AI Model Fairness Testing

Dan Williams

April 2025

1 Introduction

Artificial intelligence (AI) model usage has been increasing significantly in recent years to automate decision-making and reduce the amount of work required by humans. These models have been widely adopted in high-stakes areas such as financial decision-making, criminal justice, and employment. The predictions made in these areas can have serious consequences on individuals. However, these models often inherit biases from their training data which leads to unfair or discriminatory outcomes. Since these models can influence significant decisions it is crucial to ensure fairness to prevent unethical or legally problematic practices.[1]

Legally or ethically protected categories such as age, gender, or race, are referred to as sensitive features as these could influence outcomes in a way that leads to discrimination. We refer to other unprotected categories, such as workplace, hours worked per week, and education level, as non-sensitive features. We say that an AI model exhibits unfairness or bias when, given two identical inputs that differ only on sensitive features, it produces different outcomes. Consider the following two input samples:

$$x_1 = (\mathbf{0}, 8, 2, 4, 1), \quad x_2 = (\mathbf{1}, 8, 2, 4, 1)$$

where the feature at index 0 is a sensitive feature. If an AI model $f(\cdot)$ produces different predictions for these two inputs $f(x_1) \neq f(x_2)$ then we have detected a fairness bug, meaning that the model's decision is influenced by the sensitive feature rather than the non-sensitive attributes.

1.1 Fairness Testing

Fairness testing aims to detect such behaviour in AI models by generating and evaluating input samples. A common approach for this is using a random search. This approach generates test cases by randomly modifying input values and observing whether the model's prediction changes based on variations in sensitive attributes. As this approach randomly selects test cases, it is inefficient at discovering these discriminatory instances. We will introduce Genetic Algorithms (GA) to optimise this process. These iteratively refine a population of test cases based on a fitness criterion, favouring those that are more likely to reveal bias. GA applies evolutionary operations to generate new test cases, gradually modifying non-sensitive features to detect bias. After multiple generations our two input samples could become:

$$x'_1 = (\mathbf{0}, 5, 7, 4, 3), \quad x'_2 = (\mathbf{1}, 5, 7, 4, 3)$$

1.2 Goal and Evaluation Metrics

In this report, we will develop a GA-based fairness testing tool that will outperform random search in detecting bias. We will measure this using the Individual Discriminatory Instance (IDI) ratio. This is defined as:

$$\text{IDI} = \frac{I}{S}$$

where I is the number of unique individual discriminatory instances detected, and S is the total number of unique inputs generated (also referred to as the budget). A higher IDI ratio indicates that the fairness testing method is more effective at uncovering bias in the AI model.

1. Julia Stoyanovich, Bill Howe, and H. V. Jagadish, "Responsible data management" [in eng], *Proceedings of the VLDB Endowment* 13, no. 12 (2020): 3474–3488, ISSN: 2150-8097.

2 Related Work

2.1 Random Search

Random search serves as the baseline method for our study for fairness testing. It works by randomly selecting pairs of input samples that differ only in sensitive features (e.g gender, race) and evaluating whether the AI model produces different predictions. This would mean that the model has a fairness bug.

While it is simple to implement, it is inefficient and computationally expensive as test cases are chosen randomly and does not prioritise differences that are more likely to reveal fairness bugs which causes a high number of wasted comparisons. This makes it poor at detecting deeper patterns of bias which limits its effectiveness in fairness testing.

2.2 Combinatorial Testing

Combinatorial testing[2] is a method for generating test cases that samples inputs, configurations, and parameters and combines them in a systematic fashion. This technique ensures that all relevant interactions between sensitive and non-sensitive features are examined unlike random search which just blindly selects test inputs. This is useful because biases can often emerge from these interactions rather than just individual features.

The main advantage of combinatorial testing is its efficiency in covering a wide range of input variations with fewer test cases compared to exhaustive testing. However, the disadvantage that combinatorial testing is that it does not adapt based on the model's responses for each test case. It typically generates a fixed set of test cases covering feature interactions systematically and does not prioritise test cases that are more likely to reveal fairness issues based on discoveries from previous test cases.

2.3 Counterfactual Testing

Counterfactual testing[3] is a fairness evaluation method that examines whether a model changes its prediction when only a sensitive feature is changed. This is based on counterfactual fairness which ensures that an individual's decision outcome would remain the same regardless of their sensitive features.

