DAR F22 Project Fairness Auditor Notebook Template

Fairness Auditor

Daniel Kopp

September 13 2022

Contents

Weekly Work Summary

- RCS ID: koppd
- Project Name: ML Fairness
- Summary of work since last week
  - Completed this notebook as introduction into machine learning with R
  - Chose the task of generating new synthetic data set using a cGAN
  - Registered for CITI program to gain access to Mimic IV dataset but mentioned course did not exist

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.

```
# 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.

```
# 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.

```
# 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
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.

```
# 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
```

```
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
```

```
# 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!
```

```
# 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
```

```
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.

```
# 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)

# 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
```

```
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
```

```
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.

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

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

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

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

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

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

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

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

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

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

```
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.

1) Train another ML model

```
library(e1071)
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:mltools':
##
##     skewness
```

```
# 1. Train ML model here on train data
svmModel = svm(income~., data=train.scale, kernel='radial')

# 2. Get predictions on test data
svmModel.prob <- predict(svmModel, test.scale, type='response')
svmModel.pred <- ifelse(svmModel.prob > 0.5, 1, 0)

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


# 4. Evaluate equalized odds

test4 <- test.scale
test4$prob <- svmModel.prob
test4$income <- as.factor(test4$income)
```

```r
test4$gender <- as.factor(test4$gender)

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

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
fpr_diff <- fpr_results$Metric[1] - fpr_results$Metric[4]
svmModel.eo <- max(abs(tpr_diff), abs(fpr_diff))


# 5. Compare results with baselineModel and reweighingModel
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(paste0("SVM model: ", svmModel.bal_acc))
```

```
## [1] "SVM model: 0.721234168250558"
```

```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"
```

```
print(paste0("SVM model: ", svmModel.eo))
```

```
## [1] "SVM model: 0.215467746469736"
```

2) Evaluate other fairness metrics

```
# 1. List the name of the two fairness metrics
# Here I used proportional parity and accuracy parity

# 2. Measure them on baselineModel and reweighingModel
baselineModel.prop_parity <- prop_parity(data = test2,
                     outcome      = 'income',
                     outcome_base = '0',
                     group        = 'gender',
                     probs        = 'prob',
                     cutoff       = 0.5,
                     base         = '0')
```

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

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

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

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

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

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

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

```
# 3. Compare the scores and explain what you find
print("# Proportional parity scores")
```

```
## [1] "# Proportional parity scores"
```

```
baseline_pp_score <- baselineModel.prop_parity$Metric[1] - baselineModel.prop_parity$Metric[4]
reweighting_pp_score <- reweighingModel.prop_parity$Metric[1] - reweighingModel.prop_parity$Metric[4]
print(paste0("Baseline model: ", baseline_pp_score))
```

```
## [1] "Baseline model: -0.184378588568849"
```

```
print(paste0("Reweighing model: ", reweighting_pp_score))
```

```
## [1] "Reweighing model: -0.0809262615923148"
```

```
print("# Accuracy parity scores")
```

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

```
baseline_acc_score <- baselineModel.acc_parity$Metric[1] - baselineModel.acc_parity$Metric[4]
reweighting_acc_score <- reweighingModel.acc_parity$Metric[1] - reweighingModel.acc_parity$Metric[4]
print(paste0("Baseline model: ", baseline_acc_score))
```

```
## [1] "Baseline model: 0.118768858548021"
```

```
print(paste0("Reweighing model: ", reweighting_acc_score))
```

```
## [1] "Reweighing model: 0.113580951183584"
```

I have used proportional parity, which measures the proportion of positive predictions in each protected group, and accuracy parity, which measures the difference in accuracy across protected groups.We can see based on the results above the scores are both negative meaning that the males are more represented in the data. When the model is reweighed it lowers the magnitude of this score by a significant amount (~0.1) which is to be expected as this is one of the direct results of the reweighing. For the accuracy parity score, the reweighing has nearly no affect on the outcome as the outcome for men isstill predicted significantly more accurately. This means reweighing is not enoughof a mitigation strategy for balencing accuracy across protected groups.

Finding:

3) List fairness libraries

- **AI Fairness 360** is a bias mitigation library that aims to provide bias mitigation algorithms and metrics to detect bias
- **Fairmodels** is a library that contains many bias detection algorithms / metrics, bias visualization tools, and pre / post processing for bias mitigation
- **Fairness** provides a set of metrics and visualization tools to aid in bias auditing

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