features on fairness. Thus, we define fairness influence function (FIF) that computes the effect of a subset of non-sensitive features $\mathbf{S} \subseteq \mathbf{X}$ on the probability of positive prediction of a classifier given a specific sensitive group $\mathbf{A} = \mathbf{a}$, $\mathsf{FIF}(\mathbf{S}) \triangleq \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, \mathcal{D}] - \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, \mathcal{D}_{-\mathbf{S}}]$. FIF allows us to explain the sources of unfairness in the classifier. In practice, we compute FIF of $\mathbf{S}$ by replacing the probability distribution of $\mathbf{S}$ with a uniformly random distribution, referred to as $\mathcal{D}_{-\mathbf{S}}$, and reporting differences in the conditional probability of positive prediction of the classifier.

In Figure 6, we compute FIF for all features in COMPAS dataset, separately for two sensitive groups: Female ('sex' = 1) and Male. This dataset concerns the likelihood of a person of re-offending crimes within the next two years. We first observe that the base values of the probability of positive prediction are different for the two groups (0.46 vs 0.61 for Female and Male), thereby showing Male as more probable to re-offend crimes than Female. Moreover, FIF of the feature 'age' is comparatively higher in magnitude for Male than Female. This implies that while deciding recidivism the algorithm assumes that Female individuals across different ages re-offend crimes with almost the same probability and the probability of re-offending for Male individuals highly depends on age. Thus, applying FVGM to compute FIF provides us insights about sources of bias and thus, indicators to improve fairness. $\Pr[\hat{Y} = 1|\text{age_0} \geq 0.5]$

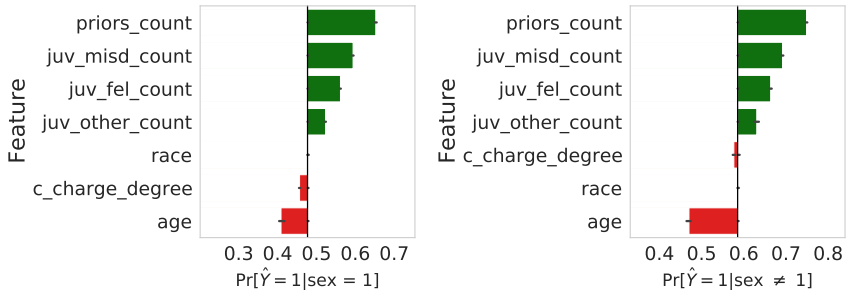


Figure 6: FIF computation for COMPAS dataset. For Female (left plot) and Male, influence of 'age' decreases and the probability of positive prediction of the classifier increases by different magnitudes.

6. Conclusion

In this paper, we propose FVGM, an efficient fairness verification framework for linear classifiers based on a novel stochastic subset-sum problem. FVGM encodes a graphical model of feature-correlations, represented as a Bayesian Network, and computes multiple group and causal fairness metrics accurately. We experimentally demonstrate that FVGM is not only more accurate and scalable than the existing verifiers but also applicable in practical fairness tasks, such as verifying fairness attacks and enhancing algorithms, and computing the fairness influence functions. As a future work, we aim to design fairness-enhancing algorithms certified by fairness verifiers, such as FVGM. Since FVGM serves as an accurate and scalable fairness verifier for linear classifiers, it will be interesting to design such verifiers for other ML models.

Acknowledgments

This work was supported in part by National Research Foundation, Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0004] and AI Singapore Programme [AISG-RP-2018-005], and NUS ODPRT Grant [R-252-000-685-13]. The computational work for this paper was performed on resources of Max Planck Institute for Software Systems, Germany and the National Supercomputing Centre, Singapore (`https://www.nscc.sg`).

References

Ifeoma Ajunwa, Sorelle Friedler, Carlos E Scheidegger, and Suresh Venkatasubramanian. Hiring by algorithm: predicting and preventing disparate impact. *Available at SSRN*, 2016. URL: `http://sorelle.friedler.net/papers/SSRN-id2746078.pdf`.

Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V Nori. FairSquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–30, 2017.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias risk assessments in criminal sentencing. *ProPublica, May*, 23, 2016.

Ankur Ankan and Abinash Panda. PGMPY: Probabilistic graphical models using Python. In *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer, 2015.

Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Probabilistic verification of fairness properties via concentration. *Proceedings of the ACM on Programming Languages*, 3 (OOPSLA):1–27, 2019.

Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. Ai fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias, Oct 2018. URL `https://arxiv.org/abs/1810.01943`.

Richard Berk. Accuracy and fairness for juvenile justice risk assessments. *Journal of Empirical Legal Studies*, 16(1):175–194, 2019.

Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems*, pages 3992–4001, 2017.

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

Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4:eaao5580, 2018.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

Bishwamittra Ghosh, Dmitry Malioutov, and Kuldeep S. Meel. Classification rules in relaxed logical form. In *Proceedings of ECAI*, 6 2020.

Bishwamittra Ghosh, Debabrota Basu, and Kuldeep S. Meel. Justicia: A stochastic SAT approach to formally verify fairness. In *Proceedings of AAAI*, 2 2021.

Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford University press, 2020.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. Verifying individual fairness in machine learning models. *arXiv preprint arXiv:2006.11737*, 2020.

Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, 2012.

Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in neural information processing systems*, pages 4066–4076, 2017.

Frank J Landy, Janet L Barnes, and Kevin R Murphy. Correlates of perceived fairness and accuracy of performance evaluation. *Journal of Applied psychology*, 63(6):751, 1978.

Nian-Ze Lee, Yen-Shi Wang, and Jie-Hong R Jiang. Solving stochastic Boolean Satisfiability under random-exist quantification. In *IJCAI*, pages 688–694, 2017.

Nian-Ze Lee, Yen-Shi Wang, and Jie-Hong R Jiang. Solving exist-random quantified stochastic Boolean Satisfiability via clause selection. In *IJCAI*, pages 1339–1345, 2018.

Michael L Littman, Stephen M Majercik, and Toniann Pitassi. Stochastic Boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001.

Barbara Martinez Neda, Yue Zeng, and Sergio Gago-Masague. Using machine learning in admissions: Reducing human and algorithmic bias in the selection process. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 1323–1323, 2021.

Ann C McGinley. Ricci v. destefano: A masculinities theory analysis. *Harv. JL & Gender*, 33:581, 2010.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

Razieh Nabi and Ilya Shpitser. Fair inference on outcomes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, pages 15–17, 1985.

David Pisinger. Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, 33(1):1–14, 1999.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *arXiv preprint arXiv:1709.02012*, 2017.

