Approaches for Fairness-Aware Classification: AdaFair, SMOTEBoost, Adaptive Sensitive Reweighting

Viacheslav Martynov [1] **Elena Kan** [1] **Julia Moroz** [1] **Alina Smolina** [1]

Abstract

Fairness-aware classification, which aims to reduce the algorithms' prejudice towards the minorities (e.g. in terms of race, gender or nationality) in various domains such as loan credit and/or job hiring, is a hot topic in the international ML society right now. This problem can in some way be reduced to the imbalanced classification and minority class prediction. In this report different algorithms for imbalanced classification (AdaFair, SMOTEBoost and Adaptive Sensitive Reweighting) were tested for popular real-world datasets.

Github repo: our project github repo link here
Video presentation: our link here

1. Introduction

In parallel to the growing demand for automation using AI, an increased number of concerns regarding accountability, fairness and transparency of such systems, especially for domains of high societal impact, has been raised over the recent years.

There is already plenty of observed incidents regarding discrimination caused by AI-based decision making systems. In Amazon's Prime case, for example, in which AI algorithm's task was to decide which areas of a city are eligible to advanced services, areas mostly inhabited by black people were ignored (racial-bias), even though the algorithm did not consider race as a feature. In another case, it was discovered that Google's AdFisher tool displayed significantly more advertisements of highly paid jobs to men than women (gender-bias).

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Viacheslav Martynov <viacheslav.martynov@skoltech.ru>, Elena Kan <elena.kan@skoltech.ru>, Julia Moroz <yulia.moroz@skoltech.ru>, Alina Smolina <alina.smolina@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

Such incidents point out the urge to consider fairness in AI algorithms in order to exploit the amazing potential of the technology for societal advance (Iosifidis & Ntoutsi, 2019).

Fairness concern formulations compare aspects of a classifier between sensitive and non-sensitive groups. A certain group of samples is recognized as *sensitive* based on a sensitive real-world attribute, such as gender, race or financial status. Bias arises when a statistical property for the distribution of label estimations for sensitive group is different for the distribution of label estimations for non-sensitive group. Fairness-aware classification methods attempt to mitigate such differences.

The main contributions of this report are as follows.

- Replicate and test algorithms for Fairness-aware classification, such as AdaFair and Adaptive Sensitive Reweighting.
- Compare these algorithms with SMOTEBoost which tackles only class imbalance problem.

All the experiments are done using the real-world datasets (Adult Census, KDD Census, Bank, COMPAS).

2. Preliminaries

2.1. Problem statement

The mathematical formulation of the problem is as follows. We assume a dataset D consisting of n i.i.d. samples drawn from a joint distribution $P(F, S, y)$: S denotes sensitive attributes such as gender and race, F denotes other non-sensitive attributes and y is the class label. For simplicity, we consider that the classiffcation problem is binary, that is, $y \in \{+, -\}$ and that there exist a single sensitive attribute $\mathcal{S}$, also binary. That is, $S \in \{\mathcal{S}, \mathcal{S}'\}$ with $\mathcal{S}$ and $\mathcal{S}'$ denoting the sensitive (it is also called *protected*) and non-sensitive (*non-protected*) group, respectively. We use the notation $\mathcal{S}_+(\mathcal{S}_-), \mathcal{S}'_+(\mathcal{S}'_-)$ to denote the protected and non-protected group for the positive (negative, respectively) class. The goal of classification is to find a mapping function $f : (F, S) \rightarrow y$ to predict the class labels of future unseen instance.

2.2. Disparate treatment and disparate impact elimination

As outlined by (Zafar et al., 2017), classification unfairness is often expressed through the notions of disparate impact and disparate mistreatment. There is *disparate impact* when the decision outcomes disproportionately benefit or hurt members of certain sensitive attribute value groups. A decision making process is said to be suffering from *disparate mistreatment* with respect to a given sensitive attribute (e.g., race) if the misclassification rates differ for groups of people having different values of that sensitive attribute (e.g., blacks and whites).

Fairness objectives aim to eliminate these types of unfairness. Thus, *disparate impact elimination* reflects the ability of a classifier to achieve statistical parity, i.e. assign the same portion of users to a class for sensitive and non-sensitive groups. For example, if financial status is a sensitive attribute for term deposit predictions, disparate impact elimination would ensure that the portion of positive predictions is the same between low-income and high-income clients. *Disparate mistreatment elimination* reflects the ability of a classifier to achieve equal misclassification rates across sound ground truth labels (i.e. not suffering from dataset construction problems, such as historical biases). For example, if race is a sensitive attribute for prediction of criminal behavior, disparate mistreatment elimination would ensure the same error rate between white and non-white defendants.

