SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

BiasTrap: Runtime Detection of Biased Prediction in Machine Learning Systems

Hussaini Mamman[1,2,*], Shuib Basri[1], Abdullateef Oluwagbemiga Balogun[1], Abdullahi Abubakar Imam[3], Ganesh Kumar[1], and Luiz Fernando Capretz[4]

[1] Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 32610 Perak, Malaysia
[2] Department of Management and Information Technology, Abubakar Tafawa Balewa University, Bauchi, 740272, Nigeria
[3] Department of Computer Sciences, Universiti Brunei Darussalam, BE1410, Brunei Darussalam
[4] Department of Electrical and Computer Engineering, Western University, London, ON N6A 5B9, Canada

ARTICLE INFO

Article history:
Received 22 June 2023
Received in revised form 23 September 2023
Accepted 21 November 2023
Available online 28 February 2024

Keywords:
Fairness Testing; Runtime Verification; Bias Detection; Machine Learning Systems

ABSTRACT

Machine Learning (ML) systems are now widely used across various fields such as hiring, healthcare, and criminal justice, but they are prone to unfairness and discrimination, which can have serious consequences for individuals and society. Although various fairness testing methods have been developed to tackle this issue, they lack the mechanism to monitor ML system behaviour at runtime continuously. This study proposes a runtime verification tool called BiasTrap to detect and prevent discrimination in ML systems. The tool combines data augmentation and bias detection components to create and analyse instances with different sensitive attributes, enabling the detection of discriminatory behaviour in the ML model. The simulation results demonstrate that BiasTrap can effectively detect discriminatory behaviour in ML models trained on different datasets using various algorithms. Therefore, BiasTrap is a valuable tool for ensuring fairness in ML systems in real time.

1. Introduction

Machine learning (ML) systems are widely utilised by various entities, including businesses and governments, to enhance their operational effectiveness and support decision-making in domains such as risk prediction [1], school admission [2], loan approval [3] and hiring [4]. Using algorithmic decision-making offers numerous benefits, as machines can rapidly process massive amounts of data[5]. However, there is increasing criticism of the lack of fairness in ML-based decision-making [6, 7]. ML systems have been found to have discriminatory implications on individuals and groups based on characteristics such as gender and race [8–10]. For instance, the COMPAS software employed by US courts to evaluate the probability of recidivism has exhibited racial bias, with black defendants being assessed with a higher risk of reoffending than their white counterparts [1]. Additionally,

* *Corresponding author.*
E-mail address: hussaini_21000736@utp.edu.my

https://doi.org/10.37934/araset.40.2.127139

gender bias has been identified in job search algorithms, with male applicants presented with higher-paying job opportunities than female applicants [11].

Fairness in software refers to ensuring that the output of software systems does not discriminate against any individual or group based on their personal characteristics (or sensitive attributes) such as race, gender, or age [8]. Sensitive attributes are illegal to be considered as part of the decision-making process in some domains, such as loan decisions [12,13]. The concept of fairness has become increasingly important in recent years [9].

The recognition of fairness as a crucial non-functional property of ML systems has made it an integral part of software quality [10,14]. Research has demonstrated that removing sensitive such as gender and ethnicity does not guarantee fair outcomes [15]. This is due to the indirect influence of sensitive attributes, where even if race is not explicitly used, a prediction model trained on address as a feature might make unfair determinations for people of a specific race who live in a certain area [16,17]. Thus, it is crucial to conduct extensive fairness testing to identify and mitigate discriminatory behaviour, promoting fairness and trustworthiness in ML systems [18].

Studies have developed fairness testing methods to detect and prevent discrimination in ML systems, including Themis [19], AEQUITAS [20], SG [21], CGFT [22] and ExGA [23]. These fairness testing methods generate test cases from the dataset used to train ML models and then check the fairness of the ML system before deployment by altering one of the sensitive attributes to uncover any discriminatory behaviour. While these existing fairness testing methods are useful for detecting discriminatory behaviour in ML systems before deployment, they do not provide a mechanism for continuously monitoring fairness at runtime while the ML system operates. This challenges maintaining fairness in dynamic and evolving environments where ML systems are often deployed [24]. Therefore, there is a critical need to develop a method that allows for runtime verification of ML system behaviour to identify instances of unfairness in the decision-making process of such systems. The objective is to answer the question: *Would the predictions made by the deployed ML model be different for a person if they belonged to protected attributes different from their current one for every prediction made by the model?* To our knowledge, no existing tool can identify biased predictions during runtime for ML systems built using tabular data.

This study proposes BiasTrap to address this gap. BiasTrap is a tool that is designed to monitor a deployed ML system and detect discrimination on every run of the system based on sensitive attributes. The continuous monitoring of ML systems can ensure fairness and prevent discrimination. BiasTrap takes a data instance sent for prediction and creates a Cartesian product of the instance based on the number of pre-defined sensitive attributes and their respective values. By doing this, BiasTrap can detect discrimination on multiple sensitive attributes, a feature that is not commonly covered in many fairness testing studies. We asked **how effectively BiasTrap detects discrimination in ML systems at runtime.**

In addition, BiasTrap can provide detailed information about the level of discrimination present in a given instance. Specifically, the tool can report the number of instances of discrimination seen in a particular instance, called the discrimination list in this study. This enables further analysis and to take prompt corrective action to ensure that the ML system remains fair and unbiased. Therefore, we asked the question: **how many discrimination list can be produced by BiasTrap?**

The remaining sections of this paper are arranged in the following manner: Section 2 provides a background on the concept of fairness and problem definition. Section 3 presents the methodology, highlighting the proposed tool's framework. Section 4 demonstrates the experimental setup procedures and Section 5 evaluates the result of our experiment. Section 6 presents the present status of fairness testing research and lastly, Section 7 concludes with final thoughts and describes plans for future work.

