A Combinatorial Approach to Fairness Testing of Machine Learning Models

Ankita Ramjibhai Patel[1], Jaganmohan Chandrasekaran[2], Yu Lei[1], Raghu N. Kacker[3], D. Richard Kuhn[3]

[1]Dept. of Computer Science and Engineering, The University of Texas at Arlington, Arlington, Texas 76019, USA
[2]Commonwealth Cyber Initiative (CCI), Virginia Tech, Arlington, Virginia 22203, USA
[3]Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, Maryland 20899, USA

***Abstract*—Machine Learning (ML) models could exhibit biased behavior, or algorithmic discrimination, resulting in unfair or discriminatory outcomes. The bias in the ML model could emanate from various factors such as the training dataset, the choice of the ML algorithm, or the hyperparameters used to train the ML model. In addition to evaluating the model's correctness, it is essential to test ML models for fair and unbiased behavior. In this paper, we present a combinatorial testing-based approach to perform fairness testing of ML models. Our approach is model agnostic and evaluates fairness violations of a pre-trained ML model in a two-step process. In the first step, we create an input parameter model from the training data set and then use the model to generate a t-way test set. In the second step, for each test, we modify the value of one or more protected attributes to see if we could find fairness violations. We performed an experimental evaluation of the proposed approach using ML models trained with tabular datasets. The results suggest that the proposed approach can successfully identify fairness violations in pre-trained ML models.**

Keywords—Fairness Testing, Algorithmic Discrimination, Bias Detection, Testing Model Bias, Testing ML model, Combinatorial Testing

I. INTRODUCTION

Machine Learning (ML) models are widely used across domains in automated decision-making processes. For example, ML-based recommender systems are used by banks to approve or deny loans for their applicants [10], companies use ML-based software applications to filter/select candidates in the hiring process [12, 13]. Despite its impressive predictive capabilities, ML models inadvertently exhibit bias and result in discriminatory behavior, also referred to as algorithmic discrimination [2, 11, 13, 39].

A bias in an ML model could be introduced via various factors such as the training dataset, the choice of the ML algorithm, or the hyperparameters used to train the ML model. Recent reports in [11, 27] illustrate the biased behavior of ML models and their adverse effects on society. Thus, in addition to an ML model's correctness, there is a need to test and ensure that the ML model behaves in an unbiased and non-discriminatory manner. In recent years, a significant amount of research has been reported on fairness testing [22]. From a testing perspective, the objective of fairness testing is to evaluate whether a model under test exhibits a consistent, non-discriminatory behavior for all its use cases?

ML models used in the automated decision-making process must avoid discriminating against sensitive characteristic features of individuals such as age, race, sex, and ethnicity [1]. The sensitive characteristics vary depending on the domain, and they are referred to as protected attributes. Assume a pre-trained model M, on receiving an input I, predicts a class label C. A discriminatory behavior (also referred to as fairness violation) of M can be broadly classified into two types: (1) individual discrimination and (2) group discrimination [1, 15]. Model M exhibits individual discrimination if M predicts a different outcome for two similar instances. Model M exhibits group discrimination if M favors or discriminates instances belonging to a specific group over the other groups.

Some recent work has been reported on fairness testing [1, 17, 18, 19, 20, 23, 24, 26]. Galhotra et al. proposed THEMIS, a causality-based random test generation technique to identify discriminatory behavior of ML models [19]. Aggarwal et al. proposed a symbolic execution-based approach to generate test instances and then use a local explanation tool called LIME to identify individual discriminations in an ML model [1]. Udeshi et al. proposed *Aeqitas,* a testing technique to discover discriminatory inputs by randomly sampling the input space [18]. The results from existing studies suggest that traditional testing techniques can effectively be adapted to identify the discriminatory behavior of ML models.

In this paper, we present a combinatorial approach to test ML models for individual discriminations. We believe that the key insight that has allowed combinatorial testing to be effective for general software testing could also apply to fairness testing. That is, while the behavior of an entire model could be affected by many factors, individual fairness violations may be caused by only a few factors. Our approach consists of two phases: *Generating T-Way Tests* and *Identifying Fairness Violation*. In Phase 1, we generate t-way tests based on a training dataset. We begin this phase with the design of an Input Parameter Model (IPM). All attributes excluding the class label attribute from the training dataset are mapped as parameters. Then, we identify representative values for each parameter based on the corresponding attribute's data type. In the case of categorical (string) attribute(s), we identify and map its unique values as representative values. For numerical attribute(s), we identify its representative values via discretization, a process of converting numerical (continuous) values into a set of discrete values [6].

Next, we identify constraints using an unsupervised learning algorithm that infers the underlying relationships among different attributes (excluding the class label) from the training dataset. The relationships identified by the learning algorithm are mapped as constraints in our IPM. Finally, we

generate abstract t-way tests that are later converted into concrete t-way tests.

