FairPipes: Data Mutation Pipelines for Machine Learning Fairness

Camille Molinier, Paul Temple, Gilles Perrouin

FairPipes: Data Mutation Pipelines for Machine Learning Fairness

Camille Molinier
ESIR, University of Rennes
Rennes, France
camille.molinier@etudiant.univ-rennes.fr

Paul Temple
University of Rennes, CNRS, Inria, IRISA
Rennes, France
paul.temple@irisa.fr

Gilles Perrouin
PReCISE/NaDI, University of Namur
Namur, Belgium
gilles.perrouin@unamur.be

ABSTRACT

Machine Learning (ML) models are ubiquitous in decision-making applications impacting citizens' lives: credit attribution, crime recidivism, etc. In addition to seeking high performance and generalization abilities, ensuring that ML models do not discriminate against citizens regarding their age, gender, or race is essential. To this end, researchers developed various *fairness* assessment techniques, comprising fairness metrics and mitigation approaches, notably at the model level. However, the sensitivity of ML models to fairness data perturbations has been less explored. This paper presents mutation-based pipelines to emulate fairness variations in the data once the model is deployed. FairPipes implements mutation operators that shuffle sensitive attributes, add new values, or affect their distribution. We evaluated FairPipes on seven ML models over three datasets. Our results highlight different fairness sensitivity behaviors across models, from the most sensitive perceptrons to the insensitive support vector machines. We also consider the role of model optimization in fairness performance, being variable across models. FairPipes automates fairness testing at deployment time, informing researchers and practitioners on the fairness sensitivity evolution of their ML models.

CCS CONCEPTS

• **Software and its engineering** → **Software testing and debugging**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Machine Learning, Fairness, Mutation Testing

ACM Reference Format:
Camille Molinier, Paul Temple, and Gilles Perrouin. 2024. FairPipes: Data Mutation Pipelines for Machine Learning Fairness. In *Proceedings of 5th International Conference on Automation of Software Test (AST 2024).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Machine Learning (ML) models are prevalent in various applications, from automatic translation to granting insurance or predicting crime recidivism chances. Thus, most of these applications impact many aspects of citizens' lives. One must design, train, and deploy high-quality ML models that are often limited to finding the best performance, *i.e.,* tuning the model to reduce prediction errors as much as possible. This performance quest motivated the design of more and more sophisticated models from decision trees [5] to various neural architectures [3]. However, performance is sometimes achieved at the expense of other qualities such as *fairness* [10]. Achieving fairness means having similar attributes regarding a specific task should result in a similar decision from the ML model [6, 10]. For instance, if two persons ask for a loan. If they both have the same status (same education level, same job, *etc.*) except that one is younger by a year than the other, if the loan is given to the younger one, it should also be given to the older one. In particular, model decisions should be free from biases concerning gender, age, or ethnicity. Unfortunately, examples of biases abound in deployed ML systems, should they be for image recognition [17], natural language processing [19], or crime recidivism prediction tasks [16].

The ML and software engineering communities have acknowledged the extent and diversity of biases in software and developed many bias mitigation methods to improve fairness, *e.g.,* [10, 27]. These communities also designed metrics to assess fairness [2], supporting fairness improvement in ML models. Yet, as noted by Yang *et al.* [26], how ML models will react to fairness perturbations once deployed is a less explored domain.

Our goal is to support ML engineers in choosing the ML algorithm best suited to their fairness and performance needs and gain confidence in the robustness of the model (regarding fairness) once in production. In this paper, we take inspiration from mutation testing techniques [21] to mimic fairness perturbations and investigate the following research questions:

- **RQ1:** *Which models are sensitive to fairness mutations at test time?*;
- **RQ2:** *Which mutation operators are the most relevant for fairness sensitivity assessment?*;
- **RQ3:** *Does model optimisation influence fairness sensitivity?*

This paper makes the following contributions:

(1) *FairPipes*, a set of automated pipelines inspired by mutation testing [21], that trains various types of ML models, injects perturbations (*mutations*) affecting protected (*sensitive*) attributes in their test sets and measure the impact of these mutations on the model's fairness properties and performance (accuracy);

AST 2024, April 2024, Lisbon, Portugal
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

(2) FairPipes adapts three existing mutation operators (*mutators*) and introduces a new operator that adds a new value for the protected attribute.
(3) FairPipe evaluation on three datasets and for seven different models, totalizing 53,900 executions of our pipelines, reveals sensitivity patterns, notably the robustness to fairness mutations of the SVM and decision trees (RQ1). We found no predominance of one mutator over the others (RQ2). Finally, optimization influences model sensitivity positively or negatively depending on model type (RQ3);
(4) *Open Science Policy.* All the results of our evaluation are available on the following website: https://zenodo.org/record/8335266. The source code of FairPipes is also available: https://github.com/Camille-Molinier/mutation-testing-ml-fairness.

In the following, we describe mutation testing and some popular measures to assess the fairness of ML models in Section 2. Section 3 presents our framework and defines a first set of mutators that can be easily extended. We run our framework against a set of ML models over different datasets in Section 4 and draw general conclusions about the sensitivity of the different ML models regarding our mutators. Section 6 concludes and discusses future directions.

2 BACKGROUND & RELATED WORK

2.1 Mutation Testing

Mutation testing [9] assesses the quality of existing tests by first injecting artificial faults to form *mutants* of a System Under Test (SUT). Then, it runs the existing tests over the newly created mutants. If testing outcomes differ from the original system, it means tests can *kill* or distinguish mutants, revealing their sensitivity to these artificial faults. These faults mimic the kind of errors that developers could make. For instance, due to fatigue, they could exchange a "$\geq$" into a ">" or a "=" into a "! =". One injects these faults through mutation operators or *mutators* to create automatically these mutants. Thus, mutation testing relies on the idea that the more sensitive test suites are to these faults, the more likely they are to find real ones. One can measure the strength of test suites by computing the ratio of killed versus non-killed mutants, called the mutation score. Mutation testing is now a mature approach to test quality, and mutation frameworks exist for many programming languages and applications [21].

