humancompatible.detect: a Python Toolkit for Detecting Bias in AI Models

Germán M. Matilla[a], Jiří Němeček[a], Illia Kryvoviaz[a], Jakub Mareček[a]

[a]*Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, {martige1, nemecj38, kryvoill, jakub.marecek}@fel.cvut.cz*

arXiv:2509.24340v2 [cs.AI] 19 Jan 2026

Abstract

There is a strong recent emphasis on trustworthy AI. In particular, international regulations, such as the AI Act, demand that AI practitioners measure data quality on the input and estimate bias on the output of high-risk AI systems. However, there are many challenges involved, including scalability (MMD) and computability (Wasserstein-1) issues of traditional methods for estimating distances on measure spaces.
Here, we present `humancompatible.detect`*, a toolkit for bias detection that addresses these challenges. It incorporates two newly developed methods to detect and evaluate bias: maximum subgroup discrepancy (MSD) and subsampled ℓ_∞ distances. It has an easy-to-use API documented with multiple examples.* `humancompatible.detect` *is licensed under the Apache License, Version 2.0.*

Keywords: artificial intelligence, fairness, bias detection, toolkit
2020 MSC: 68T01

1. Motivation and significance

With the increase in practical applications using machine-learning models and AI systems, new ethical problems have appeared. One particular problem is the proliferation (or even strengthening) of societal bias. Infamous examples of AI bias include hiring algorithms [1] or recidivism predictors [2].
In response, more and more regulatory frameworks [e.g., 3] and related standards [e.g., 4, 5] require testing for bias, without any clear agreement on the precise definition of said bias. Indeed, there are many bias measures [6].
The lack of a single definition of bias is understandable, considering the wide range of applications of AI systems. Non-exhaustively:

- Fairness in (large) language and/or vision models, the currently dominant AI models [e.g., 7, 8].
- Fairness in rankings: the main related case being job recruitment. See, for instance, [9, 10, 11].
- Fairness in recommender systems: such as the ones widely used currently by, e.g., YouTube or Spotify [9, 12].
- Spatial aspects of fairness: In contrast to continuous attributes, such as age and income, the case of discrimination based on location demands special attention [13, 14].
- Fairness for forecasting: Consider the well-known COMPAS dataset. It is traditionally treated as a classification task. A forecasting-based approach, however, obtains the best results currently reported on the dataset [15].
- Fairness in sharing-economy business models: such as the model of companies such as Uber and Cabify [16].

Even when agreeing on a particular definition of bias, several difficulties remain:

Sample Complexity and Computability. To estimate data quality, i.e., the preexisting bias in the data per se, one generally calculates a distance between the distribution of values in the training data and the general population. There are many such distances on measure spaces: total variation, MMD [17], and Wasserstein [18], to name the most popular. However, the sample complexity of these distances varies; that is, the number of samples required to achieve an accurate approximation of the selected distance is very large. Moreover, some of them (esp. Wasserstein-1) are believed to be inapproximable (cf. [19, 20]).

Choice of Protected Attributes and Intersectional Aspects. The sample complexity is also clearly tied to the number of protected subgroups. In the case of studying the data quality, one may need to test the distance between the distribution of values in the training data and the general population for each protected subgroup. The number of protected subgroups may thus enter linearly in the overall sample complexity, while being exponential in the number of protected attributes.
For example, consider The California Consumer Privacy Act, which lists a number of possible protected attributes

Nr.	Code metadata description	Metadata
C1	Current code version	0.1.4
C2	Permanent link to code/repository used for this code version	`https://github.com/humancompatible/detect/tree/6b3fb21a155d012cb244c9b5b7ead1f8a978038f`
C3	Permanent link to Reproducible Capsule	none
C4	Legal Code License	Apache-2.0
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python, pandas, Pyomo, HiGHS, SciPy
C7	Compilation requirements, operating environments & dependencies	python $\geq$ 3.10, pandas $\geq$ 2.2 pyomo $\geq$ 6.9, highspy $\geq$ 1.10, scipy $\geq$ 1.16
C8	If available Link to developer documentation/manual	`https://humancompatible-detect.readthedocs.io/en/latest/`
C9	Support email for questions	martige1@fel.cvut.cz

