Towards Model-based Bias Mitigation in Machine Learning

Alfa Yohannis
University of York
York, United Kingdom
alfa.yohannis@york.ac.uk

Dimitris Kolovos
University of York
York, United Kingdom
dimitris.kolovos@york.ac.uk

ABSTRACT

Models produced by machine learning are not guaranteed to be free from bias, particularly when trained and tested with data produced in discriminatory environments. The bias can be unethical, mainly when the data contains sensitive attributes, such as sex, race, age, etc. Some approaches have contributed to mitigating such biases by providing bias metrics and mitigation algorithms. The challenge is users have to implement their code in general/statistical programming languages, which can be demanding for users with little programming and fairness in machine learning experience. We present FairML, a model-based approach to facilitate bias measurement and mitigation with reduced software development effort. Our evaluation shows that FairML requires fewer lines of code to produce comparable measurement values to the ones produced by the baseline code.

CCS CONCEPTS

• **Theory of computation** → *Models of learning*; • **Applied computing** → **Law, social and behavioral sciences**; • **Software and its engineering** → **Source code generation**; **Domain specific languages**.

KEYWORDS

Model-Driven Engineering, Generative Programming, Bias Mitigation, Bias Metrics, Machine Learning

ACM Reference Format:
Alfa Yohannis and Dimitris Kolovos. 2022. Towards Model-based Bias Mitigation in Machine Learning. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3550355.3552401

1 INTRODUCTION

The use of machine learning is pervasive nowadays, from personal daily activities, such as face recognition for authentication and voice processing when talking to intelligent assistants (e.g., Alexa, Siri), to performing sensitive, ethical tasks, such as predicting criminal behaviours and classifying profiles for loans. While machine learning does bring efficiency, models produced by machine learning are not guaranteed to be free from bias, particularly when trained and tested with data produced in discriminatory environments. This can be unacceptable, mainly when it touches sensitive attributes, such as sex, race, age, etc., and amplifies unfairness.

In 2016, COMPASS, an algorithm for predicting recidivism, was found to produce a high false-negative rate for white people and a high false-positive rate for black people [5]. Some commercial face recognition services were also found to have significantly lower accuracy on darker-skin females [11]. Moreover, A jobs platform was found to rank qualified female candidates far lower than qualified male candidates even though they have similar properties [33]. These are some instances that show how biases in machine learning can promote unfairness.

Some approaches have contributed to the mitigation of such biases by providing bias metrics and debiasing algorithms (more in Sections 2.2 and 2.3). Different toolkits have implemented these metrics and algorithms. However, they come with different methodological approaches and capabilities, which demand users to understand them in depth before they can decide which toolkits are best for specific scenarios [34].

Data scientists usually work using their intuitions to narrow down the number of combinations of algorithms, parameters, and other factors to find the best models for given goals, datasets, and domains [37]. After that, with lots of experimentation and trial and error [12], they have to go through all the narrowed combinations and test the produced models to identify which models are the best. Moreover, regardless of the availability of machine learning libraries, data scientists have to craft the search process from scratch in general/statistical programming languages (e.g., Python, R).

Model-Driven Software Development (MDSE) leverages the abstraction of software development by hiding technical details of implementations that can be automated [10]. Thus, users can focus on essential aspects through the use of simpler modelling languages, and the target implementation can be automatically generated, which in turn improves productivity [47]. The use of MDSE would benefit data scientists since it allows them to search for the best bias mitigation methods for given cases at a higher abstraction level. They also do not have to code the implementation of the search process in general/statistical programming languages since it can be automatically generated and then fine-tuned later on.

In this paper, we introduce FairML[1], a tool that implements the MDSE approach to model and automate bias measurement and mitigation in machine learning.

MODELS '22, October 23–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9466-6/22/10…$15.00
https://doi.org/10.1145/3550355.3552401

[1]FairML prototype can be found at https://doi.org/10.5281/zenodo.7007839 and https://github.com/York-and-Maastricht-Data-Science-Group/fairml

(1) FairML raises the abstraction of bias measurement and mitigation so that users can configure their bias mitigation model in YAML (YAML Ain't Markup Language), a human-friendly declarative language [19], without having to code in general/statistical programming languages.
(2) FairML supports a certain degree of expressiveness, allowing users to experiment with different kinds of bias metrics, bias mitigation algorithms, datasets, classifiers, and their parameters to find the best combinations that reduce biases but with acceptable accuracy.
(3) Supporting tooling automatically generates Python and Jupyter Notebook files which users can execute to measure and mitigate biases on given datasets. All generated files are modifiable and extensible for fine-tuning and further development.

This paper is structured as follows. First, we discuss fairness in machine learning in Section 2. The section covers the definitions of some related terms, real examples, bias metrics, and bias mitigation. In Section 3, we present an overview of our development as well as our approach to implementing debiasing techniques in the machine learning field to automate bias mitigation. The section also presents how users use an existing toolkit in expressing a solution to mitigate bias in classification. We also demonstrate how FairML represents the solution in a more readable and concise way and can automatically produce bias measurement and mitigation code with similar correctness as the manually-crafted solutions. In section 4, we evaluate FairML on expressiveness, correctness, generation time, and execution time using the existing code of a bias mitigation toolkit as the benchmark. Section 4.1 presents and discusses the result of the evaluation. We also reflect on some lessons we learned during the research and development of FairML in Section 5. Section 6 discusses related work, and Section 7 presents the conclusions and future work of this research.

2 BIAS IN MACHINE LEARNING

This section briefly discusses fairness in machine learning, which covers the definitions of some related terms, real examples, bias metrics, and bias mitigation.

2.1 Definitions and Examples

Fairness is defined as "the absence of any prejudice or favouritism toward an individual or group based on their inherent or acquired characteristics" [36]. The absence of fairness can be caused by a bias which is a systematic error or distortion from the actual state of affairs due to flaws in data collection and processing, study design, analysis, and interpretation [38]. The unfairness happens when biases put privileged groups in an advantaged position against unprivileged groups [7]. Bias can also be caused by harmful discrimination if the distinction made is due to intentional or unintentional stereotyping and prejudice based on sensitive attributes (race, age, sex, etc.) [15, 36].

For example, in 2006, it was found that an algorithm used for recidivism prediction produces a much higher false-positive rate for black people than white people [5]. Another example is a job platform which was found to rank less qualified male candidates higher than more qualified female candidates [33]. The paper states that fairness between female and male groups has been achieved, but group fairness did not necessarily indicate fairness for individuals. Moreover, publicly available commercial face recognition online services provided by renowned vendors are found to suffer from achieving much lower accuracy on females with darker skin colour [11]. There are other examples of biases in machine learning, but these three examples already demonstrate that, in practice, machine learning is not always fair, and the unfairness can bring disadvantages to specific groups.

2.2 Bias Metrics

Some metrics have been developed to detect and measure biases in machine learning. Each metric has its way of calculating biases and therefore is preferable to others in particular situations. IBM AI Fairness 360 [26, 35] and Aequitas[2] have provided guidance for the selection, summarised in Figure 1. So, if a bias measurement requires metrics that encompass group and individual fairness, then Theil Index [7, 16] and Generalised Entropy Index [44] are preferable. They can be used as a unified metric to measure the inequality in benefit allocation for groups and individuals [26, 35]. Lower scores reflect strong fairness, while the higher ones show the opposite. Perfect fairness is indicated by a value of 0 [25].

