

Kaitlin Kartsen

Professor Ruiz

CS 4000-level ISP

8/15/2025

## **A Study on Bias and Mitigation Strategies in Machine Learning Algorithms**

### **1. Executive Summary**

The purpose of this study was to determine the effectiveness<sup>1</sup> of different bias mitigation strategies for reducing the impact of bias<sup>2, 3</sup> in data. This study was conducted through the implementation of the exponentiated gradient in-process bias mitigation strategy on a decision tree machine learning model when trained on the UCI Adult Income dataset. Bias was measured using the fairness metrics of demographic parity and equalized odds. To produce comprehensive results, these metrics were evaluated both with and without implementing the bias mitigation techniques for three datasets with varying degrees of bias and across both race and gender demographic categories<sup>4</sup>. Findings show that mitigating the demographic parity for the targeted demographic tends to produce optimal results, but considerations must be taken for the degree of bias within the dataset and trade-offs with other fairness metrics, as well as accuracy.

### **2. Methods**

This project utilized Google Colab to host a Jupyter notebook for all code and to provide a readable format for explanations of the process and results. This notebook can be accessed for a higher level of detail on the methods employed. The dataset used was “Adult Income,” a collection of census data used in machine learning contexts to predict whether an individual’s annual income exceeds \$50,000 per year (UCI).

### *a. Preparing the dataset*

To be able to use this dataset for training and testing the model, it had to first be imported into the Jupyter notebook, cleaned<sup>5</sup>, and preprocessed<sup>6</sup>. The dataset was imported from Kaggle, a data science platform that allows users to host datasets, among other things (Kaggle). Analysis of the dataset revealed a small percentage of rows containing unknown values in at least one column, as well as a few columns that did not contain useful information. These rows and columns were therefore removed during the cleaning process. More details on cleaning, along with data visualizations, are included in the Jupyter notebook.

Once the data was cleaned, it was preprocessed to be compatible with training and fairness evaluation operations. The decision tree machine learning model works by splitting data at each node based on a threshold. This requires it to make comparisons between the values within a column to judge whether they are above or below the threshold amount. For this reason, only numerical values can be used, since categorical data has no inherent value to a computer. Ordinal categorical data included the “education” column. Each level of education has already been assigned a number, which is recorded in the “educational-num” column. To remove ordinal data while preserving its meaning in the dataset, the “education” column was simply deleted, since “educational-num” does the job of preserving the information contained within it in numerical form. Nominal data included workclass, marital status, occupation, family role within the household, race, gender, and country of origin. Since nominal data has no relative ordering, these columns were transformed to numerical data through one-hot encoding<sup>7</sup>. Additionally, the calculations used to compute the fairness metrics require labels to be positive or negative. To implement this, values in the income column were transformed from strings of “>50K” and “<=50K” to True or False boolean values, respectively.

### b. Altering bias in the dataset

To evaluate how the bias mitigation techniques perform on datasets with different levels of initial bias, a version of the Adult Income dataset with more bias and a version with less bias were created. Bias was altered for gender as well as racial demographic categories. In order to increase bias, demographics with a lower percentage of positive instances had their number of positive instances decreased by half, while other demographics had their number of negative instances decreased by half. Figure 1 shows the process for reducing the number of positive instances for the female demographic category.

Figure 1. Code to reduce the number of positive female data instances.

```
#Get instances where gender is female and income is >$50k
fem_pos = more_bias_df[(more_bias_df['gender_Female'] == 1) &
(more_bias_df['income'] == True)]

#Randomly relabel half of fem_pos to be fem_neg
n_flip_fem = int(fem_pos.shape[0] * 0.50) #num of instances to flip
flip_indices_fem = fem_pos.sample(n=n_flip_fem).index #indices to flip
more_bias_df.loc[flip_indices_fem, 'income'] = False #flip the indices
```

This process was repeated for instances where race was labeled as Black, American Indian/Eskimo, or Other. Figure 2 shows the process for increasing the number of positive instances for the White demographic category.

Figure 2. Code to increase the number of positive White data instances.

```
#Get instances where race is White and income is <=$50k
white_neg = more_bias_df[(more_bias_df['race_White'] == 1) &
(more_bias_df['income'] == False)]

#Randomly relabel half of white_neg to be white_pos
n_flip_white = int(white_neg.shape[0] * 0.50) #num of instances to flip
flip_indices_white = white_neg.sample(n=n_flip_white).index #indices to flip
more_bias_df.loc[flip_indices_white, 'income'] = True #flip the indices
```

This process was repeated for instances where race was labeled as Asian/Pacific Islander or where gender was labeled as male.

To create the version of the dataset with less bias, each demographic category was resampled so that the distribution of positive and negative values matched that of the overall dataset. Figure 3 shows this process for the male demographic category.

