FROC: Building Fair ROC from a Trained Classifier

Avyukta Manjunatha Vummintala
Machine Learning Lab
IIIT Hyderabad
avyukta.v@research.iiit.ac.in

Shantanu Das
Machine Learning Lab
IIIT Hyderabad
shantanu.das@alumni.iiit.ac.in

Sujit Gujar
Machine Learning Lab
IIIT Hyderabad
sujit.gujar@iiit.ac.in

December 18, 2024

Abstract

This paper considers the problem of fair probabilistic binary classification with binary protected groups. The classifier assigns scores, and a practitioner predicts labels using a certain cut-off threshold based on the desired trade-off between false positives vs. false negatives. It derives these thresholds from the ROC of the classifier. The resultant classifier may be unfair to one of the two protected groups in the dataset. It is desirable that no matter what threshold the practitioner uses, the classifier should be fair to both the protected groups; that is, the $\mathcal{L}_p$ norm between FPRs and TPRs of both the protected groups should be at most ε. We call such fairness on ROCs of both the protected attributes ε_p-Equalized ROC. Given a classifier not satisfying ε_1-Equalized ROC, we aim to design a post-processing method to transform the given (potentially unfair) classifier's output (score) to a suitable randomized yet fair classifier. That is, the resultant classifier must satisfy ε_1-Equalized ROC. First, we introduce a threshold query model on the ROC curves for each protected group. The resulting classifier is bound to face a reduction in AUC. With the proposed query model, we provide a rigorous theoretical analysis of the minimal AUC loss to achieve ε_1-Equalized ROC. To achieve this, we design a linear time algorithm, namely FROC, to transform a given classifier's output to a probabilistic classifier that satisfies ε_1-Equalized ROC. We prove that under certain theoretical conditions, FROC achieves the theoretical optimal guarantees. We also study the performance of our FROC on multiple real-world datasets with many trained classifiers.

1 Introduction

The use of *Machine Learning based Models* (MLM) in decision-making is prevalent today. Practitioners use MLMs' predictions in college admissions, credit scores, recidivism, employment, recommender systems, etc. [1, 2]. However, there have been several reports of such MLMs discriminating against individuals belonging to certain groups based on *protected attribute* such as gender, age, race, color, and religion. E.g., in [3], predictive models are found to be biased against the black population, or the Amazon recruitment team has to stop using the AI tool for shortlisting candidates as it was biased against females [4]. [5]; [2]; [6] show that many of such predictive models are unfair to females. Such unfair instances have driven researchers toward building a fair MLM.

An MLM that achieves fairness with the least possible compromise on traditional performance guarantees such as accuracy is *desirable* MLM. Building a desirable MLM involves two main steps: a) formalizing and quantifying a fairness measure and b) designing algorithms to train MLM for quantified fairness. Researchers proposed many fairness measures, majorly belonging to two categories: (i) *individual fairness* [7] – individuals with similar input features receive similar decision treatment irrespective of their protected attribute. (ii) *Group fairness* – a particular statistical property must be similar across each protected group, e.g., *Disparate Impact (DI)*,*Equalized odds (EO)* [8].

Building Fair MLM Fair machine learning models (MLMs) can be developed by targeting different stages of the model training cycle. Approaches include: (i) *Pre-processing* methods, which act on input data to eliminate bias [9, 10]. (ii) *In-processing* algorithms, which intervene during training to incorporate fairness as a constraint or within the learning objective [11]. (iii) *Post-processing* methods, which adjust the outputs of trained MLMs to produce fair results, requiring access to sensitive attributes.

In-processing and pre-processing methods are tailored to specific fairness criteria and models, necessitating retraining for each new fairness definition. Post-processing methods, in contrast, are model-agnostic and do not depend on the training process, making them suitable for domain experts with limited MLM knowledge [12]. These methods are especially favored when retraining is infeasible, such as in large-scale systems like recommender systems [13].

Given a potentially biased scoring function, this paper addresses the challenge of constructing a fair probabilistic binary classifier with a binary-protected attribute. The goal is to ensure fairness without retraining the MLM, minimizing performance loss.

Fairness and Performance Trade-offs For classification, one of the desired characteristics of an MLM is *calibration* [14]. Suppose a classifier predicts that a given input is accepted ($Y = 1$) with probability p, then calibration demands that the fraction of the accepted population, with the same features, is p. [14, 15] have shown that calibration and equalized odds cannot be satisfied simultaneously except for highly constrained cases. Hence, researchers have been focusing on building classifiers (MLMs) with an appropriate approximate version of fairness [8]. When it comes to practitioners, they focus on *Receiver Operator Characteristics* (ROC) for evaluating a classifier as it best describes the classifiers. ROC measures the relative scores of the positive versus negative instances. The area under ROC-curve (AUC) is an appropriate performance metric to measure the predictive quality of such classifiers and to segregate positive and negative samples through ranking ([16, 17, 18]). AUC is particularly beneficial when the classifier is expected to segregate positive and negative labels, and the predictions must be fair across all threshold scores.

To make the practitioner's job effortless, we introduce a novel fairness measure, namely ε_p-Equalized ROC – no matter what threshold it uses for classification, the classifier is approximately fair, i.e., for all possible thresholds, the distance between the corresponding points of the ROC curves for both the protected group should be withing ε distance in the $\mathcal{L}_p$ norm. We aim to build a new probabilistic classifier that satisfies ε_1-Equalized ROC with the minimal loss in AUC w.r.t. to the scoring function s.

Our Approach: We assume query access to the ROC of s. First, we make sufficiently large k queries to the ROC for the protected groups and make a piece-wise linear approximation of the ROC curves of both the protected groups. Next, we transport ROCs within ε distance of each other to minimize the loss in AUC of the resultant ROC. We can achieve such transportation by randomizing scores across certain feasible classifiers for the given ROC curve. We call the space of these classifiers as *ROC Space* of s. The resultant classifier from such randomization across the ROC Space is a convex combination of these classifiers. In a nutshell, we *transform* the given s to a fair scoring function by such ROC transport. We refer to this procedure of ROC transport as `FROC`. We then geometrically prove that under certain conditions, `FROC` is *optimal*.

Our Contributions:

- We introduce a novel group fairness notion ε_p-Equalized ROC, enforcing fairness over all thresholds in a score-based classification, which is extremely useful for practitioners.
- Next, we model a post-processing problem as a problem of finding an optimal transformation $\mathcal{H}$ on a given scoring function s to minimize the performance loss due to transformation while ensuring ε_1-Equalized ROC.
- To achieve ε_1-Equalized ROC, we propose a ROC transport, `FROC`, a *post-processing* algorithm (Algorithm 1). Thus, it avoids re-training the existing MLM, which might not be fair. It also helps in explaining the decisions.
- We perform rigorous theoretical analysis. We prove that (under some conditions) `FROC` is optimal in terms of AUC loss. (Theorem 4.2).
- Finally, we demonstrate the efficacy of `FROC` via experiments.

1.1 Related Work

Fairness in Binary Classification and Ranking *Demographic Parity* (DP), *Disparate Impact* (DI), and *Equalized Odds* (EO) are widely studied group fairness notions. DP [7] and DI [9] ensure that the fraction of positive outcomes is identical across all sensitive groups. [19] introduced the 80% rule, requiring that the positive outcome rate for a minority group must be at least $4/5$ of that for the majority group. EO [20] ensures similar distributions of error rates, specifically false positives and false negatives [21]. Techniques to achieve fair MLMs include those discussed by [11]. Group fairness has been shown to be inadequate for score-based classifiers, which classify across all thresholds [22]. Consequently, researchers have proposed fairness notions based on the area under the curve (AUC). Examples include *intra-group pairwise* AUC fairness [23], *BNSP* [24], and *inter-group pairwise* AUC (xAUC) fairness [25]. [26] present a minimax learning and bias mitigation framework that integrates intra-group and inter-group AUC metrics to address algorithmic bias. [27] examine fairness in ranking problems, developing a general class of AUC-based fairness notions. They demonstrate that AUC-based fairness notions do not capture all forms of bias, as AUC summarizes classifier performance. They propose a stronger notion called point wise ROC-based fairness and design an in-processing algorithm for this purpose.

Our fairness definition (ε_p-Equalized ROC) is inspired by equalized odds for all thresholds in ranking-based classification and is suitable for post-processing algorithms. It generalizes the approach of [28], which uses the Manhattan distance as its norm. We later demonstrate the equivalency of both fairness notions (ours ε_1). Note that the notion in [28] is not motivated by the same error rates at all thresholds, and also, ours is more of a geometric approach from ROC curves, and theirs is an algebraic approach; ours is more general.

Post-processing for fair classification Post-processing techniques range from simple adjustments, such as thresholding or re-scaling, to complex methods like re-weighting or re-sampling. [20] argue that many existing fairness criteria are too restrictive, leading to sub-optimal solutions. They propose a fairness notion allowing some variation in prediction outcomes, defined by "equality of opportunity" constraints, ensuring the classifier is unbiased regarding the sensitive attribute. Their approach involves adjusting prediction thresholds for different groups based on their base rates to equalize false positive and false negative rates across groups. However, it does not involve *transporting* ROC curves. [29] examine post-processing from the perspective of transformers, defining fairness as the expectation of scores and bounding the differences between true positive rates (TPRs) and false positive rates (FPRs) across protected groups. [30] propose a model-agnostic post-processing framework for balancing fairness in bipartite ranking scenarios. [31] introduces a novel approach using Wasserstein barycenters to quantify and address the cost of fairness, demonstrating that the complexity of learning an optimal fair predictor is comparable to learning the Bayes predictor. [32] propose a framework that transforms any regularized in-processing method into a post-processing approach, extending its applicability across a broader range of problem settings. [33] identifies two key methodological errors in prior work through empirical analysis: comparing methods with different unconstrained base models and differing levels of constraint relaxation. [34] introduce a method to optimize multiple fairness constraints through group-aware threshold adaptation, learning classification thresholds for each demographic group by optimizing the confusion matrix estimated from the model's probability distribution. Unlike [34], our approach starts with the fairness notion that differences between TPRs and FPRs of different groups must be bounded. [35] use the bounded difference of counterfactual TPRs and FPRs as their fairness criterion, which differs from our ε_p-Equalized ROC definition. Our ε_p-Equalized ROC focuses on the bounded difference between TPRs and FPRs of different groups as the fairness criterion.

2 Preliminaries

Consider a practitioner interested in binary classification, each data point having a binary-protected attribute. He/she is equipped with a scoring-based classifier trained on dataset $D = \{(x_i, a_i, y_i)_{i \in 1:n}\}$. Here, for ith data sample, $x_i \in \mathcal{X} \subset \mathbb{R}^d$ denotes features, $y_i \in \{0, 1\}$ denotes the binary label, and $a_i \in \mathcal{A} = \{0, 1\}$ denotes its binary protected attribute. We consider all these three as drawn from random variables X, A, Y, respectively. There could be two scenarios - when the protected attribute is included or excluded from training ([29])—our post-processing works for both cases as long as protected attributes are accessible during post-processing.

The random variables X, A, Y are jointly distributed according to an unknown probability distribution over (x_i, a_i, y_i). The cumulative conditional distributions on $X \mid (Y = 1)$ and $X \mid (Y = 0)$ are denoted by G, H, respectively. G^a, H^a are the corresponding distributions conditioned on $A = a$ (i.e. G^a denotes the distribution of $X \mid (Y = 1, A = a)$)

2.1 Probabilistic Binary Classification

Probabilistic Binary Classifier is equipped with a scoring function $s : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ mapping the feature space to a score. A deterministic classifier returns $s(X) \in \{0, 1\}$ and a randomized one returns $s(X) \in [0, 1]$. The higher the score $s(x)$, the higher the chance of the corresponding label $y = 1$. The model prediction $\widehat{Y}$, based on certain threshold $t \in [0, 1]$, is given by $\widehat{Y} = \mathbb{I}(s(X) \geq t)$. $\mathcal{S}$ denotes the space of such scoring functions.

The practitioner decides the threshold t depending on the corresponding true positive rate (*TPR*) and false positive rate (*FPR*) ([36, 37]). For deciding t, he is supplied with ROC – *receiver operator characteristic curve* for s. The ROC depicts the relation between TPR ($G_s(t)$) and FPR ($H_s(t)$) for s at all possible thresholds t. Note that, $G_s(t) \triangleq \mathbb{P}(s(X) \geq t \mid Y = 1)$ and $H_s(t) \triangleq \mathbb{P}(s(X) \geq t \mid Y = 0)$.

ROC Curve and AUC

The plot of a ROC-curve (Definition (2.1)) is used to visualize homogeneity between two cumulative distributions [27]. The ROC curve is defined as:

Definition 2.1 (ROC-Curve). *For any two cumulative distributions g_1, g_2 defined over the set $\mathbb{R}$, the ROC-curve is defined as the plot of $ROC_{g_1,g_2}(\alpha) \triangleq 1 - g_1 \circ g_2^{-1}(1-\alpha)$ with domain $\alpha \in [0,1]$.*

The area under ROC-curve, *AUC*, represents a summary of point-wise dissimilarity between the concerned distributions. Formally, let S, S' be two independent random variables distributed according to g_1, g_2 respectively, then $AUC_{g_1,g_2} = \mathbb{P}(S' > S) + \frac{1}{2}\mathbb{P}(S' = S)$.

For a given scoring function s, we get two RVs, G_s and H_s, by varying decision thresholds. We call the corresponding ROC curve $\texttt{ROC}_s$. The area under $\texttt{ROC}_s$, i.e., $\texttt{AUC}_s = AUC_{H_s,G_s}$, is used to measure the ranking performance of a score function $s(.)$ ([38]; [17]). For a perfect classifier, $\texttt{AUC}_s = 1$, but such a classifier does not exist. Therefore, the optimal scoring function s^* maximizes the $\texttt{AUC}_s$ amongst a certain subset of $\mathcal{S}' \subset \mathcal{S}$. Formally, $s^* \in \arg\max_{s \in \mathcal{S}'} \texttt{AUC}_s$. In section 3.4, we illustrate how a sub-optimal score function with lower TPRs can be achieved by randomizing outputs of $s(\cdot)$. This process is crucial in ensuring fairness. Let $\mathcal{S}|_s$ be the space of possible scoring functions through such randomization. We call it ROC-space of s. Before designing our fair classifier, we formally define our notion of fairness in the next section.

2.2 Fairness in Classification

The typical group fairness notions in binary classifiers such as *Demographic Parity* (DP) and *Equalized Odds* (EO) are defined on deterministic predictions, i.e., in score-based classification, they work with a single threshold on scoring function s. Let t^* be the threshold set by the practitioner. The resultant classifier is said to satisfy DP if $G_s^0(t^*) + H_s^0(t^*) = G_s^1(t^*) + H_s^1(t^*)$. It satisfies the equivalence of *acceptance rates* across groups. Similarly, EO enforces equality of positive and negative error rates across protected groups, $1 - G_s^0(t^*) = 1 - G_s^1(t^*)$ and $H_s^0(t^*) = H_s^1(t^*)$.

ε_p-Equalized ROC

As discussed earlier, all group fairness notions are characterized by equality of a particular statistic across both the protected groups. In scoring-based probabilistic classifiers, these fairness notions depend on the selected threshold. To achieve fairness across all thresholds, the practitioner can choose to retrain the model and achieve the right trade-offs between TPR and FNR. However, retraining is expensive. Therefore, a desirable solution is To offer fair treatment to both protected groups using the pre-trained classifier. However, this leads to invoking the post-processing technique every time the practitioner needs to update the threshold t^*. Instead, we propose a novel fairness measure to simplify the practitioner's job. We perform post-processing on the given classifier once, and it ensures that no matter what threshold t^* they choose to make decisions, the classifier offers similar treatment to both the protected groups. That is, the individual ROCs (Here on, we shall denote the ROCs of the protected groups, i.e., $ROC_{H_s^0,G_s^0}$ and $ROC_{H_s^1,G_s^1}$ by $\texttt{ROC}_s^0$ and $\texttt{ROC}_s^1$ respectively) should be within ε distance ($\mathcal{L}_p$ norm) of each other. We call it ε_p-*Equalized ROC*. More formally,

Definition 2.2 (ε_p-Equalized ROC). *A scoring function for binary classification s with label prediction $\widehat{Y} = \mathbb{I}(s(x) \geq t)$ is said to satisfy -Equalized ROC if for all $\alpha \in (0,1)$ the following holds:*

$$|| \; \texttt{ROC}_s^1(\alpha) - \texttt{ROC}_s^0(\alpha) \; ||_p \leq \varepsilon \tag{1}$$

In ε_p-Equalized ROC, we utilize standard metrics (i.e. $\mathcal{L}_p$ norms) as the fairness statistic to quantify fairness. Thus, ε_p-Equalized ROC is feasible for post-processing algorithms. Next, we formulate the problem of fair post-processing. Note: ε_1-Equalized ROC is a generalization of Equalized Odds to all the given thresholds of the scoring function. The proofs and detailed discussion are in Appendix B.

2.3 Problem Formulation

Given $s \in \mathcal{S}$, we would like to find $h \in \mathcal{S}|_s = \mathcal{H}(s)$ – a transformation of a given scoring function such that h satisfies ε_1-Equalized ROC. Additionally, we want the loss in AUC due to transformation $\mathcal{H}$ minimal. That is, $\mathcal{L}_F = \texttt{AUC}_s - \texttt{AUC}_h$ must be minimal to retain the maximum performance guarantee of s. Thus, our goal is to get transformation $\mathcal{H}$ that solves the following optimization problem and returns the optimal transformed score h^*:

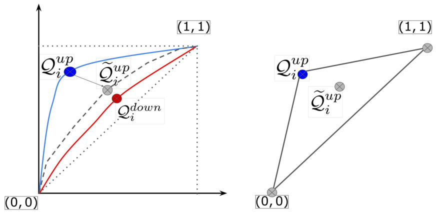

Figure 1: ROCs and convex hull

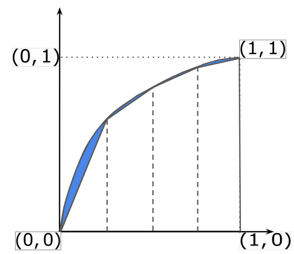

Figure 2: Shaded Area indicates $\mathcal{L}_{PLA}$

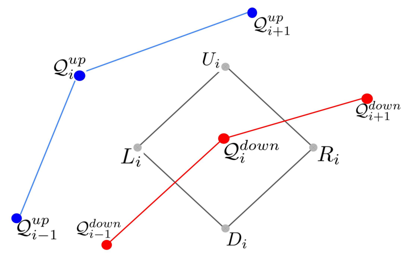

Figure 3: Norm Boundary

$$
\begin{aligned}
h^* &\in \arg\max_{h \in \mathcal{S}|_s} \ \mathtt{AUC}_h \\
\text{s.t.} & \ \ || \ \mathtt{ROC}_h{}^0(\alpha) - \mathtt{ROC}_h{}^1(\alpha) \ ||_1 \leq \varepsilon, \ \forall \alpha \in [0,1]
\end{aligned} \tag{2}
$$

3 Our Approach

First, we explain query access to $\mathtt{ROC}_s$ to sample from the desired statistic at various thresholds and its piece-wise linear approximation in Section 3.1 and Section 3.2, respectively. Since we cannot sample a continuum of thresholds, our $\mathtt{ROC}_s$ will be discrete. In Section 3.3, we describe the transport of ROCs. Finally, we summarize our transformation as $\mathtt{FROC}$ in Section 3.4.

3.1 Query Model

Let $\mathcal{T} = \{t_1, \ldots t_k\}$ be the set of thresholds at which we sample $\mathtt{ROC}_s$ for each sensitive group ($t_i = \frac{i}{k}$). Let $\mathcal{Q}^a(t_i)$ denote the query output at threshold t_i for sensitive group $A = a$ on the $\mathtt{ROC}^a_s$. $\mathcal{Q}^a(t_i) \triangleq ROC_{H^a_s, G^a_s}(t_i)$.

Abusing notations, we use $\mathcal{Q}^a(t_i)$ and $\mathcal{Q}^a_i$ interchangeably. Let $\mathcal{Q}^a = (\mathcal{Q}^a_1, \ldots, \mathcal{Q}^a_k)$ be the sequence of all query outputs for group a. In the next section, we construct the piece-wise linear approximation of the group-wise ROC curves using the group-wise query outputs $\mathcal{Q}^a$.

3.2 Piece-wise Linear Approximation (PLA) of ROC-curves

To obtain the piece-wise linear approximation (PLA), we sample k points from ROC and construct a straight line from $\mathcal{Q}^a_i$ to $\mathcal{Q}^a_{i+1}$ for all $i = 1 \ldots k-1$. Lastly, we join $(0,0)$ to $\mathcal{Q}^a_1$ (see Figure 2). Following these steps on the query sets $\mathcal{Q}^a$ will generate the PLAs for protected groups $a \in \{0,1\}$. We denote by $\widehat{G^a_s}, \widehat{H^a_s}$, the cumulative distributions induced by the linear approximation of the ROC-curve on s.

Due to PLA, we incur a loss $\mathcal{L}_{LPA}$ in AUC_{H_s,G_s}(shaded region in Figure (2)). $\mathcal{L}_{LPA}$ is inversely proportional to the number of queries k, see Section 4.1 for bounds on this loss. Hence, we shall ignore this loss in our fairness analysis as it can be brought arbitrarily close to 0 by increasing k.

3.3 Transporting ROCs for ε_1-Equalized ROC

Since we are using post-processing technique to ensure fairness, it is impossible to shift any $\mathtt{ROC}$ above its current position, i.e., build a classifier corresponding to any point in the epigraph (the points above the ROC curve) of $\mathtt{ROC}_s$ just with the help of s. Interestingly, a classifier representing a point in the hypograph (points below the curve) of $s \cap \mathcal{S}$ can be obtained through randomization on the predicted scores (see Chapter 3 in [39]). The key idea involves abstracting out the convex hull (Fig 1) formed by the three points $(0,0)$, $(1,1)$ and $\mathcal{Q}^{up}_i$, and sampling outcomes from classifiers representing $(0,0)$, $(1,1)$[1] and $\mathcal{Q}^{up}_i$ with specific probabilities. By taking convex combinations of the three aforementioned points in the ROC space, we can represent any point lying in their convex hull. The exact convex combinations are described in **C2**. We leverage this property to achieve ε_1-Equalized ROC. We denote this space as

[1]Note that $(0,0)$ and $(1,1)$ represent 'always reject' and 'always accept' classifiers.

ROC-space of $s - \mathcal{S}|_s$. Each point in $\mathcal{S}|_s$ represents a binary classifier in terms of its performance at a certain threshold t. Each point is of the form $(FPR(t), TPR(t))$.

In the realm of binary classification, it is a common occurrence for one group to be subject to discrimination. Specifically, if we plot $\texttt{ROC}_s^0, \texttt{ROC}_s^1$, we will find that one of the ROCs is notably situated below the other. For this study, the ROC predominantly above the other will be designated as ROC_{up}, while the other ROC will be referred to as ROC_{down}. We believe this is a reasonable assumption because we observed that in most classifiers (for which present the results and others we explored on the datasets mentioned in Section E3) the ROCs don't intersect or intersect at regions where $FPR \leq 0.2$ or $TPR \geq 0.5$. Typically, no practitioner will work in those areas of ROCs. We leave for future work to address intersecting ROCs.

Let $\mathcal{Q}^{up}$, $\mathcal{Q}^{down}$ be the corresponding set of query points for $\texttt{ROC}_{up}, \texttt{ROC}_{down}$ respectively. We also denote their fair counterparts by $\widetilde{\mathcal{Q}}^{up}$, $\widetilde{\mathcal{Q}}^{down}$.

Algorithm Definitions

We need to transport $\texttt{ROC}_{up}$ towards $\texttt{ROC}_{down}$ such that the new ROCs are within ε distance of each other. Our approach is geometric. We need to identify certain points/curves in the epigraph of $\texttt{ROC}_{down}$ as follows.

Definition 3.1 (Norm Boundary). *The set of all points within ε distance (ℓ_1 norm) from $\mathcal{Q}_i^{down}$ is known as the* norm set $\mathfrak{C}_i$. *Formally, we have*

$$\mathfrak{C}_i \triangleq \{x : x \in [0,1]^2, ||x - \mathcal{Q}_i^{down}||_1 \leq \varepsilon\}$$

The set of all points exactly ε distance (in $\mathcal{L}_1$ norm) from $\mathcal{Q}_i^a$ is known as Norm Boundary $\mathfrak{B}_i$. *Formally,*

$$\mathfrak{B}_i \triangleq \{x : x \in [0,1]^2, ||x - \mathcal{Q}_i^{down}||_1 = \varepsilon\}$$

Additionally, we denote the vertices of the Norm Boundary Rhombus (starting from the top most point and moving clockwise) as U_i, R_i, D_i, and L_i.

We say that an index $i \in [1, 2, \ldots, k]$ is a Boundary Cut index when ROC_{up} intersects the Norm Boundary $\mathfrak{B}_i$. Formally,

Definition 3.2 (Boundary Cut). *Index $i \in [1, 2, \ldots, k]$ is a* Boundary Cut index *when $\mathfrak{B}_i \cap ROC_{up} \neq \phi$.*

We now define the three kinds of shifts that will be used in our Algorithm: For a given $i \in [1, 2, \ldots, k]$, Upshift is the transportation of $\mathcal{Q}_i^{up}$ to the point U_i.

Definition 3.3 (UpShift). *For a given $i \in [1, 2, \ldots, k]$, Upshift is the transportation of $\mathcal{Q}_i^{up}$ to the point U_i. Formally,* UpShift *can be defined as the function that returns a fair threshold $\widetilde{\mathcal{Q}}_i^{up}$ (i.e. U_i) by taking the $\mathcal{Q}_i^{down}$ and ε as the arguments.*

For a given $i \in [1, 2, \ldots, k]$, Leftshift is the transportation of $\mathcal{Q}_i^{up}$ to the point L_i. Formally,

Definition 3.4 (LeftShift). *LeftShift is a function that returns a fair threshold $\widetilde{\mathcal{Q}}_i^{up}$ (i.e. L_i) by taking the $\mathcal{Q}_i^{down}$ and ε as the arguments.*

Definition 3.5 (CutShift). *For a given $i \in [1, 2, \ldots, k]$ (representing the index of the ROC_{down}), we run through all the points of the ROC_{up} and return the set of all points that intersect the Norm Boundary $\mathfrak{B}_{\mathrm{i}}$. Formally, we define* Cutshift *as a function that takes $\mathcal{Q}_i^{down}$ and ε as the arguments and returns $ROC_{up} \cap \mathfrak{B}_i$. The set $ROC_{up} \cap \mathfrak{B}_i$ can be represented as $\{p_{left}, p_{right}\}$ denoting the points at the intersection of ROC_{up} at the* ***left-side*** *of the Norm Boundary and the* ***right-side*** *of the Norm Boundary respectively.*

Now, we elaborate on the above procedure to transport points from ROC_{up} towards ROC_{down}.