While this provides a structured way to detect bias, it doesn't take into account feature influence. For example, some sensitive features might often influence other features so a simple swap (e.g changing male to female) might not accurately reflect a real-world counterfactual. This can also be computationally expensive, particularly in complex models with interdependent features.

2.4 AEQUITAS

AEQUITAS[4] is a state-of-the-art fairness testing approach that consists of a directed two-phase search method. It begins with a broad random exploration to identify at least one discriminatory case. It then conducts a local search in the neighbourhood of this case to uncover additional cases. This strategy has been shown to be far more efficient than random search as it can quickly identify subtle unfair behaviour such as when the prediction changes when a sensitive feature is altered. However, the behaviour of only performing a local search is a key limitation as it may overlook biases in other regions of the input space (local optima).

2.5 Adversarial Testing

Adversarial testing approaches fairness testing as an optimisation problem. It utilises the model gradients to generate discriminatory cases which is useful for deep neural networks. The Adversarial Discrimination Finder (ADF)[5] is an example of such an algorithm. It operates in two phases: a global search that perturbs inputs along the gradient to find discriminatory cases and then a local refinement that finds minimally different cases that give different predictions. It is able to explore significantly more of the input space by following the model's gradients and so it can discover more discriminatory cases compared to existing methods. However in order to access the model's gradients, it needs white-box access to the model which limits its applicability.

2. Ankita Ramjibhai Patel et al., "A Combinatorial Approach to Fairness Testing of Machine Learning Models," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (2022), 94–101, https://doi.org/10.1109/ICSTW55395.2022.00030.

3. Matt J Kusner et al., "Counterfactual Fairness" [in eng], 2017,

4. Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay, "Automated directed fairness testing," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE '18 (Montpellier, France: Association for Computing Machinery, 2018), 98–108, ISBN: 9781450359375, https://doi.org/10.1145/3238147.3238165, https://doi.org/10.1145/3238147.3238165.

5. Peixin Zhang et al., "Automatic Fairness Testing of Neural Classifiers Through Adversarial Sampling" [in eng], *IEEE transactions on software engineering* (New York) 48, no. 9 (2022): 3593–3612, ISSN: 0098-5589.

3 Solution

This section describes the design and implementation of our fairness testing method with the aim that we will beat the baseline of Random Search. We will use a Genetic Algorithm (GA) to efficiently generate test cases that will discover biases in our AI model. GAs[6] are evolutionary search methods particularly suitable for efficiently exploring large input spaces. By iteratively refining input samples based on a defined fitness criteria, GAs focus on test cases that are most likely to uncover biases which then improves the discovery rate of fairness bugs. To quantify bias, we define a fitness function F that measures the absolute difference in model predictions for two input samples x_1 and x_2 differing only in sensitive attributes:

$$F(x_1, x_2) = |f(x_1) - f(x_2)|$$

This function allows us to evaluate the extent of discrimination in the model's outputs where higher values of $F(x_1, x_2)$ indicate greater bias and these test cases will be more valuable for further exploration in the selection part of the algorithm. The fairness testing tool works as follows:

1. **Initialisation**: Generate an initial population of input pairs where each pair differs only in sensitive features and all non-sensitive features remain the same. This ensures that fairness bugs are only because of differences in the sensitive features.
2. **Fitness Evaluation**: Use the fitness function $F(x_1, x_2)$ to calculate the model's bias.
3. **Selection**: Using tournament selection, choose test pairs with the highest fitness scores. These pairs are more likely to expose further bias and will act as the parents for the next generation.
4. **Crossover**: Combine non-sensitive features from selected parent pairs to create new child pairs with probability P_c. This allows exploration of new feature combinations that might reveal biases that wouldn't have been detected using random generation.
5. **Mutation**: Apply small random changes (perturbations) to non-sensitive attributes in child pairs with probability P_m. This prevents the algorithm from becoming trapped in local optima.
6. **Evaluation**: Evaluate newly created child pairs using the fitness function to determine if they uncover previously undetected biases.
7. **Iteration**: Repeat the evolutionary process for a number of generations until the test budget is exhausted.

