D. Demmler, D. Krupka, H. Federrath. (Hrsg.): INFORMATIK 2022,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2022 1275

Fairness in Regression – Analysing a Job Candidates Ranking System

Karla Markert * [1] [2], Afrae Ahouzi * [1], Pascal Debus [1]

Abstract: Fairness is one of the pillars of any well-functioning society. Recent law-making in the EU regulates the machine-centered approach and thus increases the necessity for certifiable fairness approaches. In this paper, we adapt previous literature on certifiable fairness in classification systems to a regression model for simplified candidates ranking. This model serves as an illustration for an application that should work fairly even if built upon a biased data set. With our synthetic dataset we are able to analyse the challenges of different fairness notions. Although the fairness training manages to improve the certifiable individual fairness, some of the encoded bias remains. We discuss the challenges we faced, including the selection of suitable parameters and the trade off between accuracy and fairness. We hope to encourage more research into fairness improvement and certification, within and beyond group and individual fairness.

Keywords: bias; fairness; regression; machine learning; candidates ranking

1 Introduction

Over the last couple of years, fuelled by the availability of big data and large scale computational capabilities, artificial intelligence (AI) has continuously pushed the state of the art in a broad variety of application areas, such as computer vision, natural language processing, anomaly detection, robotics, and many more. This immense success comes along with a paradigm shift on how AI and machine learning (ML) algorithms are implemented: deterministic rules set by domain experts are replaced by *data-driven*, end-to-end (deep) learning techniques that are almost universally applicable to any use case as long as enough high-quality data is available.

An AI making a decision instead of a human, does not change the fact that the idea of fairness is deeply rooted in our concepts of societies and participation. Recent law-making in the EU regulates the machine-centered approach. The General Data Protection Regulation (GDPR), for example, enforces accountability for AI for sensitive applications. As a consequence, fair or at least discrimination-free AI becomes a compliance issue for many companies employing AI in sensitive areas with exposure to substantial legal risk. But also beyond

[1] Fraunhofer AISEC, Cognitive Security Technologies, Lichtenbergstraße 11, 85748 Garching, Germany karla.markert@aisec.fraunhofer.de
[2] Technical University Munich, Center for Doctoral Studies in Informatics and its Applications, Boltzmannstraße 3, 85748 Garching, Germany
* The authors contributed equally to this work.

doi:10.18420/inf2022_109

compliance pressure, companies often have their own corporate social responsibility (CSR) agenda, diversity goals or quotas they wish to implement. All these fairness goals need to be adequately implemented for the supporting AI systems while still in accordance with legal requirements [HKH21].

In this paper, we introduce a simple job candidates ranking system that is built upon an artificially generated biased data set. We adapt [Ru20] for this regression task and show the challenges in finding suitable fairness-related parameters to then discuss the trade off between accuracy and fairness with respect to our simple model. This is especially interesting, if the data set comprises a strong bias.

This paper is structured as follows. In Section 2, we present our artificially generated data set and the model used for classification. In Section 3, we provide some theoretical background on the fairness concept we are using. This is followed by a fairness evaluation in Section 4, presenting some challenges we have faced. We finish with a discussion of our results in Section 5.

2 Dataset and Model

In this section, we motivate how we synthetically created our dataset and what requirements made us choose our model's architecture.

2.1 Dataset

Since there are no publicly available datasets with controlled biases, we have decided to create our own dataset. This dataset fulfills two requirements: (1) it allows us to introduce and exclude biases easily and in a controlled way; (2) it allows to rank suitable candidates.

While the idea can be applied to any kind of job, we have decided to limit ourselves to jobs in IT for simplicity. The data is divided into two independent datasets: (1) a training dataset (of 6000 job offers with 60 applicants each) that is split into a training split and a validation split; (2) a test dataset (of 1000 job offers with 10 applicants each) that is used for the fairness evaluation. Each consist of 25% single females, 25% married females, 25% single males, and 25% married males. Each applicant has several attributes: attributes related to their suitability for the job, e.g., their highest degree; and protected attributes not related to their qualification, e.g., their marital status. All categorical features can be one-hot encoded. The work experience is treated as numerical values. For an overview on all the attributes, see Table 3 in the Attachments. The job offers include all but the sensitive attributes as job requirements, see Table 1. It contrasts one sample job offer with one sample candidate. As basis for our evaluation we choose a very basic candidate scoring approach: The job offer has five requirements, of which the candidate fulfills three. Hence, this candidate would be scored $\frac{3}{5}$ on an unbiased scale, where zero corresponds to no match and one corresponds to