Algorithm for ROC Transport

We provide a geometric algorithm that returns a classifier equivalent to the scoring function h^* in $\mathcal{S}|_s$.

Note that, Algorithm 1 treats ROC_{down} as *implicitly* fair. Also, by $Area(\square ABCD)$, we denote the area of the quadrilateral whose vertices are A, B, C, and D. This area is easily found in this context by splitting $\square ABCD$ into two disjoint triangles- ΔABC and ΔACD and using the Herons formula [40] on each triangle.

For example, consider $Area(\Delta \mathcal{Q}_i^{up} \mathcal{Q}_{i-1}^{up} L_i)$. Let $a = ||\mathcal{Q}_i^{up} \mathcal{Q}_{i-1}^{up}||_2$, $b = ||\mathcal{Q}_i^{up} L_i||_2$ and $c = ||\mathcal{Q}_{i-1}^{up} L_i||_2$. Additionally, we define $s = \frac{a+b+c}{2}$. Then, it is true that:

$$Area(\Delta \mathcal{Q}_i^{up} \mathcal{Q}_{i-1}^{up} L_i) = \sqrt{s(s-a)(s-b)(s-c)}$$

Algorithm 1: FAIRROC ALGORITHM

Require: ROC_{up}, ROC_{down}, ε
Ensure: $FairROC_{up}$, $FairROC_{down}$
Initialize $i \leftarrow 1$, $k \leftarrow$ length(ROC_{up})
$FairROC_{up} \leftarrow \emptyset$, $FairROC_{down} \leftarrow ROC_{down}$
while $i < k - 1$ **do**
  $i \leftarrow i + 1$
  if BOUNDARYCUT(i, ε) == TRUE **then**
    $p_{left}, p_{right} \leftarrow$ CUTSHIFT$(i, ROC_{up}, ROC_{down})$
    if $FPR(\mathcal{Q}_i^{up}) \geq FPR(\mathcal{Q}_i^{down})$ **then**
      $\widetilde{\mathcal{Q}}_i^{up} \leftarrow p_{right}$
    else
      $\widetilde{\mathcal{Q}}_i^{up} \leftarrow p_{left}$
    end if
  else if $\mathcal{Q}_i^{up} \in$ HYPOGRAPH(ROC_{down}) **then**
    $\widetilde{\mathcal{Q}}_i^{up} \leftarrow \mathcal{Q}_i^{up}$
    continue
  else
    if Area$(\square \mathcal{Q}_{i+1}^{up} \mathcal{Q}_i^{up} \mathcal{Q}_{i-1}^{up} L_i) \geq$ Area$(\square \mathcal{Q}_{i+1}^{up} \mathcal{Q}_i^{up} \mathcal{Q}_{i-1}^{up} U_i)$ **then**
      $\widetilde{\mathcal{Q}}_i^{up} \leftarrow U_i$
    else
      $\widetilde{\mathcal{Q}}_i^{up} \leftarrow L_i$
    end if
  end if
  $FairROC_{up} \leftarrow$ APPEND$(\widetilde{\mathcal{Q}}_i^{up})$
end while
=0

3.4 Obtaining fair classifier from the updated ROCs

The algorithm described in the previous subsection returns the fair ROC curves according to ε_1-Equalized ROC. As a final step, we need to find the transformed classifier. We call it `ConstructClassifier`$(FairROC_{up}$,$FairROC_{down}$,ROC$_s^0$,ROC$_s^1)$ which returns a probabilistic binary classifier representing $h = \mathcal{H}(s)$ such that it represents the FairROCs. We construct one using the procedure explained in Section 3.3.

Now, we establish the optimality of our solution within specific assumptions.

4 Theoretical Analysis

As described in Section (3.2), we work with PLA of the ROC curves $ROC_{H_s^a, G_s^a}$, $a \in \{0, 1\}$. This causes a *loss* in area under ROC. We denote this loss by $\mathcal{L}_{PLA}$ and is quantified as the difference in AUCs of $ROC_{H_s^a, G_s^a}$ and $ROC_{\widehat{H_s^a}, \widehat{G_s^a}}$.

In Section 3.3, transporting the ROC query points, $\mathcal{Q}^{up}$ introduces a decrease of the area under the ROC curve due to the transformation of scoring function s to h. We denote this loss by $\mathcal{L}_{AUC}$. This loss can be quantified as the difference in AUCs of $ROC_{\widehat{H_s^a}, \widehat{G_s^a}}$ and $ROC_{H_h^a, G_h^a}$ The total loss in AUC, $\mathcal{L}$, induced by `FROC` is given by: $\mathcal{L} = \mathcal{L}_{PLA} + \mathcal{L}_{AUC}$

4.1 PLA Loss analysis

We start our analysis by making a few standard assumptions regarding the continuity and differentiability of the cumulative distributions on the family of scoring functions $\mathcal{S}$. We adopt a less stringent assumption than that presented in [27], as we impose only an upper bound on the slopes. This contrasts with the approach in [27], which necessitates both an upper and lower bound on the slopes.

Assumption 4.1. *We assume that the rate of change (with respect to the thresholds t) of the TPRs and FPRs are upper bounded. I.e. we assume that $\exists\, u_T, u_F \in \mathbb{R}$ such that $\frac{d\,TPR}{dt} \leq u_T$ and $\frac{d\,FPR}{dt} \leq u_F$.*

Theorem 4.1. *Let $ROC_{\widehat{H_s^a},\widehat{G_s^a}}$ be the* PLA *of $ROC_{H_s^a,G_s^a}$ over the query set of k equidistant thresholds, $\mathcal{T} = \{t_i \mid t_i = i/k \; \forall i \in [k]\}$. The corresponding $\mathcal{L}_{PLA}$ is bounded as: $\mathcal{L}_{PLA} \leq \frac{1}{2}\frac{u_T u_F}{k}$*

4.2 AUC loss analysis

We start our analysis by making a few assumptions regarding the spacing of the ROC thresholds and the ROC curve.

Assumption 4.2. *We have two assumptions:*

- $\forall i \in \{1, 2, \dots, k\}$*, we assume that* $FPR(\mathcal{Q}_{i-1}^{down}) \leq FPR(\mathcal{Q}_i^{up}) \leq FPR(\mathcal{Q}_{i+1}^{down})$.
- *We assume that the ROC_{up} can intersect any Norm boundary (i.e. $(\mathfrak{B}_i)_{i\in\{1,2,\dots,k\}}$) at most 2 times.*

We note that even if **Assumption 4.2** does not hold, `FROC`remains operational and continues to produce outputs that are -Equalized ROCfair. However, under these conditions, the optimality with respect to AUC is not guaranteed, as **Theorem 4.4** no longer applies.

Theorem 4.2. *If a given classifier s is piece-wise linear and satisfies assumption 4.2, the ROCs returned by* `FROC` *represent the classifier solving optimization problem 2.*

4.3 Optimally fair points and Norm Boundary

This section proves that all optimally fair points must lie on some Norm Boundary. We do this by establishing that the performance of any point in the Norm Set can be improved by appropriate transportation to a point on the Norm Boundary.

Theorem 4.3. *(Norm Boundary) If $(\widetilde{\mathcal{Q}}_i^{up})_{i\in\{1,2,\dots,k\}}$ is the set of optimal fair (points that maximize the AUC and also satisfy the ε_1 -Equalized ROC) thresholds must necessarily be a subset of $(\mathfrak{B}_i)_{i\in\{1,2,\dots,k\}}$.*

Theorem 4.4. *(CutShift) If index i is a Boundary cut point, then the CutShift operation must be performed. Of the 2 points (p_{left} and p_{right}) returned by the Cutshift operation, the point that is closer to $\mathcal{Q}_i^{up}$ must be chosen i.e.$\widetilde{\mathcal{Q}}_i^{up} = argmin_{p\in\{p_{left},p_{right}\}}|FPR(\mathcal{Q}_i^{up}) - FPR(p)|$*

Theorem 4.5. *(UpShift) If index i is not a Boundary cut point and if $Area(\square\mathcal{Q}_{i+1}\mathcal{Q}_i\mathcal{Q}_{i-1}L_i \geq Area(\square\mathcal{Q}_{i+1}\mathcal{Q}_i\mathcal{Q}_{i-1}U_i)$, then UpShift operation must be performed. The resulting point (U_i) is the new fair point $\widetilde{\mathcal{Q}}_i^{up}$. Otherwise, the LeftShift operation must be performed. The resulting point (L_i) is the new fair point $\widetilde{\mathcal{Q}}_i^{up}$.*

The proofs of all the above theorems are given in the appendix. However, the following is brief sketch of the proof:
Step 1: We prove that all optimally fair points $(\widetilde{\mathcal{Q}}_i^{up})_{i\in\{1,2,\dots,k\}}$ must lie on the Norm Boundaries of the corresponding $\mathcal{Q}_i^{down}$. (i.e. $(\mathfrak{B}_i)_{i\in\{1,2,\dots,k\}}$)
Step 2: We then prove that if $\mathfrak{B}_i \cap ROC_{up} \neq \phi$, then the CutShift transportation is the optimal transportation.
Step 3: We then prove that if $\mathfrak{B}_i \cap ROC_{up} = \phi$, then, based on the Cover and aforementioned area condition, the UpShift or the LeftShift transportation is the optimal transportation.

In the next section, we experimentally analyze `FROC`.

5 Empirical Analysis

5.1 Experimental Setup

Datasets: We train different classifiers on the widely-used ADULT [41] and COMPAS [42] benchmark datasets, selecting MALE and FEMALE as protected groups in ADULT, and BLACK and OTHERS in COMPAS. ROCs are generated, with additional experiments on datasets like CelebA in Appendix E and F.

Classifiers: We test `FROC` on ROCs from the following classifiers: [2]. C1: *FNNC*([11]): This is a neural network-based classifier with a target parameter for fairness. C2: *Logistic Regression* and C3: *Random Forest* We used the code from the author's GitHub for C1 and sklearn implementations for C2 and C3.

Post-Processing methods: We compare `FROC` against the following baselines: B1: *FairProjection-CE* and *FairProjection-KL* [43]: Transforms the score to achieve mean equalized odds fairness through information projection.

[2]We choose these classifiers as per the availability of experiment hyper-parameters from other in-processing and post-processing benchmarks.

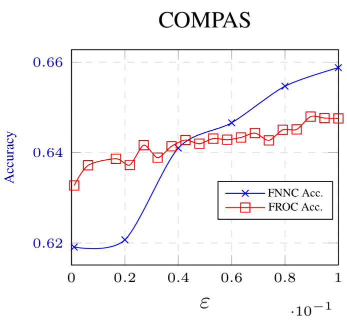

Figure 4: C1 vs. C1-FROC

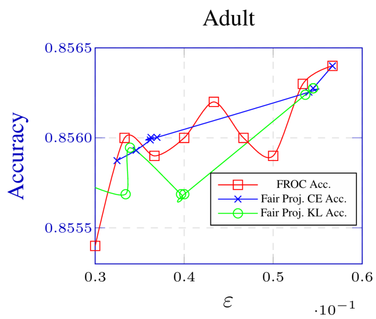

Figure 5: C3-Fair Fair vs. C3-FROC

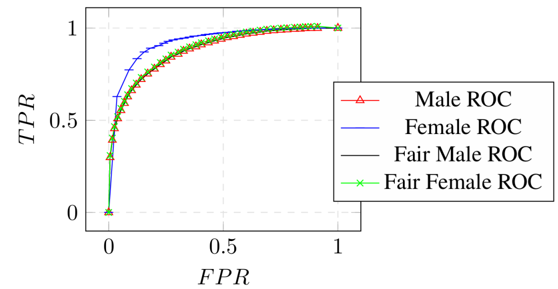

Figure 6: C2 Before and After FROC

5.2 Experiments

We train C1 on both datasets, C2 and C3 on the Adult dataset, and generate their ROCs for all the protected groups. FNNC, we train by ignoring its fairness components in the loss function and then generate ROC. We then invoke FROC for different ε values and check the best possible threshold for accuracy. We refer to the new classifier as C1-C3-FROC.

Baseline Post-Processing Method: We evaluate FROC, and the baselines B1 on ADULT dataset against the fairness metric *mean equalized odds*(B2) [43] in Figs. 5. For consistent comparison, we adopt the training parameters for base classifiers from [43] and keep it identical across all experiments.

5.3 Results

We show the results on the COMPAS and Adult dataset (using FNNC and FROC) here, along with a comparison with existing post-processing baselines. The remaining experimental observations are detailed in the supplementary. **Figure 6** displays the ROC curves (Before and After FROC) for both males and females, on the ADULT dataset for C2. The female ROC consistently occupies the higher position, indicating a positive bias for males. This establishes ROC_0 as our counterpart to ROC_{down}. Thus, we apply FROC to the alternate curve, ROC_1, showcased in the figure. Before FROC, the maximum difference between Male ROC and Female ROC is 0.08. However, after post-processing with FROC, the loss in accuracy is $< 0.1\%$ for $\varepsilon = 0.05$. In general, across all experiments (more experiments in Appendix), we observe a 7-8% improvement in fairness, FROCincurs at most a 2% drop in accuracy. As seen in **Figure 4** and **Figure 5** for smaller values of ε, we also observe the performance may beat FNNC and the post-processing methods. We assign it to the fact that FNNC (and the other methods) may overachieve the target fairness for smaller values of ε (Evident from Table 2 [11]). FROC drops AUC minimally to achieve target fairness.

6 Conclusion

In this work, we addressed the problem of practitioners aiming to achieve fair classification without retraining MLMs. Specifically, we provide a post-processing framework that takes a potentially unfair classification score function and returns a probabilistic fair classifier. The practitioner need not worry about fairness across different thresholds, so we proposed a new notion ε_1-Equalized ROC (Definition 2.2), which ensures fairness for all thresholds. To achieve ε_1-Equalized ROC, we proposed FROC (Algorithm 1), which transports the ROC for each sensitive group within ϵ distance while minimizing the loss in AUC of the resultant ROC. We geometrically proved its optimality conditions (Theorem 4.2) and bounds under certain technical assumptions. We observed empirically that its performance might differ at most by 2% compared to an in-processing technique while ensuring stronger fairness and avoiding retraining. We leave it for future work to explore the possibility of different distance metrics for fairness and optimizing for different performance measures.

Note

The official code for this paper can be found in this link.

Appendix

A Notation Table

Notation	Description
ε	Fairness measure of ε_1-Equalized ROC and FROC
ROC	Receiver Operator Characteristic (plot of FPR vs. TPR)
AUC	Area under ROC curve
s	Scoring function
k	Number of queries submitted to the scoring function
D	Dataset
x_i	Feature vector
$\mathcal{X}$	Sample space of feature vectors
y_i	Binary label
a_i	Binary protected attribute
X	Random vector modeling feature vectors
Y	Random variable modeling labels
A	Random variable modeling protected attributes
$\mathcal{S}$	Space of scoring functions
$\mathcal{S}\|_s$	Space of feasible scoring functions
$G_s(t)$	$\mathbb{P}(s(X) \geq t \| Y = 1)$
$H_s(t)$	$\mathbb{P}(s(X) \geq t \| Y = 0)$
$G_s^a(t)$	$\mathbb{P}(s(X) \geq t \| Y = 1, A = a)$
$H_s^a(t)$	$\mathbb{P}(s(X) \geq t \| Y = 0, A = a)$
$\mathtt{ROC}_s$	ROC_{H_s,G_s}
$\mathtt{AUC}_s$	AUC of $\mathtt{ROC}_s$
$\mathcal{Q}^a$	Sequence of query point from Group a
$\mathcal{Q}_i^a$	Query point of Group a at threshold t_i
$\mathcal{L}_{LPA}$	Loss due to Linear Piecewise Approximation
$\mathfrak{C}_i$	Norm Set of i^{th} threshold
$\mathfrak{B}_i$	Norm Boundary of i^{th} threshold

Table 1: Mathematical Notations

B Relation to Equalized Odds

Equalized Odds is defined in [11] and [8], is the sum of the absolute differences of the FNR and the FPR of both the protected groups. Formally,

$$EO \triangleq |FPR_0 - FPR_1| + |FNR_0 - FNR_1|$$

However, this defintion is equivalent to ε_1-Equalized ROCsince $|FPR_0 - FPR_1| + |FNR_0 - FNR_1| = |FPR_0 - FPR_1| + |(1 - TPR_0) - (1 + TPR_1)| = |FPR_0 - FPR_1| + |TPR_0 - TPR_1|$.

C Algorithm Description

C.1 FROC

Definition C.1 (Norm Boundary). *The set of all points within ε distance (ℓ_1 norm) from $\mathcal{Q}_i^{down}$ is known as the* norm set $\mathfrak{C}_i$. *Formally, we have:*

$$\mathfrak{C}_i \triangleq \{x : x \in [0, 1]^2, ||x - \mathcal{Q}_i^{down}||_1 \leq \varepsilon\}$$

The set of all points exactly ε distance from $\mathcal{Q}_i^a$ is known as Norm Boundary $\mathfrak{B}_i$. *Formally,*

$$\mathfrak{B}_i \triangleq \{x : x \in [0,1]^2, ||x - \mathcal{Q}_i^{down}||_1 = \varepsilon\}$$

Additionally, we denote the vertices of the Norm Boundary Rhombus (starting from the top most point and moving clockwise) as U_i, R_i, D_i, and L_i.

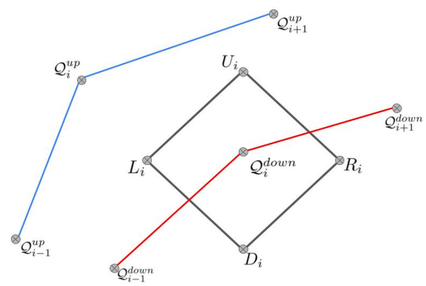

Figure 7: The area inside the rhombus is the Norm set $\mathfrak{C}_i$. The boundary (denoted by the thick bold border) is $\mathfrak{B}_i$. The topmost point of $\mathfrak{B}_i$ is denoted by U_i

We say that a $i \in [1, 2, \ldots, k]$ is a Boundary Cut point when ROC_{up} intersects the Norm Boundary $\mathfrak{B}_i$. Formally,

Definition C.2 (Boundary Cut)**.** *$i \in [1, 2, \ldots, k]$ is a* Boundary Cut point *when $\mathfrak{B}_i \cap ROC_{up} \neq \phi$.*

This is illustrated in the **Figure 8**.

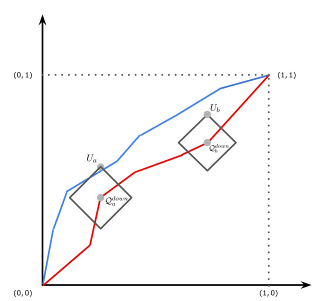

Figure 8: We have two points - $\mathcal{Q}_a^{down}$ and $\mathcal{Q}_b^{down}$ (in increasing order of FPR). We find that $\mathcal{Q}_a^{down}$ is a Boundary Cut point, whereas $\mathcal{Q}_b^{down}$ is not.

We now define the three kinds of shifts that will be used in our Algorithm: For a given $i \in [1, 2, \ldots, k]$, Upshift is the transportation of $\mathcal{Q}_i^{up}$ to the point U_i.

Definition C.3 (UpShift)**.** *For a given $i \in [1, 2, \ldots, k]$, Upshift is the transportation of $\mathcal{Q}_i^{up}$ to the point U_i. Formally,* UpShift *can be defined as the function that returns a fair threshold $\widetilde{\mathcal{Q}}_i^{up}$ (i.e. U_i) by taking the $\mathcal{Q}_i^{down}$ and ε as the arguments.*

This is illustrated in the following **Figure 9**.

For a given $i \in [1, 2, \ldots, k]$, Leftshift is the transportation of $\mathcal{Q}_i^{up}$ to the point L_i. Formally,

Definition C.4 (LeftShift)**.** *LeftShift is a function that returns a fair threshold $\widetilde{\mathcal{Q}}_i^{up}$ (i.e. L_i) by taking the $\mathcal{Q}_i^{down}$ and ε as the arguments.*

This is illustrated in the following **Figure 10**.

Definition C.5 (CutShift)**.** *For a given $i \in [1, 2, \ldots, k]$ (representing the index of the ROC_{down}), we run through all the points of the ROC_{up} and return the set of all points that intersect the Norm Boundary $\mathfrak{B}_i$. Formally, we define* Cutshift *as a function that takes $\mathcal{Q}_i^{down}$ and ε as the arguments and returns $ROC_{up} \cap \mathfrak{B}_i$. The set $ROC_{up} \cap \mathfrak{B}_i$ can be*

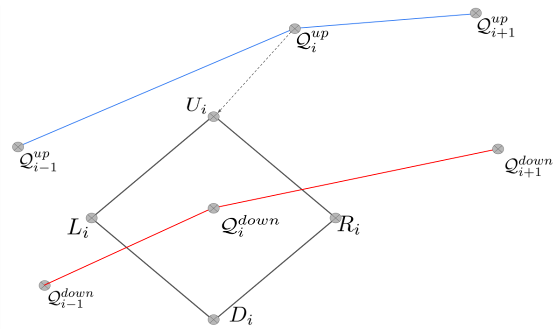

Figure 9: UpShift

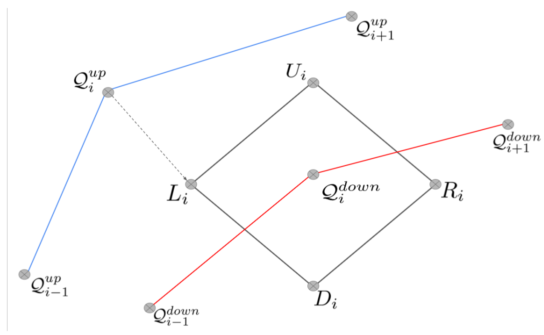

Figure 10: LeftShift

represented as $\{p_{left}, p_{right}\}$ *denoting the points at the intersection of* ROC_{up} *at the* ***left-side*** *of the Norm Boundary and the* ***right-side*** *of the Norm Boundary respectively.*

This is illustrated in the following **Figure 11**.

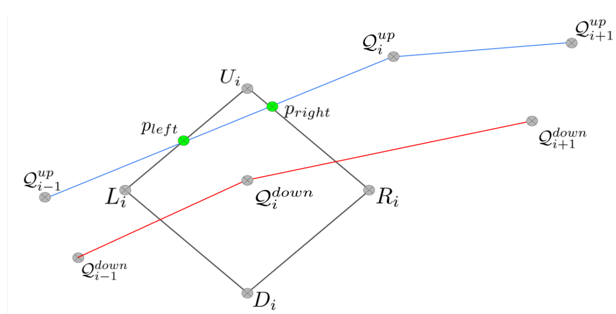

Figure 11: CutShift

Note that the two intersection points - p_{left} and p_{right} will be to the right of $\mathcal{Q}_i^{up}$ when $FPR(\mathcal{Q}_i^{up}) \leq FPR(\mathcal{Q}_i^{down})$. Note that it is also possible for p_{left} to lie on the line segment $\overline{L_i D_i}$ instead of line segment $\overline{U_i L_i}$ when $\mathcal{Q}_i^{up}$ has sufficiently low TPR.

We elaborate on the above procedure to transport points from ROC_{up} towards ROC_{down} in the following subsection.

C.2 Randomization to obtain new classifiers

Theorem C.1. *If $\mathcal{Q}_a, \mathcal{Q}_b, \mathcal{Q}_c$ are points in $\mathcal{S}|_s$ forming a convex hull Δ and $\mathcal{Q} \in \Delta$, then the classifier equivalent to $\mathcal{Q}$ can be obtained by following the below procedure. For each test data point x, use the following randomization scheme:*

$$Classifier_{\mathcal{Q}}(x) = \begin{cases} Classifier_{\mathcal{Q}_a}(x) & \text{w.p. } p_a \\ Classifier_{\mathcal{Q}_b}(x) & \text{w.p. } p_b \\ Classifier_{\mathcal{Q}_c}(x) & \text{w.p. } 1 - p_a - p_b \end{cases} \tag{3}$$

Here, we have, $p_a = \frac{c_1 b_2 - c_2 b_1}{a_1 b_2 - a_2 b_1}$, $p_b = \frac{c_1 a_2 - c_2 a_1}{a_1 b_2 - a_2 b_1}$ and

$$a_1 = TPR(\mathcal{Q}_a) - TPR(\mathcal{Q}_c) \text{ and } a_2 = FPR(\mathcal{Q}_a) - FPR(\mathcal{Q}_c)$$
$$b_1 = TPR(\mathcal{Q}_b) - TPR(\mathcal{Q}_c) \text{ and } b_2 = FPR(\mathcal{Q}_b) - FPR(\mathcal{Q}_c)$$
$$c_1 = TPR(\mathcal{Q}) - TPR(\mathcal{Q}_c) \text{ and } c_2 = FPR(\mathcal{Q}) - FPR(\mathcal{Q}_c)$$

D Theoretical Results

D.1 Piecewise Linear Approximation

Theorem D.1. *Let $ROC_{\widehat{H^a_s},\widehat{G^a_s}}$ be the* PLA *of $ROC_{H^a_s,G^a_s}$ over the query set of k equidistant thresholds, $\mathcal{T} = \{t_i \mid t_i = i/k \; \forall i \in [k]\}$ then the corresponding $\mathcal{L}_{PLA}$ is bounded as:*

$$\mathcal{L}_{PLA} \leq \frac{1}{2}\frac{u_T u_F}{k^2} \times k = \frac{1}{2}\frac{u_T u_F}{k}$$

Proof. In **Figure 12**, shaded area is the approximation loss $\mathcal{L}_{PLA}$. Let us consider the situation where $ROC_{H^a_s,G^a_s}$

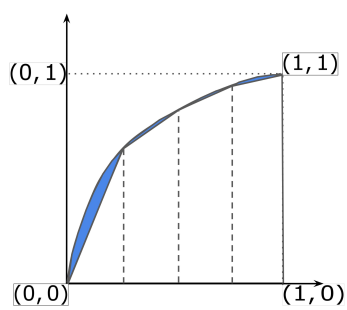

Figure 12: $\mathcal{L}_{PLA}$

maximally deviates from its PLA $ROC_{\widehat{H^a_s},\widehat{G^a_s}}$. To find an upper bound to this area, we must stretch it till the dotted line

The area cannot go beyond the dotted line (**Figure 14**)because ROCs are one-to-one and monotonically increasing functions. So, our goal now, is to bound the sum of areas of the blue shaded triangles. We have the base of each triangle to be $\frac{1}{k} \times u_F$ (since k thresholds and maximum slope of FPR with respect to the thresholds is u_F). We have the maximum possible height of each triangle $\frac{1}{k} \times u_T$ (since k thresholds and maximum slope of TPR with respect to the thresholds is u_T). This makes the maximum possible area of each triangle $\frac{u_T u_F}{2k^2}$. So, for an interval between thresholds t_i, t_{i+1}, the loss incurred is $\leq \frac{1}{2}\frac{u_T u_F}{k^2}$. To extend this for the entire ROC over k intervals, we have:

$$\mathcal{L}_{PLA} \leq \frac{1}{2}\frac{u_T u_F}{k^2} \times k = \frac{1}{2}\frac{u_T u_F}{k}$$

$\square$

Therefore, we can infer:

$$\lim_{k\to\infty} \mathcal{L}_{LPA} = \lim_{k\to\infty} \frac{1}{2}\frac{u_T u_F}{k} = 0$$

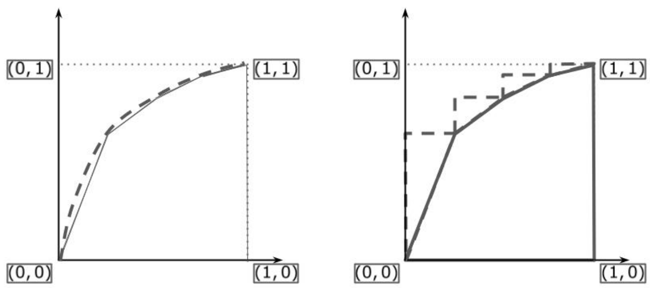

Figure 13: Maximally streching the ROC (Dotted line)

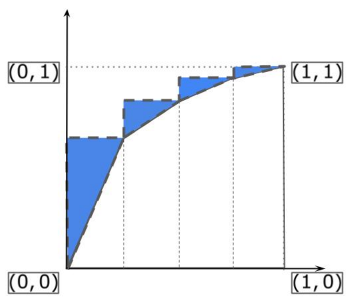

Figure 14: The area shaded by the darker shade of blue is the maximum possible loss of AUC due to Linear Interpolation.

