

Package ‘fairmodels’

September 21, 2020

Type Package

Title Flexible Tool for Bias Detection, Visualization, and Mitigation

Version 0.2.2

Description Measure fairness metrics in one place for many models. Check how big is model's bias towards different races, sex, nationalities etc. Use measures such as Statistical Parity, Equal odds to detect the discrimination against unprivileged groups. Visualize the bias using heatmap, radar plot, biplot, bar chart (and more!). There are various pre-processing and post-processing bias mitigation algorithms implemented.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports DALEX, ggplot2, patchwork, ggdendro, ggrepel, scales

Suggests ranger, gbm, knitr, rmarkdown, covr, testthat, spelling

RoxygenNote 7.1.1.9000

VignetteBuilder knitr

URL <https://modeloriented.github.io/fairmodels/>

BugReports <https://github.com/ModelOriented/fairmodels/issues>

Language en-US

NeedsCompilation no

Author Jakub Wiśniewski [aut, cre],
Przemysław Biecek [aut] (<<https://orcid.org/0000-0001-8423-1823>>)

Maintainer Jakub Wiśniewski <jakwisn@gmail.com>

Repository CRAN

Date/Publication 2020-09-21 16:20:04 UTC

R topics documented:

adult	3
adult_test	4
all_cutoffs	5
calculate_group_fairness_metrics	6
ceteris_paribus_cutoff	6
choose_metric	8
compas	9
confusion_matrix	10
disparate_impact_remover	11
expand_fairness_object	12
fairness_check	13
fairness_heatmap	16
fairness_pca	18
fairness_radar	19
german	20
group_matrices	21
group_metric	22
group_model_performance	24
metric_scores	25
performance_and_fairness	26
plot.all_cutoffs	27
plot.ceteris_paribus_cutoff	28
plot.chosen_metric	29
plot.fairness_heatmap	30
plot.fairness_object	32
plot.fairness_pca	33
plot.fairness_radar	34
plot.group_metric	35
plot.metric_scores	36
plot.performance_and_fairness	37
plot.stacked_metrics	38
plot_density	39
plot_fairmodels	40
pre_process_data	42
print.all_cutoffs	43
print.ceteris_paribus_cutoff	44
print.chosen_metric	45
print.fairness_heatmap	46
print.fairness_object	47
print.fairness_pca	48
print.fairness_radar	49
print.group_metric	50
print.metric_scores	51
print.performance_and_fairness	52
print.stacked_metrics	53
resample	54

<i>adult</i>	3
reweight	55
roc_pivot	56
stack_metrics	58
Index	59

adult	<i>Adult dataset</i>
-------	----------------------

Description

adult dataset consists of many columns containing various information about relationship, hours worked per week, workclass etc... and about salary, whether more than 50K a year or not. Lot's of possible protected attributes such as sex, race age. Some columns contain level "unknown" and these values are not removed and removing them depends on user as they might contain some information.

Usage

```
data(adult)
```

Format

A data frame with 32561 rows and 15 variables:

- salary** factor, <=50K/>50K whether a person salary exceeds 50K a year or not
- age** integer, age of person
- workclass** factor, field of work
- fnlwt** numeric
- education** factor, completed education degree
- education_num** numeric, education number in converted from education factor, the bigger the better
- marital_status** factor
- occupation** factor, where this person works
- relationship** factor, relationship information
- race** factor, ethnicity of a person
- sex** factor, gender of a person
- capital_gain** numeric
- capital_loss** numeric
- hours_per_week** numeric, how many hours per week does this person work
- native_country** factor, in which country was this person born

Source

Data from UCL <https://archive.ics.uci.edu/ml/datasets/adult>

`adult_test`*Adult test dataset*

Description

`adult_test` dataset consists of many columns containing various information about relationship, hours worked per week, workclass etc... and about salary, whether more than 50K a year or not. Lot's of possible protected attributes such as sex, race age. Some columns contain level "unknown" and these values are not removed and removing them depends on user as they might contain some information. Data is designed for testing and ready to go.

Usage

```
data(adult_test)
```

Format

A data frame with 16281 rows and 15 variables:

salary factor, <=50K/>50K whether a person salary exceeds 50K a year or not

age integer, age of person

workclass factor, field of work

fnlwgt numeric

education factor, completed education degree

education_num numeric, education number in converted from education factor, the bigger the better

marital_status factor

occupation factor, where this person works

relationship factor, relationship information

race factor, ethnicity of a person

sex factor, gender of a person

capital_gain numeric

capital_loss numeric

hours_per_week numeric, how many hours per week does this person work

native_country factor, in which country was this person born

Source

Data from UCL <https://archive.ics.uci.edu/ml/datasets/adult>

all_cutoffs	<i>All cutoffs</i>
-------------	--------------------

Description

Create `all_cutoffs` object and see how with the change of cutoffs parity loss of fairness metrics changes. Value of cutoff changes equally for all subgroups. User can pick which fairness metrics to create the object with via `fairness_metrics` vector.

Usage

```
all_cutoffs(x, grid_points = 101, fairness_metrics = fairness_check_metrics())
```

Arguments

<code>x</code>	object of class <code>fairness_object</code>
<code>grid_points</code>	numeric, grid for cutoffs to test. Number of points between 0 and 1 spread evenly
<code>fairness_metrics</code>	character, name of metric or vector of multiple metrics names

Value

`all_cutoffs` object, data.frame containing information about label, metric and parity_loss at particular cutoff

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")
```

```
ac <- all_cutoffs(fobject,  
                 fairness_metrics = c("TPR",  
                                     "FPR"))  
plot(ac)
```

calculate_group_fairness_metrics
Calculate fairness metrics in groups

Description

Create data.frame from group_matrices object containing metric scores for each subgroup.

Usage

```
calculate_group_fairness_metrics(x)
```

Arguments

x object of class group_matrices

Value

group_metric_matrix object It's a data.frame with metrics as row names and scores for those metrics for each subgroup in columns

ceteris_paribus_cutoff
Ceteris paribus cutoff

Description

Ceteris paribus cutoff is way to check how will parity loss behave if only cutoff for one subgroup was changed. By using parameter new_cutoffs parity loss for metrics with new cutoffs will be calculated. Note that cutoff for subgroup (passed as parameter) will change no matter new_cutoff's value at that position. When parameter cumulated is set to true, all metrics will be summed and facets will collapse to one plot with different models on it. Sometimes due to the fact that some metric might contain NA for all cutoff values, cumulated plot might be present without this model.

Usage