Euclidean Distance, Manhattan Distance, and Mahalanobis Distance are metrics to measure individual fairness. These metrics are used to measure the distance of the same individual in the original and debiased datasets [7]. They are required to ensure consistency since data transformation, when applying bias mitigation in preprocessing, can achieve group fairness but might cause unfairness to individuals. In [13], these metrics are used as constraints to limit the distortion of its bias mitigation so that both group fairness and individual fairness can be achieved.

There are two main world views for group fairness: equal and proportional fairness [26, 35]. Equal fairness means under-represented groups should have a similar chance as the rest regarding prediction results and is usually applied to systems perceived to have structural discrimination. For example, SAT is suspected to contain structural discrimination [26, 35]. Statistical Parity Difference [18, 26, 35] and Disparate Impact [20, 26, 35] are the common metrics to measure equal fairness.

Statistical Parity Difference is computed as the difference of probabilities of being labelled favourably between the privileged and unprivileged groups (Equation 1) [4, 7, 18].

$$SPD = Pr(\hat{Y} = 1|D = \text{unpriv}) - Pr(\hat{Y} = 1|D = \text{priv}) \quad (1)$$

The ideal value of the metric is 0. A negative value implies that the labelling favours the privileged group, while a positive value implies the opposite [4, 7]. Disparate Impact [4, 7, 20] computes fairness as the ratio of probabilities of being labelled favourably between the unprivileged and privileged groups (Equation 2).

$$DI = \frac{Pr(\hat{Y} = 1|D = \text{unprivileged})}{Pr(\hat{Y} = 1|D = \text{privileged})} \quad (2)$$

The ideal value of this metric is 1.0. A value < 1 indicates that the labelling advantages the privileged group, and a value > 1 disadvantages the unprivileged group [4, 7].

[2] http://aequitas.dssg.io/static/images/metrictree.png

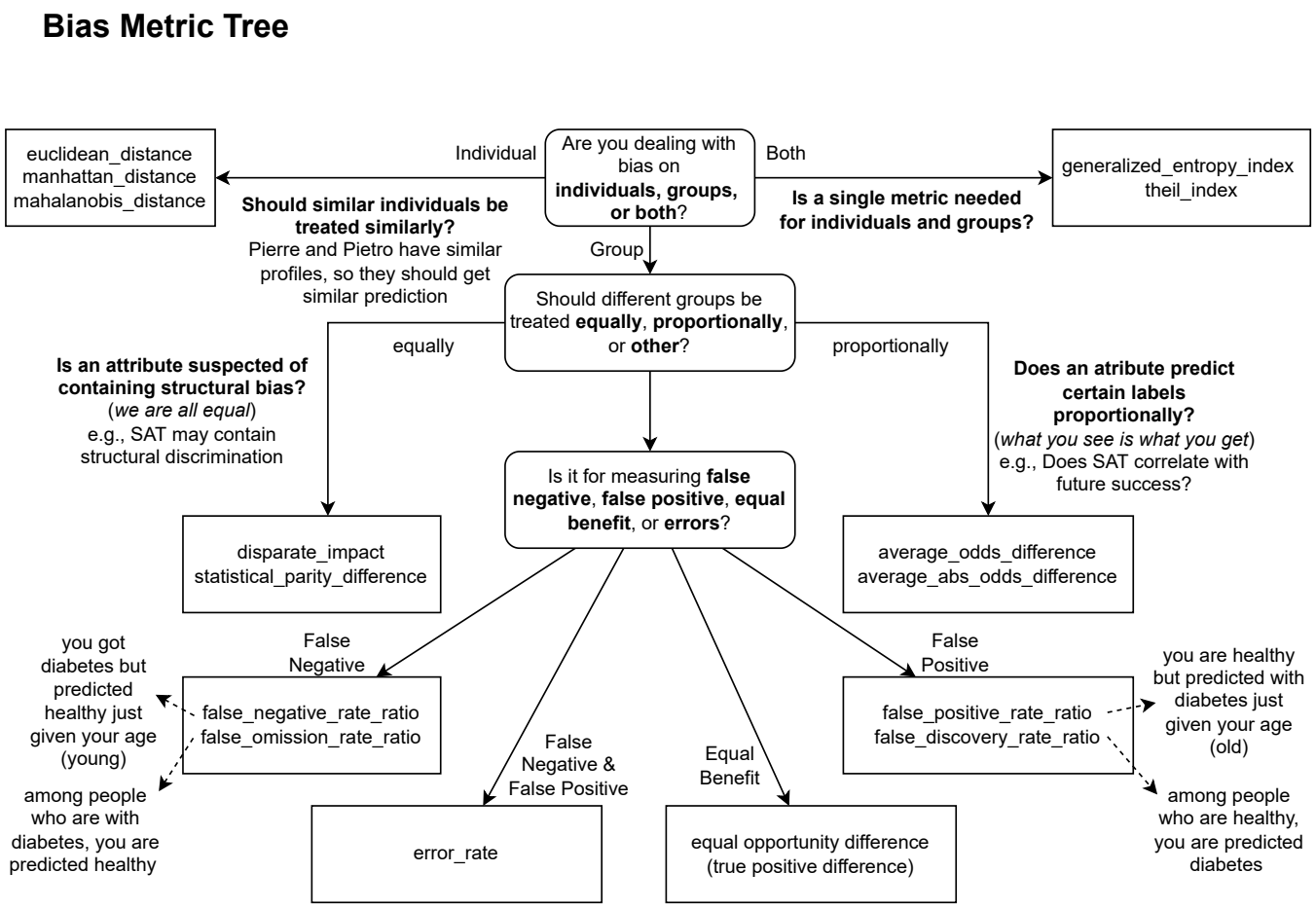


Figure 1: The decision tree to automatically select the most appropriate bias metrics for particular situations.

In contrast, proportional fairness, a.k.a. what you see is what you get (WYSIWYG), perceives that certain features correlate with certain labels, and therefore the features can be used to predict the labels. For example, the scores of SAT correlate with incomes in the future [26, 35]. Average Odds Difference is a metric to measure proportional fairness. The metric computes bias as the average difference of the false-positive rate ($FPR = FP/N$, false positives / all negatives) and true-positive rate ($TPR = TP/P$, true positives / all positives) between unprivileged and privileged groups (Equation 3) [4, 7].

$$AOD = \frac{1}{2} * ((FPR_{D=\text{unprivileged}} - FPR_{D=\text{privileged}}) + (TPR_{D=\text{unprivileged}} - TPR_{D=\text{privileged}})) \quad (3)$$

The ideal value of the metric is 0. A negative value indicates the privileged group is more favourable than the unprivileged group while a positive value indicates the opposite [4, 7].

There are also other metrics that fall in between equal and proportional fairness. For example, Equal Opportunity Difference [4, 7] is a metric that calculates bias as the difference of true positive rates between the unprivileged and the privileged groups (Equation 4).

$$EOD = TPR_{D=\text{unprivileged}} - TPR_{D=\text{privileged}} \quad (4)$$

The true-positive rate is the ratio of true positives to the total number of all positive outcomes ($TPR = TP/P$) for a given group. The ideal value is 0, and a negative value indicates a higher preference for the privileged group while a positive one is the opposite [4, 7]. Other metrics in this space are False Negative Rate Ratio and Difference, False Omission Rate Ratio and Difference, Error Rate, False Positive Rate Ratio and Difference, and False Discovery Rate Ratio and Difference [26, 35].