David Solans, Battista Biggio, and Carlos Castillo. Poisoning attacks on algorithmic fairness. *arXiv preprint arXiv:2004.07401*, 2020.

Nikolaj Tollenaar and PGM Van der Heijden. Which method predicts recidivism best? a comparison of statistical, machine learning and data mining predictive models. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 176(2):565–584, 2013.

Gerhard J Woeginger and Zhongliang Yu. On the equal-subset-sum problem. *Information Processing Letters*, 42(6):299–302, 1992.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial Intelligence and Statistics*, pages 962–970, 2017.

Junzhe Zhang and Elias Bareinboim. Fairness in decision-making—the causal explanation formula. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Indre Zliobaite. On the relation between accuracy and fairness in binary classification. *arXiv preprint arXiv:1505.05723*, 2015.

Appendix A. Broader Impact

We have presented an efficient framework for verifying fairness of linear classifiers. This work does not bear any negative societal impact. The sole purpose of our framework is to detect bias of machine learning models that are deployed in high-stake decision making.

Appendix B. Background

B.1 Fairness Metrics

In this section, we state the fairness metrics verified in this paper in further details.

B.1.1 Group Fairness.

Group fairness is categorized into three families: independence, separation and sufficiency, of which FVGM verifies independence and separation metrics. The independence metrics state that the predicition of the classifier should be independent of compound sensitive groups. Formally, independence notion specifies an equal probability of positive prediction across all sensitive groups for a classifier $\mathcal{M}$, i.e., $\Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}] = \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}'], \forall \mathbf{a}, \mathbf{a}' \in A$. Since satisfying independence exactly is hard, relaxations of independence fairness metrics, such as *disparate impact* and *statistical parity* (Dwork et al., 2012; Feldman et al., 2015), are proposed.

Disparate impact (DI) (Feldman et al., 2015) measures the ratio of probabilities of positive prediction between the most favored group and least favored group, and prescribe it to be close to 1. Formally, a classifier satisfies $(1-\epsilon)$-disparate impact if, for $\epsilon \in [0, 1]$,

$$\min_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}] \geq (1-\epsilon) \max_{\mathbf{a}'} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}'].$$

Another popular relaxation of independence metrics is *statistical parity* (SP) that measures the difference of probability of positive prediction among sensitive groups, and prescribe this to be near zero. Formally, an algorithm satisfies ϵ-statistical parity if, for $\epsilon \in [0, 1]$,

$$\max_{\mathbf{a},\mathbf{a}'} |\Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}] - \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}']| \leq \epsilon.$$

For both disparate impact and statistical parity, lower value of ϵ indicates higher group fairness of the classifier $\mathcal{M}$.

In the *separation (or classification parity)* notion of fairness, the predicted label $\hat{Y}$ of a classifier $\mathcal{M}$ is independent of the sensitive features $\mathbf{A}$ given the class labels Y. In case of binary classifiers, a popular separation metric is *equalized odds* (EO) (Hardt et al., 2016) that computes the difference of false positive rates (FPR) and the difference of true positive rates (TPR) among all compound sensitive groups. Lower value of equalized odds indicates better fairness. A classifier $\mathcal{M}$ satisfies ϵ-equalized odds if, for all compound sensitive groups $\mathbf{a}, \mathbf{a}' \in A$, $\max_{\mathbf{a},\mathbf{a}'} |\Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, Y = 0] - \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}', Y = 0]| \leq \epsilon$, and $\max_{\mathbf{a},\mathbf{a}'} |\Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, Y = 1] - \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}', Y = 1]| \leq \epsilon$.

B.1.2 Path-specific Causal Fairness.

Let $\mathbf{a}_{\max} \triangleq \arg\max_{\mathbf{a}} \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}]$. We consider mediator features $\mathbf{Z} \subseteq \mathbf{X}$ sampled from the conditional distribution $\mathcal{Z}_{|\mathbf{A}=\mathbf{a}_{\max}}$. This emulates the fact that mediator variables have

the same sensitive features $\mathbf{a}_{\max}$. For $\epsilon \in [0, 1]$, path-specific causal fairness is defined as $\max_{\mathbf{a},\mathbf{a}'} |\Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, \mathbf{Z}] - \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}', \mathbf{Z}]| \leq \epsilon$.

Therefore, PCF constrains that $\hat{Y}$ is not directly dependent of $\mathbf{A}$ while $\mathbf{A}$ may indirectly affects $\hat{Y}$ only through $\mathbf{Z}$. PCF is a variation of counterfactual fairness and causal fairness without mediator features (Bastani et al., 2019).

Example 4 *Following (Bastani et al., 2019), we consider a classifier that decides the hiring of employees based on three features: gender (sensitive), years of experience (non-sensitive), and college-participation (mediator). It is practical to consider that gender* $\in$ *{male, female} can affect the college-participation of individuals, and all three features are determining factors for the hiring process. Let 'male' be the most favored group by the classifier, for instance. Path-specific causal fairness (PCF) ensures that a female candidate should be given a job offer with similar probability as a male candidate (by constraining* $\epsilon \approx 0$*). She, however, went to (participated in) college as if she were a male candidate while other non-mediator features such as 'years of experience' are the same. Therefore, PCF measures the effect of gender on job offer, but ignores the effect of gender on whether candidates went to college.*

Appendix C. Proofs of Theoretical Results

Lemma 1 *Let* n' *be the number of existential and universal variables in* $\mathbf{B}$*. Let* $w_\exists = \sum_{B_i \in \mathbf{B}|q_i=\exists} \max\{w_i, 0\}$ *and* $w_\forall = \sum_{B_i \in \mathbf{B}|q_i=\forall} \min\{w_i, 0\}$ *be the considered sum of weights of existential and universal variables, respectively. We can exactly solve* S3P *using dynamic programming with time complexity* $\mathcal{O}((n - n')(\tau + |w_{neg}| - w_\exists - w_\forall) + n')$*. The space complexity is* $\mathcal{O}((n - n')(\tau + |w_{neg}| - w_\exists - w_\forall))$*.*

Proof Case 1: All n variables in B have randomized quantifiers.

At step-i of the dynamic programming (Eq. (3)), we modify the residual threshold of that step, namely τ_i, either by subtracting w_i or by retaining it. Now, we observe that the residual threshold τ_i for any $i \in \{1, \ldots, n\}$ will be in $[0, \tau + |w_{neg}|]$. This holds because if τ_i crosses these bounds, the dynamic programming is terminated as shown in Equation (4). Since all weights of $\{w_i\}_{i=1}^n$ are integers, the maximum number of values that the residual threshold can take, is $(\tau + |w_{neg}|)$. Thus, we need to store at most $n(\tau + |w_{neg}|)$ values in the memory for performing dynamic programming with n variables and $(\tau + |w_{neg}|)$ number of possible weights. Thus, the space complexity is $\mathcal{O}(n(\tau + |w_{neg}|))$.

