Bonus Assignment: Fairness Metrics and Bias Mitigation

Machine Learning

Fall 2019

Learning Objectives

- Gain some familiarity with the subject of fairness in machine learning.
- Understand confusion matrix terminology.
- Understand and implement commonly-used fairness metrics.
- Understand the core ideas behind bias mitigation algorithms.
- Practice auditing models for bias and implementing bias mitigation techniques.

1 Algorithmic Fairness

Recall in the readings on COMPAS that Propublica and Northpointe had different ideas of what was fair and what was not in regards to algorithmically predicting recidivism. COMPAS claimed that their model was fair based on the metric of sufficiency, also known as predictive rate parity, which compares the total number of true positives to the number of predicted positives. Northpointe claimed that the model was unfair because it did not satisfy the metric of separation, meaning that the false positive rates and true positive rates were not consistent between the protected attribute and the rest of the population. These are two commonly used metrics of fairness, but there are many more. In this lesson we will introduce you to other common fairness metrics, and when it is appropriate to use them.

External Resource(s)

Review the Propublica analysis of COMPAS and the Northpointe rebuttal for context. We will be diving deeper into the COMPAS example in this lesson, so we really want to make sure you understand the cultural context behind this controversy, but don't worry too much about the math yet since we will cover it in more depth in this lesson.

2 Confusion Matrices

In order to understand how these metrics work, we need an in-depth understanding of the confusion matrix, since this is the basis of most fairness metrics. The confusion matrix is a 2x2 chart where the y axis is whether the model output predicts a positive or negative response, and the x axis is whether or not the actual or measured output is positive or negative. This gives us four quadrants: true positives, false positives, true negatives, and false negatives.

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

While these are used for metrics of fairness, they are also often used for metrics of general accuracy. It is worth stating that the confusion matrix only works for binary classification, so if the model outputs are continuous, a cutoff threshold may be necessary, and this approach has to be adapted for models with multiple output classes. These confusion matrix metrics can be more informative than more general accuracy metrics like log loss, because they can pinpoint the strong and weak points of a model. The four quadrants can also be used to calculate a number of metrics that can show how the model performs with regards to different concerns. We have already gone over some basic examples of this in class, when we discussed that measures that are assistive (like a model that tells a patient whether they need further testing for a medical condition) might want to minimize false negatives, while measures that are punitive (like a model that determines whether someone should be sent to prison) might want to minimize false positives.

The values in each quadrant of our confusion matrix can be used to calculate the following important metrics:

- **True Positive Rate (TPR)** is calculated by dividing true positives by total actual positives.
- **False Positive Rate (FPR)** is calculated by dividing false positives by total actual negatives.
- **Positive Predictive Value (PPV)** is calculated by dividing true positives by total predicted positives.
- **Negative Predictive Value (NPV)** is calculated by dividing true negatives by total predicted negatives.
- **True Negative Rate (TNR)** is calculated by dividing true negatives by total actual negatives.
- **False Negative Rate (FNR)** is calculated by dividing false negatives by total actual positives.
- **False Discovery Rate (FDR)** is calculated by dividing false positives by total predicted positives.
- **False Omission Rate (FOR)** is calculated by dividing false negatives by total predicted negatives.
- **Critical Success Index (CSI)** is calculated by dividing true positives by actual positives plus false positives.

Exercise 1 (60 minutes)

Consider the following confusion matrix:

	Actual Positive	**Actual Negative**
Predicted Positive	100	20
Predicted Negative	5	40

(a) Compute the metrics listed above for this confusion matrix.

☆ Solution

TPR = 100 / (100 + 5) ≈ .95
FPR =20/(20 + 40) ≈ .33
PPV =100/(100 + 20) ≈ .83
NPV =40/(40 + 5) ≈ .89
TNR =40/(40 + 20) ≈ .67
FNR =5/(100 + 5) ≈ .05
FDR =20/(100 + 20) ≈ .17
FOR =5/(5 + 40) ≈ .11
CSI =100/(100 + 5 + 20) = .8

(b) If you've completed part a, you may have noticed that each metric above other than CSI has an inverse (for example the True Positive Rate is equal to 1 - False Negative Rate). Find the inverse for each of the metrics listed above.