Table 1: Code metadata (mandatory)

Framework	Detection	Intersections	Low sample complexity	Optimal	Ref
Themis-ml	✗	✗	-	-	[21]
FAT Forensics	✓	✗	-	-	[22]
Fairlearn	✗	✓	-	-	[23]
AI Fairness 360	✓	✓	?	✗	[24]
FairTest	✓	✓	✓	✗	[25]
`humancompatible.detect`	✓	✓	✓	✓	here

Table 2: Comparison to common Python fairness frameworks in four dimensions. Whether they allow for detecting bias (not just measuring bias of a predefined group), if they allow for considering group intersections natively, whether some of those detection methods have low sample complexity, and whether the methods have (optimality) guarantees. When no detection method is available, we omit the last two dimensions. The most similar, FairTest, focuses on identifying protected subgroups using a tree structure that aligns with the predictor's output. Importantly, none of the other frameworks consider the `MSD` distance or the subsampled ℓ_∞ norm.

and their values to be protected. The number of protected subgroups can go up to 5,772,124,799 on US Census data [26]. Estimating the bias for each such protected subgroup is clearly challenging.

Other fairness toolkits. There is a variety of existing toolkits, of which only some allow for bias detection. Overall are mostly concerned with either training fair models or fairness evaluation. We provide an overview of various toolkits in Table 2. There are additional toolkits focused on visualiation (e.g., What-If Tool), however, we ommit those from the comparison since they are visual tools rather than evaluation toolkits.
Themis-ml [21] and Fairlearn [23] do not allow for detecting (most) biased subgroups, and detection part of FAT Forensics [22] does not allow for the study of intersections of protected groups. AI Fairness 360 [24] contains only the MDSS method for the detection of intersectional bias, which uses a non-standard definition of bias and it's sample complexity is unclear. It is also a probabilistic method. FairTest [25] contains methods with low sample complexity, however these only test how aligned are protected attributes to the prediction, evaluating correlations rather than standard bias measures. They form a greedy tree, separating groups into intersectional subgroups, without guarantees on optimality.

2. Software description

We introduce `humancompatible.detect`, an open-source toolkit of methods for bias detection. Designed for ease of use, its simple API and guided examples are accessible to non-expert users. It is available on PyPI[1]. The toolkit allows for reliable detection of intersectional bias and evaluation of group bias even when the number of samples is low. Fig 1 illustrates the use of the toolkit.

2.1. Software architecture

As per Figure 1, the two main toolkit objectives are `detect` and `evaluate`. The first one spots the most biased subgroup in the data, while the second informs us about the bias against a given subgroup within the dataset. The data can be passed in multiple ways.

Input data. There are three modes of input:

- In-memory tables (pandas DataFrame). The dataset is passed in as the input data `X` and the binary target column `y`.

[1] https://pypi.org/project/humancompatible-detect/

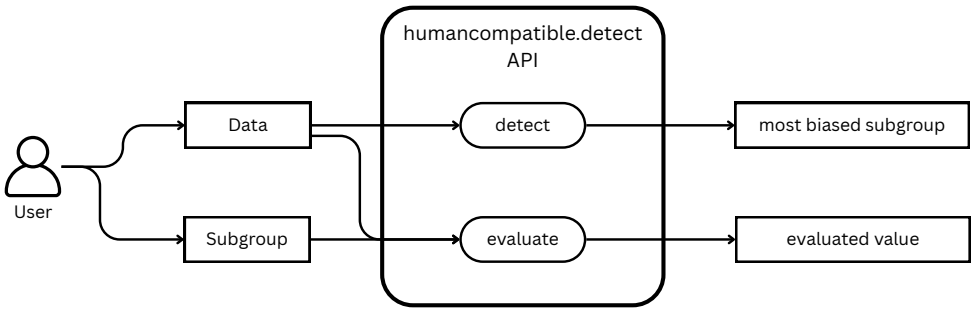


Figure 1: A diagram of possible uses of the `humancompatible.detect` toolkit. In addition to evaluating bias of a protected group, it allows for detection of the most biased subgroup given the dataset.