In order to construct the dynamic programming table, we have to call the dp function $\mathcal{O}(n(\tau + |w_{neg}|))$ times, in the worst-case. Thus, the time complexity of our method is $\mathcal{O}(n(\tau + |w_{neg}|)$.

Case 2: n' variables have existential or universal quantifiers and $n - n'$ variables have randomized quantifiers in B.

According to Eq. (3), $w_\exists$ and $w_\forall$ are the fixed weights of all existential and universal variables, respectively. Therefore, we need to consider at most $\tau + |w_{neg}| - w_\exists - w_\forall$ values of residual weights for random variables. By applying analysis in Case 1, the space and time complexity is derived as $\mathcal{O}((n - n')(\tau + |w_{neg}| - w_\exists - w_\forall))$.

We note that there is an additional time complexity of $\mathcal{O}(n')$ for existential and universal variables in Eq. (3). Thus the time complexity becomes $\mathcal{O}((n-n')(\tau+|w_{neg}|-w_\exists-w_\forall)+n')$. We, however, do not require to store any entry for existential and universal variables in dp

function and thus, the space complexity remains the same as $\mathcal{O}((n-n')(\tau+|w_{neg}|-w_{\exists}-w_{\forall}))$. ■

Lemma 2 *Let $\mathbf{V} \subseteq \mathbf{B}$ be the set of vertices in the Bayesian network and n'' be the number of existential and universal variables in $\mathbf{B} \setminus \mathbf{V}$. Let $w'_{\exists} = \sum_{B_i \in \mathbf{B}\setminus\mathbf{V}|q_i=\exists} \max\{w_i, 0\}$ and $w'_{\forall} = \sum_{B_i \in \mathbf{B}\setminus\mathbf{V}|q_i=\forall} \min\{w_i, 0\}$ be the sum of considered weights of existential and universal variables, respectively that only appear in $\mathbf{B} \setminus \mathbf{V}$. To exactly compute* S3P *with correlated variables in dynamic programming approach, time complexity is $\mathcal{O}(2^{|\mathbf{V}|} + (n - n'' - |\mathbf{V}|)(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall}) + n'')$ and space complexity is $\mathcal{O}((n - n'' - |\mathbf{V}|)(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall}))$.*

Proof We first separate analysis of space and time complexity for variables in $\mathbf{V}$ and in $\mathbf{B} \setminus \mathbf{V}$. For each Boolean variable in $\mathbf{V}$, we enumerate all assignments, which has time complexity of $2^{|\mathbf{V}|}$ and there is no space complexity as discussed in Section 3.3.

For variables in $\mathbf{B} \setminus \mathbf{V}$, we apply analysis from Lemma 2, where we consider $(n-n''-|\mathbf{V}|)$ random variables, n'' existential/universal variables, and residual weights can take at most $(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall})$ values. Hence, time complexity is $\mathcal{O}((n - n'' - |\mathbf{V}|)(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall}) + n'')$, and space complexity is $\mathcal{O}((n - n'' - |\mathbf{V}|)(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall}))$

Combining two cases, overall time complexity is $\mathcal{O}(2^{|\mathbf{V}|} + (n - n'' - |\mathbf{V}|)(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall}) + n'')$ and space complexity is $\mathcal{O}((n - n'' - |\mathbf{V}|)(\tau + |w_{neg}| - w'_{\exists} - w'_{\forall}))$. ■

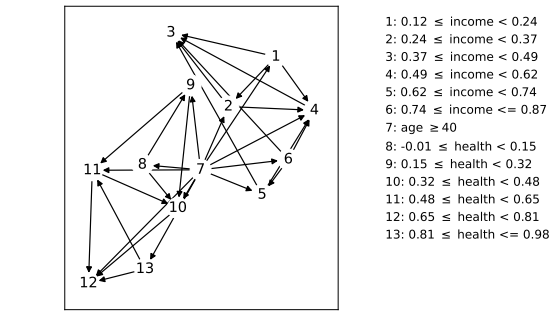
Appendix D. Extended Experimental Evaluations

Each experiment is performed on Intel Xeon E7 − 8857 v2 CPUs with 16GB memory, 64bit Linux distribution based on Debian OS and clock speed 3 GHz. In the following, we discuss extended experimental results.

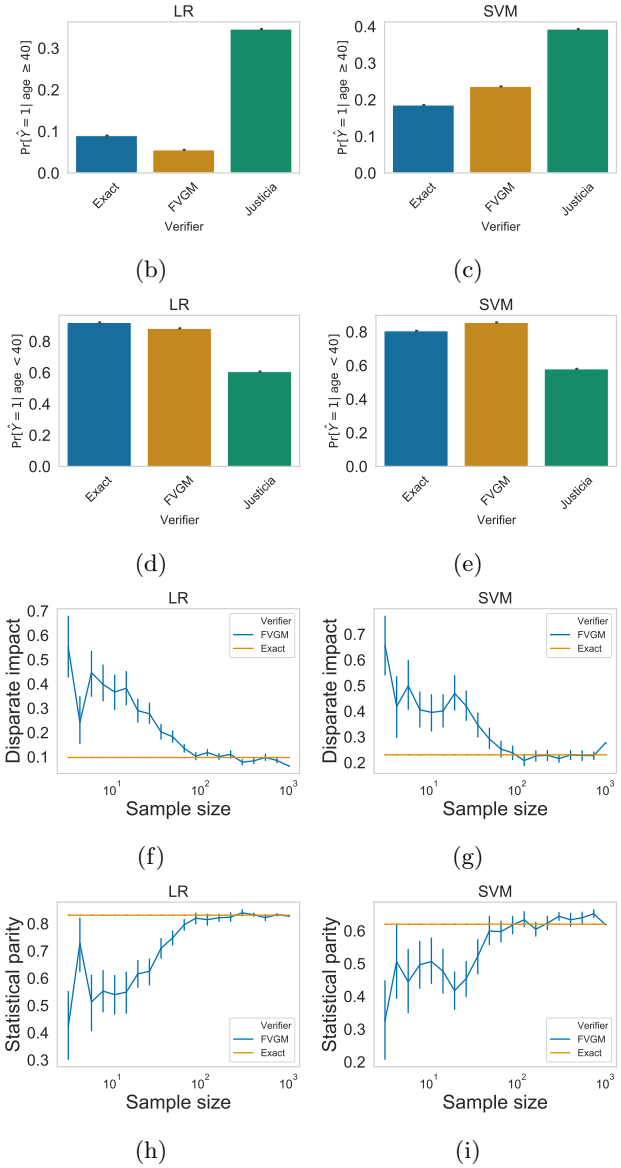
D.1 Accuracy Comparison Among Different Verifiers