2.3 Bias Mitigation

Debiasing algorithms have been developed to reduce biases in machine learning, and they can be categorised based on the stages where they are applied in machine learning pipelines: pre-processing, in-processing, and post-processing. The work in [26, 35] provides guidance for choosing the most appropriate debiasing algorithms for specific situations. The guidance is summarised in Figure 2.

Reference [35] recommends mitigating unfairness as early as possible, particularly in the pre-processing stage, since biases that are reduced in that stage are intrinsic in datasets. Unfortunately, there are conditions where altering datasets is prohibited. So, if modifying datasets is not allowed, Reweighing [27] can be an option

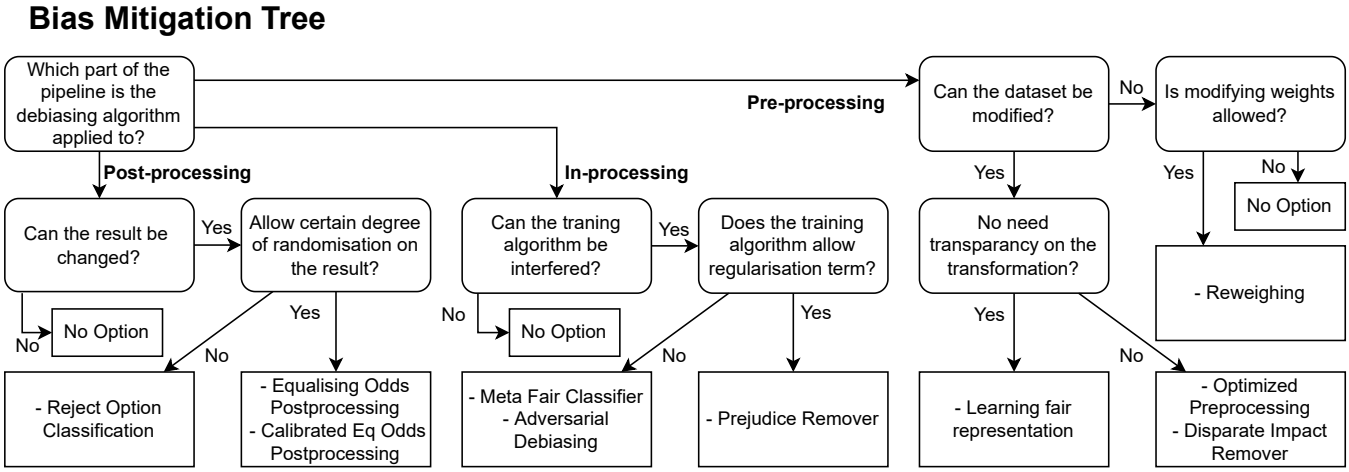


Figure 2: The decision tree to select the most appropriate debiasing algorithms for particular situations.

since it does not alter the values of features but works by altering only the weight of each label/prediction to gain fairness. If modifying datasets is allowed, Optimised Preprocessing [13], Disparate Impact Remover [20], and Learning Fair Representation (LFR) [50] can be suitable options except that LFR does not provide transparency since it encodes data into a latent space, and therefore the first two are preferable when transparency of dataset transformation is required. Optimized Preprocessing learns a probabilistic transformation that modifies the labels and features of a dataset, but with objectives and constraints on individual distortion, data fidelity, and group fairness [13, 35]. Disparate Impact Remover alters the values of features to increase the fairness of groups while preserving rank-ordering within groups [20, 35]. LFR pre-processes a dataset by encoding data into a latent space and obfuscates the values of protected attributes [35, 50].

Bias mitigation can also be applied in in-processing by interfering with training algorithms. The debiasing algorithms applied in processing mainly work by penalising biases, usually using fairness constraints or objectives that limit or regularise biases in classifiers [35]. For instance, Prejudice Remover [29] adds a discrimination-aware regularisation term to the learning objective. However, it is limited only to the learning algorithms that support regularisation terms [26, 35]. Because of the limitation, users can consider other in-processing algorithms that allow for a more general set of learning algorithms, such as Adversarial Debiasing [26, 35]. Adversarial Debiasing [51] learns a classifier to maximise its prediction accuracy while also lowering an adversary's ability to determine the protected attribute from the predictions. This approach makes the predictions unable to carry any group discrimination information that the adversary can exploit and therefore leads to a fair classifier [26]. Other in-processing debiasing algorithms are Meta Fair Classification [14], Gerryfair Classification [30, 31], Exponentiated Gradient Reduction [2], and Grid Search Reduction [2, 3].

Post-processing bias mitigation can only be performed if the result of a prediction can be changed. Reject Option Classification [28] favours outcomes to unprivileged groups and does the opposite to privileged groups in a confidence band around the decision boundary with the highest uncertainty. Equalised Odds Postprocessing [23, 39] and Calibrated Equalised Odds Postprocessing [39] try to find probabilities with which to change output labels to optimise equalised odds objectives. The former achieves it by solving linear programs while the latter optimises over calibrated classifier scores. The two equalised odds algorithms have a randomised component, and both can be preferred because of that characteristic. Otherwise, the reject option algorithm is the alternative option since it is deterministic.

3 MODEL-BASED BIAS MITIGATION

This section presents our approach to delivering model-based bias mitigation in machine learning.

3.1 Toolkit Selection

MDSE relies heavily on code generation to produce the code of target software. The generated code often uses existing tools or software libraries to implement certain functions in the respective domains. Machine learning is no exception; there are existing toolkits designed to reduce ethical biases in machine learning that can be leveraged.

Lee et al. [34] analysed the landscape of open-source toolkits in algorithmic ethics, particularly for fair machine learning. With criteria that the toolkit 1) should be open source, 2) is likely to be used by practitioners, and 3) is implementing the fairness-related methodology, and through a focus group discussion of data scientists, they selected the six best toolkits for fair machine learning and used them further in their analysis. The toolkits are Aequitas[41], Google What-If [48], Scikit-fairnet/scikit-lego [42, 43], Fairlearn[9], and IBM AI Fairness 360 [7].

We evaluated these six toolkits to assess their suitability as infrastructure for our model-based approach. The main criteria for the selection were 1) it should be open source, 2) it should support bias mitigation, 3) it should provide an Application Programming Interface (API) to be programmable and integrated into the model-based approach, and 4) it should support various bias measurements and

mitigation algorithms to enable users to find the best bias mitigation strategies.

All the toolkits meet the first criterion since they are all open-source projects. We excluded Aequitas due to its inability to perform bias mitigation (second criterion); it only supports bias measurement. We also removed Google What-If since it does not have an API[3] to be integrated with other applications (third criterion); its customisation is limited to custom prediction function[4]. IBM AI Fairness 360 has more features than the rest. It claims it offers at least ten state-of-the-art bias mitigation algorithms and 77 bias metrics[5] while Fairlearn and Scikit-fairness/lego come second and third [34]. All of them meet the fourth criterion. However, since IBM AI Fairness 360 has more features than the rest, we selected it as the infrastructure for our model-based solution.