Figure 3. Code to resample data in the male demographic category.

```
#Get overall positive rate-- done once at beginning of process
pos_rate = (less_bias_df['income'] == True).mean()

#Identify rows to resample
male_pos = less_bias_df[(less_bias_df['gender_Male'] == 1) &
(less_bias_df['income'] == True)]
male_neg = less_bias_df[(less_bias_df['gender_Male'] == 1) &
(less_bias_df['income'] == False)]

#Get number of male instances
n_male = len(less_bias_df[less_bias_df['gender_Male'] == 1])

#Calculate number of male instances that should be positive to have the
#same positive rate as the overall data
n_male_pos = int(n_male * pos_rate)

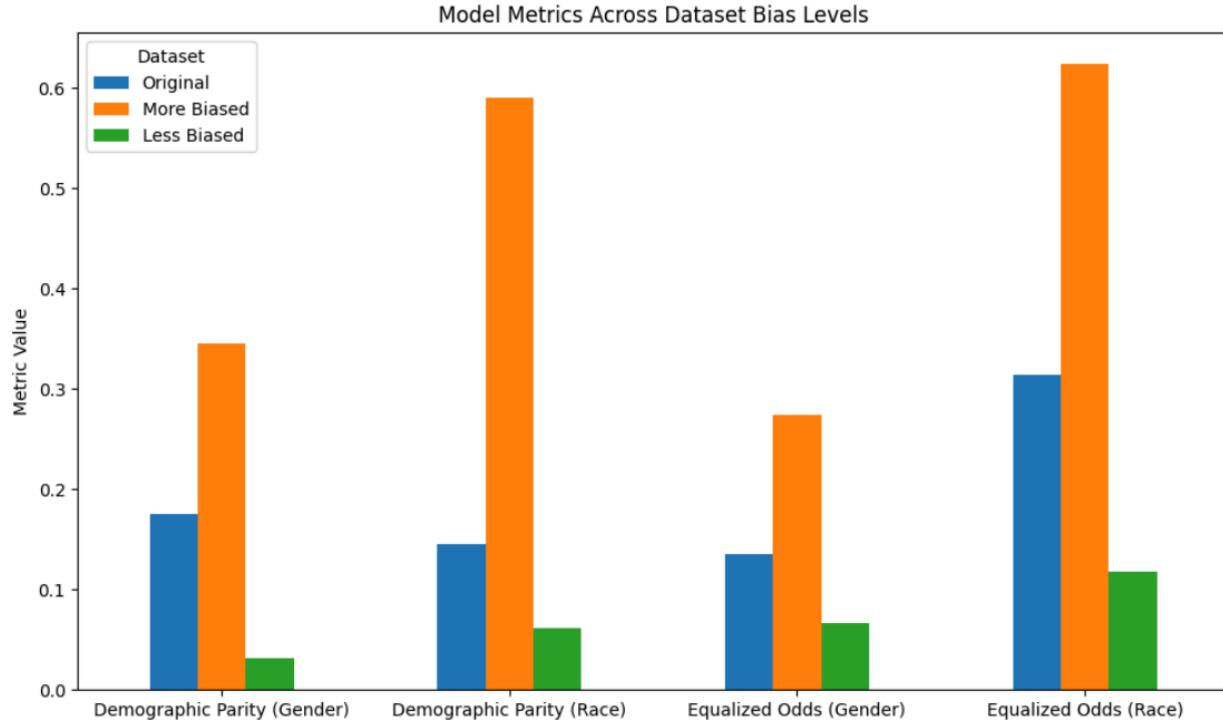
#Resample positive and negative male data according to n_male_pos
resample_male_pos = male_pos.sample(n=n_male_pos, replace=True)
resample_male_neg = male_neg.sample(n=n_male - n_male_pos, replace=True)

#Drop original male data instances from dataset
indices_to_drop = male_pos.index.union(male_neg.index)
less_bias_df = less_bias_df.drop(indices_to_drop)

#Add resampled male data instances to dataset
less_bias_df = pd.concat([less_bias_df, resample_male_pos,
resample_male_neg])
```

This process was done for all demographic categories, and code was refactored to limit unnecessary repetition. Figure 4 shows the results for each fairness metric after creating each version of the dataset.

Figure 4. Fairness metrics for each version of the dataset.



### *c. Training the model*

A decision tree is a simple machine learning algorithm that is good for classifying categorical data, which is why it was chosen for this project to classify individuals based on income. First, the dataframe storing the dataset was divided into a feature matrix and a list of labels by separating the income column. Then it was split into training and testing data using the `train_test_split` method from the `sklearn` library. The `DecisionTreeClassifier` class, also imported from the `sklearn` library, was used to create the decision tree model, train it on the training data, and test it on the testing data, outputting a list of the model's predictions. For each version of the dataset, a decision tree model was trained and tested in this way, providing predictions with which to evaluate accuracy and fairness metrics.

*d. Evaluating the predictions for each model*

Each model was evaluated on the accuracy and fairness of its predictions. Accuracy was computed using the accuracy\_score metric imported from sklearn. The formula for accuracy is: Accuracy = [(Number of true positive predictions + Number of true negative predictions) / Total number of predictions] \* 100%. Figure 5 shows an example of this calculation and the code to implement it.

Figure 5. Accuracy calculation example and code.

```
#Example accuracy calculation using similar numbers to real data
TP = 8868           FP = 3250           TN = 26604           FN = 6500

Accuracy = [(8868 + 26604) / (8868 + 3250 + 26604 + 6500) * 100%
Accuracy = 35472 / 45222 * 100 = 78.44%


#Code used to find accuracy
accuracy = accuracy_score(y_test, predictions)
```

Fairness was evaluated using the demographic parity difference and equalized odds difference metrics for both gender and race, due to these metrics' popularity and applicability to real-world use cases. The demographic parity difference is based on the selection rate of each demographic category, while the equalized odds difference is based on the true positive rate and false positive rate for each demographic category. Each intermediate step was calculated using the MetricFrame class from the fairlearn Python package, and the final results for each fairness metric were computed using the demographic\_parity\_difference and equalized\_odds\_difference methods, also from fairlearn. The formula for calculating demographic parity difference is: Demographic parity = Highest result for selection rate among demographic categories - Lowest result for selection rate among demographic categories. The formula for calculating selection rate is: Selection rate = Number of positive predictions made by the model / Total number of predictions made. Selection rate is calculated for each demographic category within a demographic so the

values can be compared for calculating demographic parity. Figure 6 shows an example of these calculations and code to get the result for demographic parity.

Figure 6. Demographic parity calculations examples and code.

```
#Example demographic parity calculation using gender
TP (male) = 15           TP (female) = 6
FP (male) = 5            FP (female) = 9
TN (male) = 33           TN (female) = 14
FN (male) = 7            FN (female) = 11

Selection rate male = (15 + 5) / (15 + 5 + 33 + 7) = 20 / 60 = 0.33
Selection rate female = (6 + 9) / (6 + 9 + 14 + 11) = 15 / 40 = 0.38
Demographic parity = 0.38 - 0.33 = 0.05

#Example demographic parity calculation using race
TP (white) = 30          TP (black) = 15          TP (asian) = 20
FP (white) = 5            FP (black) = 3            FP (asian) = 10
TN (white) = 8            TN (black) = 25          TN (asian) = 30
FN (white) = 4            FN (black) = 8            FN (asian) = 5

Selection rate white = (30 + 5) / (30 + 5 + 8 + 4) = 35 / 47 = 0.74
Selection rate black = (15 + 3) / (15 + 3 + 25 + 8) = 18 / 51 = 0.35
Selection rate asian = (20 + 10) / (20 + 10 + 30 + 5) = 30 / 65 = 0.46
Demographic parity = 0.74 - 0.35 = 0.39

#Code used to find demographic parity for gender, no bias change
demographic_parity_difference(
    y_true=y_test,
    y_pred=predictions,
    sensitive_features=X_test[['gender_Male', 'gender_Female']])
)

#Code used to find demographic parity for race, no bias change
demographic_parity_difference(
    y_true=y_test,
    y_pred=predictions,
    sensitive_features=X_test[['race_White', 'race_Black',
    'race_Asian-Pac-Islander', 'race_Amer-Indian-Eskimo', 'race_Other']])
)
```