```
ceteris_paribus_cutoff(
  x,
  subgroup,
  new_cutoffs = NULL,
  fairness_metrics = fairness_check_metrics(),
  grid_points = 101,
  cumulated = FALSE
)
```

Arguments

x	object of class <code>fairness_object</code>
subgroup	character, name of subgroup (level in protected variable)
new_cutoffs	list of cutoffs with names matching those of subgroups. Each value should represent cutoff for particular subgroup. Position corresponding to subgroups in levels will be changed. Default is <code>NULL</code>
fairness_metrics	character, name of metric or vector of multiple metrics
grid_points	numeric, grid for cutoffs to test. Number of points between 0 and 1 spread evenly.
cumulated	logical, if <code>TRUE</code> facets will collapse to one plot and parity loss for each model will be summed. Default <code>FALSE</code> .

Value

`ceteris_paribus_cutoff` data.frame containing information about label, metric and parity_loss at particular cutoff

Examples

```
data("compas")

# positive outcome - not being recidivist
two_yr_recidivism <- factor(compas$Two_yr_Recidivism, levels = c(1,0))
y_numeric <- as.numeric(two_yr_recidivism) -1
compas$Two_yr_Recidivism <- two_yr_recidivism

lm_model <- glm(Two_yr_Recidivism~.,
  data=compas,
  family=binomial(link="logit"))

rf_model <- ranger::ranger(Two_yr_Recidivism ~.,
  data = compas,
  probability = TRUE,
  num.trees = 200)

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

```
fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = compas$Ethnicity,
                        privileged = "Caucasian")

cpc <- ceteris_paribus_cutoff(fobject, "African_American")

plot(cpc)
```

choose_metric

Choose metric

Description

Extracts metrics from `metric_data` from fairness object. It allows to visualize and compare chosen metric values across all models.

Usage

```
choose_metric(x, fairness_metric = "FPR")
```

Arguments

`x` object of class `fairness_object`

`fairness_metric`

char, name of fairness metric, one of metrics:

- `TPR_parity_loss` - parity loss of True Positive Rate (Sensitivity, Recall, Equal Odds)
- `TNR_parity_loss` - parity loss of True Negative Rate (Specificity)
- `PPV_parity_loss` - parity loss of Positive Predictive Value (Precision)
- `NPV_parity_loss` - parity loss of Negative Predictive Value
- `FNR_parity_loss` - parity loss of False Negative Rate
- `FPR_parity_loss` - parity loss of False Positive Rate
- `FDR_parity_loss` - parity loss of False Discovery Rate
- `FOR_parity_loss` - parity loss of False Omission Rate
- `TS_parity_loss` - parity loss of Threat Score
- `ACC_parity_loss` - parity loss of Accuracy
- `F1_parity_loss` - parity loss of F1 Score
- `MCC_parity_loss` - parity loss of Matthews correlation coefficient

Details

some of metrics give same parity loss as others (for example TPR and FNR and that is because $TPR = 1 - FNR$)

Value

chosen_metric object It is a list with following fields:

- parity_loss_metric_data data.frame with columns: parity_loss_metric and label
- metric chosen metric
- label character, vector of model labels

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                           data = german,
                           probability = TRUE,
                           num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                         protected = german$Sex,
                         privileged = "male")

cm <- choose_metric(fobject, "TPR")
plot(cm)
```

compas

Modified COMPAS dataset

Description

compas dataset. From ProPublica: across the nation, judges, probation and parole officers are increasingly using algorithms to assess a criminal defendant's likelihood to re-offend.

Usage

```
data(compas)
```

Format

A data frame with 6172 rows and 7 variables:

Details

Two_yr_Recidivism factor, 1/0 for future recidivism or no recidivism. Models should predict this values

Number_of_Priors numeric, number of priors

Age_Above_FourtyFive factor, 1/0 for age above 45 years or not

Age_Below_TwentyFive factor, 1/0 for age below 25 years or not

Misdemeanor factor, 1/0 for having recorded misdemeanor(s) or not

Ethnicity factor, Caucasian, African American, Asian, Hispanic, Native American or Other

Sex factor, female/male for gender

Source

The original source of data is <https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-da>
Modified data used here comes from <https://www.kaggle.com/danofer/compass/> (propublica-CompassRecidivism_data_fairml.csv)

confusion_matrix	<i>Confusion matrix</i>
------------------	-------------------------

Description

Calculates confusion matrix for given cutoff

Usage

```
confusion_matrix(probs, observed, cutoff)
```

Arguments

probs	numeric, vector with probabilities given by model
observed	numeric, vector with actual values from outcome, either 0 or 1
cutoff	numeric, single value denoting cutoff/threshold

Value

object of class `confussion_matrix` It is a list with following fields:

- `tpnumber` of True Positives
- `fpnumber` of False Positives
- `ttnumber` of True Negatives
- `fnnumber` of False Negatives

Examples

```
probs    <- rnorm(20, 0.4, 0.1)
observed <- round(runif(20))

confusion_matrix(probs, observed, 0.5)
```

disparate_impact_remover

Disparate impact remover

Description

Disparate impact remover is a pre-processing bias mitigation method. It removes bias hidden in numeric columns in data. It changes distribution of ordinal features of data with regard to earth mover distance. It works best if among subgroups there is similar number of observations.

Usage

```
disparate_impact_remover(data, protected, features_to_transform, lambda = 1)
```

Arguments

data	data.frame, data to be transformed
protected	factor, vector containing sensitive information such as gender, race etc... If vector is character it will transform it to factor.
features_to_transform	character, vector of column names to be transformed. Columns must have numerical, ordinal values
lambda	numeric, amount of repair desired. Value from 0 to 1, where 0 will return almost unchanged dataset and 1 fully repaired dataset

Details

This is implementation of geometric method which preserves ranks unlike combinatorial repair. lambda close to 1 denotes that distributions will be very close to each other and lambda close to 0 means that densities will barely change. Note that although lambda equal 0 should mean that original data will be returned, it usually changes distributions slightly due to pigeonholing. The number of pigeonholes is fixed and equal to $\min(101, \text{unique}(a))$, where a is vector with values for subgroup. So if some subgroup is not numerous and the distribution is discrete with small number of variables then there will be small number of pigeonholes. It will affect data significantly.

Value

repaired data (data.frame object)

References

This method was implemented based on Feldman, Friedler, Moeller, Scheidegger, Venkatasubramanian 2015 <https://arxiv.org/pdf/1412.3756.pdf>

Examples

```
library("ggplot2")

set.seed(1)
# custom data frame with kind and score
custom_data <- data.frame(kind = as.factor(c(rep("second", 500), rep("first", 500))),
                          score = c(rnorm(500, 400, 40), rnorm(500, 600, 100)))

ggplot(custom_data, aes(score, fill = kind)) + geom_density(alpha = 0.5)

fixed_data <- disparate_impact_removaler(data = custom_data,
                                       protected = custom_data$kind,
                                       features_to_transform = "score",
                                       lambda = 0.8)

ggplot(fixed_data, aes(score, fill = kind)) + geom_density(alpha = 0.5)

