Automated Directed Fairness Testing

Sakshi Udeshi
Singapore Univ. of Tech. and Design
Singapore
sakshi_udeshi@mymail.sutd.edu.sg

Pryanshu Arora
BITS Pilani
India
pryanshu23@gmail.com

Sudipta Chattopadhyay
Singapore Univ. of Tech. and Design
Singapore
sudipta_chattopadhyay@sutd.edu.sg

ABSTRACT

Fairness is a critical trait in decision making. As machine-learning models are increasingly being used in sensitive application domains (e.g. education and employment) for decision making, it is crucial that the decisions computed by such models are free of unintended bias. But how can we automatically validate the fairness of arbitrary machine-learning models? For a given machine-learning model and a set of sensitive input parameters, our Aequitas approach automatically discovers discriminatory inputs that highlight fairness violation. At the core of Aequitas are three novel strategies to employ probabilistic search over the input space with the objective of uncovering fairness violation. Our Aequitas approach leverages inherent robustness property in common machine-learning models to design and implement scalable test generation methodologies. An appealing feature of our generated test inputs is that they can be systematically added to the training set of the underlying model and improve its fairness. To this end, we design a fully automated module that guarantees to improve the fairness of the model.

We implemented Aequitas and we have evaluated it on six state-of-the-art classifiers. Our subjects also include a classifier that was designed with fairness in mind. We show that Aequitas effectively generates inputs to uncover fairness violation in all the subject classifiers and systematically improves the fairness of respective models using the generated test inputs. In our evaluation, Aequitas generates up to 70% discriminatory inputs (w.r.t. the total number of inputs generated) and leverages these inputs to improve the fairness up to 94%.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**;

KEYWORDS

Software Fairness, Directed Testing, Machine Learning

ACM Reference Format:
Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated Directed Fairness Testing. In *Proceedings of the 2018 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18), September 3–7, 2018, Montpellier, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3238147.3238165

ASE '18, September 3–7, 2018, Montpellier, France

ACM ISBN 978-1-4503-5937-5/18/09…$15.00
https://doi.org/10.1145/3238147.3238165

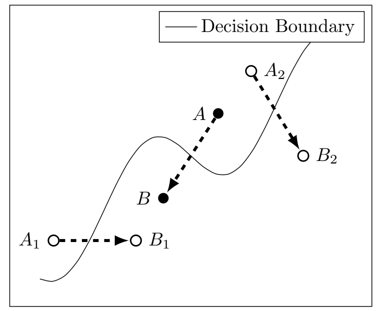


Figure 1: Classifier fairness

1 INTRODUCTION

Nondiscrimination is one of the most critical factors for social protection and equal human rights. The basic idea behind non-discrimination is to eliminate any societal bias based on sensitive attributes, such as race, gender or religion. For example, it is not uncommon to discover the declaration of following non-discrimination policy in universities [12]:

> *"The University is committed to a policy of equal opportunity for all persons and does not discriminate on the basis of race, color, national origin, age, marital status, sex, sexual orientation, gender identity, gender expression, disability, religion, height, weight, or veteran status in employment, educational programs and activities, and admissions"*

Due to the massive progress in machine learning in the last few decades, its application has now escalated over a variety of sensitive domains, including education and employment. The key insight is to primarily automate decision making via machine-learning models. On the flip side, such models may introduce unintended societal bias due to the presence of bias in their training dataset. This, in turn, violates the non-discrimination policy that the respective organization or the nation is intended to fight for. The validation of machine-learning models, to check for possible discrimination, is therefore critically important.

In this paper, we are concerned about the case that any two individuals who are similar with respect to a job at hand should also be treated in a similar fashion during decision making. Thus, we focus towards *individual fairness*, as it is critical for eliminating societal bias and aim to check for discrimination that might violate *individual fairness* [2]. The precise nature of such discrimination depends on the machine-learning model and its input features. Consequently, given a machine-learning model and the input features of the model, it is possible to systematically explore the input space and discover inputs that induce discrimination. We call such inputs *discriminatory inputs.* The primary objective of this paper is to design scalable techniques that facilitate rapid discovery of discriminatory inputs. In particular, given a machine-learning model and a set of discriminatory input features (e.g. race, religion, etc.), our Aequitas approach *automatically discovers inputs to clearly highlight the discriminatory nature of the model under test.*

As an example, consider the decision boundary of a classifier shown in Figure 1. Assume the two points **A** and **B** that differ only in being $Gender_A$ or $Gender_B$. Despite being vastly similar, except in the gender aspect, the model classifies the points **A** and **B** differently. If we consider that such a classifier is used to predict the level of salary, then it certainly introduces unintended societal bias based on gender. Such unfair social biases not only affect the decisions of today but also might amplify it for future generations. The reason behind the discrimination (i.e. unfairness), as shown between points **A** and **B**, can be due to outdated training data that unintentionally introduces bias in certain attributes of the classifier model, e.g., gender in Figure 1. Using our Aequitas approach, we automatically discover the existence of inputs similar to **A** and **B** with high probabilities. These inputs, then, are used to systematically retrain the model and reduce its unfairness.

The reason Aequitas works is due to its directed strategy for test generation. In particular, Aequitas exploits the inherent robustness property of common machine learning models for systematically directing test generation. As a result of this robustness property, the models should exhibit low variation in their output(s) with small perturbations in their input(s). For example, consider the points $\mathbf{A}_1$ and $\mathbf{A}_2$ which are in the neighbourhood of the point **A**. Since the point **A** exhibits discriminatory nature, it is likely that both points $\mathbf{A}_1$ and $\mathbf{A}_2$ will be discriminatory, as reflected via the presence of points $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively. In our Aequitas approach, we first randomly sample the input space to discover the presence of discriminatory inputs (e.g. point **A** in Figure 1). Then, we search the neighbourhood of these inputs, as discovered during the random sampling, to find the presence of more inputs (e.g. points $\mathbf{A}_1$ and $\mathbf{A}_2$ in Figure 1) of the same nature.

An appealing feature of Aequitas is that it leverages the generated test inputs and systematically retrains the machine-learning model under test to reduce its unfairness. The retraining module is completely automatic and it therefore acts as a significant aid to the software engineers to improve the (individual) fairness of machine-learning models. The directed test generation and automated retraining set Aequitas apart from the state-of-the-art in fairness testing [5]. While existing work [5] also considers test generation, such tests were generated randomly. If the discriminatory inputs are located only in specialized locations of the input space, then random test generators are unlikely to be effective in finding individuals discriminated by the corresponding model. To this end, Aequitas empirically validates that a directed test generation, to uncover the discriminatory input regions, is indeed more desirable than random test generation. Moreover, Aequitas provides statistical evidence that if it fails to discover any discriminatory input, then the machine-learning model under test is fair with high probability.