The spread of ML algorithms and concerns about their correct behavior in deployed software led to the transfer of the mutation testing paradigm to machine learning [15, 18, 20, 24]. As a result, many mutators were proposed, from changing the training data labels to removing neurons in a deep neural network [15]. The design of fairness-aware mutators is more recent and less explored [14, 25]. This paper explores the impact of such fairness operators on the ML models at test time.

2.2 Fairness Assessment

While performance prediction measures, such as accuracy or F1-score, are the most predominant metrics to optimize ML models while learning, other measures, such as fairness, are gaining interest. This interest is probably due to the use of ML models in a growing number of decision-making algorithms impacting people's lives. A prediction model being fair is supposed to treat any data equally as long as the decision that is being taken does not involve taking into account any characteristics that would favor or harm a part of the population. ML models are not exempt from biases due to training data, feature extraction, or model hypotheses. This motivates the need to assess and mitigate such biases [8].

The fairness community developed several fairness measures. We can think of demographic parity [10], equalized odds [12], equalized opportunity [12], statistical parity [11, 27], disparate impact [7, 11], or threshold testing [23]. They can be grouped into families depending on their mathematical expression as in the FairML book [2].

In the following, we will focus on two of the most popular measures: *demographic parity* and *equalized opportunity*. Their mathematical expressions can be stated in different ways (*e.g.,* a ratio or a difference). Here, we focus on the form that uses differences as shown in Equations 1 and 2. Both measures use the predicted value of the classifier (called $\hat{Y}$) and a protected attribute, that we call A. The population is divided into two groups: the *disadvantaged* one (identified by $A = 1$) and the *privileged* one (identified by $A = 0$). *demographic parity* is defined as an absolute difference [10]:

$$DP = \left| P\hat{Y} = 1 | A = 0 - P\hat{Y} = 1 | A = 1 \right| \leq \epsilon \quad (1)$$

In the end, this definition states that no difference (or in between an ϵ margin) should be observed in the prediction between two data when only the value of the protected attribute changes (which is not supposed to impact the prediction anyway). Similarly, equal opportunity is defined as follows:

$$EO = \left| P\hat{Y} = 1 | A = 0, Y = 1 - P\hat{y} = 1 | A = 1, Y = 1 \right| \leq \epsilon \quad (2)$$

where $P\hat{Y} = 1$ is the prediction of an ML model to the supposedly disadvantaged group, A is the protected attribute (on which the model should not rely to take a decision), and $Y = 1$ is the true outcome. While the two measures have a similar form, equal opportunity takes into consideration the information that the true outcome is $Y = 1$ which is ignored by demographic parity (and is one of the main arguments against using it).

3 OUR APPROACH

3.1 Overview

From a conceptual point of view, we want to propose a framework that can assess whether already deployed ML models are sensitive (in terms of fairness but also accuracy) to potential changes in the data fed to them. In this sense, the ML models cannot be modified. Their training and optimization processes were done on datasets before any changes. To assess model sensitivity, we take inspiration from mutation testing.

First, we define a set of mutators (*i.e.*, operators of modifications that we describe right after). The proposed list is *not exhaustive*, and it reuses some previously defined in the literature while others are new. Figure 1 presents our mutation workflow. The upper part describes a typical training and hyper-optimization process. Once the model is trained, it is evaluated using a test set (upper right part of the figure) that gives the baseline performance measures (*i.e.*, accuracy but also the fairness measures, see Section 2). Then, we apply the mutators to generate new test sets (bottom part). Generally, mutators target changes over the protected attribute, which should not impact the decisions taken by the model. The evaluation procedure (*i.e.,* the evaluation of the performances) is applied again to the modified datasets. Measures are retrieved (bottom right) and can be compared with the ones drawn from the original test set. The final step is to conclude via statistical tests.

3.2 Mutators

Other approaches have been proposed that are similar to our idea. They have defined some operators we have reused in our experiments [1, 13]. We present them hereafter with some others we added as they might reveal behaviours of interest. Two main types of operators are described in the literature: (i) the *parametric* operators and (ii) the *non-parametric* ones. *Parametric* operators typically take a parameter defining the proportion of data to modify, while *non-parametric* operators will operate over the whole dataset. We consider four operators: two *parametric* and two *non-parametric* ones.

First, probably the most intuitive, the *column shuffle* operator. It randomly replaces the value of a protected attribute from one data sample to another. We made it *parametric* so that a specific proportion of the dataset has its value replaced.

For instance, if the operator runs with a parameter set to 12%, it means that 12% of the dataset is randomly chosen and extracted. Then, inside this subset, we change the value of the protected attribute (via permutations). As no other checks are done, permutations may occur between two data having the same feature value regarding the protected attribute. Finally, the value did not change, but a permutation did occur. For instance, this may happen when the distribution of the values of the protected attribute is not balanced.

The second *parametric* operator is called the *new value introducer*. This is a *completely new* operator. It introduces a new value to the set of possible values a protected attribute can take. This way, we can simulate that data may evolve.

Specifically, if the parameter is set to 12%, we change the value of the protected attribute for 12% of randomly chosen data to a new value that was not reported. The newly introduced value is 'null' by default. Yet, our framework has been designed to let users choose any values not reported before (for this specific attribute). As only the protected attribute is changed, and since it should not influence the prediction, we should not expect to see changes in the decisions over the modified data.

