

DAR F22 Project Fairness Auditor Assignment 2 Notebook

Hannah Powers

09/14/2022

Contents

Weekly Work Summary	1
Personal Contribution	2
Discussion of Primary Findings	2
Evaluating Bias Mitigation Algorithms	2
TODO	8

Weekly Work Summary

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

- RCS ID: powerh
- Project Name: Fairness Auditor
- Summary of work since last week
 - Trained and tested a random forest model
 - Found Equalized Odds and Balanced Accuracy for random forest model
 - Found Theil Index and Disparate Impact scores for baseline and reweighing models
- NEW: Summary of github issues added and worked
 - Having trouble calling aif360 from R, seems to be loading the python library to use in R but having trouble getting it to load
 - Karan aware of issue and working on it
- Summary of github commits
 - powerh-09142022
 - Committed assignment 2
- List of presentations, papers, or other outputs
 - A General Framework for Fair Regression: <https://arxiv.org/pdf/1810.05041.pdf>
 - * Fairness techniques for kernel regression with a focus on decision trees
 - A Convex Framework for Fair Regression: <https://arxiv.org/pdf/1706.02409.pdf>
 - * Fairness regularizers for linear and logistic regression

- Controlling Attribute Effect in Linear Regression: <https://ieeexplore.ieee.org/document/6729491>
 - * Uses propensity modeling and constraints to control bias in linear regression
- Fairness-Aware Learning through Regularization Approach: <https://ieeexplore.ieee.org/document/6137441>
 - * Regularization for fairness applied to logistic regression
- 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 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.

```
# 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(tptr_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("Reweighting model: ", reweighingModel.bal_acc))
```

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

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

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

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

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

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

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

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

TODO

1) Train another ML model

Evaluating Balanced Accuracy and Equalized Odds on a random forest model.

```
# 1. Train ML model here on train data
train.scale$income <- as.factor(train.scale$income)
rfModel <- randomForest(income~., data=train.scale)
```

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

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

```
# 4. Evaluate equalized odds
test2.rf <- test.scale
test2.rf$pred <- rfModel.pred
test2.rf$income <- as.factor(test2.rf$income)
test2.rf$gender <- as.factor(test2.rf$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.rf,
```



```

outcome      = 'income',
outcome_base = '0',
group        = 'gender',
preds        = 'pred',
base         = '0')

```

```

## Warning in equal_odds(data = test2.rf, 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.rf,
outcome      = 'income',
outcome_base = '0',
group        = 'gender',
preds        = 'pred',
base         = '0')

```

```

## Warning in fpr_parity(data = test2.rf, 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)
rfModel.eo <- max(abs(tpr_diff), abs(fpr_diff))

```

We compare the Balanced Accuracy and Equalized Odds scores for the random forest model, baseline model, and reweighing model.

```

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

```

```

## [1] "Balanced Accuracy scores"

```

```

print(paste0("Random forest model: ", rfModel.bal_acc))

```

```

## [1] "Random forest model: 0.724975749134574"

```

```

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 score")
```

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

```
print(paste0("Random forest model: ", rfModel.eo))
```

```
## [1] "Random forest model: 0.16209210042601"
```

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

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

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

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

Finding: Of the three models, the baseline model has the best Balanced Accuracy, followed closely by the random forest model and then the reweighing model. As for Equalized Odds, the baseline and random forest models perform nearly the same. The reweighing model outperforms both and is closest to being considered fair.

2) Evaluate other fairness metrics

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

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>

Theil Index and Disparate Impact

```
# 1. List the name of the two fairness metrics
```

```
# 2. Measure them on baselineModel and reweighingModel
```

```
# Evaluate TPR difference
```

```
pp_results <- 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
```

```
pp_diff <- pp_results$Metric[1] - pp_results$Metric[4]
```

```
# Evaluate FPR difference
```

```
ap_results <- 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
```

```
ap_diff <- ap_results$Metric[1] - ap_results$Metric[4]
```

```
# Evaluate TPR difference
```

```
pp_results2 <- 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
```

```
pp_diff2 <- pp_results2$Metric[1] - pp_results2$Metric[4]
```

```
# Evaluate FPR difference
```

```
ap_results2 <- 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
```

```
ap_diff2 <- ap_results2$Metric[1] - ap_results2$Metric[4]
```