2.3. Metrics

The disparate impact is measured by the p% rule, an empirical rule which does not allow sensitive group identification to be lower then a set percentage of non-sensitive group identification:

$$pRule = \min\left\{\frac{P\left(\hat{y}_i = 1 \mid i \in \mathcal{S}\right)}{P\left(\hat{y}_i = 1 \mid i \notin \mathcal{S}\right)}, \frac{P\left(\hat{y}_i = 1 \mid i \notin \mathcal{S}\right)}{P\left(\hat{y}_i = 1 \mid i \in \mathcal{S}\right)}\right\}$$

The disparate mistreatment is measured by how deviation from set goals differs between the sensitive and non-sensitive group. For that the following metrics are used:

$$D_{FPR} = P\left(\hat{y}_i \neq y_i \mid y_i = 1, i \in \mathcal{S}\right) - P\left(\hat{y}_i \neq y_i \mid y_i = 1, i \notin \mathcal{S}\right)$$
$$D_{FNR} = P\left(\hat{y}_i \neq y_i \mid y_i = 0, i \in \mathcal{S}\right) - P\left(\hat{y}_i \neq y_i \mid y_i = 0, i \notin \mathcal{S}\right)$$

To estimate the overall disparate mistreatment this two metrics are combined as sum: $Eq.odds = |D_{FPR}| + |D_{FNR}|$.

3. Related work

Approaches towards fairness classification can be pre-, in-, and post-processing. First methods are typically model-agnostic and therefore, any classifier is applicable after the pre-processing phase. Massaging (Kamiran & Calders, 2009) and re-weighting (Calders et al., 2009) change data distributions directly. In-processing method (Zafar et al., 2017), inserts a set of convex-concave constraints, which aim to minimize equalized odds, into logistic regression. Example of post-processing method, (Fish et al., 2016) shifts the decision boundary of AdaBoost to minimize discrimination.

4. Algorithms and models

4.1. AdaFair

AdaFair, proposed in (Iosifidis & Ntoutsi, 2019), a fairness-aware boosting method that achieves parity between the two groups (thus achieving fairness) while maintaining high TPRs and TNRs (thus tackling class imbalance).

4.1.1. Algorithm description

AdaFair is based on AdaBoost and extends its instance weighting strategy for each round based on the thus far observed ensemble fairness. In each round, the weak learner focuses on both hard classification examples and on the discriminated group per class. The discriminated group, per class, is identified dynamically at each boosting round and its effect on the instance weighting is evaluated based on a cumulative notion of fairness u_i that considers the fairness behavior of the thus far built ensemble model in the particular round. Finally, at the end of the training phase, we select the best sequence of weak learners which achieves high performance and fairness. Pseudo-code, from (Iosifidis & Ntoutsi, 2019), for training section is described in Algorithm 1.

In the beginning, instance weights w and costs u are initialized. Next, a weak learner is trained upon a given weight distribution while the error rate, α_j, $\delta FNR^{1:j}$, $\delta FPR^{1:j}$, and u_i are computed for the current round. Formulas for their calculation are given in the next section. After u is computed, instance weights are estimated and normalized by a factor Z.

4.1.2. Cumulative fairness costs

Let $j : 1-T$ be the current boosting round, where T is a user defined parameter indicating the number of boosting rounds. Let $H_{1:j}(x) = \sum_{i=1}^{j} \alpha_i h_i(x)$ be the ensemble model up to round j. The cumulative boosting fairness of the model $H_{1:j}(x)$ at round j is defined in terms of $|\delta FPR|$, $|\delta FNR|$ for both protected and non-protected groups, as follows:

$$\delta FNR^{1:j} = \frac{\sum_{i=1}^{|\mathcal{S}'_+|} 1 \cdot I(\sum_{k=1}^{j} \alpha_k h_k(x_i^{\mathcal{S}'_+}) \neq y_i)}{|\mathcal{S}'_+|} - \frac{\sum_{i=1}^{|\mathcal{S}_+|} 1 \cdot I(\sum_{k=1}^{j} \alpha_k h_k(x_i^{\mathcal{S}_+}) \neq y_i)}{|\mathcal{S}_+|}$$

$$\delta FPR^{1:j} = \frac{\sum_{i=1}^{|\mathcal{S}'_-|} 1 \cdot I(\sum_{k=1}^{j} \alpha_k h_k(x_i^{\mathcal{S}'_-}) \neq y_i)}{|\mathcal{S}'_-|} - \frac{\sum_{i=1}^{|\mathcal{S}_-|} 1 \cdot I(\sum_{k=1}^{j} \alpha_k h_k(x_i^{\mathcal{S}_-}) \neq y_i)}{|\mathcal{S}_-|}$$

The fairness related costs are dynamically estimated in each round. In particular, the fairness related cost u_i for an