# lambda 1 gives identical distribution, lambda 0 (almost) original distributions
fixed_data_unchanged <- disparate_impact_removaler(data = custom_data,
                                                  protected = custom_data$kind,
                                                  features_to_transform = "score",
                                                  lambda = 0)

ggplot(fixed_data_unchanged, aes(score, fill = kind)) + geom_density(alpha = 0.5)

fixed_data_fully_changed <- disparate_impact_removaler(data = custom_data,
                                                       protected = custom_data$kind,
                                                       features_to_transform = "score",
                                                       lambda = 1)

ggplot(fixed_data_fully_changed, aes(score, fill = kind)) +
  geom_density(alpha = 0.5) +
  facet_wrap(kind~., nrow = 2)
```

expand_fairness_object

Expand Fairness Object

Description

Unfold fairness object to 3 columns (metrics, label, score) to construct better base for visualization.

Usage

```
expand_fairness_object(x, scale = FALSE, drop_metrics_with_na = FALSE)
```

Arguments

```
x                object of class fairness_object
scale            logical, if TRUE standardised.
drop_metrics_with_na
                 logical, if TRUE metrics with NA will be omitted
```

Value

object of class `expand_fairness_object`. It is a `data.frame` with scores for each metric and model.

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

expand_fairness_object(fobject, drop_metrics_with_na = TRUE)
```

Description

Fairness check creates `fairness_object` which measures different fairness metrics and wraps data, explainers and parameters in useful object. This is fundamental object in this package. It enables to visualize fairness metrics and models in many ways and compare models on both fairness and performance level. Fairness check acts as merger and wrapper for explainers and fairness objects. While other fairness objects values are not changed, fairness check assigns cutoffs and labels to provided explainers so same explainers with changed labels/cutoffs might be gradually added to fairness object. Users through `print` and `plot` methods may quickly check values of most popular fairness metrics. More on that topic in details.

Usage

```
fairness_check(
  x,
  ...,
  protected = NULL,
  privileged = NULL,
  cutoff = NULL,
  label = NULL,
  epsilon = 0.8,
  verbose = TRUE,
  colorize = TRUE
)
```

Arguments

<code>x</code>	object created with <code>explain</code> or of class <code>fairness_object</code>
<code>...</code>	possibly more objects created with <code>explain</code> and/or objects of class <code>fairness_object</code>
<code>protected</code>	factor, protected variable (also called sensitive attribute), containing privileged and unprivileged groups
<code>privileged</code>	factor/character, one value of <code>protected</code> , in regard to what subgroup parity loss is calculated
<code>cutoff</code>	numeric, vector of cutoffs (thresholds) for each value of protected variable, affecting only explainers.
<code>label</code>	character, vector of labels to be assigned for explainers, default is explainer label.
<code>epsilon</code>	numeric, boundary for fairness checking, lowest acceptable ratio of metrics
<code>verbose</code>	logical, whether to print information about creation of fairness object
<code>colorize</code>	logical, whether to print information in color

Details

Fairness check

Metrics used are made for each subgroup, then base metric score is subtracted leaving loss of particular metric. If absolute loss of metrics ratio is not within acceptable boundaries than such metric is marked as "not passed". It means that values of metrics should be within (`epsilon`, `1/epsilon`)

boundary. The default ratio is set to 0.8 which adhere to US 80 of scores achieved in metrics by privileged subgroup. For example if $TPR_{unprivileged}/TPR_{privileged}$ is less than 0.8 then such ratio is sign of discrimination. On the other hand if $TPR_{privileged}/TPR_{unprivileged}$ is more than 1.25 ($1/0.8$) than there is discrimination towards privileged group. Epsilon value can be adjusted to user's needs. There are some metrics that might be derived from existing metrics (For example Equalized Odds - equal TPR and FPR for all subgroups). That means passing 5 metrics in fairness check asserts that model is even more fair. In `fairness_check` models must always predict positive result. Not adhering to this rule may lead to misinterpretation of the plot. More on metrics and their equivalents: <https://fairware.cs.umass.edu/papers/Verma.pdf> [https://en.wikipedia.org/wiki/Fairness_\(machine_learning\)](https://en.wikipedia.org/wiki/Fairness_(machine_learning))

Parity loss - visualization tool

Parity loss is computed as follows: $M_{parity_loss} = \sum(\text{abs}(\log(\text{metric}/\text{metric}_{privileged})))$

where:

M - some metric mentioned above

metric - vector of metric scores from each subgroup `metric_privileged` - value of metric vector for privileged subgroup

`base_metric` - scalar, value of metric for base subgroup

Value

An object of class `fairness_object` which is a list with elements:

- `parity_loss_metric_data` - data.frame containing parity loss for various fairness metrics. Created with following metrics:
 - TPR - True Positive Rate (Sensitivity, Recall)
 - TNR - True Negative Rate (Specificity)
 - PPV - Positive Predictive Value (Precision)
 - NPV - Negative Predictive Value
 - FNR - False Negative Rate
 - FPR - False Positive Rate
 - FDR - False Discovery Rate
 - FOR - False Omission Rate
 - TS - Threat Score
 - STP - Statistical Parity
 - ACC - Accuracy
 - F1 - F1 Score
- `groups_data` - metrics across levels in protected variable
- `groups_confusion_matrices` - confusion matrices for each subgroup
- `explainers` - list of DALEX explainers used to create object
- `cutoffs` - list of cutoffs for each explainer and subgroup
- `fairness_check_data` - data.frame used for for plotting `fairness_object`
- ... - other parameters passed to function

References

Zafar, Valera, Rodriguez, Gummadi (2017) <https://arxiv.org/pdf/1610.08452.pdf>

Hardt, Price, Srebro (2016) <https://arxiv.org/pdf/1610.02413.pdf>

Verma, Rubin (2018) <https://fairware.cs.umass.edu/papers/Verma.pdf>

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          max.depth = 3,
                          num.trees = 100,
                          seed = 1)

explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

plot(fobject)
```

fairness_heatmap

Fairness heatmap

Description

Create `fairness_heatmap` object to compare both models and metrics. If parameter `scale` is set to `TRUE` metrics will be scaled to median = 0 and sd = 1. If NA's appear heatmap will still plot, but with gray area where NA's were.

Usage

```
fairness_heatmap(x, scale = FALSE)
```


Arguments

`x` object of class `fairness_object`
`scale` logical, if code TRUE metrics will be scaled to mean 0 and sd 1. Default FALSE

Value

`fairness_heatmap` object.

It is a list with following fields:

- `heatmap_data` - `data.frame` with information about score for model and parity loss metric
- `matrix_model` - matrix used in dendrogram plots
- `scale` - logical parameter passed to `fairness_heatmap`
- `label` - character, vector of model labels

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list(female = 0.4),
                        label = c("lm_2", "rf_2"))

fh <- fairness_heatmap(fobject)

plot(fh)
```

 fairness_pca

Fairness PCA

Description

Calculate PC for metric_matrix to see similarities between models and metrics. If omit_models_with_NA is set to TRUE models with NA will be omitted as opposed to default behavior, when metrics are omitted.

