fairmodels: A Flexible Tool For Bias Detection, Visualization, And Mitigation

Jakub Wiśniewski, Przemysław Biecek

Abstract Machine learning decision systems are getting omnipresent in our lives. From dating apps to rating loan seekers, algorithms affect both our well-being and future. Typically, however, these systems are not infallible. Moreover, complex predictive models are really eager to learn social biases present in historical data that can lead to increasing discrimination. If we want to create models responsibly then we need tools for in-depth validation of models also from the perspective of potential discrimination. This article introduces an R package **fairmodels** that helps to validate fairness and eliminate bias in classification models in an easy and flexible fashion. The **fairmodels** package offers a model-agnostic approach to bias detection, visualization and mitigation. The implemented set of functions and fairness metrics enables model fairness validation from different perspectives. The package includes a series of methods for bias mitigation that aim to diminish the discrimination in the model. The package is designed not only to examine a single model, but also to facilitate comparisons between multiple models.

Introduction

Responsible machine learning and in particular fairness are gaining attention within machine learning community. The reason for this is that predictive algorithms are becoming more and more decisive and influential in our lives. This impact could be less or more significant in areas ranging from user's feed on social platforms, displayed adds and recommendations at an online store to loan decisions, social scoring, and facial recognition systems used by police and authorities. Sometimes it leads to automated systems that learn some undesired bias preserved in data for some historical reason. Whether seeking a job (Lahoti et al., 2019) or having one's data processed by court systems (described in ProPublica in Angwin et al. (2016)), sensitive attributes such as sex, race, religion, ethnicity etc. might play a major role in the decision. Even if such variables are not directly included in the model, they are often captured by proxy variables such as zip code (a proxy for the race and wealth), purchased products (a proxy for gender and age), eye colour (a proxy for ethnicity). As one would expect they are able to give an unfair advantage to a privileged group. Discrimination takes the form of more favorable predictions or higher accuracy for a privileged group. For example, some popular commercial gender classifiers were found to perform the worst on darker females (Buolamwini and Gebru, 2018). From now on such unfair and harmful decisions towards people with specific sensitive attribute will be called biased.

While there are historical, economical or efficiency reasons for this to happen, such decisions are unacceptable for ethical reasons and sometimes due to local law regulations. The problem is not simple, especially when the only criterion set for the system is performance. In some problems we observe a trade-off between accuracy and fairness where lower discrimination, leads to lower performance Kamiran and Calders (2011). Sometimes labels, which are considered ground truth might also be biased (Wick et al. (2019)) and when controlling for that bias the performance and fairness might improve at the same time.

The bias in machine learning systems has potentially many different sources. In Mehrabi et al. (2019) authors categorized bias into its types like historical bias, where unfairness is already embedded into the data reflecting the world, observer bias, sampling bias, ranking and social biases and many more. That shows how many dangers are potentially hidden in the data itself. Whether one would like to act on it or not, it is essential to detect bias and make well-informed decisions which consequences could potentially harm many groups of people. Repercussions of such systems can be unpredictable. As argued by Barocas et al. (2019) machine learning systems are even able to aggravate the disparities between groups, which is called by the authors feedback loops. Sometimes the risk of potential harms resulting from the usage of such systems is high. This was noticed for example by the Council of Europe that wrote the set of guidelines where it states that the usage of facial recognition for the sake of determining persons sex, age, origin, or even emotions should be mostly prohibited (Council of Europe, 2021).

Related Work

Assembling predictive models is nowadays getting easier. Packages like **h2o** (H2O.ai, 2017) provide AutoML frameworks where non-experts can train quickly accurate models without deep domain knowledge. Model validation should also be that simple. Several frameworks have emerged for Python to verify various fairness criteria, the most popular are **aif360** (Bellamy et al., 2019), **fairlearn** (Bird et al., 2020), or **aequitas** (Saleiro et al., 2018). They have various features for detection, visualization and mitigation of the bias in machine learning models. For the R language, until recently the only available tool was the **fairness** package which compares various fairness metrics for specified subgroups. The **fairness** package is very helpful, but it lacks some features. For example it does not allow to compare the machine learning models between each other and to aggregate fairness metrics to facilitate the visualization, but most of all it does not give a quick verdict whether a model is fair or not. Package **fairadapt** aims at removing bias from machine learning models by implementing pre-processing procedure described in Plečko and Meinshausen (2019). Our package tries to combine the detection and mitigation processes. It encourages the user to experiment with the bias, try different mitigation methods and compare results. The package **fairmodels** not only allows for that comparison between models and multiple exposed groups of people, but it gives direct feedback if the model is fair or not (more on that in the next section). Our package also equips the user with fairness object that can be tailored to individual needs. If a model does not meet fairness criteria, there are various pre-processing and post-processing bias mitigation algorithms implemented and ready to use. It aims to be a complete tool for dealing with discriminatory models.

In particular, we show how to use this package to address four key questions: How to measure bias? (see section Acceptable amount of bias), How to detect bias? (see section Fairness metrics)m How to visualize bias? (see section Visualizing bias), How to mitigate bias? (see section Bias mitigation).

Measuring and detecting bias

Fairness metrics

Machine learning models just like human-based decisions can be biased against people with certain sensitive attributes which are also called protected groups. They consist of subgroups - people who share the same sensitive attribute, like gender, race or some other feature.

To address this problem we need to first introduce fairness criteria. Following Barocas et al. (2019), we will present these criteria based on the following notation.

- Let $A \in \{a, b, ...\}$ mean protected group and values $A \neq a$ denote membership to unprivileged subgroups while $A = a$ membership to privileged subgroup. To simplify the notation we will treat this as a binary variable, but all results hold if A has larger number of groups.
- Let $Y \in \{0, 1\}$ be a binary label (binary target = binary classification) where 1 is preferred, favorable outcome.
- Let $R \in [0, 1]$ be a probabilistic response of model, and $\hat{Y} = 1$ when $R \geq 0.5$, otherwise $\hat{Y} = 0$.

According to Barocas et al. (2019) most discrimination criteria can be derived as tests that validate following probabilistic definitions:

- Independence, i.e. $R \perp A$,
- Separation, i.e. $R \perp A \mid Y$,
- Sufficiency, i.e. $Y \perp A \mid R$.