D.2 Boundary Optimality

All optimal points lie on the Norm boundary

Theorem D.2. *(Norm Boundary) If* $(\widetilde{\mathcal{Q}}_i^{up})_{i\in\{1,2,\ldots,k\}}$ *is the set of optimal fair (points that maximize the AUC and also satisfy the* ε *fairness criteria) thresholds must necessarily be a subset of* $(\mathfrak{B}_i)_{i\in\{1,2,\ldots,k\}}$.

Proof. (Proof by Contradiction) Let us assume that some point C in the interior of the Norm Set is the optimal fair (point that leads to ROC with maximum possible AUC while satisfying ε_1-Equalized ROC) point. As we can see in **Figure 15**, we have transported $\mathcal{Q}_i^{up}$ to C in the interior of the Norm set. The shaded area denotes the AUC loss due to this transformation. However, as seen in the next figure **Figure 16**, the AUC loss can be decreased by choosing a point (we choose the CutShift point) on the Norm boundary. Thus, we can always decrease AUC loss by choosing a point on the Norm Boundary. Formally, if point C was the optimal fair point, then the AUC loss with respect to that point is $Area(\square\mathcal{Q}_{i-1}^{up}C\mathcal{Q}_{i+1}^{up}\mathcal{Q}_i^{up})$.
However, if A is the optimal fair point (Fig 16), then the AUC loss with respect to that point is $Area(\square\mathcal{Q}_i^{up}A\mathcal{Q}_{i+1}^{up})$. However, we notice that: $Area(\square\mathcal{Q}_{i-1}^{up}C\mathcal{Q}_{i+1}^{up}\mathcal{Q}_i^{up}) = Area(\square\mathcal{Q}_i^{up}A\mathcal{Q}_{i+1}^{up}) + Area(\square\mathcal{Q}_{i-1}^{up}C\mathcal{Q}_{i+1}^{up}A)$. Since $Area(\square\mathcal{Q}_{i-1}^{up}C\mathcal{Q}_{i+1}^{up}A) \geq 0$, we have:

$$Area(\square\mathcal{Q}_{i-1}^{up}C\mathcal{Q}_{i+1}^{up}\mathcal{Q}_i^{up}) \geq Area(\square\mathcal{Q}_i^{up}A\mathcal{Q}_{i+1}^{up})$$

This is a contradiction to the assumption that C is the optimal fair point. Therefore, C is not an optimal fair point. $\square$

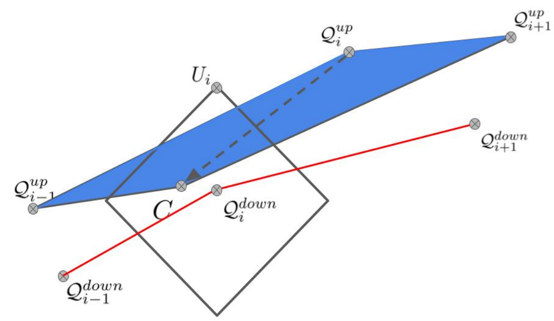

Figure 15: The blue colored region indicates the AUC loss.

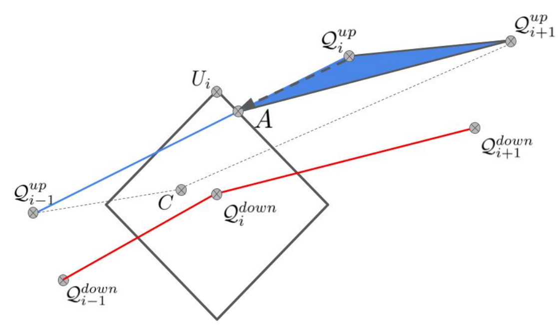

Figure 16: The dark blue colored region indicates the new AUC loss. The light blue region indicates the previous AUC loss.

D.3 CutShift Optimality

Theorem D.3. *If i is a Boundary cut point, then the CutShift operation must be performed. Of the 2 points (p_{left} and p_{right}) returned by the Cutshift operation, the point that is closer to $\mathcal{Q}_i^{up}$ must be chosen i.e.*

$\widetilde{\mathcal{Q}}_i^{up} = argmin_{p \in \{p_{left}, p_{right}\}} |FPR(\mathcal{Q}_i^{up}) - FPR(p)|$

Proof. (Proof by Contradiction) Let us assume that some point C on the Norm Boundary is the optimal fair (point that leads to ROC with maximum possible AUC while satisfying ε_1-Equalized ROC) point. As we can see in **Figure 17**, we have transported $\mathcal{Q}_i^{up}$ to C in the interior of the Norm set. The shaded area denotes the AUC loss due to this transformation. However, as seen in the next figure Fig 16, the AUC loss can be decreased by choosing a point (we choose the CutShift point) on the Norm boundary. Thus, we can always decrease AUC loss by choosing a point on the Norm Boundary. Formally, if point C was the optimal fair point, then the AUC loss with respect to that point is $Area(\square \mathcal{Q}_{i-1}^{up} C \mathcal{Q}_{i+1}^{up} \mathcal{Q}_i^{up})$.
However, if A is the optimal fair point (Fig 18), then the AUC loss with respect to that point is $Area(\square \mathcal{Q}_i^{up} A \mathcal{Q}_{i+1}^{up})$. However, we notice that:$Area(\square \mathcal{Q}_{i-1}^{up} C \mathcal{Q}_{i+1}^{up} \mathcal{Q}_i^{up}) = Area(\square \mathcal{Q}_i^{up} A \mathcal{Q}_{i+1}^{up}) + Area(\square \mathcal{Q}_{i-1}^{up} C \mathcal{Q}_{i+1}^{up} A)$. Since $Area(\square \mathcal{Q}_{i-1}^{up} C \mathcal{Q}_{i+1}^{up} A) \geq 0$, we have:

$$Area(\square \mathcal{Q}_{i-1}^{up} C \mathcal{Q}_{i+1}^{up} \mathcal{Q}_i^{up}) \geq Area(\square \mathcal{Q}_i^{up} A \mathcal{Q}_{i+1}^{up})$$

This is a contradiction to the assumption that C is the optimal fair point. Therefore, C is not an optimal fair point.

□

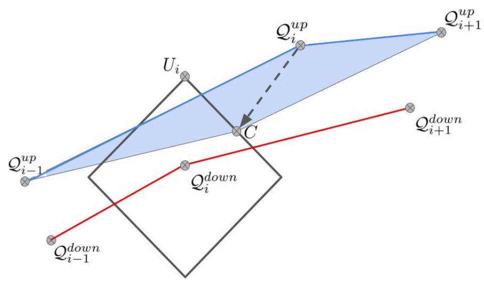

Figure 17: CutShift Operation is not followed. The light blue area indicates the AUC loss due to this operation.

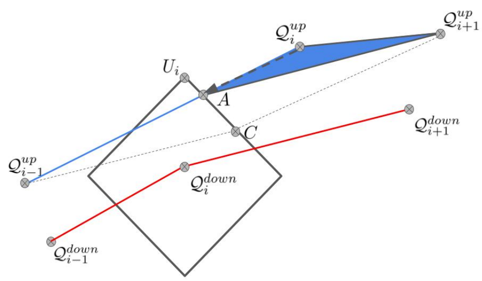

Figure 18: CutShift Operation is followed. The dark blue area indicates the AUC loss due to this operation. It is lesser than the previous AUC loss as seen in Figure 9.

D.4 Upshift and Left Shift

Theorem D.4 (UpShift)**.** *If i is not a Boundary cut point and if $Area(\square \mathcal{Q}_{i+1}\mathcal{Q}_i\mathcal{Q}_{i-1}L_i \geq Area(\square \mathcal{Q}_{i+1}\mathcal{Q}_i\mathcal{Q}_{i-1}U_i)$, then UpShift operation must be performed. The resulting point (U_i) is the new fair point $\widetilde{\mathcal{Q}}_i^{up}$. Else, LeftShift operation must be performed. The resulting point (L_i) is the new fair point $\widetilde{\mathcal{Q}}_i^{up}$.*

Proof. By a similar argument, as the previous proofs, we argue (through **Figure 19**, **Figure 21** and **Figure 22**), we can prove that either the point recommended by UpShift (U_i) or LeftShift (L_i) is the optimal point. So, to decide between them, we use Heron's formula to find the area of both quadrilaterals and then compare their areas to find the least AUC loss. We can use Heron's formula to find the area of a quadrilateral in the following way: If $\square ABCD$ is a quadrilateral with vertices A, B, C and D. This area is easily found in this context by splitting $\square ABCD$ into two disjoint triangles- ΔABC and ΔACD and using the Herons formula [40] on each triangle. For example, consider $Area(\Delta \mathcal{Q}_i^{up}\mathcal{Q}_{i-1}^{up}L_i)$. Let $a = ||\mathcal{Q}_i^{up}\mathcal{Q}_{i-1}^{up}||_2$, $b = ||\mathcal{Q}_i^{up}L_i||_2$ and $c = ||\mathcal{Q}_{i-1}^{up}L_i||_2$. Additionally, we define $s = \frac{a+b+c}{2}$. Then, it is true that:

$$Area(\Delta \mathcal{Q}_i^{up}\mathcal{Q}_{i-1}^{up}L_i) = \sqrt{s(s-a)(s-b)(s-c)}$$

□

The optimality of AUC (Theorem 4.2) follows from Theorem D.2, Theorem D.3 and Theorem D.4.

D.5 Sample Complexity

If the **Assumption 4.2** holds true, then we have the following analysis:

- All UpShift Operations will be constant time ($O(1)$).
- All CutShift Operations will also be constant time ($O(1)$). This is because **Assumption 4.2** ensures that we do not have to run through the entire length of ROC_{up} to find the intersection points i.e. p_{left} and p_{right}.

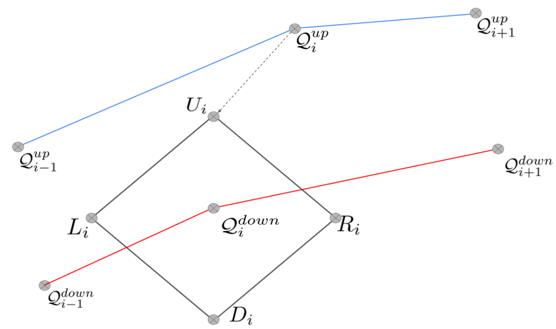

Figure 19: The dotted arrow represents the UpShift transportation of the point from $\mathcal{Q}_i^{up}$ to U_i

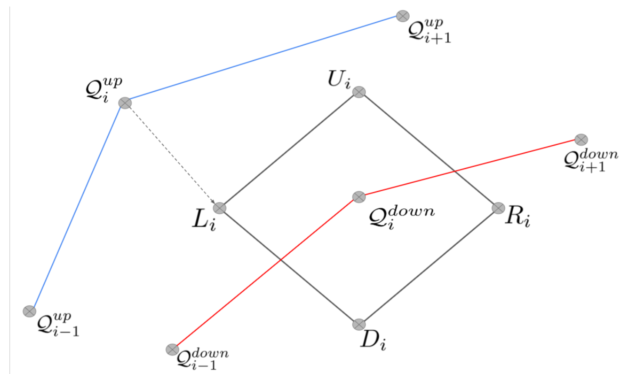

Figure 20: The dotted arrow represents the LeftShift transportation of the point from $\mathcal{Q}_i^{up}$ to U_i

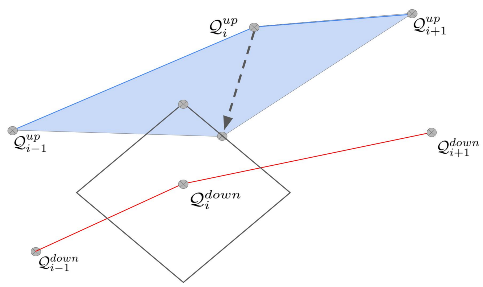

Figure 21: UpShift Operation is not followed. The light blue area indicates the AUC loss due to this operation.

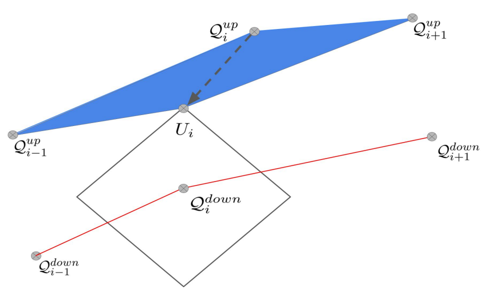

Figure 22: UpShift Operation is followed. The dark blue area indicates the AUC loss due to this operation. It is lesser than the previous AUC loss as seen in Figure 11.

Therefore, the running time of `FROC` is $O(k)$. However, when no assumptions are made, then the CutShift operation is no longer $O(1)$. We may have to run through the entire length of ROC_{up} to find the intersection points i.e. p_{left} and p_{right}. This makes the CutShift operation $O(k)$. Therefore, the time complexity of `FROC` is $O(k^2)$.

D.6 Further Variants

Multiple Protected Groups

Our approach is extendable to scenarios involving multiple protected groups. The procedure begins by applying the FROC algorithm to the ROC curve that is immediately above the bottom-most ROC curve. Subsequently, FROC is applied to the ROC curve directly above the one previously processed. This iterative application continues until the top ROC curve is reached. While this method ensures -Equalized ROCfairness across all protected groups, the proof of optimality remains an open question.

Intersection of ROC Curves

In cases where the ROC curves intersect more than twice, our algorithm will still produce a fair output. However, the existing optimality theorems do not apply in such scenarios. When intersections occur, the FROC algorithm can be applied to the dominant segments of the ROC curves—those portions where no intersections are present.

E Experiments

E.1 Datasets

UCI Adult Dataset

The Adult Dataset [41] comprises 48,842 instances, each containing 14 attributes, including both categorical and continuous variables. The dataset was designed to predict whether an individual's income exceeds $50,000 per year, making it suitable for binary classification tasks. The features include demographic information such as age, education level, marital status, occupation, work hours per week, and native country, among others.

COMPAS Recidivism Dataset

COMPAS Dataset [42] is a widely-discussed and controversial dataset utilized in the field of criminal justice and fairness-aware machine learning. The COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) dataset is commonly employed to explore the potential bias and fairness issues that may arise in predictive models used for criminal justice decisions. The COMPAS dataset consists of historical data on defendants who were considered for pretrial release in a U.S. county. The data includes various features extracted from defendant profiles, such as age, race, gender, past criminal history, pending charges, and other pertinent factors. Additionally, the dataset contains binary labels indicating whether a defendant was rearrested within a specific period after their release.

E.2 Protected Groups

In the context of this paper, we consider the relative performance of the classifiers with respect to the different protected groups - sex (Male and Female) for the Adult Dataset and Race (African Americans and Others) for the COMPAS Dataset.

E.3 Experiment Details

We have performed statistical analysis on FROC, but not on the original classifier. This is because studying the fairness-accuracy trade-of is our goal (as opposed to studying the performance of the baseline classifier). However, it must be noted that since the ROC shifting is deterministic, all randomness emerges from the post-shift classifier builder. For the statistical analysis, we have 10 iterations of the experiment as ε runs from 0.001 to 0.1 in 20 intervals. (Except for the case of Random Forest Gini (Adult) : 0.001 to 0.2 in 20 intervals.)

E.4 Plots

Adult Dataset - Weighted ensemble L2

- We have applied FROC with the our fairness parameter $\varepsilon = 0.01$ in **Figure 24**. As promised, the resulting ROCs are 'closer' to each other.
- In **Figure 25** and **Figure 26**, we have the Accuracy vs. ε_1 and the Disparate Impact vs. ε_1 plot.

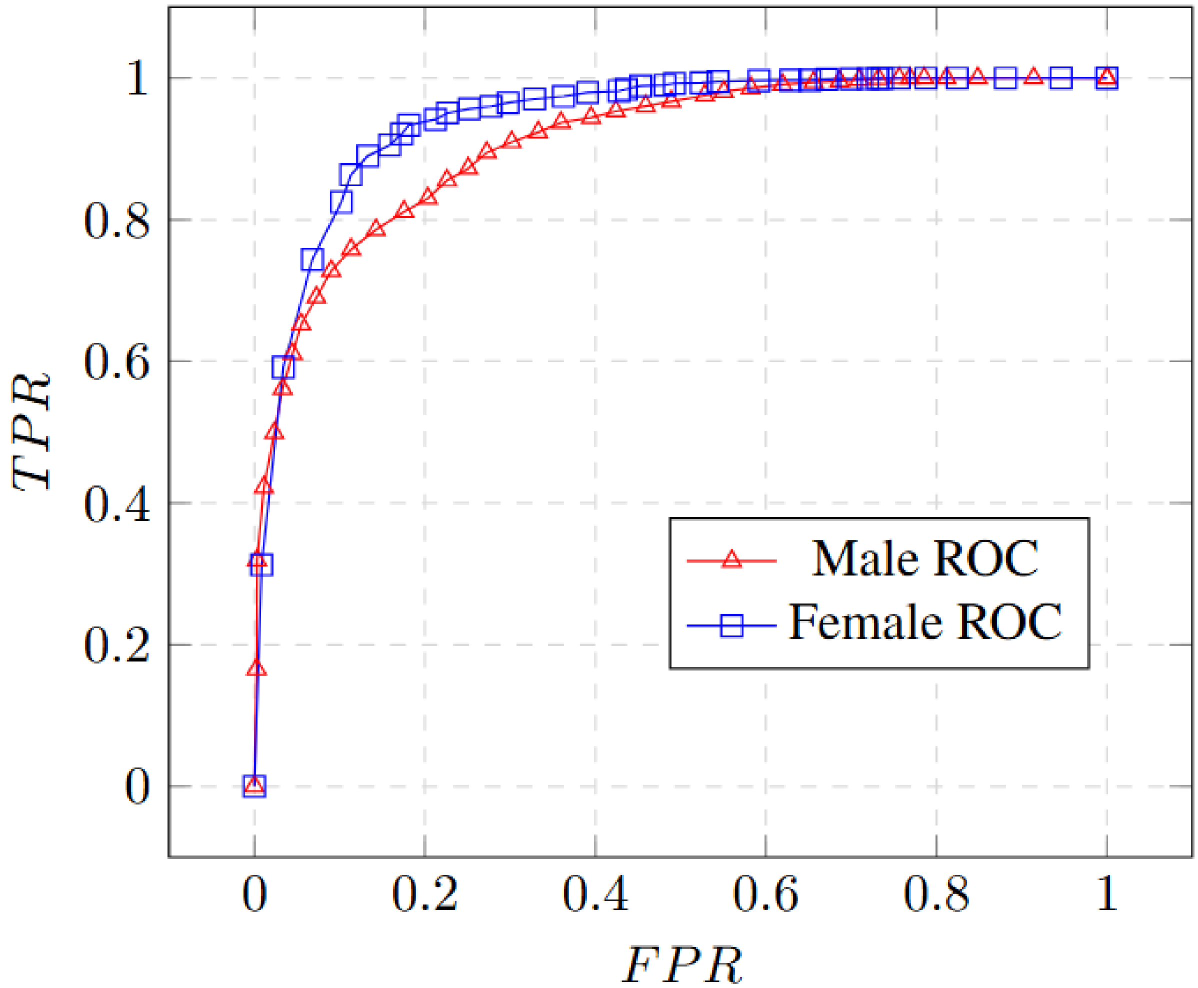

Figure 23: Weighted Ensemble L2 Baseline ROCs for Adult Dataset

- This analysis gives us a maximum variance of 1.88×10^{-6} and a maximum CoV (Coefficient of Variation) of 0.15% for Accuracy.
- As for the Disparate Impact, the analysis gives us a maximum variance of 2.25×10^{-5} and a maximum CoV of 0.55%.
- As seen in the plots, we observe that a 1% drop in Accuracy improves the Disparate Impact by 5%.
- Finally, in **Figure 27**, we have the AUC loss vs. ε plot. As seen in the figure, the AUC loss decays to 0 as our fairness constraint loosens.

Adult Dataset - Random Forest Gini

- We have applied FROC with the our fairness parameter $\varepsilon = 0.01$ in **Figure 29**. As promised, the resulting ROCs are 'closer' to each other.
- In **Figure 30** and **Figure 31**, we have the Accuracy vs. ε_1 and the Disparate Impact vs. ε_1 plot.
- This analysis gives us a maximum variance of 8.3×10^{-7} and a maximum CoV (Coefficient of Variation) of 0.1% for Accuracy.
- As for the Disparate Impact, the analysis gives us a maximum variance of 7.59×10^{-6} and a maximum CoV of 0.75%.

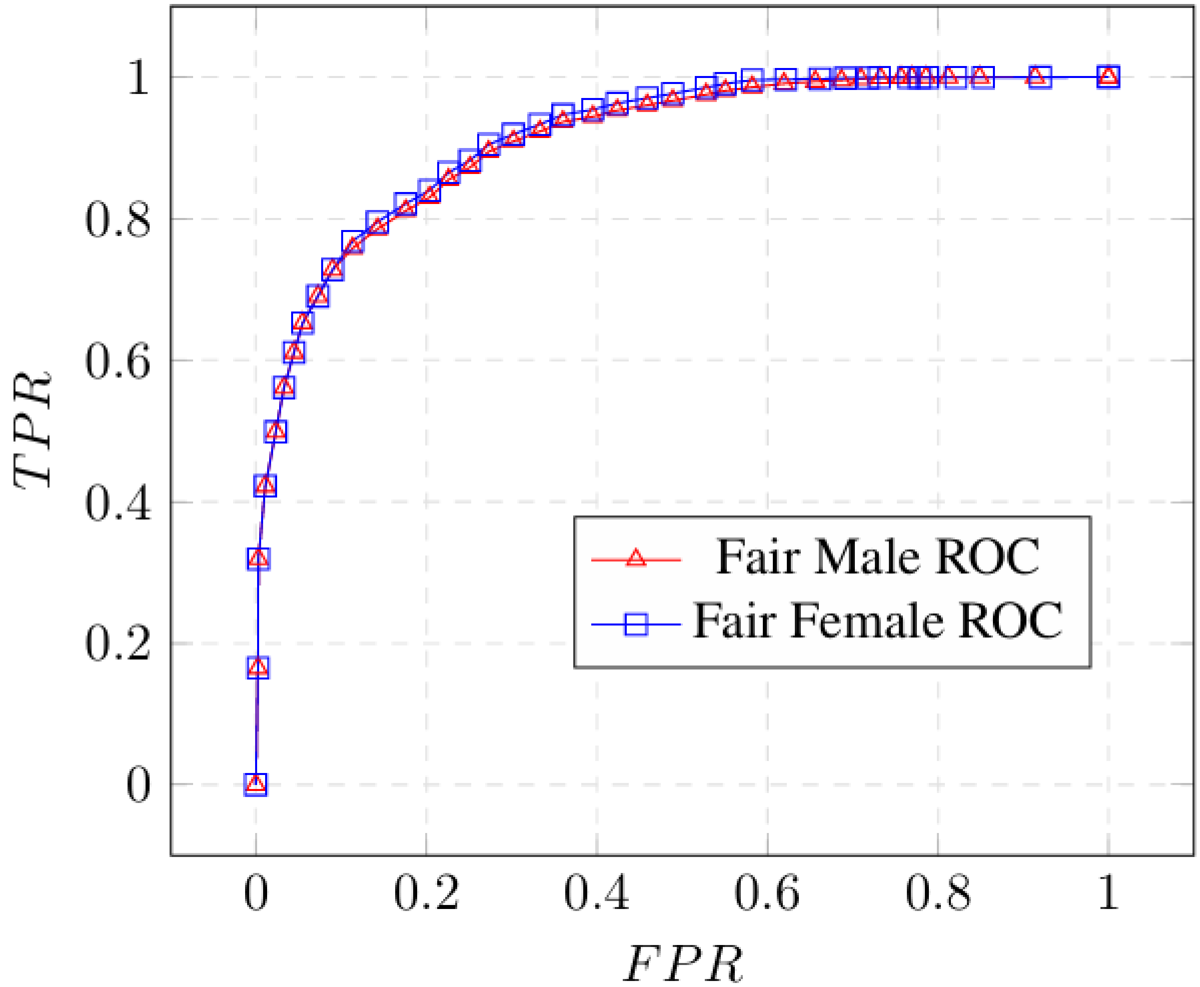

Figure 24: (Fair $\varepsilon_1 = 0.01$) Weighted Ensemble L2-FROC ROCs for Adult Dataset

- As seen in the plots, we observe that a 1% drop in Accuracy improves the Disparate Impact by 7%.
- Finally, in **Figure 32**, we have the AUC loss vs. ε_1 plot. As seen in the figure, the AUC loss decays to 0 as our fairness constraint loosens.

Adult Dataset - FNNC

- We have applied FROC with the our fairness parameter $\varepsilon_1 = 0.01$ in **Figure 43**. As promised, the resulting ROCs are 'closer' to each other.
- In **Figure 35**, we have the Accuracy vs. ε_1 and the Disparate Impact vs. ε_1 plot. We also have the $\varepsilon_{FNNC} vs. \varepsilon_{FROC}$ plot.
- We find that in the FNNC is slightly lower than FROC in terms of accuracy. We assign it to the fact that FNNC may overachieve the target fairness for smaller values of ε_1 (Evident from Table 2 [Padala and Gujar 2021]). FROC drops AUC minimally to achieve target fairness.
- This analysis gives us a maximum variance of 6.6×10^{-7} and a maximum CoV (Coefficient of Variation) of 0.09% for Accuracy.
- As for the Disparate Impact, the analysis gives us a maximum variance of 1×10^{-4} and a maximum CoV of 1.26%.
- As seen in the plots, we observe that a 1% drop in Accuracy improves the Disparate Impact by 5%.