a full match. Of course, more elaborate schemes such as weighting of different requirements can be used. This is, however, irrelevant to our approach. In order to distinguish between applicants who just fulfill the requirements and applicants who fulfill all requirements and are even better qualified, we have introduced a bonus of maximum 0.2 for the latter. Hence, the range for the unbiased score is $[0, 1.2]$. We normalize this to fit the interval $[0, 1]$.

		Job Offer	**Candidate 1**
	education	"Bachelor"	"Bachelor"
work experience	back end developer	0	0
	business intelligence analyst	1	0
	cloud engineer	0	0
	data scientist	0	0
	database developer	0	0
	front end developer	0	0
	full-stack developer	0	0
	network system administrator	0	0
	software developer	5	5
	UX developer	0	0
	skills	"C", "Python"	"Python"

Tab. 1: Sample job offer contrasted with a sample candidate. Not required attributes are marked in gray. The job requirements that are not fulfilled are marked in red. As the candidate fulfills three out of five requirements, the respective score would be $\frac{3}{5}$.

For the synthetic discrimination, before normalizing, we add a negative value to the score to synthetically imitate discrimination against certain groups (women or married people), whereas we add a positive value to the score in order to synthetically privilege certain groups (men or single people). For this, we use `truncnorm.rvs()` defined in `scipy.stats`[*] to generate a normally distributed bias that is clipped to a desired value range specified below. With this, we generate a bias b that is added to ("male", "single") or subtracted from ("female", "married") the unbiased score. We use the interval $[0, 0.15]$ for the gender bias and the interval $[0, 0.1]$ for the bias based on the marital status with a scaling factor of 0.5. Hence, women or married people are not worse qualified, they just get a lower scoring for the same qualification.

2.2 Model

In accordance with literature on the Adult[*] dataset [Ko96], we chose a logistic regression model to classify our synthetic data.

[*] For more information, see `https://docs.scipy.org/doc/scipy-0.13.0/reference/generated/scipy.stats.truncnorm.html`.

[*] Available at `https://archive.ics.uci.edu/ml/datasets/Adult`.

In order to analyse fairness, we use our biased data to train models. Based on a job offer and a candidate's description, the model should reliably predict a score. As we deliberately incorporate a bias into our model, the model is supposed to predict the biased score. We expect our model to fulfill the following requirements:

- *Input*: vector of job description concatenated with the candidate's description.
- *Output*: score to evaluate the candidate's suitability for the specific job.
- *Behaviour*: for biased data, we want the model to incorporate the bias.

Our code for generating the data set outputs the concatenated vector containing both the candidate's and the job's features. During pre-processing, the categorical features as well as the skill lists within each job-candidate vector are one-hot encoded and the numerical features are normalized. A schematic drawing of this is displayed in Figure 1.