Usage

```
fairness_pca(x, omit_models_with_NA = FALSE)
```

Arguments

x object of class fairness object
 omit_models_with_NA logical, if TRUE omits rows in metric_matrix, else omits columns (default)

Value

fairness_pca object It is list containing following fields:

- pc_1_2 - amount of data variance explained with each component
- rotation - rotation from stats::prcomp
- x - x from stats::prcomp
- sdev - sdev from stats::prcomp
- label - model labels

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) - 1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

```
fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list( female = 0.4),
                        label = c("lm_2", "rf_2"))

fpca <- fairness_pca(fobject)

plot(fpca)
```

fairness_radar

Fairness radar

Description

Make `fairness_radar` object with chosen `fairness_metrics`. Note that there must be at least three metrics that does not contain NA.

Usage

```
fairness_radar(x, fairness_metrics = fairness_check_metrics())
```

Arguments

`x` object of class `fairness_object`

`fairness_metrics` character, vector of metric names, at least 3 metrics without NA needed. If NULL default metrics will be used, which are ones from `fairness_check` plot.

Value

`fairness_radar` object. It is a list containing:

- `radar_data` - `data.frame` containing scores for each model and parity loss metric
- `label` - model labels

Examples

```

data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                           data = german,
                           probability = TRUE,
                           num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                          protected = german$Sex,
                          privileged = "male")

fradar <- fairness_radar(fobject, fairness_metrics = c("ACC",
                                                    "STP",
                                                    "TNR",
                                                    "TPR",
                                                    "PPV"))

plot(fradar)

```

german

Modified German Credit data dataset

Description

german dataset. Data contains information about people and their credit risks.

Usage

```
data(german)
```

Format

A data frame with 1000 rows and 10 variables:

Risk factor, good/bad risk connected with giving the credit. Models should predict this values

Sex factor, male/female , considered to be protected group

Job numeric, job titles converted to integers where 0- unemployed/unskilled, 3- management/ self-employed/highly qualified employee/ officer

Housing factor, rent/own/free where this person lives

Saving.accounts factor, little/moderate/quite rich/rich/not_known, where not_known indicates NA

Checking.account factor, little/moderate/rich/not_known, where not_known indicates NA

Credit.amount numeric, amount of money in credit

Duration numeric, duration of credit

Purpose factor, purpose of credit

Age numeric, age of person that applied for credit

Source

Data from kaggle <https://www.kaggle.com/kabure/german-credit-data-with-risk/>. The original source is UCL [https://archive.ics.uci.edu/ml/datasets/Statlog+\(German+Credit+Data\)](https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)).

group_matrices	<i>Group confusion matrices</i>
----------------	---------------------------------

Description

Calculates confusion matrices for each subgroup

Usage

```
group_matrices(protected, probs, preds, cutoff)
```

Arguments

protected	vector containing protected variable
probs	character name of column with probabilities
preds	numeric, vector with predictions
cutoff	numeric cutoff for probabilities, default = 0.5

Value

group_matrices object It is a list with values:

For each subgroup:

- subgroup
 - tp - number of true positives
 - fp - number of false positives
 - tn - number of true negatives
 - fn - number of false negatives

Examples

```

data("compas")

glm_compas <- glm(Two_yr_Recidivism~., data=compas, family=binomial(link="logit"))
y_prob <- glm_compas$fitted.values

y_numeric <- as.numeric(compas$Two_yr_Recidivism)-1

gm <- group_matrices(compas$Ethnicity,
                     y_prob,
                     y_numeric,
                     cutoff = list(Asian = 0.45,
                                    African_American = 0.5,
                                    Other = 0.5,
                                    Hispanic = 0.5,
                                    Caucasian = 0.4,
                                    Native_American = 0.5))

gm

```

group_metric

Group metric

Description

Group metric enables to extract data from metrics generated for each subgroup (values in protected variable) The closer metric values are to each other, the less bias particular model has. If parity_loss parameter is set to TRUE, distance between privileged and unprivileged subgroups will be measured. When plotted shows both fairness metric and chosen performance metric.

Usage

```

group_metric(
  x,
  fairness_metric = NULL,
  performance_metric = NULL,
  parity_loss = FALSE,
  verbose = TRUE
)

```

Arguments

x object of class fairness_object
fairness_metric character, fairness metric name
performance_metric character, performance metric name

parity_loss	logical, if TRUE parity loss will supersede basic metric
verbose	logical, whether to print information about metrics on console or not. Default TRUE

Details

Available metrics:

Fairness metrics:

- TPR
- TNR
- PPV
- NPV
- FNR
- FPR
- FDR
- FOR
- TS
- ACC
- F1

Performance metrics

- recall
- precision
- accuracy
- f1
- auc

Value

`group_metric` object. It is a list with following items:

- `group_metric_data` - data.frame containing fairness metric scores for each model
- `performance_data` - data.frame containing performance metric scores for each model
- `fairness_metric` - name of fairness metric
- `performance_metric` - name of performance metric

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

gm <- group_metric(fobject, "TPR", "f1", parity_loss = TRUE)

plot(gm)
```

group_model_performance

Group model performance

Description

Special method for model performance evaluation. Counts number of tp, tn, fp, fn for each subgroup (and therefore potentially distinct cutoff), sums afterwards.

Usage

```
group_model_performance(x, protected, cutoff, performance_metric)
```

Arguments

x	object created with explain
protected	factor, vector with levels as subgroups
cutoff	vector of thresholds for each subgroup
performance_metric	name of performance metric

Value

score in performance metric between 0 and 1

metric_scores	<i>Metric scores</i>
---------------	----------------------

Description

Creates `metric_scores` object to facilitate visualization. Check how the metric scores differ among models, what is this score, and how it changes for example after applying bias mitigation technique. The vertical black lines denote the scores for privileged subgroup. It is best to use only few metrics (using `fairness_metrics` parameter)

Usage

```
metric_scores(x, fairness_metrics = fairness_check_metrics())
```

Arguments

`x` object of class `fairness_object`
`fairness_metrics` character, vector with fairness metric names. Default metrics are ones in `fairness_check_plot`

Value

`metric_scores` object. It is a list containing:

- `metric_scores_data` - `data.frame` with information about score in particular subgroup, metric, and model
- `privileged` - name of privileged subgroup

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) - 1

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

rf_model <- ranger::ranger(Risk ~ .,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)
```