The rationale behind selecting GA for fairness testing is that it can efficiently search large input spaces as they prioritise test cases that will expose biases. On the other hand, Random Search blindly generates test cases and doesn't improve the test population after each iteration. As a result, this will maximise our bias detection.

4 Setup

We conduct our experiments using the eight real-world datasets shown in Table 1. These datasets cover a range of sensitive decision-making areas. Each dataset has a corresponding pre-trained deep neural network (DNN) model.

Dataset	**Domain**	$\lvert f_s \rvert$	$\lvert f \rvert$	**Available Size**	**Search Space**
ADULT	Finance	3	11	45,222	4.81×10^9
COMPAS	Criminology	2	13	6,172	1.45×10^8
LAW SCHOOL	Education	2	12	20,708	9.20×10^6
KDD	Criminology	2	19	284,556	4.13×10^{15}
DUTCH	Finance	2	12	60,420	3.58×10^7
CREDIT	Finance	3	24	30,000	2.01×10^{12}
CRIME	Criminology	2	22	2,215	4.19×10^8
GERMAN	Finance	2	20	1,000	8.85×10^9

Table 1: Real-world datasets where $|f_s|$ denotes the number of sensitive features and $|f|$ denotes the number of total features.

6. Scott Thede, "An introduction to genetic algorithms," *Journal of Computing Sciences in Colleges* 20 (October 2004).

4.1 Parameter Settings

For our Random Search baseline, we set the number of test cases to 1,000 per dataset. To compare the effectiveness of the Genetic Algorithm (GA), we use the parameter settings shown in Table 2.

Population Size	Generations	Tournament Size	Crossover Probability P_c	Mutation Probability P_m
100	50	3	0.8	0.5

Table 2: GA parameter settings used in experiments.

4.2 Statistical Analysis

To evaluate whether the Genetic Algorithm significantly outperforms the Random Search baseline, we conduct statistical hypothesis tests on the Individual Discriminatory Instance (IDI) ratios collected from 10 runs across each dataset. We first test the normality of the paired differences in IDI ratios using the Shapiro-Wilk test using `scipy.stats.shapiro()`.[7] If the data is normally distributed, we apply a paired t-test using `scipy.stats.ttest_rel()` otherwise we use the Wilcoxon signed-rank test, using `scipy.stats.wilcoxon()`. These tests were selected as they provide a statistically robust comparison suitable for repeated measures data.

5 Experiments

Tables 3 and 4 report the Individual Discriminatory Instances (IDI) ratios across 10 runs for each dataset using Random Search (RS) and Genetic Algorithm (GA). In most datasets, GA achieves significantly higher average IDI ratios compared to RS.

Dataset	**IDI**$_1$	**IDI**$_2$	**IDI**$_3$	**IDI**$_4$	**IDI**$_5$	**IDI**$_6$	**IDI**$_7$	**IDI**$_8$	**IDI**$_9$	**IDI**$_{10}$	**IDI**$_{AVG}$
Adult	0.430	0.433	0.448	0.442	0.424	0.451	0.400	0.413	0.415	0.433	0.423
Compas	0.062	0.064	0.072	0.057	0.062	0.050	0.067	0.077	0.068	0.072	0.065
Law School	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000
Kdd	0.042	0.034	0.032	0.028	0.027	0.034	0.037	0.032	0.025	0.035	0.033
Dutch	0.024	0.019	0.019	0.019	0.018	0.025	0.015	0.025	0.016	0.020	0.020
Credit	0.305	0.297	0.327	0.321	0.302	0.320	0.285	0.306	0.317	0.284	0.306
Crime	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
German	0.090	0.076	0.085	0.074	0.087	0.090	0.069	0.094	0.078	0.073	0.082

Table 3: IDI ratios across 10 runs for each dataset using Random Search. The final column shows the average IDI ratio **IDI**$_{AVG}$.

