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 Aeqitas approach automatically discovers discriminatory inputs that highlight fairness violation. At the core of Aeqitas are three novel strategies to employ probabilistic search over the input space with the objective of uncovering fairness violation. Our Aeqitas 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 Aeqitas 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 Aeqitas 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, Aeqitas 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

1 INTRODUCTION

Nondiscrimination is one of the most critical factors for social protection and equal human rights. The basic idea behind nondiscrimination 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 nondiscrimination 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 Aeqitas 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 GenderA or GenderB. 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 Aeqitas 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 Aeqitas works is due to its directed strategy for test generation. In particular, Aeqitas 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 A1 and A2 which are in the neighborhood of the point A. Since the point A exhibits discriminatory nature, it is likely that both points A1 and A2 will be discriminatory, as reflected via the presence of points B1 and B2, respectively. In our Aeqitas 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 neighborhood of these inputs, as discovered during the random sampling, to find the presence of more inputs (e.g., points A1 and A2 in Figure 1) of the same nature.

An appealing feature of Aeqitas 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 Aeqitas 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, Aeqitas empirically validates that a directed test generation, to uncover the discriminatory input regions, is indeed more desirable than random test generation. Moreover, Aeqitas 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 Aeqitas (Section 2), we make the following contributions:

1. We present Aeqitas, 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 Aeqitas based on python. Our implementation and all experimental data are publicly available (Section 5).
4. We evaluate our Aeqitas approach with six state-of-the-art classifiers including a classifier that was designed with fairness in mind. Our evaluation reveals that Aeqitas is effective in generating discriminatory inputs and improving the fairness of the classifiers under test. In particular, Aeqitas 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 Aeqitas Aeqitas aims to discover the violation of individual fairness [2] in machine-learning models. This means, Aeqitas 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 Aeqitas. For the rest of the paper, we will simply
use the term fairness (instead of individual fairness) in the light of our Aeqitas 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], Aeqitas 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 GenderA or GenderB. 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 $\delta$ be a small value. If $f$ is robust, then $f(i + \delta)$ is a small value. 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, Aeqitas is designed to avoid these adversarial input regions by systematically directing the test generators. Intuitively, Aeqitas 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 input parameters, say $P_1, P_2, \ldots, P_n$. We assume $I_k$ captures the domain of input parameter $P_k$. Therefore, $I = I_1 \times I_2 \times \ldots \times I_n$. 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 $I_p$. 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 $I_p$. $I_p$, in turn, was discovered in the first step of Aeqitas.

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) $\delta$ is added to $I_p$. 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 $\delta$ of $I_p$ consistently yields discriminatory inputs, then the perturbation $\delta$ is employed with higher probability. Since $\delta$ 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
The number of input parameters to the machine-learning models under test is limited. A pre-determined discrimination threshold $\gamma$ is in place, and $\mathbb{D}$ is the input domain of the model. Let $P_i$ be the $i$-th input parameter of the model, and $P = \bigcup_{i=1}^n P_i$ be the set of all input parameters. Let $P_{\text{disc}}$ denote the set of sensitive or potentially discriminatory input parameters (e.g., gender). Clearly, $P_{\text{disc}} \subseteq \bigcup_{i=1}^n P_i$.

We denote $I_p$ as the value of input parameter $p$ in input $I \in \mathbb{D}$. To this end, we leverage the law of large numbers (LLN) to systematically generate test inputs that exhibit similar characteristics. In our Aeqitas 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.

**How Aeqitas can be used to improve software fairness?**

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

### 4 Detailed Approach

In this section, we discuss our Aeqitas 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.

**Table 1: Notations used in Aeqitas approach**

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

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

**Figure 3:** Our Aeqitas approach at a glance

**Figure 3:** Our Aeqitas approach at a glance

![Figure 3](image-url)

- **Neighbourhood of $I$ is obtained by adding small change $\delta$ to an input parameter.**
- **Figure 3 illustrates Aeqitas approach when $I$ and $I'$ were discovered in the first step. Then, the second step explored the neighbourhood of $I$ by adding small changes $\delta$ to an input parameter.**

**Estimation of discriminatory inputs**

An appealing feature of Aeqitas is that we can estimate the percentage of discriminatory inputs in $\mathbb{D}$. 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 $\frac{|\mathbb{D}|}{K}$ over a large number of trials. According to LLN, the average of these ratios closely approximates the actual percentage of discriminatory inputs in $\mathbb{D}$. Figure 2(b) highlights such convergence after only 400 trials when $K$ was chosen to be 1000.