Those criteria and their relaxations might be expressed via different metrics based on confusion matrix for certain subgroup. To check if those fairness criteria are addressed we propose checking 5 metrics among privileged group (a) and unprivileged group (b):

- Statistical parity: $P(\hat{Y} = 1|A = a) = P(\hat{Y} = 1|A = b)$.
  Statistical parity (STP) ensures that fractions of assigned positive labels are the same in subgroups. It is equivalent of Independence (Dwork et al., 2012).
- Equal opportunity: $P(\hat{Y} = 1|A = a, Y = 1) = P(\hat{Y} = 1|A = b, Y = 1)$.
  Checks if classifier has equal True Positive Rate (TPR) for each subgroup. It is a relaxation of Separation (Hardt et al., 2016).
- Predictive parity: $P(Y = 1|A = a, \hat{Y} = 1) = P(Y = 1|A = b, \hat{Y} = 1)$
  Measures if model has equal Positive Predictive Value (PPV) for each subgroup. It is relaxation of Sufficiency (Chouldechova, 2016).
- Predictive equality: $P(\hat{Y} = 1|A = a, Y = 0) = P(\hat{Y} = 1|A = b, Y = 0)$.
  Warrants that classifiers have equal False Positive Rate (FPR) for each subgroup. It is relaxation of Separation (Corbett-Davies et al., 2017).

- (Overall) Accuracy equality: $P(\hat{Y} = Y | A = a) = P(\hat{Y} = Y | A = b)$.
  Makes sure that models have the same Accuracy (ACC) for each subgroup. (Berk et al., 2017)

The reader should note that if the classifier passes Equal opportunity and Predictive equality, then it also passes Equalized Odds (Hardt et al., 2016), which is an equivalent to Separation criteria.

While defining the metrics above we assumed that there are only 2 subgroups. This was done to facilitate notation, but there might be more unprivileged subgroups. A perfectly fair model would pass all criteria for each subgroup Barocas et al. (2019).

Acceptable amount of bias

It would be hard for any classifier to maintain exactly the same relations between subgroups. That is why some margins around the perfect agreement are needed. To address this issue, as the default setting we accepted the four-fifths rule (Code of Federal Regulations, 1978) as the benchmark for discrimination rate which states that *"A selection rate for any race, sex, or ethnic group which is less than four-fifths ($\frac{4}{5}$) (or eighty percent) of the rate for the group with the highest rate will generally be regarded by the Federal enforcement agencies as evidence of adverse impact[...]."* The selection rate is originally represented by statistical parity, but we adopted this rule to define acceptable rates between subgroups for all metrics. There are a few caveats to the preceding citation concerning the size of the sample and the boundary itself. Nevertheless, the four-fifths rule is a good guideline to adhere to. In the implementation, this boundary is represented by ε and it is adjustable by the user but the default value will be 0.8. This rule is often used, but in each specific case one should see if the fairness criteria should be set differently.

Let $\varepsilon > 0$ be the acceptable amount of a bias. In this article we would say that the model is not discriminatory for a particular metric if the ratio between every unprivileged $b, c, ...$ and privileged subgroup a is within $(\varepsilon, \frac{1}{\varepsilon})$. The common choice for the epsilon is 0.8, which corresponds to the four-fifths rule. For example, for the metric Statistical Parity (STP), a model would be ε-non-discriminatory for privileged subgroup a if it satisfies

$$\forall_{b \in A \setminus \{a\}} \;\; \varepsilon < STP_{ratio} = \frac{STP_b}{STP_a} < \frac{1}{\varepsilon}. \tag{1}$$

Evaluating fairness

The main function in the **fairmodels** package is `fairness_check`. It returns `fairness_object` which can be either visualized or processed by other functions. This will be further explained in Structure section. When calling `fairness_check` for the first time the 3 arguments are:

1. **explainer** - an object that combines model and data that gives unified interface for predictions. It is a wrapper over a model created with the **DALEX** (Biecek, 2018) package.
2. **protected** - a factor, vector containing sensitive attributes (protected group) . It does not need to be binary. Each level denotes distinct subgroup. The most common examples are gender, race, nationality etc.
3. **privileged** - a character/factor denoting level in protected vector which is suspected to be most privileged one.

Example

In the following example we are using *German Credit Data* dataset (Dua and Graff, 2017). In the dataset, there is information about people like age, sex, purpose, credit amount, etc. For each person there is Risk assessed with taking credit, either good or bad. It will be a target variable.

First, we create a model.

```
data("german")

lm_model <- glm(Risk~., data = german, family = binomial(link = "logit"))
```

Then, create a wrapper that unifies the model interface.

```
library("DALEX")

y_numeric <- as.numeric(german['Risk']) -1
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)
```

Finally, create and plot the fairness checks.

```
library("fairmodels")

fobject <- fairness_check(explainer_lm,
                          protected = german['Sex'],
                          privileged = "male")
plot(fobject)
print(fobject)
```

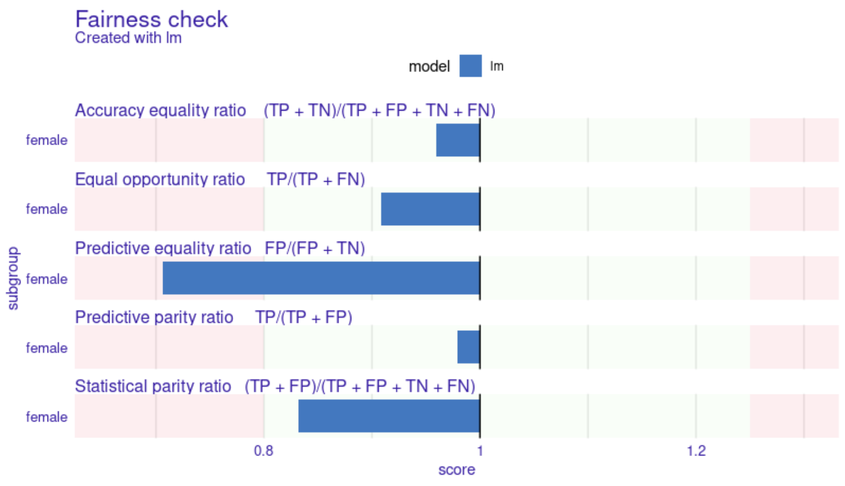
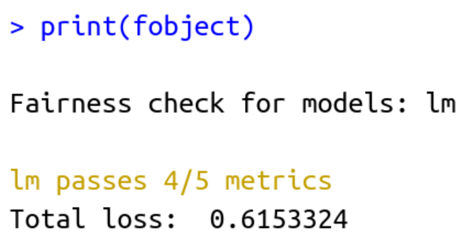