- CSV file, with a designated target column. Requiring the `path` to this data and the name of the `column`.
- Two-sample comparison, where the goal is to identify subgroups whose representations differs most between two distributions, represented by the datasets. Both datasets must share the same columns. One simply passes two pandas DataFrames `X1` and `X2` as arguments.

Detect bias. To detect bias in data the toolkit alows to use MSD. It can check the exponentially growing number of dataset subgroups and spot the most biased subgroup in the dataset. It returns a simple conjunctive rule representation of a subgroup (e.g.,"AGE between 18-25 AND SEX = Male"). The toolkit can be easily extended to allow for other methods, such as SPSF [27]. See Figure 2 for a graphical description.

Evaluate bias. Independent of learning which subgroup is the most (dis)advantaged within the data, we can quantify the bias for any subgroup. The MSD method measures the subgroup discrepancy of a given subgroup, which is specified as a conjunction of attributes. Additionally, with our second method, subsampled ℓ_∞, we can test whether the bias in a particular subgroup is greater or smaller than a value of our choice. For this task, both methods require additional parameters to be passed as keyword arguments. See Figure 3 for a graphical visualization of the process.
Internally, both tasks use a routine to preprocess the input data, `prepare_dataset`, which is also illustrated in further detail in Fig 4.
For convenience, we provide a unified wrapper for users, `detect_and_score(...)` combining the detection and evaluation into one step. It takes the data (in either of the three formats), the protected attributes and target specification, and returns the corresponding subgroup description together with the computed bias score (or pass/fail decision, depending on the method).

2.2. Software functionalities

We propose a toolkit that implements two approaches to bias estimation. With these, users can (i) discover the most impacted subgroup using Maximum Subgroup Discrepancy, and (ii) check a concrete subgroup against a tolerance using the subsampled ℓ_∞ test.

Maximum Subgroup Discrepancy. When detecting bias, it is crucial to look at all intersections of protected groups, because marginal fairness might be achieved despite exhibiting severe bias when looking at the intersections. As a type of curse of dimensionality, the number of group intersections (subgroups) grows exponentially with the number of protected attributes. This complicates the evaluation of intersectional bias in multiple ways. One has to face the exponential time complexity of testing each subgroup, while having exponentially fewer and fewer samples. A useful approach to this would be to consider the bias as a distance between two distributions, but as mentioned, most distributional distances have exponential sample complexity.
Maximum Subgroup Discrepancy (MSD) [26] addresses the issues of high sample complexity of intersectional fairness. It is a distributional distance, defined as

$$\mathrm{MSD}(\mu, \nu) = \sup_{S \in \mathcal{S}} |\mu(S) - \nu(S)|,$$

where μ and ν are two distributions, $\mathcal{S}$ is the set of all subgroups, and $\mu(S)$ is the probability of sampling a point from the subgroup S given the distribution μ. In simpler terms, MSD looks for the subgroup with the highest difference in probability between two distributions. The MSD algorithm has linear sample complexity in the number of protected attributes, returns naturally defined subgroups (as conjunctions of feature value pairs), and the group that is guaranteed to have the highest discrepancy for practical dataset sizes with many protected attributes. Using mixed-integer optimization, it obtains the provably optimal solution (i.e. the maximally discrepant subgroup).

Subsampled Distances on Measure Spaces. Our second approach addresses the same issue regarding protected group intersections in a different manner. In particular, we subsample a test whether the distance between the two distributions is less than a given value (Δ). For a given subgroup, we can check whether the given subgroup faces a