Using the concrete t-way tests, in phase 2, we identify individual fairness violations using a counterfactual approach. Given a t-way test instance, we generate perturbations that are similar to the t-way instance by modifying the value of one or more protected attributes while retaining the values of non-protected attributes and respecting all the constraints. On receiving the perturbated instance(s) as input, if the ML model results in an outcome (C`) that differs from the outcome (C) produced for the original t-way instance, then the ML model is considered to exhibit an individual fairness violation (C != C`).

We report an experimental evaluation of the proposed approach. Three widely used datasets, namely *Adult Income[29], German Credit[30], and COMPAS[31],* are used as our subject datasets. We build ML classifiers (models) for each dataset using four popular Machine Learning algorithms, namely *Logistic Regression, Random Forest, Support Vector Machines, and Deep Neural Network.* We generate t-way test sets based on the datasets and test the ML models for fairness violations. Our results suggest that the combinatorial approach can successfully identify fairness violations in ML models. In some cases, more than 40% of t-way test cases resulted in a fairness violation. Furthermore, the results indicate that t-way test cases generated using our approach can identify a substantial number of fairness violations across different types of ML classifiers. This suggests that the proposed approach is model-agnostic and can be adopted to test fairness violations for different ML models.

The remainder of this paper is organized as follows. Section II provides the introduction of Fairness Testing. Section III presents our approach, illustrated with an example. In Section IV, we present our experimental design, reports the results, and discussion about our results. Section V discusses the existing work on fairness testing. Section VI provides the concluding remarks and plans for future work.

II. Background

ML Model: To build an ML model, first, a practitioner selects an ML algorithm; provides a training dataset and a set of hyperparameters as input to the ML algorithm. Then, the ML algorithm infers a decision logic based on the underlying patterns discovered from the training dataset. The derived decision logic is referred to as the ML model. An ML model exhibits fair behavior if it does not favor or discriminate against a specific individual or a particular group.

Protected Attributes: The attributes from the training dataset that are sensitive and need to be protected against discrimination are referred to as protected attributes [22]. Example of protected attributes includes *Race, Color, Religion, Sex*, and *Familial Status* [28]. Fairness testing aims to evaluate and assure that the ML model exhibits a non-discriminatory behavior.

Individual Discrimination: Given two valid inputs (instances) that differ only by the protected attribute(s), an ML model is expected to predict the same outcome for both the inputs. Otherwise, the ML model is considered to exhibit individual discrimination [1]. For example, consider two applicants with identical credit history but differ only by their *Race*. If an ML model approves the loan for one applicant while rejecting the other, the model exhibits individual discrimination. In the rest of the paper, we refer to individual discrimination simply as a fairness violation unless otherwise specified.

Group Discrimination: If an ML model favors or discriminates instances belonging to a specific group over the other groups, it is considered group discrimination. For example, Amazon AI recruiting tool preferred male candidates over female candidates in the candidate hiring process [2]. Buolamwini et al. demonstrated that commercially available facial recognition software misclassifies more female faces than male faces [39].

Counterfactual Explanation: Explainable Artificial Intelligence (XAI) tools generate explanations for decisions made by ML models [34]. A counterfactual approach is one of the two commonly used approaches to explain a model's decision.

Assume a pre-trained model M, on receiving an input I, predicts a class label C. A counterfactual approach identifies a minimum set of features that, if removed, shall result in a different prediction [35]. That is, a counterfactual is generated by making minor change(s) to the original instance, resulting in a different outcome than the original prediction.

III. Approach

This section presents a combinatorial approach to identify fairness violations of pre-trained ML models that take an instance as input and outputs a prediction. Figure 1 presents the overview of our approach. It consists of two major phases: (1) Generating T-Way Tests, where a t-way test set is generated; and (2) Identifying fairness violation, where the t-way tests are executed to detect fairness violations.

A. Phase 1: Generating T-Way Tests

In Phase 1, we generate t-way test cases (instances) based on the training dataset. A training dataset consists of numerical attributes, categorical attributes, or a combination of both. We first create an Input Parameter Model (IPM) for the training dataset. All attributes except the class label attribute from the training dataset are mapped as a parameter in the IPM. Next, we identify representative values for each identified parameter (attribute) based on its data type (numeric or categorical).

For a categorical attribute, we identify and map all its unique values as parameter values. For a numerical attribute, we identify the parameter values using an entropy-based discretization approach. Discretization is a process of converting numerical (continuous) values into a set of discrete values. A numerical attribute is divided into a small number of intervals, where each interval is mapped to a bin [6]. In entropy-based discretization, the entropy is calculated based on the class label. Then, the entropy-based approach tries to find the best split (bins) where the majority of values in a bin belong to the same class label [40]. The bins identified using the discretization approach are mapped as the parameter values. That is, if a numeric attribute is divided into *n* bins using the discretization approach, then "n" bins are considered as the attribute's representative values.