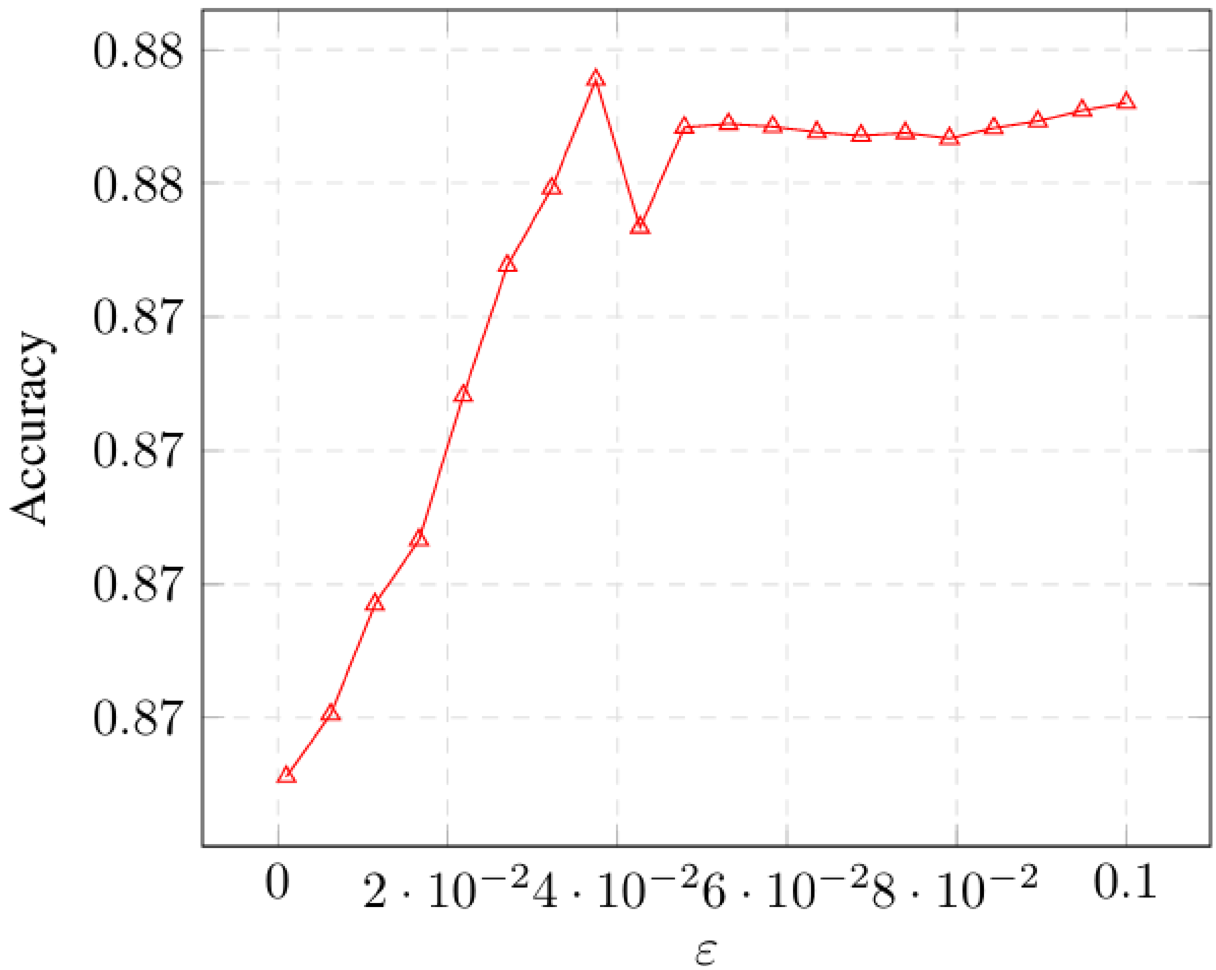

Figure 25: Weighted Ensemble L2-FROC Accuracy vs. ε_1 (Adult)

- Finally, in **Figure 36**, we have the AUC loss vs. ε_1 plot. As seen in the figure, the AUC loss decays to 0 as our fairness constraint loosens.

COMPAS Dataset - Weighted ensemble L2

- We have applied FROC with the our fairness parameter $\varepsilon_1 = 0.01$ in **Figure 38**. As promised, the resulting ROCs are 'closer' to each other.
- In **Figure 39** and **Figure 40**, we have the Accuracy vs. ε_1 and the Disparate Impact vs. ε_1 plot.
- This analysis gives us a maximum variance of 1.44×10^{-5} and a maximum CoV (Coefficient of Variation) of 0.54% for Accuracy.
- As for the Disparate Impact, the analysis gives us a maximum variance of 1.6×10^{-4} and a maximum CoV of 1.69%.
- As seen in the plots, we observe that a 1% drop in Accuracy improves the Disparate Impact by 7%.
- Finally, in **Figure 41**, we have the AUC loss vs. ε_1 plot. As seen in the figure, the AUC loss decays to 0 as our fairness constraint loosens.

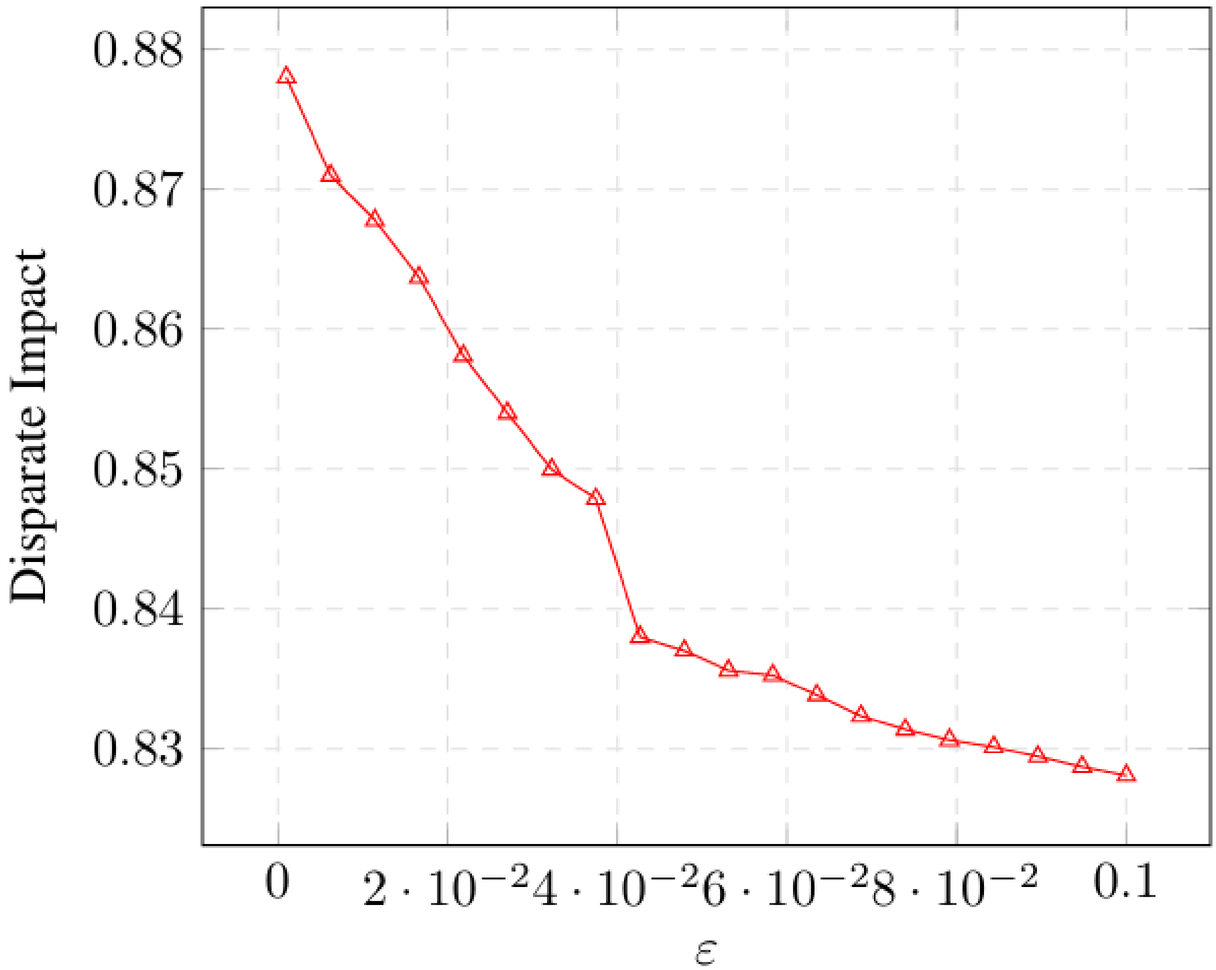

Figure 26: Weighted Ensemble L2-FROC Disparate Impact vs. ε_1 (Adult)

COMPAS Dataset - Random Forest Gini

- We have applied FROC with the our fairness parameter $\varepsilon = 0.01$ in **Figure 43**. As promised, the resulting ROCs are 'closer' to each other.
- In **Figure 44** and **Figure 45**, we have the Accuracy vs. ε_1 and the Disparate Impact vs. ε_1 plot.
- This analysis gives us a maximum variance of 9.63×10^{-6} and a maximum CoV (Coefficient of Variation) of 0.44% for Accuracy.
- As for the Disparate Impact, the analysis gives us a maximum variance of 2×10^{-4} and a maximum CoV of 1.56%.
- As seen in the plots, we observe that a 1% drop in Accuracy improves the Disparate Impact by 7%.
- Finally, in **Figure 46**, we have the AUC loss vs. ε_1 plot. As seen in the figure, the AUC loss decays to 0 as our fairness constraint loosens.

COMPAS Dataset - FNNC

- We have applied FROC with the our fairness parameter $\varepsilon_1 = 0.01$ in **Figure 48**. As promised, the resulting ROCs are 'closer' to each other.
- In **Figure 49**, we have the Accuracy vs. ε_1 and the Disparate Impact vs. ε_1 plot.

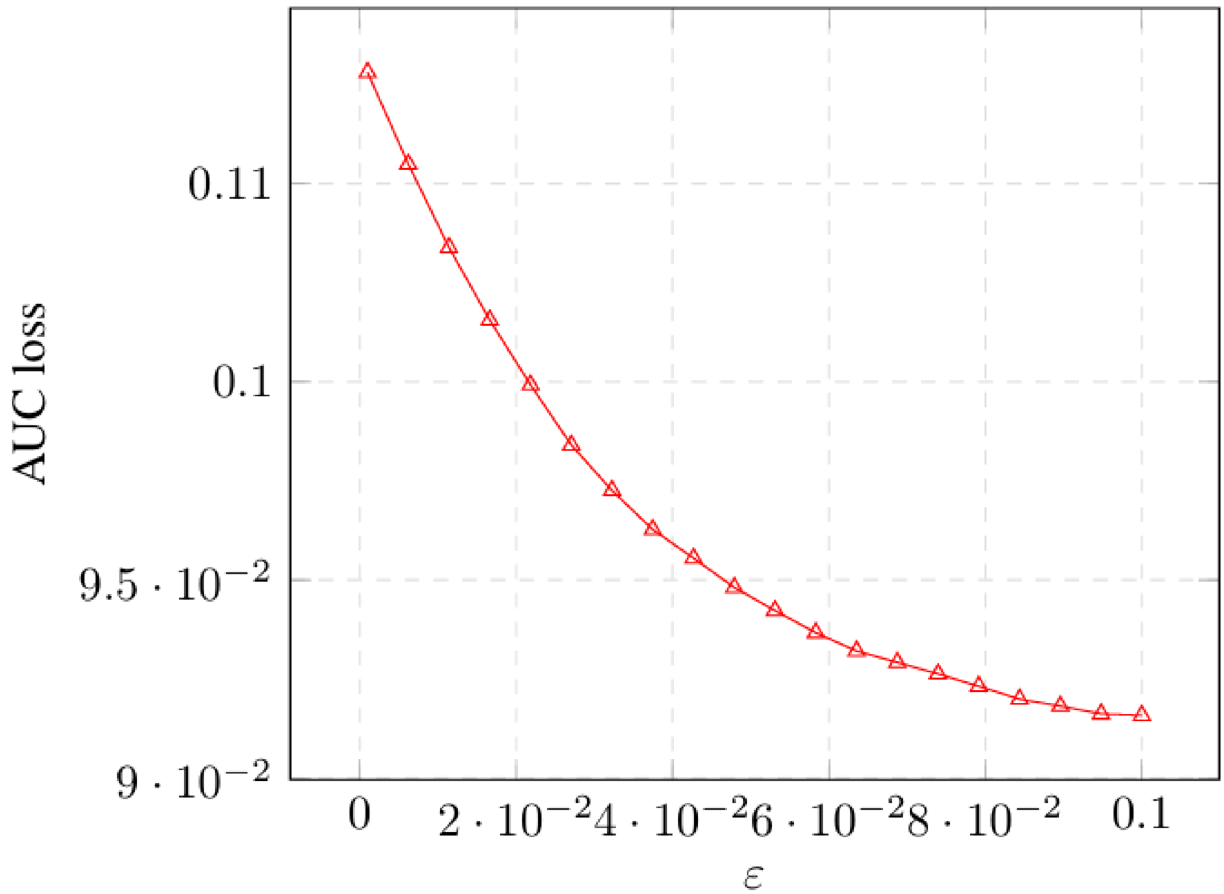

Figure 27: Weighted Ensemble L2-FROC AUC loss vs. ε_1 (Adult)

- We find that in the FNNC is slightly lower than FROC in terms of accuracy. We assign it to the fact that FNNC may overachieve the target fairness for smaller values of ε_{FNNC}, (Evident from Table 2 [[11]]). FROC drops AUC minimally to achieve target fairness.
- This analysis gives us a maximum variance of 4.83×10^{-6} and a maximum CoV (Coefficient of Variation) of 0.43% for Accuracy.
- As for the Disparate Impact, the analysis gives us a maximum variance of 2.48×10^{-5} and a maximum CoV of 0.5%.
- As seen in the plots, we observe that a 1% drop in Accuracy improves the Disparate Impact by 3%.
- Finally, in **Figure 50**, we have the AUC loss vs. ε_1 plot. As seen in the figure, the AUC loss decays to 0 as our fairness constraint loosens.

CelebA Dataset

- We have applied FROC with the our fairness parameter $\varepsilon_1 = 0.01$ in **Figure 52**. As promised, the resulting ROCs are 'closer' to each other.
- This analysis gives us a maximum variance of 1.9×10^{-7} and a maximum CoV (Coefficient of Variation) of 0.07% for Accuracy (**Figure 53**).
- As for the Disparate Impact, since both the ROCs are very close to begin with, we find that there is not much improvement in terms of performance.
- The AUC is also similar in nature - it shows no clear trend.

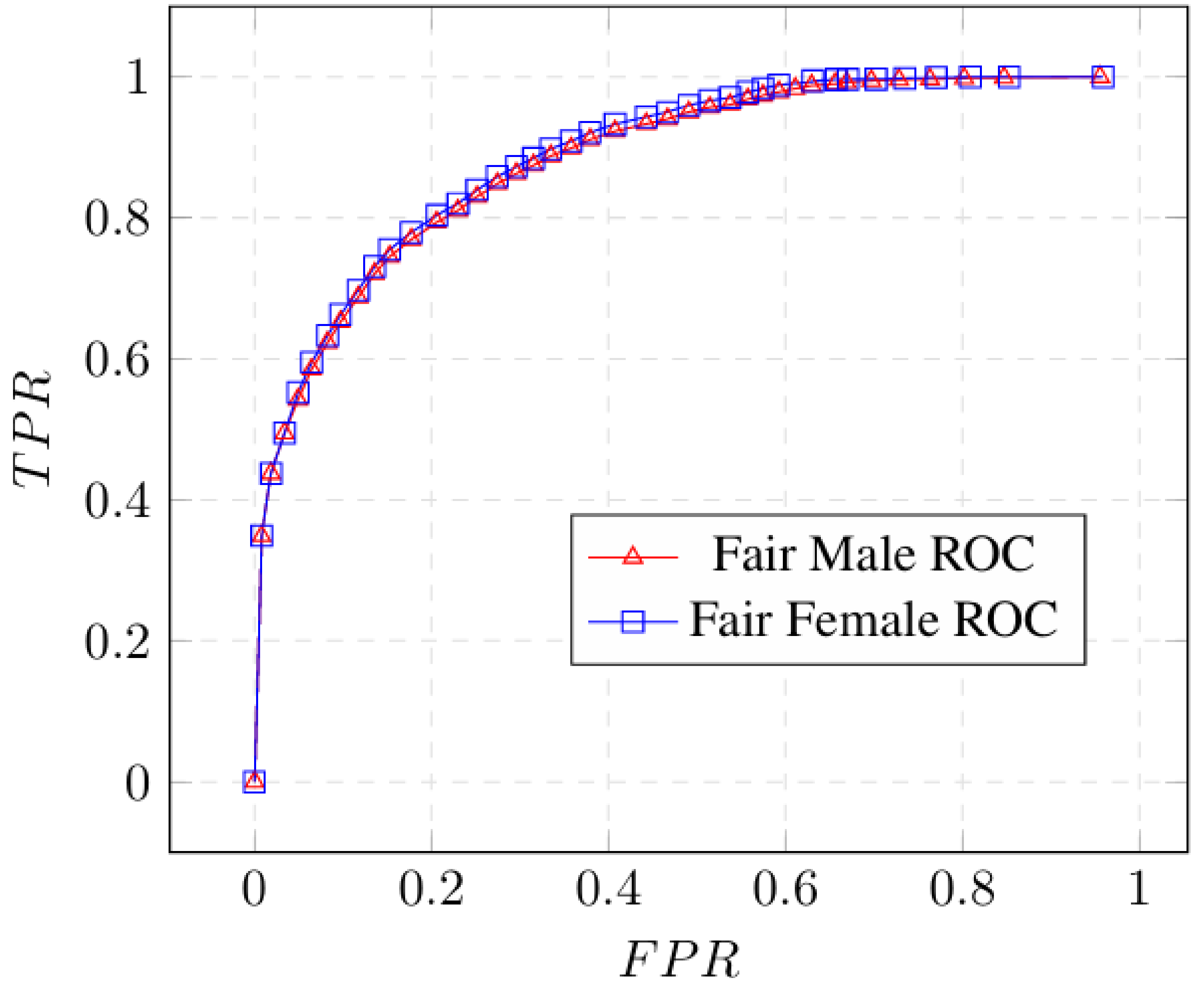

Figure 28: Random Forest (Gini) Baseline ROCs for Adult Dataset

F FROC implementation in Python

The official and cleaned-up version of the code for this paper can be found in this link.

F.1 Preprocessing Code (Adult)

```
from autogluon.tabular import TabularDataset, TabularPredictor
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import metrics
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
import math
import pandas as pd
import random as rd
import math




df_old = pd.read_csv('https://autogluon.s3.amazonaws.com/datasets/Inc/train.csv')
```

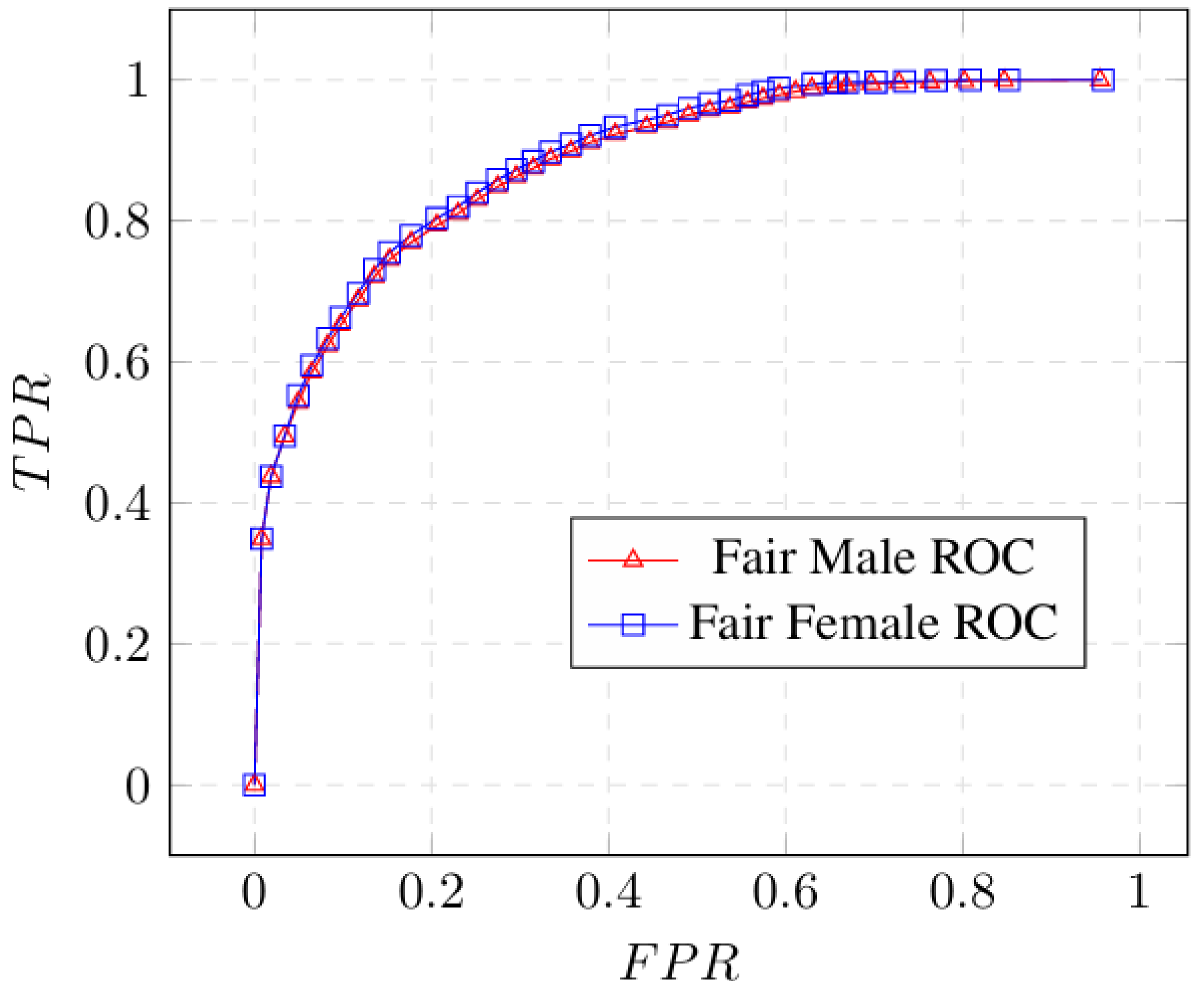

Figure 29: (Fair $\varepsilon_1 = 0.01$) Random Forest (Gini)-FROC ROCs for Adult Dataset

```

# column_names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num',
#                 'marital-status', 'occupation', 'relationship', 'race', 'sex',
#                 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'class']



# df_old = pd.read_csv('/content/adult.data' , header = None , names = column_names)
# df.columns =  column = ['age' , 'workclass' , 'fnlwgt' , 'education-num' , 'marital-status' , 'occupation' , 'relationship', 'race' , 'sex' , 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country' , 'class']
# Adult dataset is being loaded.
df = df_old.fillna(0)

print(df)
# print(df_old1)

## Modify for binary labels
df['class'].loc[df['class'] == '<=50K'] = 0
```

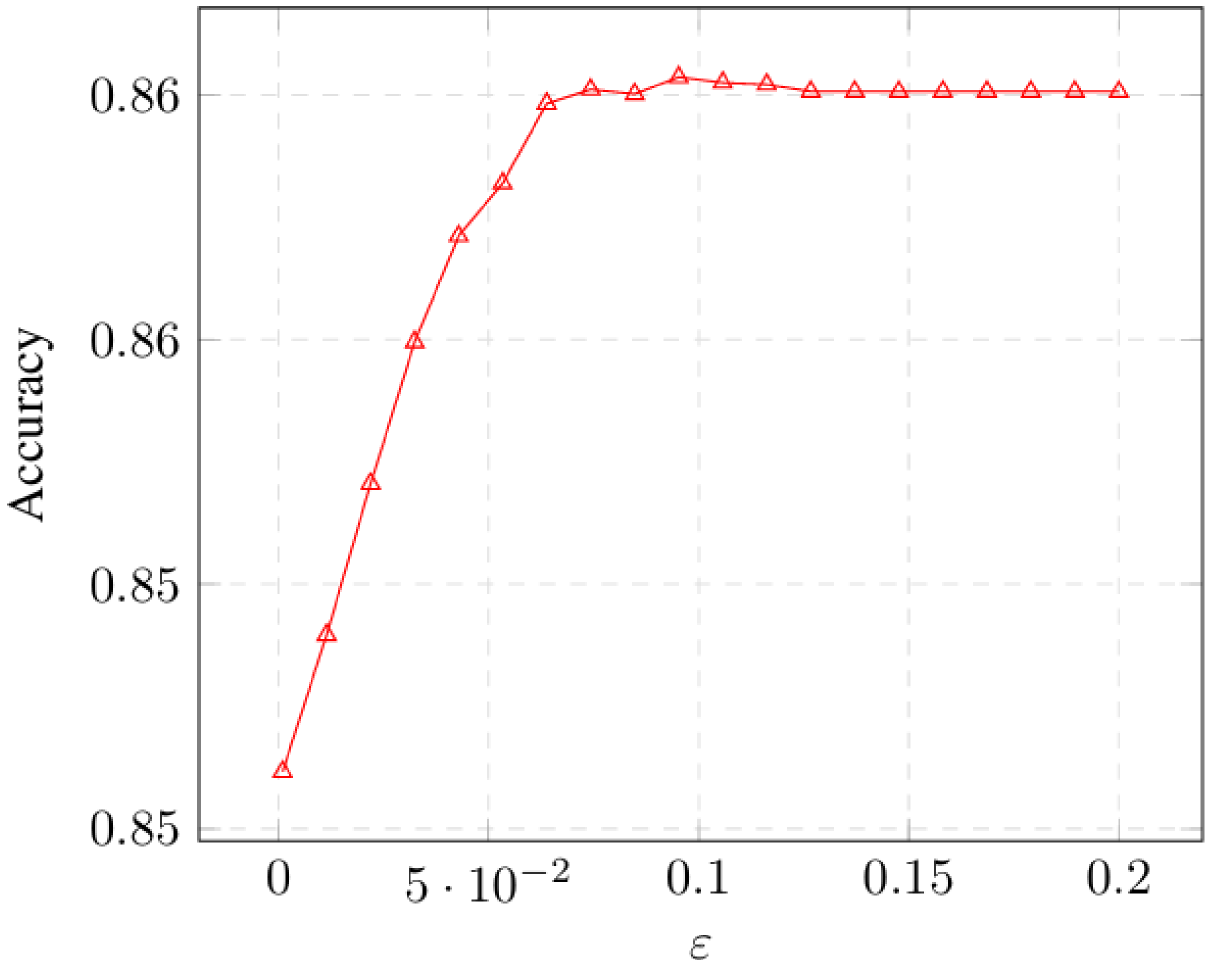

Figure 30: Random Forest (Gini)-FROC Accuracy vs. ε_1 (Adult)

```
df['class'].loc[df['class'] == '>50K'] = 1

## Create the dataset
for i in list(df.columns):
    df[i] = df[i].astype('category').cat.codes


## Modify for binary protected attributes
df['sex'].loc[df['sex'] == 'Male'] = 1
df['sex'].loc[df['sex'] == 'Female'] = 0
print(df)



test_data = TabularDataset('https://autogluon.s3.amazonaws.com/datasets/Inc/test.csv')
y_test = test_data[label]  # values to predict
DF = test_data
test_data_nolab = test_data.drop(columns=[label])  # delete label column to prove we're not cheating
# test_data_nolab = test_data.drop(columns=[])  # delete label column to prove we're not cheating
```