2. Background

2.1 Fairness in ML Systems

In decision-making, fairness means being unbiased and not showing preference toward a person or group based on their inherent or acquired characteristics [8]. These characteristics that need protection from unfairness are known as protected attributes or sensitive attributes. Examples of legally recognised protected classes include race, colour, sex, religion, national origin, citizenship, age, pregnancy, familial status, handicap status, veteran status, and genetic information [25]. Ensuring fairness has become critical ML-based software to avoid discrimination against certain people, especially minorities [26].

In ML, fairness can be divided into two main categories: individual and group. *Individual fairness* ensures that similar individuals receive similar outcomes, regardless of their protected attributes, such as race or gender. This type of fairness is based on the idea that individuals should be treated as individuals and not judged based on their membership in a particular group [6,19]. On the other hand, *group fairness* ensures that different groups are treated similarly, regardless of their characteristics. This type of fairness is based on the idea that certain groups have been historically disadvantaged and, therefore, should be protected from further discrimination [6,27,28].

It is crucial to consider both individual and group fairness when developing and deploying ML models. A model that only considers individual fairness may inadvertently discriminate against certain groups. An example of this is Amazon's ML hiring system, which showed a preference for male candidates over female ones [4]. In contrast, a model that only considers group fairness may unfairly treat individuals differently, even if they are similar in their relevant characteristics.

This paper uses bias, discrimination, and unfairness interchangeably with similar meanings, as is common in the literature on software fairness.

2.2 Fairness Testing

Fairness testing is a field of software testing that focuses on detecting fairness bugs in ML systems that cause a discrepancy between obtained and required fairness conditions [29,30]. There are two types of fairness testing: offline testing, which evaluates the fairness of a model based on the training data it received, and online testing, which monitors and assesses the fairness of a deployed ML system in a real-world environment. Online fairness testing is crucial for maintaining fairness in decision-making and detecting any biases that may have infiltrated the system over time, enabling corrective action to be taken promptly [29,31].

2.3 Runtime Verification

Runtime Verification (RV) is a technique that involves analysing a system while it is running to determine if it meets a specific set of correctness criteria. The primary goal of this method is to verify whether the system behaves in the way it is intended to during its operation [32]. RV is a rigorous formal method that analyses the behaviour of software during runtime. It complements traditional exhaustive verification techniques such as model checking and theorem proving with a more practical approach that examines a single execution trace of a system. Although RV has limited execution coverage, it can provide precise information on the system's runtime behaviour [33].

The process of runtime verification involves generating a monitor for a given property, instrumenting the system under scrutiny to create events for the monitor, and analysing the system's

execution through the monitor's analysis of the events. This can occur during the execution or afterwards, assuming events are written to a log [32].

2.4 Problem Definition

This study focuses on the scenario where an ML model $f = < X, A >$ is deployed in a remote environment, trained on a dataset $X = \{x_1, x_2, \dots, x_n\}$ where each input sample $x \in X$ is represented as a d-dimensional feature vector. The feature space $A = \{a_1, a_2, \dots, a_n\}$ contains features $a \in A$, where a contains protected features $P \in a$, where $P = \{p_1, p_2, \dots, p_n\}$. Protected features are sensitive attributes that should not be used to discriminate against individuals unfairly. Each p_i is associated with a set of values S called the domain of p_i, represented by $S(p_i)$, such that $\big(S(p_i)\big)\ i \in$ n is pairwise disjoint. An input x_i exhibits discrimination or bias when two samples exist x and x' such that their attributes satisfy certain conditions as expressed in equations (1)-(3). We assumed a task where each x undergoes a binary classification, $\hat{Y} = f < X, Y >$, in which f is a function of variables known at decision time, $f: X \rightarrow \{0, 1\}$, and the actual outcome represented by $Y = (y_1, y_2, \dots, y_n)$ is unknown.

$$\exists a_i \in x, \forall p_k\ \omega\ P, a_i(k) \neq a_i'(k) \quad (1)$$

$$\forall a_j \in x, \forall p_l\ \underline{\omega}\ P, a_j(l) = a_j'(l) \quad (2)$$

$$f(x) \neq f(x') \quad (3)$$

Definition 1: Fairness

Consider a set of individuals represented by X, and a set of possible labels for everyone represented by Y. Let P define a set of sensitive attributes such as gender, race or ethnicity, and let Z represent a set of non-sensitive attributes. The goal of an ML predictive model represented by D is to make accurate predictions of Y given X and Z, while ensuring that P is not used in a discriminatory manner in making these predictions. The training data used to develop the model should not contain any bias or discrimination based on P, and the model should be evaluated for fairness to ensure that it is not discriminating against any individuals or groups based on their sensitive attributes.

$$F(D) = Pr[Y = y|X, P = p] = Pr[Y = y|X, P = p'] \quad (4)$$

Definition 2: Multi-attribute discrimination

Let $p_1, p_2, \dots, p_n$ be the n sensitive attributes of interest and let $s_1, s_2, \dots, s_n$ represent the different values or levels of these attributes. The set of multi-attributes that are defined by the sensitive attributes can be expressed as the Cartesian product $S = s_1 \times s_2 \times \dots \times s_n$, which is a set of all possible combinations of the entries of s. Multi-attribute discrimination refers to the unfair treatment of individuals based on multiple protected attributes, such as race, gender, age, etc. [34,35]. It occurs when an ML system unfairly discriminates against specific individuals or groups based on their combinations of protected attributes, resulting in inequitable outcomes for certain subgroups of the population. This type of discrimination is also called intersectional discrimination, as it occurs at the intersection of multiple protected attributes [36,37].