☆ Solution

True Positive Rate is the inverse of False Negative Rate.

True Negative Rate is the inverse of False Positive Rate.

Positive Predictive Value is the inverse of False Discovery Rate.

Negative Predictive Value is the inverse of False Omission Rate.

(c) As we mentioned above, each metric tells us something different about our model. We gave two examples above of when we would want to minimize our False Negative Rate or our False Positive Rate. For each metric, come up with a type of scenario where it would be especially important to minimize or maximize the given metric.

☆ Solution

It would be a good idea to maximize the True Positive Rate or Negative Predictive Value and minimize the False Negative rate or False Omission Rate if you were creating a model that was assistive in nature. These metrics are important for assistive scenarios because they can help ensure that the people who need help receive it and are not skipped in the case of a false negative.

However, the True Positive Rate and False Negative Rate are more meaningful and have a wider range of values for scenarios where a sizable proportion of the population should be classified as positive, due to the fact that we are only focusing on people who are actually positive. A good example for these two metrics would be financial assistance with college tuition.

Alternatively, the Negative Predictive Value and and False Omission rate are more meaningful and have a wider range of values when a very small proportion of the population satisfies the positive condition, due to the fact that we are focusing on people who are classified as negative, regardless of what they actually are. An example of a scenario where these would be useful is in medical testing for a rare disease.

It would be a good idea to maximize the True Negative Rate or Positive Predictive Value and minimize the False Positive Rate or False Discovery Rate in the case of a model that determines a punitive outcome. These metrics are important for punitive scenarios because they can help ensure that innocent people are not punished in the case of a false positive.

However, the True Negative Rate and False Positive Rate are more meaningful and have a wider range of values when a sizable proportion of the population should be classified as positive, due to the fact that we are focusing on people who are actually negative, regardless of how they are classified. A good example for these two metrics could be deciding when to hand out speeding tickets.

Alternatively, the False Discovery Rate and Positive Predictive Value have a wider range of values and are more meaningful when a very small proportion of the population should be classified as positive, due to the fact that we are focusing on people who are classified as positive, regardless of whether they actually are. An example of when this could be used could be a model designed to decide whether someone has committed a rare and obscure crime.

3 Measures of Bias

Confusion matrix metrics can also be used to measure fairness and bias. It is important to clarify that in this section, when we use the word bias, we do not mean it in the statistical sense. When we are talking about algorithmic bias and fairness, we are referring to the phenomenon of models discriminating against protected or underprivileged groups (based on attributes such as race and gender), which often arises as a factor of bias that is integrated into the dataset used for training. This is common, because datasets are often created by researchers with implicit bias, or they measure phenomena that are already subject to systemic bias.

When detecting algorithmic bias, we typically identify a protected class (such as people of color or women). Many of the popular measures of bias compare the confusion matrix metrics we went over in the last section between the protected and privileged class, either by taking a straightforward difference, or by taking a ratio between the values of the two metrics. These are often referred to as parity of a given metric, for example False Positive Rate Parity, which measures the

difference in False Positives of the two groups. Many of the most commonly used metrics have a second name, for example the Equal Opportunity Difference is the same as True Positive Rate Parity.

There are many measures of bias, but today we will be focusing on the five that are related to the IBM Fairness 360 demo, which we will walk you through later in the lesson. By understanding these metrics, you will have the tools to understand other, similar metrics that you may want to use in the future.