Algorithm 1 AdaFair

Input $D = (x_i, y_1)_1^N, T, \epsilon$
(1) Initialize $w_i = 1/N$ and $u_i = 0$, for $i = 1, 2, \ldots, N$
for $j = 1, 2, \ldots, T$ **do**
Train a classifier h_j to the training data using weights w_i
Compute the error rate $err_j = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} w_i}$.
Compute the class weight $\alpha_j = \frac{1}{2} \cdot ln \frac{1-err_j}{err_j}$.
Compute fairness-related $\delta FNR^{1:j}$
Compute fairness-related $\delta FPR^{1:j}$
Compute fairness-related costs u_i
Update distribution as $w_i \leftarrow \frac{1}{Z_j} w_i \cdot e^{\alpha_j \cdot \hat{h_i} \cdot I(y_i \neq h_j(x_i))} \cdot (1 + u_i)$. // Z_i is the normalization coefficient; $\hat{h_j}$ is the confidence score.
end for
Output hypothesis: $H = \sum_{j=1}^{T} \alpha_i \cdot h_j(x)$

instance x_i in the boosting round j is computed as follows:

$$\mathrm{u_i} = \begin{cases} |\delta FNR^{1:j}|, & \text{if } I\left((y_i \neq h_j(x_i)) \wedge |\delta FNR^{1:j}| > \epsilon\right), x_i \in s_+, \delta FNR^{1:j} > 0 \\ |\delta FNR^{1:j}|, & \text{if } I\left((y_i \neq h_j(x_i)) \wedge |\delta FNR^{1:j}| > \epsilon\right), x_i \in \bar{s}_+, \delta FNR^{1:j} < 0 \\ |\delta FPR^{1:j}|, & \text{if } I\left((y_i \neq h_j(x_i)) \wedge |\delta FPR^{1:j}| > \epsilon\right), x_i \in s_-, \delta FPR^{1:j} > 0 \\ |\delta FPR^{1:j}|, & \text{if } I\left((y_i \neq h_j(x_i)) \wedge |\delta FPR^{1:j}| > \epsilon\right), x_i \in \bar{s}_-, \delta FPR^{1:j} < 0 \\ 0, & \text{otherwise} \end{cases}$$

4.1.3. Optimization of hyperparameters

AdaFair requires optimizing number of weak learners and their maximum depth. This is done by manual grid search, where n varies from 5 to 20, and $depth$ from 2 to 5. Also, we decided not to introduce a strict mathematical formulation for AdaFair optimal hyperparameters, as it's a topic of a different deep research. We look at the following metrics: Accuracy, Balanced Accuracy, Eq. Odds, TPR and TNR for protected and non-protected groups, and comparing them to ones, provided by vanilla AdaBoost, choose the best hyperparameters. A model is chosen, if it yields lowest Eq.odds, maintaining a good balance in TPR and TNR (precisely, increasing TPR of a protected group, maintaining high TPR of non-protected group and high TNR for both groups). Figures 1 and 2 show, how these metrics may change for a particular case.

4.2. SMOTEBoost

SMOTEBoost is a novel approach for learning from imbalanced data sets, based on a combination of the SMOTE algorithm and the boosting procedure (Chawla et al., 2003). This algorithm creates synthetic examples from the rare or minority class, thus indirectly changing the updating weights and compensating for skewed distributions. SMOTEBoost applied to several highly and moderately imbalanced data sets show improvement in prediction performance on the minority class and overall improved F-values.

4.2.1. SMOTE

SMOTE (Synthetic Minority Oversampling Technique) algorithm was used for eliminating the imbalanced dataset. SMOTE creates synthetic instances of the minority class by operating in the "feature space" rather than the "data space". This algorithm synthesizes new minority instances between existing minority ones. In the nearest neighbor computations for the minority classes we use Euclidean distance for the continuous features The new synthetic minority samples for the continuous features are created as follows:

1. Calculate the difference between a feature vector and one of its k nearest neighbors.
2. Multiply this difference by a random number between 0 and 1.
3. Add this difference to the feature value of the original feature vector, thus creating a new feature vector.

4.2.2. Algorithm description

We only SMOTE for the minority class examples in the distribution D_t at the iteration t. This has an implicit effect of increasing the sampling weights of minority class cases, as new examples are created in D_t. The synthetically created minority class cases are discarded after learning a classifier at iteration t. So new examples are constructed in each iteration t by sampling from D_t. The error-estimate after each boosting iteration is on the original training set. Thus, we try to maximize the skewed class dataset's margin by adding new minority class cases before learning a classifier in a boosting iteration.

The proposed SMOTEBoost algorithm, shown in Algorithm 2, proceeds in a series of T rounds. A weak learning algorithm is called and presented with a different distribution D_t altered by emphasizing particular training examples in every round. The distribution is updated to give wrong classifications higher weights than correct classifications. Unlike standard boosting, where the distribution D_t is updated uniformly, for example, from both the majority and minority classes, in the SMOTEBoost technique, the distribution D_t is updated. The examples from the minority class are oversampled by creating synthetic minority class examples. The entire weighted training set is given to the weak learner to compute the weak hypothesis h_t. In the end, the different hypotheses are combined into a final hypothesis h_{fn}.

