DAR F22 Project Fairness Auditor HW2 Notebook

David Qiu

9/12/22

Contents

Weekly Work Summary

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

- RCS ID: qiud
- Project Name: ML Fairness
- Summary of work since last week
  - Looked over the datasets and code.
  - Trained and tested this dataset with a SVM classifier.
  - Evaluated on two new fairness metrics.
  - Found multiple packages for bias mitigation and fairness evaluation.
- NEW: Summary of github issues added and worked
  - None
- Summary of github commits
  - branch name: qiud-hw2
  - The commit for this hw
- List of presentations, papers, or other outputs
  - N/A
- List of references (if necessary)
- Indicate any use of group shared code base
- Indicate which parts of your described work were done by you or as part of joint efforts

Personal Contribution

- Familarized myself with the datasets and completed assignment 2.
- Found various packages for fairness evaluation.

Discussion of Primary Findings

- Discuss primary findings:
  - I researched various methods of evaluating fairness and wanted to know how well the reweighing model would perform on the negative predicted value parity and specificity parity values.
  - Unexpectedly, the reweighing model performed worse than the baseline model on the specificity parity metric.
- **Required:** Provide illustrating figures and/or tables
  - The links to researched metrics are located in part 3 of this notebook.

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.

```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
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)

# 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
```

```
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.

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.

```
# 1. Train ML model here on train data
newModel <- svm(income ~ ., data = train.scale, family = binomial)

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

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

# 4. Evaluate equalized odds

#make a copy of the dataset
test4 <- test.scale
test4$prob <- newModel.prob
test4$income <- as.factor(test4$income)
test4$gender <- as.factor(test4$gender)

#Evaluate TPR
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
```

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

#Evaluate FPR
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]

#Obtain equalized odds
newModel.eo <- max(abs(tpr_diff), abs(fpr_diff))

# 5. Compare results with baselineModel and reweighingModel
print(paste("Balanced Accuracy: Baseline:", baselineModel.bal_acc, "Reweighing:", reweighingModel.bal_a
```

```
## [1] "Balanced Accuracy: Baseline: 0.731132055066914 Reweighing: 0.71559050053529 New Model 0.7212341
```

```r
print(paste("Equalized Odds: Baseline:", baselineModel.eo, "Reweighing:", reweighingModel.eo, "New Model
```

```
## [1] "Equalized Odds: Baseline: 0.174426399671302 Reweighing: 0.108108254222261 New Model: 0.21546774
```

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

#The two fairness metrics to be used are negative predicted value parity(npv) and
#specificity parity (spec_parity).

# 2. Measure them on baselineModel and reweighingModel

bl_spec_result <- 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
bl_spec_diff <- bl_spec_result$Metric[1] - bl_spec_result$Metric[4]

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

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

```
bl_npv_diff <- bl_npv_result$Metric[1] - bl_npv_result$Metric[4]

rw_spec_result <- 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
```

```
rw_spec_diff <- rw_spec_result$Metric[1] - rw_spec_result$Metric[4]

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

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

```
rw_npv_diff <- rw_npv_result$Metric[1] - rw_npv_result$Metric[4]

# 3. Compare the scores and explain what you find
print(paste("Negative Predicted Value Parity: Baseline:", bl_npv_diff, "Reweighing:", rw_npv_diff))
```

```
## [1] "Negative Predicted Value Parity: Baseline: 0.101644168603244 Reweighing: 0.14468854658868"
```

```
print(paste("Specificity Parity: Baseline:", bl_spec_diff, "Reweighing:", rw_spec_diff))
```

```
## [1] "Specificity Parity: Baseline: 0.0874126521672595 Reweighing: 0.0186613748671527"
```

Findings: Evaluating on the two new metrics Negative Predicted Value Parity and Specificity Parity, the outcome reveals that the reweighing model has a higher negative predicted value parity but a lower specificity parry.

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.

Findings:

fairmodels R Package - Determines fairness based on twelve metrics.

- Offers various ways of visualizing fairness and bias.
- https://github.com/ModelOriented/fairmodels

AIF360 R package

- Contains Individual and Group Fairness Metrics
- Pre-processing, In-processing, and Post-processing algorithms to mitigate bias.
- https://www.datascienceseed.com/wp-content/uploads/2020/10/An-Open-Source-Toolkit-for-R-to-Mitigate-Discriminat pdf

Brglm/Brglm2 packages

- Contains various methods for bias reduction
- https://cran.r-project.org/web/packages/brglm2/index.html

Various other CRAN Packages

- https://cran.r-project.org/web/packages/BUCSS/index.html - Corrects publication Bias
- https://cran.r-project.org/web/packages/DDL/index.html - Debiased Lasso Regression
- https://cran.r-project.org/web/packages/dipw/index.html - Debiased Score Weighting
- https://cran.r-project.org/web/packages/eive/index.html - Reduce in-variable bias in linear regression.
- https://cran.r-project.org/web/packages/EValue/index.html - Sensitivity analysis for bias.
- https://cran.r-project.org/web/packages/ibr/index.html - Iterative Bias Reduction
- https://cran.r-project.org/web/packages/informedSen/index.html - Tests for Bias
- https://cran.r-project.org/web/packages/mbrglm/index.html - Median bias reduction.
- https://cran.r-project.org/web/packages/MethodCompare/index.html - Bias assessment plots.
- https://cran.r-project.org/web/packages/mlr3fairness/index.html - Fairness auditing tools and bias mitigation techniques

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