# MLKFairness[1]

| | |
|---|---|
| Team Members | Amir Asiaee, Ph.D.<br>Assistant Professor<br>Department of Biostatistics<br>Vanderbilt University Medical Center<br>amir.asiaeetaheri@vumc.org<br><br>Kaveh Aryanpoo<br>Department of Informatics<br>King's College London<br>k.aryanpoo@gmail.com |
| Contact Information | amir.asiaeetaheri@vumc.org<br>(612)986-6639 |
| Video Documentation | https://youtu.be/LQ1Lk3V-4og |
| GitHub Repository | https://github.com/aryanpoo/mlkfairness |

## Abstract

We present a framework to detect and mitigate bias in AI/ML methods used in healthcare. Our solution includes a detection method that identifies five forms of bias and the FABulous algorithm (Fair Adaptive Boosting), which can both mitigate a selected bias in existing models or generate fair models from scratch. FABulous has a modular architecture that allows for adaptive bias mitigation over time, and stakeholders can probe the pipeline's success at various points, fostering trust and improving transparency. Additionally, our guidelines provide healthcare professionals with the flexibility to select fairness criteria from the five supported metrics for different healthcare settings. In summary, our framework aims to assist healthcare professionals in ensuring the fairness and impartiality of AI/ML methods used in healthcare while improving trust and transparency.

[1] MLK is for Martin Luther King (1929–1968) who "advanced civil rights for people of color in the United States through nonviolence and civil disobedience" (Wikipedia)

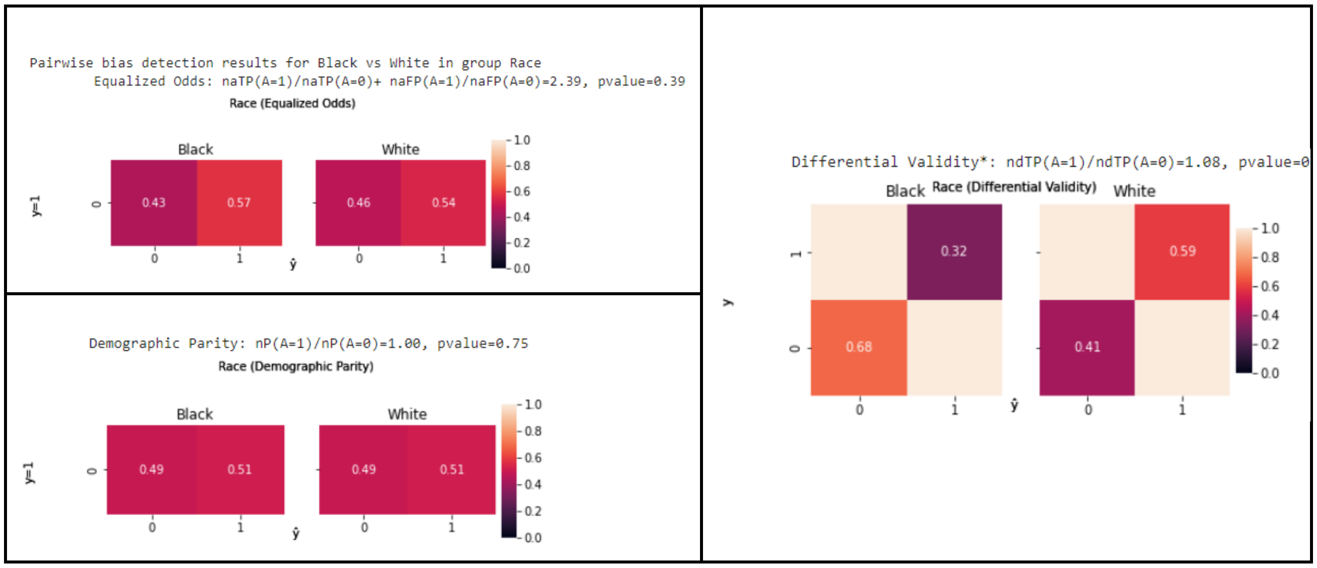
## 1. GitHub Code

Here is the link to our repository: https://github.com/aryanpoo/mlkfairness You can find the demo of bias detection here. We include a few images in this report. Our tool detects multiple forms of bias. Below we generate a random data set and use our `BiasDetector` object to generate a report. We show only a few outcomes, extreme colors suggest more bias.

```
from mlkfairness.detector import BiasDetector
from mlkfairness.data import Dataset, load_hmda, get_random_dataset
```

```
ds = get_random_dataset()
```

```
adetector = BiasDetector(ds)
ret = adetector.generate_report()
```



```
Pairwise bias detection results for Black vs White in group Race
        Equalized Odds: naTP(A=1)/naTP(A=0)+ naFP(A=1)/naFP(A=0)=2.39, pvalue=0.39
```

```
Demographic Parity: nP(A=1)/nP(A=0)=1.00, pvalue=0.75
```

```
Differential Validity*: ndTP(A=1)/ndTP(A=0)=1.08, pvalue=0
```

## 2. Methodology Overview

AI/ML methods can revolutionize healthcare by improving outcomes and reducing costs. However, bias can have serious consequences in healthcare. Our team has developed a framework for detecting and mitigating bias in AI/ML methods. **Our unique approach aims to detect and mitigate many forms of biases, track and correct them dynamically over time, and is adaptable to any AI/ML model, making it highly flexible for a wide range of healthcare scenarios**.