Regarding the *non-parametric* operators, first we define the *column killing* operator. It changes the value of an attribute to the same value for all the data. This way, no statistical relations can be drawn. This attribute becomes useless in the decision-making, and if the operator is applied on the protected attribute, as it should occur in the decision process, we expect to see no changes at all in the predictions.

As the number of changes in the outcome increases after using this operator, it suggests a strong correlation between the modified attribute and the decision, which could show that the model was biased. From an implementation point of view, the value assigned to all the data is chosen arbitrarily as the value from the first data of the dataset.

Our last operator is called the *redistribution* operator. It is also *non-parametric* and tries to balance the occurrence of the different values that the protected attribute can take. Again, the idea behind this operator is that if all values are equally represented, their significance in the decision process will be lowered.

4 EVALUATION

To evaluate our approach, we answer the following research questions (RQs):

- **RQ1**: Which models are sensitive to fairness mutations at test time?
- **RQ2**: Which operators are the most relevant for fairness sensitivity assessment?
- **RQ3**: Does model optimisation influence fairness sensitivity?

4.1 Dataset description and preprocessing

We use the *adult census*, *German credit*, and *COMPAS* datasets. They are all built for classification tasks and are probably the most popular datasets when evaluating the fairness properties of ML models. Table 1 presents a short description of these three datasets. While the number of features remains similar between the datasets, the number of samples differs. All datasets have only two possible labels, but their representation (% of data with a label) varies from 24% in the Adult dataset to 70% in the German credit dataset. Compas seems almost balanced with 45.5%.

We did not perform preprocessing over the data except for the *COMPAS* dataset. We tried to reproduce the dataset used in the ProPublica study as it showed that ML models were biased. Thus, we reproduce the same preprocessing [1] that they have proposed.

4.2 ML models

To perform our evaluation and assess the sensitivity of ML models concerning changes in the data we selected seven different ML models. They cover different families of algorithms (among others: support vector machines (SVMs), decision trees, and nearest neighbors). Despite deep learning models being state of the art regarding accuracy measures, they

[1] we reproduce the processing in cells 3 and 4 reported at ProPublica GitHub repo

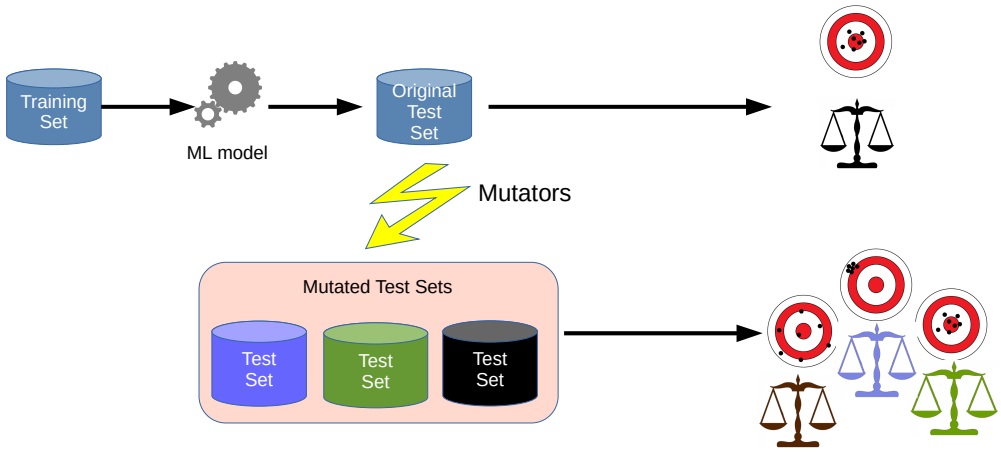

Figure 1: An overview of the proposed framework. It starts with the training of an ML model (upper part). It is evaluated using a test set to observe accuracy and fairness measures (top right). Mutation operators are applied to the test set, and the ML model is again evaluated but using the modified versions of the test set (bottom part). New accuracy and fairness measures are observed and can be used to compare with the original ones.

Foreign worker

	Column shuffle			New class			Column killing			Redistribution		
Model	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod
dt	-	-	NS	-	-	+	-	-	=	-	-	+
dt_opt	=	=	=	=	=	=	=	=	=	=	=	=
rf	NS	NS	NS	NS	-	+	=	=	=	-	+	-
rf_opt	NS	NS	NS	-	-	+	=	=	=	-	-	+
ada	NS	-	+	-	-	+	=	=	=	-	-	+
ada_opt	NS	NS	NS	NS	-	+	=	=	=	+	-	+
xbgc	NS	NS	NS	NS	-	+	=	=	=	+	-	+
xgbc_opt	=	=	=	=	=	=	=	=	=	=	=	=
svm	=	=	=	=	=	=	=	=	=	=	=	=
svm_opt	NS	NS	NS	-	-	+	=	=	=	-	-	+
knn	=	=	=	=	=	=	=	=	=	=	=	=
knn_opt	=	=	=	=	=	=	=	=	=	=	=	=
mlpc	NS	NS	NS	+	-	NS	=	=	=	+	-	+
mlpc_opt	NS	NS	NS	-	-	+	=	=	=	-	-	+

Figure 2: Accuracy, Dpd, and Eod evolutions from the *German* dataset when the Foreign Worker attribute is selected as a protected attribute. Each row is a model (optimized or not), and the operators are represented via columns. A cell reports the comparison between the 50 runs and the original value (when no modifications were used). A red cell with a "-" reports that all the runs report a lower value than the original. A green cell with a "+" reports that all the runs report a higher value than the original. An orange cell with a "- +" reports that sometimes the runs report a higher value and sometimes a lower value. A white cell with a "=" reports no difference (p-value=1 in this extreme case). If the computed p-value is higher than $10e^{-3}$ but < 1, the cell is black with "NS" inside (not significant).

require an amount of data to be trained that is orders of magnitude higher than the one proposed by the considered datasets for evaluation (see Table 1). Furthermore, deep learning models generally perform worse (or at best equally) than non-deep ones for tabular data [4]. Therefore, we excluded them but may consider them in the future. Yet, we still consider multi-layered perceptrons that may be a (simpler) surrogate model.