Next, to derive constraints, we first modify the dataset by mapping all the numeric attributes to their respective bins identified via discretization. Our goal is to identify the relationships among all attributes, excluding the class label attribute. Hence, we remove the class label attribute from the

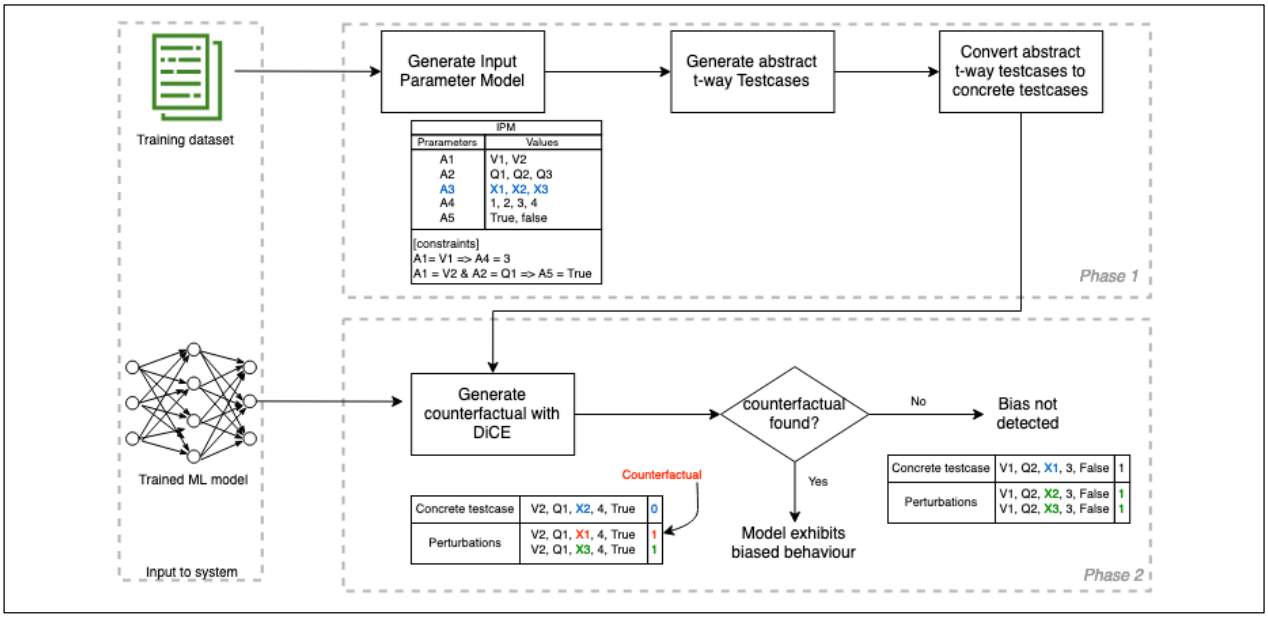

FIGURE 1 – APPROACH OVERVIEW

dataset. Then, the modified dataset is provided as an input to Apriori, an association rule mining algorithm [23]. The apriori algorithm discovers association rules in a two-step process. First, they identify the relationship among attributes that appear together more frequently in the training dataset. Next, the algorithm calculates association rules between frequently appearing attributes (identified in the previous step) using a statistical score. These association rules are mapped as constraints in our IPM. Using constraints enables us to generate valid test cases. Then, we generate an abstract t-way test set using ACTS [4, 14].

The final step in phase 1 is to derive a concrete test set from the abstract test set. Recall that our approach discretizes the numeric attributes. Therefore, in this step, for numeric attributes, we replace the abstract value with a value (selected randomly) from the corresponding bin. The t-way concrete test set (test instances) is used to identify the fairness violations of the pre-trained ML model.

B. Phase 2: Identifying Fairness Violation

In Phase 2, we identify fairness violations by perturbing the test instances using a counterfactual approach. First, each test instance (i.e. a concrete test set generated from Phase 1) is provided as an input to the pre-trained ML model, and its predicted class label is recorded. Then, we generate a set of perturbated instances (from the test instance) by changing the value of one or more protected attribute(s) while retaining the value of non-protected attributes, execute the model with the perturbated instances and record its predicted class label. If the predicted class label for any of the perturbated instances differs from the predicted class label of the original test instance, then it is considered a fairness violation. Otherwise, the model is considered to exhibit fair behavior.

C. Example

We illustrate our approach using an example. Assume an ML classifier is used to predict the admission decisions for prospective candidates. The ML classifier is trained using a dataset that consists of 5 attributes, namely *Gender*, *Race, State*, *Final Score*, and *Decision*. *Gender*, *Race,* and *State* are categorical attributes with 3, 4, and 10 unique values, respectively. *Final Score* is a numerical attribute. *Decision* is a class label with two values – *Accept*, *Reject.* Based on domain knowledge, we identify *Gender* and *Race* as protected attributes among the four attributes.