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

ms <- metric_scores(fobject, fairness_metrics = c("TPR", "STP", "ACC"))
plot(ms)
```

performance_and_fairness

Performance and fairness

Description

Measure performance in both fairness metric and

Usage

```
performance_and_fairness(x, fairness_metric = NULL, performance_metric = NULL)
```

Arguments

`x` object of class `fairness_object`
`fairness_metric` fairness metric, one of those in `fairness_object`
`performance_metric` performance metric, one of

Details

Creates `performance_and_fairness` object. Measure model performance and model fairness metric at the same time. Choose best model according to both metrics. When plotted y axis is inverted to accentuate that models in top right corner are the best according to both metrics.

Value

`performance_and_fairness` object. It is list containing:

- `paf_data` - performance and fairness data. frame containing fairness and performance metric scores for each model
- `fairness_metric` - chosen fairness metric name
- `performance_metric` - chosen performance_metric name
- `label` - model labels

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list( female = 0.4),
                        label = c("lm_2", "rf_2"))

paf <- performance_and_fairness(fobject)

plot(paf)
```

plot.all_cutoffs *Plot all cutoffs*

Description

All cutoffs plot allows to check how parity loss of chosen metrics is affected by the change of cutoff. Values of cutoff are the same for all subgroups (levels of protected variable) no matter what cutoff values were in fairness_object.

Usage

```
## S3 method for class 'all_cutoffs'
plot(x, ..., label = NULL)
```

Arguments

x all_cutoffs object
 ... other plot parameters
 label character, label of model to plot. Default NULL. If default prints all models.

Value

ggplot2 object

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

ac <- all_cutoffs(fobject,
                 fairness_metrics = c("TPR",
                                     "FPR"))

plot(ac)
```

plot.ceteris_paribus_cutoff

Ceteris paribus cutoff plot

Description

Ceteris paribus cutoff is way to check how will parity loss behave if we changed only cutoff in one subgroup. It plots object of class `ceteris_paribus_cutoff`. It might have two types - default and cumulated. Cumulated sums metrics and plots it all in one plot. When default one is used all chosen metrics will be plotted for each model.

Usage

```
## S3 method for class 'ceteris_paribus_cutoff'  
plot(x, ...)
```

Arguments

```
x          ceteris_paribus_cutoff object  
...        other plot parameters
```

Value

ggplot2 object

Examples

```
data("german")  
  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                          data = german,  
                          probability = TRUE,  
                          num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                        protected = german$Sex,  
                        privileged = "male")  
  
cpc <- ceteris_paribus_cutoff(fobject, "female")  
plot(cpc)  
  
cpc <- ceteris_paribus_cutoff(fobject, "female", cumulated = TRUE)  
plot(cpc)
```

plot.chosen_metric *Plot chosen metric*

Description

Choose metric from parity loss metrics and plot it for every model. The one with the least parity loss is more fair in terms of this particular metric.

Usage

```
## S3 method for class 'chosen_metric'
plot(x, ...)
```

Arguments

```
x                object of class chosen_metric
...              other objects of class chosen_metric
```

Value

ggplot2 object

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

cm <- choose_metric(fobject, "TPR")
plot(cm)
```

plot.fairness_heatmap *Plot Heatmap*

Description

Heatmap shows all parity loss metrics across all models while displaying similarity between variables (in form of dendrograms). All metrics are visible. Some have identical values as it should be in terms of their parity loss (eg. TPR parity loss == FNR parity loss, because $TPR = 1 - FNR$). NA's in metrics are gray.

Usage

```
## S3 method for class 'fairness_heatmap'
plot(
  x,
  ...,
  midpoint = NULL,
  title = NULL,
  subtitle = NULL,
  text = TRUE,
  text_size = 3,
  flip_axis = FALSE
)
```

Arguments

x	fairness_heatmap
...	other fairness_heatmap objects
midpoint	numeric, midpoint on gradient scale
title	character, title of the plot
subtitle	character, subtitle of the plot
text	logical, default TRUE means it shows values on tiles
text_size	numeric, size of text
flip_axis	logical, whether to change axis with metrics on axis with models

Value

list of ggplot2 objects

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
  data = german,
  probability = TRUE,
  num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
```

```

        protected = german$Sex,
        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list(female = 0.4),
                        label = c("lm_2", "rf_2"))

fh <- fairness_heatmap(fobject)

plot(fh)

```

plot.fairness_object *Plot fairness object*

Description

Plot fairness check enables to look how big differences are between base subgroup (privileged) and unprivileged ones. If bar plot reaches red zone it means that for this subgroup fairness goal is not satisfied. Multiple subgroups and models can be plotted. Red and green zone boundary can be moved through epsilon parameter, that needs to be passed through `fairness_check`.

Usage

```

## S3 method for class 'fairness_object'
plot(x, ...)

```

Arguments

```

x          fairness_object object
...        other plot parameters

```

Value

```

ggplot2 object

```

Examples

```

data("german")

y_numeric <- as.numeric(german$Risk) -1

lm_model <- glm(Risk~.,

```



```

      data = german,
      family=binomial(link="logit"))

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          max.depth = 3,
                          num.trees = 100,
                          seed = 1)

explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

plot(fobject)

```

plot.fairness_pca *Plot fairness PCA*

Description

Plot pca calculated on fairness_object metrics. Similar models and metrics should be close to each other. Plot doesn't work on multiple fairness_pca objects. Unlike in other plots here other fairness_pca objects cannot be added.

Usage

```

## S3 method for class 'fairness_pca'
plot(x, scale = 0.5, ...)

```

Arguments

x	fairness_pca object
scale	scaling loadings plot, from 0 to 1
...	other plot parameters

Value

ggplot2 object

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list( female = 0.4),
                        label = c("lm_2", "rf_2"))

fpca <- fairness_pca(fobject)

plot(fpca)
```

plot.fairness_radar *Plot fairness radar*

Description

Makes radar plot showing different fairness metrics that allow to compare models.