3.2 Constructs and Workflow

To model bias mitigation in machine learning, a Domain-Specific Language should be able to allow users to express the essential constructs and workflows in the domain [47]. The work in [7] has documented the important constructs and typical workflows/pipelines in implementing bias mitigation using the IBM AI Fairness 360 toolkit, and we discuss them briefly here.

Set Up Datasets (T01). Bias mitigation commonly starts with *datasets* setup. In this task, users define the *source* of every dataset which is commonly a *CSV file*. They also determine which attribute is the *predicted attribute*, including the *favourable class* in the attribute, and the *sensitive attributes* as well as their *privileged* and *unprivileged classes*. The datasets can also be loaded from the *built-in datasets* provided by certain toolkits. The *training*, *validation*, and *test datasets* are also set here, whether they are all come from different datasets or the same dataset but is *split* in different ratios. They also select the *bias metrics* and *mitigation algorithms* that will be applied to datasets, *models*, and *predictions*.

Measure Original Biases Prior to Prediction (T02). In this task, users measure the *original biases* of the datasets in different metrics, such as accuracy, mean difference, etc. The results are later used as benchmarks when compared with the measurement results after prediction (T04) or bias mitigation (T06).

Train and Predict (T03). This task *trains models* commonly using certain *classifiers*/*classification algorithms* on the train datasets defined in Task T01. If required, the training can extend to *validation* to tune hyper-parameters in order to obtain the best *parameters* for the models. After that, users use the models to *predict* on the test datasets defined in Task T01. This task can be skipped if a prediction is not required. For example, we want to measure and mitigate biases only in the original datasets (pre-processing).

Measure Original Biases Post Prediction (T04). In this task, users measure the original accuracies and biases after performing the prediction in Task T03. The results are used as benchmarks later when comparing them against biases after bias mitigation. This action is skipped if prediction or Task T03 is not performed.

Mitigate Bias (T05). In this task, users choose the type of debiasing algorithms – pre-processing, in-processing, or post-processing – they want to apply. *Validation* can also be performed if hyper-parameter tuning is required to obtain the best parameters for the debiasing algorithms.

Measure Biases After Debiasing (T06). In this task, users measure accuracies and biases after applying debiasing algorithms in Task T05 using the test datasets.

Conclude (T07). In this task, the biases after debiasing are compared against the original biases obtained in Tasks T02 and T04. Users then analyse the results to find the best combinations of datasets, classifiers, and debiasing algorithms with biases and accuracy that fit their purposes. The comparisons can be presented in numbers, tables, and charts to facilitate users to analyse the effects of the datasets, classifiers, and debiasing algorithms.

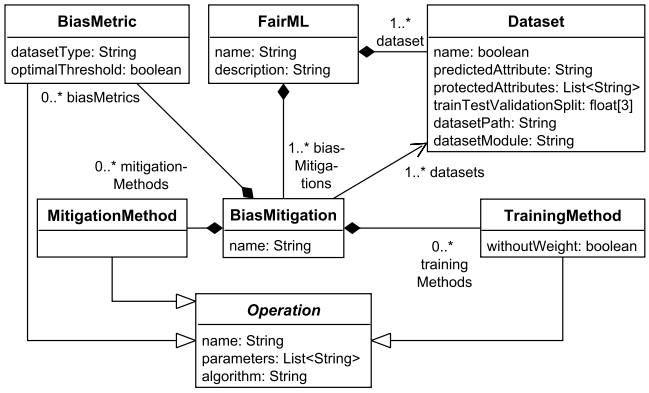


Figure 3: The Metamodel of FairML.

3.3 Metamodel

The constructs identified above have been incorporated into the metamodel of FairML so that users can express them when defining their bias mitigation experiments. Figure 3 shows a simplified version of the metamodel. The `FairML` class represents a project of bias mitigation in machine learning. It can contain one or more datasets and bias mitigations.

The `Dataset` class defines the datasets. In the class, we specify a dataset's name, predicted attribute, protected attributes, the split ratio between train, test, and validation, and the path of its CSV file.

The `BiasMitigation` class specifies the bias mitigations to be executed in the FairML project. The *name* attribute is used as the identifier of bias mitigations. The class composes three other significant classes: `TrainingMethod`, `MitigationMethod`, and `BiasMetric`. All are derived from the `Operation` abstract class since they can be executed, have parameters, and return results. The attributes *name* and *parameters* are used as the identifiers and to define the parameters of operations respectively. The `algorithm` attribute of the `Operation` class defines the selected algorithms executed by each deriving class. The `TrainingMethod` class is responsible to define the classifiers for training and prediction, while the `MitigationMethod` class specifies the debiasing algorithms for bias mitigation. `BiasMetric` class determines the metrics to be used to measure fairness.

[3]https://groups.google.com/g/what-if-tool/c/zw4Rk5kxPIM
[4]https://pair-code.github.io/what-if-tool/get-started/
[5]https://aif360.mybluemix.net/

3.4 FairML Example

Listing 1 shows a FairML model, that conforms to the metamodel in Figure3, expressed in YAML, and is the FairML version of an IBM AI Fairness 360's example demo[6]. The example uses the *adult*[7] dataset for training and testing predictive models. Instead of reducing biases in the dataset (pre-processing bias mitigation), the example applies Exponentiated Gradient Reduction technique to reduce biases in the in-processing stage of the machine learning pipeline. It concludes with a comparison of the accuracies and fairness of the models produced with and without the in-processing bias mitigation. The original example has 93 lines of Python code, excluding commented and empty lines, almost three times longer than Listing 1.

The model has a name and description, `Demo` and `Debiasing using EGR` respectively (lines 3-4). The model has one dataset (line 7) and one bias mitigation (line 15). The dataset is named `Adult` and is loaded from a csv file, `load_preproc_data_adult.csv`. We also set `sex` and `race` as its protected attributes, `Income Binary` as the predicted attribute, and split the original dataset to train, test, and validation datasets with ratios 7:3:0 (lines 7-12).

Lines 15-32 defines the definition of the bias mitigation executed in the `Demo` model. The bias mitigation is named `Exponentiated Gradient Reduction` and use the `Adult` dataset defined at lines 7-12 as its `dataset`.

The bias mitigation uses three different metrics to analyse the trade-off between accuracy and fairness (mean difference, average odds difference) (lines 27-32) when the prediction uses a Logistic Regression classifier only without debiasing (lines 19-21) and when the prediction is debiased using Exponentiated Gradient Reduction (lines 23-25). The former only uses one parameter; `'lbfgs'` is set as the `solver`, while the latter has three parameters; the same classifier is used as the `estimator`, `'EqualizedOdds'` is used as the `constraint`, and the protected attributes are not dropped.