**Why Aeqitas works?**

The reason Aeqitas 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 $\delta$, 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 Aeqitas 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.
**Definition 2.** *(Perturbation)* We define perturbation \( q \) as a function \( q : \mathbb{I} \times (P \setminus P_{\text{disc}}) \times \Gamma \rightarrow \mathbb{I} \) where \( \Gamma = \{-1, +1\} \) captures the set of directions to perturb an input parameter. If \( I' = g(I, p, \delta) \) where \( I \in \mathbb{I}, p \in P \setminus P_{\text{disc}} \) 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. \( \Gamma \) 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 Aeqitas 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 \( P_{\text{disc}} \subseteq P \). This leads to a set of inputs \( \mathbb{I}^{(d)} \). We note that the set of sensitive parameters (e.g. race, religion, gender) \( P_{\text{disc}} \) 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 neighbours are likely to be discriminatory for robust models.

To select the neighbourhood of *disc_inputs*, Aeqitas perturbs an input \( I \in \textit{disc_inputs} \) by changing the value of some parameter \( p \in P \setminus P_{\text{disc}} \) (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 \( p \in P \setminus P_{\text{disc}} \) with probability \( \sigma_{pr}(p) \) (cf. Algorithm 2). For all \( p \in P \setminus P_{\text{disc}} \), initially \( \sigma_{pr}(p) \) was assigned to \( \frac{1}{|P \setminus P_{\text{disc}}|} \). Once \( p \) is chosen its value is perturbed by \( \delta = -1 \) with probability \( \sigma_{c}(p) \) and by \( \delta = +1 \) with probability \( 1 - \sigma_{c}(p) \). \( \sigma_{c}(p) \) is initialized to 0.5 for all parameters in \( p \in P \setminus P_{\text{disc}} \).

---

**Figure 4: An overview of our Aeqitas approach**

**Algorithm 1 Global Search**

```plaintext
1: procedure GLOBAL_EXP(P, P_{disc})
2: disc_inps ← \emptyset
3:▷ N is the number of trials in global search
4: for i in (0, N) do
5: Select an input \( I \in \mathbb{I} \) at random
6:▷ \( \mathbb{I}^{(d)} \) extends \( I \) with all possible values of \( P_{\text{disc}} \)
7: \( \mathbb{I}^{(d)} \) ← \{ \( I' \mid \forall p \in P \setminus P_{\text{disc}}, I_p = I'_p \} \)
8: if \( (\exists I', I' \in \mathbb{I}^{(d)}, |f(I) - f(I')| > \gamma) \) then
9: disc_inps ← disc_inps \cup \{I\}
10: end if
11: end for
12: return disc_inps
13: end procedure
```

**Algorithm 2 Local Search**

```plaintext
1: procedure LOCAL_EXP(disc_inps, P, P_{disc}, \Delta_{c}, \Delta_{pr})
2: Test ← \emptyset
3: Let \( P' = P \setminus P_{\text{disc}} \)
4: Let \( \sigma_{pr}(p) = \frac{1}{|P'|} \) for all \( p \in P' \)
5: Let \( \sigma_{c}(p) = 0.5 \) for all \( p \in P' \)
6: for \( I \in \textit{disc_inps} \) do
7:▷ N is the number of trials in local search
8: for i in (0, N) do
9: Select \( p \in P' \) with probability \( \sigma_{pr}(p) \)
10: Select \( \delta = -1 \) with probability \( \sigma_{c}(p) \)
11:▷ Note that \( I \) is modified as a side-effect of modifying \( I_p \)
12: \( I_p ← I_p + \delta \)
13:▷ \( \mathbb{I}^{(d)} \) extends \( I \) with all values of \( P_{\text{disc}} \)
14: \( \mathbb{I}^{(d)} \) ← \{ \( I' \mid \forall p \in P \setminus P_{\text{disc}}, I_p = I'_p \} \)
15: if \( (\exists I', I' \in \mathbb{I}^{(d)}, |f(I) - f(I')| > \gamma) \) then
16:▷ Add the perturbed input \( I \)
17: Test ← Test \cup \{I\}
18: end if
19: update_prob(I, p, Test, \Delta_{c}, \Delta_{pr})
20: end for
21: end for
22: return Test
23: end procedure
```
Algorithm 3 \textsc{Aequitas} semi-directed update probability