Definition 3: Discrimination list

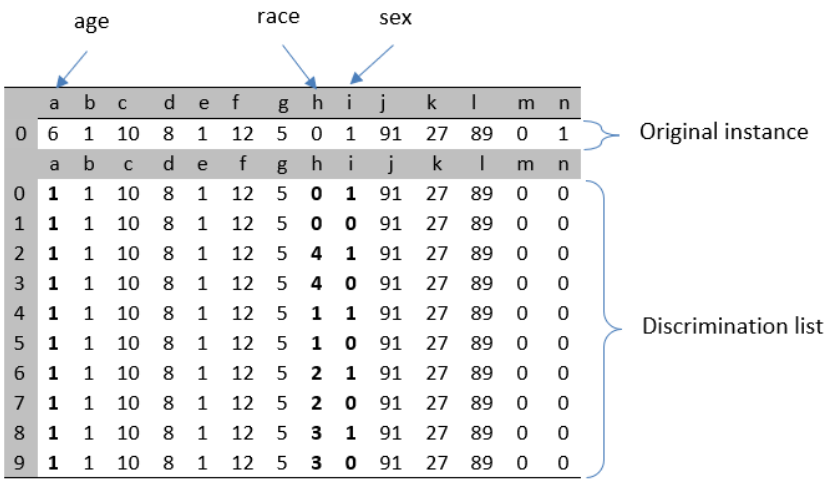
Let x be the original instance with non-protected attributes denoted by Z and protected attributes denoted by S. Further, let f be the trained ML model and Y be the output label. Generating new instances based on the combination of protected attributes and their values while keeping the non-protected attributes constant can be represented as $x' = (Z, S')$, where S' represents the new values of the protected attributes. If there exist instances x' such that $f(x') \neq f(x)$, then those instances form the discrimination list.

A discrimination list refers to instances that exhibit different prediction outcomes from the original instance after generating new instances by keeping the non-protected attributes constant and varying the protected attributes. In other words, the discrimination list consists of instances predicted differently from the original instance due to the variation in the protected attributes. We infer that these individuals would have received dissimilar classification results based on their protected attributes despite having similar non-protected features. Figure 1 represents this scenario.

$$D = \{x' : f(x') \neq f(x), x' = (Z, S')\} \tag{5}$$

Where X is the original instance, Z is the non-protected attributes, S is the protected attributes, X' is the generated instance, and f represents the trained ML model.

Based on our experiment, we found that the discrimination list can vary based on an instance. This suggests that the combination of non-protected attributes can significantly influence the prediction outcomes.



	a	b	c	d	e	f	g	h	i	j	k	l	m	n
0	6	1	10	8	1	12	5	0	1	91	27	89	0	1
	a	b	c	d	e	f	g	h	i	j	k	l	m	n
0	**1**	1	10	8	1	12	5	**0**	**1**	91	27	89	0	0
1	**1**	1	10	8	1	12	5	**0**	**0**	91	27	89	0	0
2	**1**	1	10	8	1	12	5	**4**	**1**	91	27	89	0	0
3	**1**	1	10	8	1	12	5	**4**	**0**	91	27	89	0	0
4	**1**	1	10	8	1	12	5	**1**	**1**	91	27	89	0	0
5	**1**	1	10	8	1	12	5	**1**	**0**	91	27	89	0	0
6	**1**	1	10	8	1	12	5	**2**	**1**	91	27	89	0	0
7	**1**	1	10	8	1	12	5	**2**	**0**	91	27	89	0	0
8	**1**	1	10	8	1	12	5	**3**	**1**	91	27	89	0	0
9	**1**	1	10	8	1	12	5	**3**	**0**	91	27	89	0	0

Fig. 1. An example of a discrimination list

3. Methodology

The proposed tool, BiasTrap, is a runtime verification tool to ensure fairness in ML systems. In Figure 2, the authors demonstrate the integration of BiasTrap with an ML system, which houses models trained on tabular datasets and accessed via a REST API. BiasTrap comprises two main components: data augmentation and bias detection.

3.1 ML system

An ML system is a system that incorporates one or more ML models as part of its components [38,39]. In practice, an ML model is often integrated into a larger software system, which may also include conventional software components responsible for tasks such as user interaction, input preprocessing, and functional logic that cannot be directly inferred from the training data. [40,41].

This study integrates BiasTrap into a Django back-end system where all the ML models are deployed. BiasTrap monitors each run of the predictive models to uncover biased outcomes.