- **Statistical Parity Difference** is computed as the difference of the rate of favorable outcomes received by the unprivileged group to the privileged group.
  - The ideal value of this metric is 0.
  - Fairness for this metric is between -0.1 and 0.1.
  - Calculated as the difference between predicted positive of the privileged and unprivileged groups.
  - A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group.
- **Equal Opportunity Difference** is computed as the difference of true positive rates between the unprivileged and the privileged groups.
  - The ideal value of this metric is 0.
  - A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group.
  - Fairness for this metric is between -0.1 and 0.1
  - Calculated as the difference between the privileged TPR and unprivileged TPR.
- **Average Odds Difference** is computed as the average difference of false positive rate and true positive rate between unprivileged and privileged groups.
  - The ideal value of this metric is 0.
  - A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group.
  - Fairness for this metric is between -0.1 and 0.1
  - Calculated as the average of the difference between the privileged FPR and unprivileged FPR and the difference between the unprivileged TPR and privileged TPR
- **Disparate Impact** is computed as the ratio between rate of favorable outcomes for the unprivileged group to that of the privileged group.
  - The ideal value of this metric is 1.0.
  - A value < 1 implies higher benefit for the privileged group and a value >1 implies a higher benefit for the unprivileged group.
  - Fairness for this metric is between 0.8 and 1.2.
  - Calculated as unprivileged predicted positive/privileged predicted positive.
- **Theil Index** is computed as the generalized entropy of benefit for all individuals in the dataset, with alpha = 1. It measures the inequality in benefit allocation for individuals. This is a measure of non-randomness or redundancy.

- A value of 0 implies perfect fairness.
- Fairness is indicated by lower scores, higher scores are problematic.
- We will not focus on this metric, but Wikipedia has a decent article about it.

Sometimes these metrics conflict, so we need to choose carefully based on our application which ones to minimize.

Exercise 2 (60 Minutes)

(a) Calculate the Statistical Parity Difference, Equal Opportunity Difference, Average Odds Difference, and Disparate Impact of the confusion matrices below. What do you notice? Is this model fair?

Privileged Class

	Actual Positive	**Actual Negative**
Predicted Positive	100	20
Predicted Negative	10	40

Unprivileged Class

	Actual Positive	**Actual Negative**
Predicted Positive	40	18
Predicted Negative	12	100

☆ Solution

Statistical Parity Difference

= (TP + FP)/(total population) of privileged group - (TP + FP)/(total population) of unprivileged group

= (40 + 18)/(40 + 18 + 12 + 100) - (100 + 20)/(100 + 20 + 10 + 40)

= 58/170 - 120/170

≈ -0.36

By this metric, the model is quite unfairly biased against the unprivileged group.

Equal Opportunity Difference

= TP/(TP + FN) of unprivileged group - TP/(TP + FN) of privileged group

= 40/52 - 100/110

≈ -0.14

This is close to the window of fairness, but there is still bias against the unprivileged group.

Average Odds Difference

= (unprivileged TPR - privileged TPR)/2 + (privileged FPR - unprivileged FPR)/2

= (TP/(TP + FN) of unprivileged group - TP/(TP + FN) of privileged group)/2 + (FP/(FP + TN) of privileged group - FP/(FP + TN) of privileged group)/2

= -.14/2 + (20/60 - 18/100)/2

≈ 0.007

By this definition the model is fair, since the privileged class has a worse False Positive Rate which cancels out the fact that the unprivileged class has a worse True Positive Rate.

Disparate Impact

= ((TP + FP)/total population of unprivileged group)/((TP + FP)/total population of privileged group)

=(58/170)/(120/170)

$\approx .48$

By this metric, our model is strongly biased against our unprivileged group.

As we can see, it is good to check a variety of metrics, because sometimes a model can be strongly biased by most metrics, but still fall within the window of fairness for one or two metrics.

(b) Consult this decision tree. Try to understand why each decision leads to the metrics that it does. Come up with a scenario where the model above could be used fairly.

☆ Solution

While the model above is biased against the unprivileged group in most metrics, the fact that it technically falls within the window of fairness for the Average Odds Difference is interesting. While it would generally be a bad idea to use a model like the one above, there is a conceivable scenario where it would be beneficial to use it: if a positive outcome was assistive to the privileged class and harmful to the unprivileged class, and a negative outcome was assistive to the unprivileged class and harmful to the privileged class. That said, this model is unfair by most metrics and should probably be retrained or modified.