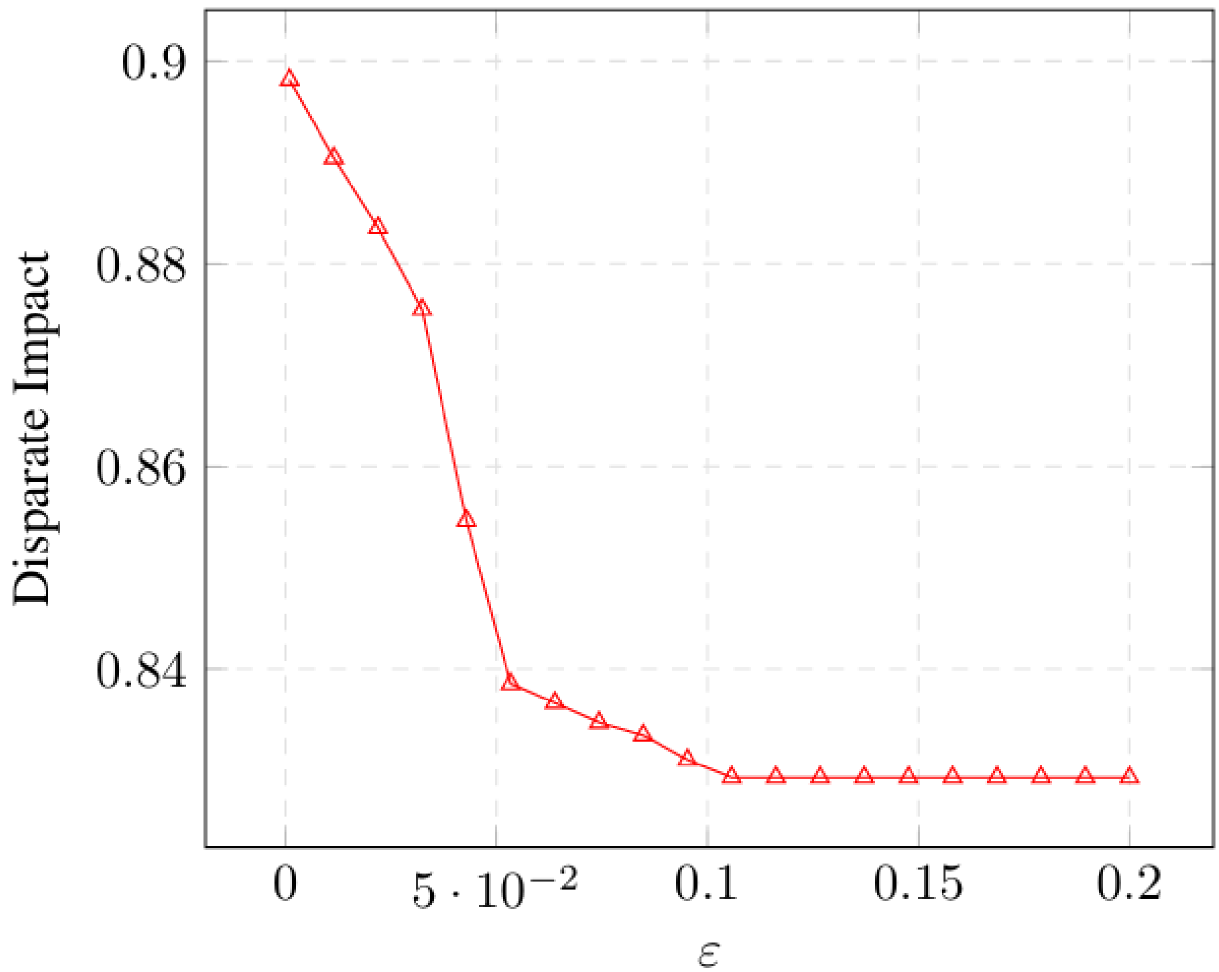

Figure 31: Random Forest (Gini)-FROC Disparate Impact vs. ε_1 (Adult)

```
test_data_nolab.head()
```

F.2 Preprocessing Code (COMPAS)

```
from autogluon.tabular import TabularDataset, TabularPredictor
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import metrics
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
import pandas as pd


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


data = TabularDataset('/content/propublica_data_for_fairml.csv')
data.info()
data.columns
```

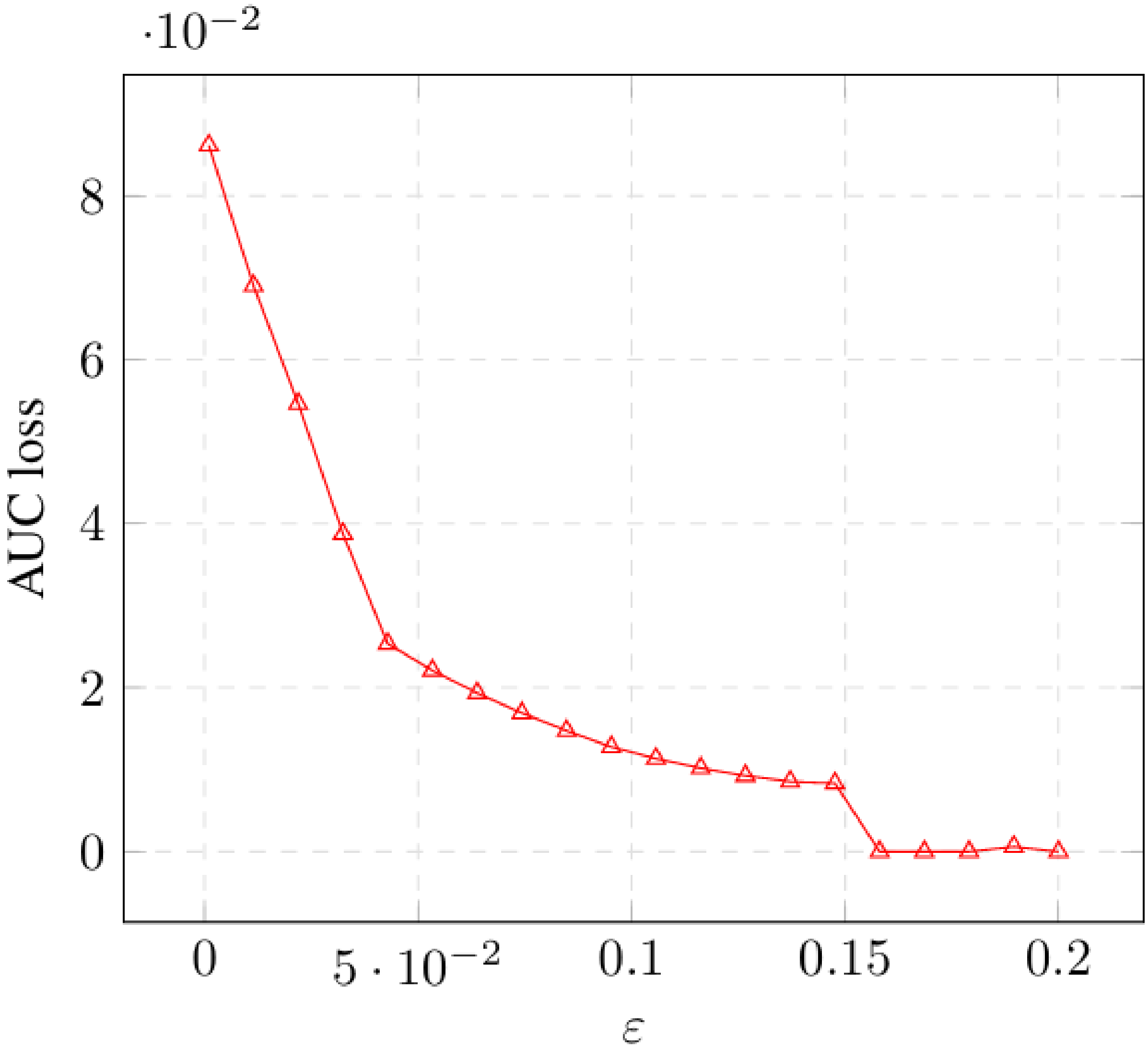

Figure 32: Random Forest (Gini)-FROC AUC loss vs. ε_1 (Adult)

```
label = 'Two_yr_Recidivism'
print("Summary of Two_yr_Recidivism variable: \n", data[label].describe())
#### Train test split ###
train_ix  = np.random.randint(0, len(data), int(0.8*len(data)))
# train_ix = range(len(data))
train = data.iloc[train_ix,:]
# train_data = train.iloc[train_ix, :]
train_data = train.iloc[:, [1,2,3,4,5,6,7,8,9,10,11]]
print(train_data)
train_labels = train.iloc[:,0]
# train_labels = train_labels[:, 0]
print(train_labels)



test_ix  = np.random.randint(0, len(data), int(0.2*len(data)))
# train_ix = range(len(data))
test = data.iloc[train_ix,:]
```

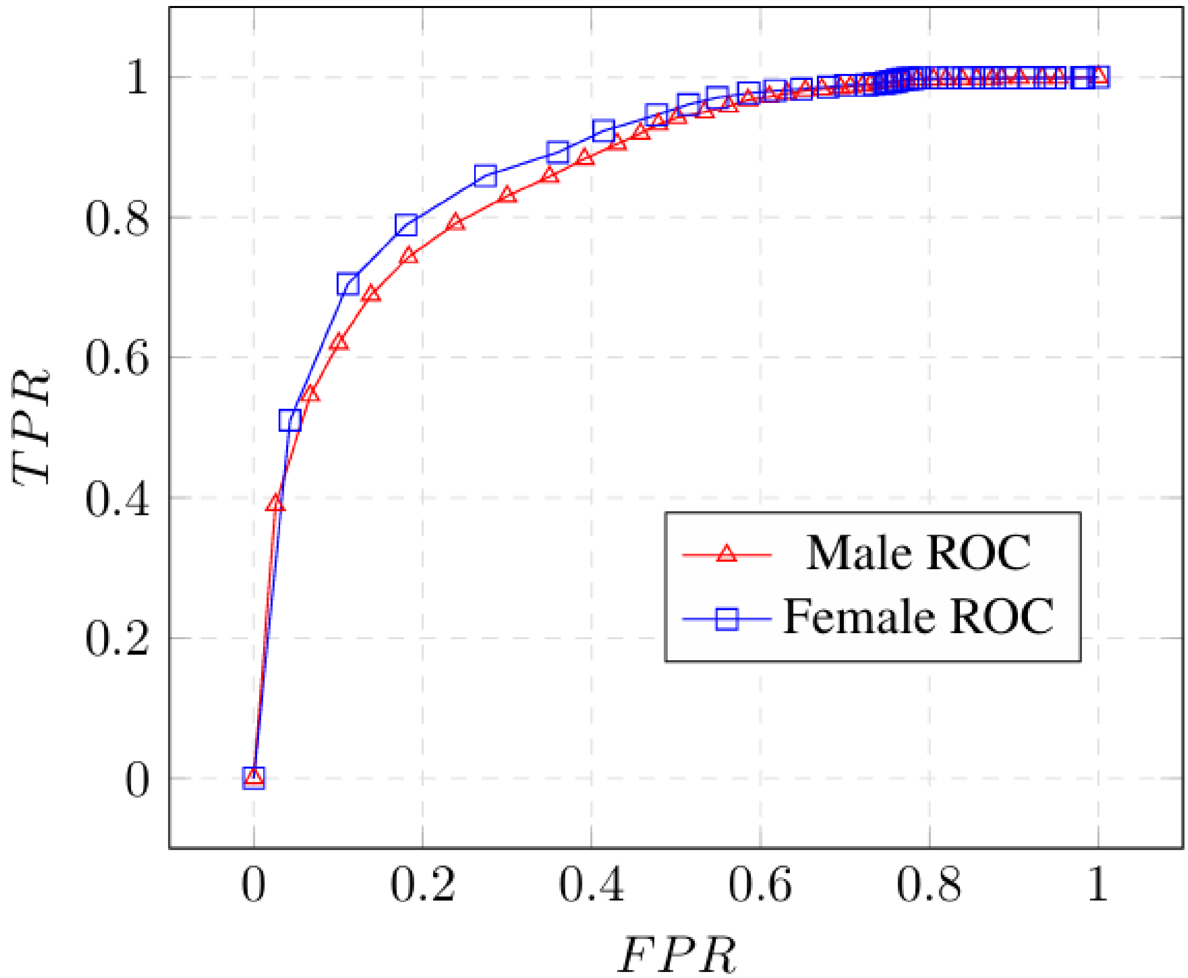

Figure 33: FNNC Baseline ROCs for Adult Dataset

```
test_data = test.iloc[:, [1,2,3,4,5,6,7,8,9,10,11]]
print(test_data)
test_labels = test.iloc[:,0]
assert isinstance(test_labels, (np.ndarray, pd.Series))
# test_labels = test_labels[:, 0]


# train_data = pd.DataFrame(train_data, columns = ['Number_of_Priors', 'score_factor','Age_Above_FourtyFive', 'Age_Below_TwentyFive', 'African_American','Asian', 'Hispanic', 'Native_American', 'Other', 'Female','Misdemeanor'])
# train_labels = pd.DataFrame(train_labels, columns = ['Two_yr_Recidivism'])

# test_data = pd.DataFrame(test_data, columns = ['Number_of_Priors', 'score_factor','Age_Above_FourtyFive', 'Age_Below_TwentyFive', 'African_American','Asian', 'Hispanic', 'Native_American', 'Other', 'Female','Misdemeanor'])
# test_labels = pd.DataFrame(test_labels, columns = ['Two_yr_Recidivism'])
# train_prot =  tf.keras.utils.to_categorical(prot[train_ix, np.newaxis], num_classes=num_classes)
# train_labels = tf.keras.utils.to_categorical(data[train_ix, -1], num_classes=num_classes)
# train_labels = np.append(train_labels, train_prot, 1)
```

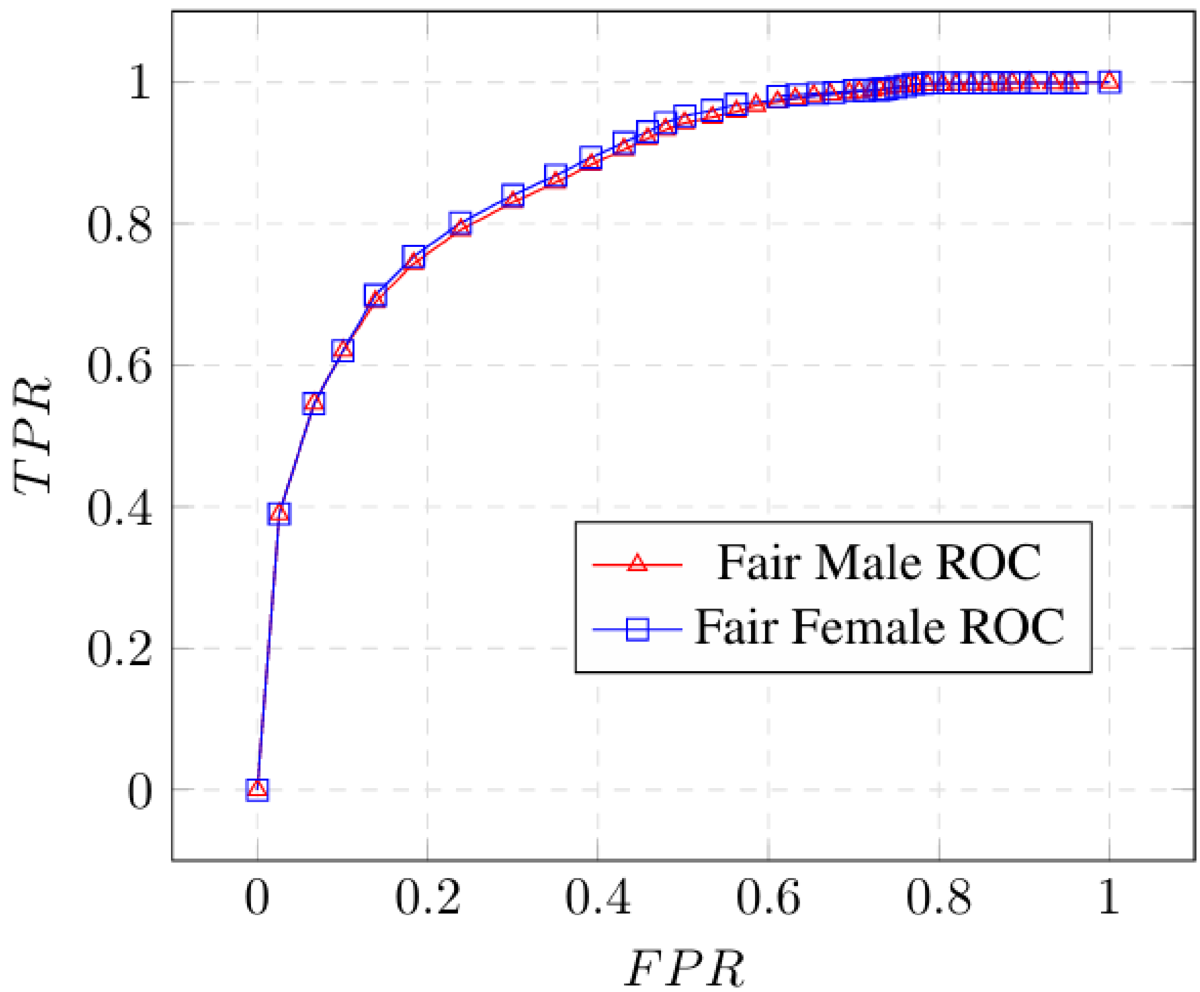

Figure 34: (Fair $\varepsilon_1 = 0.01$) FNNC-FROC ROCs for Adult Dataset

```

# test_ix = np.random.randint(0, len(data), int(0.2*len(data)))
# test_data = data[test_ix, :-1]
# test_prot =  tf.keras.utils.to_categorical(prot[test_ix, np.newaxis], num_classes
    =num_classes)
# test_labels = tf.keras.utils.to_categorical(data[test_ix, -1], num_classes=
    num_classes)
# test_labels = np.append(test_labels, test_prot, 1)
```

F.3 Preprocessing Code (CelebA)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.metrics import auc
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import normalize
from copy import deepcopy

# load and summarize the dataset
from pandas import read_csv
from collections import Counter
```

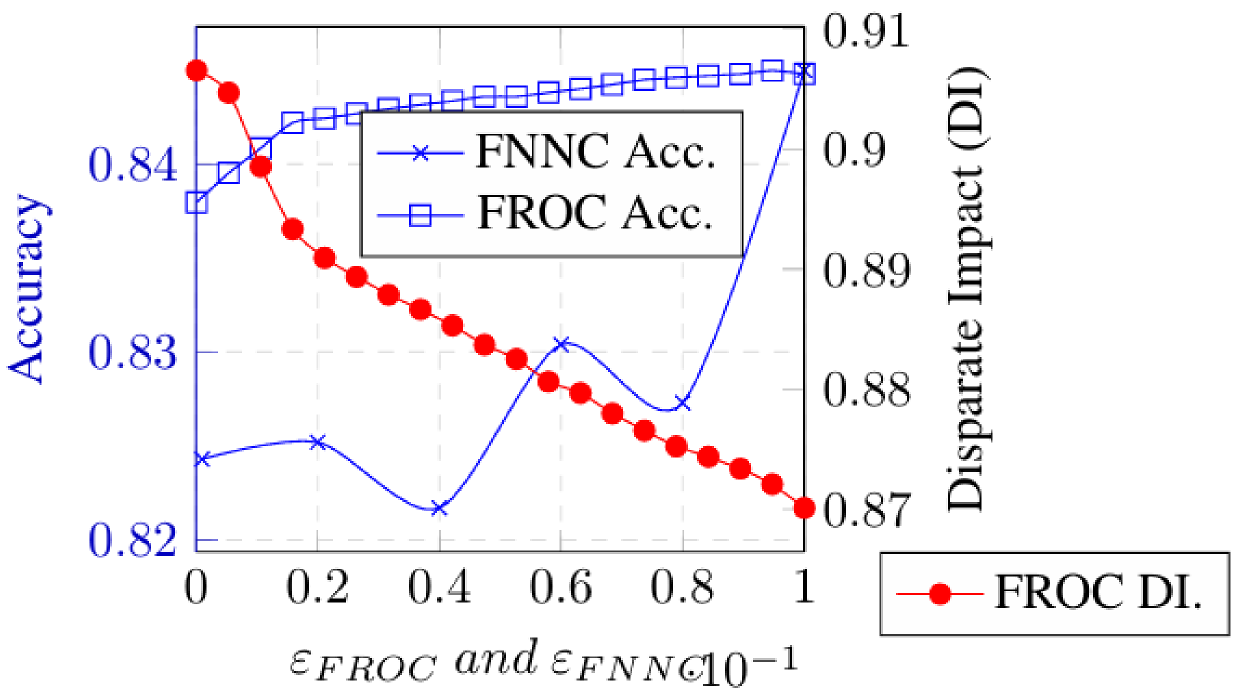

Figure 35: FNNC-FROC Accuracy vs. ε_1 (Adult)

```
# define the dataset location
filename = 'adult.csv'
# load the csv file as a data frame
df = read_csv(filename, header=None, na_values='?')
# drop rows with missing
df = df.dropna()
# summarize the shape of the dataset
print(df.shape)
# summarize the class distribution
target = df.values[:,-1]
counter = Counter(target)
for k,v in counter.items():
  per = v / len(target) * 100
  print('Class=%s, Count=%d, Percentage=%.3f%%' % (k, v, per))

# select columns with numerical data types
num_ix = df.select_dtypes(include=['int64', 'float64']).columns
# select a subset of the dataframe with the chosen columns
subset = df[num_ix]
# create a histogram plot of each numeric variable
# subset.hist()
plt.show()

# fit a model and make predictions for the on the adult dataset
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
```

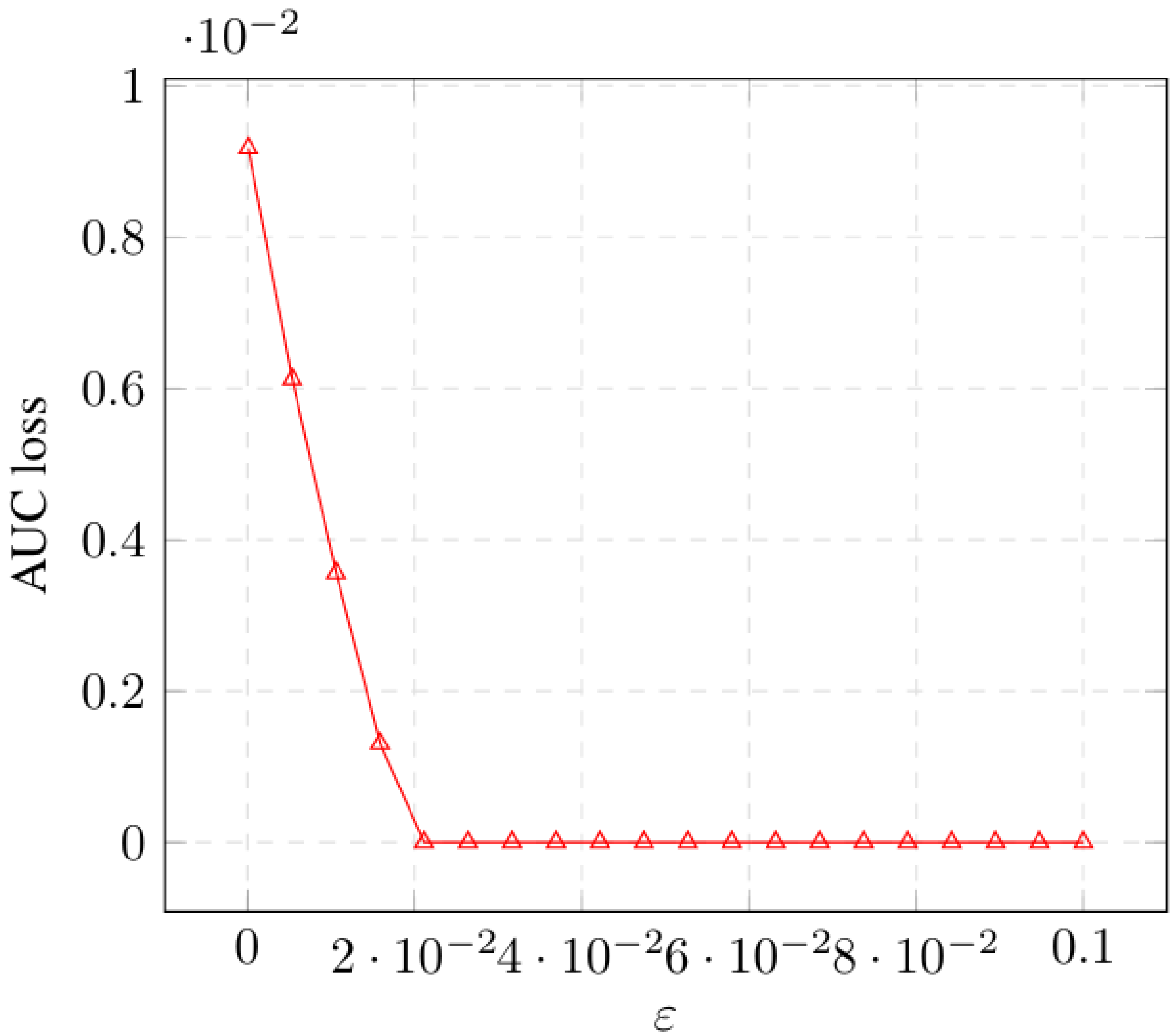

Figure 36: FNNC-FROC AUC loss vs. ε_1 (Adult)

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from imblearn.pipeline import Pipeline

# load the dataset
def load_dataset(full_path):
  # load the dataset as a numpy array
  dataframe = read_csv(full_path, header=None, na_values='?')
  # drop rows with missing
  dataframe = dataframe.dropna()
  # split into inputs and outputs
  last_ix = len(dataframe.columns) - 1
  X, y = dataframe.drop(last_ix, axis=1), dataframe[last_ix]
  # select categorical and numerical features
  cat_ix = X.select_dtypes(include=['object', 'bool']).columns
  num_ix = X.select_dtypes(include=['int64', 'float64']).columns
  # label encode the target variable to have the classes 0 and 1
  y = LabelEncoder().fit_transform(y)
  return X.values, y, cat_ix, num_ix

# define the location of the dataset
```