Dataset	**IDI**$_1$	**IDI**$_2$	**IDI**$_3$	**IDI**$_4$	**IDI**$_5$	**IDI**$_6$	**IDI**$_7$	**IDI**$_8$	**IDI**$_9$	**IDI**$_{10}$	**IDI**$_{AVG}$
Adult	0.830	0.890	0.940	0.880	0.860	0.980	0.880	0.790	0.960	0.900	0.891
Compas	0.840	0.760	0.760	0.830	0.720	0.790	0.780	0.810	0.770	0.890	0.795
Law School	0.000	0.000	0.070	0.000	0.000	0.000	0.000	0.000	0.470	0.000	0.054
Kdd	0.450	0.460	0.510	0.360	0.450	0.530	0.480	0.540	0.310	0.480	0.457
Dutch	0.780	0.670	0.830	0.840	0.650	0.720	0.640	0.710	0.680	0.760	0.728
Credit	0.710	0.810	0.860	0.710	0.820	0.790	0.760	0.830	0.800	0.871	0.796
Crime	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
German	0.870	0.810	0.860	0.850	0.890	0.820	0.800	0.810	0.790	0.890	0.839

Table 4: IDI ratios across 10 runs for each dataset using Genetic Algorithm. The final column shows the average IDI ratio **IDI**$_{AVG}$.

Table 5 gives the average IDI ratios and the p-values from the statistical tests comparing GA and RS. GA significantly outperformed RS on six of the eight datasets with extremely small p-values (less than 0.05). The only exceptions were the Crime dataset, where neither method detected discriminatory instances, and the Law School dataset, where improvements were not statistically significant.

7. SciPy, *Statistical Functions Documentation*, https://docs.scipy.org/doc/scipy/reference/stats.html.

Dataset	RS IDI$_{AVG}$	GA IDI$_{AVG}$	p-value
Adult	0.423	0.891	4.99×10^{-10}
Compas	0.065	0.795	4.01×10^{-12}
Law School	0.000	0.054	2.85×10^{-1}
Kdd	0.033	0.457	1.30×10^{-8}
Dutch	0.020	0.728	1.57×10^{-10}
Credit	0.306	0.796	8.76×10^{-10}
Crime	0.000	0.000	–
German	0.082	0.839	2.76×10^{-13}

Table 5: Average IDI ratios for Random Search (RS) and Genetic Algorithm (GA) with p-values from statistical test.

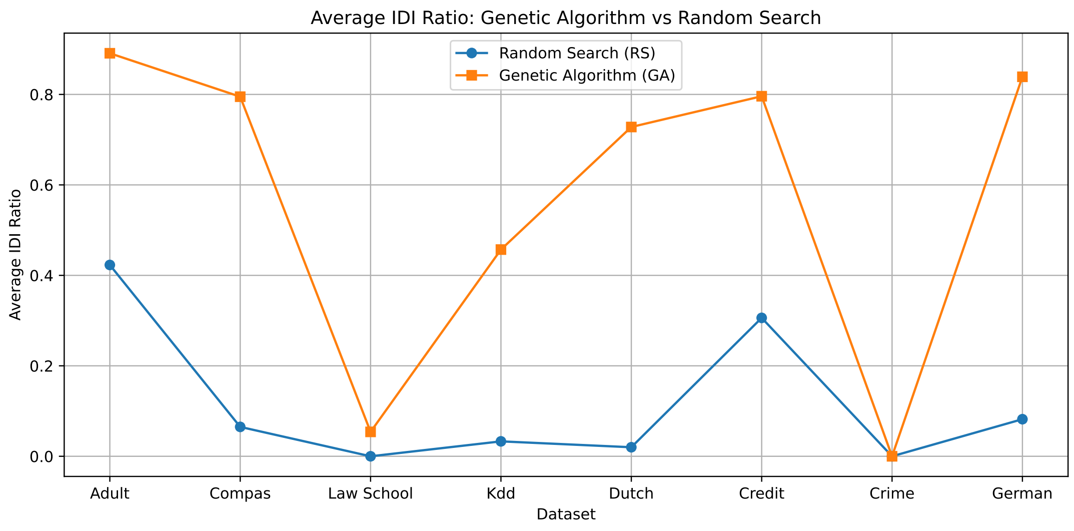

Figure 1: Average IDI Ratios comparing Genetic Algorithm (GA) and Random Search (RS).

Figure 1 illustrates the average IDI ratios for Random Search and Genetic Algorithm across the datasets. The GA outperformed Random Search with particularly significant improvements on datasets like Compas and German.