4.2.3. Optimization of hyperparameters

The SMOTE parameter is the number of synthetic examples to create. This parameter can vary for each dataset. It would be helpful to know a priori the number of SMOTE that need to be entered for each dataset. For example, consider the

Algorithm 2 SMOTEBoost

Input Set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$, where $C_p, (C_p < C)$ corresponds to a minority (positive) class.
Let $B = \{(i, y) : i = 1, \ldots, m, y \neq y_i\}$.
Initialize $D_1(i, y) = 1/|B|$ for $(i, y) \in B$.
for $t = 1, 2, \ldots, T$ **do**
Modify distribution D_t by creating N synthetic examples from minority class C_p using the SMOTE algorithm.
Train a weak learner using distribution D_t.
Compute weak hypothesis $h_t : X \times Y \to [0, 1]$.
Compute the pseudo-loss of hypothesis h_t:
$\epsilon_t = \sum_{(i,y)\in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$.
Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$ and $w_t = (1/2) \cdot (1 - h_t(x_i, y) + h_t(x_i, y))$.
Update D_t: $D_{t+1}(i, y) = (D_t(i, y)/Z_t) \cdot \beta_t^{w_t}$
end for
Output the final hypothesis:

$$h_{fin} = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) \cdot h_t(x, y)$$

Table 1. Final values for *recall*, *precision* and *F-value* for minority class for *mammography* data set.

SMOTE	Recall	Precision	F-value
100	0.72	0.72	**0.72**
200	0.56	0.7	0.63
300	0.59	0.76	0.67
500	0.6	0.78	0.68

Mammography dataset (Woods, 1993). There are 10923 examples in the majority class and 260 examples in the minority class. Different values for the SMOTE parameter, ranging between 100 and 500, were used for the minority class. Next, for the best SMOTE parameter, we will plot the dependence of f-value on the number of boosting rounds. The experimental results for data set are presented in Table 1 and Figure 3. Analyzing Figure 3, it is apparent that SMOTEBoost achieved higher F-values than the AdaBoost. The implementation of SMOTEBoost for other datasets is available in our GitHub repository.

4.3. Adaptive Sensitive Reweighting

In the paper (Krasanakis et al., 2018) Adaptive Sensitive Reweightening (ASR) is introduced as a process that iteratively adapts training sample weights with a theoretically grounded model, and it allows to better achieve fairness objectives, such as trade-offs between accuracy and disparate impact elimination or disparate mistreatment elimination.

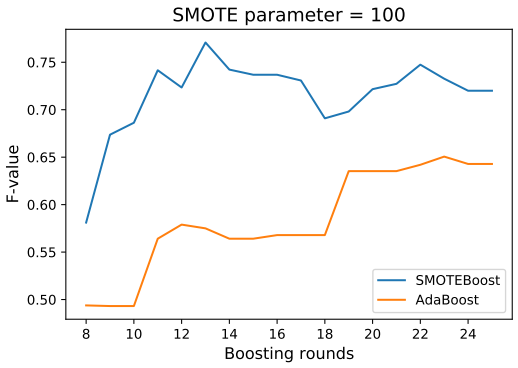

Figure 1. F-values for the minority class for *mammography* data set

4.3.1. Algorithm description

The general formulation for different fairness goals on a classifier is as follows. For training samples i, features x_i and class labels y_i there exist underlying (i.e. observable) class labels $\tilde{y_i}$ that yields estimated labels $\hat{y_i}$ which conform to designated fairness and accuracy trade-offs. The training process has two goals simultaneously: 1) make the classifier yield accurate predictions, i.e. $\min \hat{P}(\hat{y_i} \neq y_i)$ and 2) make the classifier predictions approach the underlying labels, i.e. $\min \hat{P}(\hat{y_i} \neq \tilde{y_i})$. However, estimating the underlying labels and directly using them for training can be argued to be an act of data falsification under certain legal settings. The authors (Krasanakis et al., 2018) proposed selecting weights w_i for training samples i that make weighted training on data labels equivalent to unweighted training on underlying labels:

$$\min \textstyle\sum_i w_i \hat{P}(\hat{y_i} \neq y_i)$$

$$w_i \hat{P}(\hat{y_i} \neq y_i) = \hat{P}(\hat{y_i} \neq \tilde{y_i}) \quad \forall i$$

The next step is to adequately estimate $\hat{P}(\hat{y_i} \neq \tilde{y_i})$ using only $\hat{P}(\hat{y_i} \neq y_i)$, or in other words, express w_i through $\hat{P}(\hat{y_i} \neq y_i)$. For that, the following Convex Underlying Label Error Perturbation (CULEP) model was proposed:

$$w_i = \alpha_i L_{\beta_i}\left(\hat{P}(\hat{y_i} \neq y_i)\right) + (1 - \alpha_i) L_{\beta_i}\left(-\hat{P}(\hat{y_i} \neq y_i)\right)$$

$$\beta_i = \left\{ \begin{array}{ll} \beta_{\mathcal{S}} & \text{if } i \in \mathcal{S} \\ \beta_{\mathcal{S}'} & \text{if } i \notin \mathcal{S} \end{array} \right\} \geq 0, \quad \alpha_i = \left\{ \begin{array}{ll} \alpha_{\mathcal{S}} & \text{if } i \in \mathcal{S} \\ \alpha_{\mathcal{S}'} & \text{if } i \notin \mathcal{S} \end{array} \right\} \in [0, 1]$$