The formula for calculating equalized odds difference is: Equalized odds = Maximum of either the True Positive Rate (TPR) gap or the False Positive Rate (FPR) gap for the demographic. The formula for calculating the TPR gap is: TPR gap = Highest result for TPR - Lowest result for TPR. The formula for calculating the FPR gap is: FPR gap = Highest result for FPR - Lowest result for FPR. The formula for calculating TPR is: TPR = Number of true positive predictions made by the model / (Number of true positive predictions + Number of false negative predictions made by the model). The formula for calculating FPR is: FPR = Number of false positive predictions made by the model / (Number of false positive predictions + Number of true negative predictions made by the model). Figure 7 shows examples of these calculations, as well as code to get the result for equalized odds.

Figure 7. Equalized odds calculations examples and code.

```
#Example equalized odds calculation using gender
TP (male) = 15           TP (female) = 6
FP (male) = 5             FP (female) = 9
TN (male) = 33            TN (female) = 14
FN (male) = 7             FN (female) = 11

TPR male = 15 / (15 + 7) = 15 / 22 = 0.68
TPR female = 6 / (6 + 11) = 6 / 17 = 0.35
FPR male = 5 / (5 + 33) = 5 / 38 = 0.13
FPR female = 9 / (9 + 14) = 9 / 23 = 0.39
TPR gap = 0.68 - 0.35 = 0.33
FPR gap = 0.39 - 0.13 = 0.26
Equalized odds = max(0.33, 0.26) = 0.33

#Example equalized odds calculation using race
TP (white) = 30          TP (black) = 15          TP (asian) = 20
FP (white) = 5             FP (black) = 3            FP (asian) = 10
TN (white) = 8             TN (black) = 25          TN (asian) = 30
FN (white) = 4             FN (black) = 8            FN (asian) = 5
```

```

TPR white = 30 / (30 + 4) = 30 / 34 = 0.88
TPR black = 15 / (15 + 3) = 15 / 18 = 0.83
TPR asian = 20 / (20 + 5) = 20 / 25 = 0.80
FPR white = 5 / (5 + 8) = 5 / 13 = 0.38
FPR black = 3 / (3 + 25) = 3 / 28 = 0.11
FPR asian = 10 / (10 + 30) = 10 / 40 = 0.25
TPR gap = 0.88 - 0.80 = 0.08
FPR gap = 0.38 - 0.11 = 0.27
Equalized odds = max(0.08, 0.27) = 0.27

```

```

#Code used to find equalized odds for gender, no bias change
equalized_odds_difference(
    y_true=y_test,
    y_pred=predictions,
    sensitive_features=X_test[['gender_Male', 'gender_Female']],
)

```

```

#Code used to find equalized odds for race, no bias change
equalized_odds_difference(
    y_true=y_test,
    y_pred=predictions,
    sensitive_features=X_test[['race_White', 'race_Black',
    'race_Asian-Pac-Islander', 'race_Amer-Indian-Eskimo', 'race_Other']],
)

```

### e. Implementing bias mitigation techniques

In total, bias was mitigated in four different ways: exponentiated gradient applied to demographic parity for gender, exponentiated gradient applied to demographic parity for race, exponentiated gradient applied to equalized odds for gender, and equalized odds for race. The process of mitigating bias during the model training stage using these strategies and the ExponentiatedGradient, DemographicParity, and EqualizedOdds classes imported from fairlearn required some tweaks to the process of training the model discussed in Section 2c. The first tweak was in splitting the data into training and testing data. An array of sensitive features was created from the dataset and included each demographic category for the demographic for which bias was being mitigated.

This array was then included in the arguments for the `train_test_split` method and used to create training and testing data just for those features. Figure 8 shows the code that implemented this adjustment, mitigating gender bias as calculated using demographic parity in the less biased dataset.

Figure 8. Code to split the less biased dataset into training and testing data to implement the exponentiated gradient bias mitigation technique when applied to demographic parity for gender

```
#Create the array of sensitive features
sensitive_features_Lg=Xlb[['gender_Male', 'gender_Female']]

#Split the dataset
X_train_Lg, X_test_Lg, y_train_Lg, y_test_Lg, gender_train_Lg,
gender_test_Lg = train_test_split(Xlb, ylb, sensitive_features_Lg,
test_size=0.2)
```

Then, before training, the model was wrapped in an `ExponentiatedGradient` object. This object is then used to train and test the model. Figure 9 shows the code to implement this step for mitigating racial bias as calculated using demographic parity in the more biased dataset.

Figure 9. Code to wrap the decision tree model in the bias mitigator object in order to implement the exponentiated gradient bias mitigation technique when applied to demographic parity for race for the more biased dataset

```
#Wrap the model in the mitigator
mitigator_Mr_DP = ExponentiatedGradient(
    model,
    constraints=DemographicParity()
)
```

The final tweak was during the training process. The sensitive features training array created during splitting is included in the training function. Figure 10

shows this step implemented for mitigating gender bias as calculated using equalized odds in the originally biased dataset.

Figure 10. Code to train the decision tree model to implement the exponentiated gradient bias mitigation technique when applied to equalized odds for gender for the originally biased dataset

```
#Train the decision tree model wrapped in the bias mitigator
mitigator_Og_EO.fit(X_train_Og, y_train_Og,
sensitive_features=gender_train_Og)
```

Except for the variable names, whether the features for race or gender were used when defining sensitive features, and whether the ExponentiatedGradient constraints used a DemographicParity or EqualizedOdds object, the code for each implementation of bias mitigation was the same across datasets, bias type, and demographic targeted.