1: procedure \textsc{update\_prob}(I, p, Test, \delta, \Delta_D, \Delta_P)
2: if \((I \in \text{Test} \land \delta = -1) \lor (I \notin \text{Test} \land \delta = +1)\) then
3: \(\sigma_p[p] \leftarrow \min(\sigma_p[p] + \Delta_p, 1)\)
4: end if
5: if \((I \notin \text{Test} \land \delta = -1) \lor (I \in \text{Test} \land \delta = +1)\) then
6: \(\sigma_c[p] \leftarrow \max(\sigma_c[p] - \Delta_c, 0)\)
7: end if
8: end procedure

Algorithm 4 \textsc{Aequitas} fully-directed update probability

1: procedure \textsc{update\_prob}(I, p, Test, \delta, \Delta_D, \Delta_P)
2: if \((I \in \text{Test} \land \delta = -1) \lor (I \notin \text{Test} \land \delta = +1)\) then
3: \(\sigma_c[p] \leftarrow \min(\sigma_c[p] + \Delta_c, 1)\)
4: end if
5: if \((I \notin \text{Test} \land \delta = -1) \lor (I \in \text{Test} \land \delta = +1)\) then
6: \(\sigma_p[p] \leftarrow \max(\sigma_p[p] - \Delta_p, 0)\)
7: end if
8: if I \in Test then
9: \(\sigma_P[p] \leftarrow \sigma_P[p] + \Delta_P\)
10: \(\sigma_P[p] \leftarrow \frac{\sigma_P[p]}{\sum_{x \in P \setminus P_{disc}} \sigma_P[x]}\) for all \(p \in P \setminus P_{disc}\)
11: end if
12: end procedure

\textsc{Aequitas} employs three different strategies, namely \textsc{Aequitas} random, \textsc{Aequitas} semi-directed and \textsc{Aequitas} fully-directed, to update the probabilities in \(\sigma_P\) and \(\sigma_c\). 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 \textsc{Aequitas}.

\textbf{Aequitas random.} \textsc{Aequitas} random does not update the initial probabilities assigned to \(\sigma_p\) and \(\sigma_c\). This results in \(\delta\) (i.e. perturbation value) and \(p\) (i.e. the parameter to perturb) both being chosen randomly. Intuitively, \textsc{Aequitas} random explores inputs around the neighbourhood of \textit{disc\_inputs} (i.e. set of discriminatory inputs discovered via global search) uniformly at random. Nevertheless, \textsc{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.

\textbf{Aequitas semi-directed.} \textsc{Aequitas} semi-directed drives the test generation by systematically updating \(\sigma_c\), 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 \(\sigma_c[p]\) and the probability that \(\delta = +1\) is \(1 - \sigma_c[p]\). If the perturbed input is discriminatory (cf. Definition 1), then we increase the probability associated with \(\sigma_c[p]\) by a pre-determined offset \(\Delta_c\). Otherwise, \(\sigma_c[p]\) is reduced by the same offset \(\Delta_c\). Intuitively, the updates to probabilities in \(\sigma_c\) prioritise a direction \(\delta \in \{-1, +1\}\) when the respective direction results in discriminatory inputs.

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

4.3 Estimation using LLN

An attractive feature of \textsc{Aequitas} is that we can estimate the percentage of discriminatory inputs in \(I\) for any given model. We leverage the Law of Large Numbers (LLN) from probability theory to accomplish this. Let \(\Lambda\) 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. \(\Lambda\) then outputs the percentage \(\overline{m} = \frac{m'}{m} \times 100\).

\(\Lambda\) is conducted \(K\) times. In each instance of the experiment, we collect the outcome \(m_1, m_2, \ldots, m_K\). Let \(\overline{M} = \frac{1}{K} \sum_{i=1}^{K} m_i\). 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,

\[ K \rightarrow \infty \quad \overline{M} \rightarrow M^* \]

where \(M^*\) is the true percentage of the discriminatory inputs present in \(I\) 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, \textsc{Aequitas} has a completely automated module that guarantees reduction of the percentage of discriminatory inputs in \(I\). 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 \textsc{Aequitas}. \textsc{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
Retraining