The remainder of the paper is organized as follows. After providing an overview of Aequitas (Section 2), we make the following contributions:

(1) We present Aequitas, a novel approach to systematically generate discriminatory test inputs and uncover the fairness violation in machine-learning models. To this end, we propose three different strategies with varying levels of complexity (Section 4).
(2) We present a fully automated technique to leverage the generated discriminatory inputs and systematically retrain the machine-learning models to improve its fairness (Section 4).
(3) We provide an implementation of Aequitas based on python. Our implementation and all experimental data are publicly available (Section 5).
(4) We evaluate our Aequitas approach with six state-of-the-art classifiers including a classifier that was designed with fairness in mind. Our evaluation reveals that Aequitas is effective in generating discriminatory inputs and improving the fairness of the classifiers under test. In particular, Aequitas generated up to 70% discriminatory inputs (w.r.t. the total number of inputs generated) and improved the fairness up to 94% (Section 5).

After discussing the related work (Section 6), we outline different threats to validity (Section 7) before conclusion and consequences (Section 8).

2 BACKGROUND

In this section, we will discuss the critical importance of fairness testing and outline the key insight behind our approach.

Importance of fairness The usage of machine learning is increasingly being observed in areas that are under the purview of anti-discrimination laws. In particular, application domains such as law enforcement, credit, education and employment can all benefit from machine learning. Hence, it is crucial that decisions influenced by any machine-learning model are free of any unnecessary bias.

As an example, consider a machine-learning model that predicts the income levels of a person. It is possible that such a model was trained on a dataset, which, in turn was unfairly biased to a certain gender or a certain race. As a result, for all equivalent characteristics, barring the gender or race, the credit worthiness of a person will be predicted differently by this model. If financial institutions used such a model to determine the credit worthiness of an individual, then individuals might be disqualified only on the basis of their gender or race. Such a discrimination is certainly undesirable, as it reinforces and amplifies the unfair biases that we, as a society are continuously fighting against.

Fairness in Aequitas Aequitas aims to discover the violation of *individual fairness* [2] in machine-learning models. This means, Aequitas aims to find instances of pair of inputs I and I' that are classified differently despite being vastly similar. The similarity between inputs I and I' is based on a set of potentially discriminatory input parameters (see Definition 1). Detecting the violation of individual fairness is challenging. This is because inputs that are prone to the violation of individual fairness might be located only in specific regions of the input space of a model. Consequently, specialized and directed techniques are required to rapidly locate these input regions. This is the primary motivation behind the development of Aequitas. For the rest of the paper, we will simply use the term fairness (instead of individual fairness) in the light of our Aequitas approach (see Definition 1).

Towards fair machine-learning models A naive approach to design *fair* machine-learning models is to ignore certain sensitive attributes such as race, color, religion, gender, disability, or family status. It is natural to assume that if such attributes are held back from decision making, then the respective model will not discriminate. Unfortunately, such an approach of accomplishing fairness through *blindness* fails. This is because of the presence of redundant encoding in the training dataset [15]. Due to the redundant encoding, it is frequently possible to predict the unknown (sensitive) attributes from other seemingly innocuous features. For example, consider certain ethnic groups in a city that are geographically bound to certain areas. In such cases, even if a machine-learning model in a financial institute does not use ethnicity as a parameter to decide credit worthiness, it is possible to guess ethnicity from geographic locations, which indeed might be a parameter for the model. Therefore, it is critical to systematically test a machine-learning model to validate its fairness property.

Why fairness testing is different In contrast to classic software testing, testing machine-learning models face additional challenges. Typically, these models are deployed in contexts where the formal specification of the software functionality is difficult to develop. In fact, such models are designed to learn from existing data because of the challenges in creating a mathematical definition of the desired software properties. Moreover, an erroneous software behaviour can be rectified by retraining the machine-learning models. However, for classic software, a software bug is typically fixed via modifying the responsible code.

State-of-the-art in fairness testing The state-of-the-art in systematic testing of software fairness is still at its infancy. In contrast to existing work [5], Aequitas focuses on directed test generation strategy. As evidenced by our evaluation, this is crucial to locate specific input regions that violate individual fairness. To illustrate our objective, consider a machine-learning model f and its inputs I and I'. I differs from I' only in being assigned a different value in a potentially discriminatory input parameter. For example, if *gender* is the potentially discriminatory input parameter, then I will be different from I' only in being $Gender_A$ or $Gender_B$. We are interested to discover inputs I or I', where the difference in outputs of the model, captured via $|f(I) - f(I')|$, is beyond a pre-determined threshold. We call such inputs I or I' to be *discriminatory inputs* for the model f. It is important to note that the discrimination threshold and the potentially discriminatory input parameters are supplied by the users of our tool. In the preceding example, the potentially discriminatory input parameter, i.e., *gender* can be specified by the user. Similarly, users can also fine tune the value at which $|f(I) - f(I')|$ is considered to be discriminatory.

Robustness in machine learning Robustness is a notion that says that the output of a machine-learning model is not dramatically affected by small changes to its input [3]. Assume a model f, let i be the input to f and δ be a small value. If f is robust, then $f(i) \approx f(i + \delta)$. Nevertheless, existing techniques provide evidence to find inputs that violate this robustness property. Such inputs are called adversarial inputs [14] [7] [13]. However, adversarial inputs generally cover only a small fraction of the entire input space. This is evident by the fact that adversarial inputs need to be crafted using very specialized techniques. Additionally, Aequitas is designed to avoid these adversarial input regions by systematically directing the test generators. Intuitively, Aequitas achieves this by reducing the probability to explore an input region when tested inputs from the region did not exhibit discriminatory nature (see Algorithm 2 for details). Consequently, if adversarial or non-robust input regions do not exhibit discriminatory nature, such regions will eventually be explored only with very low probability.

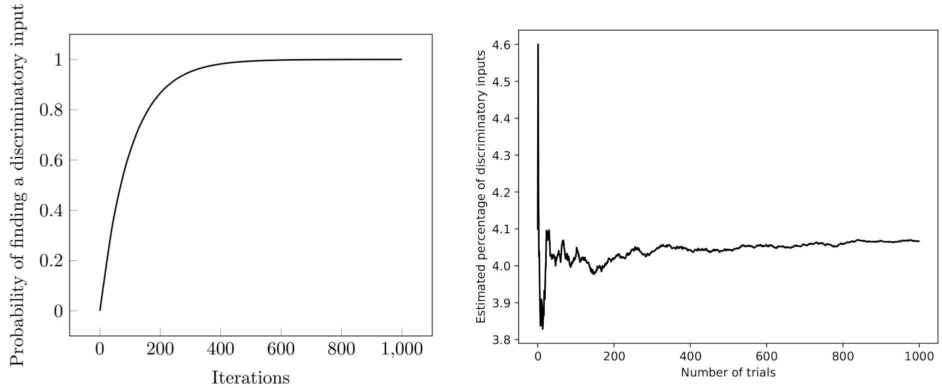
3 APPROACH AT A GLANCE

We propose, design and evaluate three schemes, with varying levels of complexities, to systematically uncover software fairness problems. The crucial components of our approach are outlined below.