#### *f. Evaluating the model predictions after bias mitigation*

After implementing the four types of bias mitigation on the three versions of the dataset, 12 sets of predictions had been generated. For each set of predictions, the accuracy, demographic parity for race, demographic parity for gender, equalized odds for race, and equalized odds for gender were calculated in exactly the same manner as when it was done on the predictions generated by models that had not used bias mitigation techniques. For example, Figure 11 shows the code for finding the demographic parity for race for the more biased dataset after undergoing bias mitigation to improve equalized odds for gender.

Figure 11. Code to compute the demographic parity for race for the more biased dataset after undergoing bias mitigation to improve equalized odds for gender.

```
demographic_parity_difference(
    y_true=y_test_Mg,
```

```

y_pred=pred_Mg_EO,
sensitive_features=X_test_Mg[['race_White', 'race_Black',
'race_Asian-Pac-Islander', 'race_Amer-Indian-Eskimo', 'race_Other']]
)

```

### 3. Results

Table 1 provides the numbers calculated for each metric, demographic, dataset version, and bias mitigation strategy combination.

Table 1. Accuracy and fairness for gender and race as calculated using demographic parity and equalized odds for three versions of a dataset varying in level of bias after training a decision tree model with exponentiated gradient bias mitigation for demographic parity of race and gender, equalized odds of race and gender, and no bias mitigation.

Bias Mitigation	Dataset Version	Metric	Results: Gender	Results: Race
No Bias Mitigation	Less Bias	Demographic Parity	0.03	0.06
		Equalized Odds	0.07	0.12
		Accuracy	92.96%	
	Original Bias	Demographic Parity	0.18	0.15
		Equalized Odds	0.14	0.31
		Accuracy	79.16%	
	More Bias	Demographic Parity	0.35	0.59
		Equalized Odds	0.27	0.62
		Accuracy	66.12%	
Exponentiated Gradient for Demographic Parity of Gender	Less Bias	Demographic Parity	0.008	0.10
		Equalized Odds	0.10	0.17
		Accuracy	92.66%	

Exponentiated Gradient for Demographic Parity of Race	Original Bias	Demographic Parity	0.05	0.10
		Equalized Odds	0.02	0.17
		Accuracy	75.48%	
	More Bias	Demographic Parity	0.05	0.52
		Equalized Odds	0.003	0.53
		Accuracy	64.65%	
	Less Bias	Demographic Parity	0.04	0.08
		Equalized Odds	0.15	0.19
		Accuracy	92.06%	
	Original Bias	Demographic Parity	0.19	0.08
		Equalized Odds	0.14	0.17
		Accuracy	76.88%	
	More Bias	Demographic Parity	0.27	0.07
		Equalized Odds	0.31	0.21
		Accuracy	62.22%	
Exponentiated Gradient for Equalized Odds for Gender	Less Bias	Demographic Parity	0.004	0.10
		Equalized Odds	0.06	0.16
		Accuracy	92.32%	
	Original Bias	Demographic Parity	0.17	0.17
		Equalized Odds	0.09	0.28
		Accuracy	78.30%	
	More Bias	Demographic Parity	0.29	0.59

		Equalized Odds	0.24	0.53
		Accuracy	66.55%	
Exponentiated Gradient for Equalized Odds for Race	Less Bias	Demographic Parity	0.03	0.07
		Equalized Odds	0.14	0.09
		Accuracy	91.86%	
	Original Bias	Demographic Parity	0.20	0.14
		Equalized Odds	0.14	0.26
		Accuracy	78.05%	
	More Bias	Demographic Parity	0.35	0.53
		Equalized Odds	0.32	0.46
		Accuracy	67.30%	

#### 4. Interpretations

Figure 12 shows scatter plots of fairness versus bias mitigation techniques for each of the four fairness metric/demographic combinations. The demographic parity metric graphs show that the exponentiated gradient for demographic parity bias mitigation technique tends to produce the same value for demographic parity for the targeted demographic regardless of the initial level of bias within the data. This indicates that the technique can be very useful for datasets with a high initial demographic parity (aka low fairness), but can have a negligible impact or even increase bias when the dataset has a low initial demographic parity. Therefore, we can conclude that the best practice for increasing fairness in a model by mitigating demographic parity involves measuring the demographic parity beforehand, to ensure it is not so low that implementing in-process bias mitigation has the opposite of the intended effect.

These two graphs also show that the most impactful of the four mitigation strategies for mitigating demographic parity for a specific demographic is the exponentiated gradient for demographic parity bias mitigation technique for the specific demographic that is intended to be mitigated. This aligns with reasonable expectations that the best tool for a task is the one specifically intended for it.

In contrast, the equalized odds metric graphs show that the most impactful strategy for mitigating equalized odds for a specific demographic is the exponentiated gradient for demographic parity bias mitigation technique for the specific demographic that is intended to be mitigated. In general, and especially for the dataset with the most bias, this strategy performs better than directly targeting equalized odds. In fact, directly targeting equalized odds showed an increase in bias for datasets with less initial bias. This indicates that in order to mitigate equalized odds, it may be more useful to target demographic parity for the same demographic than to target equalized odds directly.

Figure 12. Fairness Value vs. Bias Mitigation Techniques for Datasets by Bias Level