Listing 1: Bias mitigation using Demo Exponentiated Gradient Reduction expressed in YAML.

```
1  ?nsuri: fairml
2  fairml:
3  - name: Demo
4  - description: Debiasing using EGR
5
6  # set the dataset
7  - dataset:
8    - name: Adult
9    - predictedAttribute: Income Binary
10   - protectedAttributes: sex, race
11   - trainTestValidationSplit: 7, 3
12   - datasetPath: load_preproc_data_adult.csv
13
14 # define the bias mitigation
15 - biasMitigation:
16   - name: Exponentiated Gradient Reduction
17   - dataset: Adult
18
19   - trainingMethod:
20     - algorithm: LogisticRegression
21     - parameters: solver='lbfgs'
22
23   - mitigationMethod:
24     - algorithm:
           ExponentiatedGradientReduction
25     - parameters: estimator=LogisticRegression
           (solver='lbfgs'), constraints='
           EqualizedOdds', drop_prot_attr=False
26
27   - biasMetric:
28     - name: accuracy
29   - biasMetric:
30     - name: mean_difference
31   - biasMetric:
32     - name: average_odds_difference
```

[6]https://github.com/Trusted-AI/AIF360/blob/master/examples/demo_exponentiated_gradient_reduction.ipynb.
[7]https://archive.ics.uci.edu/ml/datasets/Adult

3.5 Wizard

While FairML provides the means to specify bias mitigation, users still need to understand all the supported debiasing algorithms and bias metrics in order to use them properly. Being able to help them construct a FairML model using a wizard that asks questions in natural language is important. The wizard follows the decision trees in Figures 1 and 2, and both have been discussed in Sections 2.2 and 2.3 respectively.

After completing the wizard, FairML automatically generates a FairML model n Flexmi (see below). The file is modifiable by users if they want to add other or modify existing debiasing algorithms and bias metrics later on. The wizard also automatically generates the Python and Jupyter notebook files of the produced model.

Listing 2: Some of the questions asked in the FairML's wizard to assist users to select the best debiasing algorithms and bias metrics.

```
1  ...
2  ---- Mitigation Algorithm ----
3  # 1. Pre-processing
4  Apply bias mitigation in pre-processing (
       default: true):
5  The weights of the dataset are modifiable (
       default: true):
6  The bias mitigation allows latent space (
       default: false):
7  ...
8  ...
9  ---- Bias Metric ----
10 Measure group fairness (default: true):
11 Measure individual fairness (default: false):
12 Use single metrics for both individuals and
       groups (default: false):
13 Measure equal fairness (default: false):
14 ...
```

3.6 Generation

Figure 4 depicts the transformation that consumes FairML models and produces Python and Jupyter Notebook files. Users define their

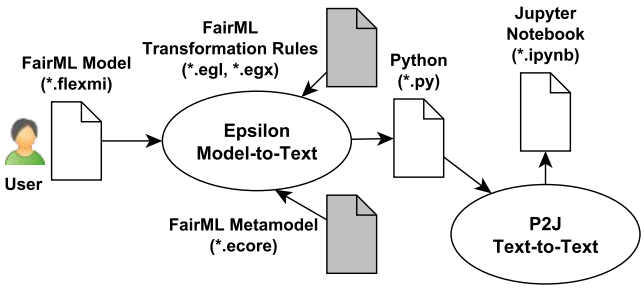


Figure 4: Model-to-text transformation in fairML.

bias mitigation models in Flexmi[8] [32]. Flexmi is a reflective textual syntax for EMF Models [45] that uses fuzzy matching to map tags and attributes in a YAML/XML document to Ecore class/features names [45] in a target metamodel. Besides supporting the default XML-based format, Flexmi also can read models expressed in the YAML-flavoured format. A transformation written in Epsilon Generation Language (EGL) [40] consumes models that conform to the FairML metamodel and produces Python files. Noted that the same transformation could have been implemented using any model-to-text language. The generated Python files are then transformed into Jupyter notebook files using the P2J[9] engine.

3.7 Generated Files

Based on the model defined in Listing 1, FairML generates a Jupyter notebook file containing the code that performs the bias mitigation, an explanation of the bias measured, and references to the selected classifiers and debiasing algorithms to help users understand the applied bias mitigation and metrics measured. It also contains a table with colour coding that compares the bias metrics measured by different datasets, classifiers, and debiasing algorithms to help users select the best combinations that fit their contexts.

As an example, Figure 5 shows the summary table, which is part the generated notebook's screenshot, of the bias measurements produced by Listing 1. There are three combinations of prediction. In the first combination, we only measure the Mean Difference, a.k.a. Statistical Parity Difference, in the original dataset. After performing prediction using Logistic Regression, the second combination measures some metrics without any bias mitigation. The table shows the mean difference is slightly worsened to -0.206, moving away from 0, advantaging the privileged group (negative sign). After applying the Exponentiated Gradient Reduction as the debiasing algorithm, the prediction accuracy is slightly reduced to 0.79 but with significant improvement in fairness; both Mean Difference and Average Odds Difference are closer to zero than their values prior to the bias mitigation.

There are situations when users are presented with many different combinations of datasets, classifiers, debiasing algorithms, and bias metric values. Among these options, they usually have to choose which combinations are best for their bias mitigation strategies. Therefore, we assist users by formatting bold the best measured value in each metric and also colour code the values. The coding grades from white to yellow to green; each corresponds to the respective metrics' worst, medium, and best measured values. In Figure 5, users can effortlessly notice that the second combination has the best accuracy. They could also identify that the third option is the least-biased combination but still high in accuracy.

[8]https://www.eclipse.org/epsilon/doc/flexmi
[9]https://pypi.org/project/p2j/

4 EVALUATION

We have evaluated the expressiveness, correctness, conciseness, and execution time of FairML. For expressiveness evaluation, we aimed to answer this question, "*Can FairML support real-world bias mitigation use cases?*". For that, we used the examples provided by IBM Fairness AI 360 as the baseline to assess the expressiveness of FairML. The toolkit comes with some examples (Table 1) that demonstrate the use-cases of the toolkit in real-world contexts. We used the 12 examples that come in version 3.0 of the toolkit. We identified all the elements – bias metrics, debiasing algorithms, classifiers, and datasets – used in each example to measure its complexity. More elements covered by the examples more complex cases FairML can express.

We compared the bias metric values measured (measurement values) in the original examples with the values measured in the generated code to evaluate the correctness of FairML's generated code. Both should produce equal or similar values, within the range of ± 0.1 tolerance.

For conciseness evaluation, we compared the ratio of the written vs. generated code of FairML vs. Code in the examples. While this metric does not always guarantee productivity due to many factors (see Section 4.2), the number of lines written by users is significantly reduced. In the measurement, whitespace and commented lines were not counted.

We also measured the execution time of FairML. First, we measured the *generation time* – the time that FairML takes to generate the bias detection and mitigation code. Second, we compared the time taken by the generated target code against the example code to complete their operations (*generated execution time* vs. *original execution time*). The evaluation was performed on a machine with Windows 10 64-bit operating system, 11th Gen Intel(R) Core(TM) i9-11900H @ 2.50GHz 8-core processors, 32.0 GB RAM DDR4, OpenJDK Runtime Environment 18.9 (build 11.0.14.1+1), and Python 3.9.7.

4.1 Results and Discussion

In this section, we present and discuss the results of our FairML evaluation[10].

