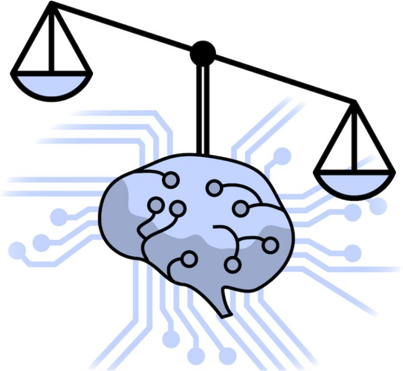
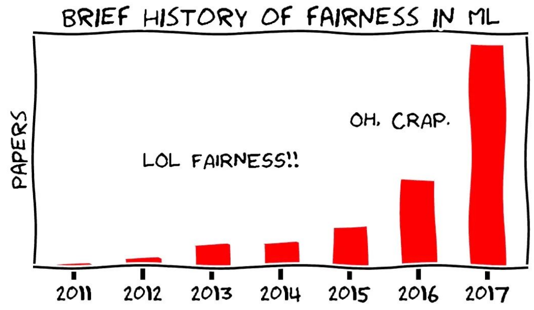


Boosting for Fairness-Aware Classification

Team Caravella:
Ivan Safonov
Nikita Morozov
Ekaterina Fadeeva
Artur Goldman

Problem

- Issue: potential discrimination in ML-based decision making
- Example: Amazon's Prime 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.
- Example: Google's AdFisher tool displayed significantly more advertisements of highly paid jobs to men than women (gender-bias)



Main idea

- Class-imbalance is an inherited problem of fairness.
- Detect and fix the protected class.
- Try to equalize classification metric between protected and unprotected classes.
- Use “equalized odds” metric to control fairness of classification between protected and unprotected classes. The smaller metric, the better performance
- Control high values of TNR’s and TPR’s to avoid case of misclassifying everything
- Incorporate metrics into AdaBoost algorithm and other classification models

$$\delta FPR = P(y \neq \hat{y}|\bar{s}_-) - P(y \neq \hat{y}|s_-) \delta FNR = P(y \neq \hat{y}|\bar{s}_+) - P(y \neq \hat{y}|s_+) \quad (1)$$

$$Eq.Odds = |\delta FPR| + |\delta FNR|$$

SMOTEBoost

Algorithm combines boosting and SMOTE resampling for minority class on each iteration.

- Given: 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_m, ($C_m < C$) corresponds to a minority class.
- Let $B = \{(i, y): i = 1,\ldots,m, y \neq y_i\}$
- Initialize the distribution D_1 over the examples, such that $D_1(i) = 1/m$.
- For $t = 1, 2, 3, 4, \ldots T$
  1. Modify distribution D_t by creating N synthetic examples from minority class C_m using the SMOTE algorithm
  2. Train a weak learner using distribution D_t
  3. Compute weak hypothesis h_t: $X \times Y \rightarrow [0, 1]$
  4. Compute the pseudo-loss of hypothesis h_t:
     $$\varepsilon_t = \sum_{(i,y)\in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$
  5. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2)\cdot(1-h_t(x_i,y)+h_t(x_i,y_i))$
  6. Update D_t : $D_{t+1}(i, y) = (D_t(i,y) / Z_t) \cdot \beta_t^{w_t}$
     where Z_t is a normalization constant chosen such that D_{t+1} is a distribution.
- Output the final hypothesis: $h_{fn} = \arg\max_{y\in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$

Fig. 1. The SMOTEBoost algorithm

RUSBoost

Faster and simpler alternative to SMOTEBoost. Uses undersampling instead of SMOTE resampling.