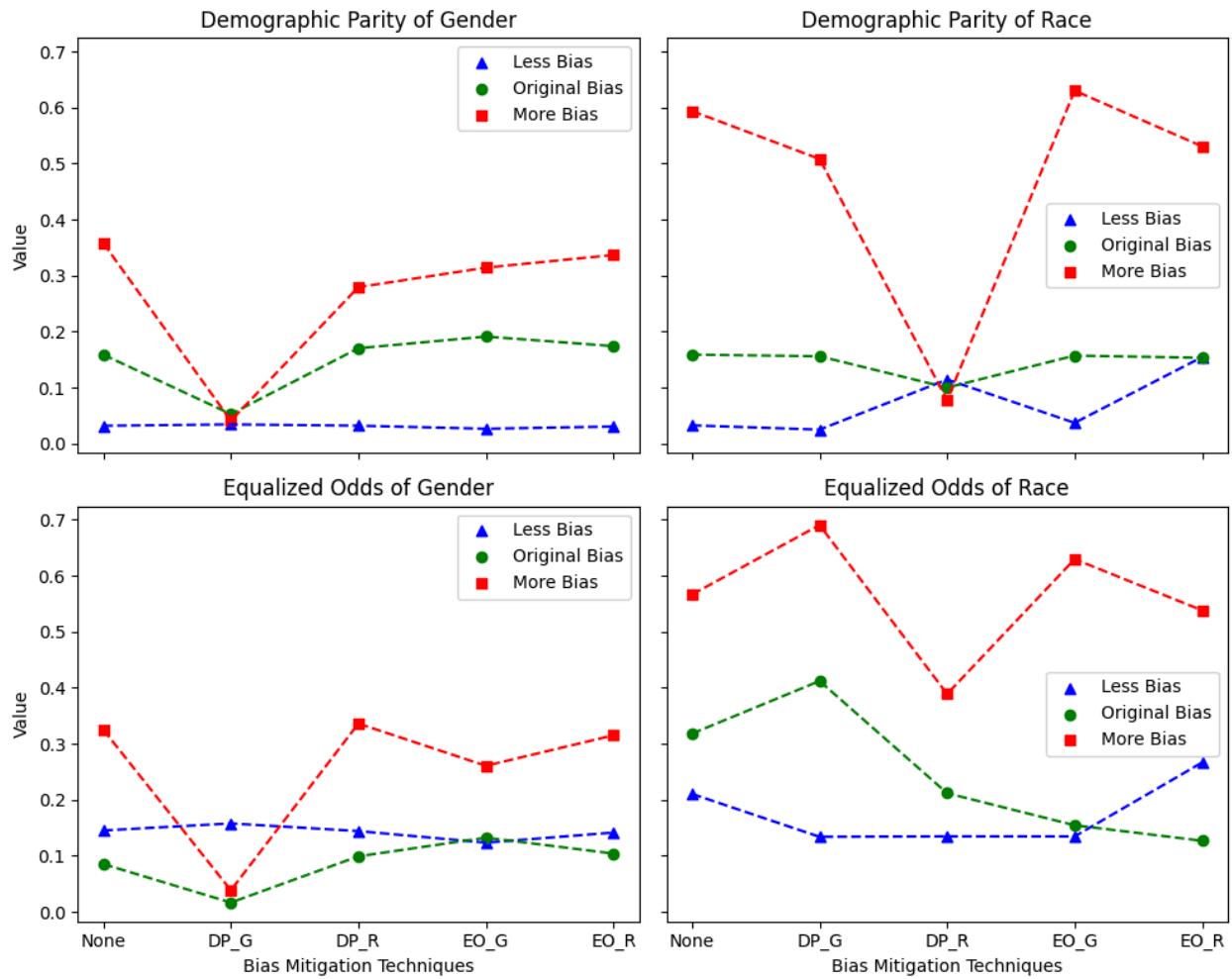


Figure 13 displays a scatter plot of accuracy versus bias mitigation techniques for each of the four fairness metric/demographic combinations. This graph shows that, in general, implementing in-process fairness techniques slightly decreases the accuracy of the model. The exponentiated gradient for

demographic parity bias mitigation technique appears to negatively impact accuracy more than the exponentiated gradient for equalized odds bias mitigation technique. Taken along with Figure 12, this indicates a potential trade-off between fairness and accuracy.

Figure 13. Accuracy vs Bias Mitigation Techniques for Datasets by Bias Level

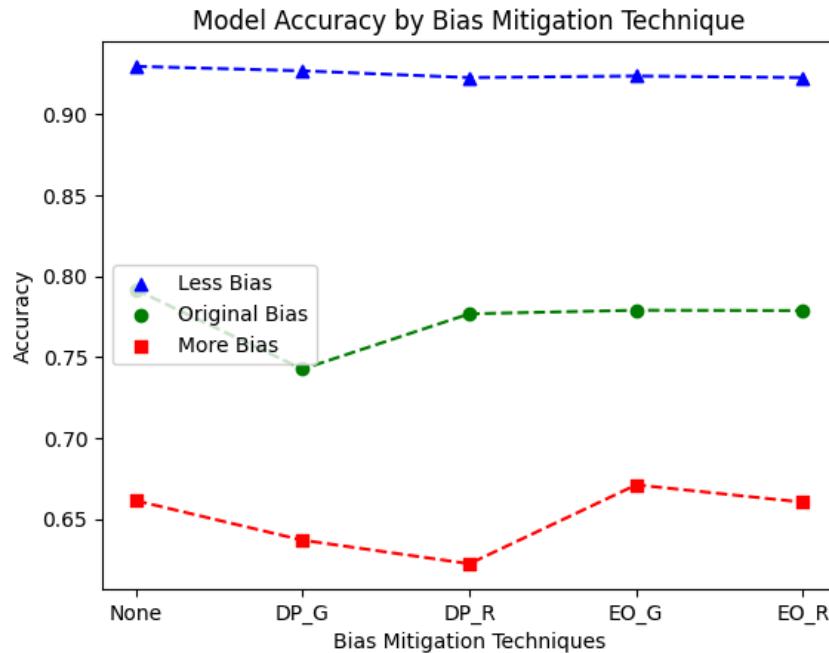


Figure 14 displays scatter plots of fairness versus dataset bias level for each of the four fairness metric/demographic combinations. These graphs more clearly demonstrate the most impactful bias mitigation technique to mitigate each bias metric. To mitigate demographic parity of gender, the results very clearly show that implementing exponentiated gradient for demographic parity of gender is highly superior to implementing any other mitigation strategy, and its impact grows with the level of bias in the dataset. We can conclude that it should always be the implemented strategy for mitigating this type of bias, and it is not necessary to check the bias level in the dataset first. When mitigating demographic parity of race, we can see that implementing the exponentiated gradient for demographic parity of gender bias mitigation strategy is better than any other strategy, but only for datasets above a certain level of this bias. If the dataset has a low level of demographic parity for race, using no mitigation strategies is optimal. In the equalized odds of gender graph, the extent to which bias mitigation of demographic parity for gender is more effective than bias mitigation of equalized odds for gender can be seen clearly. In the equalized odds for race graph, the optimal bias mitigation strategy seems

to be less consistent. Best practice may be to test mitigation for both demographic parity and equalized odds, and choose the one that produces the more favorable results.