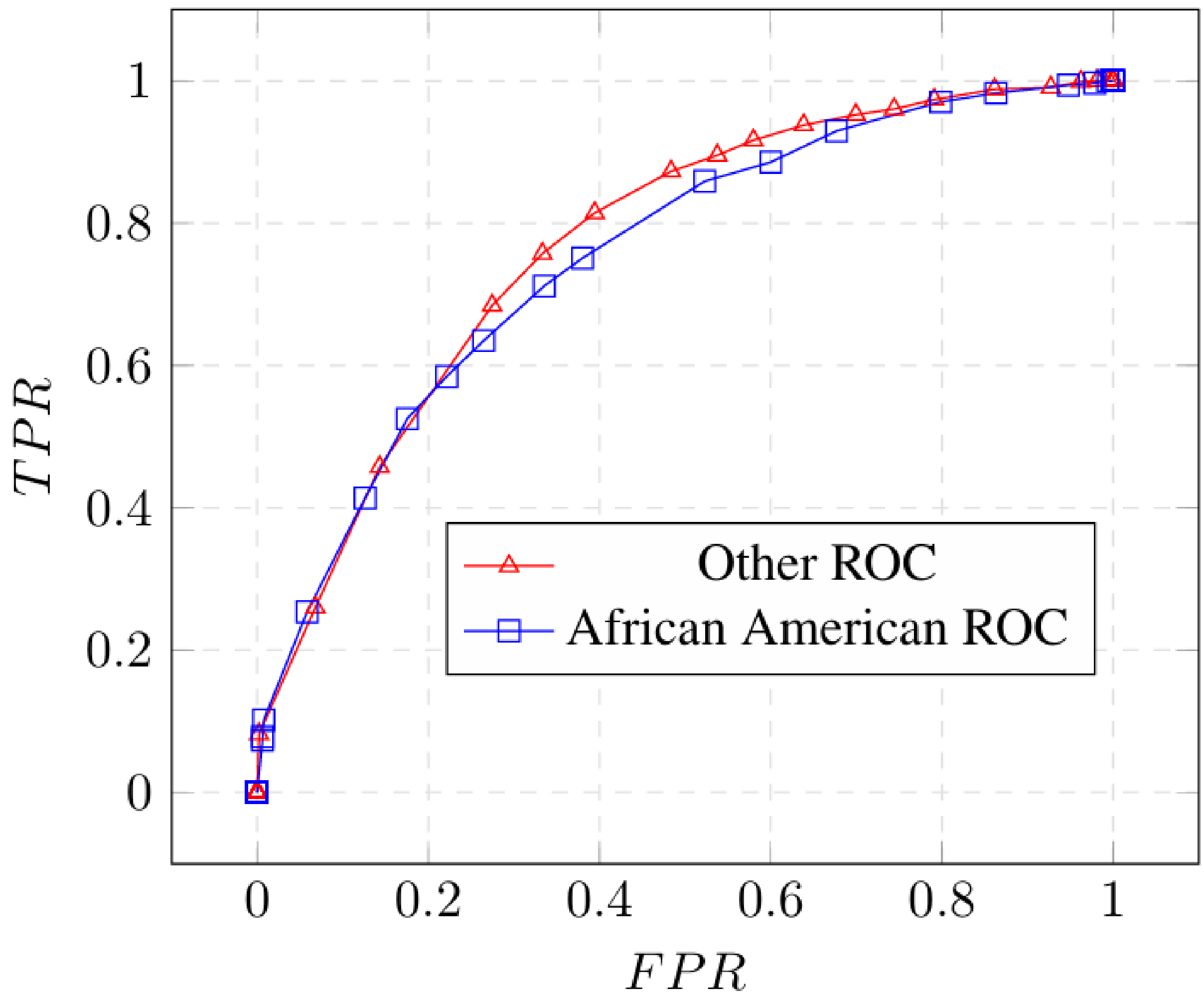

Figure 37: Weighted Ensemble L2 Baseline ROCs for COMPAS Dataset

```
full_path = 'adult.csv'
# load the dataset
X, y, cat_ix, num_ix = load_dataset(full_path)
# define model to evaluate
model = GradientBoostingClassifier(n_estimators=100)
model2 = SVC()

# one hot encode categorical, normalize numerical
ct = ColumnTransformer([('c',OneHotEncoder(handle_unknown = 'ignore'),cat_ix), ('n'
    ,MinMaxScaler(),num_ix)])
# define the pipeline
pipeline = Pipeline(steps=[('t',ct), ('m',model)])
pipeline2 = Pipeline(steps=[('t',ct), ('m',model2)])
# split test and train data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
# fit the model
trained_model = pipeline.fit(X_train, y_train)
trained_model2 = pipeline2.fit(X_train, y_train)
```

F.4 FROC

```
def Cover( curve_x , curve_y , x , y ):
```

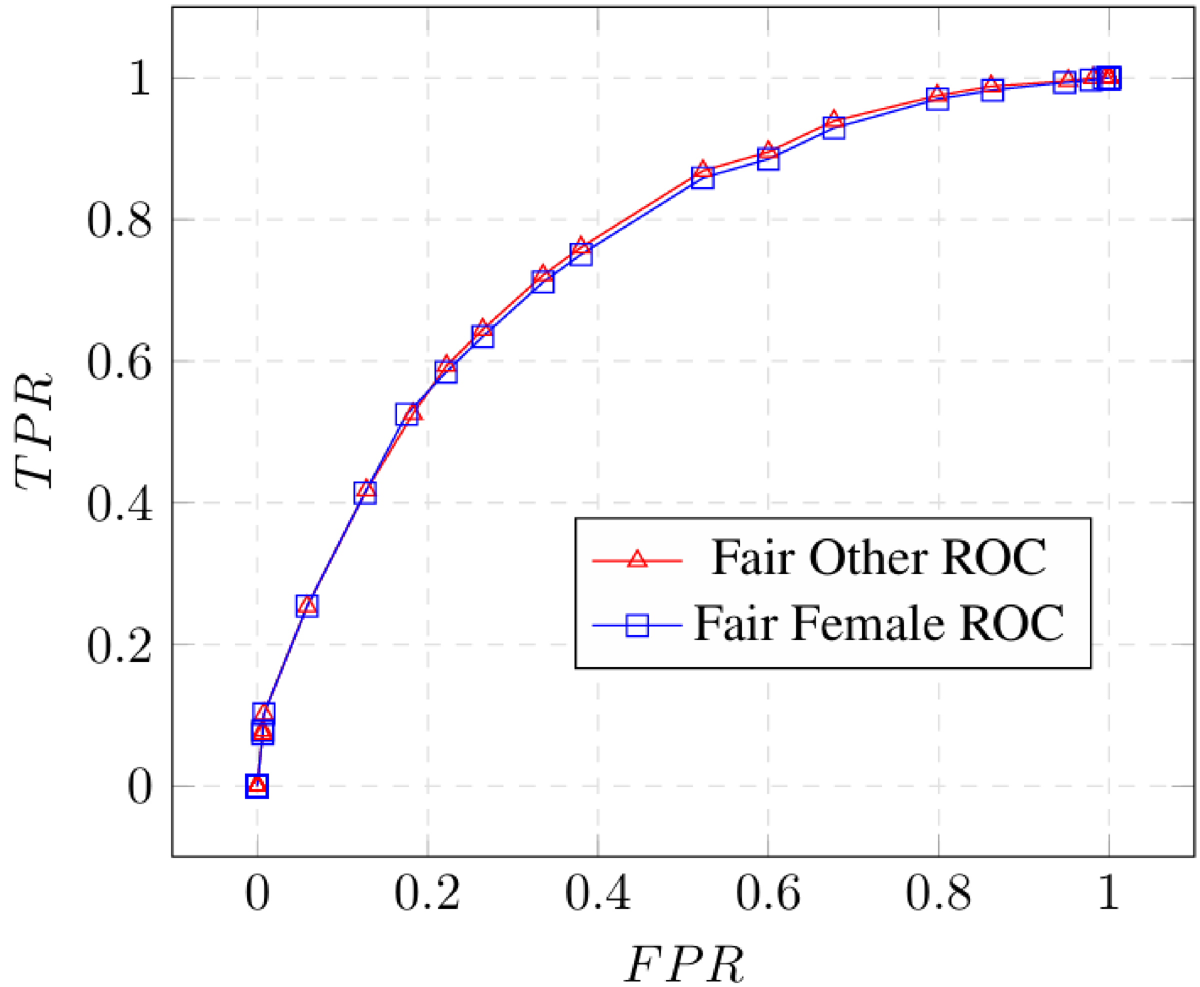

Figure 38: (Fair $\varepsilon_1 = 0.01$) Weighted Ensemble L2-FROC ROCs for COMPAS Dataset

```
  j = 0
  for i in range(len(curve_x)):
    if ( i == len(curve_x)-1):
      print("Case")
      if( x <= curve_x[i] ):
        if( y <= curve_y[i]):
          return 1
        else:
          return 0
      else:
        return 0
      continue
    if (curve_x[i] <= x and curve_x[i+1] >= x):
      if( y <= curve_y[i] + (curve_y[i+1] - curve_y[i])*(x - curve_x[i])/(curve_x[i
+1] - curve_x[i]) ):
        return 1
      else:
        return 0

def LinInterpolFill(X, Y, n):
    """
```

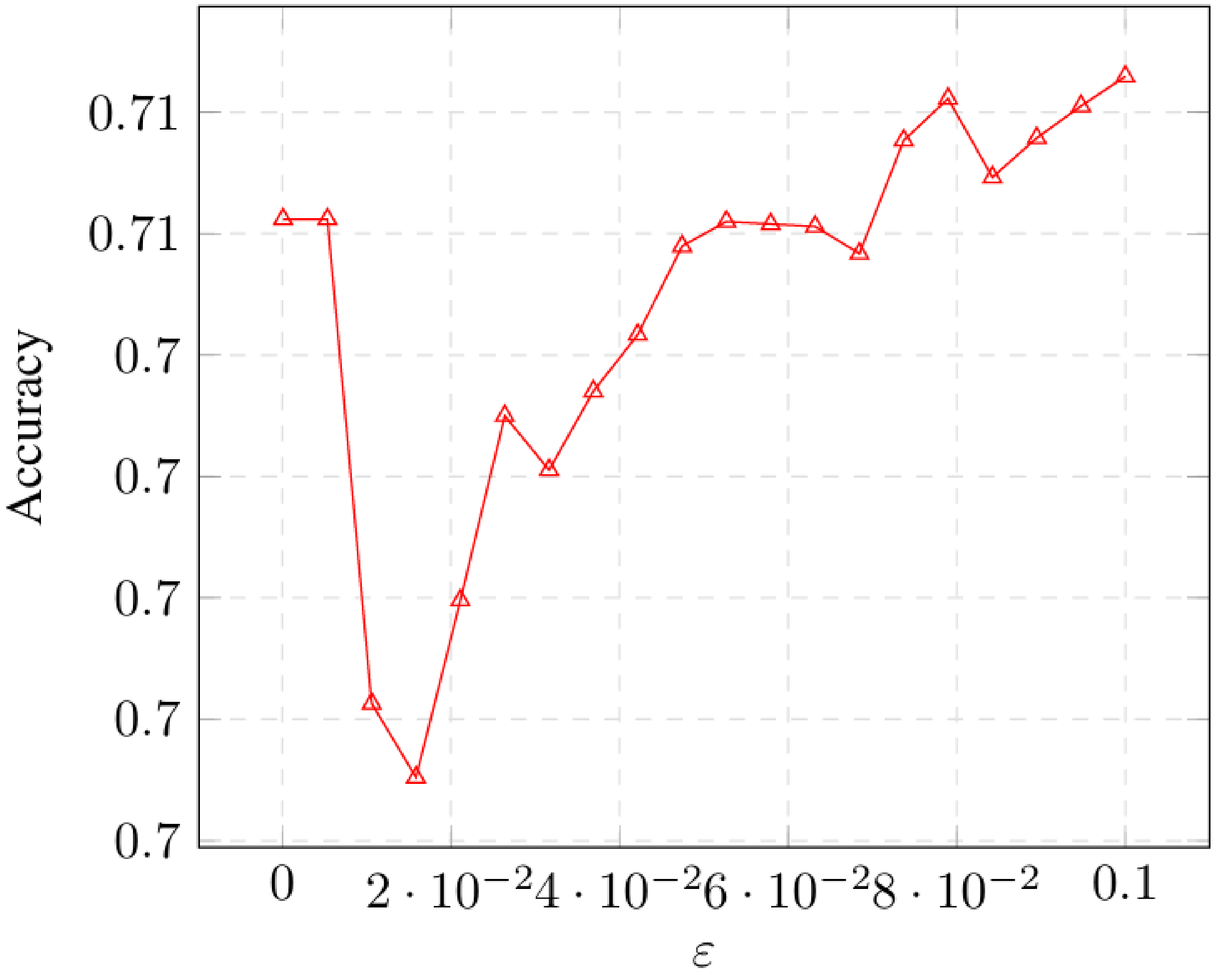

Figure 39: Weighted Ensemble L2-FROC Accuracy vs. ε_1 (COMPAS)

```
    Linearly interpolate between consecutive (X,Y) coordinates and fill in n points between them.

    Parameters:
    X (list): List of x-coordinates.
    Y (list): List of y-coordinates.
    n (int): Number of points to interpolate between each consecutive (X,Y) pair.

    Returns:
    x_interpolated (list): List of interpolated x-coordinates.
    y_interpolated (list): List of interpolated y-coordinates.
    """

    x_interpolated = []
    y_interpolated = []

    for i in range(len(X)-1):
        x0 = X[i]
        x1 = X[i+1]
        y0 = Y[i]
        y1 = Y[i+1]

        for j in range(n+1):
```

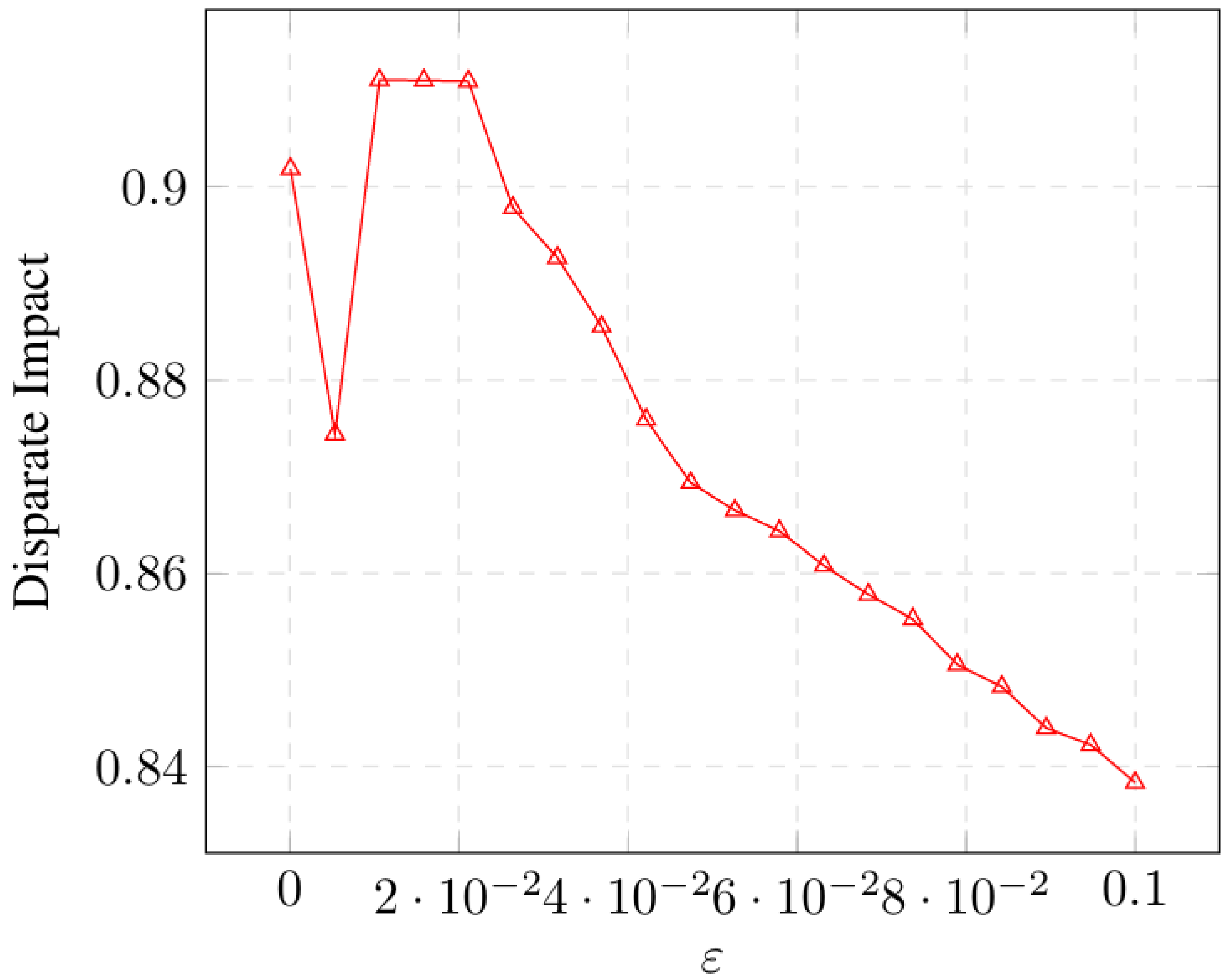

Figure 40: Weighted Ensemble L2-FROC Disparate Impact vs. ε_1 (COMPAS)

```
            x_j = x0 + (x1-x0)*j/n
            y_j = y0 + (y1-y0)*j/n
            x_interpolated.append(x_j)
            y_interpolated.append(y_j)

    return x_interpolated, y_interpolated


def FROC_original( iFPR0 , iTPR0 , iFPR1 , iTPR1 , granularity , epsilon ):
  # plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )
  FPR0 = iFPR0.copy()
  TPR0 = iTPR0.copy()
  FPR1 = iFPR1.copy()
  TPR1 = iTPR1.copy()
  FPR0 = np.flip(FPR0)
  TPR0 = np.flip(TPR0)
  FPR1 = np.flip(FPR1)
  TPR1 = np.flip(TPR1)

  FFPR0 = FPR0.copy()
  FTPR0 = TPR0.copy()
  FFPR1 = FPR1.copy()
```

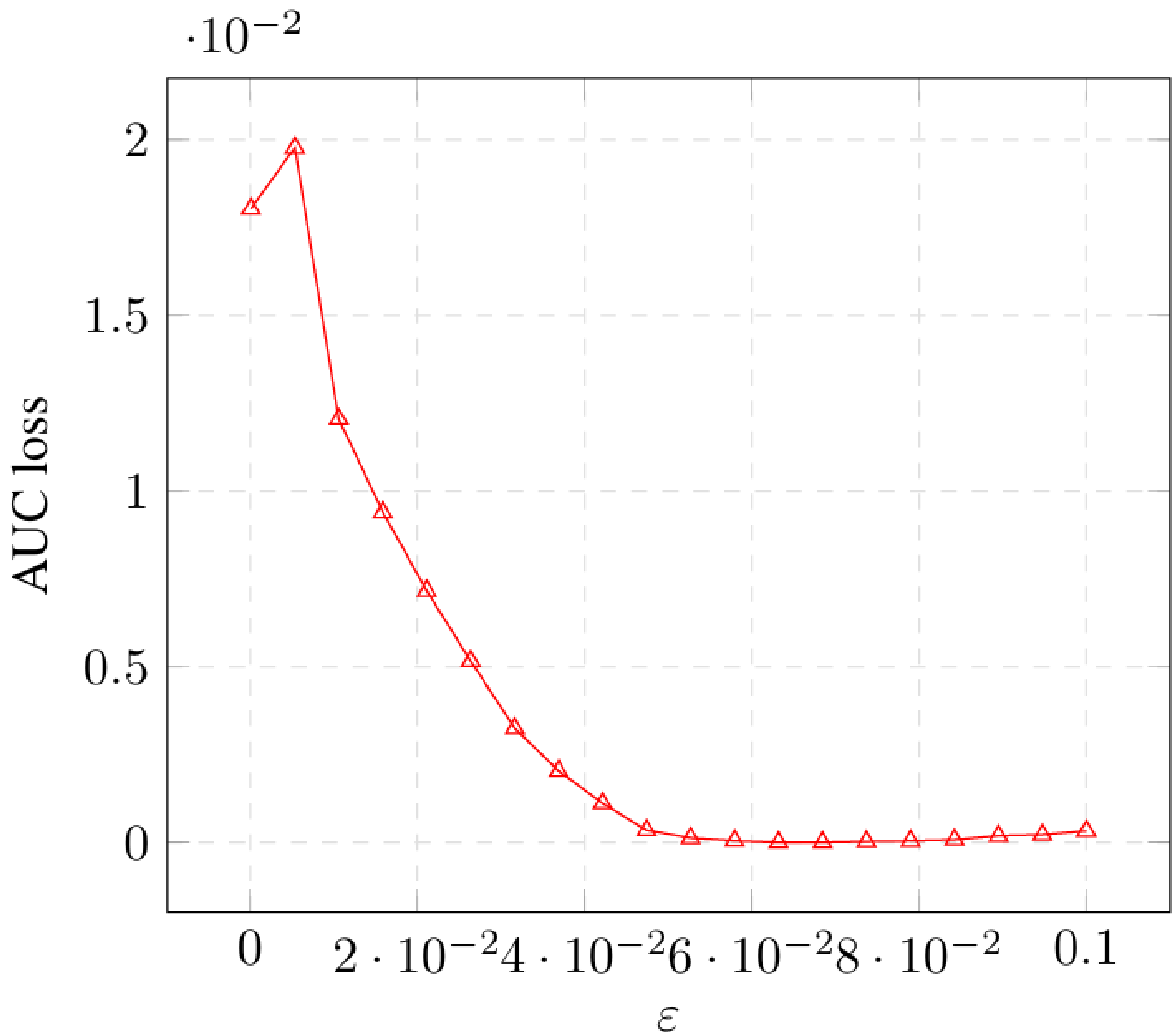

Figure 41: Weighted Ensemble L2-FROC AUC loss vs. ε_1 (COMPAS)

```
FTPR1 = TPR1.copy()

# plt.plot( FFPR0 , FTPR0 )


# plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )
linFPR0 , linTPR0 = LinInterpolFill( FPR0 , TPR0 , granularity)
# plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )

init = 0.2
fin = 1

n = len(FPR0)
notFair = list(range(n))
# plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )

for i in range(len(notFair)):
  if( FPR0[i] < init or FPR0[i] > fin):
    notFair[i] = 'f'
```

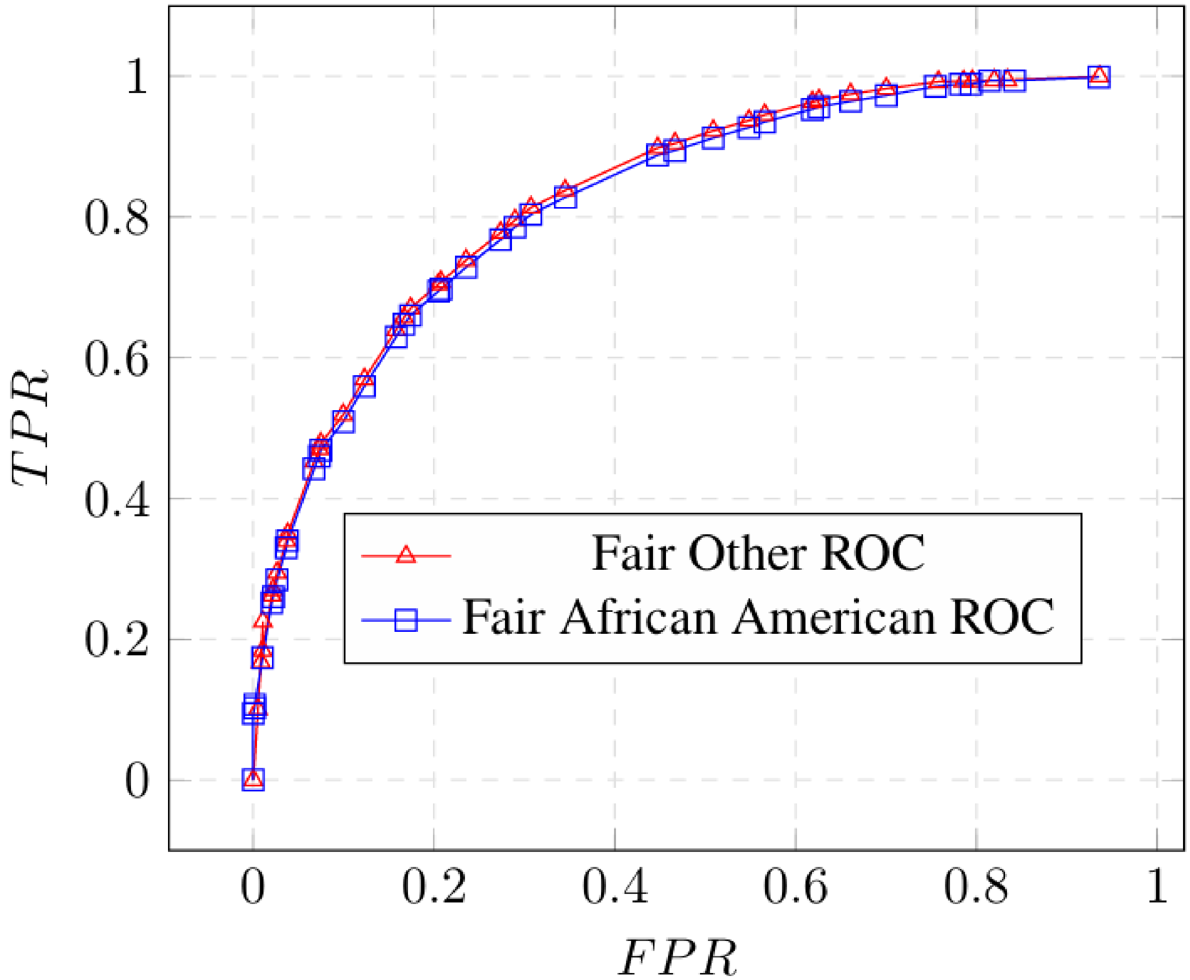

Figure 42: Random Forest (Gini) Baseline ROCs for COMPAS Dataset

```
        # print("Preprocessing range removed: ",i)
        FFPR0[i] = FPR1[i]
        FTPR0[i] = TPR1[i]
    # plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )

    for i in range(len(notFair)):
      if( abs(FPR0[i] - FPR1[i]) + abs(TPR0[i] - TPR1[i]) <= epsilon ):
        notFair[i] = 'f'
        # print("Preprocessing already fair: ",i)
    # plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )

    while 'f' in notFair:
      notFair.remove('f')

    plt.plot( FFPR0 , FTPR0 , FFPR1 , FTPR1 )


    for i in range(len(notFair)):
      # print("In loop")
      # plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )
      # print(Cover(FPR0 , TPR0 , FPR0[notFair[i]] , TPR0[notFair[i]]+epsilon))
      # print("Group0: ",FPR0[notFair[i]] , TPR0[notFair[i]])
```

