Pre-processing measures

Imports

```
library(fairmodels)
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.2.0 --

## v broom        0.8.0     v recipes      0.2.0
## v dials        0.1.1     v rsample      0.1.1
## v dplyr        1.0.9     v tibble       3.1.7
## v ggplot2      3.3.6     v tidyr        1.2.0
## v infer        1.0.0     v tune         0.2.0
## v modeldata    0.1.1     v workflows    0.2.6
## v parsnip      0.2.1     v workflowsets 0.2.1
## v purrr        0.3.4     v yardstick    0.0.9

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```
library(rpart)
```

```
##
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':
##
##     prune
```

```
library(discrim)
```

```
##
## Attaching package: 'discrim'

## The following object is masked from 'package:dials':
##
##     smoothness
```

```
source("../scripts/metrics_on_dataset.R")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v readr   2.1.2     v forcats 0.5.1
## v stringr 1.4.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
```

Data

```
df <- read_rds("../data/selection.rds") %>%
  select(-rating, -gender)
```

Massaging

```
nB_df <- df %>%
  mutate(accepted = as.factor(accepted),
         index = 1:1000)
fitted_Bayes_model <- naive_Bayes() %>%
  fit(accepted ~ nationality + test_score + english_cert + extracurricular, nB_df)
nB_predictions <- predict(fitted_Bayes_model, nB_df, type="prob")
nB_df <- nB_df %>%
  mutate(prob_FALSE = nB_predictions[[".pred_FALSE"]],
         prob_TRUE = nB_predictions[[".pred_TRUE"]])
prob_true = sum(nB_df$accepted == "TRUE")/nrow(nB_df)
non_dutch_acceptance = nrow(filter(nB_df, accepted == "TRUE", nationality == "Non_Dutch")) /
  nrow(filter(nB_df, nationality == "Non_Dutch"))
print(non_dutch_acceptance)
```

```
## [1] 0.111332
```

```
while(non_dutch_acceptance < round(prob_true, 3)) {
  # Flip highest Non-Dutch
  highest_non_dutch <- nB_df %>%
    filter(accepted == "FALSE", nationality == "Non_Dutch") %>%
    arrange(desc(prob_TRUE)) %>%
    slice_head(n = 1) %>%
    select(index) %>%
    as.integer()
```

```
  # Flip lowest Dutch
  lowest_dutch <- nB_df %>%
    filter(accepted == "TRUE", nationality == "Dutch") %>%
    arrange(prob_TRUE) %>%
    slice_head(n = 1) %>%
    select(index) %>%
    as.integer()

  nB_df <- nB_df %>%
    mutate(accepted = replace(accepted, index == highest_non_dutch, "TRUE"),
           accepted = replace(accepted, index == lowest_dutch, "FALSE"))

  non_dutch_acceptance = nrow(filter(nB_df, accepted == "TRUE", nationality == "Non_Dutch")) /
    nrow(filter(nB_df, nationality == "Non_Dutch"))
  # print(non_dutch_acceptance)
}
nB_df <- nB_df %>%
  select(-index, -prob_FALSE, -prob_TRUE)

model <- decision_tree(mode = "classification")
massaging_results <- all_metrics(nB_df, model, df)
```

```
## `summarise()` has grouped output by 'nationality'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'nationality'. You can override using the
## `.groups` argument.
```

```
print_all_metrics("Massaging", massaging_results)
```

```
## [1] "Massaging"
## [1] "Group fairness"
## # A tibble: 4 x 4
##   nationality accepted total  perc
##   <fct>       <fct>    <int> <dbl>
## 1 Dutch       FALSE      416  83.7
## 2 Dutch       TRUE        81  16.3
## 3 Non_Dutch   FALSE      421  83.7
## 4 Non_Dutch   TRUE        82  16.3
## [1] "Causal discrimination"
## [1] 0
## [1] "Unawareness"
## # A tibble: 4 x 4
##   nationality predicted_accepted total  perc
##   <fct>       <fct>              <int> <dbl>
## 1 Dutch       FALSE                416  83.7
## 2 Dutch       TRUE                  81  16.3
## 3 Non_Dutch   FALSE                409  81.3
## 4 Non_Dutch   TRUE                  94  18.7
```

Reweighting

```
weights <- reweight(df$nationality, as.numeric(df$accepted))
weighted_model <- rpart(accepted ~ nationality + test_score + english_cert + extracurricular,
      df, weights)
weighted_df <- df %>%
  mutate(predicted = as.logical(predict(weighted_model, df)))
print("Reweighting")
```

```
## [1] "Reweighting"
```

```
print("Group fairness")
```

```
## [1] "Group fairness"
```

```
print(group_fairness(weighted_df, nationality, predicted)[[1]])
```

```
## `summarise()` has grouped output by 'nationality'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 4 x 4
##   nationality predicted total  perc
##   <fct>       <lgl>     <int> <dbl>
## 1 Dutch       FALSE       390  78.5
## 2 Dutch       TRUE        107  21.5
## 3 Non_Dutch   FALSE       447  88.9
## 4 Non_Dutch   TRUE         56  11.1
```

```
print("Causal Discrimination")
```

```
## [1] "Causal Discrimination"
```

```
pop_size <- nrow(df)
# Flip nationalities
inverted_df <- df %>%
  mutate(nationality = ifelse(nationality == "Dutch", "Non_Dutch", "Dutch"))
# Add prediction column
eval_df <- df %>%
  mutate(inverted_accepted = as.logical(predict(weighted_model, inverted_df)),
         different = accepted != inverted_accepted)
print(sum(eval_df$different)/pop_size)
```

```
## [1] 0.123
```

```
print("Unawareness")
```

```
## [1] "Unawareness"
```

```
weighted_unaware_model <- rpart(accepted ~ test_score + english_cert + extracurricular, df, weights)
weighted_unaware_df <- df %>%
  mutate(predicted = as.logical(predict(weighted_unaware_model, df)))
print(group_fairness(weighted_unaware_df, nationality, predicted)[[1]])
```

```
## `summarise()` has grouped output by 'nationality'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 4 x 4
##   nationality predicted total  perc
##   <fct>       <lgl>     <int> <dbl>
## 1 Dutch       FALSE       390  78.5
## 2 Dutch       TRUE        107  21.5
## 3 Non_Dutch   FALSE       379  75.3
## 4 Non_Dutch   TRUE        124  24.7
```

Resampling

```
resample_indexes <- resample(df$nationality, as.numeric(df$accepted))
resampled_df <- df[resample_indexes, ]
model <- decision_tree(mode = "classification")
resamping_results <- all_metrics(resampled_df, model, df)
```

```
## `summarise()` has grouped output by 'nationality'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'nationality'. You can override using the
## `.groups` argument.
```

```
print_all_metrics("Resampling", resamping_results)
```

```
## [1] "Resampling"
## [1] "Group fairness"
## # A tibble: 4 x 4
##   nationality accepted total  perc
##   <fct>       <lgl>    <int> <dbl>
## 1 Dutch       FALSE      416  83.7
## 2 Dutch       TRUE        81  16.3
## 3 Non_Dutch   FALSE      421  83.7
## 4 Non_Dutch   TRUE        82  16.3
## [1] "Causal discrimination"
## [1] 0.123
## [1] "Unawareness"
## # A tibble: 4 x 4
##   nationality predicted_accepted total  perc
##   <fct>       <fct>              <int> <dbl>
## 1 Dutch       FALSE                402  80.9
## 2 Dutch       TRUE                  95  19.1
## 3 Non_Dutch   FALSE                435  86.5
## 4 Non_Dutch   TRUE                  68  13.5
```