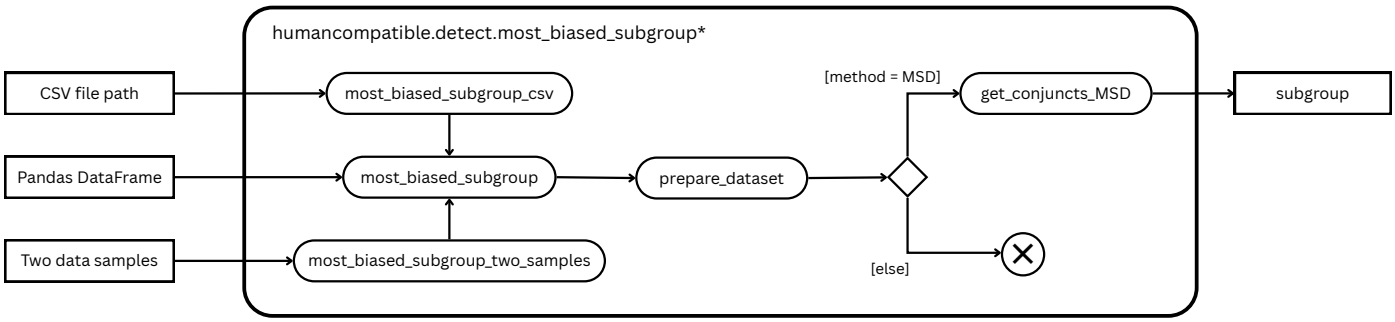


Figure 2: `detect_bias` activity diagram. Depending on the input mode used, different functions can be run, always calling `most_biased_subgroup`. After that, the data preprocessing takes places with `prepare_dataset`. Subsequently, the program stops if a method different from MSD has been selected, otherwise outputs the most affected subgroup as a result. This allows for easy extensibility. More details on the `prepare_dataset` are in Figure 4.

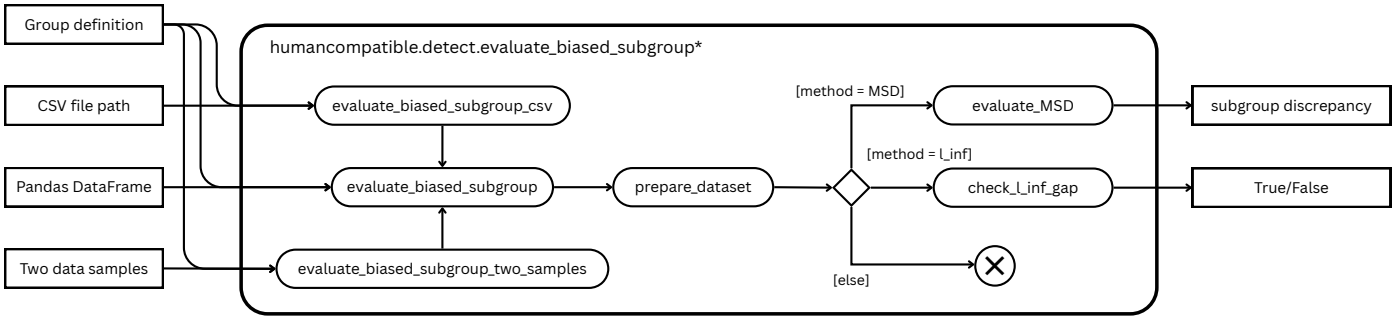


Figure 3: `evaluate_bias` activity diagram. Both input and preprocessing are identical to `detect_bias`, with the addition of keyword arguments specifying the (sub)group (Group definition) and potentially other arguments (e.g., the Δ threshold). The output can either be a value (e.g., subgroup discrepancy) or a boolean value whether bias is above a given threshold (e.g., in subsampled ℓ_∞). More details on the `prepare_dataset` are in Figure 4.

bias within the threshold Δ. In large datasets, we may have to subsample this test, but [28] have developed guaranties on the size of the error with a certain probability. Such guaranties are sometimes known as probably approximately correct (PAC) learning.

3. Illustrative example

We illustrate the toolkit on a synthetic example dataset. There are three columns: *Race* (categorical: Green, Blue, Purple), *Age* (categorical buckets: 0-18, 18-30, 30-45, 45-60), and a binary *Target* (e.g., approval). The dataset is built so that marginal fairness holds, but one intersectional subgroup is disadvantaged, as shown in Figure 5.
We treat *Race* and *Age* as protected attributes, and use MSD to find the subgroup.

```
from humancompatible.detect import detect_and_score

rule, msd_val = detect_and_score(
    csv_path = "01_data.csv",
    target_col = "Target",
    protected_list = ["Race", "Age"],
    method = "MSD",
)
```

The toolkit returns a rule as follows (after interpreting the rule):

```
print("The group is: " +
      " AND ".join(str(r) for _, r in rule))
print("MSD value is:", msd_val)
```

```
The group is:  Race = Blue AND Age = 0-18
MSD value is:  0.111111
```