Global search In the first step of all our proposed schemes, we uniformly sample the inputs and record the discriminatory inputs that we find. In the light of uniformly sampling the input space, we can guarantee, with very high probability, to discover a discriminatory input, if such an input exists. For instance, Figure 2(a) highlights the probability of finding a discriminatory input in an input space with only 1% discriminatory inputs. Therefore, if discriminatory inputs exist, the first step of our proposed schemes guarantee to find at least one such input with high probabilities.

Local search The second step of our proposed schemes share the following hypothesis: *If there exists a discriminatory input $I \in \mathbb{I}$, where $\mathbb{I}$ captures the input domain, then there exist more discriminatory inputs in the input space closer to I*. The input domain $\mathbb{I}$ can be considered as the cartesian product of the domain of n input parameters, say $P_1, P_2, \ldots, P_n$. We assume $\mathbb{I}_k$ captures the domain of input parameter P_k. Therefore, $\mathbb{I} = \mathbb{I}_1 \times \mathbb{I}_2 \times \ldots \times \mathbb{I}_n$. An input parameter $p \in \bigcup_{i=1}^{n} P_i$ can be potentially discriminatory if the output of the machine-learning model should not be biased towards specific values in $\mathbb{I}_p$. Without loss of generality, we assume a subset of parameters $P_{disc} \subseteq \bigcup_{i=1}^{n} P_i$ to be potentially discriminatory. For an input $I \in \mathbb{I}$, we use I_k to capture the value of parameter P_k within input I. Based on this notion, we explore the following methods to realize our hypothesis. Our methods differ on how we systematically explore the neighbourhood of a discriminatory input $I^{(d)}$. $I^{(d)}$, in turn, was discovered in the first step of Aequitas.

(1) First a parameter $p \in \bigcup_{i=1}^{n} P_i \setminus P_{disc}$ is randomly chosen. Then a small perturbation (i.e. change) δ is added to $I_p^{(d)}$. Typically $\delta \in \{-1, +1\}$ as we consider integer and real-valued input parameters in our evaluation.
(2) In the second method, we assign probabilities on how to perturb a chosen parameter. A specific parameter $p \in \bigcup_{i=1}^{n} P_i \setminus P_{disc}$ is still chosen uniformly at random. However, if a given perturbation δ of $I_p^{(d)}$ consistently yields discriminatory inputs, then the perturbation δ is employed with higher probability. Since δ typically belongs to a small set of values, such a strategy works efficiently in practice.
(3) The third method augments the second method by refining probabilities to perturb an input parameter. Concretely, if



(a) (b)

Figure 2: (a) Probability of finding discriminatory inputs, (b) Estimation of the percentage of discriminatory inputs

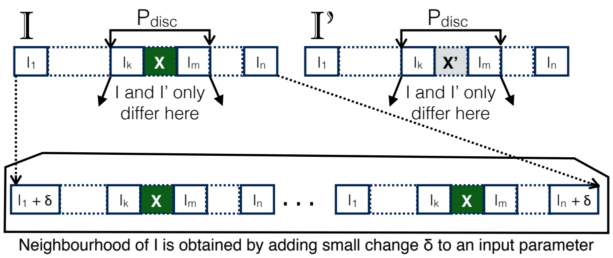


Figure 3: Our Aequitas approach at a glance

perturbing the value of parameter $p \in \bigcup_{i=1}^{n} P_i \setminus P_{disc}$ consistently yields discriminatory inputs, then the parameter p will be significantly more likely to be chosen for perturbation.

Our proposed methodologies are fully automated, they do not require the source code of the models and work efficiently in practice for state-of-the-art classifiers.

Figure 3 illustrates Aequitas approach when I and I' were discovered in the first step. Then, the second step explored the neighbourhood of I by adding small changes δ to an input parameter.

Estimation of discriminatory inputs An appealing feature of Aequitas is that we can estimate the percentage of discriminatory inputs in $\mathbb{I}$. To this end, we leverage the law of large numbers (LLN) in probability theory. In particular, we generate K inputs uniformly at random and check whether they can lead to discriminatory inputs. Assume that $K' \leq K$ inputs turn out to be discriminatory. We compute the ratio $\frac{K'}{K}$ over a large number of trials. According to LLN, the average of these ratios closely approximates the actual percentage of discriminatory inputs in $\mathbb{I}$. Figure 2(b) highlights such convergence after only 400 trials when K was chosen to be 1000.

Why Aequitas works? The reason Aequitas works is because of the robustness property of common machine-learning models. In particular, if we perturb the input to a model by some small δ, then the output is not expected to change dramatically. As we expect the machine-learning models under test to be relatively robust, we can leverage their inherent robustness property to systematically generate test inputs that exhibit similar characteristics. In our Aequitas approach, we focus on the discriminatory nature of a given input. We aim to discover more discriminatory inputs in the proximity of an already discovered discriminatory input leveraging the robustness property.

Table 1: Notations used in Aequitas approach

n	The number of input parameters to the machine-learning model under test
$\mathbb{I}$	The input domain of the model
P_i	The i-th input parameter of the model
P	Set of all input parameters, i.e., $P = \bigcup_{i=1}^{n} P_i$
P_{disc}	Set of sensitive or potentially discriminatory input parameters (e.g. gender). Clearly, $P_{disc} \subseteq \bigcup_{i=1}^{n} P_i$
I_p	The value of input parameter p in input $I \in \mathbb{I}$
γ	A pre-determined discrimination threshold

How Aequitas can be used to improve software fairness? We have designed a fully automated module that leverages on the discriminatory inputs generated by Aequitas and retrains the machine-learning model under test. We empirically show that such a strategy provides useful capabilities to a developer. Specifically, our Aequitas approach automatically improves the fairness of machine-learning models via retraining. For instance, in certain decision tree classifiers, our Aequitas approach reduced the fraction of discriminatory inputs up to 94%.

4 DETAILED APPROACH

In this section, we discuss our Aequitas approach in detail. To this end, we will use the notations captured in Table 1.

Our approach revolves around discovering *discriminatory inputs* via systematic *perturbation*. We introduce the notion of discriminatory inputs and perturbation formally before delving into the algorithmic details of our approach.