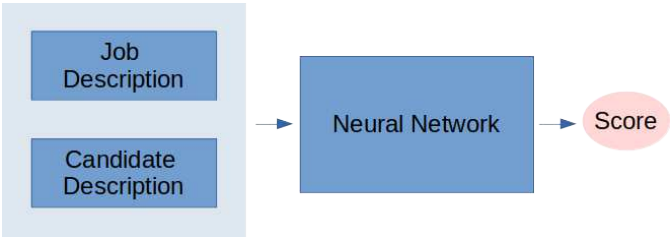


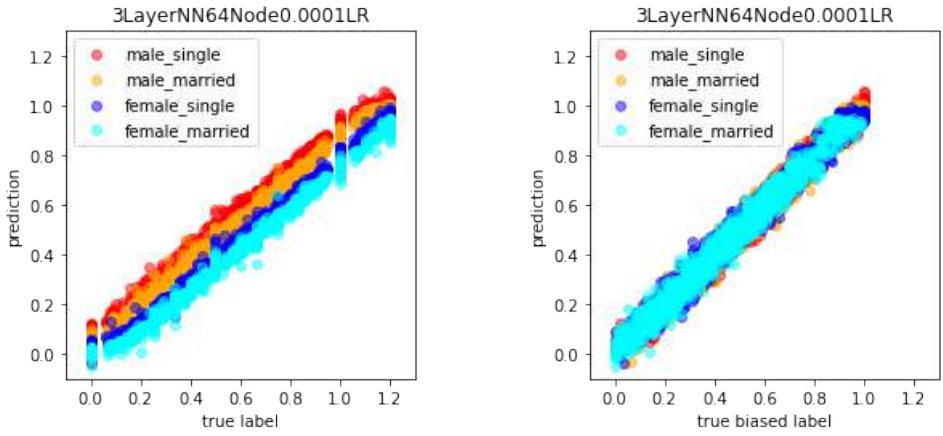
Fig. 1: Combined, the job description and the candidate's description are fed to the neural network. Based on the overlap and a potential artificial bias, the candidate gets assigned a score for its suitability to the job.

In order to find a model that well incorporates the bias, we have tried several different models. How well a model learns to output the unbiased or biased score, depends on the following settings:

- The *number of jobs* and the *number of candidates per job*: If there are too few jobs, the network cannot learn the score properly. For only few jobs and too many candidates per job, the network tends to overfit. Preferably, the number of jobs should be equal to or exceed the number of candidates, so that the network can learn correctly.
- The amount of *nodes per layer*: We found that models with too few nodes per layer were not able to learn the scoring function properly. In our case, a model comprised of two layers with 64 nodes and an output layer of one node delivers the best performance. A model with more nodes per layer or more layers did not provide further improvement.

There are different mechanisms to overcome these challenges, e.g., regularization against overfitting. However, in our most successful models, we did not need to use regularization,

as we could use an adequate model architecture combined with a big and well-adjusted data set of 6000 jobs with 60 candidates each. The model we use for the evaluation has 3 layers: 2 layers with 64 nodes each with RELU activation and an output layer with only one node and no activation. The model has the best MSE error when trained for 180 epochs but we choose to train it for only 30 epochs for the evaluation, as the fairness training takes longer. We use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Further, we set the learning rate to 0.0001. This model has the advantage of learning the data and its underlying bias (see Figure 2), while still being fairly simple.



(a) The bias learnt by the neural network.

(b) The prediction error in the neural network.

Fig. 2: A graphical representation of the bias and the prediction error for our chosen model. The model incorporates the bias from the training set, as male candidates are preferred over female ones and single candidates over married ones.

3 Background and Related Work

In this section, we introduce the fairness concept according to [Ru20] that we adapt for the regression task.

3.1 Certifying Individual Fairness

Ruoss et al. [Ru20] presented a concept to certify individual fairness*. It allows us to find a model M consisting of an encoder f_θ and a classifier h_φ that ensures that two inputs x and x' that are considered similar according to some metric ϕ also achieve similar scores, according to some metric μ:

$$\mathbb{E}_{x\sim D}\left[\forall x' \in \mathbb{R}^n : \phi\left(x, x'\right) \Rightarrow \mu\left(M(x), M(x')\right)\right].$$

* For the corresponding code, see `https://github.com/eth-sri/lcifr`.

They introduce two loss functions, $\mathcal{L}_C$, the classification loss (e.g., cross entropy), and $\mathcal{L}_F$, the fairness loss ensuring that similar inputs x and x' are placed not too far from each other in the latent representation space: $|f_\theta(x) - f_\theta(x')| \leq \delta$. Based on this, they can use local robustness certificates for h_φ to calculate M's fairness.

First, they train the encoder f_θ. In order to make sure that the encoding includes all necessary information, they use both losses: (1) the classification loss for a new classifier q trained on the latent representation; and (2) the fairness loss. With (1) it is assured that the representations are still informative, while (2) structures the data in such a way that similar points are put together, which is important for the later fairness training.

With this at hand, they train the classifier h_φ. Since the classifier takes $f_\theta(x)$ as its input, similar data points are already close to each other. Hence, methods that ensure local robustness also ensure fairness. More concretely, for a data point x, they construct a minimal ϵ ball around $f_\theta(x)$ that contains all similar points. Then, for all points in there, it is checked if they are assigned the same class label. If for every x the data points within the ϵ ball around it are mapped to the same class label, the model is certified to be fair. In this paper, we apply this technique to regression data.