We consider two versions of these models: the optimised one, which is supposed to be used in real-world conditions as its hyper-parameters have been optimised for the dataset at hand; and the non-optimised one, for which all parameters

Relationship

	Column shuffle			New class			Column killing			Redistribution		
Model	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod
dt	-	-	-	-	-	-	-	-	-	-	-	-
dt_opt	-	-	-	-	-	-	-	-	-	-	-	-
rf	-	-	-	-	-	-	-	-	-	-	=	-
rf_opt	-	-	-	-	-	-	-	-	-	-	=	-
ada	-	-	-	-	-	NS	-	-	-	-	-	-
ada_opt	-	-	-	-	-	-	-	-	-	-	-	-
xbgc	-	-	-	-	- +	-	-	-	-	-	=	-
xgbc_opt	-	-	-	-	+	-	-	-	-	-	-	-
svm	=	=	=	=	=	=	=	=	=	=	=	=
svm_opt	NS	=	+	NS	-	+	=	=	=	-	=	+
knn	NS	-	-	NS	-	=	=	-	-	-	-	=
knn_opt	-	-	NS	-	-	NS	-	-	=	-	-	-
mlpc	-	-	NS	-	-	-	-	-	-	-	-	-
mlpc_opt	-	-	-	-	-	-	-	-	-	-	-	-

Figure 3: Accuracy, Dpd, and Eod evolutions from the *Adult* dataset when the Relationship attribute is selected as a protected attribute. Each row is a model (optimized or not), and the operators are represented via columns. A cell reports the comparison between the 50 runs and the original value (when no modifications were used). A red cell with a "-" reports that all the runs report a lower value than the original. A green cell with a "+" reports that all the runs report a higher value than the original. An orange cell with a "- +" reports that sometimes the runs report a higher value and sometimes a lower value. A white cell with a "=" reports no difference (p-value=1 in this extreme case). If the computed p-value is higher than $10e^{-3}$ but < 1, the cell is black with "NS" inside (not significant).

Age

	Column shuffle			New class			Column killing			Redistribution		
Model	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod	Accuracy	Dpd	Eod
dt	-	=	=	-	=	=	-	=	=	-	=	=
dt_opt	=	=	=	=	=	=	=	=	=	=	=	=
rf	+	=	=	NS	=	=	+	=	=	+	=	=
rf_opt	-	=	=	-	=	=	=	=	=	+	=	=
ada	-	=	=	-	=	=	=	=	=	-	=	=
ada_opt	+	=	=	NS	=	=	+	=	=	+	=	=
xbgc	NS	=	=	-	=	=	=	=	=	+	=	=
xgbc_opt	=	=	=	=	=	=	=	=	=	=	=	=
svm	NS	=	=	NS	=	=	=	=	=	-	=	=
svm_opt	NS	=	=	+	=	=	-	=	=	+	=	=
knn	-	=	=	NS	=	=	-	=	=	+	=	=
knn_opt	-	=	=	-	=	=	-	=	=	-	=	=
mlpc	-	NS	=	-	-	=	+	=	=	-	=	=
mlpc_opt	-	=	=	-	=	=	+	=	=	-	=	=

Figure 4: Accuracy, Dpd, and Eod evolutions from the *Compas* dataset when the Age attribute is selected as a protected attribute. Each row is a model (optimized or not), and the operators are represented via columns. A cell reports the comparison between the 50 runs and the original value (when no modifications were used). A red cell with a "-" reports that all the runs report a lower value than the original. A green cell with a "+" reports that all the runs report a higher value than the original. An orange cell with a "- +" reports that sometimes the runs report a higher value and sometimes a lower value. A white cell with a "=" reports no difference (p-value=1 in this extreme case). If the computed p-value is higher than $10e^{-3}$ but < 1, the cell is black with "NS" inside (not significant).