Usage

```
## S3 method for class 'fairness_radar'
plot(x, ...)
```

Arguments

x	fairness_radar object
...	other plot parameters

Value

ggplot2 object

References

code based on ModelOriented auditor package, thanks agosiewska! <https://modeloriented.github.io/auditor/>

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                           data = german,
                           probability = TRUE,
                           num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                         protected = german$Sex,
                         privileged = "male")

fradar <- fairness_radar(fobject)

plot(fradar)
```

plot.group_metric *Plot group metric*

Description

Plot chosen metric in group. Notice how models are treating different subgroups. Compare models both in fairness metrics and in performance. Parity loss can be enabled when creating group_metric object.

Usage

```
## S3 method for class 'group_metric'
plot(x, ...)
```

Arguments

x object of class group_metric
... other group_metric objects and other parameters

Value

list of ggplot2 objects

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

gm <- group_metric(fobject)

plot(gm)
```

plot.metric_scores *Plot metric scores*

Description

Plot metric scores

Usage

```
## S3 method for class 'metric_scores'
plot(x, ...)
```

Arguments

x metric_scores object
... other plot parameters

Value

ggplot2 object

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

ms <- metric_scores(fobject, fairness_metrics = c("TPR", "STP", "ACC"))
plot(ms)
```

plot.performance_and_fairness

Plot fairness and performance

Description

visualize fairness and model metric at the same time. Note that fairness metric parity scale is reversed so that the best models are in top right corner.

Usage

```
## S3 method for class 'performance_and_fairness'
plot(x, ...)
```

Arguments

x performance_and_fairness object
 ... other plot parameters

Value

ggplot object

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list( female = 0.4),
                        label = c("lm_2", "rf_2"))

paf <- performance_and_fairness(fobject)

plot(paf)
```

plot.stacked_metrics *Plot stacked Metrics*

Description

Stacked metrics is like plot for chosen_metric but with all unique metrics stacked on top of each other. Metrics containing NA's will be dropped to enable fair comparison.

Usage

```
## S3 method for class 'stacked_metrics'  
plot(x, ...)
```

Arguments

```
x          stacked_metrics object  
...        other plot parameters
```

Value

ggplot2 object

Examples

```
data("german")  
  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                          data = german,  
                          probability = TRUE,  
                          num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                        protected = german$Sex,  
                        privileged = "male")  
  
sm <- stack_metrics(fobject)  
plot(sm)
```

plot_density

Plot fairness object

Description

Plot distribution for models output probabilities. See how being in particular subgroup affects models decision.

Usage

```
plot_density(x, ...)
```

Arguments

```
x          object of class fairness_object
...        other plot parameters
```

Value

```
ggplot2 object
```

Examples

```
data("compas")

glm_compas <- glm(Two_yr_Recidivism~., data=compas, family=binomial(link="logit"))

y_numeric <- as.numeric(compas$Two_yr_Recidivism)-1

explainer_glm <- DALEX::explain(glm_compas, data = compas, y = y_numeric)

fobject <- fairness_check(explainer_glm,
                        protected = compas$Ethnicity,
                        privileged = "Caucasian")

plot_density(fobject)
```

```
plot_fairmodels      Plot fairmodels
```

Description

Easier access to all plots in fairmodels. Provide plot type (that matches to function name), pass additional parameters and plot.

Usage

```
plot_fairmodels(x, type, ...)

## S3 method for class 'explainer'
plot_fairmodels(x, type = "fairness_check", ..., protected, privileged)

## S3 method for class 'fairness_object'
plot_fairmodels(x, type = "fairness_check", ...)

## Default S3 method:
plot_fairmodels(x, type = "fairness_check", ...)
```


Arguments

x	object created with fairness_check or with explain
type	character, type of plot. Should match function name in fairmodels. Default is fairness_check.
...	other parameters passed to fairmodels functions.
protected	factor, vector containing sensitive attributes such as gender, race, etc...
privileged	character/factor, level in factor denoting privileged subgroup

Details

types (function names) available:

- fairness_check
- stack_metrics
- fairness_heatmap
- fairness_pca
- fairness_radar
- group_metric
- choose_metric
- metric_scores
- performance_and_fairness
- all_cutoffs
- ceteris_paribus_cutoff

Value

ggplot2 object

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

lm_model <- glm(Risk~.,
                data = german,
                family=binomial(link="logit"))
explainer_lm <- DALEX::explain(lm_model, data = german[, -1], y = y_numeric)

# works with explainer when protected and privileged are passed
plot_fairmodels(explainer_lm,
                type = "fairness_radar",
                protected = german$Sex,
                privileged = "male")
```

```
# or with fairness_object
fobject <- fairness_check(explainer_lm,
                        protected = german$Sex,
                        privileged = "male")

plot_fairmodels(fobject, type = "fairness_radar")
```

```
pre_process_data      Pre-process data
```

Description

Function aggregates all pre-processing algorithms for bias mitigation. User passes unified arguments and specifies type to receive transformed data. frame

Usage

```
pre_process_data(data, protected, y, type = "resample_uniform", ...)
```

Arguments

data	data.frame
protected	factor, protected attribute (sensitive variable) containing information about gender, race etc...
y	numeric, numeric values of predicted variable. 1 should denote favorable outcome.
type	character, type of pre-processing algorithm to be used, one of: <ul style="list-style-type: none"> • resample_uniform • resample_preferential • reweight • disparate_impact_removal
...	other parameters passed to pre-processing algorithms

Value

modified data (data.frame). In case of type = 'reweight' data has feature '_weights_' containing weights that need to be passed to model. In other cases data is ready to be passed as training data to a model.

Examples

```
data("german")

pre_process_data(german,
                german$Sex,
                as.numeric(german$Risk)-1,
                type = "disparate_impact_removal",
                features_to_transform = 'Age')
```

```
print.ceteris_paribus_cutoff  
      Print ceteris paribus cutoff
```

Description

Print ceteris paribus cutoff

Usage

```
## S3 method for class 'ceteris_paribus_cutoff'  
print(x, ...)
```

Arguments

x	ceteris_paribus_cutoff object
...	other print parameters

Examples

```
data("german")  
  
german <- german[1:500,]  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                          data = german,  
                          probability = TRUE,  
                          num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                        protected = german$Sex,  
                        privileged = "male")  
  
ceteris_paribus_cutoff(fobject, "female")
```

```
print.chosen_metric Print chosen metric
```

Description

Choose metric from parity loss metrics and plot it for every model. The one with the least parity loss is more fair in terms of this particular metric.

Usage

```
## S3 method for class 'chosen_metric'  
print(x, ...)
```

Arguments

x	chosen_metric object
...	other print parameters

Examples

```
data("german")  
  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                          data = german,  
                          probability = TRUE,  
                          num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                        protected = german$Sex,  
                        privileged = "male")  
  
cm <- choose_metric(fobject, "TPR")  
print(cm)
```

```
print.fairness_heatmap
```

Print fairness heatmap

Description

Print fairness heatmap

Usage

```
## S3 method for class 'fairness_heatmap'  
print(x, ...)
```

Arguments

x	fairness_heatmap object
...	other print parameters

Examples

```
data("german")  
  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                          data = german,  
                          probability = TRUE,  
                          num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                        protected = german$Sex,  
                        privileged = "male")  
  