We have considered a synthetic problem for comparing accuracy among different verifiers. For Example 1, we consider 'age ≥ 40' as a Bernoulli random variable with probability 0.5. For 'income' feature (I), we consider two Gaussian distributions $\Pr[I|\text{age} \geq 40] \sim \mathcal{N}(0.6, 0.1)$ and $\Pr[I|\text{age} < 40] \sim \mathcal{N}(0.4, 0.1)$ separated by two age groups. Moreover, for 'fitness' feature (F), we consider two Gaussian distributions $\Pr[F|\text{age} \geq 40] \sim \mathcal{N}(0.7, 0.1)$ and $\Pr[F|\text{age} < 40] \sim \mathcal{N}(0.3, 0.1)$. On this data, the trained LR and SVM classifier has decision boundary as $7.26I + 7.4F - 1.34A \geq 6.62$ and $9.37I + 9.75F - 0.34A \geq 9.4$, respectively.

In Figure 7a we show the Bayesian Network on discretized features, in particular for income and fitness features. In Figure 7b to Figure 7e, we show the probability of positive prediction of different classifiers computed by different verifiers, where FVGM outputs closest to exactly computed values, in comparison with Justicia. In Figure 7f to Figure 7i, we show the effect of sample size on FVGM in measuring fairness metrics: disparate impact and statistical parity, where with increasing sample size, the estimate becomes more accurate.



(a)



(b) (c) (d) (e) (f) (g) (h) (i)

Figure 7: Measuring accuracy of different fairness verifiers for Example 1. LR, SVM, DT refers to Logistic Regression classifier, Support Vector Machine, and Decision Tree, respectively.

Table 2: Extended results for verification of different fairness enhancing algorithms using FVGM. Numbers in bold refer to fairness improvement w.r.t. different fairness metrics. RW and OP refer to reweighing and optimized-preprocessing algorithms respectively.

Dataset	Sensitive	Algo.	ΔDI	ΔPCF	ΔSP	Δ EO
Adult	race	RW	**0.53**	**-0.06**	**-0.06**	**-0.02**
		OP	**0.57**	**-0.07**	**-0.07**	**-0.02**
	sex	RW	**0.96**	**-0.16**	**-0.15**	**-0.08**
		OP	**0.43**	**-0.08**	**-0.08**	0.03
COMPAS	race	RW	**0.13**	**-0.07**	**-0.07**	**-0.06**
		OP	**0.15**	**-0.08**	**-0.08**	**-0.05**
	sex	RW	**0.1**	**-0.04**	**-0.04**	0.04
		OP	**0.09**	**-0.04**	**-0.04**	**-0.03**
German	age	RW	**0.52**	**-0.53**	**-0.52**	**-0.47**
		OP	**0.53**	**-0.53**	**-0.53**	**-0.51**
	sex	RW	−0.06	0.06	0.06	0.02
		OP	−0.12	0.12	0.12	0.07

D.2 Scalability Comparison Among Different Verifiers

In Figure 8, we present the runtime of different fairness verifiers while varying the number of features in different datasets. We observe that with an increase of features, the runtime increases in general.

D.3 Verifying Fairness Algorithms on Multiple Fairness Metrics

We show extended results on verifying fairness attack in Figure 9 for two fairness metrics: disparate impact (DI) and statistical parity (SP). We observe that FVGM can detect poisoning attack for both metrics.

In Figure 10 we show verification results on compound sensitive groups with respect to multiple fairness metrics. In this figure, we observe that with an increase in the number of groups, fairness metrics worsens—disparate impact decreases and other three metrics increases. Additionally, in Table 2, we show extended results for German and COMPAS dataset on verifying fairness algorithms: reweighing and optimized pre-processing algorithms.

D.4 Performance Analysis of Bayesian Network

In Figure 11, we analyze the performance of encoding Bayesian Networks of differing complexity. We define the complexity of the network as $\frac{|V|}{|\mathbf{X} \cup \mathbf{A}|}$, which is the ratio between the number of features appearing in the network and total features. In this figure, as the ratio increases, both computation time of FVGM and learning time of Bayesian Network increase.

Appendix E. Fairness Influence Functions (FIF)

Now, we present an elaborate discussion on computing fairness influence functions. A Fairness Influence Function, denoted as $\mathsf{FIF}(\cdot)$, is computed with respect to a *quantity of interest*, for example, the probability of positive prediction of the classifier or different fairness metrics such as DI and SP. Let $\mathbf{S} \subseteq \mathbf{X}$ be a set of non-sensitive features, for which we are interested in computing their influence. A general approach to compute $\mathsf{FIF}(\mathbf{S})$ is to replace each feature in $\mathbf{S}$ with random values and report differences in the quantity of interest (Datta et al., 2016). Let $\mathcal{X}_i'$ denote the modified marginal distribution where we replace feature $X_i \in \mathbf{S}$ with random values. For example, $\mathcal{X}_i'$ can be viewed as an uniform distribution within the support of X_i. We then define the modified product distribution, denoted as $\mathcal{D}_{-\mathbf{S}}$ by combining all marginal distributions, where each feature in $\mathbf{S}$ has a modified distribution.

$$\mathcal{D}_{-\mathbf{S}} = \prod_{i|X_i \in \mathbf{S}} \mathcal{X}_i' \prod_{i|X_i \in X \setminus \mathbf{S}} \mathcal{X}_i \prod_{j=1}^{n} \mathcal{A}_j$$

We first give a general definition of influence function of $\mathbf{S}$ on a quantity, say Q, in the following.

$$\mathsf{FIF}(\mathbf{S}) \triangleq Q(\mathcal{D}) - Q(\mathcal{D}_{-\mathbf{S}})$$

Intuitively, influence of $\mathbf{S}$ is the difference in a quantity computed for the original distribution $\mathcal{D}$ and the modified distribution $\mathcal{D}_{-\mathbf{S}}$. In the following, we define influence function in terms of the probability of positive prediction of the classifier.

$$\mathsf{FIF}(\mathbf{S}) = \Pr[\hat{Y} = 1|\mathcal{D}] - \Pr[\hat{Y} = 1|\mathcal{D}_{-\mathbf{S}}]$$

Influence function also generalizes to the probability of positive prediction specific to compound sensitive groups. For a sensitive group $\mathbf{a} \in \mathbf{A}$, we define group-specific influence as follows.

$$\mathsf{FIF}_{\mathbf{a}}(\mathbf{S}) \triangleq \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, \mathcal{D}] - \Pr[\hat{Y} = 1|\mathbf{A} = \mathbf{a}, \mathcal{D}_{-\mathbf{S}}]$$

Experimental Analysis. We empirically instantiate computation of FIF and its consequences for a logistic regression classifier on COMPAS and Adult datasets.

