DAR F22 Project Fairness Auditor Notebook Template

Fairness Auditor

Jonathan Li

9/12/2022

Contents

Weekly Work Summary

NOTE: Follow an outline format; use bullets to express individual points.

- RCS ID: LIJ47
- Project Name: ML Fairness
- Summary of work since last week

  Completed Assignment 1 and 2
- NEW: Summary of github issues added and worked

  None
- Summary of github commits https://github.rpi.edu/DataINCITE/DeFi_DAR_F22/commit/801442ae5c66ef53ab3da7278663e552a7480f00-assigment1
- List of presentations, papers, or other outputs

  None
- List of references (if necessary)

  None
- Indicate any use of group shared code base

  None
- Indicate which parts of your described work were done by you or as part of joint efforts

  None

Personal Contribution

Finishing Assignment1 and 2. Found R packages for fairness and bias mitigation

Discussion of Primary Findings

I compared reweighing model with a baseline model and found that it had a slight improvement for fairness.

Required: Provide illustrating figures and/or tables

I have some figures in the bottom of this notebook showing the results of the fairness metrics I used.

Evaluating Bias Mitigation Algorithms

This notebook includes the steps to 1) Process the data before training a Machine Learning model 2) Split the data into train-test 3) Train a Machine Learning model (without bias mitigation) - BaselineModel 4) Evaluate the utility and fairness metrics of BaselineModel 5) Train a Machine Learning model with a bias mitigation algorithm (Reweighing) - ReweighingModel 6) Evaluate the utility and fairness metrics of this ReweighingModel 7) Compare the scores for the two models

Load required libraries

We load the required libraries for this project.

1) Process data

Load the required dataset file. The dataset is the Adult Income dataset. We are predicting whether the outcome variable `income`, having two classes: (a) >50K or (b) <=50K. We use the protected attribute as `gender`. It has two values: (a) Female and (b) Male.

```r
# Read data
filename <- "../data_files/dataset.csv"
df <- read.csv(filename)

# Look at the top rows
head(df)
```

```
##   age workclass fnlwgt    education educational.num     marital.status
## 1  25   Private 226802         11th               7      Never-married
## 2  38   Private  89814      HS-grad               9 Married-civ-spouse
## 3  28 Local-gov 336951   Assoc-acdm              12 Married-civ-spouse
## 4  44   Private 160323 Some-college              10 Married-civ-spouse
## 5  18         ? 103497 Some-college              10      Never-married
## 6  34   Private 198693         10th               6      Never-married
##          occupation  relationship  race gender capital.gain capital.loss
## 1 Machine-op-inspct     Own-child Black   Male            0            0
## 2   Farming-fishing       Husband White   Male            0            0
## 3   Protective-serv       Husband White   Male            0            0
## 4 Machine-op-inspct       Husband Black   Male         7688            0
## 5                 ?     Own-child White Female            0            0
## 6     Other-service Not-in-family White   Male            0            0
##   hours.per.week native.country income
## 1             40  United-States  <=50K
## 2             50  United-States  <=50K
## 3             40  United-States   >50K
## 4             40  United-States   >50K
## 5             30  United-States  <=50K
## 6             30  United-States  <=50K
```

The data can be processed to make it suitable for training Machine Learning models such as removing rows with missing values, removing repeated columns etc.

```r
# Convert marital-status to simpler categories
# If marital.status is either never-married, divorced, separated, widowed or single,
# the assigned value is 0 else 1
df$marital.status <- ifelse((df$marital.status == "Never-married") |
                                       (df$marital.status == "Divorced") |
                                       (df$marital.status == "Separated") |
                                       (df$marital.status == "Widowed") |
                                       (df$marital.status == "Single"), 0, 1)

# Remove rows with missing values (denoted by ?)
df[df == '?'] <- NA
df <- na.omit(df)

# Convert categorical columns to numerical and then change to integer type
df$gender <- ifelse(df$gender == "Male", 1, 0)
df$income <- ifelse(df$income == ">50K", 1, 0)

# Drop extra columns not to be used for model training
df <- subset(df, select = -c(`education`, `age`, `hours.per.week`, `fnlwgt`,
                       `capital.gain`, `capital.loss`, `native.country`))

# One-hot encode categorical columns
df$workclass <- as.factor(df$workclass)
df$occupation <- as.factor(df$occupation)
df$relationship <- as.factor(df$relationship)
df$race <- as.factor(df$race)
df <- one_hot(as.data.table(df))

# Save processed data
saved_filename <- "../data_files/processed_dataset.csv"
write.csv(df, saved_filename, row.names = FALSE)
```