3.2 Data Augmentation Component

Data augmentation is a common technique in ML used to increase the size of a dataset by generating additional instances with the same features as the original instances but with some variations. The goal is to improve the robustness of the model and its ability to generalise to new, unseen data. In the context of fairness, the data augmentation component specifically focuses on generating more instances with the same features as the original instance but with different sensitive attributes. Suppose an instance to be predicted has certain sensitive attributes, such as race and gender. In that case, the data augmentation component will generate additional instances with the same features but different sensitive attributes. For example, suppose an instance to be predicted has sensitive attributes of “black” and “female”, respectively. In that case, the data augmentation component can generate additional instances with the same features but with sensitive attributes of “white male”, “black male”, and “white female” to ensure that the model is not biased towards an individual based race and gender. This technique helps to ensure that the model’s predictions are fair and unbiased for all individuals, regardless of their sensitive attributes.

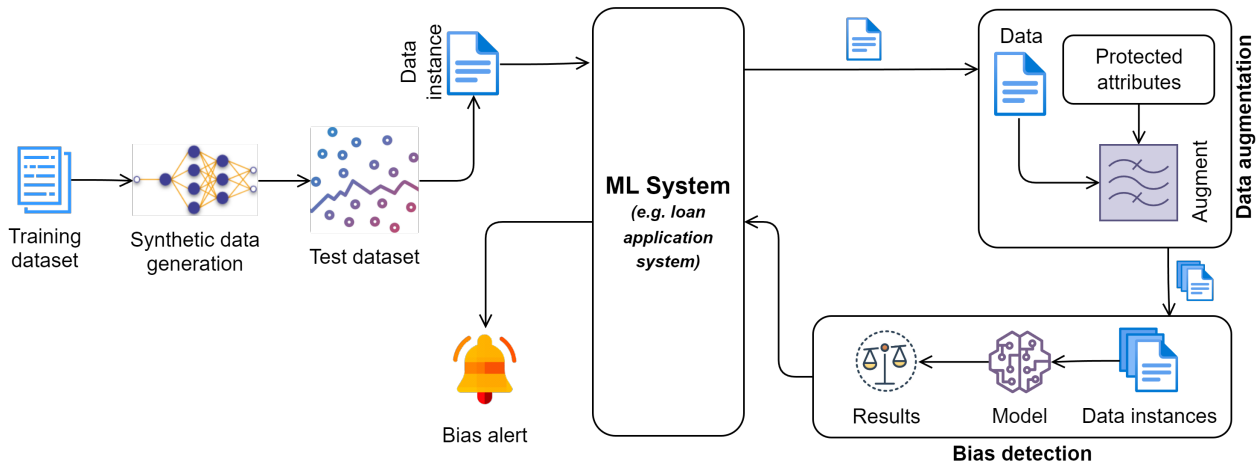


Fig. 2. Example of integrating BiasTrap with an ML system

3.3 Bias Detection Component

The bias detection component is crucial in ensuring that an ML system is fair and unbiased. Without this component, it would be difficult to identify discriminatory behaviour and protect against harmful biases in the system.

Instances produced in the data augmentation component are fed into the bias detection component. The bias detection component then passes all instances, including the original and generated instances, to the ML model for prediction. If any instances produce a different prediction

result than the original instance, the original instance is considered discriminatory. BiasTrap generates a warning alert to notify a user when a discriminatory instance is discovered. By alerting the user to potential discriminatory behaviour, BiasTrap allows them to take corrective action to ensure that their model is not unfairly biased against certain individuals or groups. This can involve revisiting the training data or modifying the model's parameters to ensure fairness.

4. Experimental Setup

4.1 Datasets

We conducted experiments using three commonly used public benchmark datasets to evaluate algorithmic fairness. The datasets are described as follows:

- **Census Income Dataset:** This dataset contains 32,561 samples and 13 features, including three sensitive attributes - *gender*, *age*, and *race*. The labels indicate whether an adult's income is above or below $50K.
- **German Credit Dataset:** This dataset includes 600 samples with 20 various features. It is used to evaluate the credit level of applicants (i.e., good or bad) based on their personal information. The sensitive attributes are *gender* and *age*.
- **Compas:** This dataset is used to predict the possibility of a defendant's recidivism based on their criminal history, demographics, and other relevant factors. The dataset consists of 7,214 samples and 10 features. The sensitive attributes are sex and race.

4.2 Test Data Generation

To evaluate the performance of BiasTrap, we simulated its effectiveness by creating synthetic datasets using CTGANs [42]. Synthetic datasets are computer-generated datasets that imitate the statistical properties of real-world datasets but are not derived from real-world events [43]. These synthetic datasets are created using generative models that learn the statistical structure of real-world datasets and generate new synthetic data samples based on that structure [44]. In this study, CTGAN was utilised as a set of synthetic data generators for tabular data, which utilises deep learning techniques to learn from real data and produce synthetic data that closely resembles the original data. As a result, the synthetic data created by CTGAN exhibit high fidelity to the original data.

By employing this approach, we can obtain a distinct test dataset that accurately reflects the characteristics of actual users for a common ML system, despite being distinct from the training dataset. Some result of the synthetic (fake) test dataset generated from the COMPAS dataset could be seen in Figure 3 & 4. We used 1,000 number of epochs and generated 4,000 samples.