4.1.1 Expressiveness and Correctness. FairML was able to reproduce all the 12 examples in Table 1. In total, the scenarios covered 6 unique datasets (Adult, German, Compas, MEPSDataset19, MEPSDataset20, MEPSDataset21), 6 classifiers (Logistic Regression, Linear Regression, Linear SVR, Decision Tree, Kernel Ridge, Random Forest), 11 bias mitigation algorithms (Adversarial Debiasing, Calibrated Equalising Odds, Disparate Impact Remover, Exponentiated Gradient Reduction, Gerryfair, Learning Fair Representation, Reweighing, Meta Fair Classification, Optimized Preprocessing, Reject Option Classification, Prejudice Remover), and 13 bias metrics

[10]The data of the evaluation can be found at https://github.com/York-and-Maastricht-Data-Science-Group/fairml/blob/main/data/evaluation.xlsx

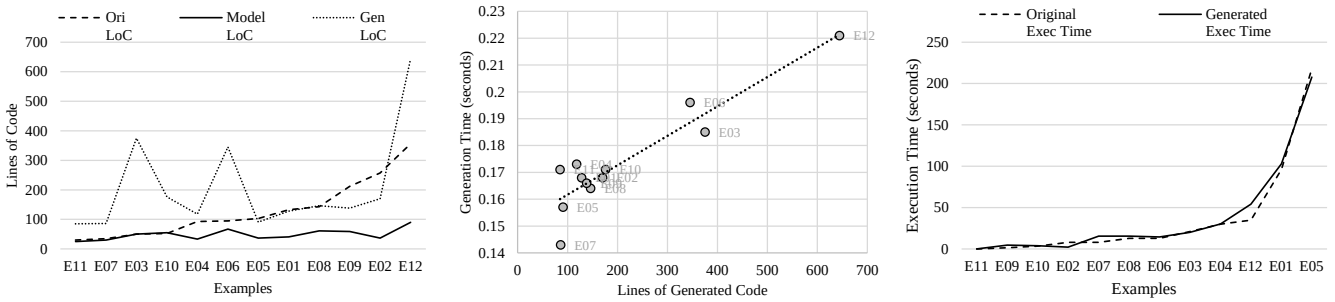
Worst = White, Best = Green, Mid = Yellow

	Mitigation	Dataset	Classifier	accuracy	mean_difference	average_odds_difference
1	Original	Adult(7.0:3.0:0.0)		None	-0.198048	None
2	Original	Adult(7.0:3.0:0.0)	LogisticRegression solver='lbfgs'	**0.804204**	-0.205572	-0.272736
3	ExponentiatedGradientReduction estimator=LogisticRegression(solver='lbfgs'), constraints='EqualizedOdds', drop_prot_attr=False	Adult(7.0:3.0:0.0)	ExponentiatedGradientReduction estimator=LogisticRegression(solver='lbfgs'), constraints='EqualizedOdds', drop_prot_attr=False	0.787552	**-0.052739**	**0.010994**

Figure 5: The summary table displayed in the generated Juypter notebook file based on the model defined in Listing 1.

Table 1: IBM AI Fairness 360 examples and the numbers of unique datasets, classifiers, debiasing algorithms, bias metrics, and measurement values in each example.

Code	Example/File Name (*.ipynb)	Datasets	Classifiers	Debiasing Algorithms	Bias Metrics	Measurements
E01	Demo Adversarial Debiasing	Adult	N/A	Adversarial debiasing	Accuracy, Balanced accuracy, Disparate impact, Average odds, Statistical parity, Equal opportunity, Theil index	28
E02	Demo Calibrated Eq Odd Postprocessing	Adult, German, Compas	Logistic regression	Calibrated eq odds postprocessing	Mean difference, False positive rate, False negative rate, Balanced accuracy, Equal opportunity	13
E03	Demo Disparate Impact Remover	Adult	Logistic regression	Disparate impact remover	Disparate impact	11
E04	Demo Exponentiated Gradient Reduction	Adult	Logistic regression	Exponentiated gradient reduction	Accuracy, Mean difference, Average odds	8
E05	Demo LFR	Adult	Logistic regression	Learning fair representation	Balanced accuracy, Mean difference, Disparate impact	5
E06	Demo Meta Classifier	Adult	N/A	Meta fair classifier	Accuracy, Balanced accuracy, Disparate impact, False discovery rate	13
E07	Demo Optim Preproc Adult	Adult	N/A	Optimized preprocessing	Mean difference	2
E08	Demo Reject Option Classification	Adult, German, Compas	Logistic Regression	Reject option classification	Balanced accuracy, Disparate impact, Average odds, Statistical parity, Equal opportunity, Theil index	26
E09	Demo Reweighing Preproc	Adult, German, Compas	Logistic Regression	Reweighing	Balanced accuracy, Disparate impact, Average odds,. Statistical parity, Equal opportunity, Theil index	16
E10	Demo Short Gerryfair Test	Adult	Linear Regression, Linear SVR, Decision Tree, Kernel Ridge	Gerryfair	Gamma disparity	13
E11	Tutorial Credit Scoring	German	N/A	Reweighing	Mean difference	2
E12	Tutorial Medical Expenditure	MEPS-Dataset19, MEPS-Dataset20, MEPS-Dataset21	Random Forest, Logistic Regression	Reweighing, Prejudice remover	Balanced accuracy, Disparate impact, Average odds, Statistical parity, Equal opportunity, Theil index	90



(a) original vs. model lines of code **(b) generation time vs. lines of code** **(c) original vs. generated code's execution time**

Figure 6: The evaluation of FairML on conciseness, generation time, and execution time.

(Accuracy, Balanced Accuracy, Mean Difference, Statistical Parity Difference, Disparate Impact, Equal Opportunity Difference, Theil Index, Gamma Disparity, Average Odds Difference, Mean Absolute Error, False Positive Rate, False Negative Rate, False Discovery Rate). In addition, FairML also supports loading data from external CSV files. These allow users to express their bias mitigation strategies in different combinations of datasets, classifiers, debiasing algorithms, and bias metrics.

Regarding correctness, due to the randomness in machine learning processes, not all values produced by the generated code are exactly equal to their respective values in the examples. However, the values are still within the range of ± 0.1 tolerance of their respective ones.

4.1.2 Conciseness. Figure 6a shows a comparison of the numbers of lines of code (LoC) between the original examples, FairML models, and generated code. It shows that model LoC (solid line) tends to be less than the original LoC (dashed line), indicating that users would write less code to produce the same results as in the examples (see Section 4.1.1 for correctness). Moreover, the efficiency also gets higher as the number of original LoC increases. From examples E11 (leftmost) to E12 (rightmost), where original LoC goes up from 30 to 356, we can notice the pattern that the efficiency is also increased with users only need to write from 83% to 25% of the LoC in their respective original examples.

We do expect that FairML generates more LoC (dotted line) than the original examples as they contain features that are not included in the original examples, such as (1) the code for explaining the classifiers, bias mitigation algorithms, and bias metrics being used, as well as (2) different approach in displaying measurement results in the forms of summary tables and charts.

4.1.3 Generation and Execution Time. Figure 6b shows the duration – generation time – taken by FairML to produce generated code. In terms of generation time, FairML can generate every generated-code version of the original examples in less than 0.2 seconds, and it can achieve the rate of 2,927 lines/second (644 lines/0.22 seconds) based on the example E12.