Figure 1: Graphical output for the `fairness_check()`. The default value for ε is set to 0.8. The light green area which are values within $(\varepsilon, \frac{1}{\varepsilon})$ signifies an acceptable difference in fairness metrics. Fairness metric names are given along the formulas used to calculate the score in some subgroup to facilitate interpretation. The ratio here means that after metric scores were calculated, the values for unprivileged groups (here female) were divided by values for the privileged subgroup (here men) as in formula (1).



```
> print(fobject)

Fairness check for models: lm

lm passes 4/5 metrics
Total loss:  0.6153324
```

Figure 2: Numerical output for the `fairness_check()`. Total loss here should be interpreted as the sum of heights of bars in Figure 1

Figure 1 presents the output from the `fairness_check()`. In this example, fairness criteria are satisfied in all but one metric. The logistic regression model has a lower false positive rate (FP/(FP+TN))) in the unprivileged group than in the privileged group. It exceeds acceptable limit set by ε, thus it does not satisfy the Predictive Equality ratio criteria.

For a quick assessment if a model passes fairness criteria `fairness_check()` object might be summarized with the `print()` function as in Figure 2. Total loss is the sum of fairness metrics, see equation 3 for more details.

It is rare that a model perfectly meets all the fairness criteria. Therefore, a very useful feature is the ability to compare several models on the same scale. In the example below, we add two more explainers to the fairness assessment. Now `fairness_object` (in code: `fobject`) wraps three models together with different labels and cutoffs for subgroups. The `fairness_object` can be later used as basis for another `fairness_object`. In detail, running `fairness_check()` for the first time explainer/explainers have to be provided along with three arguments described at the start of this section. As shown below, when providing explainers with `fairness_object`, those arguments are not necessary as they are already part of the object.

http://fairmodels.drwhy.ai/

First, let us create two more models based on the *German Credit Data*. The first one will be logistic regression model that uses fewer columns and has access to Sex feature. The second is random forest from **ranger** (Wright and Ziegler, 2017). It will be trained on the whole dataset.

```
discriminative_lm_model <- glm(Risk~.,
         data   = german[c("Risk", "Sex","Age", "Checking.account", "Credit.amount")],
         family = binomial(link = "logit"))

library("ranger")
rf_model <- ranger::ranger(Risk ~.,
                           data = german,
                           probability = TRUE,
                           max.depth = 4,
                           seed = 123)
```

These models differ in the way how the predict function works. To unify operations on these models, we need to create **DALEX** explainer objects. The label argument specifies how these models are named on plots.

```
explainer_dlm <- DALEX::explain(discriminative_lm_model,
                                data = german[c("Sex", "Age",
                                                "Checking.account",
                                                "Credit.amount")],
                                y = y_numeric,
                                label = "discriminative_lm")

explainer_rf <- DALEX::explain(rf_model,
                                data = german[,-1],
                                y = y_numeric)
```

Now we are ready to perform an assessment of fairness.

```
fobject <- fairness_check(explainer_rf, explainer_dlm, fobject)
plot(fobject)
print(fobject)
```

When plotted (Figure 3) new bars appear on familiar plane. Those are new metric scores for added models. Figure 4 shows the numerical summary for these three models that is printed into the console.

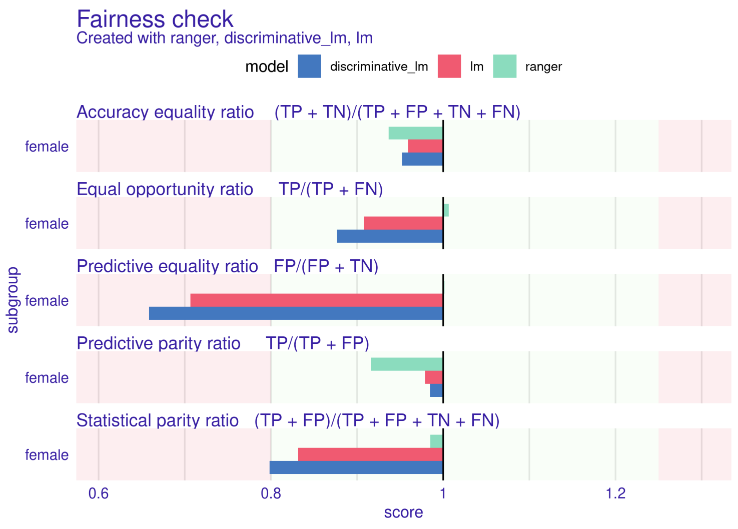
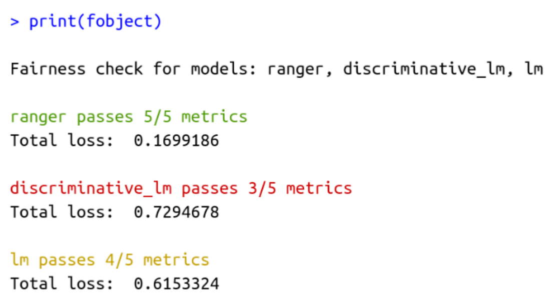


Figure 3: Graphical output for the `fairness_check()` for three models.



```
> print(fobject)

Fairness check for models: ranger, discriminative_lm, lm

ranger passes 5/5 metrics
Total loss:  0.1699186

discriminative_lm passes 3/5 metrics
Total loss:  0.7294678

lm passes 4/5 metrics
Total loss:  0.6153324
```

Figure 4: Numerical summary for the `fairness_check()` for three models.