Our framework has two main components. The first is a detection method that identifies various forms of bias and alerts us if the method becomes biased over time due to complex healthcare processes or latent biases. The second is our **FABulous algorithm, a fair version of the popular Adaptive Boosting algorithm** that generates unbiased machine learning models which can incorporate *any classifier*. FABulous has a modular architecture that enables us to extend it and adaptively mitigate biases emerging over time, and it can also accept any pre-trained ML model and debias its outcome.

Our framework is not a replacement for human judgment but rather a tool that healthcare professionals with both technical and healthcare expertise can use to select and mitigate one or more biases. Our framework provides various measures of bias for human judgment to ensure that critical decision-making remains in the hands of humans. To assist in the selection of relevant fairness criteria for each task, we have included guidelines in our report for different healthcare settings. We recognize the importance of selecting appropriate fairness criteria and have taken steps to support this critical decision-making process.

| # | Covariates (Features) | Binary Protected Features | Model Label (Truth) | Binary Outcome (Prediction) | Time of Prediction |
|---|---|---|---|---|---|
| 1 | $x_1 = (x_1^{(1)}, \ldots, x_p^{(1)})$ | $a_1 = (a_1^{(1)}, \ldots, a_s^{(1)})$ | $y_1$ | $\hat{y}_1$ | $t = 1$ |
| 2 | $x_2 = (x_1^{(2)}, \ldots, x_p^{(2)})$ | $a_2 = (a_1^{(2)}, \ldots, a_s^{(2)})$ | $y_2$ | $\hat{y}_2$ | $t = 5$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $n$ | $x_n = (x_1^{(n)}, \ldots, x_p^{(n)})$ | $a_n = (a_1^{(n)}, \ldots, a_s^{(n)})$ | $y_n$ | $\hat{y}_n$ | $t = 140$ |

**Table 1**. **Machine Learning Dataset with Protected Feature One-Hot Encoded.** This table displays a machine learning dataset that includes a protected feature (e.g., race) that is represented in vector $a$ as a one-hot encoding, where each element of the vector $a$ represents a value for the protected feature (white, black, Asian, etc.), and exactly one of them is non-zero. Fairness metrics are used to evaluate fairness toward a specific subgroup determined by one element of vector $a$ (e.g., black) compared to a baseline subgroup (e.g., white). During training, the ML model has access to samples from triplets $(x_i, a_i, y_i)$, and then the trained model is used to predict the outcome for new samples $\{(x_i, a_i, y_i)\}_{\{i=1\}}^{\{n\}}$ at time points \$$t$\$ during either test or deployment phases. To preserve privacy, the bias detection method has access only to $\{(a_i, y_i, \hat{y}_i)\}_{i=1}^{n}$, and sometimes, information about the time of prediction, i.e., $t$.

## 2.1. Metrics

Our focus is solely on binary outcomes, as per competition guidelines, limiting our analysis to classification methods and associated notions of bias and fairness. Table 1 shows a typical training dataset for classification. We identified eleven definitions of fairness through a literature review, classified into oblivious and non-oblivious measures. Oblivious measures only use $(Y, A, \hat{Y}\ )$ to define fairness, where $Y$ is the true label, $A$ represents the protected feature, and $\hat{Y}$ is the predicted outcome, Table 2. Out of the eleven definitions, five fall under the category of oblivious measures and involve equality constraints with conditional probabilities, representing social and predictive biases, as shown in Table 3. However, non-oblivious measures that depend on sample features $(X)$ cannot be detected due to the problem statement's restrictions.

| Variable | Description | Domain |
|---|---|---|
| $Y$ | Model Label | $\{-1, +1\}$ |
| $\hat{Y}$ | Binary Outcome | $\{-1, +1\}$ |
| $A$ | Protected Feature | $\{0, 1\}$ |

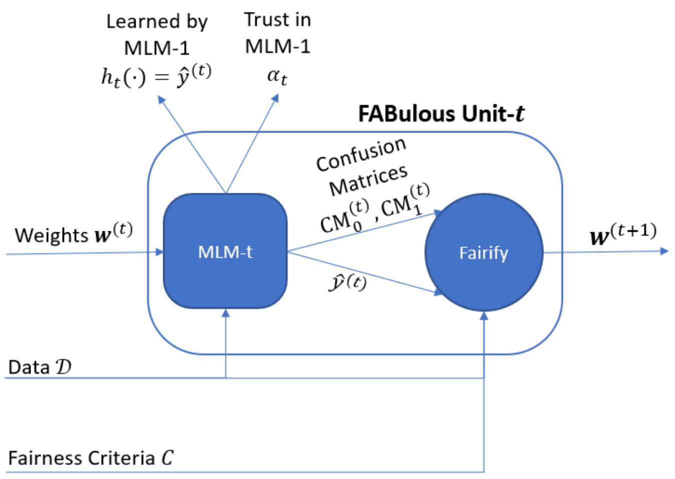
**Table 2. Random variables and their domains available for bias detection.** The binary label and outcome are assumed to take values from $\{-1, +1\}$. The protected attribute is binary and takes the value of $0$ for the baseline group (e.g., white) and $1$ for the minority group (e.g., black) under investigation.