For both the datasets, we consider biological 'sex' as the sensitive features. In both cases, we denote the sensitive groups 'male' and 'female' as 'sex=0' and 'sex=1', respectively. FIFs computed for these sensitive groups show influence of different features and their relevant disparities. We illustrate the results in Figure 12.

Figure 12a illustrates that the probability of positive prediction for the 'male' population in the dataset is 0.61. If we now replace the feature 'age' with uniformly random values, the probability of positive prediction for the same group becomes 0.46, Thus, the FIF of the feature 'age' for 'male' group is 0.15. Similarly, all the red bars in Figures 12a-12b, 12e-12c indicate a positive influence of that feature and the green bars indicate a negative influence of the feature on the individual getting classified to $\hat{Y} = 1$.

In Figures 12e and 12f, we illustrate the influence of different features on the disparate impact between the two groups 'male' and 'female'. The green bar indicates that removing a feature causes an increase in DI, i.e. fairness in classification, and the red bar indicates the

opposite. Alternatively, we can conclude that bigger is the green bar for a feature, higher is the bias-inducing effect of it.

Thus, from Figure 12e, we conclude that 'age' is the most bias-inducing feature among the two groups 'male' and 'female'. From Fig 12a and 12b, we observe that 'age' is a decisive feature for 'male' while it is comparatively insignificant for 'female'.

In case of Adult dataset, the probability of positive prediction for 'male' and 'female' groups are 0.45 and 0.3 respectively. Among all the features, we observe that 'race' has a positive impact on the classification for both the groups. This means removing the 'race' information decreases the chance of getting classified to higher economic group. On the other hand, 'capital-gain' has the highest influence on the classification for both groups. It is also the most biased-inducing feature as it varies most disparately among the two groups as the 'capital-gain' differs significantly between two groups.

Appendix F. Fairness Verification of CNF Classifiers with Feature-correlations

Existing verifiers such as Justicia focuses mainly on classifiers expressible as a CNF formula. But none of the existing verifiers consider correlation among the features while computing fairness metrics. In this section, we address fairness verification of CNF-based classifiers with explicit consideration of correlations among features. In particular, we present how to encode a Bayesian network, which captures correlated features, into a SSAT-based formulation that is tailored for verifying CNF classifiers. To this end, we first discuss the basics of SSAT followed by an elaboration of the proposed methodology.

F.1 Background: Stochastic Boolean Satisfiability (SSAT)

Let $\mathbf{B} = \{B_1, \ldots, B_m\}$ be a set of Boolean variables taking assignment in $\{0,1\}^m$. A *literal* l_i is a variable B_i or its complement $\neg B_i$. A propositional formula ϕ defined over $\mathbf{B}$ is in CNF if $\phi \triangleq \wedge_i C_i$ is a conjunction of clauses and each clause $C_i \triangleq \vee_j l_j$ is a disjunction of literals. A CNF formula ϕ is satisfied (called SAT) if there is an assignment σ over $\mathbf{B}$ that evaluates formula ϕ to 1 (or true)—at least one literal in each clause is true and all clauses are true. We additionally consider a quantification over each Boolean variable B_i, denoted by $q_i \in \{\exists, \forall, \mathfrak{R}^{p_i}\}$, where $\exists$ is existential, $\forall$ is universal and $\mathfrak{R}$ is a randomized quantifier with $p_i = \Pr[B_i = 1]$. The SSAT problem takes *an ordered set of quantifiers* over $\mathbf{B}$ and a CNF formula ϕ and computes the *probability of satisfaction* of ϕ given quantifiers. Formally, a SSAT formula $\Phi \triangleq (\{q_i\}_{i=1}^m, \phi)$ and the SSAT problem computes $\Pr[\Phi] \in [0,1]$. The semantics of SSAT formula is in the following.

1. $\Pr[\text{true}] = 1$, $\Pr[\text{false}] = 0$,
2. $\Pr[\Phi] = \max_{B_1}\{\Pr[\Phi|_{B_1}], \Pr[\Phi|_{\neg B_1}]\}$ if B_1 is existentially quantified ($\exists$),
3. $\Pr[\Phi] = \min_{B_1}\{\Pr[\Phi|_{B_1}], \Pr[\Phi|_{\neg B_1}]\}$ if B_1 is universally quantified ($\forall$),
4. $\Pr[\Phi] = p\Pr[\Phi|_{B_1}] + (1-p)\Pr[\Phi|_{\neg B_1}]$ if B_1 is randomized quantified ($\mathfrak{R}^p$) with probability $p = \Pr[B_1 = 1]$,

In SSAT, we recursively solve for each Boolean variable in $\mathbf{B}$ starting with the first variable B_1 where $\Phi|_{B_1}$ (resp. $\Phi|_{\neg B_1}$) is the SSAT formula with CNF ϕ substituted by an assignment of B_1 as true (resp. false) and quantifiers $\{q_i\}_{i=2}^{m}$.

In this paper, we are interested in two specific SSAT formulations: exists-random (ER) SSAT formulas and universal-random (UR) SSAT formulas. In ER-SSAT (resp. UR-SSAT), $\{q_i\}$ is constructed such that existential (resp. universal) quantified variables is followed by randomized quantified variables. For more details on SSAT formulas, we refer to (Ghosh et al., 2021; Lee et al., 2017, 2018).