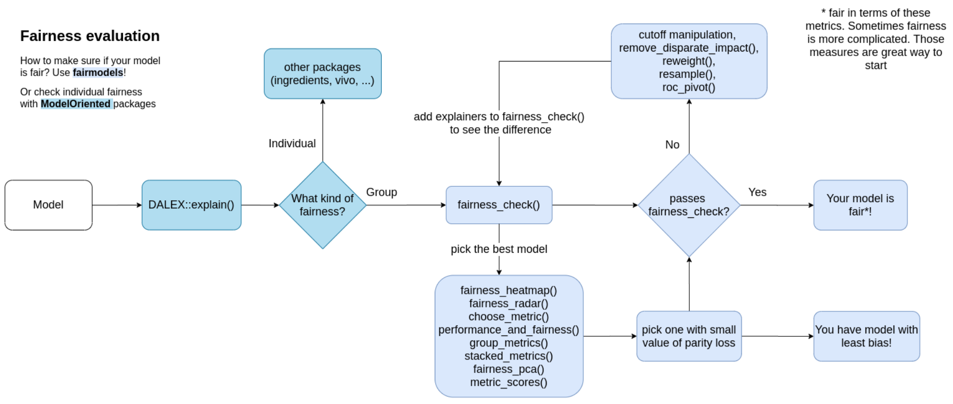

Fairness evaluation
How to make sure if your model is fair? Use fairmodels!
Or check individual fairness with ModelOriented packages
Model
DALEX::explain()
What kind of fairness?
Individual
other packages (ingredients, vivo, ...)
Group
fairness_check()
add explainers to fairness_check() to see the difference
pick the best model
fairness_heatmap() fairness_radar() choose_metric() performance_and_fairness() group_metrics() stacked_metrics() fairness_pca() metric_scores()
passes fairness_check?
No
cutoff manipulation, remove_disparate_impact(), reweight(), resample(), roc_pivot()
Yes
Your model is fair*!
* fair in terms of these metrics. Sometimes fairness is more complicated. Those measures are great way to start
pick one with small value of parity loss
You have model with least bias!

Figure 5: Flowchart for the fairness assessment with the **fairmodels** package

Package Architecture

The **fairmodels** package provides a unified interface for predictive models independently of their internal structure. Using model agnostic approach with **DALEX** explainer facilitates this process (Biecek, 2018). For each explainer, there is a unified way to check if explained model lives up to user fairness standards. Checking fairness with **fairmodels** is straightforward and can be done with three-step pipeline.

classification model %>% explain() %>% fairness_check()

The output of such a pipeline is an object of class `fairness_object` which is a unified structure to wrap model explainer or multiple model explainers and other `fairness_objects` in a single container. Aggregation of fairness measures is done based on groups defined by model labels. This is why model explainers (even those wrapped by `fairness_objects`) must have different labels. Moreover, some visualizations for model comparison assume that all models are created from the same data. Of course, each model can use different variables or use different feature transformations, but the order and amount of rows shall stay the same.

To create `fairness_object`, at least one explainer needs to be passed to `fairness_check()` function which returns the said object. While creating `fairness_object` metrics for all subgroups are calculated from confusion matrices. The `fairness_object` has numerous fields, some of them are:

- **parity_loss_metric_data** - data.frame containing parity loss for each metric and classifier,
- **groups_data** - list of metric scores for each metric and model,
- **group_confusion_matrices** - list of values in confusion matrices for each model and metric,
- **explainers** - list of DALEX explainers. When explainers and/or `fairness_object` are added, then explainers and/or explainers extracted from `fairness_object` are added to that list,
- **label** - character vector of labels for each explainer.
- ... - other fields.

The `fairness_object` methods are used to create numerous objects that help to visualize bias. In next sections we list more detailed functions for deeper exploration of bias. Detailed relations between objects created with **fairmodels** are depicted in Figure 6. The general overview of the workflow is presented in Figure 5.

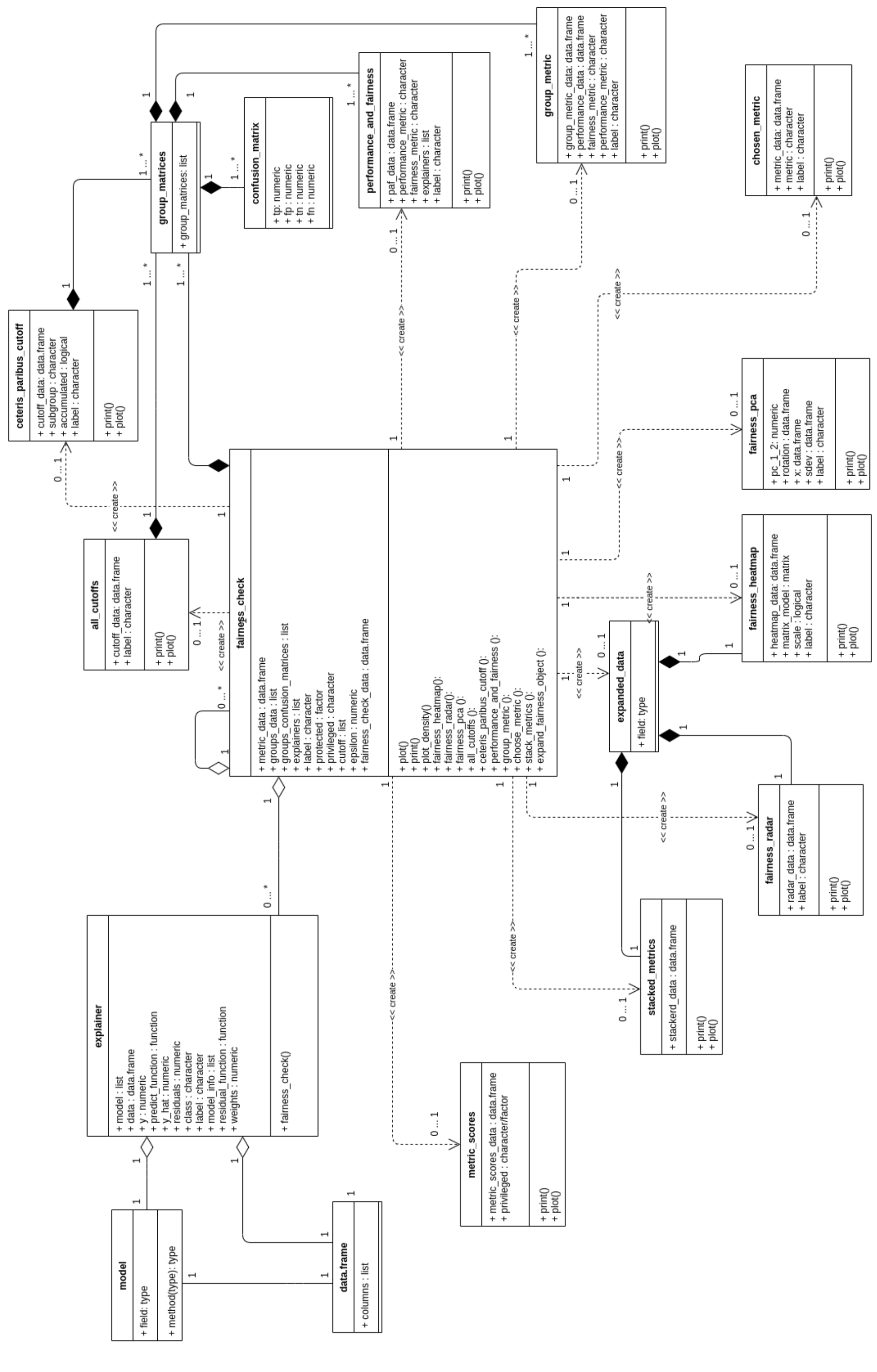


Figure 6: Class diagram for objects created by functions from the **fairmodels** package.