3.2 Adaption for Regression Task

The code corresponding to [Ru20] is designed to suit binary classification problems. Several adaptions have to be included in order to make the code work with deep neural network-based regression tasks. These include (1) adapting the code to work on a deep classifier (the original code did not use any hidden layers) and (2) introducing intervals to define an analogue to same classifications for the certification.

Adapting Fairness Training for a Deep Regression Model The original code trains an encoder in a min-max fashion using the fairness loss $\mathcal{L}_F$ as well as a classification loss $\mathcal{L}_C$ while training its own classifier. Our encoder has an input layer of 55 nodes, a hidden layer and output layer of 30 nodes each. Its classifier has the same architecture as the classifier that is trained later with the latent data from the fully trained encoder.

We imitate that by adapting the encoder to train a regression model with our chosen architecture. Instead of the binary cross entropy loss from the original implementation, we use the mean squared error of the regression model as the regression loss. We adapt the fairness loss for regression, see next Subsection.

Once the encoder is fully trained, we use it to get the latent representation of our data and use that to train the final robust regression model, which can then be certified. Training of this regression model is also done in a min-max fashion, in order to make the model robust to perturbations within the ℓ_∞-ball with radius ϵ. For the fairness training, it is

necessary to choose an appropriate similarity constraint. We define the following constraints for similarity:

$$\phi(x, x') = \bigwedge_{i \in \text{Cat} \setminus \{\text{gender, marital status}\}} (x_i = x'_i) \bigwedge_{j \in \text{Num}} |x_j - x'_j| \leq \alpha_j$$

Hence, the categorical features "education" and "skills" have to match perfectly *for the relevant features*, while the numerical features, i.e., the candidates' qualifications may differ by at most α_j. By choosing $\alpha_j = 0.2 \; \forall j$, we ensure that for similar individuals the numerical features differ by maximally 0.2 after feature normalization. By setting $\delta = 0.01$, we ensure that the distance between the latent representations of two similar data-points is at most 0.01.

Adapting the Certification for a Regression Model During the certification process, we check if the output for points similar to a chosen point p is the same as for the original data point p. As suggested by [Ru20], we determine the output range of each neuron and in particular the output node by transforming input similarity constraints and model architecture into a mixed integer linear programming (MILP) problem and solving it using Gurobi*. Once we have the upper and lower bounds of the output node, we can use them to check if all the similar data points end up in the same class and if that class coincides with the output of our original data point.

In the case of regression, we have to define a similarity notion for similar model outputs. Here, demanding that similar points obtain the exact float prediction is too strict. Instead, we allow the predictions of similar points to our point to differ by a small value ϵ'. This follows the suggestion for the post condition from [Ru20],

$$\mu = \begin{cases} 1, & \text{if } \|M(x) - M(x')\| \leq \epsilon', \\ 0, & \text{otherwise.} \end{cases}$$

If we give more weight to individual fairness, by setting γ higher, we can fit our prediction within this interval. But as we see further below, this can lead our model to output the same prediction for all data-points. To avoid that, we choose our ϵ' to be big enough to fit more predictions and small enough to fulfill the notion of individual fairness. In the code, this implies using the upper and lower bound of the output neuron to determine whether the predictions of all similar points end up within a distance $\epsilon' = 0.1$ of the prediction of the original data points.

4 Fairness Evaluation and Challenges

In the following, we describe the two main challenges we faced when adapting the fairness notions for regression models.

* See `https://www.gurobi.com/`.

Challenge 1: Parameters for Distances In order for our algorithm to work, we have to set several distance parameters: α for the attributes' similarity, δ for the latent representations, and ϵ' for the final output. These parameters have to fit the underlying data set. However, how to choose them precisely is not obvious and highly influences the system's outcome. As explained in Sec. 3.2, we have decided to use the following values: $\alpha = 0.2$, $\delta = 0.01$ and $\epsilon' = 0.1$.