Algorithm RUSBoost

Given: Set S of examples $(x_1, y_1), ..., (x_m, y_m)$ with minority class $y^r \in Y$, $|Y| = 2$

Weak learner, $WeakLearn$

Number of iterations, T

Desired percentage of total instances to be represented by the minority class, N

1 Initialize $D_1(i) = \frac{1}{m}$ for all i.

2 Do for $t = 1, 2, ..., T$

  a Create temporary training dataset S'_t with distribution D'_t using random undersampling

  b Call $WeakLearn$, providing it with examples S'_t and their weights D'_t.

  c Get back a hypothesis $h_t : X \times Y \to [0, 1]$.

  d Calculate the pseudo-loss (for S and D_t):

  $$\epsilon_t = \sum_{(i,y):y_i \neq y} D_t(i)(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

  e Calculate the weight update parameter:

  $$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t}.$$

  f Update D_t:

  $$D_{t+1}(i) = D_t(i)\alpha_t^{\frac{1}{2}(1+h_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}.$$

  g Normalize D_{t+1}: Let $Z_t = \sum_i D_{t+1}(i)$.

  $$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_t}.$$

3 Output the final hypothesis:

$$H(x) = \underset{y \in Y}{\text{argmax}} \sum_{t=1}^{T} h_t(x, y) \log \frac{1}{\alpha_t}.$$

AdaFair

AdaFair uses AdaBoost approach but iteratively changes the distribution to equally count for minority and non-minority samples.

Algorithm 1 Training phase

Input: $D = (x_i, y_i)_1^N, T, \epsilon$
Output: Ensemble H

(1) Initialize $w_i = 1/N$ and $u_i = 0$, for $i = 1, 2, \ldots, N$
(2) For $j = 1$ to T:
   (a) Train a classifier h_j to the training data using weights w_i.
   (b) Compute the error rate $\mathrm{err}_j = \frac{\sum_{i=1}^{N} w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} w_i}$
   (c) Compute the weight $\alpha_j = \frac{1}{2} \cdot \ln(\frac{1-\mathrm{err}_j}{\mathrm{err}_j})$
   (d) Compute fairness-related $\delta FNR^{1:j}$
   (e) Compute fairness-related $\delta FPR^{1:j}$
   (f) Compute fairness-related costs u_i
   (g) Update the distribution as
   $w_i \leftarrow \frac{1}{Z_j} w_i \cdot e^{\alpha_j \cdot \hat{h}_j(x) \cdot \mathbb{I}(y_i \neq h_j(x_i))} \cdot (1 + u_i)$
   // Z_j is normalization factor; $\hat{h}_j$ is the confidence score
(3) Output $H(x) = \sum_{j=1}^{T} \alpha_i h_j(x)$

Our method: Attentive Gradient Boosting

- We can take any iterative algorithm (consider Gradient Boosting)
- On each iteration we will set weights of objects (on the first iteration 1.0)
- Weights should help the algorithm to make FNR/FPR for protected and non-protected groups closer
- There are different strategies for calculating weights. We used the following simplest. Consider $dFNR = FNR_{protected} - FNR_{non\text{-}protected}$
- If $dFNR > \varepsilon$ we want to make FNR for protected smaller, FNR for non-protected bigger. So let's set weights for (positive, protected) objects 1.0, for (positive, non-protected) objects C ($C < 1.0$). By doing so we ask model to be more accurate on (positive, protected) and less accurate on (positive, non-protected)
- For other possibilities formulas are similar
- For each dataset parameter C can be chosen to have a very good result (we took just $C=0.7$)