2) Split data into train-test

We begin by splitting the data into train-test split.

```r
# Set seed for reproducibility
set.seed(0)

# Split data into train-test
# 70% data to be used for training
# 30% data to be used for testing

# Get indices
training_size <- floor(0.7*nrow(df))
train_ind <- sample(seq_len(nrow(df)), size = training_size)

# Split data
names(df) <- make.names(names(df))
train.raw <- df[train_ind, ]
test.raw <- df[-train_ind, ]

# Scale train-test data
# except for income and gender
```

```r
pp = preProcess(subset(train.raw, select = -c(`gender`, `income`)))
train.scale <- predict(pp, subset(train.raw, select = -c(`gender`, `income`)))
test.scale <- predict(pp, subset(test.raw, select = -c(`gender`, `income`)))

# Attach income and gender to scaled data
train.scale$income <- train.raw$income
test.scale$income <- test.raw$income
train.scale$gender <- train.raw$gender
test.scale$gender <- test.raw$gender
```

3) Train a ML model - BaselineModel

Once the data is split, we train a Logistic Regression model on the training data. `income` is used as the outcome variable.

```r
# Train a Logistic Regression model
baselineModel <- glm(income ~ ., data = train.scale, family = binomial)

# Get prediction on test data
baselineModel.prob <- predict(baselineModel, test.scale, type = 'response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
baselineModel.pred <- ifelse(baselineModel.prob > 0.5, 1, 0)
```

4) Evaluate utility and fairness metrics for BaselineModel

First, the utility is evaluated using Balanced Accuracy. Balanced Accuracy measures the accuracy for both classes of an outcome variable. We use the function `bacc()` to evaluate balanced accuracy.

```r
# Calculate Balanced Accuracy
baselineModel.bal_acc <- bacc(as.factor(test.scale$income), as.factor(baselineModel.pred))
```

Next, the fairness is evaluated for the given model. The fairness is evaluated for Gender protected attribute with two classes: Male and Female. We calculate Equalized Odds. Link: https://kozodoi.me/r/fairness/packages/2020/05/01/fairness-tutorial.html

```r
# Create a copy of the dataset for fairness evaluation
test2 <- test.scale
test2$prob <- baselineModel.prob
test2$income <- as.factor(test2$income)
test2$gender <- as.factor(test2$gender)

# Evaluate TPR difference
# NOTE: In the library `fairness`, Equalized Odds is defined as separation which is
# the TPR difference only. This is not the same Equalized Odds calculated here.
tpr_results <- equal_odds(data         = test2,
                          outcome      = 'income',
                          outcome_base = '0',
                          group        = 'gender',
                          probs        = 'prob',
                          cutoff       = 0.5,
                          base         = '0')
```

```
## Warning in equal_odds(data = test2, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
tpr_diff <- tpr_results$Metric[1] - tpr_results$Metric[4]

# Evaluate FPR difference
fpr_results <- fpr_parity(data         = test2,
                          outcome      = 'income',
                          outcome_base = '0',
                          group        = 'gender',
                          probs        = 'prob',
                          cutoff       = 0.5,
                          base         = '0')
```

```
## Warning in fpr_parity(data = test2, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
fpr_diff <- fpr_results$Metric[1] - fpr_results$Metric[4]

# Evaluate Equalized Odds (EO)
# We define equalized odds as discussed in class
baselineModel.eo <- max(abs(tpr_diff), abs(fpr_diff))
```

5) Train ML model with Reweighing

We train a Logsitic Regression model similar to step 3 while also applying Reweighing bias mitigation algorithm.

```r
# Apply reweighing before model training
# Get weights during Reweighing
reweighing_weights <- reweight(train.scale$gender, train.scale$income)
```

```
##
## changing protected to factor
```

```r
# Train a Logistic Regression model
reweighingModel <- glm(income ~ ., data = train.scale, family = binomial, weights = reweighing_weights)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

```r
# Get prediction on test data
reweighingModel.prob <- predict(reweighingModel, test.scale, type = 'response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
reweighingModel.pred <- ifelse(reweighingModel.prob > 0.5, 1, 0)
```

6) Evaluate utility and fairness metrics for ReweighingModel

Similar to step 4, evaluate balanced accuracy and equalized odds.

```r
# Calculate Balanced Accuracy
reweighingModel.bal_acc <- bacc(as.factor(test.scale$income), as.factor(reweighingModel.pred))

# Create a copy of the dataset for fairness evaluation
test3 <- test.scale
test3$prob <- reweighingModel.prob
test3$income <- as.factor(test3$income)
test3$gender <- as.factor(test3$gender)
```

```r
# Evaluate TPR difference
tpr_results <- equal_odds(data         = test3,
                       outcome      = 'income',
                       outcome_base = '0',
                       group        = 'gender',
                       probs        = 'prob',
                       cutoff       = 0.5,
                       base         = '0')
```

```
## Warning in equal_odds(data = test3, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
tpr_diff <- tpr_results$Metric[1] - tpr_results$Metric[4]

# Evaluate FPR difference
fpr_results <- fpr_parity(data         = test3,
                       outcome      = 'income',
                       outcome_base = '0',
                       group        = 'gender',
                       probs        = 'prob',
                       cutoff       = 0.5,
                       base         = '0')
```

```
## Warning in fpr_parity(data = test3, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
fpr_diff <- fpr_results$Metric[1] - fpr_results$Metric[4]

# Evaluate Equalized Odds (EO)
reweighingModel.eo <- max(abs(tpr_diff), abs(fpr_diff))
```

7) Compare the scores for the two models

We compare the Balanced Accuracy and Equalized Odds scores.