Figure 6c shows the performance of the generated code on execution time against the original examples (in seconds). In general, the execution time of the generated code (solid line) takes slightly more time than the original examples (dotted line). We do anticipate this as the generated-code versions have more features – having more lines of code – as stated in Section 4.1.2.

4.2 Threats to Validity

While we have developed and tested FairML based on the documentation and examples of IBM AI Fairness 360, we have not performed user evaluation to obtain feedback from experienced users in the field. This would require the participation of a number of data scientists with experience in bias identification and mitigation, which is a very scarce resource [34]. Nevertheless, the documentation and the examples we reproduced were written by experts and the data they operate on are real-world data from a variety of domains.

5 LESSONS LEARNT

Some lessons that we learnt from this work are:

- Bias identification and mitigation are supported by a number of well-understood algorithms that target different types of metrics and biases. Therefore, they can provide a solid foundation for a well-bounded DSL, such as FairML.
- Bias identification and mitigation processes are easy to get wrong (e.g., applying algorithms in the wrong orders). Expressing them in a DSL and using model-to-text transformation impose certain structures on the generated code, which is not automatically performed when the code is manually crafted.
- Generating Jupyter notebooks from models directly through model-to-text transformation or a model-to-JSON transformation is technically demanding. Still, we can avoid it by leveraging 3rd party engines such as P2J, which can transform textual Python programs into Jupyter notebooks.

6 RELATED WORK

Some toolkits have been developed to measure bias in machine learning. Another FairML, a toolbox developed by [1], audits the fairness of a predictive model by calculating the relative effects of its different inputs on its predictions by leveraging four input ranking algorithms and model compression. FairTest [46] calculates the associations between sensitive attributes and predicted labels to check biases in a dataset. It also provides a methodology to identify the input space's regions where an algorithm might produce unusual high rates of errors. Themis [22], a bias toolbox,

allows automatic generation of tests to measure discrimination in the decisions of a predictive system. Fairness Measures [49] supports the measurement of different bias metrics, such as disparate impact, average odds ratio, and mean difference. Aequitas [41] is an auditing toolkit that measures fairness in different metrics. It also provides a decision tree to guide users in choosing the most appropriate metrics for a particular situation.

Some other toolkits are also able to mitigate biases, not only for measuring fairness. They are ThemisML [6], Fairness Comparison [21], Aequitas [41], Google What-If [48], Scikit-fairnet/scikit-lego [42, 43], Fairlearn [9], and IBM AI Fairness 360 [7].

Rapidminer [24], Knime [8], and Orange [17] are platforms that use low-code, model-based approaches to allow users to visually program machine learning pipelines by assembling component-based procedures. They support data exploration, transformation, visualisation, and different algorithms for machine learning and data mining. Moreover, all of them are extensible, which means users can add new modules or scripts. However, as far as we are aware, there are no built-in modules for measuring and mitigating bias.

The closest existing work to FairML is Arbiter [52], a domain specific-language designed for ethical machine learning. It is an SQL-like declarative language to define how to train machine learning models with four components to describe its ethical practices: transparency, fairness, accountability, and reproducibility. Unfortunately, the implementation is limited only to a particular metric and classifier[11].

7 CONCLUSIONS AND FUTURE WORK

This paper presented FairML, a toolkit that implements a model-based approach to automate bias mitigation. Using FairML, users can generate bias mitigation code in a concise and declarative manner and generate executable Python code from them. Furthermore, in terms of correctness, the generated code produces similar bias metric values to the ones measured in the original examples.

For future work, the number of lines of code generated by FairML can still be further reduced by merging repeatable lines into functions and removing unnecessary code determined by firstly performing static analysis. As a knock-on effect, removing unnecessary code can optimise the execution time of the generated code.

8 ACKNOWLEDGEMENTS

This work has been funded through the York-Maastricht partnership's Responsible Data Science by Design programme (https://www.york.ac.uk/maastricht). We thank the Maastricht team for all their valuable contributions.

REFERENCES

[1] Julius A Adebayo et al. 2016. *FairML: ToolBox for diagnosing bias in predictive modeling*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[2] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. 2018. A Reductions Approach to Fair Classification. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 60–69. https://proceedings.mlr.press/v80/agarwal18a.html

[3] Alekh Agarwal, Miroslav Dudik, and Zhiwei Steven Wu. 2019. Fair Regression: Quantitative Definitions and Reduction-Based Algorithms. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 120–129. https://proceedings.mlr.press/v97/agarwal19d.html

[4] AI Fairness 360 (AIF360) Authors. 2022. AI Fairness 360 documentation. https://aif360.readthedocs.io/en/stable/ Accessed: 2022-01-30.

[5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks. ProPublica (2016). , 23 pages. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing Accessed: 2022-01-18.

[6] Niels Bantilan. 2018. Themis-ml: A Fairness-Aware Machine Learning Interface for End-To-End Discrimination Discovery and Mitigation. *Journal of Technology in Human Services* 36, 1 (2018), 15–30. https://doi.org/10.1080/15228835.2017.1416512 arXiv:https://doi.org/10.1080/15228835.2017.1416512

[7] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. arXiv:1810.01943 [cs.AI] https://arxiv.org/abs/1810.01943

[8] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. 2008. KNIME: The Konstanz Information Miner. In *Data Analysis, Machine Learning and Applications*, Christine Preisach, Hans Burkhardt, Lars Schmidt-Thieme, and Reinhold Decker (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 319–326.

[9] Sarah Bird, Miro Dudík, Richard Edgar, Brandon Horn, Roman Lutz, Vanessa Milan, Mehrnoosh Sameki, Hanna Wallach, and Kathleen Walker. 2020. *Fairlearn: A toolkit for assessing and improving fairness in AI.* Technical Report MSR-TR-2020-32. Microsoft. https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/

[10] M. Brambilla, J. Cabot, and M. Wimmer. 2017. *Model-Driven Software Engineering in Practice*. Morgan & Claypool Publishers. https://books.google.co.uk/books?id=dHUuswEACAAJ

[11] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research, Vol. 81)*, Sorelle A. Friedler and Christo Wilson (Eds.). PMLR, 77–91. https://proceedings.mlr.press/v81/buolamwini18a.html

[12] C. Byrne. 2017. *Development Workflows for Data Scientists*. O'Reilly Media. https://books.google.co.uk/books?id=84HgwQEACAAJ

[13] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. 2017. Optimized Pre-Processing for Discrimination Prevention. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf

[14] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. 2019. Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (Atlanta, GA, USA) *(FAT* '19)*. Association for Computing Machinery, New York, NY, USA, 319–328. https://doi.org/10.1145/3287560.3287586

[15] Jiahao Chen, Nathan Kallus, Xiaojie Mao, Geoffry Svacha, and Madeleine Udell. 2019. Fairness Under Unawareness: Assessing Disparity When Protected Class Is Unobserved. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (Atlanta, GA, USA) *(FAT* '19)*. Association for Computing Machinery, New York, NY, USA, 339–348. https://doi.org/10.1145/3287560.3287594

[16] Pedro Conceição and Pedro Ferreira. 2000. 1The Young Person's Guide to the Theil Index: Suggesting Intuitive Interpretations and Exploring Analytical Applications.