After evaluating the bias for the obtained subgroup, we got the MSD value of 0.111 . . . (e.g., an 11.1% gap between the positive-outcome and the negative outcome for the given subgroup). MSD confirms that *young Blue* individuals are the most disadvantaged group, even though the data looks fair on the marginals. This is why checking intersections matters.
To test the subsampled ℓ_∞ norm, we add a new protected attribute *Gender* (categorical: M, F) the previous dataset, while keeping the original demographic parity properties. We consider the concrete subgroup *Gender* $= F$, and test whether its distribution of positive outcomes differs from the full population by at most a tolerance Δ in the ℓ_∞ norm.
For a threshold $\Delta = 0.125$, the method reports that the most impacted subgroup bias is *greater* than 0.125 (fail). In other words, the maximum difference between the subgroup and the overall positive outcome histograms is over the threshold.

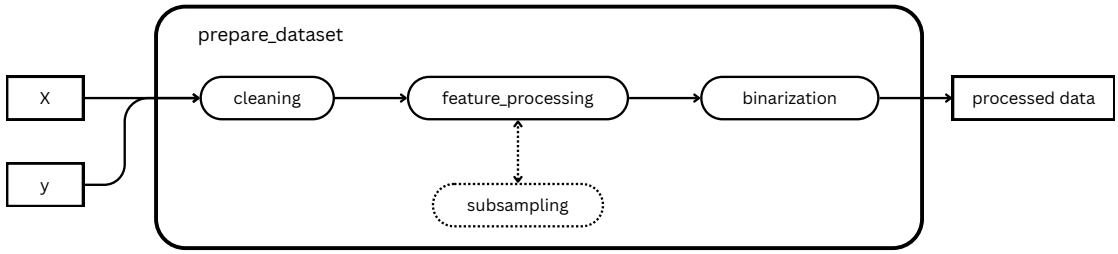


Figure 4: `prepare_dataset` activity diagram. This function preprocesses the input data in the form of a pandas DataFrame, `y`, being the target column and `X`, the rest of the data. It prunes incomplete rows, allows for binarization, and also for specialized feature processing, such as mapping categorical variables into variables with fewer distinct values or custom binning of continuous variables. It also allows for subsampling the data, if passed data is too large. This can speed up the processing without reducing accuracy too much, since implemented methods have low sample complexity.

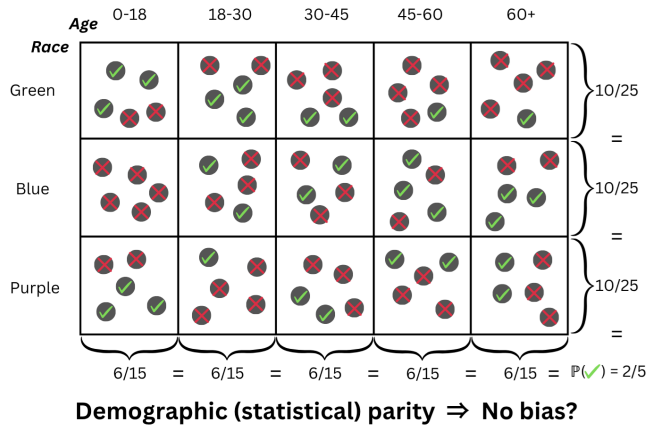


Figure 5: An illustration of why evaluating marginal bias does not suffice. A classifier accepts the same proportion of each age and race group, but rejects the entire subgroup (group intersection) of young blue people. This bias would go undetected unless group intersections are considered. The graphic is a modification of Figure 1 in [26].

4. Impact

Even in the best-studied case of fairness of classification, most of the proposed bias measures, such as statistical parity or equality of opportunity, focus on protected groups in isolation. This approach often misses the intersections of groups or requires one to sift through an exponential number of group intersections (subgroups). Considering all subgroups is crucial, as illustrated by Figure 5, where data is marginally fair, but intersectionally biased.