The Impossibility Theorem of Fairness

As we explored in the previous problems, we often cannot satisfy every fairness metric at the same time. This leads to an interesting ethical discussion about when it is appropriate to use a given metric, and when it is okay to accept a failing score on a given fairness metric. Above we show a decision tree for certain metrics that we took from Aequitas, a bias detection toolkit created by the University of Chicago Center for Data Science and Public Policy. If you have some extra time we highly recommend checking this project out, but we will be focusing on a similar toolkit from IBM in this assignment. Choosing a fairness metric in a way that is ethical is a highly subjective task. Even though decision trees exist to help policymakers and model creators evaluate their models, this is an active area of research and discussion. One great resource for learning about up and coming developments in this field is the FAT conference, where researchers and technologists discuss and present research on Fairness, Accountability, and Transparency in ML. At the heart of these debates is the Impossibility Theorem of Fairness, which essentially says that there are many fairness metrics that are mutually exclusive in the vast majority of cases.

External Resource(s) 60 Minutes

This lecture on the tradeoffs of different fairness metrics talks about the societal impact of different fairness metrics, and explains the Impossibility Theorem of Fairness in more depth.

✔ Understanding Check

What is group fairness? What is individual fairness? Why are these two notions of fairness in conflict?

☆ Solution

Group fairness is focused on comparing metrics between privileged and unprivileged groups to ensure that they have comparable outcomes. Individual fairness is focused on treating the same data and situations the same way across individuals regardless of group affiliation. These can be in conflict because often time to achieve equal representation of positive outcomes between groups, the each group needs to have different criteria for triggering a positive outcome, under the assumption that their average input data or situations are not the same.

Bias Mitigation

Now that we have discussed different ways to detect bias, we want to think about how we can reduce bias and algorithmic discrimination. There are a number of bias mitigation algorithms, and each has several variations, but all bias mitigation algorithms can be broken up into three categories.

Pre-processing

Pre-processing is the practice of modifying a dataset before training a model. Models learn bias from uneven representation in datasets, stemming from problems with sampling distribution, or from existing social phenomena that create uneven distribution of input situations and outcomes between demographic groups. The theory behind preprocessing is that if a dataset represents a fair distribution of outcomes between demographic groups, any model trained on that data will learn to classify in a fair manner. Preprocessing is often the preferred method of bias mitigation, since it allows for transparency in the bias mitigation process, and it does not change the model training process. It also allows for those who release datasets to ensure that models using those datasets are fair, regardless of who is making them. However, it requires foresight to use since it needs to be implemented before the model is trained, so often model creators opt for in-processing or post-processing.

In-processing

In-processing is the practice of considering notions of fairness while training a model. This is often done by integrating fairness metrics into the error or optimization function while training. In-processing is popular because it builds fairness directly into the model design, unlike pre-processing that only builds fairness into the training data.

Post-processing

Post processing is the practice of modifying an existing model's output to satisfy notions of fairness. Post-processing only focuses on group fairness, and it does not consider individual fairness. For this reason, post-processing is often less preferable that the other types of bias mitigation algorithms, which have to ability to consider both group fairness and individual fairness. However, post-processing is a good choice when a model is already trained and cannot be rebuilt.

External Resource(s) 30 minutes

Walk through this IBM demo of their different bias mitigation algorithms using the COMPAS dataset. Take note of which algorithms do the best job of mitigating bias for each metric, and take note of the differences in model accuracy before and after applying the algorithms.

Reweighing

We won't go in depth on multiple bias mitigation algorithms, but we will focus in on one simpler bias mitigation technique: reweighing. As you saw from the demo, this is an incredibly powerful technique for mitigating bias, and it has a very small effect on accuracy. The reweighing algorithm assigns a different weight to each datapoint, with the goal of creating an equal distribution of positive and negative outcomes between the privileged and protected class. Optionally, if you want to go more in depth on how this is done computationally, you can read the initial paper where reweighing was introduced. This paper also includes other preprocessing techniques, but it is somewhat dense. We will implement this algorithm through an external library in the companion notebook.

External Resource(s)

In our companion notebook we practice using the aif360 toolkit from IBM to practice auditing models for bias and implementing mitigation algorithms.

Mitigating Bias in the COMPAS Datset

In this notebook you'll have a chance to determine which fairness metrics are relevant to the COMPAS dataset, and to apply them to a few simple models. You will start with a simple model based on the unprocessed data, and then you will use one of the bias mitigation algorithms we covered to see if it makes a difference.