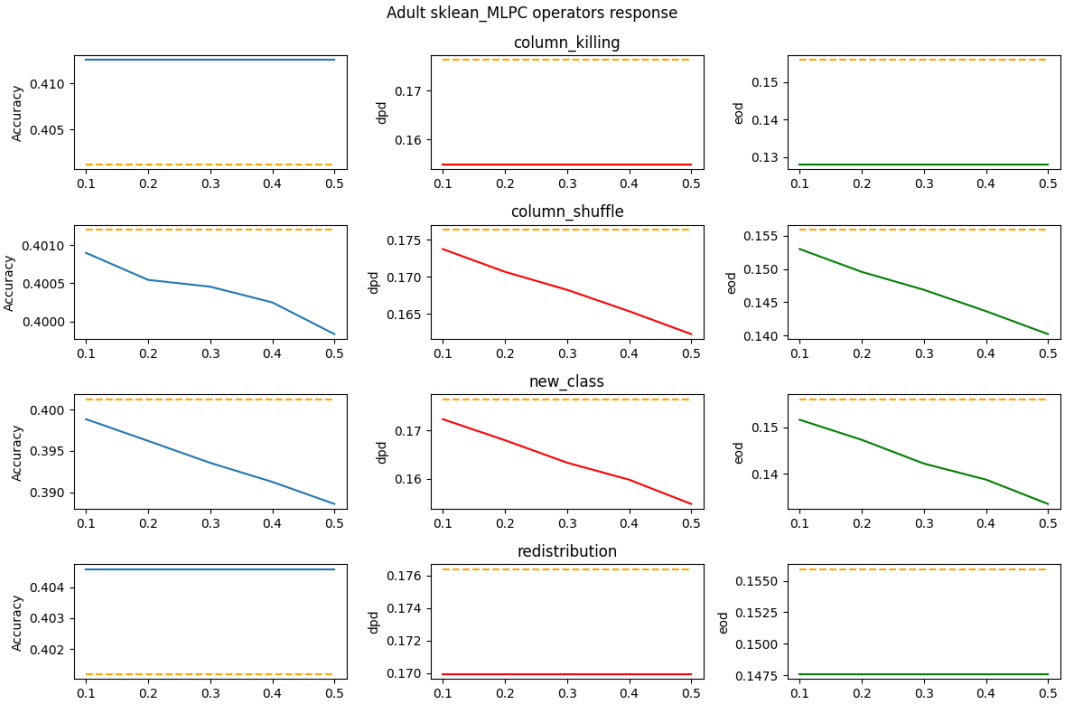

Figure 5: Results for adult dataset and sex protected attribute for a non-optimized mlpc. The subplots are arranged by operator in rows and metric in columns. In all subplots, the yellow dotted curve represents the value without mutation, and the full curve represents the measure with mutation.

Table 1: Description of the three datasets used in the evaluation: Adult, COMPAS, and German credit. We report the number of samples, the number of features per data, what were the features that we used as protected attributes, what is the favourable outcome (*i.e.*, Y=0)), and the task that an ML model is trying to solve using the dataset.

Dataset	#Samples	#Features	Protected attributes	Favorable label (% of data with this label)	ML Task
Adult	32561	15	marital-status, relationship, race, sex, native-country	income > 50k (24%)	Predict if a person's income is higher than 50K per year
Compas	6172	14	sex, age, age_cat, race	two_year_recid==1 (45.5%)	Predict if a defendant is likely to commit a recidivism act in less than two years
German	1000	21	ex-and-marital-status, age (years), foreign worker	Creditability==1 (70%)	Predict if a person is admissible for a credit

are set to the default value given by the library (in our case scikit-learn [22]).

We optimised our models using a grid-search strategy. Because of the combinatorial explosion resulting from considering more and more parameters to optimise, we selected a fixed number of them for each algorithm that we detail hereafter. The different classifiers' implementations and the grid-search strategy come from the scikit-learn library [22].

First, regarding the decision tree classifiers [2], we optimized the following parameters: *max_depth*, *ccp_alpha*, and *min_samples_leaf*. Respectively representing the maximum depth of the tree, the cost-complexity trade-off to select between potential candidates and the minimal number of samples per leaf for pruning.

[2]Using the DecisionTreeClassifier

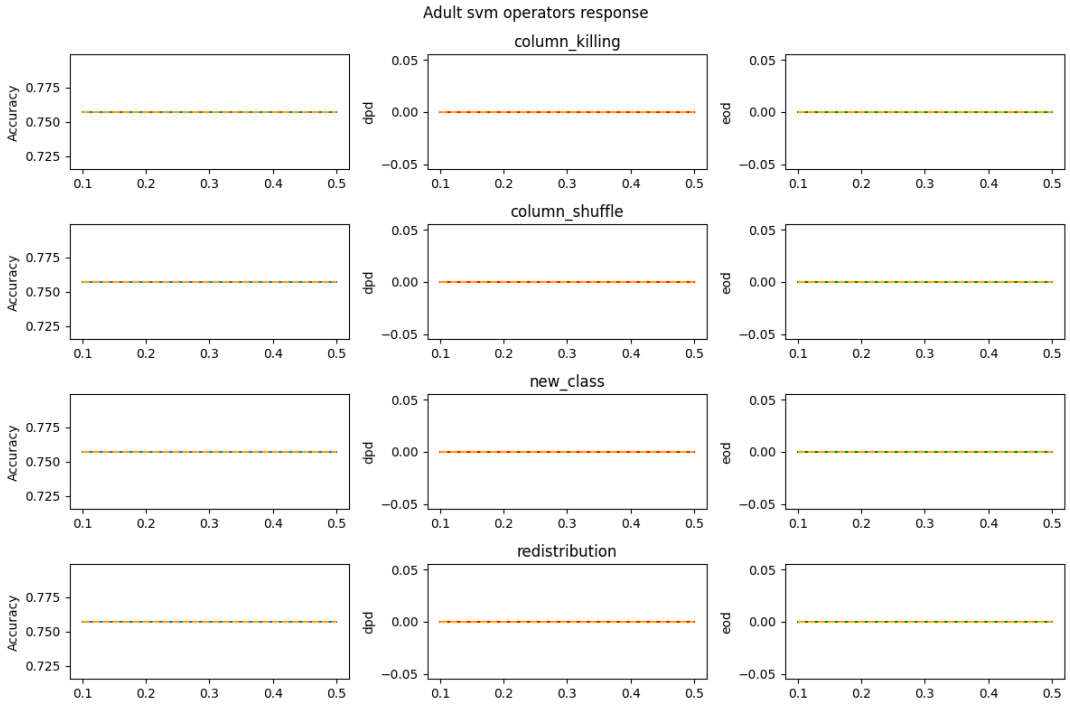

Figure 6: Results for adult dataset and relationship protected attribute for a non-optimized SVM. The subplots are arranged by operator in rows and metric in columns. In all subplots, the yellow dotted curve represents the value without mutation, and the full curve represents the measure with mutation.