```r
print("# Balanced Accuracy scores")
```

```
## [1] "# Balanced Accuracy scores"
```

```r
print(paste0("Baseline model: ", baselineModel.bal_acc))
```

```
## [1] "Baseline model: 0.731132055066914"
```

```r
print(paste0("Reweighing model: ", reweighingModel.bal_acc))
```

```
## [1] "Reweighing model: 0.71559050053529"
```

```r
print("# Equalized Odds scores")
```

```
## [1] "# Equalized Odds scores"
```

```r
print(paste0("Baseline model: ", baselineModel.eo))
```

```
## [1] "Baseline model: 0.174426399671302"
```

```r
print(paste0("Reweighing model: ", reweighingModel.eo))
```

```
## [1] "Reweighing model: 0.108108254222261"
```

Finding: As a higher Balanced Accuracy score is better, Baseline model has a better performance than the Reweighing. On the other hand, a lower Equalized Odds score is better, Reweighing model is better. A model with Equalized Odds less than or equal to 0.1 is considered to be fair. So, Reweighing model is very close to being fair.

TODO

1) Train another ML model

TODO: Try to applying another classification Machine Learning model of your choice. Evaluate Balanced Accuracy and Equalized Odds on the generated model.

```r
if (!require("xgboost")) {
  install.packages("xgboost")
  library(xgboost)
}
```

```
## Loading required package: xgboost

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
# 1. Train ML model here on train data
train_features <- train.scale
train_features$income <- NULL
train_labels <- train.scale$income
xgb <- xgboost(data = as.matrix(train_features), label = data.matrix(train_labels), max_depth = 30, eta

# 2. Get predictions on test data
test_features <- test.scale
test_features$income <- NULL
test_labels <- test.scale$income
xgb.prob <- predict(xgb, as.matrix(test_features), type = "response")
xgb.pred <- ifelse(xgb.prob > 0.5, 1, 0)

# 3. Evaluate balanced accuracy
xgb.bal_acc <- bacc(as.factor(test.scale$income), as.factor(xgb.pred))

# 4. Evaluate equalized odds
test4 <- test.scale
test4$prob <- xgb.prob
test4$income <- as.factor(test4$income)
test4$gender <- as.factor(test4$gender)
XGB_tpr_results <- equal_odds(data = test4,
outcome = 'income',
outcome_base = '0',
group = 'gender',
probs = 'prob',
cutoff = 0.5,
base = '0')
```

```
## Warning in equal_odds(data = test4, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
XGB_tpr_diff <- XGB_tpr_results$Metric[1] - XGB_tpr_results$Metric[4]

XGB_fpr_results <- fpr_parity(data = test4,
outcome = 'income',
outcome_base = '0',
group = 'gender',
probs = 'prob',
cutoff = 0.5,
base = '0')
```

```
## Warning in fpr_parity(data = test4, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
XGB_fpr_diff <- XGB_fpr_results$Metric[1] - XGB_fpr_results$Metric[4]
xgb.eo <- max(abs(XGB_tpr_diff), abs(XGB_fpr_diff))