4.3 Subject Models

Our approach is evaluated by employing four ML classifiers that are commonly used in fairness testing studies [19–21,23]. They are Decision Tree (DT), Random Forest (RF), Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP). XGBoost (XGB) classifier is also included in the evaluation. Each classifier is trained on the four datasets, thus, producing sixteen classifiers for the experiments. To maintain consistency with earlier studies [45], we train the ML classifiers using their default configuration, as provided by the sci-kit learn library [46].

4.4 Metrics

We measure the effectiveness of our approach by evaluating the number of fairness violations (discriminatory instances) exposed by our technique based on the combination of all protected attributes. The greater the number of discriminatory instances detected, the more effective our technique is considered. Additionally, we employ a discriminatory list to identify instances that are discriminated against from any data instance sent for prediction.

Our approach's effectiveness is determined by the number of discriminatory instances identified through the combination of all protected attributes. The greater the number of discriminatory instances detected, the more effective our technique is considered. Additionally, we employ a discriminatory list to identify instances that are discriminated against from any data instance sent for prediction.

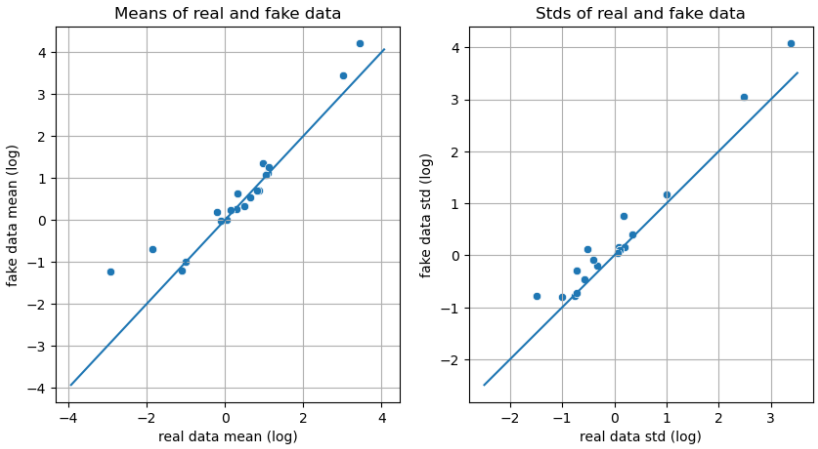


Fig. 3. Absolute log mean and standard deviation

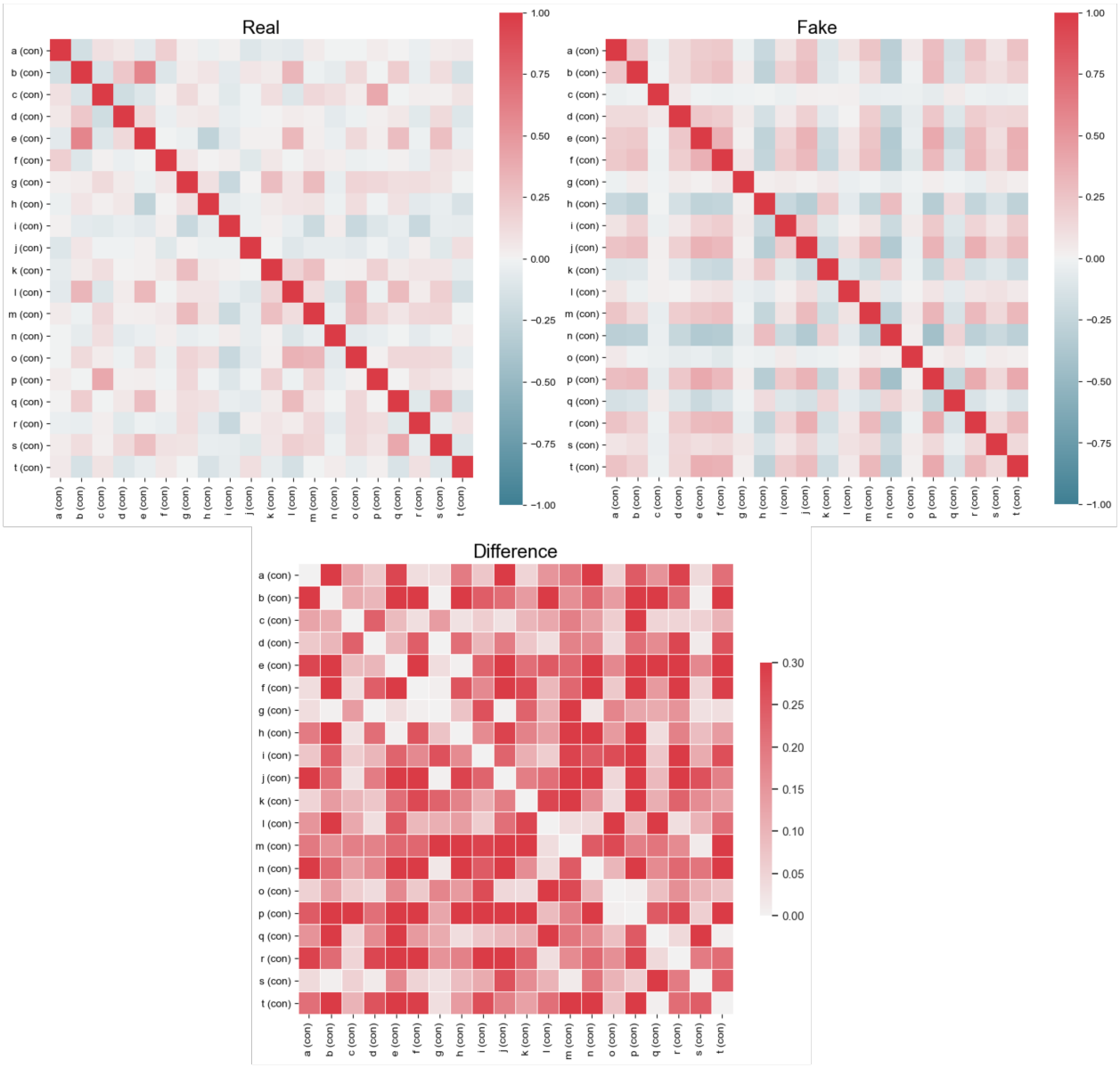


Fig. 4. Confusion matrix for the real and synthetic datasets and the difference between them

5. Evaluation Results

Table 1 shows the results of a simulation of the BiasTrap tool applied to several popular ML models on three different datasets: COMPAS, Credit, and Census. The simulation evaluated the performance of the models in terms of fairness and their ability to detect for any biases the models. We used discriminatory sample number (DSN) [20,23] and [47] discrimination list (DL) as a metrics. The simulation results indicate that BiasTrap tool was able to detect instances of discrimination in the models which shows its effectiveness.

For the COMPAS dataset, the decision tree model had the highest number of discriminatory instances with 2,297 for the discriminatory instances (Disc Inst). It also produced the highest number 66,565 for the discrimination listing (Disc List). BiasTrap performed less effective on random forest and SVM models, with 2,027 and 1,655 discriminatory instances for the Disc Inst and 48,331 and 31,691 for the Disc List, respectively. The multi-layer perceptron and XGBoost models had slightly

higher discriminatory instances than RF, with 2,140 and 2,189. It is interesting to note that MLP produced lesser number of Disc List 40,670 than RF model.

Table 1
Result of Runtime Verification of Fairness using BiasTrap

Datasets	Decision Tree		Random Forest		Support Vector Machine		Multi-layer Perceptron		XGBoost	
	Disc Inst	Disc List	Disc Inst	Disc List	Disc Inst	Disc List	Disc Inst	Disc List	Disc Inst	Disc List
COMPAS	2,297	66,565	2,027	48,331	1,655	31,691	2,140	40,670	2,189	49,462
Credit	1,149	6,580	1,024	4,527	2,991	10,493	2,597	10,028	1,281	5,837
Census	1,851	44,055	1,035	24,264	684	20,360	1,537	40,487	910	20,499