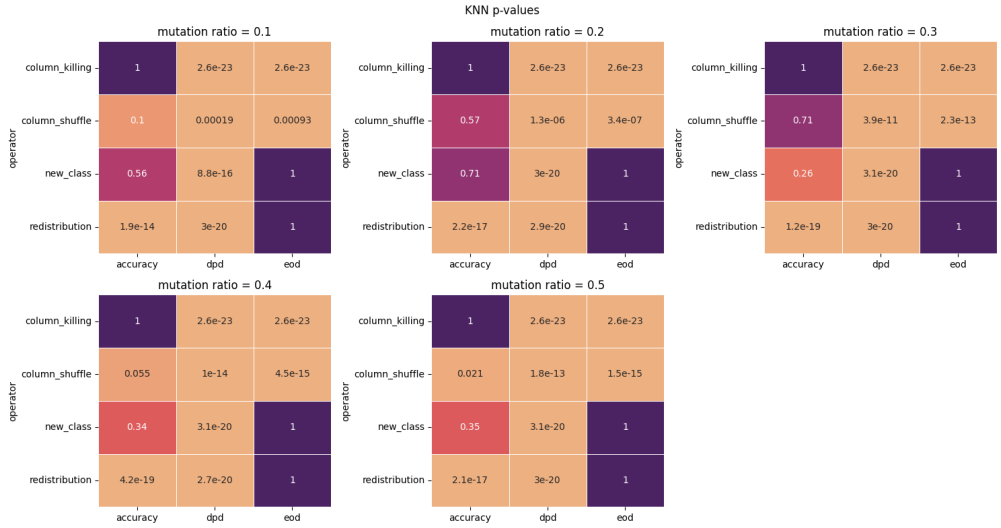

Figure 7: P-values for the *Adult* dataset when *Relationship* is used as the protected attribute and a KNN classifier is trained. For each matrix, the operators are the row, and the measures (Accuracy, Dpd, and Eod) are the columns.

Another popular family of classifiers is ensemble models. We decided to use random forests [3], adaBoost [4], and XGBoost [5]. Regarding the random forests, the following parameters were optimised: *n_estimators* that defines the

[3]Using the RandomForestClassifier
[4]Using the AdaBoostClassifier

[5]Using the dmlc XGBoost implementation

number of trees in the forest, *max_depth* that defines the maximum depth of each tree, *bootstrap* that defines whether to use a bootstrap sample for training and *oob_score* which is the out-of-bag score (an alternative to the accuracy). AdaBoost classifiers were optimised considering the following parameters: *estimator* that defines the model used inside the ensemble, *n_estimators* that defines the number of estimators, and *learning_rate* defining the importance of each model for the final decision. XGBoost models are optimised by exploring: *n_estimators* that defines the number of gradient boosted trees, *learning_rate* that defines the learning rate, and *max_depth* that limits the maximum depth of the trees.

SVMs are different kinds of classifiers that are differentiable as opposed to trees and ensemble methods. We used the scikit-learn SVC implementation [6]. The optimization is performed over the following parameters: *kernel* defining the kernel type to be used (we consider the three most popular kernels: radial basis function (RBF), polynomial, and linear), *C* defining the regularisation parameter, and *gamma* (only when the RBF kernel is used) defining the kernel coefficient in the SVM optimization problem.

We have also used the K-nearest-neighbor (KNN) algorithm [7]. KNNs are mostly considered as an unsupervised method. Yet, different implementations exist, and some allow for turning it into a supervised technique. We optimised the following parameters: *n_neighbors* defining the number of neighbours to consider for taking a decision, *metric* defining the metric to use for distance computation, and *p* defining the power parameter for the Minkowski metric.

The last classifier we used is the multi-layer perceptron (MLP) [8]. We tuned the following parameters: *hidden_layer_sizes* that defines both the number of layers and the number of neurons per layer, *solver* defining the solver for weight optimisation, and *max_iter* defining the maximum number of iterations for optimisation.

4.3 Settings

By definition, ML algorithms are statistical methods implying that random effects may occur and bias our results and conclusions. To mitigate this threat, we ran our experiments 50 times. We measure the evolution of the accuracy and the evolution of two fairness measures (*i.e.*, demographic parity difference (Dpd) and equal opportunity difference (Eod)) and we report p-values and effect sizes to show statistical significance.

As we wrote in Section 4.2, we used grid search optimization while training the model. We also consider the case where users might not want to spend too much effort in optimization and thus will quickly prototype with default values. This leads to a total of 14 models to train for each dataset and each protected attribute. Thus, we ran: (i) for the Adult dataset, 14 models x 5 protected attributes x 50 repetitions $= 3,500$ evaluations that report 3 different measures each; (ii) for the Compas dataset, 14 models x 3 protected attributes x 50 repetitions $= 2,100$ evaluations; and (iii) for the German dataset, 14 models x 3 protected attributes x 50 repetitions $= 2,100$ evaluations. In total, we ran our experiment $3,500 + 2,100 + 2,100 = 7,700$ times all datasets taken into account.

Furthermore, regarding the operators to modify the datasets, we used 4 of them. 2 are parametric and 2 are non-parametric as described in Section 3.2. The 2 parametric ones have a parameter (*i.e.*, the percentage of data to modify) that we also made vary to assess the impact of modifying more and more data. This may simulate a drift in the population. We started with 0% (*i.e.*, the original dataset) and went up to 50% with an increasing step of 10% by 10% (10%, 20%, 30%, 40%, and 50%). Thus, we evaluated 6 different settings for parametric operators and one for non-parametric operators. These 7 settings were applied on all three datasets and for each model (and repeated 50 times). In the end, we ran $7,700$ x $7 = 53,900$ times our pipeline in total.

We ran these $53,900$ evaluations on a Google Colab environment to try to make the running environment as homogeneous as possible. All the code is written in Python 3.9 and does not use any GPU.

4.4 Results

For the sake of space, we focus only on some runs. Yet, all the results are available at https://zenodo.org/record/8335266.