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

```
## [1] "Proportional Parity"
```

```

print(paste0("baseline model: ", pp_diff))

## [1] "baseline model: -0.184378588568849"

print(paste0("reweighing model: ", pp_diff2))

## [1] "reweighing model: -0.0809262615923148"

print("Accuracy Parity")

## [1] "Accuracy Parity"

print(paste0("baseline model: ", ap_diff))

## [1] "baseline model: 0.118768858548021"

print(paste0("reweighing model: ", ap_diff2))

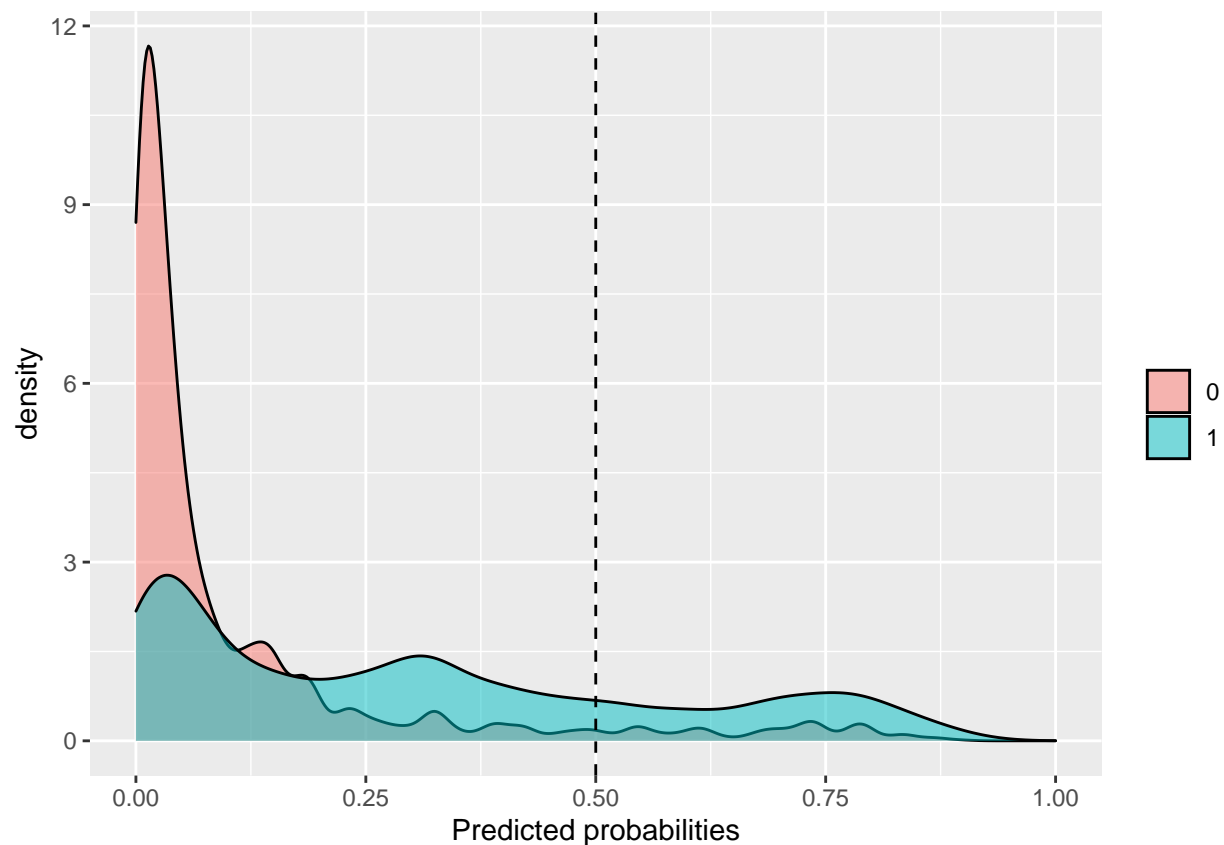
## [1] "reweighing model: 0.113580951183584"

print("Probability Density Plot")

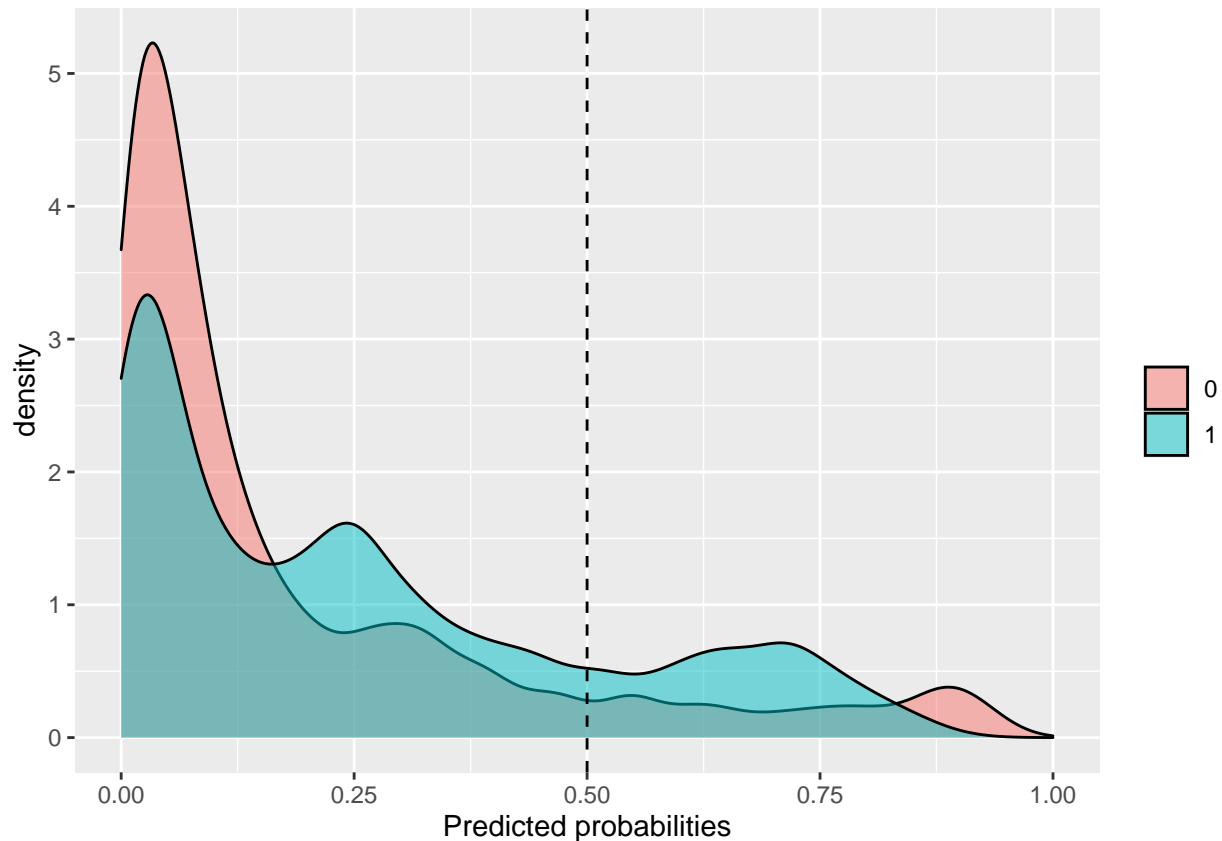
## [1] "Probability Density Plot"

pp_results$Probability_plot

```



```
pp_results2$Probability_plot
```



Finding: For proportional parity, the measure is the proportion of positive predictions within a subgroup so we hope to see values close to 0 where all subgroups see an equal proportion of positive predictions. The reweighing model outperforms the baseline model for this score and could be considered fair.

As for accuracy parity, the measure is the proportion of correct predictions within a subgroup so we hope to see values close to 0 where the model has similar accuracy between subgroups. Neither model is fair and perform about equivalent to each other.

For the probability density plots, we hope to see similar shapes between the subgroups to indicate a similar amount of people predicted to have a positive outcome. The baseline model's plot has a very different shape between subgroups but the reweighing model's plot indicates closer distributions between subgroups. This is reflected in the proportional parity measure.

3) List fairness libraries

Known Packages:

- fairness
- fairmodels
- aif360

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.

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