Figure 14. Fairness Value vs. Dataset Bias Level for Bias Mitigation Techniques

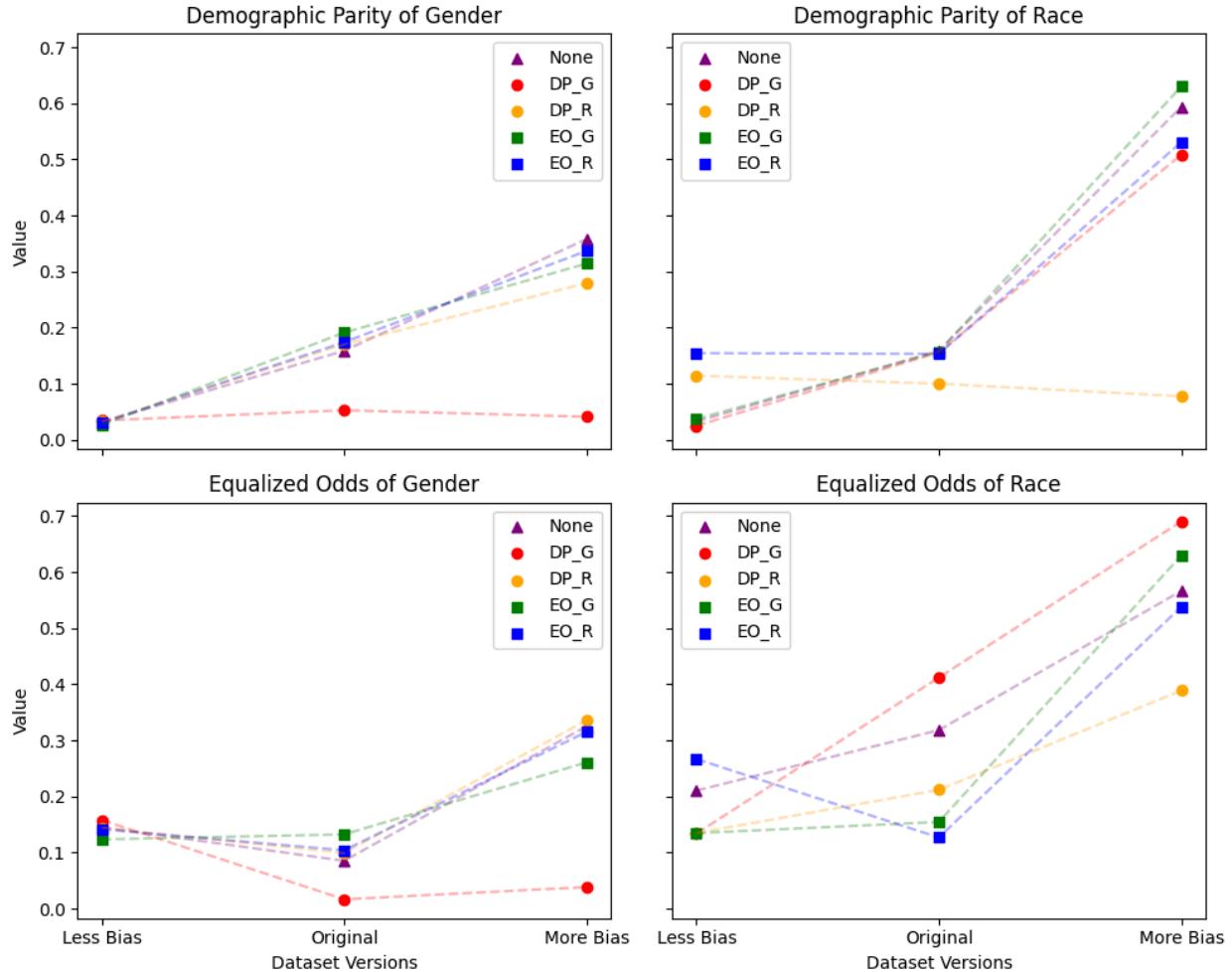
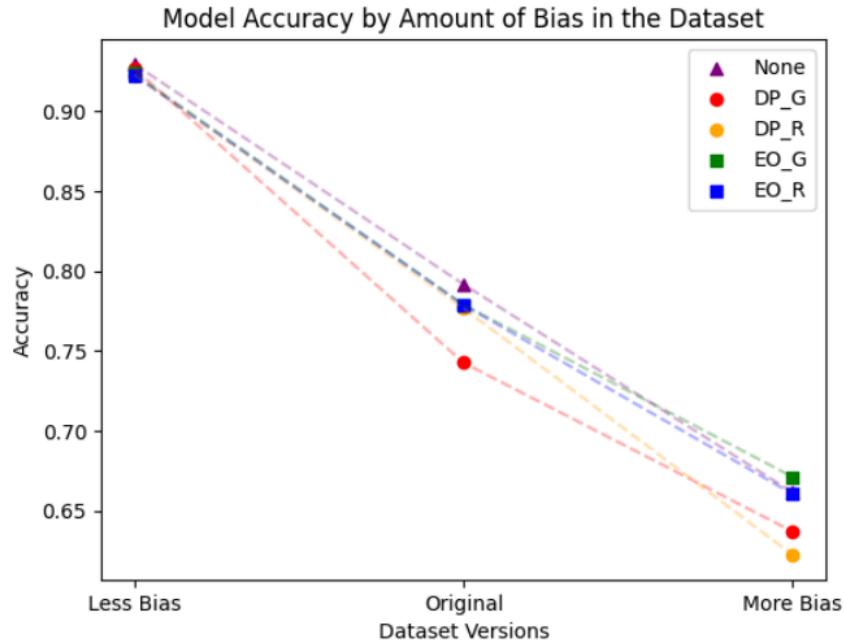


Figure 15 shows the accuracy vs dataset bias level for each of the bias mitigation strategies. This graph shows accuracy decreasing significantly based on the degree of bias in the dataset, as well as slightly lower accuracies for demographic parity bias mitigation strategies, which aligns with earlier findings. Taken with the previous figures, this indicates that a two-step approach of bias mitigation first in the dataset, then in the model when training, is the best process for minimizing bias in the model outputs.