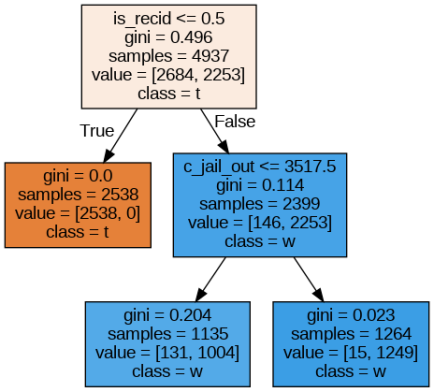

Figure 8: Graphical representation of an optimized decision tree on the *Compas* dataset. Since the max_depth parameter is limited to 2, only some of the attributes can be important for decision-making.

4.4.1 RQ1: Model Sensitivity to Test Set Mutations. Figures 3, 4, and 2 show the results of mutating selected protected attributes on our models for our three datasets. We focus on accuracy, Eod, and Dpd measures as presented previously. In these figures, each cell represents the evolution of the measure when the model is tested with an operator compared with measures from the original test set. A first observation is that the models' responses to mutations generally vary among attributes and datasets. All but one model see their fairness

[6] SVC implementation in sklearn
[7] KNN implementation in sklearn
[8] MLP implementation in sklearn

measure improve [9] at the cost of a lower accuracy score for the adult dataset. This is not the case for other datasets: for example, the mlpc model can see both its accuracy and its Dpd improving for the German dataset when the redistribution operator is applied. Similar trends can be observed with the xgbc model. Looking now at Figure 4, we can see that for most of the models, the fairness measures do not change at all while the accuracy can increase (*e.g.*, random forests (row rf)) or decrease (*e.g.*, adaboost (row ada)). Overall the two most sensitive models are the decision trees and the mlpc. For instance, Figure 5 illustrates the diversified impacts of our operators on the mlpc model. In contrast, SVMs are by far the least sensitive, Figure 6, shows an example of this insensitivity. Remarkably, this model showcases remarkable stability across a wide spectrum of scenarios. Regardless of the operator, the nature of the dataset, or the specific protected attribute under scrutiny, the SVM model consistently maintains an unwavering performance. The three measures remain the same regardless of the dataset (except in Figure 4 with the last operator). As exemplified in Figures 3, 4, and 2, it is evident that save for minor fluctuations in accuracy, this model remains remarkably resilient to variations associated with protected attributes.

Conclusion (RQ1)

Model sensitivity to fairness mutations varies across datasets. The most sensitive models are the decision trees and the mlpc. SVM is unaffected by mutations and remains stable on all datasets.

4.4.2 RQ2: Relevance of Operators. Figures 3, 4, and 2 do not exhibit any particular patterns that would favour the systematic use of one of the selected operators. That is, each operator generally affects the response of a given model differently. For example, Figure 7 the results of the p-values computed for the KNN model on the Adult dataset. We can observe that column killing does not impact at all the accuracy, hence the extreme p-value of 1, while new class and redistribution do the same regarding Eod. In terms of our selected fairness metrics, we see that Dpd is generally improved by our operators. On the German dataset, Eod is worse when the *new class* or the *Redistribution* operator is applied. One exception is the Adult dataset showing that both Dpd and Eod are improved. We did not design our operators so that they necessarily improve the fairness measures, so mutations can be either adverse or beneficial. Again FairPipes focuses on the sensitivity analysis and not fairness improvement. The exploitation of the adversarial perturbations during training time to improve fairness in models can be found in previous work [8].

[9] note that, since Dpd and Eod are differences (see Equations 1 and 2), when they decrease it means the model becomes fairer

Conclusion (RQ2)

There is no redundancy across our four mutation operators, each of them impacting models differently. Column shuffle is the least impacting one, yielding no or insignificant differences in fairness metrics.

4.5 RQ3: Influence of Model Optimisation

The optimized version of decision trees (row dt_opt) is also stable overall, and in particular more stable than the non-optimized version (row dt). We assume this behaviour comes from the reduced size of the tree. Since not all features can be selected to build the tree, only the most discriminant ones are selected (regarding the Gini impurity index). On the other hand, the non-optimized version has no limit of depth, thus all features are likely to be selected at least once. Figure 8 shows one of the trained optimized trees. Finally, we can also notice that adaboost and non-optimized random forests behave very similarly. We assume that this is due to the selection of random forests as a base classifier of our adaboost. This stability trend does not happen for all models, with optimized SVMs being more sensitive than non-optimized ones. One possible explanation is that optimized SVMs learn a more precise separation function to better “fit" the data and thus are more sensitive to variations.

Conclusion (RQ3)

Optimization plays a significant role in model sensitivity to fairness: decision trees can move from the most sensitive to the less sensitive models. The opposite is also possible (SVM).

5 THREATS TO VALIDITY

We now discuss some threats that may mitigate our conclusions.

5.1 Internal threats

The first threat is the combinatorial explosion that forces us to select only some instances. The choice of the algorithms and the grid search do not cover the whole space of possible models to train and evaluate. Yet, we selected a set of ML algorithms covering common families of algorithms. The comparison of optimised models via grid search and non-optimal models extends our selection and shows the difference when an optimisation strategy is applied.

A second threat is that we focused on some features that we defined as protected as they seem less relevant for the defined task. These protected attributes are the ones studied in previous works [16] and are known to favour unfairness when taken into account while training ML models. Yet, other hidden (or unexplored) correlations can be left and should be addressed.

We proposed a framework inspired by mutation testing and defined four different operators to bring modifications to the

data. These operators are a first attempt to check whether they could impact trained ML models. We implemented both unit and statistical tests to check the framework's functionality and ensure the validity of our results. For reproducibility purposes, it is available on our Github: https://github.com/Camille-Molinier/mutation-testing-ml-fairness.

5.2 External threats

We used three different datasets that are popular in ML fairness-related works. These show different characteristics in the number of data and the number of features. We cannot guarantee that our observations generalise to other datatsets.