Disclaimer: The models we will be using in this assignment are simple, and they may be very different from the models used by Northpointe. This is simply meant to demonstrate how powerful these bias mitigation algorithms can be, but it is not a reflection of the actual COMPAS model.

We will start by loading our fairness toolkit and dataset.

In [1]:

```
# Load all necessary packages
import numpy as np
from aif360.algorithms.preprocessing.optim_preproc_helpers.data_preproc_functions\
        import load_preproc_data_compas
```

In [2]:

```
#Load dataset from aif360 examples
dataset_orig = load_preproc_data_compas(['race'])

#Split into train and test
#the seed parameter random seed, which will ensure the same train/test split each time
#this helps with ensuring our interpretation makes sense, feel free to remove this parameter to check f
or consistency of results
dataset_orig_train, dataset_orig_test = dataset_orig.split([0.7], shuffle=True, seed=0)

#These groups will be used for determining fairness later
#Our privileged group is "Caucasian", and our unprivileged group is "Not Caucasian"
privileged_groups = [{'race': 1}]
unprivileged_groups = [{'race': 0}]
```

Creating a model

We want to check if an unmodified model exhibits bias against people of color, so we have to create a model, and pick a metric of fairness. While we do not know how Northpointe created their model, we can still detect if bias arrises just from using their dataset. While Northpointe and ProPublica both have ideas of which fairness metric is appropriate to use, we want to show you how to choose a fairness metric based on the utility of a model, so we may end up using a different metric than either organization has suggested.

Notebook Exercise 1

Create a simple classifier of your choosing using the training dataset, then check accuracy on the training and testing set and compare it to the accuracy in the aif360 demo from the notebook exercise.

Hint: you are allowed to import external libraries to create your model

In [3]:

```
#***Solution***
#We have opted for a logistic regression since it is simple and easy to interpret, feel free to use ano
ther type of model

from sklearn.linear_model import LogisticRegression

X_train = dataset_orig_train.features
y_train = dataset_orig_train.labels.flatten()
X_test = dataset_orig_test.features
y_test = dataset_orig_test.labels.flatten()
model_orig = LogisticRegression()
model_orig.fit(X_train, y_train)
```

Out[3]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [4]:

```
#***Solution***
print("Accuracy on training set: ", (model_orig.predict(X_train) == y_train).mean())
print("Accuracy on testing set: ", (model_orig.predict(X_test) == y_test).mean())
#These are both within 2% of the accuracy in the IBM demo, so we will call this model a success
```

```
Accuracy on training set:  0.6572820790471035
Accuracy on testing set:  0.6805555555555556
```

Notebook Exercise 2

As we discussed in our section of the impossibility theorem, we have to choose fairness metrics to optimize, because we cannot satisfy all of them since they often contradict each other. Let's recall what we know about COMPAS, and refer to our decision tree again. If we were to choose one metric to optimize, based on the decision tree what would be a good choice?

Solution

It would not make sense to attempt to have equal representation of racial groups in the prison system for a number of reasons, but one of the most obvious reasons is that we do not have equal representation of racial groups in the United States. So, we will choose the right branch of the tree. Since this tool is used to determine sentencing, COMPAS is by definition punitive, so we move to the left branch. This last question is where the answer gets a little bit subjective. As of 2016, 698 people per 100,000 were in the US prison system, or a little under .7 percent. This may seem like a small percentage, but we would argue that this is still meaningfully large given the size of the US population. So, we would advocate for choosing the right branch (False Positive Rate Parity), but a case could be made for choosing the left branch (False Discovery Rate Parity). This is also the metric that ProPublica suggests is best.

Evaluating our model for bias

Now that we have created our model and chosen a bias metric to focus on, we want to evaluate our initial model for fairness. We will dive into the aif360 toolkit and direct you to the relevant documentation so that you are comfortable implementing this workflow in future projects.

Notebook Exercise 3

Use this documentation from the aif360 toolkit to implement the fairness metric we chose in the last exercise for our initial unmodified model. Is this model fair by our chosen definition?