Visualizing bias

In **fairmodels** there are 12 metrics based on confusion matrices for each subgroup, see Table 1 for the complete list. Some of them were already introduced before.

Metric	Formula	Name	Fairness criteria
TPR	$\frac{TP}{TP+FN}$	True positive rate	Equal opportunity (Hardt et al., 2016)
TNR	$\frac{TN}{TN+FP}$	True negative rate	
PPV	$\frac{TP}{TP+FP}$	Positive predictive value	Predictive parity (Chouldechova, 2016)
NPV	$\frac{TN}{TN+FN}$	Negative predictive value	
FNR	$\frac{FN}{FN+TP}$	False negative rate	
FPR	$\frac{FP}{FP+TN}$	False positive rate	Predictive equality (Corbett-Davies et al., 2017)
FDR	$\frac{FP}{FP+TP}$	False discovery rate	
FOR	$\frac{FN}{FN+TN}$	False omission rate	
TS	$\frac{TP}{TP+FN+FP}$	Threat score	
STP	$\frac{TP+FP}{TP+FP+TN+FN}$	Positive rate	Statistical parity (Dwork et al., 2012)
ACC	$\frac{TP+TN}{TP+TN+FP+FN}$	Accuracy	Overall accuracy equality (Berk et al., 2017)
F1	$\frac{2 \cdot PPV * TPR}{PPV+TPR}$	F1 score	

Table 1: Fairness metrics

Not all metrics are needed to determine if the discrimination exists but they are helpful to acquire a fuller picture. To facilitate the visualization over many subgroups, we introduce function that maps metric scores among subgroups to single value. This function, which we called *parity_loss*, has an attractive property. Due to usage of absolute value of natural logarithm it will return the same value whether ratio is inversed or not.

So for example when we would like to know the parity loss of Statistical Parity between unprivileged (b) and privileged (a) subgroups we mean value like this:

$$STP_{parity\_loss} = \left| \ln \left(\frac{STP_b}{STP_a} \right) \right|. \tag{2}$$

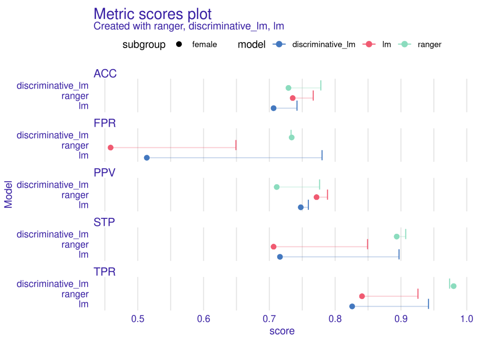
This notation is very helpful because it allows to accumulate STP_{parity_loss} over all unprivileged subgroups, so not only in binary case

$$STP_{parity\_loss} = \sum_{i \in \{a,b,...\}} \left| \ln \left(\frac{STP_i}{STP_a} \right) \right|. \tag{3}$$

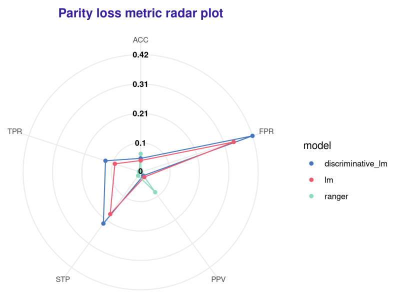
The *parity_loss* relates strictly to ratios. The classifier is more fair if *parity_loss* is low. This property is helpful in visualizations.

There are several modifying functions that operate on `fairness_object()`. Their usage will return other objects which will be visualized in the following chapter on the class diagram (Fig 6). The objects can then be plotted with generic `plot()` function. Additionally, there is a special plotting function that works immediately on `fairness_object` which is `plot_density`. In some functions the user can directly specify which metrics shall be visible in the plot. The detailed technical introduction for all these functions is presented in **fairmodels** manual. Plots visualizing different aspects of *parity_loss* can be created with one of following pipelines:

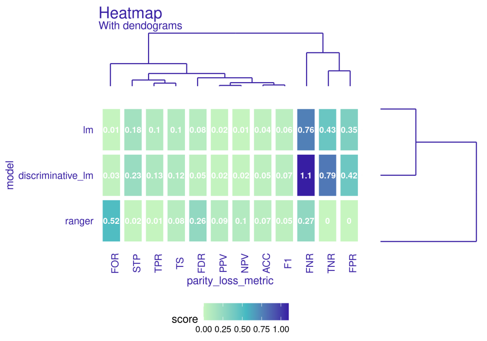
- `fairness_object` %>% **modifying_function(...)** %>% **plot()**
  This pipe is preferred and allows setting parameters in both modifying functions and certain plot functions which is not the case with the next pipeline.
- `fairness_object` %>% **plot_fairmodels(type = modifying_function, ...)**
  Additional parameters are passed to the modifying functions and not to the plot function.