Definition 1. ***(Discriminatory Input and fairness)*** *Let f be a classifier under test, γ be the pre-determined discrimination threshold (e.g. chosen by the user), and $I \in \mathbb{I}$. Assume $I' \in \mathbb{I}$ such that there exists a non-empty set $Q \subseteq P_{disc}$ and for all $q \in Q$, $I_q \neq I'_q$ and for all $p \in P \setminus Q$, $I_p = I'_p$. If $|f(I) - f(I')| > \gamma$, then I is called a discriminatory input of the classifier f and is an instance that manifests the violation of (individual) fairness in f.*

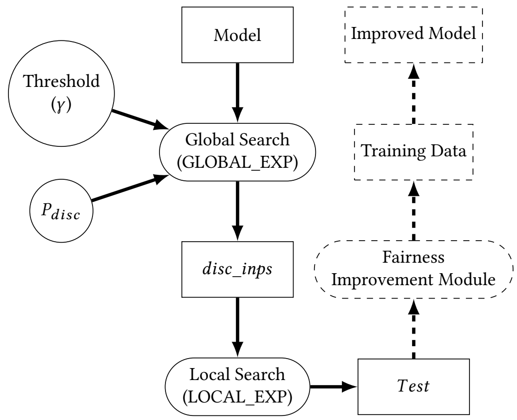


Figure 4: An overview of our AEQUITAS approach

Definition 2. ***(Perturbation)*** *We define perturbation g as a function $g : \mathbb{I} \times (P \setminus P_{disc}) \times \Gamma \rightarrow \mathbb{I}$ where $\Gamma = \{-1, +1\}$ captures the set of directions to perturb an input parameter. If $I' = g(I, p, \delta)$ where $I \in \mathbb{I}$, $p \in P \setminus P_{disc}$ and $\delta \in \Gamma$, then $I'_p = I_p + \delta$ and for all $q \in P \setminus \{p\}$, we have $I'_q = I_q$.*

It is worthwhile to mention that the set of directions to perturb an input parameter, i.e. Γ can easily be extended with more possibilities to perturb. Besides, it can also be customized with respect to different input parameters. However, for the sake of brevity, we will stick with the simplified version stated in Definition 2.

An overview of our overall approach appears in Figure 4. The main contribution of this paper is an automated test generator to discover fairness violation. This involves two stages: 1) global search (*GLOBAL_EXP*) and 2) local search (*LOCAL_EXP*) over the input domain $\mathbb{I}$. Optionally, the generated test inputs can be leveraged to retrain the model under test and improve fairness.

In the following, we will describe the crucial components of our AEQUITAS approach, as shown in Figure 4.

4.1 Global Search

The motivation behind our global search (cf. procedure GLOBAL_EXP in Algorithm 1) is to discover some points in $\mathbb{I}$ that can be used to drive our local search algorithm. To this end, we first select an input I randomly from the input domain. Input I, then, is used to generate a set of inputs that cover all possible values of sensitive parameters $P_{disc} \subseteq P$. This leads to a set of inputs $\mathbb{I}^{(d)}$. We note that the set of sensitive parameters (e.g. race, religion, gender) P_{disc} typically has a small size. Therefore, despite the exhaustive nature of generating $\mathbb{I}^{(d)}$, this is practically feasible. Finally, we discover the discriminatory inputs (cf. Definition 1) within $\mathbb{I}^{(d)}$ and use the resulting discriminatory input set for further exploration during our local search over $\mathbb{I}$.

4.2 Local Search

In this test generation phase, we take the inputs generated by our global search (i.e. *disc_inputs*) and then search in the neighbourhood of *disc_inputs* to discover other inputs with similar characteristics (cf. procedure LOCAL_EXP in Algorithm 2). Our search strategy is motivated from the robustness property inherent in common machine-learning models. According to the notion of robustness, the neighbourhood of an input should produce similar output. Therefore, it becomes logical to search the neighbourhood of *disc_inputs*, as these are the discriminatory inputs and their neighbourhood are likely to be discriminatory for robust models.

To search the neighbourhood of *disc_inputs*, AEQUITAS perturbs an input $I \in disc\_inputs$ by changing the value of some parameter $p \in P \setminus P_{disc}$ (i.e. I_p). The value of the parameter p is perturbed by $\delta \in \{-1, +1\}$. We note that as a side-effect of changing I_p, input I is automatically modified. This modified version of I is further perturbed in subsequent iterations of the inner loop in Algorithm 2. Our AEQUITAS approach chooses a parameter $p \in P \setminus P_{disc}$ with probability $\sigma_{pr}[p]$ (cf. Algorithm 2). For all $p \in P \setminus P_{disc}$, initially $\sigma_{pr}[p]$ was assigned to $\frac{1}{|P \setminus P_{disc}|}$. Once p is chosen its value is perturbed by $\delta = -1$ with probability $\sigma_v[p]$ and by $\delta = +1$ with probability $1 - \sigma_v[p]$. $\sigma_v[p]$ is initialized to 0.5 for all parameters in $p \in P \setminus P_{disc}$.

Algorithm 1 Global Search

procedure GLOBAL_EXP(P, P_{disc})
  $disc\_inps \leftarrow \phi$
  ▷ N is the number of trials in global search
  for i in (0, N) **do**
    Select an input $I \in \mathbb{I}$ at random
    ▷ $\mathbb{I}^{(d)}$ extends I with all possible values of P_{disc}
    $\mathbb{I}^{(d)} \leftarrow \{I' \mid \forall p \in P \setminus P_{disc}.\ I_p = I'_p\}$
    if $(\exists I, I' \in \mathbb{I}^{(d)}, |f(I) - f(I')| > \gamma)$ **then**
      $disc\_inps \leftarrow disc\_inps \cup \{I\}$
    end if
  end for
  return $disc\_inps$
end procedure

Algorithm 2 Local Search

procedure LOCAL_EXP($disc\_inps$, P, P_{disc}, Δ_v, Δ_{pr})
  Test $\leftarrow \phi$
  Let $P' = P \setminus P_{disc}$
  Let $\sigma_{pr}[p] = \frac{1}{|P'|}$ for all $p \in P'$
  Let $\sigma_v[p] = 0.5$ for all $p \in P'$
  for $I \in disc\_inps$ **do**
    ▷ N is the number of trials in local search
    for i in (0, N) **do**
      Select $p \in P'$ with probability $\sigma_{pr}[p]$
      Select $\delta = -1$ with probability $\sigma_v[p]$
      ▷ Note that I is modified as a side-effect of modifying I_p
      $I_p \leftarrow I_p + \delta$
      ▷ $\mathbb{I}^{(d)}$ extends I with all values of P_{disc}
      $\mathbb{I}^{(d)} \leftarrow \{I' \mid \forall p \in P \setminus P_{disc}.\ I_p = I'_p\}$
      if $(\exists I, I' \in \mathbb{I}^{(d)}, |f(I) - f(I')| > \gamma)$ **then**
        ▷ Add the perturbed input I
        Test $\leftarrow$ *Test* $\cup \{I\}$
      end if
      update_prob(I, p, *Test*, δ, Δ_v, Δ_{pr})
    end for
  end for
  return *Test*
end procedure

Algorithm 3 Aequitas semi-directed update probability

procedure UPDATE_PROB(I, p, *Test*, δ, Δ_v, Δ_{pr})
  if $(I \in Test \wedge \delta = -1) \vee (I \notin Test \wedge \delta = +1)$ **then**
    $\sigma_v[p] \leftarrow \min(\sigma_v[p] + \Delta_v, 1)$
  end if
  if $(I \notin Test \wedge \delta = -1) \vee (I \in Test \wedge \delta = +1)$ **then**
    $\sigma_v[p] \leftarrow \max(\sigma_v[p] - \Delta_v, 0)$
  end if
end procedure