We begin the first phase by generating an IPM. We identify the four attributes (excluding the class label attribute) *Gender, Race, State, Final Score* as parameters. For the three categorical attributes (*Gender, Race, State*), their unique values are identified as parameter values; We discretize the *Final Score* into eight bins, and they (bins) are identified as its parameter values. Next, to identify constraints, we use the Apriori algorithm. The algorithm identifies two association rules by analyzing the training dataset {*State* = CA => *Final Score* >= 70, *State* = GA => *Final Score* < 90}. We map these association rules as constraints in our IPM. Next, we generate 80 abstract t-way test cases (t=2) followed by deriving the concrete test cases.

Using constraints allows us to generate valid t-way tests. That is, in the generated t-way tests, if the state is CA, then the final score will always be greater than or equal to 70. Similarly, if the state is GA, then the final score will always be less than 90.

In the second phase, we use a counterfactual XAI tool to identify if there exists a counterfactual for any test instance from the t-way test set. We provide the ML classifier, dataset, test instance, a list of protected attributes (*Gender*, *Race*) as an input to the counterfactual tool. Recall that our goal is to identify if changing the protected attributes results in a different outcome. The tool successfully identifies a counterfactual for one of the test instances from the concrete test set: (**male**, **white**, CA, 92). The ML classifier predicts *Admit* for the test instance (original prediction). For one of the perturbated instances: (***female***, ***black***, CA, 92), we observe that the model predicts *Reject*. As the counterfactual indicates, modifying the protected attribute(s) results in a different outcome, suggesting a fairness violation of the ML classifier.

IV. EXPERIMENTS

In this section, we first present the design of our experiments, including the research question(s), the datasets, the subject models, discretization techniques, the counterfactual generation tool, and the metrics used to identify fairness violations. Next, we present and discuss the results of our experiments. Finally, the threats to validity are discussed.

A. Research Questions

Our experiments are designed to answer the following research question: How effective is our combinatorial testing-based approach in fairness testing of ML models?

B. Datasets

In our experiments, we use three datasets, namely the *Adult Income [29]*, *German-Credit [30],* and *COMPAS* [31] datasets, that are among the most widely used in the fairness testing domain [1, 19, 32, 33]. Bellamy et al. presented IBM AI Fairness 360, a software library to detect and mitigate bias in AI models [5]. They made their scripts publicly accessible. We reuse their scripts and preprocess the subject datasets [7].

- The German credit dataset is used to classify individuals as either *good or bad* credit risk based on their personal and financial information. The dataset consists of 1000 instances and 21 attributes (8 numerical + 13 categorical). Among the 20 attributes(excluding the class label), *Sex and Age* are treated as protected attributes.
- The Adult dataset contains census information of individuals that is used to determine if an individual can earn more than $50,000 per year. The dataset has 13 attributes with five numerical and eight categorical attributes. The Adult dataset has two protected attributes: *Sex and Race*.
- The Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) dataset consists of information of defendants such as criminal history, prison time, demographics, and it is used to predict the likelihood of a defendant to re-offend (recidivism). The dataset contains 7214 records with four categorical and five numerical attributes. *Sex and Race* are treated as protected attributes. Note that, the original COMPAS dataset had 52 attributes. We preprocessed the dataset as per AI Fairness 360 [5, 7] and retain ten attributes.

C. Subject Models

We evaluate our approach using four ML classifiers. Three (out of four) ML classifiers are implemented using classical ML algorithms, namely Logistic Regression(LR), RandomForest(RF), and Support Vector Machines(SVM) that are commonly used in fairness testing studies [1,17, 19, 33]. In addition to this, we also use a fourth ML classifier, a simple Deep Neural Network(DNN) model with two hidden layers, in our experiments. Thus, for each dataset, we train and build four ML classifiers. Overall, we use twelve 12 ML classifiers (subject models) in our experiments. Similar to earlier studies [33], we train the ML classifiers using their default configuration, provided by the sci-kit learn library [37].

D. Discretization

We use a decision-tree (entropy-based) algorithm to discretize numeric attributes [37, 38]. The user can specify the depth of the decision tree. A decision tree of depth *n* will generate a maximum of 2^n bins. Note that the depth of a decision tree could affect the size of the t-way test cases generated using our approach. A higher value (tree depth) can result in a significantly large number of t-way test cases, thus making it computationally expensive to identify fairness violations in ML models. As a trade-off, similar to LIME, a state-of-the-art XAI tool [8, 9], in our experiments, we limit the depth of the decision tree to a value of 3. Thus, a numeric attribute can be discretized into a maximum of 2^3 bins (8 bins). That is, a discretized numeric attribute will have at most eight bins.