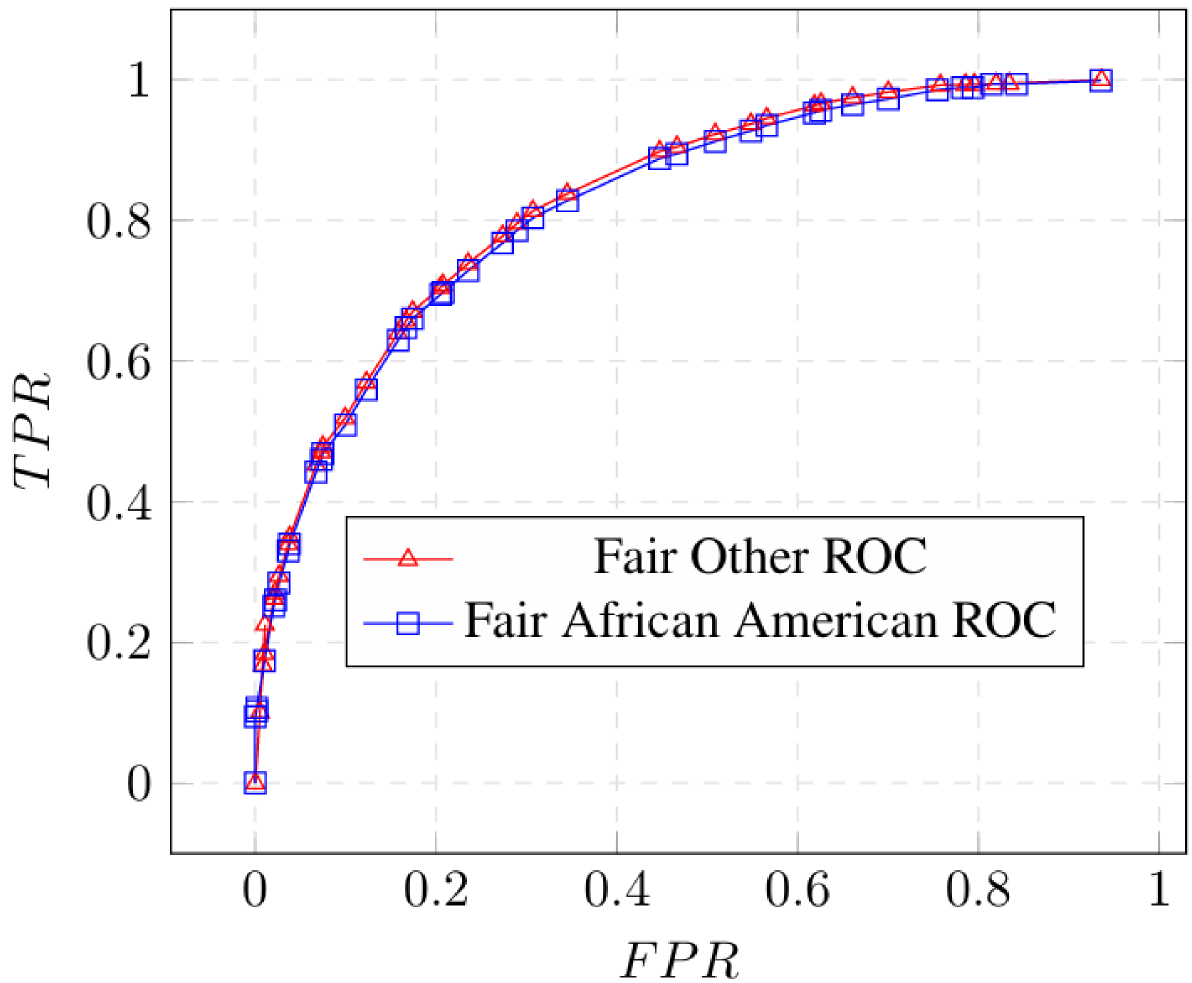

Figure 43: (Fair $\varepsilon_1 = 0.01$) Random Forest (Gini)-FROC ROCs for COMPAS Dataset

```
    # print("Group1: ",FPR1[notFair[i]] , TPR1[notFair[i]])
    if( Cover(FPR0 , TPR0 , FPR1[notFair[i]] , TPR1[notFair[i]]+epsilon)  == 1):
      # plt.plot( iFPR0 , iTPR0 , iFPR1 , iTPR1 )
      FTPR0[notFair[i]] = TPR1[notFair[i]] + epsilon
      FFPR0[notFair[i]] = FPR1[notFair[i]]
      # print("Upshift done", FFPR0[notFair[i]] , FTPR0[notFair[i]])
    else:
      for j in range(len(linFPR0)-1):
        if( abs(linFPR0[j] - FPR1[notFair[i]]) + abs(linTPR0[j] - TPR1[notFair[i]]) <= epsilon and abs(linFPR0[j+1] - FPR1[notFair[i]]) + abs(linTPR0[j+1] - TPR1[notFair[i]]) > epsilon  ):
          # print("Cut")
          FFPR0[notFair[i]] = linFPR0[j]
          FTPR0[notFair[i]] = linTPR0[j]
        # else:
          # print("Not Cut")

  # print( notFair )
  return FFPR0 ,FTPR0 , FFPR1 , FTPR1
```

F.5 Building the Classifier

```
def findInterval( vector , value ):
```

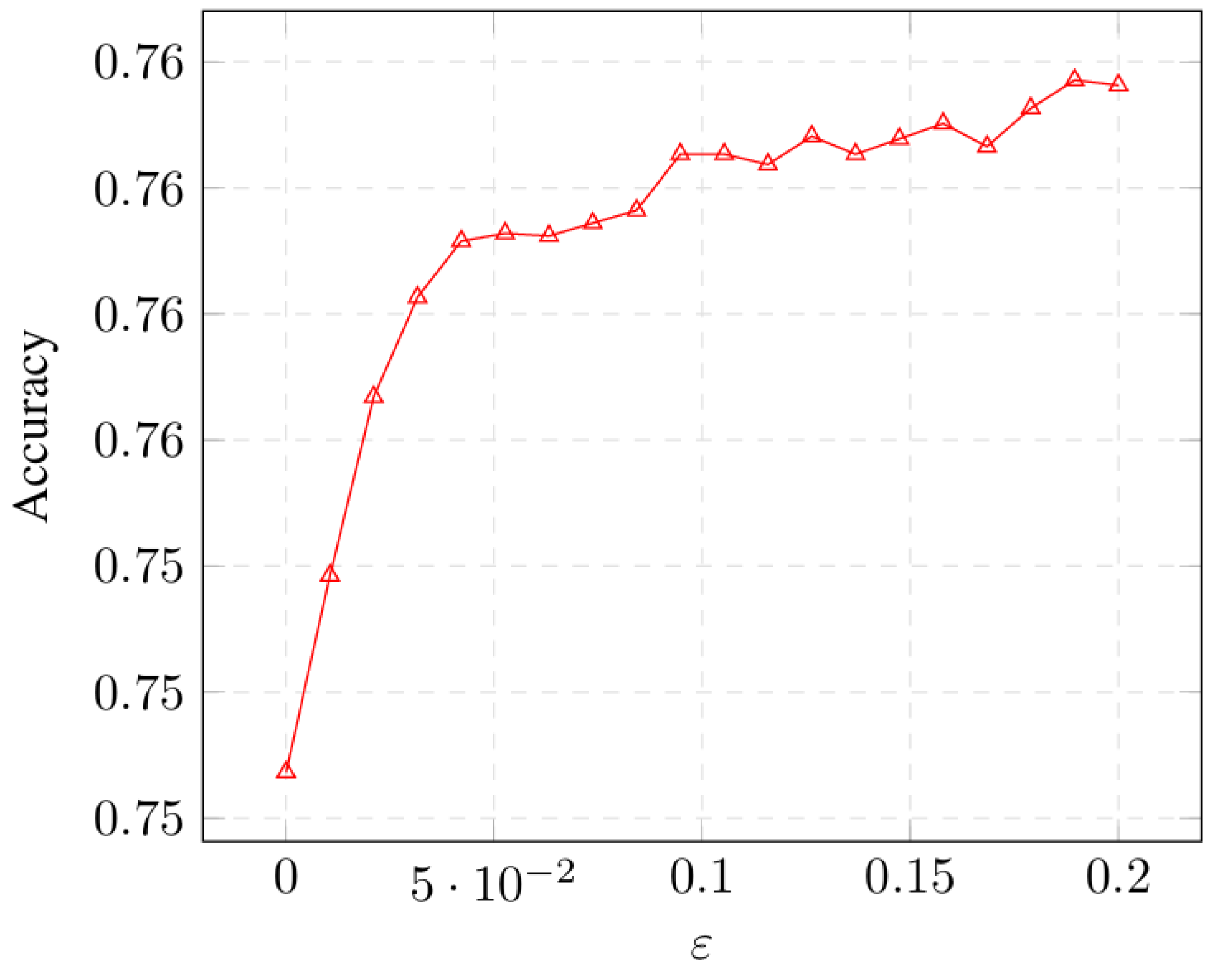

Figure 44: Random Forest (Gini)-FROC Accuracy vs. ε_1 (COMPAS)

```
    # vector is a sorted vector.
    # if the vector is not sorted in a decreasing order, then declare an error.
    # Check if the vector is sorted in a decreasing order.
    for i in range(1, vector.shape[0]):
        if vector[i] > vector[i - 1]:
            print("Error: vector is not sorted in a decreasing order.")
            return -1

    # iF the value is outside the range of the vector, then throw an error.
    if value < vector[-1] or value > vector[0]:
        print("Error: value is outside the range of the vector.")
        return -1

    # Else, find the interval in which the value lies.
    for i in range( vector.shape[0] ):
        if vector[i] < value:
            return i - 1


# Now, let us test the findInterval function
vector = np.linspace(1, 0, 10)
print(vector)
value = 0.5
```

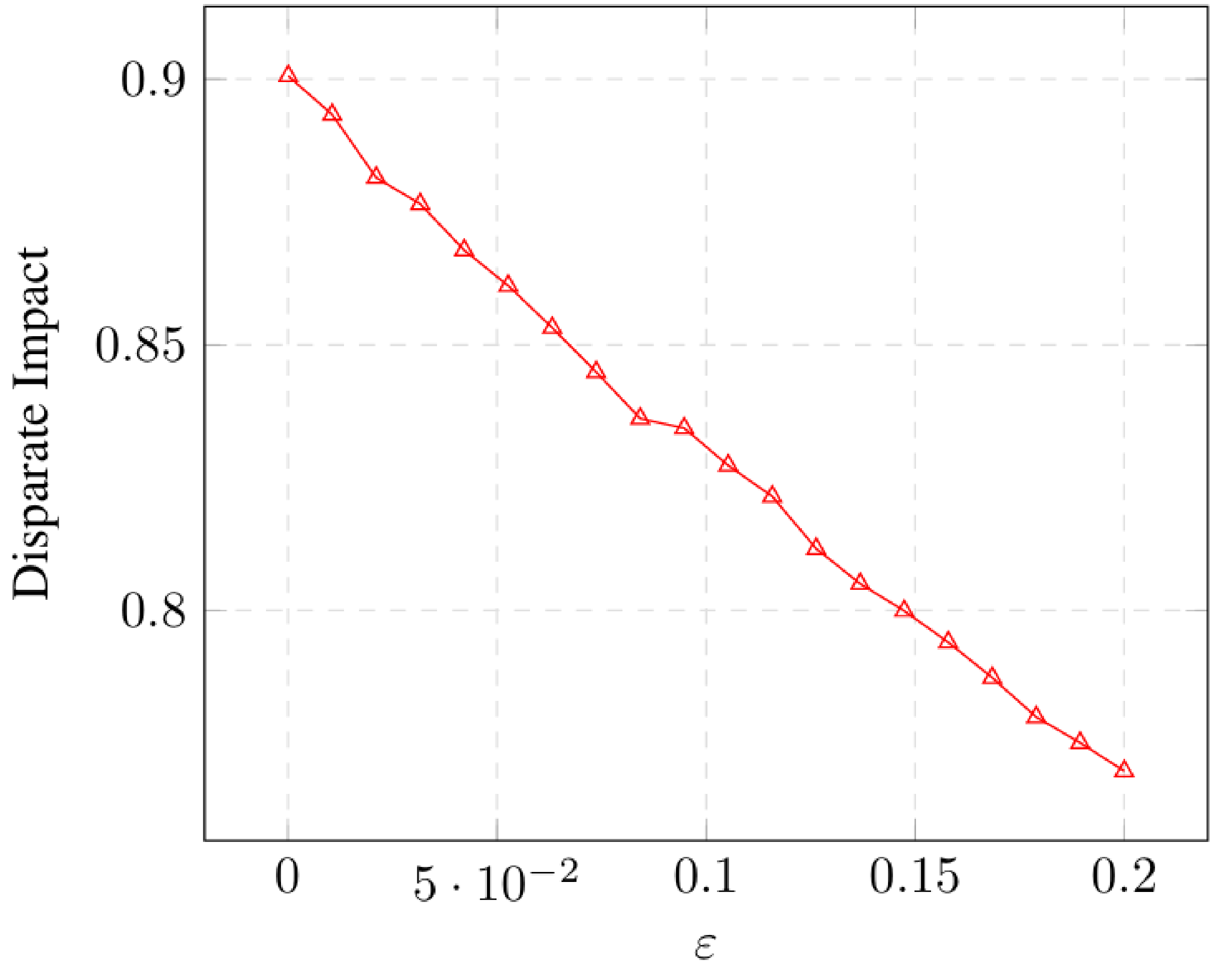

Figure 45: Random Forest (Gini)-FROC Disparate Impact vs. ε_1 (COMPAS)

```

print(findInterval(vector, value))


def returnCoeff( ul , ur , dn  , p ):
    # let ul = (a,b)
    # let ur = (c,d)
    # let p = (x,y)
    # let dn = (x,x)
    # Assert that dn[0] == dn[1] == p[0]
    assert dn[0] == dn[1] == p[0]
    a = ul[0]
    b = ul[1]
    c = ur[0]
    d = ur[1]
    x = p[0]
    y = p[1]

    # Now, if p is equal to any of the other points, then return the coefficient as
     1 for that point and 0 for the other points.
    if p[0] == ul[0] and p[1] == ul[1]:
        return (1, 0, 0)
```

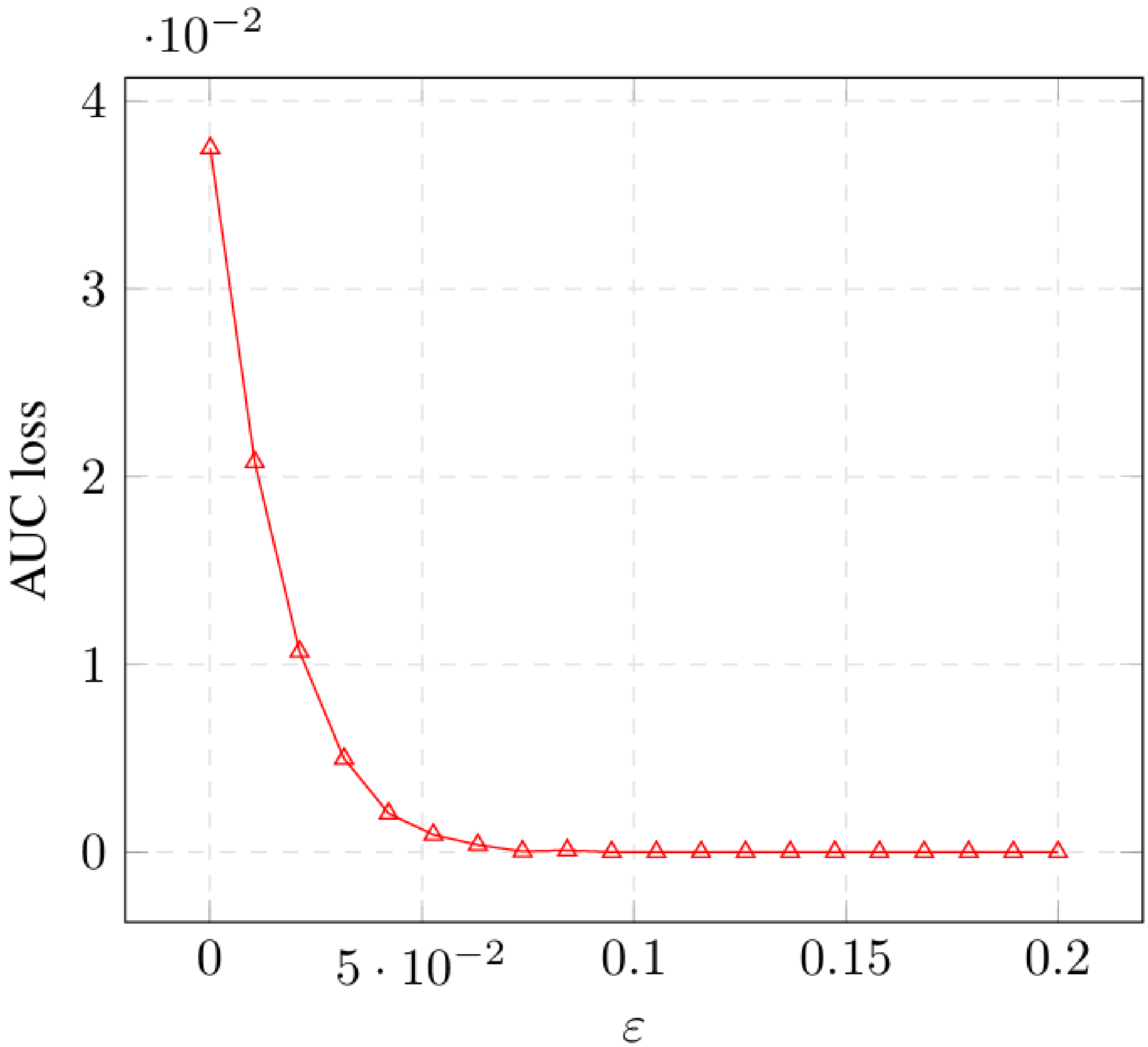

Figure 46: Random Forest (Gini)-FROC AUC loss vs. ε_1 (COMPAS)

```
    elif p[0] == ur[0] and p[1] == ur[1]:
        return (0, 1, 0)
    elif p[0] == dn[0] and p[1] == dn[1]:
        return (0, 0, 1)

    # Now, we find the coefficients of the line joining ul and ur.
    # Let h = ((c-x)/(c-a))*b + ((x-a)/(c-a))*d
    h = ((c-x)/(c-a))*b + ((x-a)/(c-a))*d
    # If c == a, then throw an error.
    if c == a:
        print("Error: Division by zero because c == a.")
        return -1

    # Now, we find C_ul, C_ur, C_dn
    C_ul = ((y - x)*(c - x))/((h - x)*(c - a))
    C_ur = ((y - x)*(x - a))/((h - x)*(c - a))
    C_dn = (h - y)/(h - x)

    # Now, if any of the coefficients are negative, then throw an error.
```

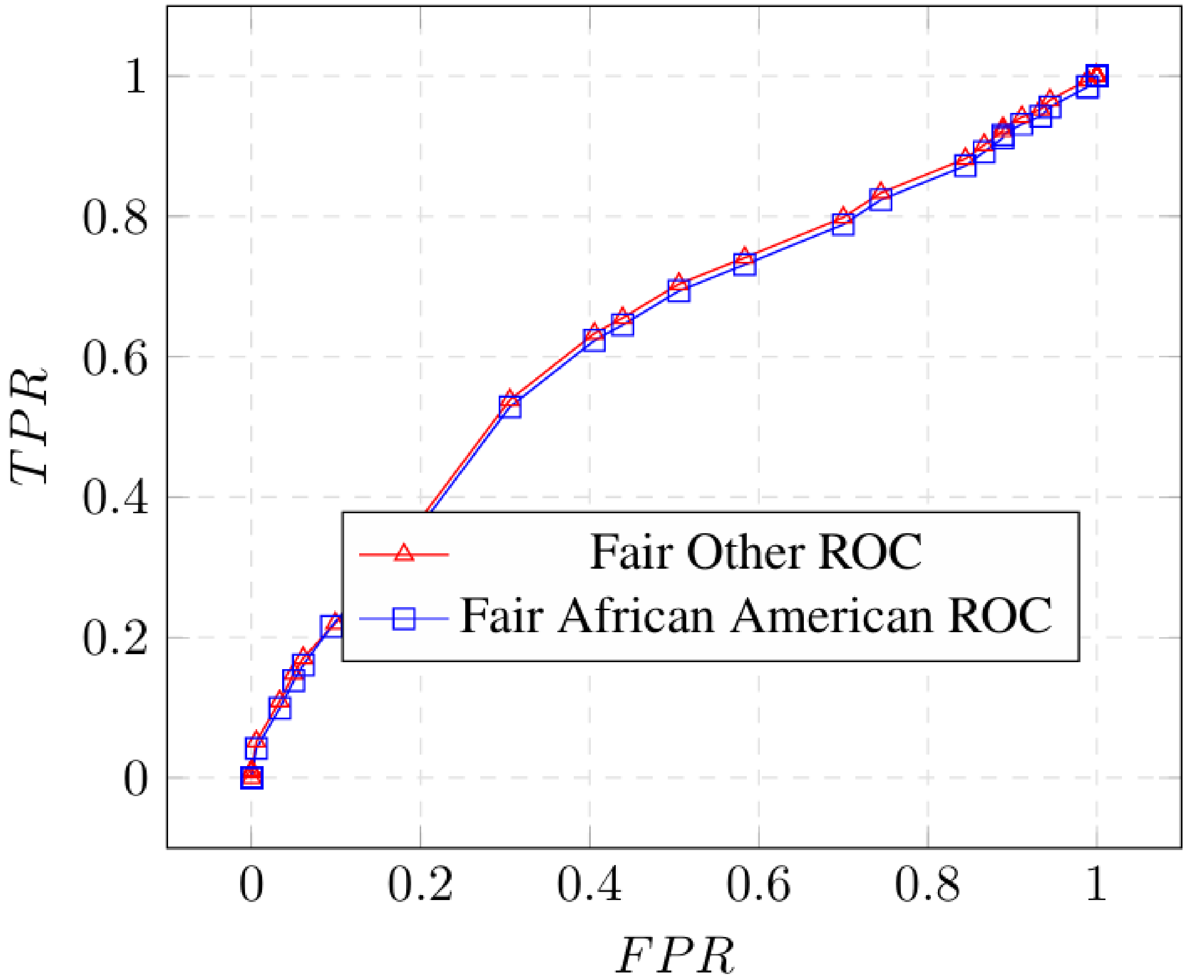

Figure 47: FNNC Baseline ROCs for COMPAS Dataset

```
    if C_ul < 0 or C_ur < 0 or C_dn < 0:
        print("Error: Negative coefficient.")
        return -1

    # Now, if any of the coefficients are greater than 1 or nan, then throw an error.
    if C_ul > 1 or C_ur > 1 or C_dn > 1 or np.isnan(C_ul) or np.isnan(C_ur) or np.isnan(C_dn):
        print("Error: Coefficient greater than 1 or nan.")
        return -1

    # Assert that C_ul + C_ur + C_dn = 1
    # print(C_ul + C_ur + C_dn)
    assert C_ul + C_ur + C_dn - 1 < 0.00001

    # If any of the coefficients are na because of division by zero, then throw an error.
    if np.isnan(C_ul) or np.isnan(C_ur) or np.isnan(C_dn):
        print("Error: Division by zero.")
        return -1

    # If any of the nocoefficients are negative, then throw an error.
```

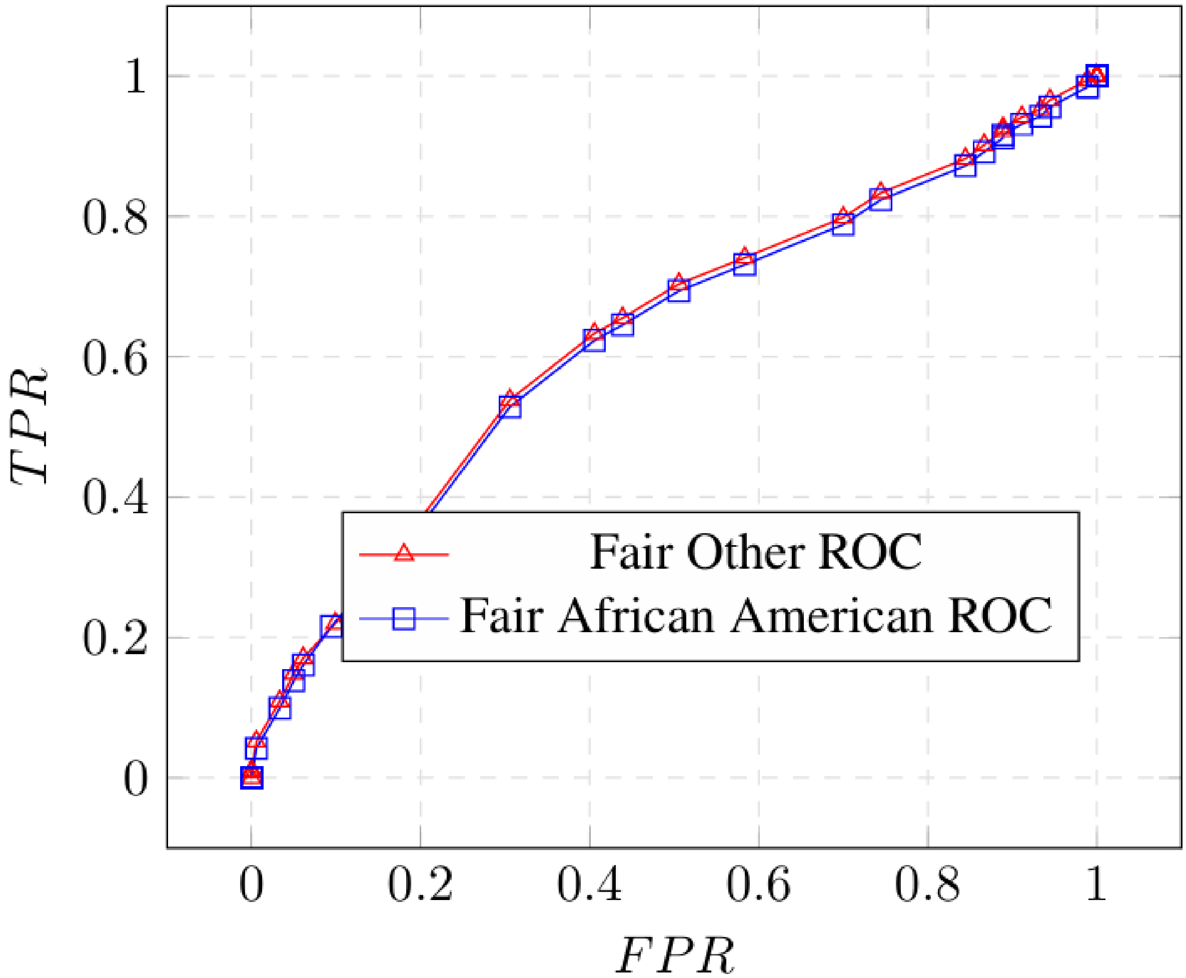

Figure 48: (Fair $\varepsilon_1 = 0.01$) FNNC-FROC ROCs for COMPAS Dataset

```
    if C_ul < 0 or C_ur < 0 or C_dn < 0:
        print("Error: Negative coefficient.")
        return -1

    # Assert that C_ul + C_ur + C_dn = 1
    # print(C_ul + C_ur + C_dn)
    assert C_ul + C_ur + C_dn - 1 < 0.00001

    # If C_ul + C_ur + C_dn != 1, then C_ul = 1 - C_ur - C_dn


    # Assert that C_ul*ul + C_ur*ur + C_dn*dn = p
    # print(C_ul*ul + C_ur*ur + C_dn*dn , p)
    assert C_ul*ul[0] + C_ur*ur[0] + C_dn*dn[0] - p[0] < 0.00001
    assert C_ul*ul[1] + C_ur*ur[1] + C_dn*dn[1] - p[1] < 0.00001

    # print(C_ul + C_ur + C_dn)


    return (C_ul, C_ur, C_dn)
```