Algorithm 4 Aequitas fully-directed update probability

procedure UPDATE_PROB(I, p, *Test*, δ, Δ_v, Δ_{pr})
  if $(I \in Test \wedge \delta = -1) \vee (I \notin Test \wedge \delta = +1)$ **then**
    $\sigma_v[p] \leftarrow \min(\sigma_v[p] + \Delta_v, 1)$
  end if
  if $(I \notin Test \wedge \delta = -1) \vee (I \in Test \wedge \delta = +1)$ **then**
    $\sigma_v[p] \leftarrow \max(\sigma_v[p] - \Delta_v, 0)$
  end if
  if $I \in Test$ **then**
    $\sigma_{pr}[p] \leftarrow \sigma_{pr}[p] + \Delta_{pr}$
    $\sigma_{pr}[p] \leftarrow \frac{\sigma_{pr}[p]}{\sum_{x \in P \setminus P_{disc}} \sigma_{pr}[x]}$ for all $p \in P \setminus P_{disc}$
  end if
end procedure

Aequitas employs three different strategies, namely Aequitas random, Aequitas semi-directed and Aequitas fully-directed, to update the probabilities in σ_{pr} and σ_v. This is to direct the test generation process with a focus on discovering discriminatory inputs. In the following, we will outline the different strategies implemented within Aequitas.

***Aequitas random*.** Aequitas random does not update the initial probabilities assigned to σ_{pr} and σ_v. This results in δ (i.e. perturbation value) and p (i.e. the parameter to perturb) both being chosen randomly. Intuitively, Aequitas random explores inputs around the neighbourhood of *disc_inputs* (i.e. set of discriminatory inputs discovered via global search) uniformly at random. Nevertheless, Aequitas random empirically outperforms a purely random search over the input space. This is because it still performs a random search in a constrained input region – specifically, the input region that already contains discriminatory inputs.

***Aequitas semi-directed*.** Aequitas semi-directed drives the test generation by systematically updating σ_v, i.e., the probabilities to perturb the value of an input parameter by $\delta = -1$ (cf. Algorithm 3). The parameter p, to perturb, is still chosen randomly. Initially, we choose $\delta \in \{-1, +1\}$ where the probability that $\delta = -1$ is $\sigma_v[p]$ and the probability that $\delta = +1$ is $1 - \sigma_v[p]$. If the perturbed input is discriminatory (cf. Definition 1), then we increase the probability associated with $\sigma_v[p]$ by a pre-determined offset Δ_v. Otherwise, $\sigma_v[p]$ is reduced by the same offset Δ_v. Intuitively, the updates to probabilities in σ_v prioritise a direction $\delta \in \{-1, +1\}$ when the respective direction results in discriminatory inputs.

***Aequitas fully-directed*.** Aequitas fully-directed extends Aequitas semi-directed by systematically updating the probabilities to choose a parameter for perturbation. To this end, we update probabilities in σ_{pr} during the test generation process (cf. Algorithm 4). Assume we pick a parameter $p \in P \setminus P_{disc}$ to perturb. Initially, we have $\sigma_{pr}[p] = \frac{1}{|P \setminus P_{disc}|}$. If the perturbation of the given parameter p by δ results in a discriminatory input, then we add a pre-determined offset Δ_{pr} to $\sigma_{pr}[p]$. To reflect this change in probability, we normalize $\sigma_{pr}[p']$ to $\frac{\sigma_{pr}[p']}{\sum_{x \in P \setminus P_{disc}} \sigma_{pr}[x]}$ for every $p' \in P \setminus P_{disc}$. Intuitively, the updates to probabilities in σ_{pr} prioritize a parameter when perturbing the respective parameter results in discriminatory inputs.

4.3 Estimation using LLN

An attractive feature of Aequitas is that we can estimate the percentage of discriminatory inputs in $\mathbb{I}$ for any given model. We leverage the Law of Large Numbers (LLN) from probability theory to accomplish this. Let Λ be an experiment. In this experiment, we generate m inputs uniformly at random. These are independent and identically distributed (IID) samples $I_1, I_2 \ldots I_m$. We execute these inputs and count the number of inputs that are discriminatory in nature. Let m' be the number of inputs that are discriminatory. Λ then outputs the percentage $\overline{m} = \frac{m' \times 100}{m}$.

Λ is conducted K times. In each instance of the experiment, we collect the outcome $\overline{m}_1, \overline{m}_2 \ldots \overline{m}_K$. Let $\overline{M} = K^{-1} \sum_{i=1}^{K} \overline{m}_i$. According to LLN, the average of the results, i.e. $\overline{M}$, obtained from a large number of trials, should be close to the expected value, and it will tend to become closer as more trials are performed. This implies as,

$$K \rightarrow \infty$$

$$\overline{M} \rightarrow M^*$$

where M^* is the true percentage of the discriminatory inputs present in $\mathbb{I}$ for the machine-learning model under test. This phenomenon was observed in our experiments. Figure 2(b) shows that the $\overline{M}$ converges only after 400 trials (i.e. $K = 400$).

4.4 Improving Model Fairness

It has been observed that generated test inputs showing the violation of desired-properties in machine-learning models can be leveraged for improving the respective properties. This was accomplished via augmenting the training dataset with the generated test inputs and retraining the model [17].

Hence, we intend to evaluate the usefulness of our generated test inputs to improve the model fairness via retraining. To this end, Aequitas has a completely automated module that guarantees reduction of the percentage of discriminatory inputs in $\mathbb{I}$. We achieve this by systematically adding portions of generated discriminatory inputs to the training dataset.

Assume *Test* be the set of discriminatory inputs generated by Aequitas. Aequitas is effective in generating discriminatory inputs and the size of the set *Test* is usually large. A naive approach to retrain the model will be to add all generated discriminatory inputs to the training dataset. Such an approach is likely to fail to improve the fairness of the model. This is because the generated test inputs are targeted towards finding discrimination and are unlikely to follow the true distribution of the training data. Therefore, blindly adding all the test inputs to the training set will bias its distribution towards the distribution of our generated test inputs. To solve this

Algorithm 5 Retraining

```
procedure RETRAINING(f, Test, training_data)
    N ← ∞
    f_cur ← f
    for i in (2, N) do
        p_i ← a random real number between (2^{i−2}, 2^{i−1})
        if p_i > 100 then
            Exit the loop
        end if
        k ← len(training_data)
        n_addn ← (p_i·k)/100
        TD_addn ← randomly selected n_addn inputs from Test
        TD_new ← training_data ∪ TD_addn
        f_new ← model trained using TD_new
        ▹ Estimate the number of discriminatory inputs (section 4.3)
        fair_cur ← LLN_Fairness_Estimation (f_cur)
        fair_new ← LLN_Fairness_Estimation (f_new)
        if (fair_cur > fair_new) then
            f_cur ← f_new
        else
            Exit the loop
        end if
    end for
    return f_cur
end procedure
```

challenge, it is important that only portions of discriminatory inputs from *Test* are added to the training dataset.