# same explainers with different cutoffs for female  
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,  
                        protected = german$Sex,  
                        privileged = "male",  
                        cutoff = list( female = 0.4),  
                        label = c("lm_2", "rf_2"))
```

```
fh <- fairness_heatmap(fobject)
print(fh)
```

print.fairness_object *Print Fairness Object*

Description

Print Fairness Object

Usage

```
## S3 method for class 'fairness_object'
print(x, ..., colorize = TRUE)
```

Arguments

x	fairness_object object
...	other parameters
colorize	logical, whether information about metrics should be in color or not

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) - 1

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

rf_model <- ranger::ranger(Risk ~ .,
                           data = german,
                           probability = TRUE,
                           max.depth = 3,
                           num.trees = 100,
                           seed = 1)

explainer_lm <- DALEX::explain(lm_model, data = german[, -1], y = y_numeric)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")
```

```
print(fobject)
```

```
print.fairness_pca    Print fairness PCA
```

Description

Print principal components after using `pca` on fairness object

Usage

```
## S3 method for class 'fairness_pca'
print(x, ...)
```

Arguments

```
x                fairness_pca object
...              other print parameters
```

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) - 1

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

rf_model <- ranger::ranger(Risk ~ .,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
```



```
      cutoff = list( female = 0.4),
      label = c("lm_2", "rf_2"))

fpca <- fairness_pca(fobject)

print(fpca)
```

print.fairness_radar *Print fairness radar*

Description

Print fairness radar

Usage

```
## S3 method for class 'fairness_radar'
print(x, ...)
```

Arguments

x	fairness_radar object
...	other print parameters

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")
```

```
fradar <- fairness_radar(fobject)
print(fradar)
```

```
print.group_metric    Print group metric
```

Description

Print group metric

Usage

```
## S3 method for class 'group_metric'
print(x, ...)
```

Arguments

x	group_metric object
...	other print parameters

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) - 1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

gm <- group_metric(fobject, "TPR", "f1", parity_loss = TRUE)

print(gm)
```

print.metric_scores *Print metric scores data*

Description

Print metric scores data

Usage

```
## S3 method for class 'metric_scores'  
print(x, ...)
```

Arguments

x	metric_scores object
...	other print parameters

Examples

```
data("german")  
  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                          data = german,  
                          probability = TRUE,  
                          num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                        protected = german$Sex,  
                        privileged = "male")  
  
ms <- metric_scores(fobject, fairness_metrics = c("TPR", "STP", "ACC"))  
ms
```

```
print.performance_and_fairness
      Print performance and fairness
```

Description

Print performance and fairness

Usage

```
## S3 method for class 'performance_and_fairness'
print(x, ...)
```

Arguments

x	performance_and_fairness object
...	other print parameters

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) -1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

# same explainers with different cutoffs for female
fobject <- fairness_check(explainer_lm, explainer_rf, fobject,
                        protected = german$Sex,
                        privileged = "male",
                        cutoff = list(female = 0.4),
                        label = c("lm_2", "rf_2"))

paf <- performance_and_fairness(fobject)
```

```
paf
```

```
print.stacked_metrics Print stacked metrics
```

Description

Stack metrics sums parity loss metrics for all models. Higher value of stacked metrics means the model is less fair (has higher bias) for subgroups from protected vector.

Usage

```
## S3 method for class 'stacked_metrics'  
print(x, ...)
```

Arguments

```
x          stacked_metrics object  
...        other print parameters
```

Examples

```
data("german")  
  
y_numeric <- as.numeric(german$Risk) -1  
  
lm_model <- glm(Risk~.,  
               data = german,  
               family=binomial(link="logit"))  
  
rf_model <- ranger::ranger(Risk ~.,  
                           data = german,  
                           probability = TRUE,  
                           num.trees = 200)  
  
explainer_lm <- DALEX::explain(lm_model, data = german[,-1], y = y_numeric)  
explainer_rf <- DALEX::explain(rf_model, data = german[,-1], y = y_numeric)  
  
fobject <- fairness_check(explainer_lm, explainer_rf,  
                         protected = german$Sex,  
                         privileged = "male")  
  
sm <- stack_metrics(fobject)  
print(sm)
```

resample

*Resample***Description**

Method of bias mitigation. Similarly to `reweight` this method computes desired number of observations if the protected variable is independent from `y` and on this basis decides if this subgroup with certain class (+ or -) should be more or less numerous. Then performs oversampling or under-sampling depending on the case. If type of sampling is set to 'preferential' and probs are provided than instead of uniform sampling preferential sampling will be performed. Preferential sampling depending on the case will sample observations close to border or far from border.

Usage

```
resample(protected, y, type = "uniform", probs = NULL, cutoff = 0.5)
```

Arguments

<code>protected</code>	factor, protected variables with subgroups as levels (sensitive attributes)
<code>y</code>	numeric, vector with classes 0 and 1, where 1 means favorable class.
<code>type</code>	character, either (default) 'uniform' or 'preferential'
<code>probs</code>	numeric, vector with probabilities for preferential sampling
<code>cutoff</code>	numeric, threshold for probabilities

Value

numeric vector of indexes

References

This method was implemented based on Kamiran, Calders 2011 <https://link.springer.com/content/pdf/10.1007/s10115-011-0463-8.pdf>

Examples

```
data("german")
data <- german
data$Age <- as.factor(ifelse(data$Age <= 25, "young", "old"))
y_numeric <- as.numeric(data$Risk) -1

rf <- ranger::ranger(Risk ~., data = data, probability = TRUE, seed = 123)

u_indexes <- resample(data$Age, y = y_numeric)
rf_u <- ranger::ranger(Risk ~., data = data[u_indexes, ], probability = TRUE, seed = 123)

explainer_rf <- DALEX::explain(rf, data = data[, -1], y = y_numeric, label = "not_sampled")
explainer_rf_u <- DALEX::explain(rf_u, data = data[, -1], y = y_numeric, label = "sampled_uniform")
```

```

p_indexes <- resample(data$Age, y = y_numeric, type = "preferential", probs = explainer_rf$y_hat)
rf_p <- ranger::ranger(Risk ~., data = data[p_indexes, ], probability = TRUE, seed = 123)

explainer_rf_p <- DALEX::explain(rf_p, data = data[, -1], y = y_numeric,
                                label = "sampled_preferential")

fobject <- fairness_check(explainer_rf, explainer_rf_u, explainer_rf_p,
                        protected = data$Age,
                        privileged = "old")

fobject
plot(fobject)

```

reweight

Reweight

Description

Function returns weights for model training. The purpose of this weights is to mitigate bias in statistical parity. In fact this could potentially worsen the overall performance in other fairness metrics. This affects also model's performance metrics (accuracy).