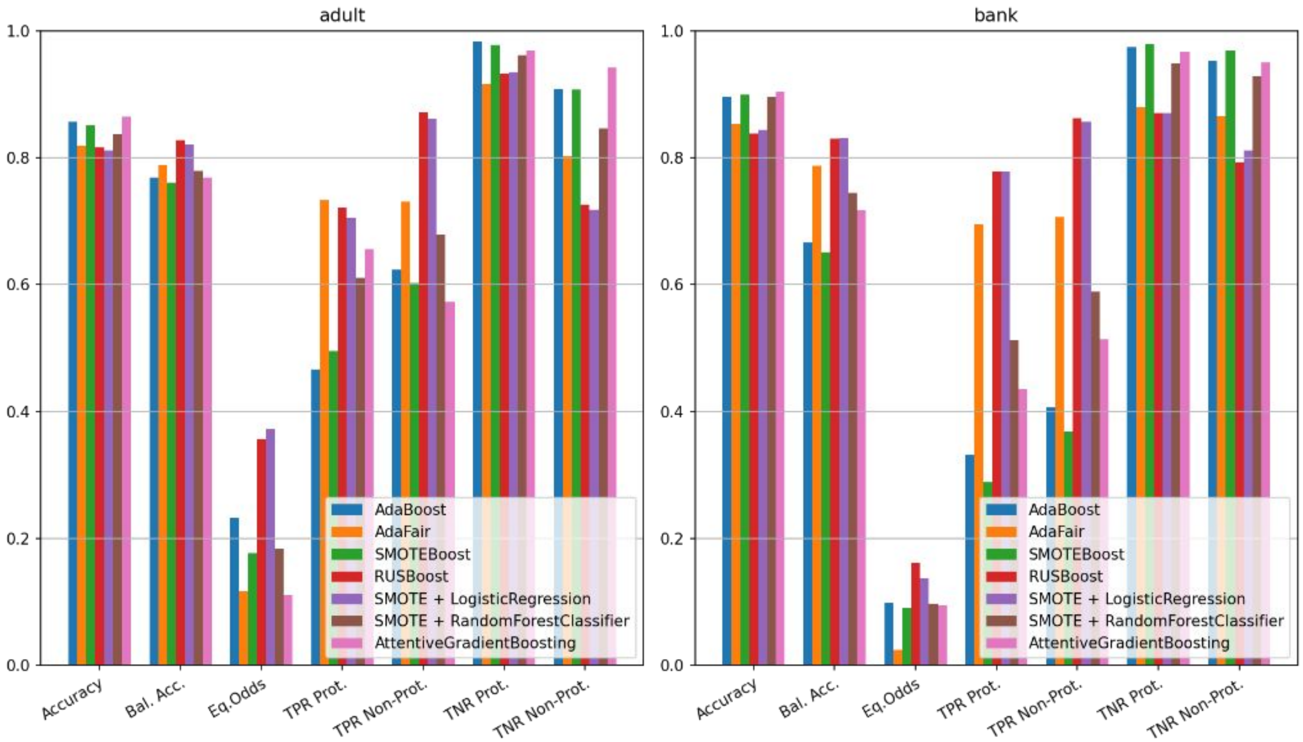
Datasets

	Adult Census	Bank	Compass	KDD Census
#Instances	45,175	40,004	5,278	299,285
#Attributes	14	16	9	41
Sen.Attr.	Gender	Marit. Status	Gender	Gender
Class ratio (+:−)	1:3.03	1:7.57	1:1.12	1:15.11
Positive class	*>50K*	*subscription*	*recidivism*	*>50K*

