

# DAR F22 Project Fairness Auditor Notebook Template

Fairness Auditor

Student Name

TBD

## Contents

Instructions (DELETE BEFORE SUBMISSION) . . . . .	1
Weekly Work Summary . . . . .	2
Personal Contribution . . . . .	2
Discussion of Primary Findings . . . . .	2
Evaluating Bias Mitigation Algorithms . . . . .	3
TODO . . . . .	8

## Instructions (DELETE BEFORE SUBMISSION)

- Use this notebook is a template for your biweekly project status assignment.
  - Use the sections starting with **BiWeekly Work Summary** as your outline for your submitted notebook.
  - Summarize ALL of your work in this notebook; **if you don't show and/or link to your work here, it doesn't exist for us!**
1. Create a new copy of this notebook in the **AssignmentX** sub-directory of your team's github repository using the following naming convention
    - `rcsid_assignmentX.Rmd` and `rcsid_assignmentX.pdf`
    - For example, `bennek_assignment03.Rmd`
  2. Document **all** the work you did on your assigned project this week **using the outline below**.
  3. You **MUST** include figures and/or tables to illustrate your work. *Screen shots are okay*, but include something!
  4. You **MUST** include links to other important resources (knitted HTML files, Shiny apps). See the guide below for help.
  5. Commit the source (`.Rmd`) and knitted (`.html`) versions of your notebook and push to github
  6. **Submit a pull request**. Please notify Dr. Erickson if you don't see your notebook merged within one day.
  7. **DO NOT MERGE YOUR PULL REQUESTS YOURSELF!!**

See the Grading Rubric for guidance on how the contents of this notebook will be graded on lms.

## Guide to linking to Shiny Apps

- On the cluster, create a sub-directory called **ShinyApps** in your home directory
  - In RStudio in the **Terminal** tab, `cd ~`
  - Then `mkdir ShinyApps`

- You only need to do this once
- In your new **ShinyApps** sub-directory, create a sub-directory to contain your app
  - In the **Terminal** tab, `cd ~/ShinyApps`
  - Then `mkdir yourcoolapp` (if `yourcoolapp` is the name of your app)
  - Then copy all of the files associated with your app (esp. the `app.R`) in that directory
- Alternatively, you can create a symbolic link in that directory, “pointing” to your working directory. In this way your shared app will always be up-to-date.
  - In the **Terminal** tab, `cd ~/ShinyApps`
  - Then (for example) `ln -s /home/yourrcs/yourappdirectory yourcoolapp`
- You can now share your app on the RPI network using this syntax:
  - `http://lp01.idea.rpi.edu/shiny/yourrcs/yourcoolapp/`

## Weekly Work Summary

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

- RCS ID: **Always** include this!
- Project Name: **Always** include this!
- Summary of work since last week
  - Describe the important aspects of what you worked on and accomplished
- NEW: Summary of github issues added and worked
  - Issues that you’ve submitted
  - Issues that you’ve self-assigned and addressed
- Summary of github commits
  - include branch name(s)
  - include browsable links to all external files on github
  - Include links to shared Shiny apps
- List of presentations, papers, or other outputs
  - Include browsable links
- 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

- Clearly defined, unique contribution(s) done by you: code, ideas, writing...
- Include github issues you’ve addressed

## Discussion of Primary Findings

- Discuss primary findings:
  - What did you want to know?
  - How did you go about finding it?
  - What did you find?
- **Required:** Provide illustrating figures and/or tables

- Embed your code in this notebook if possible.
  - If not possible, screen shots are acceptable.
  - If your figures are “live,” either include source code embedded in notebook or provide github location for their source scripts.
  - If your contributions included things that are not done in an R-notebook, (e.g. researching, writing, and coding in Python), you still need to do this status notebook in R. Describe what you did here and put any products that you created in github. If you are writing online documents (e.g. overleaf or google docs), you can include links to the documents in this notebook instead of actual text.
- \*\*\*\*

## 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 (Reweighting) - ReweightingModel 6) Evaluate the utility and fairness metrics of this ReweightingModel 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)  $\leq 50K$ . 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 fnlwtg 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.

```

# 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>

```

# 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 Logistic 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

# 2. Get predictions on test data

# 3. Evaluate balanced accuracy

# 4. Evaluate equalized odds

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

### 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

# 2. Measure them on baselineModel and reweighingModel

# 3. Compare the scores and explain what you find
```

*Finding:*

### 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:*

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