## 2.2. Detection and Mitigation of Latent Bias

To future-proof bias detection tools, we must detect both retrospective and prospective biases. We use the regular triplet $(Y, A, \hat{Y})$ and time $T$ for statistical tests on different bias types, Table 3. Users can set a **stride parameter**, and an adaptive time window's size is determined based on the **power calculation** for the required statistical test of fairness. Based on a significance level of $\alpha = 0.05$ and a power of $1 - \beta = 0.80$, we estimated that a sample size of $n = 87$ participants would be required to detect a significant difference with a minimum effect size of $0.3$ (medium effect size) using a two-sided chi-square test with one degree of freedom. Thus, after each stride, we expand the time window sufficiently to contain at least $n = 87$ samples and re-run the statistical test for bias detection. We notify the user if the test passes statistical significance at the $0.05$ level.

| Criterion Name | Definition ($y \in \{-1, +1\}$) | Corresponding Independence Assumption | Bias Type |
|---|---|---|---|
| **Equalized Odds** | $\mathbb{P}(\hat{Y} = 1 \mid Y = y, A = 0) = \mathbb{P}(\hat{Y} = 1 \mid Y = y, A = 1)$ | $\hat{Y} \perp\!\!\!\perp A \mid Y$ | Social |
| **Equal Opportunity** | $\mathbb{P}(\hat{Y} = 1 \mid Y = 1, A = 0) = \mathbb{P}(\hat{Y} = 1 \mid Y = 1, A = 1)$ | $\hat{Y} \perp\!\!\!\perp A \mid Y = 1$ | Social |
| **Demographic Parity** | $\mathbb{P}(\hat{Y} = 1 \mid A = 0) = \mathbb{P}(\hat{Y} = 1 \mid A = 1)$ | $\hat{Y} \perp\!\!\!\perp A$ | Social |
| **Calibration Fairness** | $\mathbb{P}(Y = 1 \mid \hat{Y} = y, A = 1) = \mathbb{P}(Y = 1 \mid \hat{Y} = y, A = 0)$ | $Y \perp\!\!\!\perp A \mid \hat{Y}$ | Predictive |
| **Differential Validity** | $\mathbb{P}(\hat{Y} = Y \mid A = 0) = \mathbb{P}(\hat{Y} = Y \mid A = 1)$ | $Y = \hat{Y} \perp\!\!\!\perp A$ | Predictive |

**Table 3. Definitions of Oblivious Fairness Metrics and Their Corresponding Bias Types.** The table presents the mathematical and statistical definitions of various fairness metrics that are computable using only $(Y, A, \hat{Y})$, i.e., they are oblivious. For bias detection, we tested the conditional independence assumption of each metric. The table also shows the bias types that each metric is designed to detect. Social bias occurs when the predicted outcome $\hat{Y}$ is dependent on the protected subgroup, while predictive bias happens when the predicted outcome $\hat{Y}$ is mapped to the true outcome $Y$ at different rates for different subgroups.



Learned by MLM-1 $h_t(\cdot) = \hat{y}^{(t)}$
Trust in MLM-1 $\alpha_t$
FABulous Unit-$t$
Confusion Matrices $CM_0^{(t)}, CM_1^{(t)}$
Weights $\boldsymbol{w}^{(t)}$
MLM-t
Fairify
$\boldsymbol{w}^{(t+1)}$
$\hat{y}(t)$
Data $\mathcal{D}$
Fairness Criteria $C$

**Figure 1. Illustration of a FABulous Unit.** The diagram shows how data and their corresponding weights are input into a machine learning model (MLM), which generates a classifier function $h_t(\cdot)$ and its trust weight $\alpha_t$. The model's predictions and confusion matrices are then sent to the Fairify function, which balances the weights for the next classifier to meet fairness criteria $C$. This process ensures that the final model is both accurate and fair.

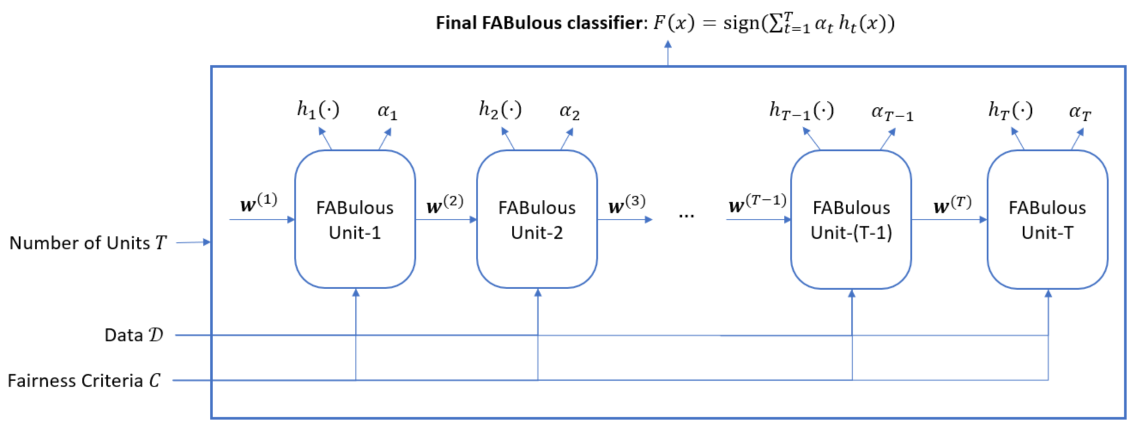


**Figure 2. Illustration of the FABulous Algorithm.** Overview of the FABulous Algorithm. Each FABulous unit generates a classifier and a trust weight and reweights the data for the next unit to account for both errors and biases in the previous classifier. The final FABulous classifier is the sign of the weighted sum of the classifiers produced by each unit, ensuring that it addresses both accuracy and fairness in the resulting model.