[17] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. 2013. Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research* 14 (2013), 2349–2353. http://jmlr.org/papers/v14/demsar13a.html

[18] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through Awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (Cambridge, Massachusetts) *(ITCS '12)*. Association for Computing Machinery, New York, NY, USA, 214–226. https://doi.org/10.1145/2090236.2090255

[19] Clark Evans, O Ben-Kiki, and I döt Net. 2017. YAML Ain't Markup Language (YAML™) Version 1.2. https://yaml.org/spec/1.2.2 Accessed: 2022-01-19.

[20] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for

[11] https://github.com/julian-zucker/arbiter

Computing Machinery, New York, NY, USA, 259–268. https://doi.org/10.1145/2783258.2783311

[21] Sorelle A. Friedler, Carlos Scheidegger, Suresh Venkatasubramanian, Sonam Choudhary, Evan P. Hamilton, and Derek Roth. 2019. A Comparative Study of Fairness-Enhancing Interventions in Machine Learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (Atlanta, GA, USA) *(FAT* '19)*. Association for Computing Machinery, New York, NY, USA, 329–338. https://doi.org/10.1145/3287560.3287589

[22] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness Testing: Testing Software for Discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (Paderborn, Germany) *(ESEC/FSE 2017)*. Association for Computing Machinery, New York, NY, USA, 498–510. https://doi.org/10.1145/3106237.3106277

[23] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf

[24] M. Hofmann and R. Klinkenberg. 2016. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. CRC Press. https://books.google.co.id/books?id=Y_wYCwAAQBAJ

[25] IBM AI Research. 2022. Welcome to LALE's API documentation! https://lale.readthedocs.io/en/latest/modules/lale.lib.aif360.util.html#lale.lib.aif360.util.theil_index Accessed: 2022-01-30.

[26] IBM Research Trusted AI. 2022. Guidance on choosing metrics and mitigation. https://aif360.mybluemix.net/resources#guidance Accessed: 2022-01-30.

[27] Faisal Kamiran and Toon Calders. 2011. Data preprocessing techniques for classification without discrimination. *Knowl. Inf. Syst.* 33, 1 (2011), 1–33. https://doi.org/10.1007/s10115-011-0463-8

[28] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. 2012. Decision Theory for Discrimination-Aware Classification. In *2012 IEEE 12th International Conference on Data Mining*. 924–929. https://doi.org/10.1109/ICDM.2012.45

[29] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-Aware Classifier with Prejudice Remover Regularizer. In *Machine Learning and Knowledge Discovery in Databases*, Peter A. Flach, Tijl De Bie, and Nello Cristianini (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 35–50.

[30] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 2564–2572. https://proceedings.mlr.press/v80/kearns18a.html

[31] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2019. An Empirical Study of Rich Subgroup Fairness for Machine Learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (Atlanta, GA, USA) *(FAT* '19)*. Association for Computing Machinery, New York, NY, USA, 100–109. https://doi.org/10.1145/3287560.3287592

[32] Dimitrios S. Kolovos, Nicholas Matragkas, and Antonio García-Domínguez. 2016. Towards Flexible Parsing of Structured Textual Model Representations. In *Proceedings of the 2nd Workshop on Flexible Model Driven Engineering co-located with ACM/IEEE 19th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2016), Saint-Malo, France, October 2, 2016 (CEUR Workshop Proceedings, Vol. 1694)*, Davide Di Ruscio, Juan de Lara, and Alfonso Pierantonio (Eds.). CEUR-WS.org, 22–31. http://ceur-ws.org/Vol-1694/FlexMDE2016_paper_3.pdf

[33] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. 2019. iFair: Learning Individually Fair Data Representations for Algorithmic Decision Making. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 1334–1345. https://doi.org/10.1109/ICDE.2019.00121

[34] Michelle Seng Ah Lee and Jat Singh. 2021. *The Landscape and Gaps in Open Source Fairness Toolkits*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445261

[35] T. Mahoney, K.R. Varshney, M. Hind, and an O'Reilly Media Company Safari. 2020. *AI Fairness: How to Measure and Reduce Unwanted Bias in Machine Learning*. O'Reilly Media, Incorporated. https://books.google.co.id/books?id=uSbfzQEACAAJ

[36] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6, Article 115 (jul 2021), 35 pages. https://doi.org/10.1145/3457607

[37] A.C. Müller and S. Guido. 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media. https://books.google.co.uk/books?id=vbQlDQAAQBAJ

[38] Oxford Reference. 2022. Bias. https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095504939 Accessed: 2022-01-16.

[39] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On Fairness and Calibration. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/b8b9c74ac526fffbeb2d39ab038d1cd7-Paper.pdf

[40] Louis M. Rose, Richard F. Paige, Dimitrios S. Kolovos, and Fiona A. C. Polack. 2008. The Epsilon Generation Language. In *Model Driven Architecture – Foundations and Applications*, Ina Schieferdecker and Alan Hartman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–16.

[41] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T. Rodolfa, and Rayid Ghani. 2019. Aequitas: A Bias and Fairness Audit Toolkit. arXiv:1811.05577 [cs.LG] https://arxiv.org/abs/1811.05577

[42] scikit-fairness. 2022. scikit-fairness. https://scikit-fairness.netlify.app/ Accessed: 2022-01-30.

[43] scikit-lego. 2022. scikit-lego. https://scikit-lego.readthedocs.io/en/latest/index.html Accessed: 2022-01-30.

[44] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P. Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. 2018. A Unified Approach to Quantifying Algorithmic Unfairness: Measuring Individual and Group Unfairness via Inequality Indices. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2239–2248. https://doi.org/10.1145/3219819.3220046

[45] D. Steinberg, F. Budinsky, and E. Merks. 2009. *EMF: Eclipse Modeling Framework*. Addison-Wesley. https://books.google.co.id/books?id=oAYcAAAACAAJ

[46] Florian Tramèr, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest: Discovering Unwarranted Associations in Data-Driven Applications. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. 401–416. https://doi.org/10.1109/EuroSP.2017.29

[47] M. Völter, T. Stahl, J. Bettin, A. Haase, S. Helsen, K. Czarnecki, and B. von Stockfleth. 2013. *Model-Driven Software Development: Technology, Engineering, Management*. Wiley. https://books.google.co.uk/books?id=9ww_D9fAKncC

[48] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2020. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 56–65. https://doi.org/10.1109/TVCG.2019.2934619

[49] "Meike Zehlike, Carlos Castillo, Francesco Bonchi, Ricardo Baeza-Yates, Sara Hajian, and Mohamed Megahed". 2017. FAIRNESS MEASURES: A Platform for Data Collection and Benchmarking in discrimination-aware ML. https://fairnessmeasures.github.io. https://fairnessmeasures.github.io

[50] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning Fair Representations. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 325–333. https://proceedings.mlr.press/v28/zemel13.html

[51] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating Unwanted Biases with Adversarial Learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (New Orleans, LA, USA) *(AIES '18)*. Association for Computing Machinery, New York, NY, USA, 335–340. https://doi.org/10.1145/3278721.3278779

[52] Julian Zucker and Myraeka d'Leeuwen. 2020. *Arbiter: A Domain-Specific Language for Ethical Machine Learning*. Association for Computing Machinery, New York, NY, USA, 421–425. https://doi.org/10.1145/3375627.3375858