Algorithm 5 Retraining
1: procedure RETRAINING(f, Test, training_data)
2:    N ← ∞
3:    f_{cur} ← f
4:    for i in (2, N) do
5:        p_i ← a random real number between (2^{i-2}, 2^{i-1})
6:        if p_i > 100 then
7:            Exit the loop
8:        end if
9:        k ← len(training_data)
10:       n_{addn} ← \frac{p_i k}{100}
11:       TD_{addn} ← randomly selected n_{addn} inputs from Test
12:       TD_{new} ← training_data ∪ TD_{addn}
13:       f_{new} ← model trained using TD_{new}
14:       « Estimate the number of discriminatory inputs (section 4.3)
15:       \text{fair}_{\text{cur}} ← LLN_Fairness_Estimation (f_{cur})
16:       \text{fair}_{\text{new}} ← LLN_Fairness_Estimation (f_{new})
17:       if (\text{fair}_{\text{cur}} > \text{fair}_{\text{new}}) then
18:           f_{cur} ← f_{new}
19:       else
20:           Exit the loop
21:    end if
22: end for
23: return f_{cur}
24: end procedure

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 \mathcal{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>
<thead>
<tr>
<th>Classifier name</th>
<th>Lines of python code</th>
<th>Input domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fair SVM</td>
<td>913</td>
<td>10^6</td>
</tr>
<tr>
<td>SVM</td>
<td>1123</td>
<td></td>
</tr>
<tr>
<td>MLPC</td>
<td>1308</td>
<td></td>
</tr>
<tr>
<td>Random Forest</td>
<td>1951</td>
<td></td>
</tr>
<tr>
<td>Decision Tree</td>
<td>1465</td>
<td></td>
</tr>
<tr>
<td>Ensemble</td>
<td>3466</td>
<td></td>
</tr>
</tbody>
</table>

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,
We measured the effectiveness via the number of discriminatory inputs by 43.2% on average with a maximum reduction of 94.36%. To this end, a model with the objective to reduce the number of discriminatory inputs generated by a test generation technique. Aeqitas provides capabilities to automatically retrain a machine-learning model with the objective to reduce 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 Aeqitas outperform a purely random approach. In particular, the rate at which our Aeqitas approach generates discriminatory inputs is significantly higher than a purely random approach. As a result, Aeqitas provides scalable and effective technique for machine learning engineers who aim to rapidly discover fairness issues in their models. Aeqitas random, Aeqitas semi-directed and Aeqitas fully-directed involve increasing level of sophistication in directing the test input generation. As a result, Aeqitas fully-directed approach performs the best among all our test generators. In particular, Aeqitas semi-directed is on an average 46.7% and up to 64.9% better than Aeqitas random. Finally, Aeqitas full-directed is on an average 29.5% and up to 56.56% better than Aeqitas semi-directed.

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

### Key results

We use three different test generation methodologies, namely Aeqitas random, Aeqitas semi-directed and Aeqitas 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, Aeqitas fully-directed involves the highest level of sophistication in searching the input space. As expected, Aeqitas fully-directed consistently outperforms the Aeqitas random and Aeqitas semi-directed, as observed from Figure 5. However, Aeqitas fully-directed and Aeqitas semi-directed demand more computational resources per unit time than Aeqitas random. As a result, Aeqitas 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 Aeqitas 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 Aeqitas approach (i.e. Aeqitas random) significantly outperforms the state-of-the-art. In our evaluation, we discovered that Aeqitas 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. Aeqitas also provides capabilities to automatically retrain a machine-learning model with the objective to reduce the number of discriminatory inputs. To this end, Aeqitas reduced the number of discriminatory inputs by 43.2% on average with a maximum reduction of 94.36%.

### Finding: Aeqitas 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 Aeqitas semi-directed, which, in turn performs up to 64.9% better than Aeqitas random, our least sophisticated approach.

### RQ1: How effective is Aeqitas in finding discriminatory inputs?

We evaluate the capability of Aeqitas 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.

Table 4 summarizes how much time each of the methods takes to generate 10,000 discriminatory inputs. On an average Aeqitas random performs 64.42% faster than the state of the art. The improvement in Aeqitas fully-directed is even more profound. On an average, Aeqitas 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 the time needed for test generation and for test execution.
Table 3: Effectiveness of AEQUITAS approach

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Random [5]</th>
<th>AEQUITAS random</th>
<th>AEQUITAS semi-directed</th>
<th>AEQUITAS fully-directed</th>
</tr>
</thead>
<tbody>
<tr>
<td>% discriminatory input</td>
<td>% discriminatory input</td>
<td># inputs generated</td>
<td>% discriminatory input</td>
<td># inputs generated</td>
</tr>
<tr>
<td>Fair SVM</td>
<td>3.45</td>
<td>39.4</td>
<td>315640</td>
<td>65.2</td>
</tr>
<tr>
<td>SVM</td>
<td>0.18</td>
<td>0.53</td>
<td>54683</td>
<td>0.574</td>
</tr>
<tr>
<td>MLPC</td>
<td>0.3466</td>
<td>2.15</td>
<td>218727</td>
<td>2.39</td>
</tr>
<tr>
<td>Random Forest</td>
<td>8.34</td>
<td>18.312</td>
<td>218727</td>
<td>21.722</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.485</td>
<td>2.33</td>
<td>153166</td>
<td>2.89</td>
</tr>
<tr>
<td>Ensemble</td>
<td>8.23</td>
<td>22.34</td>
<td>187980</td>
<td>36.08</td>
</tr>
</tbody>
</table>