## 2.3. Creative Solution of Bias Mitigation

Our approach to addressing bias in ML models is a versatile method that can be applied in various ways. Initially, it can be used to construct an unbiased FABulous classifier that adheres to the chosen fairness criteria, as shown in Figures 1 and 2. In the event that the FABulous classifier develops bias over time, our detection technique will alert the user, and we can train a new FABulous classifier with the initial one as its first unit, as depicted in Figure 3. Finally, by incorporating the specified model as the first unit of the FABulous classifier, as illustrated in Figure 4, we can debias any unfair ML model within our framework.

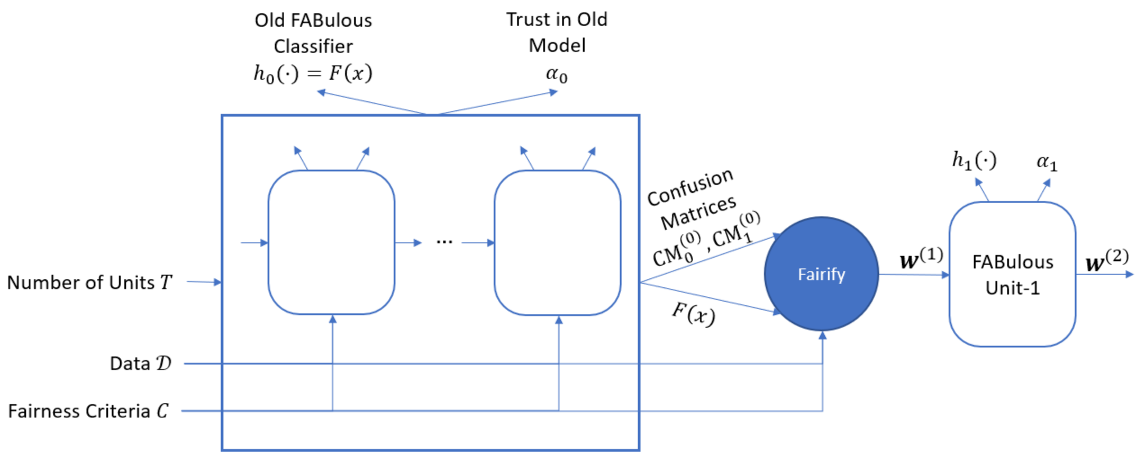


**Figure 3. Adaptive FABulous Algorithm for responding to change in bias.** The diagram shows a square representing a previously trained FABulous classifier that may become biased over time due to the complexity of the healthcare system. To address this issue, the biased classifier is treated as the "unit zero" of an encompassing FABulous classifier. Its predictions and confusion matrices are sent to the Fairify function to generate weights for the first unit of the encompassing FABulous algorithm, ensuring that the resulting model addresses changes in bias over time.

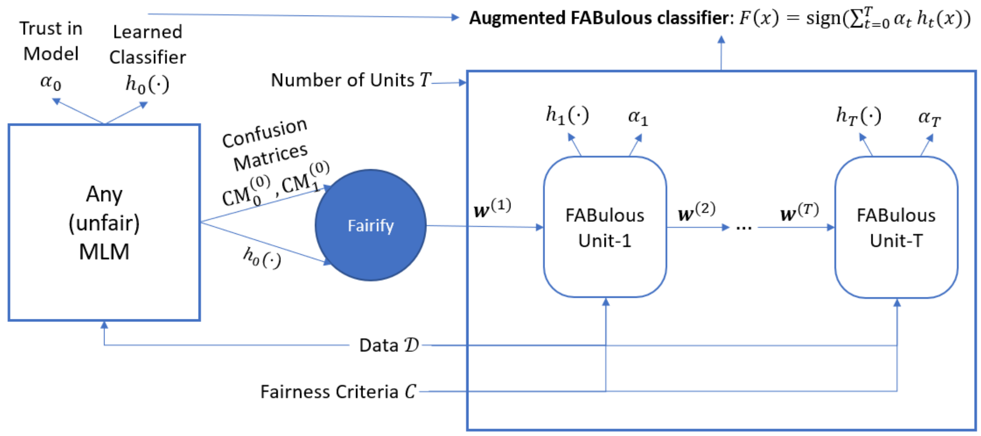


**Figure 4. A Scheme to Use FABulous for Fairifying Potentially Unfair Machine Learning Classifiers.** This scheme involves treating the learned classifier of the first ML model (MLM) as the "unit zero" of a larger augmented FABulous classifier. To address potential unfairness in an existing MLM, the confusion matrix and predictions are passed to Fairify, which generates fair weights and initializes the FABulous algorithm. The resulting "Augmented FABulous classifier" combines the input unfair MLM and the outcome of the FABulous method, resulting in a fair and accurate machine learning model.