Challenge 2: Trade Off between Accuracy and Fairness In the fairness evaluation, we look at fairness from two different perspectives. Fairness (1): the prediction for point p is fair if it is within an ϵ' distance from the unbiased score for p. Such a score is not available for real world data set, but using it in our set up might give insight on how the individual fairness training impacts the predictions. Fairness (2): the model is fair if it predicts similar outcomes for similar individuals. This is the fairness notion used in the LCIFR code. This type of fairness only relies on the predictions and is therefore easy to check for real world data. Both fairness notions as well as the data set's bias and the prediction's MSE are depicted in Figure 3. The main difference between the two of them is that for Fairness (2) it is in principle possible that all points are assigned the same value. As Fairness (1) compares the predicted values to the unbiased ground truth, assigning all inputs the same output score would not be considered fair in this case. Fairness (1) can only be calculated, if unbiased data is available. This is usually not the case. However, for our synthetic dataset, we can evaluate and compare both fairness notions. This allows us to better understand how the individual fairness training works and how it affects the predictions of the model. We can see that if the bias is too high and we get the prediction closer to the unbiased value, the MSE might increase. However, if we only reduce the unfairness with respect to Fairness (2), then the MSE might still be acceptable, even if the predicted label does not get closer to the unbiased label.

In the following, we experiment with two test data sets. Data set (A) with $b_{\text{gender}} = 0.15$ for gender bias and $b_{\text{marital}} = 0.1$ for marital status bias. Data set (B) with a $b_{\text{gender}} = b_{\text{marital}} = 0.05$ for gender bias and for marital status bias.

To find an appropriate γ, [Ru20] suggests to start with a small value of γ and keep increasing the value, until we find a γ that allows for good fairness results without heavily affecting the models' MSE. The fairness notion that is used in the code corresponds to Fairness (2). We have further introduced a notion that we will refer to as ϵ'-accuracy. Here, if the absolute difference of our model's prediction and the biased target score is maximum ϵ', we consider the prediction as still accurate. The reason for this tolerance is that we hope the fairness training moves the predictions further away from the biased score and towards the unbiased score. In Table 2, we compare the accuracy and fairness results for different values of γ. We notice that by using $\gamma \geq 0.1$, the fairly trained model (with respect to Fairness (2)) tends to output the same prediction for all data points, making the model fair but practically unusable (see the corresponding gray line for $\gamma = 1$ in Table 2a). For $0 \leq \gamma \leq 0.01$, the MSE of the predictions and the biased target score is very low, hence we

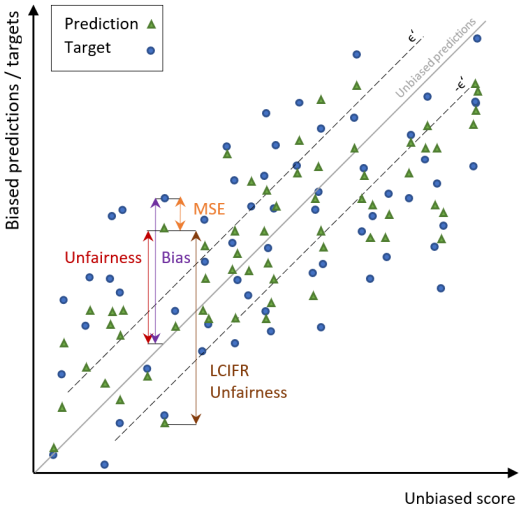


Fig. 3: Comparison of the two fairness notions. All points on the bisecting line and within an ϵ' distance around it are considered fair, as their unbiased value and their biased label coincide. We can measure fairness in two ways here. **Fairness (1) is denoted in red**: it is optimal, if the prediction of an input is equal to its unbiased label. Hence, it measures the difference between the prediction (green) and its unbiased target (bisecting line). **Fairness (2), the LCIFR fairness, is displayed in brown**: it is optimal if all the predictions for similar points are equal. Hence, it is the maximal distance between two predictions (green) corresponding to similar unbiased scores. We can see that if the bias is more than ϵ' and while optimizing for Fairness (1) the prediction approaches the unbiased value, the MSE might increase. However, if we only reduce the unfairness with respect to Fairness (2), we might get an acceptable value for the MSE, while the predicted label, however, might not get closer to the unbiased label.