The SVM model had the highest number of discriminatory instances for the Credit dataset, with 2,991 and 10,493 for the Disc Inst and Disc List, respectively. The MLP model had the second-highest number of discriminatory instances, with 2,597 and 10,028 for the Disc Inst and Disc List, respectively. The decision tree and XGBoost models had the lowest discriminatory instances, with 1,149 and 1,281 for the Disc Inst and 6,580 and 5,837 for the Disc List, respectively.

For the Census dataset, the decision tree model had the highest number of discriminatory instances, with 1,851 and 44,055 for the Disc Inst and Disc List, respectively. The SVM model had the second-highest discriminatory instances, with 684 and 20,360 for the Disc Inst and Disc List, respectively. The MLP and XGBoost models had the lowest discriminatory instances, with 1,537 and 910 for the Disc Inst method and 40,487 and 20,499 for the Disc List method, respectively.

The simulation result suggests that BiasTrap can effectively detect discrimination in deployed ML models, highlighting the importance of using such tools to ensure fairness in ML applications. However, further investigation is necessary to identify the causes of discrimination and develop appropriate corrective measures.

6. Related Works

Several studies have proposed different methods to detect discriminatory instances in ML systems. For example, Themis [19] randomly samples the input space and assesses the frequency of discriminatory occurrences by observing the system's behaviour under test. AEQUITAS [20], a two-phase fairness testing technique, explores the input space for discriminatory instances in the global phase and generates more discriminatory samples by perturbing the non-protected attributes of instances in the local phase. SG [21] combines symbolic generation and local explainability to identify discriminatory instances. ExpGA [23] constructs seed instances using an interpretable method and applies genetic algorithm to produce many discriminatory instances efficiently. Patel et al. [48] use combinatorial t-way testing, which constructs an input parameter model from the training dataset and generates test cases to discover fairness violations. These approaches are based on offline fairness testing that evaluates the fairness of an ML system before its deployment.

On the other hand, our work differs from these approaches as it focuses on online fairness testing that evaluates the fairness of an ML system on every run of the system by applying runtime verification. In this regard, we propose a model that allows the user to specify multiple protected attributes in a tabular dataset, unlike BiasRV [24], which detects gender discrimination in sentiment analysis systems based on text data and uses only gender as the protected attribute.

7. Conclusions

This study presented a framework for runtime fairness verification in ML systems called the BiasTrap. BiasTrap uses data augmentation and bias detection components to monitor the behaviour of the ML system during runtime and identify any potential biases or discrimination in the decision-making process. The system was evaluated on commonly used datasets and multiple ML classifiers, including Decision Trees, Random Forest, Support Vector Machines, Multi-layer Perceptron, and XGBoost.