Usage

```
reweight(protected, y)
```

Arguments

protected	factor, protected variables with subgroups as levels (sensitive attributes)
y	numeric, vector with classes 0 and 1, where 1 means favorable class.

Details

Method produces weights for each subgroup for each class. Firstly assumes that protected variable and class are independent and calculates expected probability of this certain event (that subgroup == a and class = c). Then it calculates the actual probability of this event based on empirical data. Finally the weight is quotient of those probabilities

Value

numeric, vector of weights

References

This method was implemented based on Kamiran, Calders 2011 <https://link.springer.com/content/pdf/10.1007/s10115-011-0463-8.pdf>

Examples

```

data("german")

data <- german
data$Age <- as.factor(ifelse(data$Age <= 25, "young", "old"))
data$Risk <- as.numeric(data$Risk) - 1

# training 2 models
weights <- reweight(protected = data$Age, y = data$Risk)

gbm_model <- gbm::gbm(Risk ~. , data = data)
gbm_model_weighted <- gbm::gbm(Risk ~. , data = data, weights = weights)

gbm_explainer <- DALEX::explain(gbm_model, data = data[,-1], y = data$Risk)
gbm_weighted_explainer <- DALEX::explain(gbm_model_weighted, data = data[,-1], y = data$Risk)

fobject <- fairness_check(gbm_explainer, gbm_weighted_explainer,
  protected = data$Age,
  privileged = "old",
  label = c("original", "weighted"))

# fairness check
fobject
plot(fobject)

# radar
plot(fairness_radar(fobject))

```

roc_pivot

Reject Option based Classification pivot

Description

Reject Option based Classifier is post-processing bias mitigation method. Method changes labels of favorable, privileged and close to cutoff observations to unfavorable and the opposite for unprivileged observations (changing unfavorable and close to cutoff observations to favorable, more in details). By this potentially wrongfully labeled observations are assigned different labels. Note that in y in DALEX explainer 1 should indicate favorable outcome.

Usage

```
roc_pivot(explainer, protected, privileged, cutoff = 0.5, theta = 0.1)
```

Arguments

explainer	created with explain
protected	factor, protected variables with subgroups as levels (sensitive attributes)
privileged	factor/character, level in protected denoting privileged subgroup
cutoff	numeric, threshold for all subgroups

theta numeric, variable specifies maximal euclidean distance to cutoff resulting in label switch

Details

Method implemented based on article (Kamiran, Karim, Zhang 2012). In original implementation labels should be switched. Due to specific DALEX methods probabilities ($y_{\hat{}}$) are assigned value in equal distance but other side of cutoff. The method changes explainers $y_{\hat{}}$ values in two cases.

1. When unprivileged subgroup is within (cutoff - theta, cutoff)
2. When privileged subgroup is within (cutoff, cutoff + theta)

Value

DALEX explainer with changed $y_{\hat{}}$. This explainer should be used ONLY by fairmodels as it contains unchanged predict function (changed predictions ($y_{\hat{}}$) can possibly be invisible by DALEX functions and methods).

References

Kamiran, Karim, Zhang 2012 <https://ieeexplore.ieee.org/document/6413831>/ ROC method

Examples

```
data("german")
data <- german
data$Age <- as.factor(ifelse(data$Age <= 25, "young", "old"))
y_numeric <- as.numeric(data$Risk) -1

lr_model <- stats::glm(Risk ~., data = data, family = binomial())
lr_explainer <- DALEX::explain(lr_model, data = data[, -1], y = y_numeric)

fobject <- fairness_check(lr_explainer,
                        protected = data$Age,
                        privileged = "old")

plot(fobject)

lr_explainer_fixed <- roc_pivot(lr_explainer,
                              protected = data$Age,
                              privileged = "old")

fobject2 <- fairness_check(lr_explainer_fixed, fobject,
                        protected = data$Age,
                        privileged = "old",
                        label = "lr_fixed")

fobject2
plot(fobject2)
```

stack_metrics	<i>Stack metrics</i>
---------------	----------------------

Description

Stack metrics sums parity loss metrics for all models. Higher value of stacked metrics means the model is less fair (has higher bias) for subgroups from protected vector.

Usage

```
stack_metrics(x, fairness_metrics = fairness_check_metrics())
```

Arguments

`x` object of class `fairness_object`
`fairness_metrics` character, vector of fairness metric names to include in plot. Default are ones in fairness check.

Value

`stacked_metrics` object. It contains `data.frame` with information about score for each metric and model.

Examples

```
data("german")

y_numeric <- as.numeric(german$Risk) - 1

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

rf_model <- ranger::ranger(Risk ~.,
                          data = german,
                          probability = TRUE,
                          num.trees = 200)

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

fobject <- fairness_check(explainer_lm, explainer_rf,
                        protected = german$Sex,
                        privileged = "male")

sm <- stack_metrics(fobject)
plot(sm)
```

Index

adult, [3](#)
adult_test, [4](#)
all_cutoffs, [5](#)

calculate_group_fairness_metrics, [6](#)
ceteris_paribus_cutoff, [6](#)
choose_metric, [8](#)
compas, [9](#)
confusion_matrix, [10](#)

disparate_impact_remover, [11](#)

expand_fairness_object, [12](#)
explain, [14](#), [24](#), [41](#), [56](#)

fairness_check, [13](#)
fairness_heatmap, [16](#)
fairness_pca, [18](#)
fairness_radar, [19](#)

german, [20](#)
group_matrices, [21](#)
group_metric, [22](#)
group_model_performance, [24](#)

metric_scores, [25](#)

performance_and_fairness, [26](#)
plot.all_cutoffs, [27](#)
plot.ceteris_paribus_cutoff, [28](#)
plot.chosen_metric, [29](#)
plot.fairness_heatmap, [30](#)
plot.fairness_object, [32](#)
plot.fairness_pca, [33](#)
plot.fairness_radar, [34](#)
plot.group_metric, [35](#)
plot.metric_scores, [36](#)
plot.performance_and_fairness, [37](#)
plot.stacked_metrics, [38](#)
plot_density, [39](#)
plot_fairmodels, [40](#)

pre_process_data, [42](#)
print.all_cutoffs, [43](#)
print.ceteris_paribus_cutoff, [44](#)
print.chosen_metric, [45](#)
print.fairness_heatmap, [46](#)
print.fairness_object, [47](#)
print.fairness_pca, [48](#)
print.fairness_radar, [49](#)
print.group_metric, [50](#)
print.metric_scores, [51](#)
print.performance_and_fairness, [52](#)
print.stacked_metrics, [53](#)

resample, [54](#)
reweight, [55](#)
roc_pivot, [56](#)

stack_metrics, [58](#)