Let p_i be the percentage, with respect to the size of the training data, that we choose at any given iteration i. If size of training data is M, then we select $\frac{p_i \cdot M}{100}$ discriminatory inputs from *Test* at random and add these discriminatory inputs to the training dataset. For $i \in [2, N]$, we set p_i randomly in a range between $[2^{i-2}, 2^{i-1}]$. The intuition behind this is to find an efficient mechanism to systematically add inputs from *Test* to the training dataset and to approximate the optimal reduction in discriminatory inputs. We terminate the process when adding inputs from *Test* to the training dataset does not decrease the estimated fraction of discriminatory inputs in $\mathbb{I}$. The currently trained model (i.e. f_{cur} in Algorithm 5) is then taken as the improved model with better (individual) fairness score. In this way, we can guarantee that our retraining process always terminates with a reduction in discriminatory inputs.

Our retraining strategy is designed to be fast without sacrificing the fairness significantly. Our main objective is to demonstrate that Aequitas generated test inputs can indeed be used by the developers to improve the individual fairness of their models. The amount of added test inputs (generated by Aequitas) is chosen from exponentially increasing intervals (i.e. the interval $[2^{i-2}, 2^{i-1}]$ in Algorithm 5). Such a strategy is taken to quickly scope the sensitivity of the model with respect to the generated test data. Moreover, by choosing a random number p_i in the interval, we try not to overshoot the value of p_i by a large margin that causes the optimal reduction of discriminatory inputs in $\mathbb{I}$. As a result, our proposed retraining strategy maintains a balance between improving model fairness and the efficiency of retraining.

It is well known that adding more data to a machine-learning algorithm is likely to lead to increased accuracy [8]. A relevant challenge here is attributed to the labeling of the generated test data. There exists a number of effective strategies to tackle this problem. One such strategy is finding the label via a simple majority of a number of classifiers [10]. Majority voting has been shown to be very effective for a wide range of problems [16] and we believe it should be readily applicable in our context of improving fairness as well. Nevertheless, test data labeling is an orthogonal problem in the domain of machine learning and we consider it to be beyond the scope of the problem targeted by Aequitas.

4.5 Termination

Aequitas can be configured to have various termination conditions depending on the particular use case of the developer. In particular, Aequitas can be terminated with the following possible conditions:

(1) Aequitas can terminate after it has generated a user specified number of discriminatory inputs from $\mathbb{I}$. This feature can be used when a certain number of discriminatory inputs need to be generated for testing, evaluation or retraining of the model.
(2) Aequitas can also terminate within a given time bound. This is useful to quickly check if the model exhibits discrimination for a particular set of sensitive parameters.

In our evaluation, we used both the termination criteria to evaluate the effectiveness and efficiency of Aequitas.

5 RESULTS

Experimental setup. We evaluate Aequitas across a wide variety of classifiers, including a classifier which was designed to be fair. Some salient features of these classifiers are outlined in Table 2. In particular, Fair SVM (cf. Table 2) was specifically designed with fairness in mind [19]. The rest of the classifiers under test are the standard implementations found in Python’s Scikit-learn machine learning library. These classifiers are used in a wide variety of applications by machine-learning engineers across the world.

Other than Fair SVM [19], we have used Scikit-learn’s Support Vector Machines (SVM), Multi Layer Perceptron (MLPC), Random Forest and Decision Tree implementations for our experiments. We also evaluate an Ensemble Voting Classifier (Ensemble), in which we take the combination of two classifier predictions. The classifiers we use are Random Forest and Decision Tree estimators (cf. Table 2).

Table 2: Subject classifiers used to evaluate Aequitas

Classifier name	Lines of python code	Input domain
Fair SVM [19]	913	10^6
SVM	1123	
MLPC	1308	
Random Forest	1951	
Decision Tree	1465	
Ensemble	3466	

All classifiers listed in Table 2 are used for predicting the income. These classifiers are trained with the data obtained from the US census [1]. The size of this training data set is around 32,000. We train all the six classifiers on this training data. The objective is to classify whether the income of an individual is above \$50,000 (captured via classifier output “+1") or below (captured via classifier output “-1"). For all the classifiers, set of discriminatory parameters,

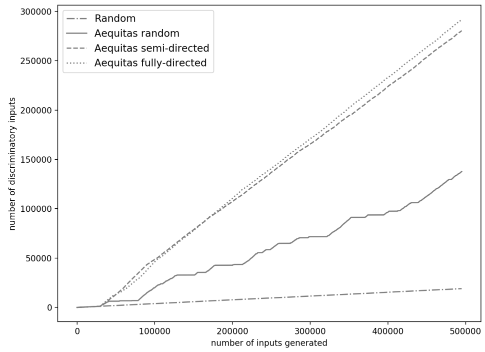


Figure 5: The effectiveness of Aequitas

i.e. P_{disc} is the *gender* of an individual. The threshold value for identifying a discriminatory input is set to zero. This means, if I differs from I' only in being $Gender_A$ or $Gender_B$, then I or I' are discriminatory inputs of a classifier f when $|f(I) - f(I')| \geq 0$. In our experiments we set the perturbation $\delta \in \{-1, +1\}$ and both Δ_v and Δ_{pr} as 0.001. These are user defined variables that guide our Aequitas approach. In particular, these variables are used to systematically refine the probabilities to choose an input parameter to perturb and to choose a perturbation value δ (cf. Section 4).

We implement Aequitas in Python, as it is a popular choice of language for the development of machine-learning models and related applications. The implementation is around 600 lines of python code. All our experiments were performed on an Intel i7 processor having 64GB of RAM and running Ubuntu 16.04.

Key results. We use three different test generation methodologies, namely Aequitas random, Aequitas semi-directed and Aequitas fully-directed. These methodologies differ with respect to the increasing levels of sophistication in systematically searching the input space (cf. Section 4.2). In particular, Aequitas fully-directed involves the highest level of sophistication in searching the input space. As expected, Aequitas fully-directed consistently outperforms the Aequitas random and Aequitas semi-directed, as observed from Figure 5. However, Aequitas fully-directed and Aequitas semi-directed demand more computational resources per unit time than Aequitas random. As a result, Aequitas random is more appropriate to use, as compared to the rest of our approaches, for testing with limited computational resources per unit time. The test subject used in Figure 5 was the Fair SVM (cf. Table 2).

To illustrate the power of our Aequitas approach over the state-of-the-art fairness testing [5], we also compare our approaches with the state-of-the-art, which, in turn is captured via "Random" in Figure 5. It is evident that even the least powerful technique implemented within our Aequitas approach (i.e. Aequitas random) significantly outperforms the state-of-the-art. In our evaluation, we discovered that Aequitas is more effective than the state-of-the-art random testing by a factor of 9.6 on average and up to a factor of 20.4. We measured the effectiveness via the number of discriminatory inputs generated by a test generation technique. Aequitas also provides capabilities to automatically retrain a machine-learning model with the objective to reduce the number of discriminatory inputs. To this end, Aequitas reduced the number of discriminatory inputs by 43.2% on average with a maximum reduction of 94.36%.

RQ1: How effective is Aequitas in finding discriminatory inputs?

We evaluate the capability of Aequitas in effectively generating discriminatory inputs. For all the subject classifiers, we measure the effectiveness of our test algorithms via the number of discriminatory inputs generated with respect to the number of total inputs generated.