In our experiments, we identify representative values for numerical attributes as follows: First, we calculate the total number of unique values for each numerical attribute. If the number of unique values is less than or equal to 8, we map the unique values as the representative values for the attribute. Otherwise, we use a decision tree algorithm with its default configuration value (not explicitly specifying the depth of the tree) and identify the representative values until either of the two conditions is satisfied. (1) the number of bins identified by the algorithm is less than or equal to 8; or (2) the number of bins is greater than 8. In the first case, we map the identified bins as the representative values for the attribute. In the second case, we re-execute the decision tree algorithm by setting the depth (of the tree) to 3 and discretizing the numerical attribute into eight bins. These bins are identified as representative values.

E. Constraints

In our experiments, we use Waikato Environment for Knowledge Analysis (WEKA), an open-source ML workbench tool, to identify the association rules from the training dataset. We preprocess the training dataset by converting all attributes to nominal datatype as required by WEKA.

Next, we provide the modified dataset (input) to WEKA and execute the algorithm (Apriori) with its default configuration values. The Apriori algorithm identifies a list of the top 10 association rules from the dataset. These association rules are used as constraints in the t-way test generation.

F. Test Generation

Using ACTS, a combinatorial test generation tool [4, 14], we generate t-way (t=2) abstracts which are then converted into concrete t-way tests.

G. Counterfactual Generation

We use DiCE, a state-of-the-art XAI tool to identify a counterfactual by modifying the values of protected attributes while retaining the values of the non-protected attributes [36]. The DiCE tool allows a user to generate a counterfactual based on specific attribute(s) [36]. Therefore, we provide a pre-trained model, the concrete t-way test instance, and protected attribute(s) as input to the DiCE tool. If successful, DiCE generates a test instance that is almost identical to the original instance but differs by the value of its protected attribute. In other words, DiCE identifies a scenario where the model predicts a different outcome on changing the protected attribute while retaining the value of all other attributes. This is considered as a fairness violation exhibited by the model.

H. Metrics

We assess our approach's effectiveness in terms of the number of fairness violations revealed by a t-way test set. The more fairness violations identified by a t-way test set, the more effective the t-way test set is considered.

I. Results and Discussion

We present and discuss our experimental results. The source code, data, and/or artifacts have been made available at [16].

Table I presents the results of fairness violations of ML models identified using t-way test sets. The Column headers in Table I are self-explanatory.

TABLE 1- FAIRNESS VIOLATIONS IDENTIFIED BY T-WAY TESTS

Datasets	Number of t-Way tests	Protected Attributes	Number of fairness violations			
			Logistic Regression (LR)	Random Forest (RF)	Support Vector Machines (SVM)	Deep Neural Network (DNN)
Adult Income	676	Sex, Race	58	25	23	11
German Credit	81	Sex, Age	5	10	9	26
COMPAS	64	Sex, Race	27	37	24	4

Recall that we identify fairness violations by perturbing each test instance using a counterfactual approach. Given a test case, if we successfully identify a counterfactual (a perturbation of the test case whose output differs from the original prediction), we consider that the test case resulted in a fairness violation.

Adult Income: For the *Adult Income dataset*, our approach generates 676 t-way tests (t=2). The results indicate that a substantial number of t-way test cases result in a fairness violation. For the LR-based model, out of 58 t-way tests, we observe that 27 t-way tests result in a fairness violation on modifying either one of the two protected attributes: 16 t-way tests cases result in a fairness violation on modifying attribute *Race* whereas 11 t-way test cases result in a fairness violation on modifying attribute *Sex*. The remaining 31 t-way test cases (out of 58) result in a fairness violation on modifying both of the protected attributes. In the case of the RF-based model, ten t-way tests result in a fairness violation on modifying the Race attribute, and one t-way test results in a fairness violation on changing the Sex attribute. Fourteen t-way tests result in a fairness violation on modifying the value of both the protected attributes. For the SVM-based model, out of 23 t-way tests, we observe that eight tests result in a fairness violation on modifying either one of the two protected attributes (Race = 5, Sex = 3), and 16 tests result in a fairness violation modifying both protected attributes.

German Credit: Based on the IPM derived from the *German Credit Dataset*, we generate 81 t-way test cases. The results indicate that t-way tests can detect fairness violations among models trained using different ML algorithms. We observe that across three ML models, t-way tests result in a fairness violation on modifying both the protected attributes (*Sex and Age*).