As function $L_{\beta_i}(p)$ it was proposed to use an exponential function: $L_{\beta_i}(p) = e^{\beta_i p}$. The detailed theoretical reasoning behind this CULEP model could be found in the full paper (Krasanakis et al., 2018). The CULEP model allows to select parameters $\alpha_{\mathcal{S}}, \alpha_{\mathcal{S}'}, \beta_{\mathcal{S}}, \beta_{\mathcal{S}'}$ such that the classifier is trained toward accuracy, disparate impact elimination and disparate mistreatment elimination objectives.

Algorithm 3 Adaptive Sensitive Reweighting

function Reweight(classifier C, data D, indexes of the sensitive group S)
$w_i \leftarrow 1 \forall i \in D$
$w_{i,prev} \leftarrow 0 \forall i \in D$
while $\sum_{i \in D} \left(w_i - w_{i,prev}\right)^2 \geq \epsilon$ **do**
train C on samples $i \in D$ and weights $\frac{w_i}{\sum_{j \in D} w_j}$
use C to obtain $\hat{P}\left(\hat{y}_i \neq y_i\right)$
estimate $\hat{P}\left(\hat{y}_i \neq \tilde{y}_i\right)$ using $\hat{P}\left(\hat{y}_i \neq y_i\right)$
$w_{i,prev} \leftarrow w_i \forall i \in D$
$w_i \leftarrow \hat{P}\left(\hat{y}_i \neq \tilde{y}_i\right) / \hat{P}\left(\hat{y}_i \neq y_i\right) \forall i \in D$
end while
return trained classifier $C, \{w_i\}$

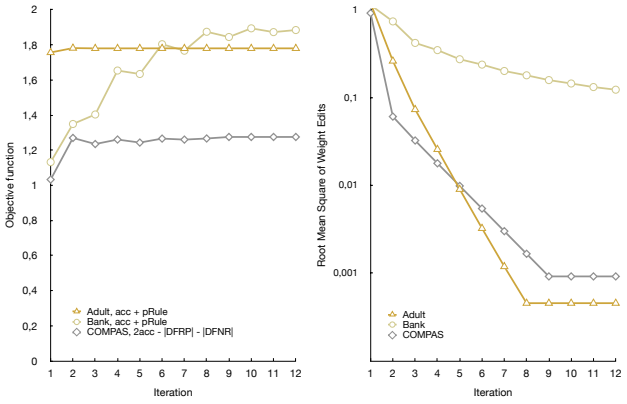

Figure 2. Objective function and weight editing across datasets for each iteration of Adaptive Sensitive Reweighting algorithm using trained CULEP parameters

4.3.2. Fairness Objectives

The classifier performance is measured by accuracy, i.e. the proportion of correctly classified samples. The disparate impact is measured by the p% rule, the disparate mistreatment is measured by sum $|D_{FPR}| + |D_{FNR}|$.

If a dataset (as Adult and Bank datasets from 5.1) suffers from disparate impact, then the classifier could be trained towards eliminating disparate impact while preserving accuracy, and the optimization process aims to $\max\left(acc + pRule\right)$. If a dataset suffers from disparate mistreatment (as the COMPAS dataset described in 5.1), then the classifier's accuracy could be considered equally important to each fairness constraints, i.e. the fairness objective for this case is $\max\left(2acc - |D_{FPR}| - |D_{FNR}|\right)$.

4.3.3. Training process

The whole training process is as follows. Four CULEP (hyper)parameters $\alpha_{\mathcal{S}}, \alpha_{\mathcal{S}'}, \beta_{\mathcal{S}}, \beta_{\mathcal{S}'}$ are optimised such that the classifier is trained towards chosen fairness objective. For that, the optimal parameters are searched in the space $(\alpha_{\mathcal{S}}, \alpha_{\mathcal{S}'}, \beta_{\mathcal{S}}, \beta_{\mathcal{S}'}) \in [0, 1]^2 \times [0, 3]^2$. Each combination of parameters is evaluated with a full run of Algorithm 3 on the training set. For this optimization the DIvided RECTangles (DIRECT) method was employed (Gablonsky et al., 2001; Finkel, 2003). For that, a scipydirect library was used. It is a python wrapper to the DIRECT algorithm (Mayer et al., 2017). When applying this algorithm to CULEP parameters optimization, the approximate upper bound on objective function evaluations was set to 320. As a base classifier of choice logistic regression with L_2-regularization ($C = 1.0$) was employed.