F.2 Methodology

For CNF classifiers (Ghosh et al., 2020), SSAT is a natural choice as it computes the probability of a CNF formula given quantifiers of variables—equivalently, the probability of positive prediction of a CNF classifier. (Ghosh et al., 2021) has proposed a SSAT based formulation for verifying a CNF classifier $\phi_{\hat{Y}}$ with a fundamental limitation of ignoring feature-correlations, which we address now. Let ϕ_{BN} be a CNF formula that encodes the Bayesian Network BN. Following (Ghosh et al., 2021), we discuss an encoding that compute the probability of positive prediction of the classifier for the most favored sensitive group with additional consideration of features-correlation. Let $\mathbf{Q}$ denote an ordered set of quantifiers over variables in the conjoined CNF $\phi_{\hat{Y}} \wedge \phi_{\mathsf{BN}}$. We then construct a SSAT formula $\Phi = (\mathbf{Q}, \phi_{\hat{Y}} \wedge \phi_{\mathsf{BN}})$ and solve it for computing $\max_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$. For computing $\min_{\mathbf{a}} \Pr[\hat{Y} = 1 | \mathbf{A} = \mathbf{a}]$ for the least favored group ,only difference is in the construction of $\mathbf{Q}$. We next discuss the construction of both ϕ_{BN} and $\mathbf{Q}$.

Encoding a Bayesian Network as a CNF Formula. Our goal is to encode the Bayesian network $\mathsf{BN} = (G, \theta)$ into a CNF formula ϕ_{BN} such that *the weighted model count* of ϕ_{BN} exactly computes a joint probability distribution (Chavira and Darwiche, 2008). We note that SSAT does not allow conditional probabilities of randomized quantified variables trivially. Hence, ϕ_{BN} contains *additional variables* to capture the conditional probabilities, as discussed next.

Let $G = (\mathbf{V}, E)$ where $\mathbf{V} \subseteq \mathbf{X} \cup \mathbf{A}$ and $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$. For each network variable $V_i \in \mathbf{V}$, we define a Boolean *indicator* variable λ_{V_i} such that $\Pr[\lambda_{V_i}] \triangleq \Pr[V_i]$. We add following constraint in ϕ_{BN} to establish the relation between λ_{V_i} and V_i.

$$\lambda_{V_i} \leftrightarrow V_i, \tag{5}$$

Intuitively, both λ_{V_i} and V_i are either true or false. This constraint can be trivially translated to clauses in CNF using the equivalence rule $a \leftrightarrow b \equiv (\neg a \vee b) \wedge (a \vee \neg a)$ for Boolean variables a, b.

We now show encoding of conditional probabilities induced by parameter θ. Let $V_i \in \mathbf{V}$ be a vertex in G where $\mathrm{Pa}(V_i) \neq \emptyset$ be V_i's parents and $|\mathrm{Pa}(V_i)| = k$. Additionally, let v and $\mathbf{u} \triangleq [u_1, .., u_k]$ be an assignment of V_i and $\mathrm{Pa}(V_i)$, respectively. To encode $\Pr[V_i = v | \mathrm{Pa}(V_i) = \mathbf{u}]$, we introduce auxiliary variable $\lambda_{v,\mathbf{u}}$ and add following constraints in ϕ_{BN}.

$$\lambda_{v,\mathbf{u}} \wedge \bigwedge_{j=1}^{k} \lambda_{u_j} \rightarrow \lambda_v \tag{6}$$

$$\neg\lambda_{v,\mathbf{u}} \wedge \bigwedge_{j=1}^{k} \lambda_{u_j} \rightarrow \neg\lambda_v \tag{7}$$

where $\lambda_v \equiv \lambda_{V_i}$. Moreover, λ_{u_j} is the indicator variable corresponding to the j^{th} parent in $\text{Pa}(V_i)$. In the above two constraints, for a fixed assignment $\mathbf{u}$ of parents $\text{Pa}(V_i)$, both λ_v and $\lambda_{v,\mathbf{u}}$ are either true or false. Hence, these two constraints encode the conditional probability of $V_i = v$ given $\text{Pa}(V_i) = \mathbf{u}$ using $\Pr[\lambda_{v,\mathbf{u}}] = \Pr[V_i = v|\text{Pa}(V_i) = \mathbf{u}]$. Both constraints can be translated to CNF clauses trivially. For example, Eq. 6 is translated as $\neg\lambda_{v,\mathbf{u}} \vee \bigvee_{j=1}^{k} \neg\lambda_{u_j} \vee \lambda_v$. We next analyze the complexity of ϕ_{BN} in terms of the number of variables and clauses.

Lemma 4 *For a Bayesian network $\mathsf{BN} = (G, \theta)$ defined over n Boolean network variables, the encoded CNF formula ϕ_{BN} has $n + |\theta|$ variables and $2(n + |\theta|)$ clauses.*

Proof Since the DAG in the Bayesian network has n vertices, we consider n indicator variables. Moreover, for encoding conditional probabilities, we consider $|\theta|$ auxiliary variables where parameter θ denotes the number of distinct conditional probabilities in the network. Hence, total variables in ϕ_{BN} is $n + |\theta|$.

According to Eq. (5), (6), (7), there are $2(n+|\theta|)$ clauses in ϕ_{BN} as discussed in Section F.2. ■

Constructing Quantifiers Q. We now discuss the ordered set of quantifiers for a SSAT formula containing CNF $\phi_{\hat{Y}} \wedge \phi_{\mathsf{BN}}$—the solution of which constitutes the maximum (minimum) probability of positive prediction of a CNF classifier. $\phi_{\hat{Y}} \wedge \phi_{\mathsf{BN}}$ contains four different variables : (i) sensitive variables $\mathbf{A}$, (ii) non-sensitive variables $\mathbf{X}$, (iii) indicator variables λ_{V_i}, and (iv) auxiliary variables $\lambda_{v,\mathbf{u}}$. Among them, (iii) and (iv) are associated with ϕ_{BN} and the rest for $\phi_{\hat{Y}}$. For computing the maximum probability of positive prediction of the classifier, we construct an exists-random-exists (ERE) SSAT formula with quantifiers $\mathbf{Q}$ as follows: we set sensitive features $\mathbf{A}$ with existential quantifiers in the beginning of $\mathbf{Q}$ followed by $\lambda_{V_i}, \lambda_{v,\mathbf{u}}$ and $X_j \in \mathbf{X} \setminus \mathbf{V}$ with randomized quantifiers. Since sensitive variables are existentially quantified—similar to stochastic subset-sum problem—the solution of SSAT formula is maximized. Finally, remaining variables appearing in the Bayesian network such as $X_i \in \mathbf{V}$ are existentially quantified in $\mathbf{Q}$ as their assignment is fixed by indicator variables λ_{V_i}. In contrast, for computing the minimum probability of positive prediction of the classifier, we consider an universal-random-exists (URE) SSAT formula where we set sensitive features $\mathbf{A}$ as universal quantifiers with all other quantifiers remaining same.