A purely random approach is not effective in generating discriminatory inputs. As observed from Figure 5, the number of discriminatory inputs generated by such an approach does not increase rapidly over the number of inputs generated. This is expected, as a purely random approach does not incorporate any systematic strategy to discover inputs violating fairness. The ineffectiveness of random testing persists across all the subject classifiers, as observed in Table 3.

As observed from Table 3, all test generation approaches implemented within Aequitas outperform a purely random approach. In particular, the rate at which our Aequitas approach generates discriminatory inputs is significantly higher than a purely random approach. As a result, Aequitas provides scalable and effective technique for machine learning engineers who aim to rapidly discover fairness issues in their models. Aequitas random, Aequitas semi-directed and Aequitas fully-directed involve increasing level of sophistication in directing the test input generation. As a result, Aequitas fully-directed approach performs the best among all our test generators. In particular, Aequitas semi-directed is on an average 46.7% and up to 64.9% better than Aequitas random. Finally, Aequitas full-directed is on an average 29.5% and up to 56.56% better than Aequitas semi-directed.

By design, Aequitas does not generate any false positives. This means that any discriminatory input generated by Aequitas are indeed discriminatory to the model under test, subject to the chosen threshold of discrimination.

Finding: Aequitas fully-directed approach outperform a purely random approach up to a factor of 20.4 in terms of the number of discriminatory inputs generated. It also performs up to 56.7% better than Aequitas semi-directed, which, in turn performs up to 64.9% better than Aequitas random, our least sophisticated approach.

RQ2: How efficient is Aequitas in finding discriminatory inputs?

Table 4 summarizes how much time each of the methods takes to generate 10,000 discriminatory inputs. On an average Aequitas random performs 64.42% faster than the state of the art. The improvement in Aequitas fully-directed is even more profound. On an average, Aequitas fully-directed is 83.27% faster than the state of the art, with a maximum improvement of 96.62% in the case of Multi Layer Perceptron.

It is important to note that the reported time in Table 4 includes both the time needed for test generation and for test execution.

Table 3: Effectiveness of Aequitas approach

Classifier	Random [5]	Aequitas random		Aequitas semi-directed		Aequitas fully-directed	
	% discriminatory input	% discriminatory input	# inputs generated	% discriminatory input	# inputs generated	% discriminatory input	# inputs generated
Fair SVM	3.45	39.4	315640	65.2	322725	70.32	357375
SVM	0.18	0.53	54683	0.574	88095	1.22	100101
MLPC	0.3466	2.15	218727	2.39	129556	2.896	141666
Random Forest	8.34	18.312	218727	21.722	264523	34.98	282973
Decision Tree	0.485	2.33	153166	2.89	179364	6.653	248229
Ensemble	8.23	22.34	187980	36.08	458910	37.9	545375

Table 4: Test generation efficiency

Classifier	Random	Aequitas random	Aequitas semi-directed	Aequitas fully-directed
Fair SVM	1589.87s	534.47s	345.65s	228.14s
SVM	7159.54s	3589.9s	2673.8s	2190.21s
MLPC	6157.23s	759.63s	431.76s	207.87s
Random Forest	9563.12s	2692.98s	1334.67s	1145.34s
Decision Tree	1035.32s	569.13s	371.89s	254.25s
Ensemble	6368.79s	2178.45s	1067.75s	989.43s

Hence, the reported time is highly dependent on the execution time of the model under test.

> **Finding:** Aequitas fully-directed is 83.27% faster than the state of the art, with a maximum improvement of 96.62% in the case of Multi Layer Perceptron.

> **RQ3: How useful are the generated test inputs to improve the fairness of the model?**

Table 5: Retraining Effectiveness

Classifier	estimated % of disc input in $\mathbb{I}$ (95% confidence interval)		%Impr	%Inps added
	before retraining	after retraining		
Fair SVM	3.86 (3.76, 3.95)	2.89 (2.64, 3.14)	25.15	15.6
SVM	0.33 (0.14, 0.51)	0.12 (0.09, 0.14)	63.54	26.9
MLPC	0.39 (0.36, 0.42)	0.28 (0.27, 0.29)	30.12	23.7
Random Forest	8.84 (8.78, 8.91)	6.68 (6.35, 7.01)	24.48	32.4
Decision Tree	0.48 (0.45, 0.51)	0.027 (0.026, 0.028)	94.36	10.6
Ensemble	7.73 (7.14, 8.32)	6.06 (5.64, 6.48)	21.58	28.3

Aequitas has a completely automated module which guarantees a decrease in the percentage of discriminatory inputs in $\mathbb{I}$. The discriminatory inputs, as discovered by Aequitas, were systematically added to the training dataset (cf. Section 4.4). The results of retraining the classifiers appear in Table 5. In general, retraining the classifiers is not significantly time consuming. In particular, each classifier was retrained within an hour. For some classifiers, such as the SVM, our retraining scheme only took a few minutes.

We leverage the law of large numbers (LLN) from statistical theory to estimate the percentage of discriminatory inputs in $\mathbb{I}$ (cf. Section 4.3). In particular, we randomly sample a large number of inputs from $\mathbb{I}$ and compute the ratio of discriminatory inputs to the total inputs sampled. This experiment is repeated a large number of times and the average of the computed ratio is used as the estimate for the percentage of discriminatory inputs in $\mathbb{I}$. We note from statistical theory that as the number of experiment is repeated a large number of times, the average of the computed ratio should be close to the expected fraction of discriminatory inputs in $\mathbb{I}$. We also compute the 95% confidence interval estimate for the percentage of discriminatory inputs in $\mathbb{I}$. It is useful to note that these intervals are fairly tight and that adds to the confidence we have in our point estimates as well.

As observed from Table 5, Aequitas is effective in reducing the percentage of discriminatory inputs in $\mathbb{I}$ for all the classifiers under test. Specifically, we observe an average improvement of 43.2%, in terms of reducing the discriminatory inputs. Using our retraining module, we added an average of only 7463 datapoints (22.92% of the original training data) to achieve the result obtained in Table 5.

> **Finding:** Retraining using Aequitas lowers the discrimination percentage in $\mathbb{I}$ by an average of 43.2% and up to 94.36%.

6 RELATED WORK

In this section, we review the related literature and position our work on fairness testing.

Fair Machine Learning Models The machine learning research community have turned their attention on designing classifiers that avoid discrimination [2, 4, 6, 9, 19]. These works primarily focus on the theoretical aspects of classifier models to achieve fairness in the classification process. Such a goal is either achieved by pre-processing training data or by modifying existing classifiers to limit discrimination. Our work is complementary to the approaches that aim to design fair machine-learning models. We introduce an efficient way to search the input domain of classifiers whose goal is to achieve fairness in decision making. We wish to provide a mechanism for these classifiers to quickly evaluate their fairness properties and help improve their fairness in decision making via retraining, if necessary.