%Impr 254.25s 100101 8.84 (8.78, 8.91) 21.722 23.7 322725 36.08 3.86 (3.76, 3.95) 70.32 6157.23s 0.12 (0.09, 0.14) 8.34 989.43s 7159.54s 65.2 431.76s 7.73 (7.14, 8.32) 264523 34.98 0.574 6368.79s 63.54 0.39 (0.36, 0.42) 1334.67s 8.23 1035.32s 32.4 0.485 282973 218727 0.53 248229 228.14s 1.22 2.33 24.48 94.36 21.58

Table 4: Test generation efficiency

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Random</th>
<th>AEQUITAS random</th>
<th>AEQUITAS semi-directed</th>
<th>AEQUITAS fully-directed</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Impr</td>
<td>%Impr</td>
<td># inputs generated</td>
<td>%Impr</td>
<td># inputs generated</td>
</tr>
<tr>
<td>Fair SVM</td>
<td>1589.87s</td>
<td>394.7%</td>
<td>345.65s</td>
<td>228.14s</td>
</tr>
<tr>
<td>SVM</td>
<td>7159.54s</td>
<td>359.9%</td>
<td>2673.86s</td>
<td>2199.216</td>
</tr>
<tr>
<td>MLPC</td>
<td>6157.23s</td>
<td>793.65s</td>
<td>431.76s</td>
<td>207.87s</td>
</tr>
<tr>
<td>Random Forest</td>
<td>9563.12s</td>
<td>2692.98s</td>
<td>1334.67s</td>
<td>1145.346</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>1035.32s</td>
<td>509.13s</td>
<td>371.89s</td>
<td>254.25s</td>
</tr>
<tr>
<td>Ensemble</td>
<td>6368.79s</td>
<td>2178.45s</td>
<td>1067.75e</td>
<td>989.43s</td>
</tr>
</tbody>
</table>

Random Forest 8.39 (7.3, 9.7) 2.39 (2.63, 3.14) 25.25 15.8
SVM 0.39 (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 (7.88, 9.81) 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) 6.86 (6.44, 6.48) 21.56 28.3

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.

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

Finding: AEQUITAS fully-directed is effective in reducing the percentage of discriminatory inputs in 1 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 has 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 preprocessing 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

Deep-Xplore [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 DNN has 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. Aeqitas 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 Aeqitas for the usage in a much wider machine-learning context.

7 THREATS TO VALIDITY

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

Robustness: Our Aeqitas 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, Aeqitas 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: Aeqitas 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, Aeqitas will not be able to successfully improve the fairness of the model. This is because Aeqitas 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: Aeqitas works on real-valued inputs. Aeqitas, in its current form, does not handle image, sound or video inputs. This, however, does not diminish the applicability of Aeqitas. 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 Aeqitas approach, namely the global and the local search employed over the input space, will still remain valid.

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

Limited discriminatory input features: We evaluate Aeqitas with discriminatory input feature gender. Hence, we cannot conclude the effectiveness of Aeqitas for other potentially discriminatory input features. However, the mechanism behind Aeqitas 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 Aeqitas—a fully automated and directed test generation strategy to rapidly generate discriminatory inputs in machine-learning models. The key insight behind Aeqitas is to exploit the robustness property of common machine learning models and use it to systematically direct the test generation process. Aeqitas provides statistical evidence on the number of discriminatory inputs in a model under test. Moreover, Aeqitas incorporates strategies to systematically leverage the generated test inputs to improve the fairness of the model. We evaluate Aeqitas with state-of-the-art classifiers and demonstrate that Aeqitas is effective in generating discriminatory test inputs as well as improving the fairness of machine-learning models. At its current state, however, Aeqitas 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.

Aeqitas provides capabilities to lift the state-of-the-art in testing machine-learning models. We envision to extend our Aeqitas approach beyond fairness testing and for machine-learning models taking complex inputs including images and videos. We hope that the central idea behind our Aeqitas 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