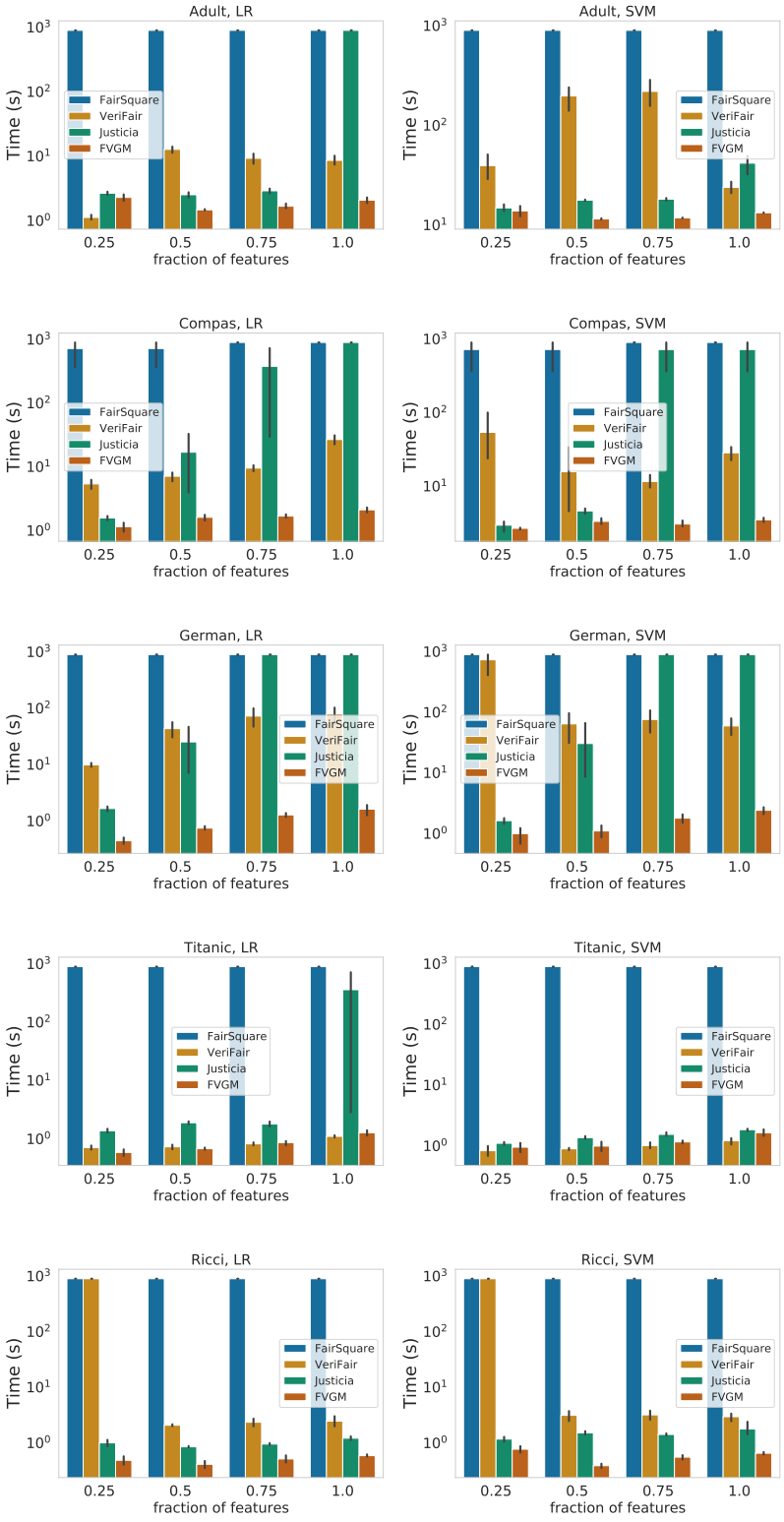


Figure 8: Effect of number of features on the runtime of different datasets for LR and SVM classifiers.

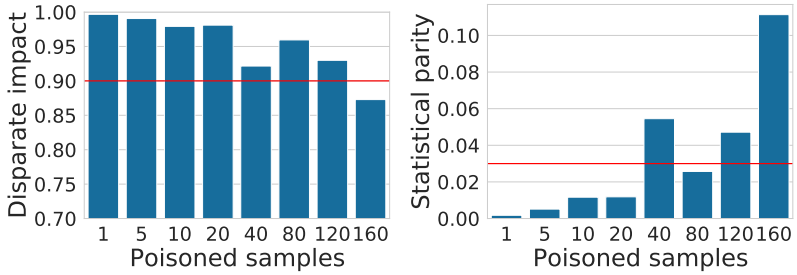


Figure 9: Verifying fairness poisoning attack using FVGM. The red line denotes safety margin, which being exceeded denotes system-vulnerability by the attack algorithm. As the number of poisoned samples increase, disparate impact (DI) decreases and statistical parity (SP) increases.