## 2.4. Inspiration for FABulous

FABulous is a variation of AdaBoost, which does binary classification by combining several weak classifiers, Algorithm 1. We train new weak classifiers on weighted training data using an adjustment scheme called Fairify, which increases accuracy and reduces bias, Algorithm 2. The final strong classifier is a weighted combination of weak classifiers based on their accuracy.

```
Algorithm 1 FABulous Algorithm
Require:
  Training dataset: $\mathcal{D} = \{(x_i, y_i, a_i)\}_{i=1}^{n}$,
  The number of weak classifiers: $T$,
  Fairness criteria: $C$.
Ensure: A strong classifier $F(x)$.
  Initialize weights $\forall i \in [n] : \mathbf{w}_i^{(t)} = \frac{1}{n}$.
  for $t = 1$ to $T$ do
    Train $t$th classifier: $\hat{y}^{(t)} \triangleq h_t(x) \leftarrow \text{Train}(\mathcal{D}, \mathbf{w}^{(t)})$.
    Compute weighted error: $\epsilon_t = \sum_{i=1}^{n} w_i^{(t)} \mathbb{1}[h_t(x_i) \neq y_i]$
    Compute weight of classifier: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
    for each value of the protected feature $a \in \{0, 1\}$ do
      $\text{TP}_a^{(t)} = \sum_{i=1, y_i=+1, a_i=a}^{n} w_i^{(t)} \mathbb{1}[\hat{y}_i^{(t)} = y_i]$, $\text{FN}_a^{(t)} = \sum_{i=1, y_i=+1, a_i=a}^{n} w_i^{(t)} \mathbb{1}[\hat{y}_i^{(t)} \neq y_i]$
      $\text{FP}_a^{(t)} = \sum_{i=1, y_i=-1, a_i=a}^{n} w_i^{(t)} \mathbb{1}[\hat{y}_i^{(t)} \neq y_i]$, $\text{TN}_a^{(t)} = \sum_{i=1, y_i=-1, a_i=a}^{n} w_i^{(t)} \mathbb{1}[\hat{y}_i^{(t)} = y_i]$
      Compute the confusion matrix: $\text{CM}_a^{(t)} = (\text{TP}_a^{(t)}, \text{FN}_a^{(t)}, \text{FP}_a^{(t)}, \text{TN}_a^{(t)})$
    end for
    Compute fairness balancing vector: $\mathbf{b}^{(t)} \leftarrow \text{Fairify}(\mathcal{D}, C, \hat{y}^{(t)}, \text{CM}_0^{(t)}, \text{CM}_1^{(t)})$.
    Update the weights $w_i^{(t+1)} = w_i^{(t)} b_i^{(t)} \exp\{-\alpha_t y_i h_t(x_i)\}$
    Normalize the weights: $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t+1)} / \sum_{i=1}^{n} w_i^{(t+1)}$
  end for
  Output the final strong classifier: $F(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\right)$
```

**Algorithm 1. Fair Adaptive Boosting (FABulous) Algorithm.** Similar to Adaboost[2], FABulous trains $T$ weak classifiers sequentially using weighted data, with the weights updated based on the errors made by previous classifiers. The focus of each subsequent classifier is on correcting the samples that the previous classifiers have misclassified. *In FABulous, we ensure fairness in the weight update scheme by using the Fairify function*, which modifies the original weight update scheme to satisfy fairness criteria specified by $C$. These weight corrections are saved in vector $\mathbf{b}$.

