# DAR F22 Project Fairness Auditor Notebook

Fairness Auditor

Lydia Halter

9/14/2022

## Contents

### Weekly Work Summary

- RCS ID: haltel
- Project Name: ML Fairness
- Summary of work since last week
  - Walked through training of a logistic regression model and evaluated its (and the reightweighed model's) fairness using equalized odds.
  - Created my own model to train (decision tree) and compared its EO fairness.
  - Found two other fairness metrics (false negative rate and false positive rate) and applied them to the baseline and reweighed models
  - Did some light research on fairness metrics and their applications
- Summary of github issues added and worked
  - N/A
- Summary of github commits
  - haltel-14sep2022
  - committed haltel-assignment2.Rmd, haltel-assignment2.pdf, and haltel-assignment2.html
- List of references (if necessary)
  - https://www.guru99.com/r-decision-trees.html
  - https://www.titanlakeml.com/tour?gclid=CjwKCAjwsfuYBhAZEiwA5a6CDOk9RGppFayI_j2ETQtELUC6xcUkF1dLQetLylGOulIhPpFap4W7VRoC2wkQAvD_BwE
  - https://developer.ibm.com/blogs/the-aif360-team-adds-compatibility-with-r/

### Personal Contribution

- I added the code and new text for the 3 sections under "TODO"

## Discussion of Primary Findings

- Discuss primary findings:
  - I wanted to know how well a decision tree model compares to a logistic regression.
  - I examined the logistic regression's balanced accuracy and equalized odds, then trained my created model whose BA and EO values I then compared. I found that with regard to BA values, my model had the lowest value i.e. it was of lesser-fit, but regarding EO, it had a median value.

## 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.

```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, ]
```

```r
# 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
```

```
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
```

```
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.

```
# 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.

## 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
library(rpart)
# training a decision tree model
treeModel <- rpart(income~., data = train.scale)
```

```
# 2. Get prediction on test data
treeModel.prob <- predict(treeModel, test.scale)
treeModel.pred <- ifelse(treeModel.prob > 0.5, 1, 0)
```

```
# 3. Evaluate Balanced Accuracy
treeModel.bal_acc <- bacc(as.factor(test.scale$income), as.factor(treeModel.pred))
```

```
# 4. Evaluate Equalized Odds (EO)
# Create a copy of the dataset for fairness evaluation
new_test2 <- test.scale
new_test2$prob <- treeModel.prob
new_test2$income <- as.factor(new_test2$income)
new_test2$gender <- as.factor(new_test2$gender)

tpr_results2 <- equal_odds(data         = new_test2,
                      outcome      = 'income',
                      outcome_base = '0',
                      group        = 'gender',
                      probs        = 'prob',
                      cutoff       = 0.5,
                      base         = '0')
```

```
## Warning in equal_odds(data = new_test2, outcome = "income", outcome_base =
## "0", : Converting data.table to data.frame
```

```
tpr_diff2 <- tpr_results2$Metric[1] - tpr_results2$Metric[4]

# Evaluate FPR difference
fpr_results2 <- fpr_parity(data         = new_test2,
                      outcome      = 'income',
                      outcome_base = '0',
                      group        = 'gender',
                      probs        = 'prob',
                      cutoff       = 0.5,
                      base         = '0')
```

```
## Warning in fpr_parity(data = new_test2, outcome = "income", outcome_base =
## "0", : Converting data.table to data.frame
```

```
fpr_diff2 <- fpr_results2$Metric[1] - fpr_results2$Metric[4]

# We define equalized odds as discussed in class
treeModel.eo <- max(abs(tpr_diff2), abs(fpr_diff2))
```

```
# 5. Compare results with baselineModel and reweighingModel
print("# Balanced Accuracy scores")
```

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

```
print(paste0("Decision Tree model: ", treeModel.bal_acc))
```

```
## [1] "Decision Tree model: 0.674509426358449"
```

```
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("Decision Tree model: ", treeModel.eo))
```

```
## [1] "Decision Tree model: 0.130033194213178"
```

```
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, which has a better performance than the Decision Tree model. On the other hand, a lower Equalized Odds score is better, so the Reweighing model is best. The second best model for an Equalized Odds score is the Decision Tree model, followed by the Baseline model. 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 compared to the relatively higher values of the other two.

*Resources* * https://www.guru99.com/r-decision-trees.html * https://www.titanlakeml.com/tour?gclid=CjwKCAjwsfuYBhAZEiwA5a6CDOk9RGppFayI_j2ETQtELUC6xcUkF1dLQetLylGOulIhPpFap4W7VRoC2wkQAvD_BwE

### 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

```
# 1. List the name of the two fairness metrics
# False Negative Rate and False Positive Rate
```

```r
# 2. Measure them on baselineModel and reweighingModel
library(fairness)
#false negative
fnr_results2 <- fnr_parity(data         = test2,
                      outcome      = 'income',
                      outcome_base = '0',
                      group        = 'gender',
                      probs        = 'prob',
                      cutoff       = 0.5,
                      base         = '0')
```

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

```r
fnr_diff2 <- fnr_results2$Metric[1] - fnr_results2$Metric[4]

fnr_results3 <- fnr_parity(data         = test3,
                      outcome      = 'income',
                      outcome_base = '0',
                      group        = 'gender',
                      probs        = 'prob',
                      cutoff       = 0.5,
                      base         = '0')
```

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

```r
fnr_diff3 <- fnr_results3$Metric[1] - fnr_results3$Metric[4]

#false positive
fpr_results2 <- 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_diff2 <- fpr_results2$Metric[1] - fpr_results2$Metric[4]

fpr_results3 <- 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_diff3 <- fpr_results3$Metric[1] - fpr_results3$Metric[4]
```

```r
# 3. Compare the scores and explain what you find
print("# False Negative Rate")
```

```
## [1] "# False Negative Rate"
```

```r
print(paste0("Baseline model: ", fnr_diff2))
```

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

```r
print(paste0("Reweighing model: ", fnr_diff3))
```

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

```r
print("# False Positive Rate")
```

```
## [1] "# False Positive Rate"
```

```r
print(paste0("Baseline model: ", fpr_diff2))
```

```
## [1] "Baseline model: -0.0874126521672595"
```

```r
print(paste0("Reweighing model: ", fpr_diff3))
```

```
## [1] "Reweighing model: -0.0186613748671527"
```

*Finding:* Ideally, we want the fairness metrics for each individual group to be 1 and thus their differences to be zero. In this case, the False Positive metric gave us values closer to zero than that of the False Negative metric, but overall, neither are incredibly far from the target. The "worst" case here is the baseline model under false negative which is almost ten times larger than that of our best case: the reweighing model under false positive which stands at an absolute value of 0.0187.

### 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:* What I found comes from the link here: https://developer.ibm.com/blogs/the-aif360-team-adds-compatibility-with-r/. It discusses the AIF360 fairness toolkit which includes metrics, models, and bias mitigation algorithms (pre-, in-, and post-processing). The GitHub (which you can install from the package) takes you through installation, examples, and common errors to help with troubleshooting.

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