Figure 15. Accuracy vs Dataset Bias Level for Bias Mitigation Techniques



Figures 16, 17, and 18 show the overall results for each dataset. Looking at this data reveals that no single bias mitigation strategy is best for all metrics. This means that when trying to mitigate multiple fairness metrics while maintaining high accuracy, any mitigation strategies chosen will result in trade-offs. Therefore, it is important to clearly define fairness for the specific use case in order to select priorities for fairness.

Figure 16. Heatmap of Bias Mitigation Strategies vs Metrics for the Less Bias Dataset

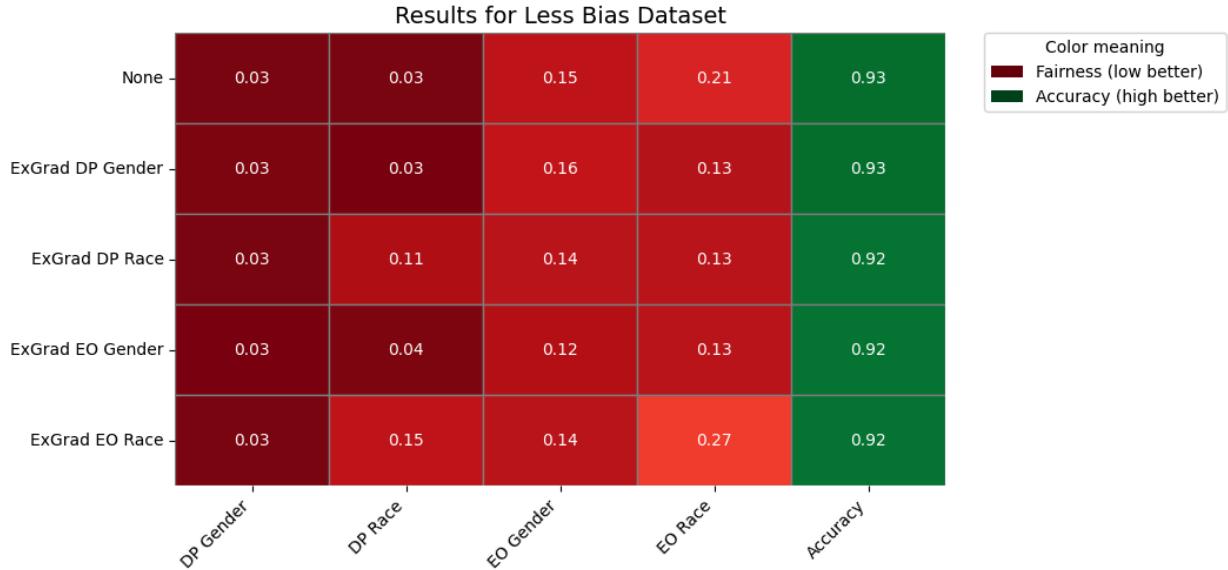


Figure 17. Heatmap of Bias Mitigation Strategies vs Metrics for the Original Dataset

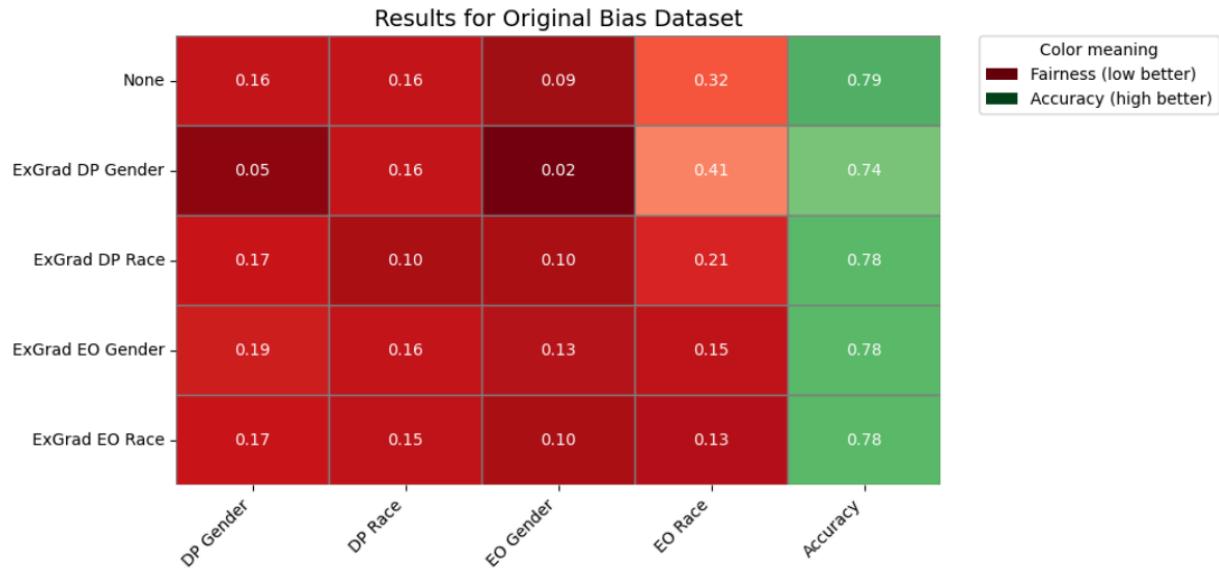
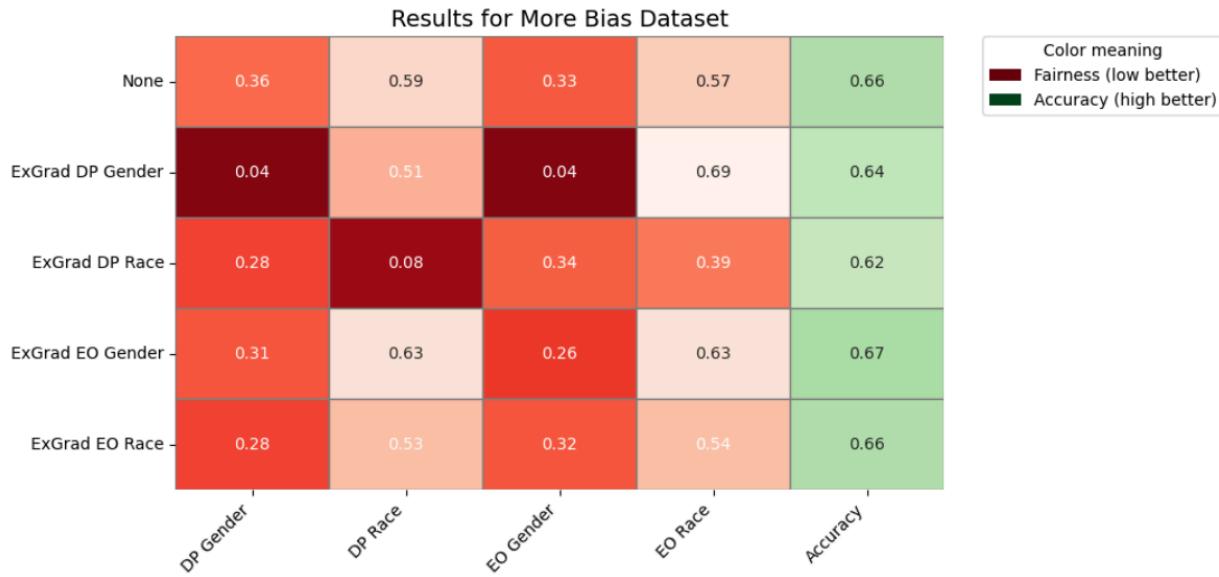