In [5]:

```
#***Solution***
#import the metric class that includes False Positive Rate Difference
from aif360.metrics import ClassificationMetric

#Create a copy of the dataset that includes the model predictions
dataset_orig_test_pred = dataset_orig_test.copy(deepcopy=True)
predicted_labels = model_orig.predict(X_test)
dataset_orig_test_pred.labels = predicted_labels

#Create the classifacation metric object
metric_orig_train = ClassificationMetric(dataset_orig_test,
                                         dataset_orig_test_pred,
                                         unprivileged_groups=unprivileged_groups,
                                         privileged_groups=privileged_groups)
print("The False Positive Rate Difference is: ", metric_orig_train.false_positive_rate_difference())
```

```
The False Positive Rate Difference is:  -0.26665469270459147
```

It is worth mentioning that we opted to use the False Positive Rate Difference, but the False Positive Rate Ratio is another metric that can be used to determine False Positive Rate Parity. We have not defined in this lesson a fair range for this metric, but if we recall from our definitions from the demo, other similar fairness definitions that are based on the difference of a confusion matrix metric (like Average Odds Difference) have a range of -.1 to .1 for fairness, with 0 meaning there is no bias. We also know that positive values indicate bias against the privileged group, while negative values indicate bias against the unprivalileged group. By this fairness definition, there is significant bias against the unprivileged group (people of color), which is consistent with what we have read about COMPAS.

Mitigating Bias

We have briefly discussed how the Reweighing algorithm works, so now we will use aif360 to reweigh the COMPAS dataset and see how it affects our chosen fairness definition. Again we will be leading you to the correct documentation, so that you can become sufficient in the aif360 workflow.

Notebook Exercise 4

Use this documentation from aif360 to reweigh our original training dataset.

Hint: you will need to save the new weights, and you will need to account for them when training your new model

In [6]:

```
#***Solution***
#import the Reweighing algorithm
from aif360.algorithms.preprocessing import Reweighing

#create the Reweighing object and get the new weights
reweigh = Reweighing(unprivileged_groups=unprivileged_groups,
                     privileged_groups= privileged_groups)
reweigh.fit(dataset_orig_train)
dataset_mod_train = reweigh.fit_transform(dataset_orig_train)
#save the new traning dataset and weights
X_train_mod = dataset_mod_train.features
y_train_mod = dataset_mod_train.labels.flatten()
weights_train_mod = dataset_mod_train.instance_weights
```

Notebook Exercise 5

Retrain the model with our reweighed dataset and check performance on the chosen fairness metric, as well as the new accuracy. Has this technique made our model more fair? How so?

In [7]:

```
model_mod = LogisticRegression()
#note that we include the optional parameter sample_weight, where we explicitly apply the weights we go
t from the last exercise
#This optional parameter by default applies a weight of 1 to each data point, unless we pass in other w
eights
model_mod.fit(X_train_mod, y_train_mod, sample_weight=weights_train_mod)
```

Out[7]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [8]:

```
print("Accuracy on training set: ", (model_mod.predict(X_train_mod) == y_train_mod).mean())
print("Accuracy on testing set: ", (model_mod.predict(X_test) == y_test).mean())
#This is within 2% of the original accuracy, so not a huge difference
```

```
Accuracy on training set:  0.6543042772062805
Accuracy on testing set:  0.6597222222222222
```

In [9]:

```
dataset_mod_test = dataset_orig_test.copy(deepcopy=True)
dataset_mod_test_pred = dataset_mod_test.copy(deepcopy=True)
predicted_labels = model_mod.predict(X_test)
dataset_mod_test_pred.labels = predicted_labels

metric_mod_train = ClassificationMetric(dataset_mod_test,
                                        dataset_mod_test_pred,
                                        unprivileged_groups=unprivileged_groups,
                                        privileged_groups=privileged_groups)
print("The False Positive Rate Difference is: ", metric_mod_train.false_positive_rate_difference())
print("This is a difference of", metric_mod_train.false_positive_rate_difference() - metric_orig_train.
false_positive_rate_difference(), "compared to the original model")
```

```
The False Positive Rate Difference is:  -0.03368493240154141
This is a difference of 0.23296976030305006 compared to the original model
```

This new False Positive Rate Difference still shows some bias against the unprivileged group, but it is within the window of fairness. This means that Reweighing is a viable mitigation algorithm for this use case.