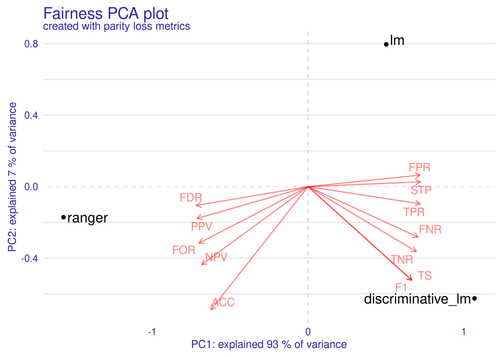
(a) metric_scores



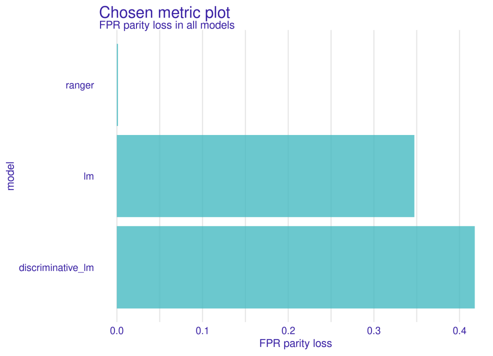
(b) fairness_radar



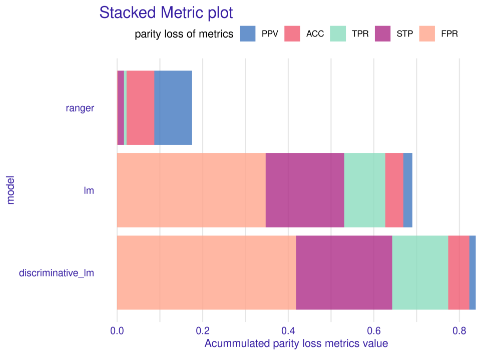
(c) fairness_heatmap



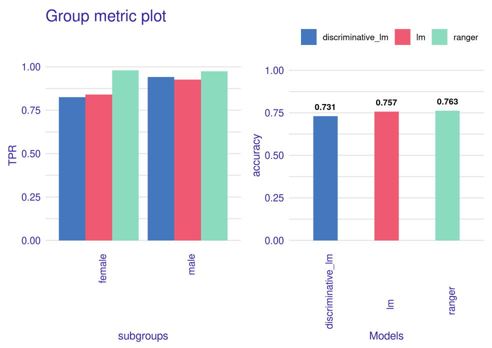
(d) fairness_pca



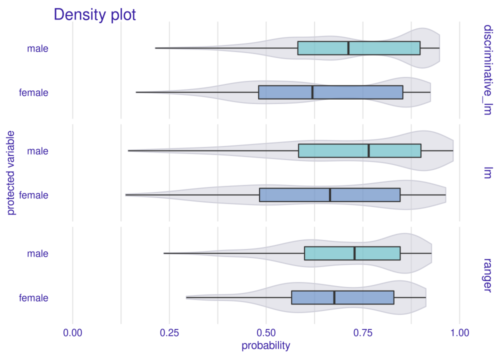
(e) choose_metric



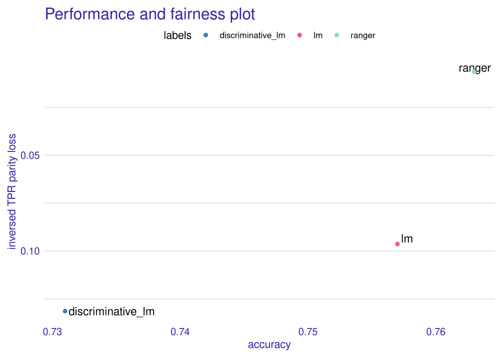
(f) stack_metrics



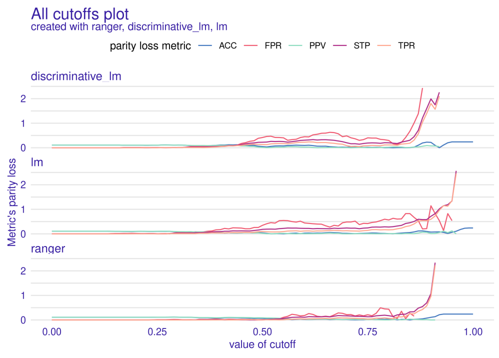
(g) group_metric



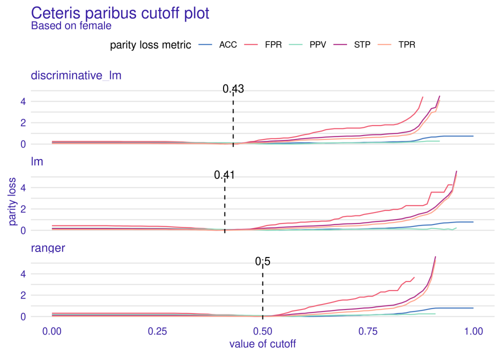
(h) plot_density



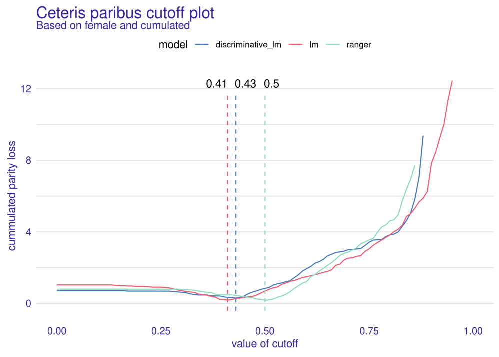
(i) performance_and_fairness



(j) all_cutoffs



(k) ceteris_paribus_cutoff



(l) ceteris_paribus_cutoff with parameter cumulated = TRUE

Figure 8: Figure depicts twelve methods for more detailed exploration of biases. The subtitle of each graph indicates the name of the function that produces that type of graph.

By using the pipelines, different kinds of plots can be obtained by superseding the **modifying_function** with function names (a-l). They can be seen in Figure 8. To see different aspects of fairness and bias, a user can choose the model with the smallest bias (b, e, f), find out if the metrics and models are similar to each other (c, d), compare models in both levels of fairness and performance (i, g), and see how cutoff manipulation might change the *parity_loss* (j, k, l).

Apart from methods (b-l), there is one special one, more related to plotting `fairness_check` (Figure 1 and 3), but uses its product - `fairness_object` like other visualization methods - function called `metric_scores` (a) shows exact rates of metrics for each model and subgroup.

Bias mitigation

What can be done if the model does not meet the fairness criteria? Machine learning practitioners might try use other algorithms or set of variables to construct unbiased models, but this does not guarantee passing the `fairness_check()`. An alternative is to use bias mitigation techniques that adjust the data or model so that fairness conditions are met. There are essentially three types of such methods. First is data pre-processing. When there are unwanted correlations between variables or sample sizes among subgroups in data, there are many ways to "correct" the data. Second one is in-processing, which is for example optimizing classifiers not only to reduce classification error but also to minimize a fairness metric. Last but not least is post-processing which modifies model output so that predictions and miss-predictions among subgroups are more alike.

The **fairmodels** package offers five functions for bias mitigation, three for pre-processing and two for post-processing algorithms. Most of these approaches are also implemented in aif360 (Bellamy et al., 2019) although in **fairmodels** there are separate implementations of them in R.