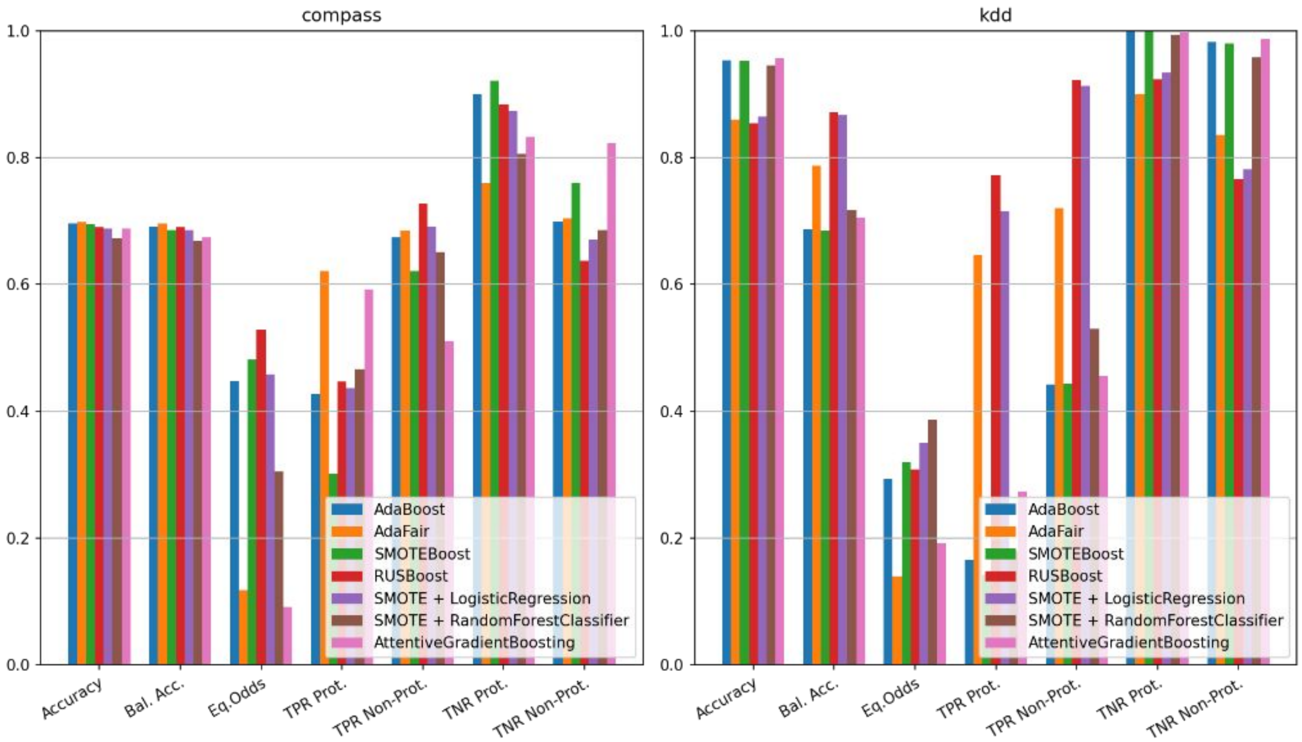
Table 1: An overview of the datasets.

Dataset	Target	Sensitive attribute	Protected group
Adult census income	Whether the annual income of a person will exceed 50K dollars	Gender	Female
KDD census income	-//- (same as previous, but the class labels were drawn from the total person income field rather than the adjusted gross income)	Gender	Female
Bank	Whether a person subscribes to the product (bank term deposit)	Marital status	Married
Compass	Whether a person will be re-arrested within two years (recidivism)	Gender	Female

Experimental results



Experimental results



Experimental results

- `AdaFair` performs better than unaware methods in terms of equalized odds, but can show worse balanced accuracy
- `AdaFair` shows good fairness in the case of both weak (compass) and strong (KDD) data imbalance
- `SMOTEBoost` is more fair (in terms of eq. odds) than `RUSBoost` but has smaller balanced accuracy
- `Attentive Gradient Boosting` is comparable with `AdaFair` and sometimes gets better accuracy and eq. Odds (e.g. both are better on adult dataset). Also hyperparameters can be chosen to give a very small eq. odds and still good accuracy.

References

1) (2019) Vasileios Iosifidis, Eirini Ntoutsi, AdaFair: Cumulative Fairness Adaptive Boosting, (https://arxiv.org/abs/1909.08982)
2) (2003) Chawla et al. SMOTEBoost: Improving Prediction of the Minority Class in Boosting (https://link.springer.com/chapter/10.1007/978-3-540-39804-2_12)
3) (2008) Chris Seiffert; Taghi M. Khoshgoftaar; Jason Van Hulse; Amri Napolitano RUSBoost: Improving classification performance when training data is skewed (https://ieeexplore.ieee.org/document/4761297)

Our team

Ivan Safonov:

Team leader

new algorithm research

Nikita Morozov:

Coding, research

data processing + baselines

Ekaterina Fadeeva:

Coding, research

AdaFair

Artur Goldman:

Coding, research

SMOTEBoost + RUSBoost

GitHub

https://github.com/isaf27/fairness-aware-classification