Fairness Testing From the software engineering point of view, the research on validating the fairness of machine-learning models is still at its infancy. A recent work [5] along this line of research defines software fairness and discrimination, including a causality-based approach to algorithmic fairness. However, in contrast to our Aequitas approach, the focus of this work is more on defining fairness and tests were generated in random [5]. In particular, Aequitas can be used as a directed test generation module to uncover discriminatory inputs and discovery of these inputs is essential to understand individual fairness [2] of a machine-learning model. In addition to this and unlike existing approach [5], Aequitas provides a module to automatically retrain the machine-learning models and reduce discrimination in the decisions made by these models.

Testing and Verification of Machine Learning models DeepXplore [16] is a whitebox differential testing algorithm for systematically finding inputs that can trigger inconsistencies between multiple deep neural networks (DNNs). The neuron coverage was used as a systematic metric for measuring how much of the internal logic of a DNNs have been tested. More recently, DeepTest [17] leverages metamorphic relations to identify erroneous behaviors in a DNN. The usage of metamorphic relations somewhat solves the limitation of differential testing, especially to lift the requirement of having multiple DNNs implementing the same functionality. Finally, a feature-guided black-box approach is proposed recently to validate the safety of deep neural networks [18]. This work uses their method to evaluate the robustness of neural networks in safety-critical applications such as traffic sign recognition.

The objective of these works, as explained in the preceding paragraph, is largely to evaluate the robustness property of a given machine-learning model. In contrast, we are interested in the *fairness property*, which is fundamentally different from robustness. Therefore, validating fairness requires special attention along the line of systematic test generation.

Search based testing Search-based testing has a long and varied history. The most common techniques are hill climbing, simulated annealing and genetic algorithms [11]. These have been applied extensively to test applications that largely fall in the class of deterministic software systems. Aequitas is the first instance in our knowledge that employs a novel search algorithm to test the fairness of machine-learning systems. We believe that we can port Aequitas for the usage in a much wider machine-learning context.

7 THREATS TO VALIDITY

The effectiveness and efficiency of Aequitas critically depends on the following factors:

Robustness: Our Aequitas approach is based on the hypothesis that the machine-learning models under test exhibit robustness. This is a reasonable assumption, as we expect the models under test to be deployed in production settings. As evidenced by our evaluation, Aequitas approach, which is based on the aforementioned hypothesis, was effective to localize the search in the vicinity of discriminatory input regions for state-of-the-art models.

Training data and access to model: Aequitas needs access to the training data and the training mechanism of the machine-learning model to be able to evaluate and retrain the model. Without access to the training data, Aequitas will not be able to successfully improve the fairness of the model. This is because Aequitas is used to generate test inputs that violate fairness and augment the original training set to improve the model under test. The generated test inputs, however, is not sufficient to train a machine-learning model from scratch.

Input Structure: Aequitas works on real-valued inputs. Aequitas, in its current form, does not handle image, sound or video inputs. This, however, does not diminish the applicability of Aequitas. Numerous real-world applications still use only real-valued data for prediction. These include applications in finance, security, social welfare, education, healthcare and human resources. Examples of applications include income prediction, crime prediction, disease prediction, job short-listing and college short-listing, among others. For models that take inputs such as images and videos, we need to incorporate additional techniques for automatically generating valid input data. However, we believe that the core idea behind our Aequitas approach, namely the global and the local search employed over the input space, will still remain valid.

Probability change parameter: The users of Aequitas will have to experiment and carefully choose Δ_v and Δ_{pr} values which change the probabilities of choosing p (i.e. the input parameter to perturb) and δ (i.e. the perturbation value). If Δ_v (respectively, Δ_{pr}) is too high, then an overshoot might occur and a certain discriminatory input region may never be explored. If Δ_v (respectively, Δ_{pr}) is too low, then the effectiveness of Aequitas semi-directed and Aequitas fully-directed would be very similar to Aequitas random. In our experiments, we evaluated with a few Δ_v and Δ_{pr} values before our results stabilized.

Limited discriminatory input features: We evaluate Aequitas with discriminatory input feature *gender*. Hence, we cannot conclude the effectiveness of Aequitas for other potentially discriminatory input features. However, the mechanism behind Aequitas is generic and allows extensive evaluation for other discriminatory input features in a future extension of the tool.

8 CONCLUSION

In this paper, we propose Aequitas– a fully automated and directed test generation strategy to rapidly generate discriminatory inputs in machine-learning models. The key insight behind Aequitas is to exploit the robustness property of common machine learning models and use it to systematically direct the test generation process. Aequitas provides statistical evidence on the number of discriminatory inputs in a model under test. Moreover, Aequitas incorporates strategies to systematically leverage the generated test inputs to improve the fairness of the model. We evaluate Aequitas with state-of-the-art classifiers and demonstrate that Aequitas is effective in generating discriminatory test inputs as well as improving the fairness of machine-learning models. At its current state, however, Aequitas does not have the capability to localize the cause of discrimination in a model. Further work is required to isolate the cause of discrimination in the model.

Aequitas provides capabilities to lift the state-of-the-art in testing machine-learning models. We envision to extend our Aequitas approach beyond fairness testing and for machine-learning models taking complex inputs including images and videos. We hope that the central idea behind our Aequitas approach would influence the rigorous software engineering principles and help validate machine-learning applications used in sensitive domains. For reproducibility and advancing the state of research, we have made our tool and all experimental data publicly available:

https://github.com/sakshiudeshi/Aequitas

ACKNOWLEDGMENT

The authors would like to thank Chundong Wang and the anonymous reviewers for their insightful comments. The first author is supported by the President's Graduate Fellowship funded by the Ministry of Education, Singapore.

REFERENCES

[1] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

[2] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 214–226, 2012.

[3] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, Mar 2018.

[4] 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, Sydney, NSW, Australia, August 10-13, 2015*, pages 259–268, 2015.

[5] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, pages 498–510, 2017.

[6] Gabriel Goh, Andrew Cotter, Maya R. Gupta, and Michael P. Friedlander. Satisfying real-world goals with dataset constraints. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2415–2423, 2016.

[7] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial examples for malware detection. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II*, pages 62–79, 2017.

[8] Alon Y. Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[9] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II*, pages 35–50, 2012.

[10] Louisa Lam and Ching Y. Suen. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 27(5):553–568, 1997.

[11] Phil McMinn. Search-based software test data generation: a survey. *Softw. Test., Verif. Reliab.*, 14(2):105–156, 2004.

[12] University of Michigan. Nondiscrimination policy notice. URL: https://hr.umich.edu/working-u-m/workplace-improvement/office-institutional-equity/nondiscrimination-policy-notice/.

[13] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 506–519, 2017.

[14] Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. Crafting adversarial input sequences for recurrent neural networks. In *2016 IEEE Military Communications Conference, MILCOM 2016, Baltimore, MD, USA, November 1-3, 2016*, pages 49–54, 2016.

[15] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568, 2008.

[16] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 1–18, 2017.

[17] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *CoRR*, abs/1708.08559, 2017. URL: http://arxiv.org/abs/1708.08559, arXiv:1708.08559.

[18] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*, pages 408–426, 2018.

[19] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 962–970, 2017.