4.3.4. Validation of the ASR+CULEP approach

To assert the validity of the ASR+CULEP approach three real-world datasets were used: Adult income dataset, Bank marketing dataset and COMPAS dataset (described in 5.1).

The training set is used to tune the CULEP model on Algorithm 3 and then train the base classifier on the training set. The evaluation set is used only to calculate the accuracy and disparate impact and mistreatment elimination of the resulting classifier. For robustness, this process was repeated 5 times and the average measures across experiments were reported. The implementation of Adaptive Sensitive Reweighting using CULEP model is available in our GitHub repository.

4.3.5. Exploring convergence

To explore the convergence after selecting CULEP model parameters, the objective functions formulated in 4.3.2 were measured for each dataset on training data. Besides, the root mean square weight edits on each iteration of Algorithm 3 was measured: $\sqrt{\frac{1}{N}\sum_i \left(w_i - w_{i,prev}\right)^2}$, where N is a number of training examples.

In Figure 2 we can see that ASR retrains the base classifier only a few times ~ 5 before converging. However, four CULEP parameters should be optimised, which leads up to $[\log_2\left(5/0.00005\right)]2^4 = 320$ evaluations of ASR by DIRECT method. Although, this could be computationally costly for more complex classifiers, this method is still suited for simpler classifiers, such as logistic regression. Practically, one experiment can take from 30 minutes (for COMPAS dataset) to 160 minutes (for Bank dataset).

4.3.6. Results for Disparate Impact and Disparate Mistreatment Elimination

The average metrics values across five experiments are shown in the diagram 3. As we can see, ASR method with CULEP model allows eliminate both disparate impact (Adult and Bank datasets) and disparate mistreatment (COMPAS dataset) for a small accuracy trade-off.

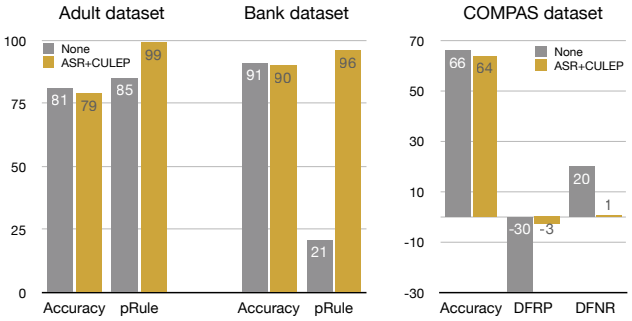

Figure 3. Comparison of the chosen metrics with and without ASR algorithm and CULEP model for logistic regression. On the left: Adult and Bank datasets disparate impact elimination. On the right: COMPAS dataset overall disparate mistreatment elimination (on both $|D_{FNR}|$ and $|D_{FPR}|$ constraints).

ASR+CULEP is able to achieve higher pRule for a slight accuracy loss, and compared to plain Logistic Regression, it attains 14% pRule gain for 2% accuracy loss on Adult dataset, and 75% pRule gain for only 1% accuracy loss on Bank dataset. When avoiding disparate mistreatment, as in the case of COMPAS dataset, ASR+CULEP reduces overall mistreatment by 46% in exchange for 2% accuracy.

5. Experiments and results

The main aim of the experiments is to evaluate the predictive performance and fairness-aware algorithms (AdaFair and ASR) vs an approach dealing with imbalance (SMOTEBoost) and vanilla AdaBoost. Upon the predictive behavior, we plot accuracy and balanced accuracy. Same time, for fairness we report on Equalized Odds, TPR and TNR values for both protected and non-protected groups. To measure the disparate impact we also report on pRule.

5.1. Experimental setup and datasets description

This section explores in details the datasets, parameter selection and evaluation. We evaluate our approach on four real-world datasets whose characteristics are summarized in Table 2. As we can see, they vary in cardinality, dimensionality and class imbalance and are therefore an interesting benchmark to assess. All these original datasets could be found in our GitHub repository.

The UCI *Adult income dataset* (Dua & Graff, 2017) comprises demographic features and a binary label indicating whether income is above 50K. For this dataset, the gender was considered as sensitive feature, and females as protected group.

The *Bank marketing dataset* (Moro et al., 2014) comprises samples with 20 features and a binary label indicating whether a client has subscribed to a term deposit. For this dataset, ages less than 25 and more than 60 years were considered as sensitive (protected).

The *COMPAS dataset* (Larson & Angwin, 2016) contains information on prisoners in Broward County and a binary label indicating whether the defendant is rearrested within two years. In this dataset only five features were picked (age category, gender, race, priors count and charge degree). For this dataset, the sensitive attribute is race, and to make it binary only White and Black (protected group) individuals were selected. The UCI *Census-Income (KDD) Data Set* (Dua & Graff, 2017) has the same prediction task as the Adult dataset. We consider females in this dataset as the protected group.