COMPAS dataset: Out of the 64 t-way test cases generated using our approach, for the RF-based ML model, more than 50% of t-way tests (37 t-way tests) result in a fairness violation. Among these 37 t-way tests, 19 t-way tests result in a fairness violation on modifying the *Race* attribute; three t-way tests result in a fairness violation on modifying the *Sex* attribute. The remaining 15 (out of 37) t-way tests result in a fairness violation on modifying both the protected attributes. For the LR-based ML model, 11 t-way tests result in a fairness violation on modifying the *Race* attribute, one t-way test results in a fairness violation on modifying the *Sex* attribute, and 15 t-way tests result in a fairness violation on modifying both the protected attributes. In the case of the SVM-based ML model, 18 tests result in a fairness violation on modifying both protected attributes, while the remaining six tests result in a fairness violation on modifying either one of the two protected attributes (Race = 5, Sex = 1).

For DNN based classifiers, we noticed DiCE takes a longer execution time to identify a counterfactual. For example, to identify a counterfactual for a pre-trained DNN model (trained with the Adult Income dataset), on average, DiCE takes 2 minutes per test case. So, it will take (676*2)/60= 22.5 hours to complete the execution. Therefore, for DNN models, we follow a brute-force approach as a workaround. That is, for each test case, using a script, we generate and execute all possible perturbations and identify if there exists a counterfactual by comparing the output with the original prediction.

Our results indicate that for the Adult Income dataset (DNN model), nine t-way tests result in a fairness violation on modifying either of the protected attributes (Race = 4, Sex = 5), while the remaining two tests result in a fairness violation on modifying both the protected attributes. For the German Credit dataset (DNN model), twelve t-way tests result in a fairness violation on modifying both the protected attributes. Additionally, fourteen t-way tests resulted in a fairness violation on modifying either *Age* (4 instances) or Sex (10 instances) attributes. For the COMPAS dataset (DNN model), we observed two tests result in a fairness violation on modifying both the protected attributes, while the other two tests result in a fairness violation on modifying either of the two protected attributes.

Overall, the results suggest the following major points:

(1) The t-way test sets generated based on the three subject datasets can identify fairness violations in pre-trained ML models.

(2) The results also indicate that the proposed approach can successfully detect fairness violations across different ML model architectures. Furthermore, on executing a t-way set across ML models (LR, RF, SVM, and DNN), we observe in the case of the Adult Income dataset and German Credit dataset, there is no overlap among the t-way test cases that result in a fairness violations across all four ML models. In the case of the COMPAS dataset, out of t-way tests that result in fairness violations, there are only four tests that are common among all ML models. This suggests a minimal overlap among the t-way test cases (from the test set) that resulted in a fairness violation across different ML models. That is, not the same set of t-way test cases triggers a fairness violation across ML models. We believe this indicates that t-way tests are effective in identifying biases introduced by both the training dataset and the learning algorithm.

(3) The results also indicate that the proposed approach can identify fairness violations with a relatively small number of test cases compared to the existing work [1]. We plan to perform a detailed comparison as part of future work.

J. Threats to Validity

Threats to external validity occur when the results from our experiments could not be generalized to other subjects. The datasets and ML models used in our study have been used in other studies in the fairness testing domain [1, 17, 19, 33]. We use four different algorithms (model architectures) to train ML models. This reduces the risk of a lack of representatives in the model architecture used in our study.

Threats to internal validity are factors that may be responsible for the experimental results without our knowledge. To mitigate the risk of human errors, we tried to automate the experimental procedure as much as possible. In particular, all the steps are automated except the identification and mapping of constraints. Further, we have manually verified some of the results if any surprising results occur. For example, on executing the t-way test cases (676 tests) generated for the *AdultIncome* dataset, 23 tests and 25 tests resulted in a fairness violation for SVM and RF models, respectively. However, for the LR model, we observed 58 tests (x2, compared to SVM and RF) that resulted in a fairness violation. In such a scenario, we manually verified the counterfactuals identified by DiCE.

V. Related Work

This section discusses existing work on fairness testing that is closely related to our work. First, we discuss existing work that focuses on testing individual discriminations of ML models. Udeshi et al. proposed *Aqeutias*, a testing technique to discover discriminatory inputs that result in an individual fairness violation [18]. In phase 1, they identify a set of discriminatory inputs from a test set generated by randomly sampling the input space (global search). In phase 2, they identify additional discriminatory inputs by changing the values of the non-protected attributes for the discriminatory instance found in the global search. Furthermore, they demonstrate that retraining the model with portions of the discriminatory inputs improves its performance. Our work is similar to theirs in generating test instances to identify individual fairness violations. However, our work differs in the following two ways: 1) Aqueitas generate test instances using a random testing approach, whereas we use a combinatorial approach to generate test instances. 2) Their approach identifies discriminatory instances from the random test set, and they (discriminatory instances) are further perturbated by searching the neighborhood. In contrast, we identify discriminatory instances using a counterfactual approach by perturbating the protected attributes defined by the user.