Another view of bias evaluation in cases when many subgroups need to be considered is bias detection. That is, detecting where bias exists in the data, rather than measuring it entirely over all protected features. Detecting such a biased subgroup allows focusing on the particular subgroup [e.g., 29]. This way, one can address the most significant biases individually through mitigation techniques tailored to the particular group, instead of treating all biases the same.

We showed that `humancompatible.detect` addresses both questions (intersectional fairness and bias detection) by allowing its users to go from tabular data to a concrete, human-readable finding in a few lines. Moreover, its two implemented methods, have an advantage over previously considered approaches such as those based on MMD [17] and Wasserstein [18] distances, in terms of the number of samples required. For MSD, the number of samples is linear in the number of protected attributes, and the sample complexity can be controlled by controlling the error (or uncertainty) in the estimate.

5. Conclusions

We have presented a toolkit for bias detection. Crucially, it can consider protected group intersections and provide bias evaluations with guarantees even for smaller datasets. It has an intuitive API and is designed for easy use by practitioners without requiring domain expertise.

CRediT author statement

Germán M. Matilla: Software, Visualization, Validation, Writing - Original Draft, **Jiří Němeček:** Methodology, Software, Writing - Original Draft, Visualization, Writing - Review & Editing, **Illia Kryvoviaz:** Software, Validation, Visualization, Writing - Original Draft, **Jakub Mareček:** Conceptualization, Supervision, Writing - Original Draft, Writing - Review & Editing.

References

[1] J. Dastin, Insight - Amazon scraps secret AI recruiting tool that showed bias against women, Reuters (Oct. 2018).
URL `https://www.reuters.com/article/world/idUSKCN1MK0AG/`

[2] Julia Angwin, Jeff Larson, Lauren Kirchner, Surya Mattu, Machine Bias (May 2016).
URL `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencin`

[3] Regulation (EU) 2024/1689, Regulation (EU) 2024/1689 of the European Parliament and of the

Council. of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act), place: OJ L, 2024/1689, 12.7.2024 (2024).

[4] IEEE Standards Association, IEEE standard for algorithmic bias considerations, IEEE Std 7003-2024 (2025) 1–59doi:10.1109/IEEESTD.2025.10851955.

[5] R. Schwartz, A. Vassilev, K. K. Greene, L. Perine, A. Burt, P. Hall, Towards a Standard for Identifying and Managing Bias in Artificial Intelligence (Mar. 2022). doi:https://doi.org/10.6028/NIST.SP.1270.
URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934464

[6] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A Survey on Bias and Fairness in Machine Learning, ACM Comput. Surv. 54 (6) (2021) 115:1–115:35. doi:10.1145/3457607.
URL https://dl.acm.org/doi/10.1145/3457607

[7] O. Ahia, S. Kumar, H. Gonen, V. Hofmann, T. Limisiewicz, Y. Tsvetkov, N. A. Smith, MAGNET: Improving the Multilingual Fairness of Language Models with Adaptive Gradient-Based Tokenization, Advances in Neural Information Processing Systems 37 (2024) 47790–47814. doi:10.52202/079017-1514.
URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5572bc595de865c1450868fd5391e9c5-Abstract-Conference.html

[8] J. Castleman, A. Korolova, Adultification Bias in LLMs and Text-to-Image Models, in: Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency, FAccT '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 2751–2767. doi:10.1145/3715275.3732178.
URL https://doi.org/10.1145/3715275.3732178

[9] E. Pitoura, K. Stefanidis, G. Koutrika, Fairness in rankings and recommenders: Models, methods and research directions, in: 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2021, pp. 2358–2361. doi:10.1109/ICDE51399.2021.00265.

[10] A. Kliachkin, E. Psaroudaki, J. Marecek, D. Fotakis, Fairness in ranking: Robustness through randomization without the protected attribute, in: 40th International Conference on Data Engineering, ICDE 2024 - Workshops, Utrecht, Netherlands, May 13-16, 2024, IEEE, 2024, pp. 201–208. doi:10.1109/ICDEW61823.2024.00032.
URL https://doi.org/10.1109/ICDEW61823.2024.00032

[11] J. Fageot, P. Blanchard, G. Bareilles, L.-N. Hoang, Generalizing while preserving monotonicity in comparison-based preference learning models, in: Thirty-ninth Conference on Neural Information Processing Systems, 2025, arXiv preprint arXiv:2506.08616.