For each dataset, all categorical features were encoded using one-hot scheme, whereas all numeric attributes were normalized by dividing with their mean value. To obtain training and test data, for Adult and Bank dataset experiments a 70 : 30 random split was performed and for the COMPAS and KDD datasets - a 50 : 50 random data split (random seed for each split was fixed for reproducibility).

In the experiments a zero tolerance to discrimination was set. We set the number of boosting rounds and maximum depth of a weak learner for AdaFair to a different number obtained from optimization process. For ASR we also search best parameters at each dataset. For SMOTEBoost, we set N (the number of synthetic instances generated per round) equal to 100.

5.2. Predictive and fairness performance

Figure 4 shows the performance on Adult Census dataset. The best balanced accuracy is achieved by AdaFair. Regarding Eq.Odds (lower values are better), as expected AdaBoost and SMOTEBoost perform worse as they do not consider fairness. ASR and AdaFair achieve best Eq. Odds, as they consider fairness. If we look closer at the values of TPRs and TNRs in every group, we see that AdaFair method achieves a little lower TPRs for both protected and non-protected groups compared to the other approaches, however it gives higher TNR for protected and non-protected groups.It means that the ASR method produce low TNRs and high TPRs, i.e., these methods "reject" more instances of the negative class in order to minimize Eq.Odds (this explains their high

Table 2. An overview of datasets.

Characteristics	Adult	Bank	COMPAS	KDD Census
Instances	48 842	41 188	6 150	199 523
Attributes	14	20	5	41
Sensitive Attr.	Gender	Age	Race	Gender
Positive class	>50K	subscription	recidivism	>50K
Class ratio (+:-)	1:3	1:7.6	1:1.1	1:15.1

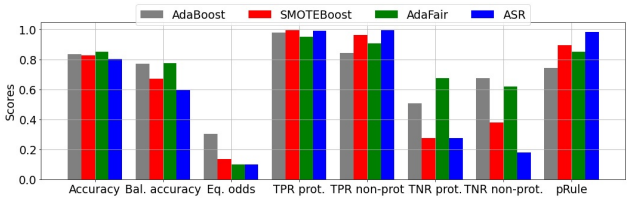

Figure 4. Result of algorithms on Adult Census.

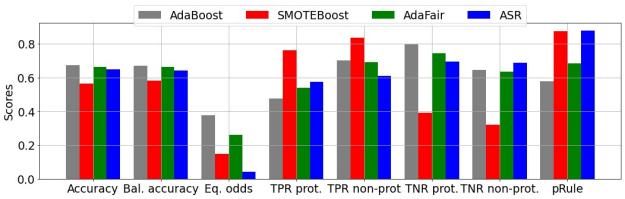

Figure 6. Result of algorithms on Compass.

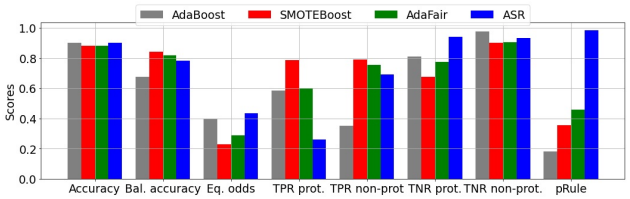

Figure 5. Result of algorithms on Bank.

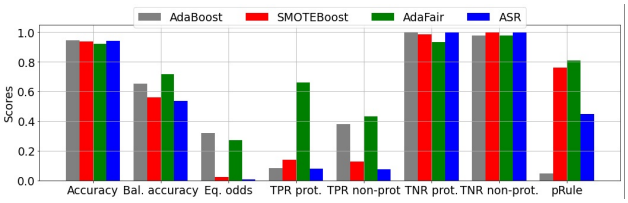

Figure 7. Result of algorithms on KDD Adult Census.

TPRs, low TNRs values). On the contrary, AdaFair achieves good performance for both classes (high TPRs, high TNRs) while maintaining low difference in TPRs, TNRs for both protected and non-protected group.

In Figure 5 - predictive performance for Bank dataset. All of proposed algorithms provide better balanced accuracy, then AdaBoost, however, fairness is increased only by AdaFair. ASR here suffers from "rejecting" instances of positive class, to increase pRule.

Figure 6 depicts prediction regarding Compass dataset. For its classification all models show increased fairness performance, comparing to AdaBoost. Accuracy and balanced accuracy of SMOTEBoost is lower than ones of other model, but it manages with detecting positive class very well. In this example, AdaFair reaches lower Eq. odds value by increasing TPR of protected class, and its TNR.

A very interesting situation can be seen in Figure 7. Predictions KDD Adult Census dataset are fair by ASR and SMOTEBoost, however, TPR for both protected and non-protected groups are very low. High accuracy is achieved by predicting correctly instances from the negative class. AdaFair brings Eq. odds value a little lower than one given by AdaBoost, but looking closer at TPRs and TNRs, we can see that predictions by this alrorithm are really what we expected. It maintains high TNR, and also increases TPR (especially of a protected group).