Galhotra et al. proposed THEMIS, a causality-based technique to measure discrimination in software [19]. They use a random test generation technique to identify discriminatory test instances. In contrast, we use combinatorial testing, a systematic test generation technique to generate test cases and identify fairness violations. Zhang et al. proposed an approach that generates discriminatory test instances for Deep Neural Network (DNNs)-based models [20]. Their work adopts a gradient descent and clustering-based approach to identify individual discriminatory instances. In contrast, we use a combinatorial testing-based approach to generate test instances. Their approach focuses on testing individual fairness violations in DNN models, whereas our approach is model agnostic and can be used to test ML models trained using different architectures.

Aggarwal et al. proposed an approach to generate test inputs and identify individual discrimination in ML models [1]. Their approach uses a combination of symbolic execution and LIME, a local explainer tool to generate test instances and identify individual discriminations. Once they identify a test instance that identifies individual discrimination, they perturb the test instance further by modifying its non-protected attributes and generating additional test instances to test the ML model for fairness violations. Similar to their work, the goal of our approach is to generate test instances and identify individual discriminations in an ML model. However, our work differs in the following ways: 1) We generate test instances using a combinatorial test approach. 2) Our approach identifies individual discrimination by perturbating a test instance using a counterfactual approach. 3) Furthermore, we do not perturb discriminatory inputs further.

Next, we discuss the existing literature on applying combinatorial testing for fairness testing. Morales et al. proposed Coverage-Guided Fairness Testing (CGFT) that aims to improve the performance of *Aequitas*, a testing technique that identifies individual discrimination in a two-step test generation process, namely *global search* and *local search*. The CGFT aims to leverage combinatorial testing by replacing the random test generation process in the *global search* phase of *Aequitas* with a t-way test generation approach and reduce the execution cost [17]. Our work is similar to theirs in using combinatorial testing to generate t-way test cases that are later used to identify individual discrimination. However, our work differs in the following way. CGFT uses an algorithm to control the number of t-way test cases generated. Hence, the test set generated using their algorithm has a combination of mixed strength t-way test cases. In contrast, all test cases generated in our approach belong to the same test strength (t=2). Furthermore, they do not use constraints in their test generation process. In contrast, we derive (from the training dataset) and use constraints in our test generation process. We believe using constraints enables our approach to generate valid and realistic t-way test cases compared to their approach.

We also note that there is a significant number of existing studies in literature, and we refer the reader to [21, 22] for a comprehensive report on existing work on fairness testing for machine learning systems.

VI. Conclusion and Future work

In this paper, we presented a combinatorial approach to identify individual fairness violations in pre-trained ML models. Our approach consists of two phases. In the first phase, based on the training dataset, we develop an IPM, derive constraints, and generate t-way test instances. In the second phase, we identify fairness violations by perturbing the t-way instances using a counterfactual approach. The key idea is to generate counterfactuals by modifying the protected attribute(s) while retaining the value of non-protected attributes of the t-way test instance. We performed an experimental evaluation of our approach using twelve ML classifiers (4 ML classifiers * 3 datasets). Our results suggest that our approach can successfully identify fairness violations

in ML models. Furthermore, our approach identifies a substantial number of fairness violations for different ML model classifiers. This suggests t-way tests are effective in identifying biases introduced by both the training dataset and the learning algorithm.

There are a few directions to continue our work. First, in our current approach, for a categorical attribute, we identify and map all its unique values as representative values in IPM. A significantly large number of t-way tests cases will be generated if the training dataset consists of a categorical attribute(s) with many unique values. We plan to investigate how to adapt the entropy-based discretization technique for categorical attributes. Second, after we detect fairness violations from a model, the next step is to identify the root cause and modify and/or retrain the model to remove those violations. We plan to explore how to leverage the t-way instances that identified fairness violations for model debugging and for model modification and retraining activities. Third, we plan to extend this approach to identify group discrimination in pre-trained ML models. Finally, we plan to conduct more empirical studies to further evaluate the effectiveness of our approach. In particular, we plan to compare the effectiveness of our approach to existing approaches such as the symbolic generation (SG) approach [1] and the CGFT approach [17].

ACKNOWLEDGMENT

This work is supported by research grant (70NANB21H092) from Information Technology Lab of National Institute of Standards and Technology (NIST).

Disclaimer: Certain software products are identified in this document. Such identification does not imply recommendation by the NIST, nor does it imply that the products identified are necessarily the best available for the purpose.

REFERENCES

[1] Aggarwal, Aniya, et al. "Black box fairness testing of machine learning models." Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019

[2] Amazon scraps secret AI recruiting tool that showed bias against women, https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G, Accessed: 2022-01-29

[3] Chakraborty, J., Xia, T., Fahid, F. M., & Menzies, T. (2019). Software engineering for fairness: A case study with hyperparameter optimization. *arXiv preprint arXiv:1905.05786*.