6 Reflection

The Genetic Algorithm (GA) fairness testing tool presented in this study effectively outperformed the baseline (Random Search) although it has some limitations. One of the main limitations is computational expense. The fitness evaluation requires multiple model predictions which becomes time and resource consuming as the population size and number of generations increase.

The effectiveness of the GA relies on manual parameter tuning which is another problem in itself. Parameters such as mutation rate, crossover probability, and population size strongly impact the performance of the GA. Poor choice of these parameters can cause fairness bugs to be missed as the algorithm could get stuck in local optima. Finding the balance between exploration (mutation) and exploitation (crossover and selection) is important for having an effective and efficient algorithm.

Moreover, this approach focuses only on detecting instances where identical inputs differing only in sensitive features lead to different predictions. While this is useful for identifying certain types of bias, it does not address broader group fairness or demographic parity. As a result, models might still exhibit some undetectable bias. Possible improvements and future work include:

- Implement dynamic parameter tuning to adjust mutation or selection rates based on search progress.
- Add parallel processing to speed up fitness evaluations so the approach is scalable to large datasets or complex models.
- Extend the algorithm to cover group fairness criteria to ensure that the fairness testing is comprehensive.

7 Conclusion

In this project, we developed and evaluated a Genetic Algorithm (GA)-based fairness testing tool to detect individual discrimination in AI models. The tool was designed to beat the Random Search (RS) baseline by more efficiently generating input test pairs that expose fairness bugs.

We conducted experiments across eight real-world datasets that covered a range of sensitive decision-making areas and measured performance using the Individual Discriminatory Instance (IDI) ratio. We then performed statistical hypothesis testing on the results we obtained. We used the Shapiro-Wilk test for normality to determine whether paired t-tests or Wilcoxon signed-rank tests were to be applied depending on if the data followed a normal distribution. Using this we determined that the GA-based approach significantly outperformed the RS baseline on six out of eight datasets. It achieved extremely low p-values such as 4.99×10^{-10} for the ADULT dataset and 4.01×10^{-12} for COMPAS. Although in the case of the CRIME dataset, both methods found no bias which may indicate an inherently fair model or demonstrate a limitation in our current testing approach.

Based on these results, we conclude that the Genetic Algorithm is a highly effective method for fairness testing that is a significant improvement over Random Search in detecting individual discriminatory instances. However, there are limitations to this approach. The GA relies on manual parameter tuning and can be computationally expensive due to repeated model evaluations. Additionally, our method focuses solely on individual fairness and does not address group fairness or broader fairness metrics.

In summary, the Genetic Algorithm approach provides a more efficient and effective method for fairness testing compared to Random Search when detecting individual discrimination in AI models.

8 Artifact

The source code, raw results data, and documentation for this project are available on GitHub at the following repository:

https://github.com/danjw03/ISE

References

Kusner, Matt J, Joshua R Loftus, Chris Russell, and Ricardo Silva. "Counterfactual Fairness" [in eng], 2017.

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)*, 94–101. 2022. https://doi.org/10.1109/ICSTW55395.2022.00030.

SciPy. *Statistical Functions Documentation.* https://docs.scipy.org/doc/scipy/reference/stats.html.

Stoyanovich, Julia, Bill Howe, and H. V. Jagadish. "Responsible data management" [in eng]. *Proceedings of the VLDB Endowment* 13, no. 12 (2020): 3474–3488. ISSN: 2150-8097.

Thede, Scott. "An introduction to genetic algorithms." *Journal of Computing Sciences in Colleges* 20 (October 2004).

Udeshi, Sakshi, Pryanshu Arora, and Sudipta Chattopadhyay. "Automated directed fairness testing." In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 98–108. ASE '18. Montpellier, France: Association for Computing Machinery, 2018. ISBN: 9781450359375. https://doi.org/10.1145/3238147.3238165. https://doi.org/10.1145/3238147.3238165.

Zhang, Peixin, Jingyi Wang, Jun Sun, Xinyu Wang, Guoliang Dong, Xingen Wang, Ting Dai, and Jin Song Dong. "Automatic Fairness Testing of Neural Classifiers Through Adversarial Sampling" [in eng]. *IEEE transactions on software engineering* (New York) 48, no. 9 (2022): 3593–3612. ISSN: 0098-5589.