The findings of this study suggest that BiasTrap can be a valuable tool for ensuring fairness in ML systems, particularly in applications where discriminatory behaviour can have significant consequences. Further research is needed to evaluate the scalability and generalizability of the BiasTrap, as well as to address potential limitations and challenges, such as the trade-off between runtime verification and computational complexity. Nonetheless, the BiasTrap framework provides a promising direction for developing more transparent, trustworthy, fair and unbiased ML systems.

Acknowledgement
This research was fully funded by Universiti Teknologi PETRONAS (UTP), Malaysia, under the Short Term Internal Research Funding (STRIF) grant (**015LA0-037**).

References

[1] Angwin, Julia, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks. (2016)
[2] Feathers, Todd. "Major Universities Are Using Race as a "High Impact Predictor" of Student Success." *Ethics of Data and Analytics: Concepts and Cases* (2022): 268. https://doi.org/10.1201/9781003278290-39
[3] Counts, Laura. Minority homebuyers face widespread statistical lending discrimination. (2018)
[4] Dastin, Jeffery. Amazon scraps secret AI recruiting tool that showed bias against women. (2018)
[5] Pessach, Dana, and Erez Shmueli. "A review on fairness in machine learning." *ACM Computing Surveys (CSUR)* 55, no. 3 (2022): 1-44. https://doi.org/10.1145/3494672
[6] Dwork, Cynthia, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. "Fairness through awareness." In *Proceedings of the 3rd innovations in theoretical computer science conference*, pp. 214-226. 2012. https://doi.org/10.1145/2090236.2090255
[7] Zemel, Rich, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. "Learning fair representations." In *International conference on machine learning*, pp. 325-333. PMLR, 2013.
[8] Mehrabi, Ninareh, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A survey on bias and fairness in machine learning." *ACM computing surveys (CSUR)* 54, no. 6 (2021): 1-35. https://doi.org/10.1145/3457607
[9] Soremekun, Ezekiel, Mike Papadakis, Maxime Cordy, and Yves Le Traon. "Software fairness: An analysis and survey." *arXiv preprint arXiv:2205.08809* (2022).
[10] Brun, Yuriy, and Alexandra Meliou. "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. 2018. https://doi.org/10.1145/3236024.3264838
[11] Vincent, James. Gender and racial bias found in Amazon's facial recognition technology (again). (2019)
[12] Singh, Arashdeep, Jashandeep Singh, Ariba Khan, and Amar Gupta. "Developing a novel fair-loan classifier through a multi-sensitive debiasing pipeline: Dualfair." *Machine Learning and Knowledge Extraction* 4, no. 1 (2022): 240-253. https://doi.org/10.3390/make4010011
[13] Singh, Jashandeep, Arashdeep Singh, Ariba Khan, and Amar Gupta. "Developing a novel fair-loan-predictor through a multi-sensitive debiasing pipeline: DualFair." *arXiv preprint arXiv:2110.08944* (2021). https://doi.org/10.3390/make4010011
[14] Angell, Rico, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. "Themis: Automatically testing software for discrimination." In *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pp. 871-875. 2018. https://doi.org/10.1145/3236024.3264590

[15] Kamishima, Toshihiro, Shotaro Akaho, and Jun Sakuma. "Fairness-aware learning through regularization approach." In *2011 IEEE 11th International Conference on Data Mining Workshops*, pp. 643-650. IEEE, 2011. https://doi.org/10.1109/ICDMW.2011.83

[16] Phelps, Edmund S. "The statistical theory of racism and sexism." *The american economic review* 62, no. 4 (1972): 659-661.

[17] Zhang, Lu, Yongkai Wu, and Xintao Wu. "A causal framework for discovering and removing direct and indirect discrimination." *arXiv preprint arXiv:1611.07509* (2016). https://doi.org/10.24963/ijcai.2017/549

[18] Zhang, Mengdi, Jun Sun, Jingyi Wang, and Bing Sun. "TESTSGD: Interpretable testing of neural networks against subtle group discrimination." *ACM Transactions on Software Engineering and Methodology* 32, no. 6 (2023): 1-24. https://doi.org/10.1145/3591869

[19] Galhotra, Sainyam, Yuriy Brun, and Alexandra Meliou. "Fairness testing: testing software for discrimination." In *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*, pp. 498-510. 2017. https://doi.org/10.1145/3106237.3106277

[20] Udeshi, Sakshi, Pryanshu Arora, and Sudipta Chattopadhyay. "Automated directed fairness testing." In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 98-108. 2018. https://doi.org/10.1145/3238147.3238165

[21] Aggarwal, Aniya, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. "Black box fairness testing of machine learning models." In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 625-635. 2019. https://doi.org/10.1145/3338906.3338937

[22] Perez Morales, Daniel, Takashi Kitamura, and Shingo Takada. "Coverage-guided fairness testing." In *International Conference on Intelligence Science*, pp. 183-199. Cham: Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-79474-3_13

[23] Fan, Ming, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. "Explanation-guided fairness testing through genetic algorithm." In *Proceedings of the 44th International Conference on Software Engineering*, pp. 871-882. 2022. https://doi.org/10.1145/3510003.3510137

[24] Yang, Zhou, Muhammad Hilmi Asyrofi, and David Lo. "Biasrv: Uncovering biased sentiment predictions at runtime." In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1540-1544. 2021. https://doi.org/10.1145/3468264.3473117