Data Pre-processing

- **Disparate impact remover**
  In **fairmodels** geometric repair, algorithm originally introduced by Feldman et al. (2015), works on ordinal, numeric features. Depending on the $\lambda \in [0, 1]$ parameter, this method will transform

the distribution of given feature. The idea is simple, given feature distribution in different subgroups the algorithm finds optimal distribution (according to earth movers distance) and transforms distribution for each subgroup to match the optimal one. The ranks of observations are preserved. Parameter λ is responsible for the amount of repair, so for full repair lambda should be set to 1.

- **Reweighting**
  Reweighting is a rather simple approach. This method was implemented according to Kamiran and Calders (2011). It focuses on mitigating statistical parity. It computes weights by dividing theoretical probability of assigning favorable label for subgroup by real (observed) probability (based in data). Theoretic probability for subgroup is computed by multiplying probability of assigning favorable label (for all population) by probability of picking observation from certain subgroup.
- **Resample**
  Resampling bases on weights calculated like in reweight. Each weight for subgroup is multiplied by the size of subgroup. Then, whether subgroup is deprived or not, observations are duplicated from either ones of favorable class or not. Two types of resampling- uniform and preferential. The uniform is making algorithm pick or omit observations randomly. Preferential is using other probabilistic classifier, later called ranker. Based on the probabilistic output of ranker, the observations are sorted and the ones with the highest/lowest ranks are either left out or duplicated depending on the case. More on that on Kamiran and Calders (2011). The **fairmodels** implementation instead of training some classifier, it uses vector of probabilities provided by user.

Model Post-processing

- **Reject Option based Classification Pivot**
  Based on Kamiran et al. (2012) in **fairmodels** *roc_pivot* method was implemented. Let $\theta \in (0, 1)$ be value that determines the radius of critical region. The center of this region is cutoff. The θ is specified by user and it should describe how big critical region should be. If assigned probability of observation is in described region, then with a certain assumption the probabilities are pivoting on the other side of cutoff. The said assumption is about membership to certain subgroup. If it is the privileged one, then probabilities are pivoting from the right side of cutoff to the left, and if it is the unprivileged subgroup than from left to right (assuming that user is predicting favorable outcome). Pivoting here means changing side of cutoff so that the distance from the cutoff stays unchanged.
- **Cutoff manipulation**
  The **fairmodels** package supports setting cutoff for each subgroup. User may pick *parity_loss* metrics of their choice and find the minimal *parity_loss*. It is part of `ceteris_paribus_cutoff()` function. Based on picked metrics, sum of parity loss is calculated for each cutoff of chosen subgroup. Then the minimal value is found. This way optimal values might be found for metrics of interest. The minimum is marked with dashed vertical line (see Figure 8 subplot l). This approach however might be to some extent concerning. Some might argue that this method of setting different cutoffs for different subgroups is unfair and is punishing privileged subgroups for something that they have no control of. Especially in individual fairness field it would be concerning if 2 similar people with different sensitive attributes would have 2 different thresholds and potentially 2 different outcomes. This is valid point and this method should be used with knowledge of all its drawbacks.

Methods listed above similarly to visualizing have two possible pipelines. All pre-processing methods can be used with 2 pipelines whereas post-processing can be used in one specific way.

1. Pre-processing pipelines
   - **data/explainer %>% method**
     Returns either weights, indexes or changed data depending of method used.
   - **data/explainer %>% pre_process_data(data, protected, y, type = ...)**
     Always returns data.frame. In case of weights data has additional column called `_weights_`.
2. Post-processing pipelines
   - `fairness_object` **%>% ceteris_paribus_cutoff(subgroup, ...) %>% print()/plot()**
     This is pipeline for creating ceteris paribus cutoff print and plot.
   - **explainer %>% roc_pivot(protected, privileged, ...)**
     The pipeline will return explainer with `y_hat` field changed.

Example

Now we will show an example usage of one pre-processing and one post-processing methods. As before, the **German Credit Data** will be used along with previously created `lm_model`. Firstly, we create new dataset using `pre_process_data` and then we use it to train the logistic regression classifier.

```
resampled_german   <- german %>% pre_process_data(protected = german$Sex,
                                                  y_numeric,
                                                  type = 'resample_uniform')

lm_model_resample  <- glm(Risk~.,
                          data   = resampled_german,
                          family = binomial(link = "logit"))

explainer_lm_resample <- DALEX::explain(lm_model_resample,
                                         data = german[,-1],
                                         y = y_numeric)
```

Then we make other explainer. We use previously created `explainer_lm` with post-processing function `roc_pivot`. We set parameter `theta = 0.05` for rather narrow area of pivot.

```
new_explainer <- explainer_lm %>% roc_pivot(protected = german$Sex,
                                             privileged = "male",
                                             theta = 0.05)
```

At the end we create `fairness_object` with explainers obtained with code above and one created in the first example to see the difference.

```
fobject <- fairness_check(explainer_lm_resample, new_explainer, explainer_lm,
                          protected = german['Sex'],
                          privileged = "male",
                          label = c("resample", "roc", "base"))

fobject %>% plot()
```

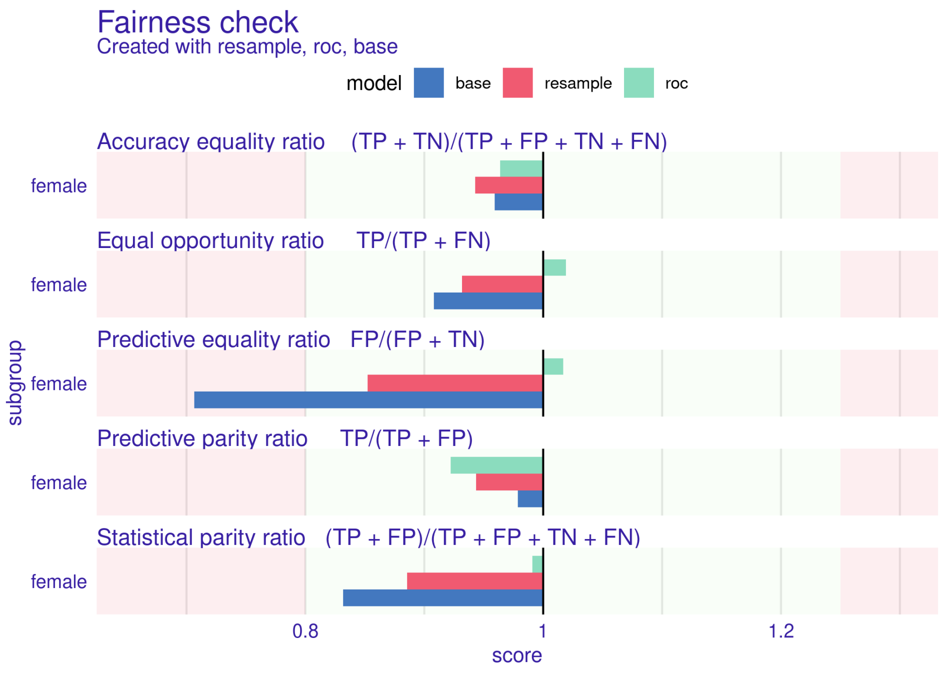


Figure 9: Graphical summary of a base model and model after two bias mitigation techniques.