[2] Schapire, R. E. (2013). Explaining AdaBoost. *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, 37-52.

```
Algorithm 2 Fairify Procedure
Require:
  Training dataset: $\mathcal{D} = \{(x_i, y_i, a_i)\}_{i=1}^n$,
  Current predictions: $\hat{y} = \{\hat{y}_i\}_{i=1}^n$
  Fairness criteria: $C$,
  Confusion matrices:
  $\mathrm{CM}_0 = (\mathrm{TP}_0, \mathrm{FN}_0, \mathrm{FP}_0, \mathrm{TN}_0), \mathrm{CM}_1 = (\mathrm{TP}_1, \mathrm{FN}_1, \mathrm{FP}_1, \mathrm{TN}_1)$.
Ensure: Balancing vector $\mathbf{b} = (b_1, \ldots, b_n)$
  $\forall i \in [n] : b_i = 1$.
  switch $C$ do
    case Equalized Odds
      $b = \sqrt{(\mathrm{TN}_1\mathrm{FP}_0)/(\mathrm{TN}_0\mathrm{FP}_1)}$
      $b' = \sqrt{(\mathrm{TP}_1\mathrm{FN}_0)/(\mathrm{TP}_0\mathrm{FN}_1)}$
      Balance $\mathrm{FP}_1$: $\forall i$ if $a_i = 1$, $y_i \neq \hat{y}_i = +1$: $b_i \leftarrow b$
      Balance $\mathrm{FP}_0$: $\forall i$ if $a_i = 0$, $y_i \neq \hat{y}_i = +1$: $b_i \leftarrow 1/b$
      Balance $\mathrm{FN}_1$: $\forall i$ if $a_i = 1$, $y_i \neq \hat{y}_i = -1$: $b_i \leftarrow b'$
      Balance $\mathrm{FN}_0$: $\forall i$ if $a_i = 0$, $y_i \neq \hat{y}_i = -1$: $b_i \leftarrow 1/b'$
    case Equal Opportunity
      $b = \sqrt{(\mathrm{TP}_1\mathrm{FN}_0)/(\mathrm{TP}_0\mathrm{FN}_1)}$
      Balance $\mathrm{FN}_1$: $\forall i$ if $a_i = 1$, $y_i \neq \hat{y}_i = -1$: $b_i \leftarrow b$
      Balance $\mathrm{FN}_0$: $\forall i$ if $a_i = 0$, $y_i \neq \hat{y}_i = -1$: $b_i \leftarrow 1/b$
    case Demographic Parity
      $b = \sqrt{\frac{\mathrm{TP}_0+\mathrm{FP}_0}{\mathrm{TN}_0+\mathrm{FN}_0}\frac{\mathrm{TN}_1+\mathrm{FN}_1}{\mathrm{TP}_1+\mathrm{FP}_1}}$
      Balance positive prediction for group 1: $\forall i$ if $a_i = 1$, $\hat{y}_i = +1$: $b_i \leftarrow b$
      Balance positive prediction for group 0: $\forall i$ if $a_i = 0$, $\hat{y}_i = +1$: $b_i \leftarrow 1/b$
    case Calibration Fairness
      $b = \sqrt{(\mathrm{TN}_1\mathrm{FN}_0)/(\mathrm{TN}_0\mathrm{FN}_1)}$
      $b' = \sqrt{(\mathrm{TP}_1\mathrm{FP}_0)/(\mathrm{TP}_0\mathrm{FP}_1)}$
      Balance $\mathrm{FN}_1$: $\forall i$ if $a_i = 1$, $y_i \neq \hat{y}_i = -1$: $b_i \leftarrow b$
      Balance $\mathrm{FN}_0$: $\forall i$ if $a_i = 0$, $y_i \neq \hat{y}_i = -1$: $b_i \leftarrow 1/b$
      Balance $\mathrm{FP}_1$: $\forall i$ if $a_i = 1$, $y_i \neq \hat{y}_i = +1$: $b_i \leftarrow b'$
      Balance $\mathrm{FP}_0$: $\forall i$ if $a_i = 0$, $y_i \neq \hat{y}_i = +1$: $b_i \leftarrow 1/b'$
    case Differential validity
      $b = \sqrt{\frac{\mathrm{TP}_0+\mathrm{TN}_0}{\mathrm{FP}_0+\mathrm{FN}_0}\frac{\mathrm{FP}_1+\mathrm{FN}_1}{\mathrm{TP}_1+\mathrm{TN}_1}}$
      Balance error for group 1: $\forall i$ if $a_i = 1$, $y_i \neq \hat{y}_i$: $b_i \leftarrow b$
      Balance error for group 0: $\forall i$ if $a_i = 0$, $y_i \neq \hat{y}_i$: $b_i \leftarrow 1/b$
  Output the balancing vector $\mathbf{b} = (b_1, \ldots, b_n)$
```

**Algorithm 2. Fairify Procedure.** This algorithm supports fairness defined in five different ways in the literature, each represented by a conditional probability equation. The probabilities are based on the confusion matrix entries, including false positives and negatives and true positives and negatives. The algorithm generates a balancing vector that adjusts the confusion matrix entries to ensure that the fairness criterion $C$ is satisfied, resulting in a fair ML model. For instance, to equalize false negative rates, Fairify may scale down (up) samples in protected (baseline) groups to balance the importance of correctly classifying false negatives between the two groups. The final classifier is the sign of the weighted sum of all classifiers.

## 3. Value Proposition

Our bias detection method can identify all known (oblivious) biases, including social and predicted biases, making it applicable to various healthcare settings. Our dynamic bias tracker is a new tool that detects emerging biases. The FABulous units in our model use an arbitrary classification method, allowing for flexibility, Figure 2. Our modular framework permits bias mitigation at different stages in the pipeline, including pre-processing (Figure 3), in-processing (Figure 2), and post-processing (Figure 4).

## 4. Healthcare Scenario

To begin with, a fairness criterion should be chosen for each specific scenario (Refer to Section 6 on “Sustainability Plan”). We can use FABulous with the time-varying bias detection algorithm to train a model from scratch. If the dynamic bias detection method detects bias, a new training phase can be initiated, using the old model as Unit zero of the new one, as shown in Figure 3. If there is an existing biased ML model in a medical scenario, we can probe it using our bias detection method and use the Augmented FABulous classifier described in Figure 4 to make its predictions fair. Note that the detection and mitigation scenarios are agnostic to the root cause of bias, and both work regardless of whether the bias is predictive or social.

## 5. Operational Requirements

Below are the necessary details to run our tool:

- OS: The tool is designed to run on Windows, Mac, and Linux operating systems.
- Memory (RAM): The minimum requirement for running the tool is 4GB of RAM.
- Disk Space: The main determinant of required disk space is the size of the input dataset. But we suggest a minimum of 1GB of disk space for installation and regular usage.
- CPU/GPU: The tool is optimized for multi-core CPUs and does not require a GPU.
- Environment Configurations: The tool requires Python 3.6 or higher to be installed on the system. Additionally, it requires the installation of several Python packages, such as NumPy, Pandas, and Scikit-Learn, which can be installed using the pip package manager.