# 5. Compare results with baselineModel and reweighingModel
```

```r
print("XGB Balanced Accuracy score")
```

```
## [1] "XGB Balanced Accuracy score"
```

```r
print(paste0("XGB model: ", xgb.bal_acc))
```

```
## [1] "XGB model: 0.730778653579793"
```

```r
print("Baseline Model Balanced Accuracy score")
```

```
## [1] "Baseline Model Balanced Accuracy score"
```

```r
print(paste0("Baseline model: ", baselineModel.bal_acc))
```

```
## [1] "Baseline model: 0.731132055066914"
```

```r
print("Reweighing Model Balanced Accuracy score")
```

```
## [1] "Reweighing Model Balanced Accuracy score"
```

```r
print(paste0("Reweighing model: ", reweighingModel.bal_acc))
```

```
## [1] "Reweighing model: 0.71559050053529"
```

```r
print("XGB Equalized Odds score")
```

```
## [1] "XGB Equalized Odds score"
```

```r
print(paste0("XGB model: ", xgb.eo))
```

```
## [1] "XGB model: 0.143478472417447"
```

```r
print("Baseline Equalized Odds score")
```

```
## [1] "Baseline Equalized Odds score"
```

```r
print(paste0("Baseline model: ", baselineModel.eo))
```

```
## [1] "Baseline model: 0.174426399671302"
```

```r
print("Reweighing Equalized Odds score")
```

```
## [1] "Reweighing Equalized Odds score"
```

```r
print(paste0("Reweighing model: ", reweighingModel.eo))
```

```
## [1] "Reweighing model: 0.108108254222261"
```

2) Evaluate other fairness metrics

TODO: Identify two other fairness metrics apart from Equalized Odds and evaluate them on two ML models: BaselineModel and ReweighingModel.

Here's resources for alternate metrics:

- https://kozodoi.me/r/fairness/packages/2020/05/01/fairness-tutorial.html
- https://github.com/Trusted-AI/AIF360/tree/master/aif360/aif360-r

```r
# 1. List the name of the two fairness metrics
# Mathews Correlation Coefficient (MCC), Specificity

# 2. Measure them on baselineModel and reweighingModel

mccResult <- mcc_parity(data = test2,
outcome = 'income',
outcome_base = '0',
group = 'gender',
probs = 'prob',
cutoff = 0.5,
base = '0')
```

```
## Warning in mcc_parity(data = test2, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
baselineModel.mcc <- mccResult$Metric

mccResult <- mcc_parity(data = test3,
outcome = 'income',
outcome_base = '0',
group = 'gender',
probs = 'prob',
cutoff = 0.5,
base = '0')
```

```
## Warning in mcc_parity(data = test3, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
reweighingModel.mcc <- mccResult$Metric

specResult <- spec_parity(data = test2,
outcome = 'income',
outcome_base = '0',
group = 'gender',
probs = 'prob',
cutoff = 0.5,
base = '0')
```

```
## Warning in spec_parity(data = test2, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
baselineModel.s <- specResult$Metric

specResult <- spec_parity(data = test3,
outcome = 'income',
outcome_base = '0',
group = 'gender',
probs = 'prob',
cutoff = 0.5,
base = '0')
```

```
## Warning in spec_parity(data = test3, outcome = "income", outcome_base = "0", :
## Converting data.table to data.frame
```

```r
reweighingModel.s <- specResult$Metric
```

```r
# 3. Compare the scores and explain what you find
print("# BaselineModel Mathews Correlation Coefficient score")
```

```
## [1] "# BaselineModel Mathews Correlation Coefficient score"
```

```r
print(baselineModel.mcc)
```

```
##                    0           1
## MCC           0.4884039    0.4878643
## MCC Parity    1.0000000    0.9988952
## Group size 4415.0000000 9152.0000000
```

```r
print("# ReweighingModel Mathews Correlation Coefficient score")
```

```
## [1] "# ReweighingModel Mathews Correlation Coefficient score"
```

```r
print(reweighingModel.mcc)
```

```
##                    0           1
## MCC           0.5330459    0.4701065
## MCC Parity    1.0000000    0.8819251
## Group size 4415.0000000 9152.0000000
```

```r
print("# BaselineModel Specificity scores")
```

```
## [1] "# BaselineModel Specificity scores"
```

```r
print(baselineModel.s)
```

```
##                            0           1
## Specificity           0.9799537    0.8925411
## Specificity Parity    1.0000000    0.9107992
## Group size         4415.0000000 9152.0000000
```

```r
print("# ReweighingModel Specificity scores")
```

```
## [1] "# ReweighingModel Specificity scores"
```

```r
print(reweighingModel.s)
```

```
##                            0           1
## Specificity           0.9442303    0.9255689
## Specificity Parity    1.0000000    0.9802364
## Group size         4415.0000000 9152.0000000
```

Finding: For MCC score, the higher the better. The MCC score for females was higher and males was slightly lower in the reweighing model compared to the baseline model. Since the improvement is higher than the drawback, it can be said that reweighing improves fairness using MCC score as a metric. For specificity score, the higher the better. Both the baseline and reweighing model have high specificity scores for male and female classes, so both models are fair using this metric.

3) List fairness libraries

TODO: Find a list of libraries in R that include fairness metrics/methods or mitigation techniques. Keywords to look for: fairness, bias, bias mitigation, fairness mitigation. Make a summary of what you find.

Finding:

https://github.com/ModelOriented/FairModels-provides fairness and bias metrics

https://github.com/matloff/EDFfair-fairness mitgation tool

https://cran.r-project.org/web/packages/predfairness/index.html-discrimination mitigation using reject option based classification

https://cran.r-project.org/web/packages/mlr3fairness/index.html-fairness auditing, bias mitigation using reweighing, fairness metrics

https://cran.r-project.org/web/packages/EValue/index.html-sensitivity analysis, selection bias

4) Be prepared to discuss your findings in class (2-3 minutes)