γ	ϵ'-Acc.	Fairn. (1)	Fairn. (2)
0	0.7550	0.6800	0.0000
0.025	0.8030	0.7040	0.0000
0.05	0.7870	0.6920	0.7220
0.07	**0.8100**	**0.7250**	0.8640
0.1	0.8080	0.7150	0.7890
1	0.6930	0.6040	1

(a) Data set (A): b_{gender}0.15 and $b_{\text{marital}} = 0.1$.

γ	ϵ'-Acc.	Fairn. (1)	Fairn. (2)
0	0.8590	0.8300	0.0000
0.025	0.8580	0.8470	0.0000
0.05	**0.8690**	**0.8510**	0.4510
0.07	0.8700	0.8470	0.9860
0.1	0.8590	0.8280	1.0000
1	0.7430	0.7030	1.0000

(b) Data set (B): $b_{\text{gender}} = b_{\text{marital}} = 0.05$.

Tab. 2: The parameter γ defines the trade off between accuracy (how well does the model predict the provided labels?) and fairness (how unbiased are the model's predictions?). Finding a good trade off can be challenging, as even a low accuracy might be acceptable if the bias is strong.

have a high accuracy. Further, the Fairness (2) notion increases with a higher value of γ,

so a suitable trade off according to [Ru20] could be $\gamma = 0.1$. Using our knowledge of the underlying data and taking the unbiased labels into account, this is not the optimal value from the list. According to Fairness (1), $\gamma = 0.07$ achieves the best fairness value. Further, increasing γ and, thus, paying more attention to Fairness (2) does not necessarily improve Fairness (1). This is particularly important for a high bias in the data set. Here, we might accept a lower accuracy because we need to overcome the bias in the data set in order to improve Fairness (1). However, achieving this cannot be deduced from just knowing the values for accuracy and Fairness (2), as outlined in the example above.

5 Discussion and Outlook

In this paper, we adapted the fairness certification process presented in [Ru20] to regression data. We also discussed how challenging improving the fairness of a job candidate ranking model can be, even for smaller systems. Some of these challenges include how to define fairness and how to find an appropriate trade off between accuracy and fairness.

Our synthetic dataset enabled us to both control the bias and have a ground truth for the fair outcome. This ground truth is usually unavailable in real world scenarios. Nonetheless, this approach gave us insight into the extent to which individual fairness training can improve the fairness of the model. Such a synthetic dataset can be used to analyze and evaluate further fairness interventions.

Our experiments showed that only improving individual fairness might not fully mitigate the bias that was learned from the data. It further showed that focusing on the certifiability of individual fairness can lead the models to be unusable, as they end up returning the same prediction for the whole dataset. Based on these findings, the work of [Ru20] can be further adapted and experimented with to improve the individual fairness training and certification process.

All in all, our work aims at encouraging more efforts into newer notions and approaches of fairness, within but also beyond the realm of group and individual fairness.

Attachment

Table 3 summarizes the attributes used in the dataset.

	Attribute	Possible Values
	gender	"male", "female"
	marital status	"single", "married"
	education	"no diploma", "Bachelor", "Master", "PhD"
work experience	back end developer	1 – 5
	business intelligence analyst	1 – 5
	cloud engineer	1 – 5
	data scientist	1 – 5
	database developer	1 – 5
	front end developer	1 – 5
	full-stack developer	1 – 5
	network system administrator	1 – 5
	software developer	1 – 5
	UX developer	1 – 5
	skills	"C", "C++", "CSS", "Go", "HTML", "Java", "JavaScript", "MSOffice", "PHP", "Python", "R", "Ruby", "SQL"

Tab. 3: Overview on the attributes for both the jobs and the candidates in the dataset. The attributes in italic, gender and marital status, correspond to the protected values.

Bibliography

[HKH21] Hauer, Marc P; Kevekordes, Johannes; Haeri, Maryam Amir: Legal perspective on possible fairness measures–A legal discussion using the example of hiring decisions. Computer Law & Security Review, 42:105583, 2021.

[Ko96] Kohavi, Ron: Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: KDD. volume 96, pp. 202–207, 1996.

[Ru20] Ruoss, Anian; Balunovic, Mislav; Fischer, Marc; Vechev, Martin: Learning Certified Individually Fair Representations. In: Advances in Neural Information Processing Systems 33. 2020.