The tool's design is technology-agnostic, ensuring that it can be deployed widely without dependencies on any specific vendors or proprietary information exchange standards that would limit the tool's impact. We acknowledge that the tool will be distributed under the BSD 3 license.

## 6. Sustainability Plan

The main challenge in applying any bias detection and mitigation algorithm is selecting a domain-appropriate notion of fairness or bias. It's critical to have a human-in-the-loop model where project goals are discussed with stakeholders, including decision-makers and those affected by the model's application, to identify different perspectives on fairness and equity. Facilitators should understand medical decisions, statistical issues, and affected individuals' preferences and options. The trade-off between utility and fairness and between different fairness metrics should be considered. One future research direction is finding a Pareto optimal solution to this multi-objective problem. Additionally, one should consider the nature of the subsequent decision or action, whether it's punitive or assistive, and the potential harm involved in making different types of errors. For punitive interventions, we should focus on false positives metrics to avoid harming individuals, and for assistive interventions, we should focus on false negatives to avoid failing to intervene when needed.

## 7. Generalizability Plan

Our tool has been designed with portability and extensibility in mind. This means that it can be easily adapted to various healthcare settings and clinical decisions, even though it was developed and tested in a particular technical environment. We have made specific design choices to optimize the tool's usability and make it easy to extend and apply to other healthcare scenarios.

To enhance the portability of our tool, we have taken measures to standardize its inputs and outputs, enabling easy integration with various healthcare systems. Furthermore, our tool can be deployed on any operating system with minimal hardware requirements, making it highly adaptable to different technical environments. Notably, our tool is designed to monitor multiple fairness metrics over time and adaptively mitigate biases in any given ML model, making it a robust and versatile solution. The modular and scalable architecture of our tool allows it to respond effectively to changes in data volume and detect and mitigate time-varying biases. To further optimize the tool's usability, we have incorporated user-friendly visualizations of bias detection results and actionable recommendations for bias mitigation, ensuring that it is easy to use and understand even for non-technical users. Our modular framework, consisting of an ML model and a Fairyfier unit (FABulous), makes it easy for non-technical individuals to understand and probe. This fosters trust in the framework and helps stakeholders to understand the amount of bias reduced by the model at different levels of the architecture. Furthermore, it can be used to quantify and visualize the fairness-utility tradeoff. We believe that our tool can be implemented as is or with minimal additional support, and we are confident that it will improve health equity and fairness in AI/ML in various healthcare settings and disciplines.

In the future, we plan to enhance the accessibility of our tool by providing detailed documentation and releasing the core code as a freely available software package. Additionally, we aim to continue expanding the tool's capabilities to address additional healthcare scenarios and clinical decisions. **To achieve fairness, we plan to explore Pareto optimal solutions that balance various fairness criteria**. We also aim to develop advanced visualizations that illustrate the fairness-utility trade-offs involved in decision-making. **A crucial research direction is investigating the role of AI explainability in detecting the root cause of biases**. By utilizing explainable ML models, we can identify whether imbalances in predictions stem from model sensitivity to certain features and, if so, whether those features act as proxies for protected features. If the bias is unexplainable by any subset of features, the ML model may be inherently biased, and further investigation can uncover the root cause. **It is essential to note that we require an AI explainability framework that is not specific to a particular ML model to maintain the flexibility of our solution.** Developing such a framework is an active research area, and some notable progress has been made[3]. Additionally, the current body of research on fairness has mainly focused on classification tasks, overlooking various healthcare applications where the target outcome is continuous. These include predicting hospitalization duration or overall survival. **Therefore, we aim to broaden our framework to mitigate bias in regression models.** We anticipate that, as adaptive boosting has been applied to regression problems, we can extend FABulous to address fair continuous outcome prediction, although further investigation is needed to determine the specifics. We are excited to continue refining and improving our tool while exploring its potential applications in the years to come.

## 8. Implementation Requirements

Our bias detection and mitigation tool can be scaled to monitor the fairness of any number of ML algorithms across multiple locations. The detection tool only needs access to the (timed) triplet $(Y, A, \hat{Y})$, for each prediction task in each location. However, to ensure effective bias mitigation, we recommend that the tool is trained on the exact same target population that it is going to be used for. This is because any change in the joint distribution of data (i.e., $\mathbb{P}(Y, A, \hat{Y})$ can affect the bias, and any debiasing attempt needs to consider it.

The issue of non-transportability of ML models is widely recognized, and transfer learning methods have been developed to address this issue. Ideally, a central institute such as the NIH or a large medical center would have access to enough data to train a fair ML model and share it with smaller hospitals for fine-tuning in both prediction and fairness aspects. However, ensuring fairness in the context of transfer learning is a complex research direction beyond the scope of our current project. Therefore, we recommend that our bias mitigation tool be trained on the same target population in which it will be deployed. This approach will help to ensure that the tool's performance remains consistent across different locations and populations and that the risk of introducing new biases is minimized.