[12] D. Liu, J. Baek, T. Eliassi-Rad, When Collaborative Filtering is not Collaborative: Unfairness of PCA for Recommendations, in: Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency, FAccT '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 2974–2990. doi:10.1145/3715275.3732190.
URL https://doi.org/10.1145/3715275.3732190

[13] D. Sacharidis, G. Giannopoulos, G. Papastefanatos, K. Stefanidis, Auditing for spatial fairness (2023). arXiv:2302.12333.
URL https://arxiv.org/abs/2302.12333

[14] D. Kyriakopoulos, D. Sacharidis, G. Giannopoulos, D. Gunopulos, T. Dalamagas, Promis: A post-processing framework for mitigating spatial bias, in: Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems, 2025.

[15] Q. Zhou, J. Marecek, R. Shorten, Fairness in forecasting of observations of linear dynamical systems, Journal of Artificial Intelligence Research 76 (2023) 1247–1280. doi:10.1613/jair.1.14050.

[16] Q. Zhou, R. Ghosh, R. Shorten, J. Mareček, Closed-loop view of the regulation of ai: Equal impact across repeated interactions, in: 2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW), IEEE, 2024, p. 176–181. doi:10.1109/icdew61823.2024.00029.
URL http://dx.doi.org/10.1109/ICDEW61823.2024.00029

[17] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, The Journal of Machine Learning Research 13 (1) (2012) 723–773.

[18] L. N. Vaserstein, Markov processes over denumerable products of spaces, describing large systems of automata, Problemy Peredachi Informatsii 5 (3) (1969) 64–72.

[19] Y. Lee, H. Boche, G. Kutyniok, Computability of optimizers, IEEE Transactions on Information Theory 70 (4) (2023) 2967–2983.

[20] Y. Lee, H. Boche, G. Kutyniok, Computability of optimizers for ai and data science, in: Handbook of Numerical Analysis, Vol. 25, Elsevier, 2024, pp. 359–388.

[21] N. Bantilan, Themis-ml: A fairness-aware machine learning interface for end-to-end discrimination discovery and mitigation (2017). arXiv:1710.06921.
URL https://arxiv.org/abs/1710.06921

[22] K. Sokol, R. Santos-Rodriguez, P. Flach, FAT Forensics: A Python toolbox for algorithmic fairness, accountability and transparency, Software Impacts (2022) 100406.

[23] H. Weerts, M. Dudík, R. Edgar, A. Jalali, R. Lutz, M. Madaio, Fairlearn: Assessing and Improving Fairness of AI Systems, Journal of Machine Learning Research 24 (2023).
URL http://jmlr.org/papers/v24/23-0389.html

[24] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, Y. Zhang, AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias (Oct. 2018).
URL https://arxiv.org/abs/1810.01943

[25] F. Tramer, V. Atlidakis, R. Geambasu, D. Hsu, J.-P. Hubaux, M. Humbert, A. Juels, H. Lin, Fairtest: Discovering unwarranted associations in data-driven applications, arXiv preprint arXiv:1510.02377 (2015).

[26] J. Němeček, M. Kozdoba, I. Kryvoviaz, T. Pevný, J. Mareček, Bias Detection via Maximum Subgroup Discrepancy, in: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, ACM, Toronto ON Canada, 2025, pp. 2174–2185. doi:10.1145/3711896.3736857.
URL https://dl.acm.org/doi/10.1145/3711896.3736857

[27] M. Kearns, S. Neel, A. Roth, Z. S. Wu, Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 2564–2572, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v80/kearns18a.html

[28] G. M. Matilla, J. Mareček, Sample complexity of bias detection with subsampled point-to-subspace distances (2025). arXiv:2502.02623.
URL https://arxiv.org/abs/2502.02623v1

[29] A. Sharma, K. Rushton, I. W. Lin, T. Nguyen, T. Althoff, Facilitating Self-Guided Mental Health Interventions Through Human-Language Model Interaction: A Case Study of Cognitive Restructuring, in: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, CHI '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–29. doi:10.1145/3613904.3642761.
URL https://dl.acm.org/doi/10.1145/3613904.3642761