Figure 18. Heatmap of Bias Mitigation Strategies vs Metrics for the More Bias Dataset



## 5. Generative AI use

Generative AI, specifically ChatGPT and Gemini, was used during this project for various tasks to develop and complete the project. These tasks included assisting with creating a breakdown of the project into a timeline of weekly objectives and deliverables to guarantee fulfillment of the project goals within the time available, while also ensuring an even distribution of a workload that would take at least the minimum required number of hours for an ISP in order to maximize both time management and results. The project breakdown also included recommendations for starting off, such as potential options for datasets and libraries, which is where I discovered the UCI Adult Income dataset and fairlearn, in addition to other datasets and libraries not used. ChatGPT also provided the Python code used to produce the heatmap graphics. Gemini is built in to Google Colab, and was used for minor debugging and its automatic predictive text suggestions were used to more quickly generate sections of code that were repetitive, namely the lists of similar variables created in the comparison sections of the notebook (“Comparing Fairness Metrics from Each Version of the Dataset” and “Comparing Bias Mitigation Techniques-- Final Results, Part 1”).

## Glossary

1. Effectiveness of a bias mitigation technique on mitigating a particular fairness metric for a specific demographic is based on the comparison between the calculated fairness metric after the technique is implemented and the calculated fairness metric after no bias mitigation technique is implemented.
2. Impact of bias is the degree to which bias in the dataset affects bias in the results, as determined through the calculation of fairness metrics,
3. Bias is the unequal distribution of outcomes among demographic subgroups.
4. Demographic categories are the subsets of a demographic. For example, the demographic categories of race in the dataset used are White, Black, Asian/Pacific Islander, Native American/Eskimo, and Other.
5. Cleaning is removing data that is not useful for training or evaluation from the dataset.
6. Preprocessing is the process of transforming data into a format compatible with model training.
7. One-hot encoding is a method for preprocessing nominal categorical data by turning it into numerical data without introducing bias or arbitrary ordering, which can skew outputs. It involves turning each unique value within a nominal data column into its own feature. Each data instance has a 1 for the column for the feature that matches the value of the original column and a 0 in every other column.

## Sources Used

Bias

<https://www.chapman.edu/ai/bias-in-ai.aspx>  
<https://arize.com/blog/understanding-bias-in-ml-models/>

Dataset

<https://www.kaggle.com/datasets/wenruliu/adult-income-dataset>  
<https://archive.ics.uci.edu/dataset/2/adult>  
<https://www.cs.toronto.edu/~delve/data/adult/adultDetail.html>

Cleaning

<https://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf>

Preprocessing

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>

Model

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>

Fairness

<https://shelf.io/blog/fairness-metrics-in-ai/>  
[https://fairlearn.org/main/user\\_guide/assessment/common\\_fairness\\_metrics.html](https://fairlearn.org/main/user_guide/assessment/common_fairness_metrics.html)  
<https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-fairness-works.htm>  
[https://www.youtube.com/watch?v=es5nT7Qhj-w&ab\\_channel=PyLadiesAmsterdam](https://www.youtube.com/watch?v=es5nT7Qhj-w&ab_channel=PyLadiesAmsterdam)  
[https://fairlearn.org/main/api\\_reference/generated/fairlearn.metrics.MetricFrame.html](https://fairlearn.org/main/api_reference/generated/fairlearn.metrics.MetricFrame.html)  
[https://fairlearn.org/v0.6.2/user\\_guide/assessment.html#:~:text=The%20fairlearn..recall%20score\(\)%20.](https://fairlearn.org/v0.6.2/user_guide/assessment.html#:~:text=The%20fairlearn..recall%20score()%20.)

Hyperparameters

<https://ken-hoffman.medium.com/decision-tree-hyperparameters-explained-49158ee1268e>

Bias mitigation

[https://fairlearn.org/v0.5.0/user\\_guide/mitigation.html](https://fairlearn.org/v0.5.0/user_guide/mitigation.html)  
[https://fairlearn.org/main/api\\_reference/generated/fairlearn.reductions.ExponentiatedGradient.html#fairlearn.reductions.ExponentiatedGradient](https://fairlearn.org/main/api_reference/generated/fairlearn.reductions.ExponentiatedGradient.html#fairlearn.reductions.ExponentiatedGradient)  
[https://www.geeksforgeeks.org/machine-learning/fairlearn-assessing-and-improving-fairness-of-ai-systems/?utm\\_source=chatgpt.com](https://www.geeksforgeeks.org/machine-learning/fairlearn-assessing-and-improving-fairness-of-ai-systems/?utm_source=chatgpt.com)