To deploy the tool in a real-world setting, we recommend a multidisciplinary team consisting of healthcare professionals, data scientists, and software developers. Figure 5 details our Standard

[3] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

Operating Procedure (SOP) workflow, which takes into account the expertise needed for implementation in real-world settings. We also provide guidance on the necessary resources and requirements for the successful implementation of our SOPs.

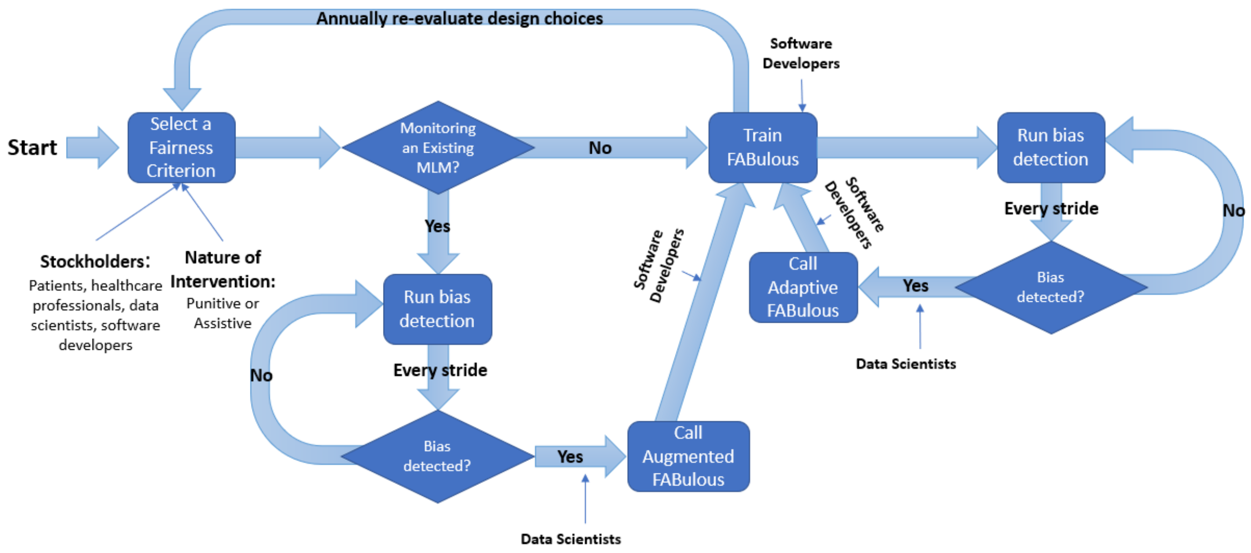


**Figure 5. Workflow of the Standard Operating Procedure.** The diagram depicts the functioning of our pipeline and identifies decision points and the required expertise for each stage. The dynamic bias detection method examines bias at every stride, which is a user-defined parameter ranging from weeks to years. We can monitor an existing (trained) ML model for bias and mitigate it using Augmented FABulous or train FABulous from scratch and monitor it. Upon detection of emerging bias, we call Adaptive FABulous for mitigation. Stakeholders must evaluate the system outcome at least annually to determine if the chosen fairness measures and ML model align with the ultimate goals or require modification.

To concretely measure the success of the implementation of our framework, we recommend tracking the combined outcome of our detection and mitigation strategies at multiple levels. At the lowest level, we can measure the amount of the disparity under the study before and after using one FABulous unit (Figure 1), as well as the whole FABulous algorithm (Figure 2), the adaptive FABulous procedure (Figure 3), and the Augmented FABulous method (Figure 4).

Measuring the impact of our bias detection and mitigation tool on patient outcomes and healthcare delivery systems is crucial for assessing its success. To do so, we recommend involving stakeholders and tracking key performance indicators, including patient satisfaction, hospital readmissions, and cost savings. Regular reviews and audits should also be conducted to ensure the tool is functioning as intended and meeting the needs of end-users. These evaluations should aim to determine if the selected fairness criteria remain valid in light of new data, identify whether biases are introduced over time, and assess the reasons behind any observed changes in bias (i.e., the method itself or input data). Finally, the most critical aspect of these evaluations is to determine if the bias mitigation has the intended consequences on patients' well-being and satisfaction.

## 9. Lessons Learned

Our team faced several challenges during the challenge, including defining fairness, mitigating multiple biases simultaneously, and the lack of a universal framework for debiasing ML models. The most important lesson we learned is that bias detection, and mitigation ultimately require human decision-making. We must carefully select fairness metrics and consider the intended and unintended consequences of removing the corresponding bias. While mathematical fairness notions are useful in clarifying problem domain assumptions, we must acknowledge that monitoring and intervention by humans are essential for successful bias mitigation. To tackle the challenge of multiple fairness definitions, our team took a comprehensive approach by identifying as many biases as possible simultaneously and tracking their changes over time. Furthermore, we developed a universal framework for mitigating bias in any ML model, which also enables the creation of a fair model from scratch with any ML algorithm as its core.

To enhance the fairness of AI/ML models in healthcare settings, we recommend that NCATS and other organizations prioritize investing in a flexible and universal framework for bias detection and mitigation. Such a framework should have standardized inputs and outputs at each level of the pipeline, allowing different healthcare systems to collectively monitor and correct biases in ML algorithms. In addition, we suggest promoting the development of transparent and explainable AI/ML models, which can provide insights into the source of emergent biases.