6. Conclusion

In this work, we have replicated AdaFair, Adaptive Sensitive Reweighting, fairness-aware boosting approaches that adapt AdaBoost and Logistic regression to fairness.

After running experiments at four real-world datasets, we can see substantional difference in Equalized odds between fairness-aware and non-fairness-aware algorithms. We included SMOTEBoost to our experiments and observed, that tackling only class imbalance problem cannot increase fairness always.

ASR method can be applied on multiple types of objectives and can also avoid disparate mistreatment. It achieves similar trade-offs between accuracy and unfairness mitigation metrics, such as pRule or Eq.Odds. The main disadvantage of this method, however, is its time-demanding procedure of searching the CULEP parameters.

In all cases, AdaFair was able to achieve parity among protected and non-protected group and at the same time maintain good classiffication performance compared to other approaches. Moreover, in cases of extreme class-imbalance, AdaFair is able to outperform methods which focus solely on class-imbalance. To increase robustness of the method, one should derive an algorithm for dynamical choosing of the number of weak learners. We already did a step towards this, comparing to algorithm given in (Iosifidis & Ntoutsi, 2019), by computation over the grid of parameters. Next step towards the improvement is to derive a mathematical rule for choosing the best combination of maximum depth and number of weak learners.

References

Calders, T., Kamiran, F., and Pechenizkiy, M. Building classifiers with independency constraints. In *2009 IEEE*

International Conference on Data Mining Workshops, pp. 13–18, 2009. doi: 10.1109/ICDMW.2009.83.

Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pp. 107–119. Springer, 2003.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Finkel, D. Direct optimization algorithm user guide. Technical report, North Carolina State University. Center for Research in Scientific Computation, 2003.

Fish, B., Kun, J., and Lelkes, Á. D. A confidence-based approach for balancing fairness and accuracy. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 144–152. SIAM, 2016.

Gablonsky, J. M. et al. Modifications of the direct algorithm. 2001.

Iosifidis, V. and Ntoutsi, E. Adafair: Cumulative fairness adaptive boosting. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 781–790, 2019.

Kamiran, F. and Calders, T. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*, pp. 1–6, 2009. doi: 10.1109/IC4.2009.4909197.

Krasanakis, E., Spyromitros-Xioufis, E., Papadopoulos, S., and Kompatsiaris, Y. Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. In *Proceedings of the 2018 World Wide Web Conference*, pp. 853–862, 2018.

Larson, J. and Angwin, J. Machine bias, May 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in\-criminal-sentencing.

Mayer, A., Ibai, and Alvi, A. andim/scipydirect: Version 1.3, October 2017. URL https://doi.org/10.5281/zenodo.1014842.

Moro, S., Cortez, P., and Rita, P. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Woods, P. *An Asian experience / Peter Woods ... [et al.]*. Studies of Asia ; 2. Dept. of Education, Queensland, Brisbane, Qld., 1993. ISBN 0724257012.

Zafar, M. B., Valera, I., Gomez Rodriguez, M., and Gummadi, K. P. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*, pp. 1171–1180, 2017.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Julia Moroz (25% of work)

- Reviewing paper (Chawla et al., 2003) on the SMOTEBoost approach.
- Data preprocessing
- Replicating SMOTEBoost algorithm. First stage.
- Presentation creation.

Alina Smolina (25% of work)

- Reviewing paper (Krasanakis et al., 2018) on the Adaptive reweighting classification topic.
- Replicating ASR algorithm.
- Preparing Github Repo
- Data preprocessing

Elena Kan (25% of work)

- Reviewing paper (Chawla et al., 2003) on the SMOTEBoost.
- Data preprocessing
- Replicating SMOTEBoost algorithm

Viacheslav Martynov (25% of work)

- Reviewing paper (Iosifidis & Ntoutsi, 2019) on the AdaFair topic.
- Data preprocessing
- Replicating AdaFair algotithm.
- Preparing code for algorithms' comparison. Giving conclusions on the results.

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: For AdaFair realization was used and modified a function _boost_discrete from sklearn package. More than 90% of own code is written totally.

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: All three algorithms (AdaFair, SMOTEBoost, Adaptive Sensitive Reweighting) and the mathematical setting are included.

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: As it was mentioned, for example in 4.3.4, all these things could be found in our github repo.

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: All info is listed in Table 2

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: As it was mentioned in 5.1, the datasets could be found in our github repo.

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: It described in 5.1

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: It is described in 5.1

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: It is described in 4.1.3, 4.2.3, 4.3.3

9. The exact number of evaluation runs is included.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: It is described in 5.1

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: It described in 2.3

12. Clearly defined error bars are included in the report.

    - ☐ Yes.
    - ☑ No.
    - ☐ Not applicable.

    Students' comment: No comment :(

13. A description of the computing infrastructure used is included in the report.

    - ☐ Yes.
    - ☐ No.
    - ☑ Not applicable.

    Students' comment: None