[25] Humphreys, Stephen. "The Equality Act, 2010." *Research Ethics* 6, no. 3 (2010): 95-95. https://doi.org/10.1177/174701611000600306

[26] Makhlouf, Karima, Sami Zhioua, and Catuscia Palamidessi. "Machine learning fairness notions: Bridging the gap with real-world applications." *Information Processing & Management* 58, no. 5 (2021): 102642. https://doi.org/10.1016/j.ipm.2021.102642

[27] Binns, Reuben. "On the apparent conflict between individual and group fairness." In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 514-524. 2020. https://doi.org/10.1145/3351095.3372864

[28] Zhang, Hantian, Xu Chu, Abolfazl Asudeh, and Shamkant B. Navathe. "Omnifair: A declarative system for model-agnostic group fairness in machine learning." In *Proceedings of the 2021 international conference on management of data*, pp. 2076-2088. 2021. https://doi.org/10.1145/3448016.3452787

[29] Chen, Zhenpeng, Jie M. Zhang, Max Hort, Federica Sarro, and Mark Harman. "Fairness testing: A comprehensive survey and analysis of trends." *arXiv preprint arXiv:2207.10223* (2022).

[30] Tramer, Florian, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. "Fairtest: Discovering unwarranted associations in data-driven applications." In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 401-416. IEEE, 2017. https://doi.org/10.1109/EuroSP.2017.29

[31] Zhang, Jie M., Mark Harman, Lei Ma, and Yang Liu. "Machine learning testing: Survey, landscapes and horizons." *IEEE Transactions on Software Engineering* 48, no. 1 (2020): 1-36. https://doi.org/10.1109/TSE.2019.2962027

[32] Falcone, Yliès, Klaus Havelund, and Giles Reger. "A tutorial on runtime verification." *Engineering dependable software systems* (2013): 141-175.

[33] Bartocci, Ezio, Yliès Falcone, Adrian Francalanza, and Giles Reger. "Introduction to runtime verification." *Lectures on Runtime Verification: Introductory and Advanced Topics* (2018): 1-33. https://doi.org/10.1007/978-3-319-75632-5_1

[34] Ghadage, Adinath, Dewei Yi, George Coghill, and Wei Pang. "Multi-stage Bias Mitigation for Individual Fairness in Algorithmic Decisions." In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 40-52. Cham: Springer International Publishing, 2022. https://doi.org/10.1007/978-3-031-20650-4_4

[35] Roy, Arjun, Jan Horstmann, and Eirini Ntoutsi. "Multi-dimensional Discrimination in Law and Machine Learning-A Comparative Overview." In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pp. 89-100. 2023. https://doi.org/10.1145/3593013.3593979

[36] Foulds, James R., Rashidul Islam, Kamrun Naher Keya, and Shimei Pan. "An intersectional definition of fairness." In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1918-1921. IEEE, 2020. https://doi.org/10.1109/ICDE48307.2020.00203

[37] Buolamwini, Joy, and Timnit Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification." In *Conference on fairness, accountability and transparency*, pp. 77-91. PMLR, 2018.

[38] Siebert, Julien, Lisa Joeckel, Jens Heidrich, Adam Trendowicz, Koji Nakamichi, Kyoko Ohashi, Isao Namba, Rieko Yamamoto, and Mikio Aoyama. "Construction of a quality model for machine learning systems." *Software Quality Journal* 30, no. 2 (2022): 307-335. https://doi.org/10.1007/s11219-021-09557-y

[39] Villamizar, Hugo, Tatiana Escovedo, and Marcos Kalinowski. "Requirements engineering for machine learning: A systematic mapping study." In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 29-36. IEEE, 2021. https://doi.org/10.1109/SEAA53835.2021.00013

[40] Muccini, Henry, and Karthik Vaidhyanathan. "Software architecture for ML-based systems: What exists and what lies ahead." In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, pp. 121-128. IEEE, 2021. https://doi.org/10.1109/WAIN52551.2021.00026

[41] Riccio, Vincenzo, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. "Testing machine learning based systems: a systematic mapping." *Empirical Software Engineering* 25 (2020): 5193-5254. https://doi.org/10.1007/s10664-020-09881-0

[42] Xu, Lei, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. "Modeling tabular data using conditional gan." *Advances in neural information processing systems* 32 (2019).

[43] Yasar, Kinza, and Laskowski, Nicole. Synthetic data. (2023)

[44] Nikolenko, Sergey I. *Synthetic data for deep learning*. Vol. 174. Springer Nature, 2021. https://doi.org/10.1007/978-3-030-75178-4

[45] Hort, Max, Jie M. Zhang, Federica Sarro, and Mark Harman. "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. 2021.

[46] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* 12 (2011): 2825-2830.

[47] Wang, Zichong, Yang Zhou, Meikang Qiu, Israat Haque, Laura Brown, Yi He, Jianwu Wang, David Lo, and Wenbin Zhang. "Towards Fair Machine Learning Software: Understanding and Addressing Model Bias Through Counterfactual Thinking." *arXiv preprint arXiv:2302.08018* (2023).

[48] Patel, Ankita Ramjibhai, Jaganmohan Chandrasekaran, Yu Lei, Raghu N. Kacker, and D. Richard Kuhn. "A combinatorial approach to fairness testing of machine learning models." In *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 94-101. IEEE, 2022. https://doi.org/10.1109/ICSTW55395.2022.00030