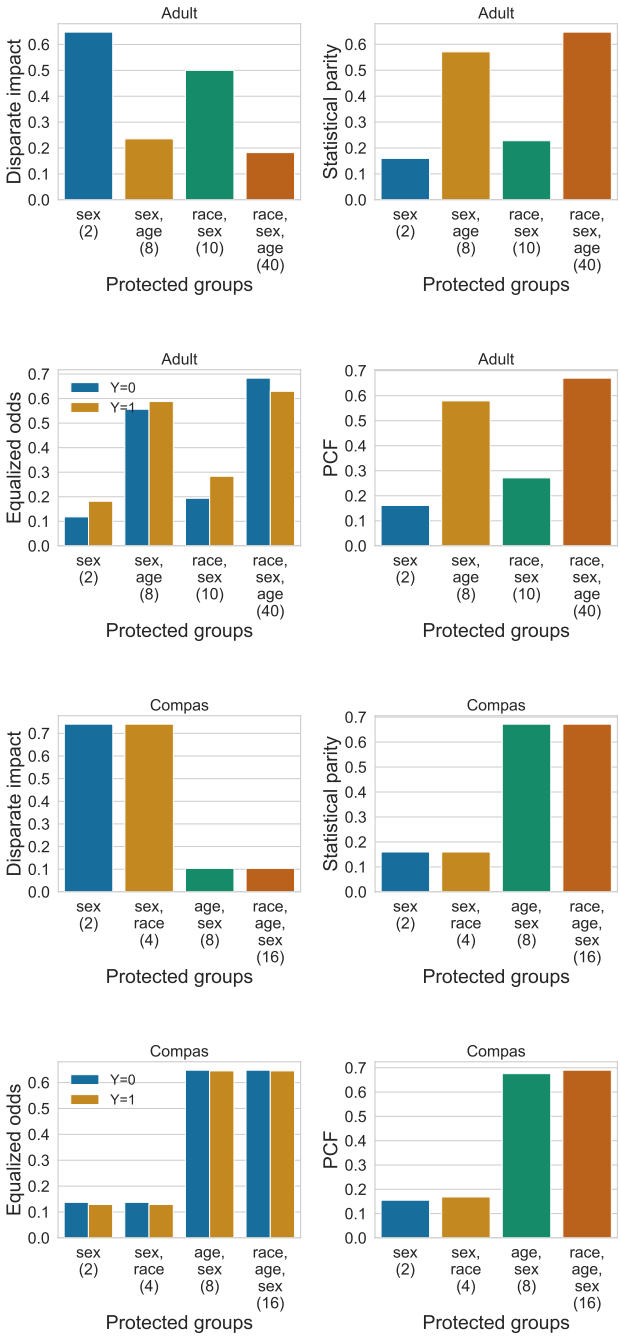


Figure 10: Verifying compound sensitive groups with respect to multiple fairness metrics. In each plot, the X-axis shows sensitive features with the number of compound groups (within parenthesis) and Y-axis shows computed group and causal fairness metrics.

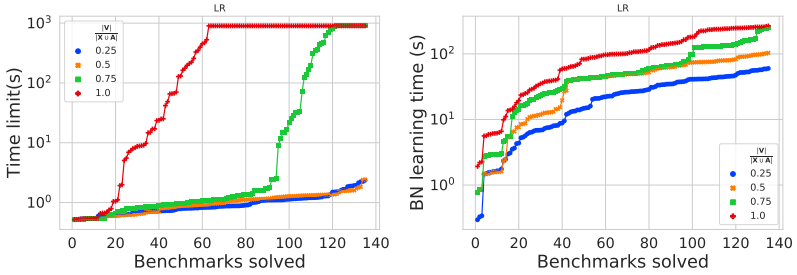
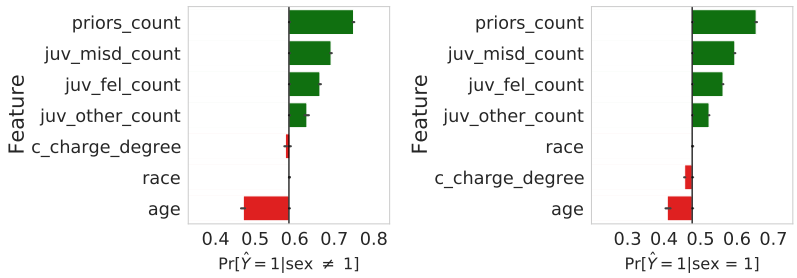
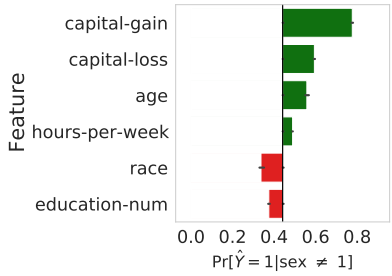


Figure 11: Effect of number of variables in the learned Bayesian Network on computation time of FVGM. In both plots, we vary $\frac{|\mathbf{V}|}{|\mathbf{X} \cup \mathbf{A}|}$, that is the ratio between the number of variables in the Bayesian Network to the number of features. We observe that as this ratio increases to 1, both runtime of FVGM (left plot) and network learning time (right plot) increase.

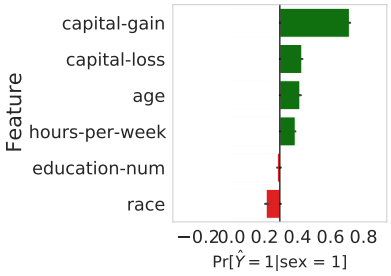


(a) COMPAS

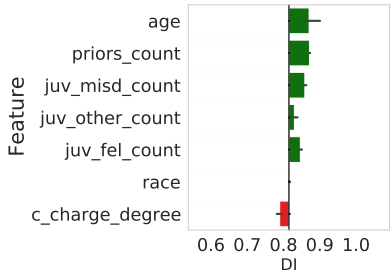
(b) COMPAS



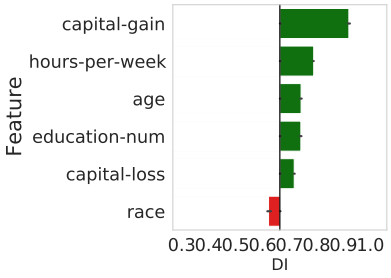
(c) Adult



(d) Adult



(e) COMPAS



(f) Adult

Figure 12: Extended results on computing feature influence functions (FIF) for COMPAS and Adult dataset.

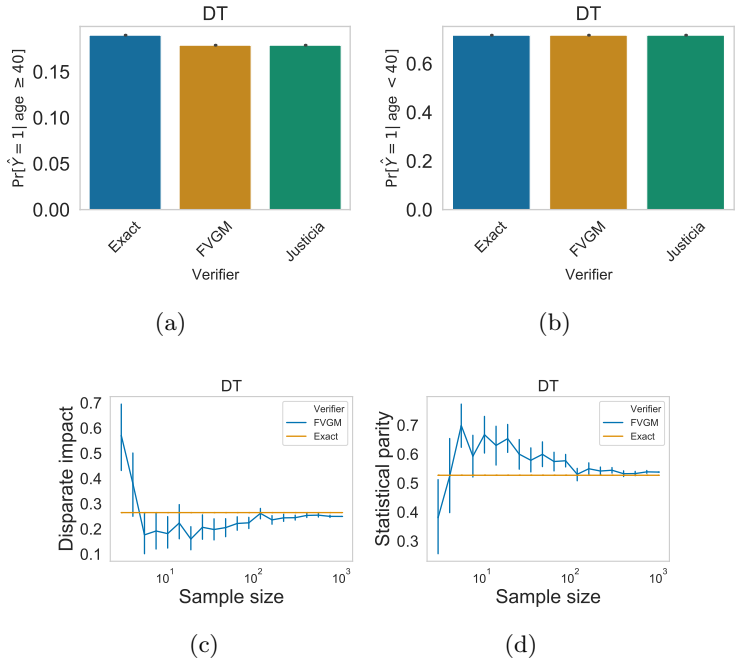


Figure 13: Computing the probability of positive prediction of Decision Tree (DT) classifiers using FVGM and Justicia. FVGM incorporates correlated features represented as a Bayesian network in Figure 7a. We also present the effect of sample size on different fairness metrics such as DI and SP for DT classifier.