Result of the code above is presented in Figure 9. The mitigation methods successfully eliminated bias in all of the metrics. Both of models are better than the original base. This is not always the case - sometimes eliminating bias in one metric may increase bias in another metric. For example let's consider a model which is perfectly accurate, but some subgroups receive little positive predictions (bias in Statistical parity). In that case, mitigating the bias in Statistical parity would increase bias in accuracy.

Summary and future work

In this paper we showed that checking for bias in machine learning model can be done in a convenient and flexible manner. The package **fairmodels** described above is a self-sufficient tool for bias detection, visualization and mitigation in classification machine learning models. We presented theory, package architecture, suggested usage of package and examples along with plots. Along the way, we introduced the core concepts and assumptions that come along the bias detection and plot interpretation.

The source code of the package, vignettes, examples and documentation can be found at https://modeloriented.github.io/fairmodels/. The stable version is available on CRAN. The code and the development version can be found on GitHub https://github.com/ModelOriented/fairmodels. This is also a place to report bugs or requests (through GitHub issues).

In future, we plan to enhance the spectrum of bias visualization plots and introduce methods for regression and individual fairness. The potential way to explore would be an in-processing bias mitigation - training models that minimize cost function and adhere to certain fairness criteria. This field is heavily developed in Python and lacks appropriate attention in R.

Acknowledgements

Work on this package was financially supported by the NCN Opus grant 2017/27/B/ST6/01307.

Bibliography

J. Angwin, J. Larson, S. Mattu, , and L. Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica*, 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. [p1]

S. Barocas, M. Hardt, and A. Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org. [p1, 2, 3]

R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilović, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, and Y. Zhang. Ai fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, 63(4/5):4:1–4:15, 2019. URL https://doi.org/10.1147/JRD.2019.2942287. [p2, 10]

R. Berk, H. Heidari, S. Jabbari, M. Kearns, and A. Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 03 2017. URL https://doi.org/10.1177/0049124118782533. [p3, 8]

P. Biecek. Dalex: Explainers for complex predictive models in r. *Journal of Machine Learning Research*, 19 (84):1–5, 2018. URL http://jmlr.org/papers/v19/18-416.html. [p3, 6]

S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, and K. Walker. Fairlearn: A toolkit for assessing and improving fairness in AI. Technical Report MSR-TR-2020-32, Microsoft, 2020. URL https://www.microsoft.com/en-us/research/publication/fairlearn-a-toolkit-for-assessing-and-improving-fairness-in-ai/. [p2]

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In S. A. Friedler and C. Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91, New York, NY, USA, 23–24 Feb 2018. URL http://proceedings.mlr.press/v81/buolamwini18a.html. [p1]

A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5, 10 2016. URL https://doi.org/10.1089/big.2016.0047. [p2, 8]

Code of Federal Regulations. Section 4d, uniform guidelines on employee selection procedures (1978), 1978. URL https://www.govinfo.gov/content/pkg/CFR-2014-title29-vol4/xml/CFR-2014-title29-vol4-part1607.xml. [p3]

S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 797–806, New York, NY, USA, 2017. Association for Computing Machinery. URL https://doi.org/10.1145/3097983.3098095. [p2, 8]

Council of Europe. Guidelines on facial recognition, 28 Jan 2021. URL https://www.coe.int/en/web/portal/-/facial-recognition-strict-regulation-is-needed-to-prevent-human-rights-violations-. [p1]

D. Dua and C. Graff. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). [p3]

C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226, New York, NY, USA, 2012. Association for Computing Machinery. URL https://doi.org/10.1145/2090236.2090255. [p2, 8]

M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 259–268, New York, NY, USA, 2015. Association for Computing Machinery. URL https://doi.org/10.1145/2783258.2783311. [p10]

H2O.ai. *H2O AutoML*, June 2017. URL http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html. H2O version 3.30.0.1. [p1]

M. Hardt, E. Price, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3315–3323. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6374-equality-of-opportunity-in-supervised-learning.pdf. [p2, 3, 8]

F. Kamiran and T. Calders. Data pre-processing techniques for classification without discrimination. *Knowledge and Information Systems*, 33, 10 2011. URL https://doi.org/10.1007/s10115-011-0463-8. [p1, 11]

F. Kamiran, A. Karim, and X. Zhang. Decision theory for discrimination-aware classification. In *2012 IEEE 12th International Conference on Data Mining*, pages 924–929, 2012. URL https://doi.org/10.1109/ICDM.2012.45. [p11]

P. Lahoti, K. P. Gummadi, and G. Weikum. ifair: Learning individually fair data representations for algorithmic decision making. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1334–1345, 2019. URL https://doi.org/10.1109/ICDE.2019.00121. [p1]

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning, 2019. URL https://arxiv.org/abs/1908.09635. [p1]

D. Plečko and N. Meinshausen. Fair data adaptation with quantile preservation, 2019. URL https://arxiv.org/abs/1911.06685. [p2]

P. Saleiro, B. Kuester, A. Stevens, A. Anisfeld, L. Hinkson, J. London, and R. Ghani. Aequitas: A bias and fairness audit toolkit, 2018. URL https://arxiv.org/abs/1811.05577. [p2]

M. Wick, S. Panda, and J.-B. Tristan. Unlocking fairness: a trade-off revisited. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8783–8792. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/373e4c5d8edfa8b74fd4b6791d0cf6dc-Paper.pdf. [p1]

M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01. [p5]

Jakub Wiśniewski
Faculty of Mathematics and Information Science
Warsaw University of Technology

http://fairmodels.drwhy.ai/

Poland
jakwisn@gmail.com

Przemysław Biecek
Faculty of Mathematics and Information Science
Warsaw University of Technology
ORCiD: 0000-0001-8423-1823
przemyslaw.biecek@pw.edu.pl