6 CONCLUSION

ML models are used in ever more applications, which they can directly impact citizens' lives such as in banking or judicial decisions. These applications require to mitigate biases as much as possible regarding their decisions avoiding possible harm to a part of the population. Usually, ML models are trained and evaluated once before deployment, but after some time, changes in data can occur. This work presents a framework that assesses the sensitivity of trained ML models to changes after deployment. In particular, we focus on the impact of changes regarding accuracy and two fairness measures. To do so, we draw inspiration from Mutation Testing, a software testing technique used to evaluate tests. The idea is to change some features from the data, evaluate an ML model on the modified test set, and compare the measure to the original results. We proposed four operators and evaluated their impact on seven different ML algorithms. Our results show that SVMs are less sensitive to changes overall, but depending on the dataset and the operator, the impact on the ML model may drastically change. Our future work naturally includes the investigation of additional operators and other datasets. We would also like to adapt these pipelines to deep learning models and other forms of data (e.g. images).

ACKNOWLEDGMENTS

The authors would like to thank German Herbay for his preliminary work. This work was partly funded by the ARIAC by Digital Wallonia and the FNRS EOS VeriLearn projects. Gilles Perrouin is a resarch associate at the FNRS.

REFERENCES

[1] Pranjal Awasthi, Matthäus Kleindessner, and Jamie Morgenstern. 2020. Equalized odds postprocessing under imperfect group information. In *International conference on artificial intelligence and statistics*. PMLR, 1770–1780.

[2] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2019. *Fairness and Machine Learning: Limitations and Opportunities*. fairmlbook.org. http://www.fairmlbook.org.

[3] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems* 13 (2000).

[4] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–21. https://doi.org/10.1109/TNNLS.2022.3229161

[5] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees*. Taylor & Francis.

[6] Yuriy Brun and Alexandra Meliou. 2018. Software Fairness. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Lake Buena Vista, FL, USA) *(ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 754–759. https://doi.org/10.1145/3236024.3264838

[7] Alexandra Chouldechova. 2017. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *Big Data* 5, 2 (2017), 153–163. https://doi.org/10.1089/big.2016.0047 PMID: 28632438.

[8] Pieter Delobelle, Paul Temple, Gilles Perrouin, Benoît Frénay, Patrick Heymans, and Bettina Berendt. 2021. Ethical adversaries: Towards mitigating unfairness with adversarial machine learning. *ACM SIGKDD Explorations Newsletter* 23, 1 (2021), 32–41.

[9] Richard A DeMillo, Richard J Lipton, and Frederick G Sayward. 1978. Hints on test data selection: Help for the practicing programmer. *Computer* 11, 4 (1978), 34–41.

[10] 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

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

[12] 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_files/paper/2016/file/9d2682367c3935defcb1f9e247a97c0d-Paper.pdf

[13] Germain Herbay. 2022. Assessing Machine Learning Fairness via Dataset Mutation.

[14] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea: A Model Behaviour Mutation Approach to Benchmarking Bias Mitigation Methods. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 994–1006. https://doi.org/10.1145/3468264.3468565

[15] Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. 2021. DeepCrime: Mutation Testing of Deep Learning Systems Based on Real Faults. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (Virtual, Denmark) *(ISSTA 2021)*. Association for Computing Machinery, New York, NY, USA, 67–78. https://doi.org/10.1145/3460319.3464825

[16] Lauren Kirchner Jeff Larson, Surya Mattu and Julia Angwin. 2016. How We Analyzed the COMPAS Recidivism Algorithm. Retrieved May 23, 2016 from https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm

[17] Brendan F. Klare, Mark J. Burge, Joshua C. Klontz, Richard W. Vorder Bruegge, and Anil K. Jain. 2012. Face Recognition Performance: Role of Demographic Information. *IEEE Transactions on Information Forensics and Security* 7, 6 (2012), 1789–1801. https://doi.org/10.1109/TIFS.2012.2214212

[18] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th international symposium on software reliability engineering (ISSRE)*. IEEE, 100–111.

[19] Pingchuan Ma, Shuai Wang, and Jin Liu. 2020. Metamorphic Testing and Certified Mitigation of Fairness Violations in NLP Models.. In *IJCAI*. 458–465.

[20] Annibale Panichella and Cynthia CS Liem. 2021. What are we really testing in mutation testing for machine learning? a critical reflection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 66–70.

[21] Mike Papadakis, Marinos Kintis, Jie Zhang, Yue Jia, Yves Le Traon, and Mark Harman. 2019. Chapter Six - Mutation Testing

Advances: An Analysis and Survey. Advances in Computers, Vol. 112. Elsevier, 275–378. https://doi.org/10.1016/bs.adcom.2018.03.015

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Emma Pierson, Sam Corbett-Davies, and Sharad Goel. 2018. Fast Threshold Tests for Detecting Discrimination. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 84)*, Amos Storkey and Fernando Perez-Cruz (Eds.). PMLR, 96–105. https://proceedings.mlr.press/v84/pierson18a.html

[24] Vincenzo Riccio, Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. 2021. DeepMetis: Augmenting a Deep Learning Test Set to Increase its Mutation Score. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 355–367. https://doi.org/10.1109/ASE51524.2021.9678764

[25] Ezekiel Soremekun, Mike Papadakis, Maxime Cordy, and Yves Le Traon. 2022. Software Fairness: An Analysis and Survey. arXiv:2205.08809 [cs.SE]

[26] Zhou Yang, Muhammad Hilmi Asyrofi, and David Lo. 2021. BiasRV: Uncovering Biased Sentiment Predictions at Runtime. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 1540–1544. https://doi.org/10.1145/3468264.3473117

[27] 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