[4] Combinatorial Testing, https://csrc.nist.gov/Projects/automated-combinatorial-testing-for-software, Accessed: 2022-01-24

[5] Bellamy, Rachel KE, et al. "AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias." *arXiv preprint arXiv:1810.01943* (2018).

[6] Kerber, R. (1992, July). Chimerge: Discretization of numeric attributes. In *Proceedings of the tenth national conference on Artificial intelligence* (pp. 123-128).

[7] Trusted-AI/AIF360: A comprehensive set of fairness measures for datasets and machine learning models, https://github.com/Trusted-AI/AIF360, Accessed: 2022-01-24

[8] Lime | discretize.py, https://github.com/marcotcr/lime/blob/fd7eb2e6f760619c29fca0187c07b82157601b32/lime/discretize.py#L205, Accessed: 2022-01-23

[9] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135-1144).

[10] The Algorithm That Beats Your Bank Manager, https://www.forbes.com/sites/parmyolson/2011/03/15/the-algorithm-that-beats-your-bank-manager/?sh=1b04a7be1ae9, Accessed: 2022-01-29.

[11] Machine Bias – Propublica, https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, Accessed: 2022-01-29.

[12] Miranda Bogen and Aaron Rieke. 2018. *Help wanted: an examination of hiring algorithms, equity*. Technical Report. and bias. Technical report, Upturn.

[13] Understanding Bias in AI-Enabled Hiring, https://www.forbes.com/sites/forbeshumanresourcescouncil/2021/10/14/understanding-bias-in-ai-enabled-hiring/?sh=190037757b96, Accessed: 2022-01-29

[14] Yu, L., Lei, Y., Kacker, R. N., & Kuhn, D. R. (2013, March). Acts: A combinatorial test generation tool. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation* (pp. 370-375). IEEE.

[15] Hort, M., Zhang, J. M., Sarro, F., & Harman, M. (2021, August). Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 994-1006).

[16] Patelankitar/tWayFairnessTesting, https://github.com/patelankitar/tWayFairnessTesting, Accessed: 2022-01-29.

[17] Morales, D. P., Kitamura, T., & Takada, S. (2021, February). Coverage-Guided Fairness Testing. In *International Conference on Intelligence Science* (pp. 183-199). Springer, Cham.

[18] Udeshi, S., Arora, P., & Chattopadhyay, S. (2018, September). Automated directed fairness testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (pp. 98-108).

[19] Galhotra, S., Brun, Y., & Meliou, A. (2017, August). Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (pp. 498-510).

[20] Zhang, P., Wang, J., Sun, J., Dong, G., Wang, X., Wang, X., ... & Dai, T. (2020, June). White-box fairness testing through adversarial sampling. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (pp. 949-960).

[21] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, *54*(6), 1-35.

[22] Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*.

[23] Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data* (pp. 207-216).

[24] Brun, Y., & Meliou, A. (2018, October). Software fairness. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 754-759).

[25] Burnett, M., Stumpf, S., Macbeth, J., Makri, S., Beckwith, L., Kwan, I., ... & Jernigan, W. (2016). GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers*, *28*(6), 760-787.

[26] Zhang, J. M., & Harman, M. (2021, May). "Ignorance and Prejudice" in Software Fairness. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 1436-1447). IEEE.

[27] Ledford, H. (2019). Millions of black people affected by racial bias in health-care algorithms. *Nature*, *574*(7780), 608-610.

[28] Housing Discrimination Under the Fair Housing Act|HUD, https://www.hud.gov/program_offices/fair_housing_equal_opp/fair_housing_act_overview, Acceessed: 2022-01-30

[29] UCI Machine Learning Repository: Adult Data Set | https://archive.ics.uci.edu/ml/datasets/adult, Accessed: 2022-01-30

[30] UCI Machine Learning Repository: Statlog (German Credit Data) Data https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data), Accessed: 2022-01-30

[31] Propublica/compas-analysis: Data and Analysis for 'Machine Bias' | https://github.com/propublica/compas-analysis, Accessed: 2022-01-30

[32] Chakraborty, J., Majumder, S., Yu, Z., & Menzies, T. (2020, November). Fairway: A way to build fair ml software. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 654-665)

[33] Hort, M., Zhang, J. M., Sarro, F., & Harman, M. (2021, August). Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 994-1006)

[34] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., ... & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, *58*, 82-115.

[35] Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., & Turini, F. (2019). Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, *34*(6), 14-23.

[36] Mothilal, R. K., Sharma, A., & Tan, C. (2020, January). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*(pp. 607-617).

[37] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

[38] Sklearn.tree.DecisionTreeClassifier – scikit-learn-1.0.2, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, Accessed: 2022-01-22

[39] Buolamwini, J., & Gebru, T. (2018, January). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency* (pp. 77-91). PMLR

[40] Supervised Binning, https://www.saedsayad.com/supervised_binning.htm, Accessed: 2022-01-30