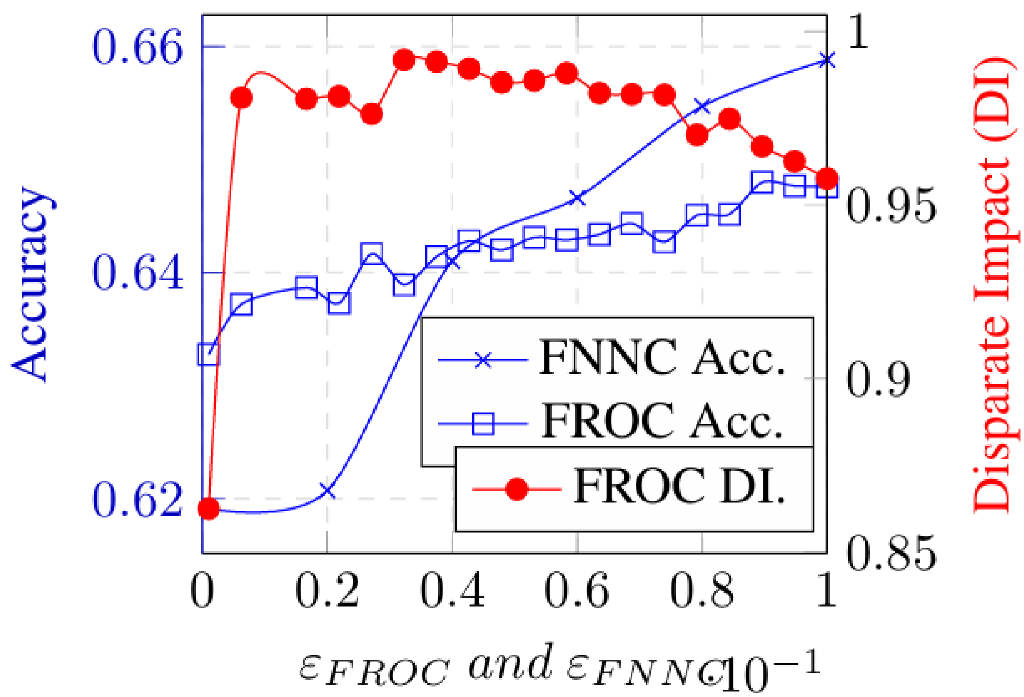

Figure 49: FNNC-FROC Accuracy vs. ε_1 (COMPAS)

```python


# Test the returnCoeff function
ul = np.array([5, 5])
ur = np.array([8, 20])
dn = np.array([6, 6])

p = np.array([6, 7])

print(returnCoeff(ul, ur, dn, p))

def buildClassifier( ROC_up , Probs_up , point , y_test_up):
    # x = point[0] , y = point[1]
    x = point[0]
    y = point[1]

    # Now, we find the interval in ROC_up[0] in which x lies.
    interval = findInterval( ROC_up[0] , x )

    # Now, thresholds = np.linspace(0, 1, 1000)
    thresholds = np.linspace(0, 1, len(ROC_up[0]))

    # Create a classifier output using threshold = thresholds[interval]
    classifier_output = np.zeros( Probs_up.shape[0] )
    classifier_output[ Probs_up >= thresholds[interval] ] = 1

    # Find the FPR and TPR of the classifier_output
    FPR = np.sum( classifier_output * (1 - y_test_up) ) / np.sum( 1 - y_test_up )
    TPR = np.sum( classifier_output * y_test_up ) / np.sum( y_test_up )

    # Assert that FPR == ROC_up[0][interval] and TPR == ROC_up[1][interval]
```

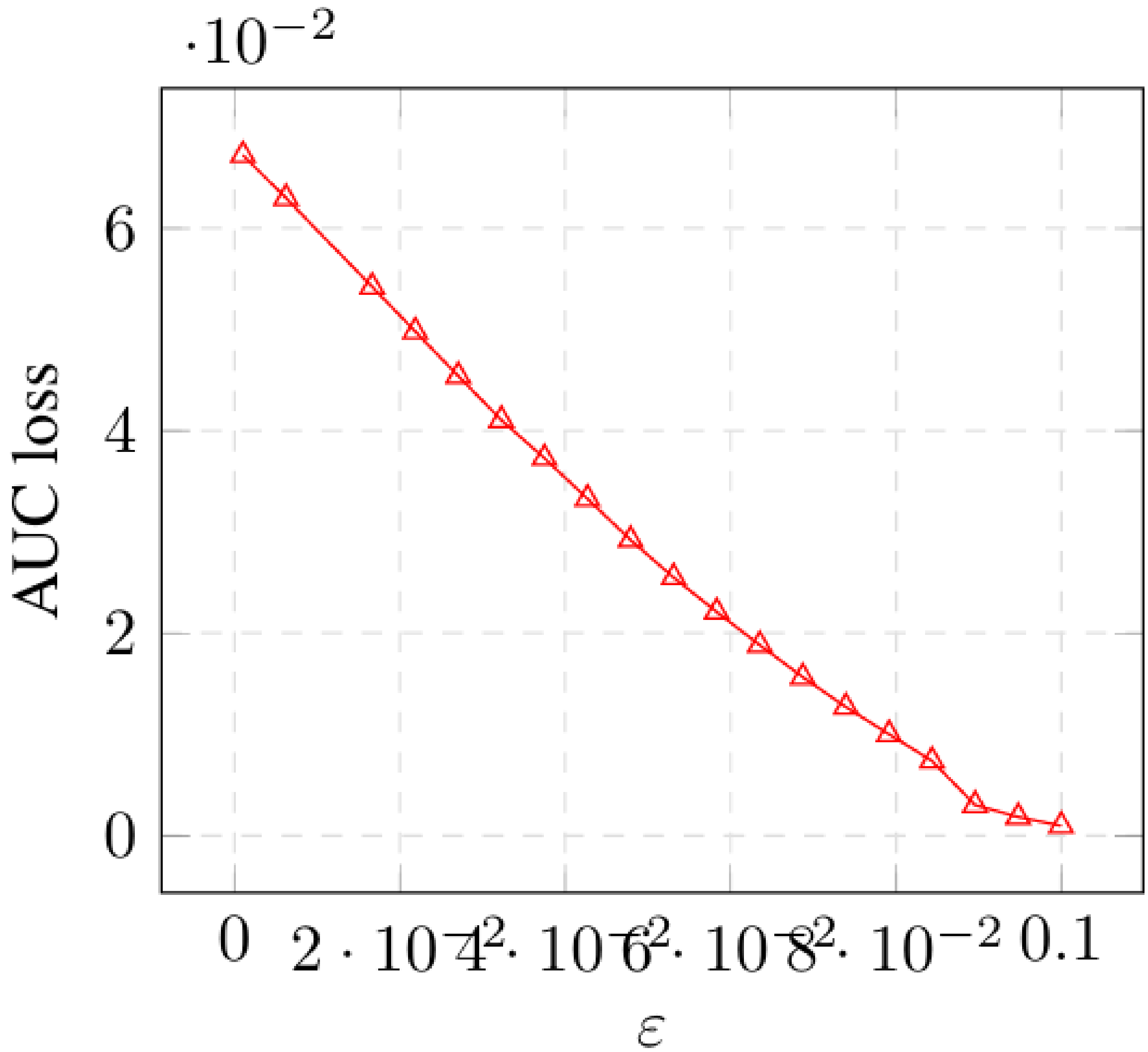

Figure 50: FNNC-FROC AUC loss vs. ε_1 (COMPAS)

```
    # print(FPR, TPR)
    # print(ROC_up[0][interval], ROC_up[1][interval])
    assert FPR == ROC_up[0][interval]
    assert TPR == ROC_up[1][interval]

    ul = np.array([ROC_up[0][interval], ROC_up[1][interval]])
    ur = np.array([ROC_up[0][interval + 1], ROC_up[1][interval + 1]])
    dn = np.array([x,x])
    p = np.array([x,y])

    C_ul , C_ur , C_dn = returnCoeff( ul , ur , dn , p )

    # print(C_ul, C_ur, C_dn)

    # Now, we create the classifier output for threshold = thresholds[interval + 1]
    classifier_output1 = np.zeros( Probs_up.shape[0] )
    classifier_output1[ Probs_up >= thresholds[interval + 1] ] = 1

    # Now, we create a random array with 1 with probability x and 0 with probability 1 - x
```

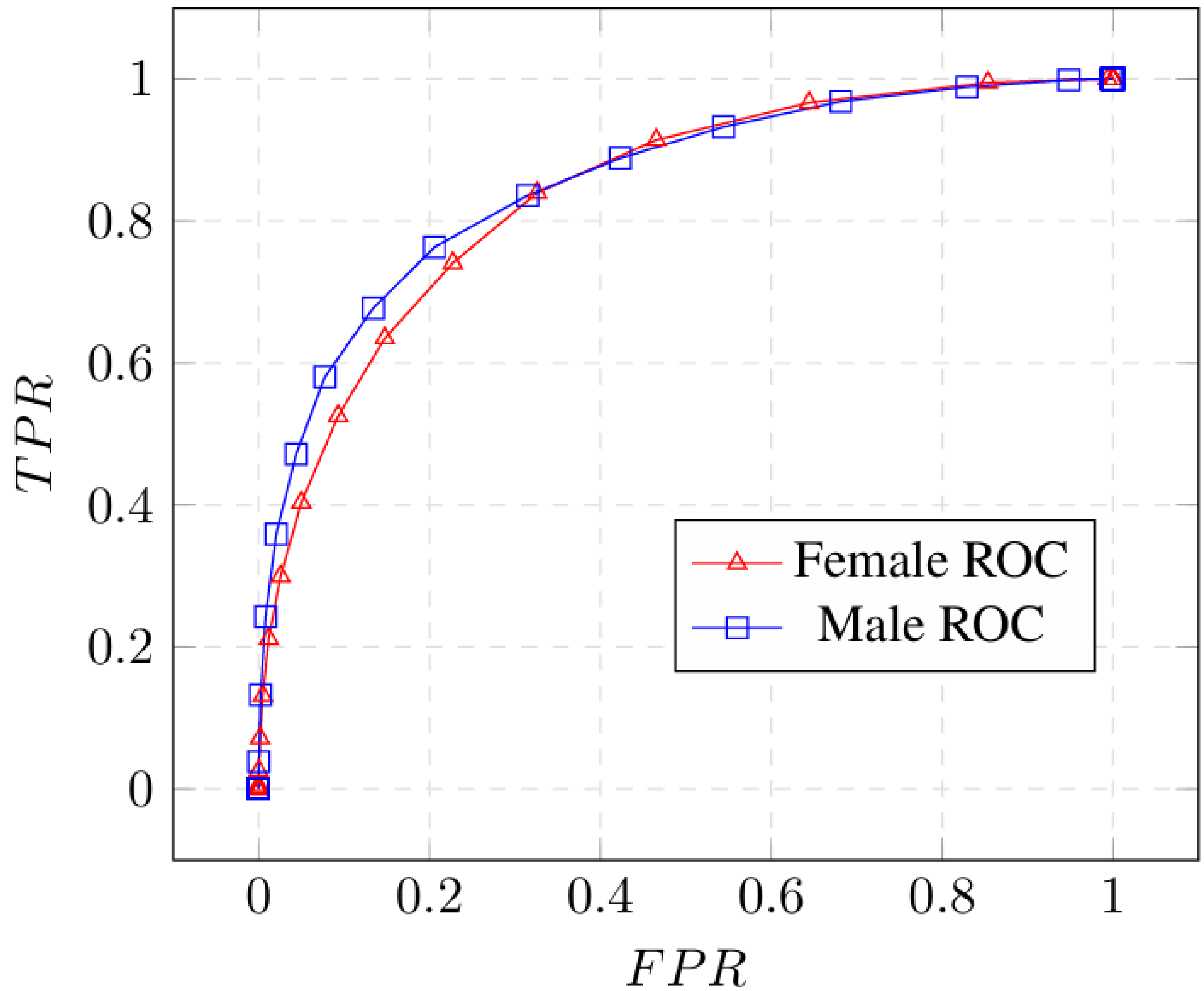

Figure 51: ResNet Baseline ROCs for CelebA Dataset

```
    rand_array = np.random.choice(2, Probs_up.shape[0], p=[1-x, x])
    # print(rand_array)

    # Check if the random array FPR and TPR are equal to x and x
    FPR = np.sum( rand_array * (1 - y_test_up) ) / np.sum( 1 - y_test_up )
    TPR = np.sum( rand_array * y_test_up ) / np.sum( y_test_up )

    # Assert that FPR == x and TPR == x
    # print(FPR, TPR , x)
    # assert FPR == x
    # assert TPR == x


    # Now, we create the final classifier output
    final_classifier_output = np.zeros( Probs_up.shape[0] )

    for i in range( Probs_up.shape[0] ):
        flag = 0
        # Create a random number that takes 0 with probability C_ul, 1 with
probability C_ur and 2 with probability C_dn
        rand_num = np.random.choice(3, 1, p=[C_ul, C_ur, C_dn])
```

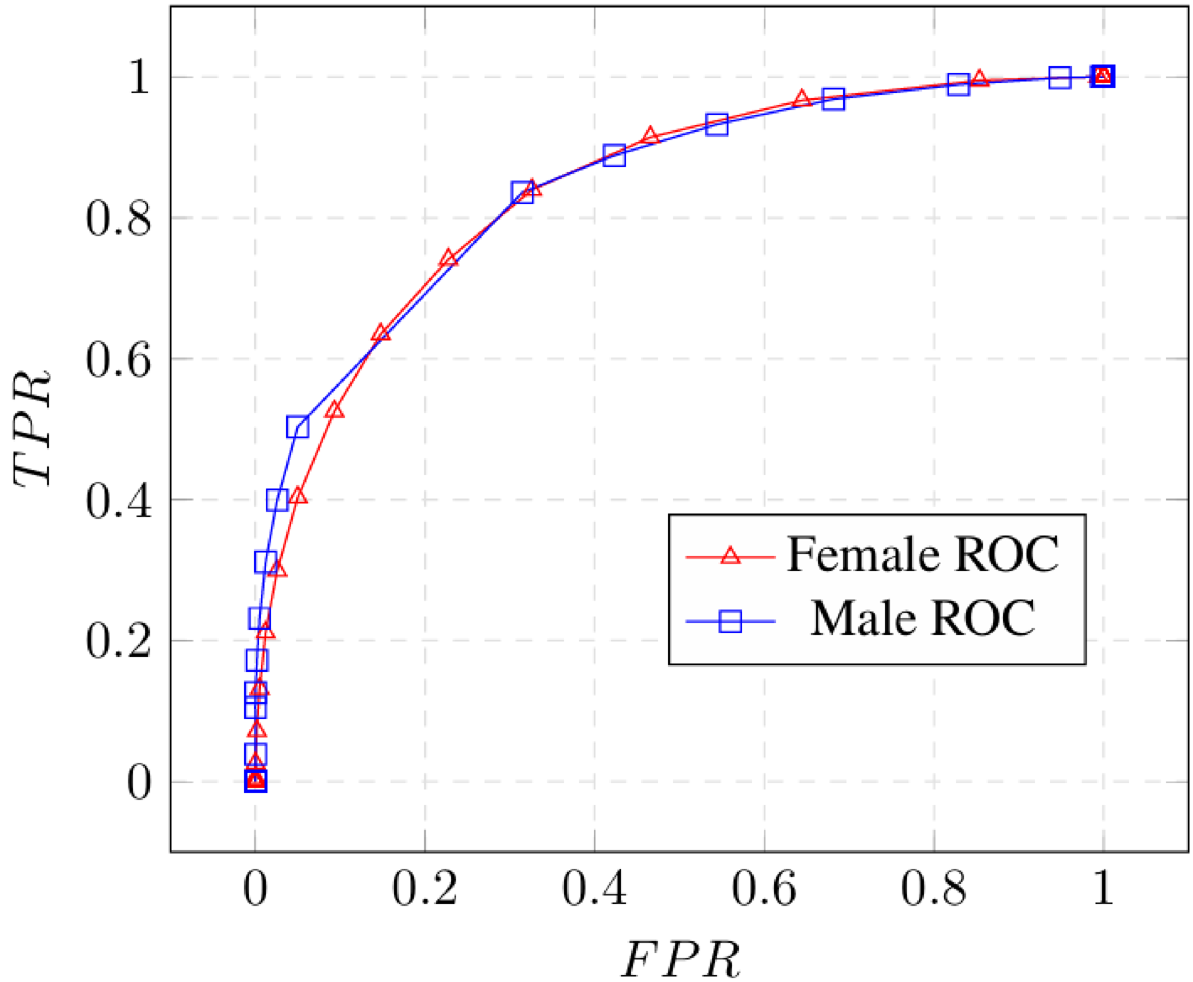

Figure 52: (Fair $\varepsilon_1 = 0.01$) ResNet-FROC ROCs for CelebA Dataset

```
        if rand_num == 0:
            final_classifier_output[i] = classifier_output[i]
            flag = 1
        elif rand_num == 1:
            final_classifier_output[i] = classifier_output1[i]
            flag = 1
        else:
            final_classifier_output[i] = rand_array[i]
            flag = 1

        # Assert that flag == 1
        assert flag == 1

    # Now, we find the FPR and TPR of the final_classifier_output
    FPR = np.sum( final_classifier_output * (1 - y_test_up) ) / np.sum( 1 -
y_test_up )
    TPR = np.sum( final_classifier_output * y_test_up ) / np.sum( y_test_up )

    # Assert that FPR == x and TPR == y
    # print(FPR, TPR)
    # print(x, y)
    assert FPR - x < 0.1
```

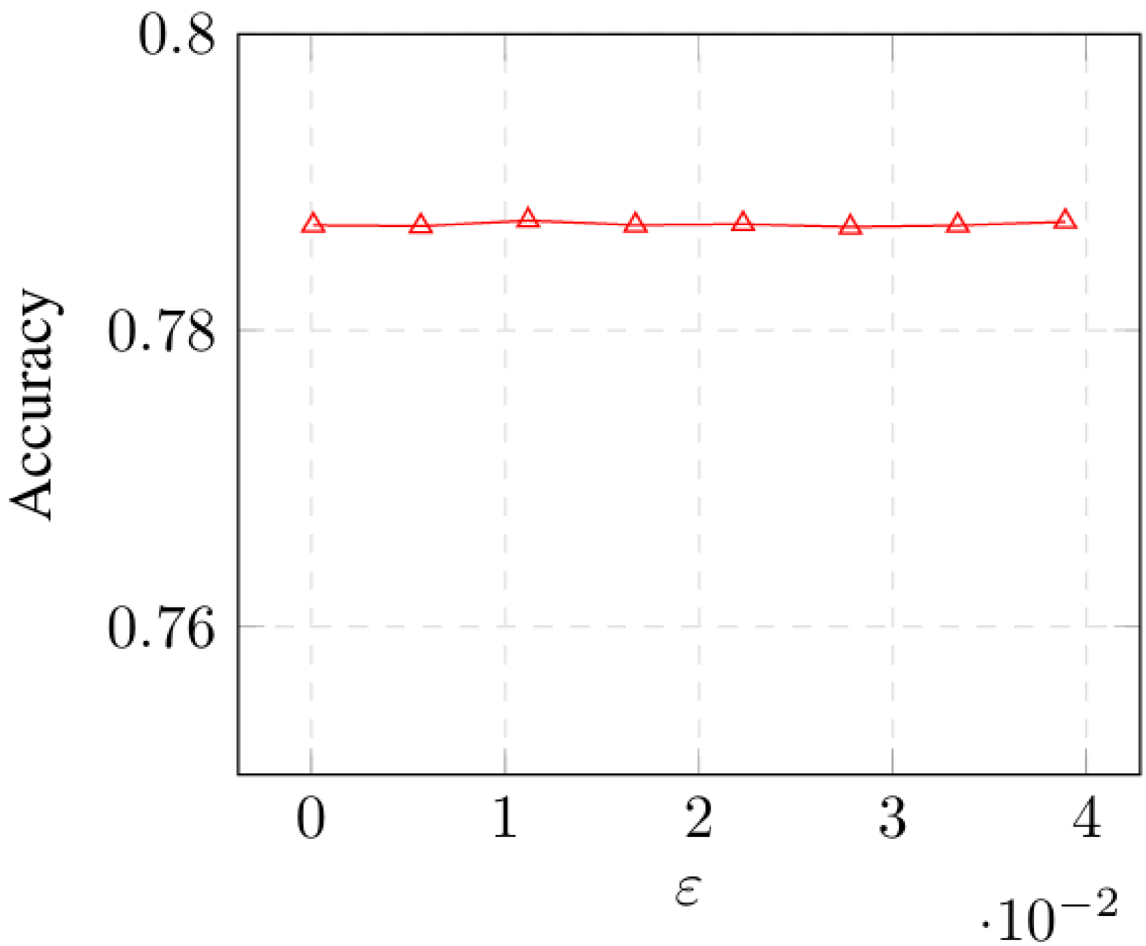

Figure 53: ResNEt-FROC Accuracy vs. ε_1 (CelebA)

```
    assert TPR - y < 0.1

    return final_classifier_output



# Let us now test the buildClassifier function
j = 60
print(Female_FROC[0][j] , Female_FROC[1][j])
print(Female_ROC[0][j] , Female_ROC[1][j])
vec = buildClassifier(Female_ROC , Female_prob[:, 1] , [Female_FROC[0][j] ,
    Female_FROC[1][j]] , Female_y_test)

def isOne( vector ):
    # vector is a vector of 0s and 1s
    # If all the elements of the vector are 1, then return 1
    # Else, return 0
    for i in range( vector.shape[0] ):
        if vector[i] != 1:
            return 0
    return 1

```

```
isOne(vec)
```

References

[1] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018.

[2] Allen N. Berger, W. Scott Frame, and Nathan H. Miller. Credit scoring and the availability, price, and risk of small business credit. *Journal of Money, Credit and Banking*, 37(2):191–222, 2005.

[3] Julia Angwin, Jeff Larson, Lauren Kirchner, and Surya Mattu. Machine bias, May 2016.

[4] Jeffrey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women, Oct 2018.

[5] P. J. Bickel, E. A. Hammel, and J. W. O'Connell. Sex bias in graduate admissions: Data from berkeley. *Science*, 187(4175):398–404, 1975.

[6] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[7] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226, New York, NY, USA, 2012. Association for Computing Machinery.

[8] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *International Conference on Machine Learning*, pages 3384–3393. PMLR, 2018.

[9] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 259–268, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[11] Manisha Padala and Sujit Gujar. Fnnc: Achieving fairness through neural networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021.

[12] D. Sleeman, M. Rissakis, S. Craw, N. Graner, and S. Sharma. Consultant-2: Pre- and post-processing of machine learning applications. *Int. J. Hum.-Comput. Stud.*, 43(1):43–63, jul 1995.

[13] Preetam Nandy, Cyrus DiCiccio, Divya Venugopalan, Heloise Logan, Kinjal Basu, and Noureddine El Karoui. Achieving fairness via post-processing in web-scale recommender systems. *2022 ACM Conference on Fairness, Accountability, and Transparency*, Jun 2022.

[14] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[15] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, 2017. PMID: 28632438.

[16] Jin Huang and Charles X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Trans. on Knowl. and Data Eng.*, 17(3):299–310, mar 2005.

[17] Stéphan Clémençon, Gábor Lugosi, and Nicolas Vayatis. Ranking and empirical minimization of u-statistics. *The Annals of Statistics*, 36(2):844–874, 2008.

[18] Meike Zehlike, Ke Yang, and Julia Stoyanovich. Fairness in ranking: A survey, 2021.

[19] Solon Barocas and Andrew D. Selbst. Big data's disparate impact. *California Law Review*, 104(3):671–732, 2016.

[20] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3323–3331, Red Hook, NY, USA, 2016. Curran Associates Inc.

[21] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the International Workshop on Software Fairness*, FairWare '18, page 1–7, New York, NY, USA, 2018. Association for Computing Machinery.

[22] Sruthi Gorantla, Amit Deshpande, and Anand Louis. On the problem of underranking in group-fair ranking. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3777–3787. PMLR, 18–24 Jul 2021.

[23] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. Fairness in recommendation ranking through pairwise comparisons. *CoRR*, abs/1903.00780, 2019.

[24] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 491–500, New York, NY, USA, 2019. Association for Computing Machinery.

[25] Nathan Kallus and Angela Zhou. *The Fairness of Risk Scores beyond Classification: Bipartite Ranking and the XAUC Metric*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[26] Zhenhuan Yang, Yan Lok Ko, Kush R Varshney, and Yiming Ying. Minimax auc fairness: Efficient algorithm with provable convergence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11909–11917, 2023.

[27] Robin Vogel, Aurélien Bellet, and St'ephan Cl'emenccon. Learning fair scoring functions: Bipartite ranking under roc-based fairness constraints. In *AISTATS*, 2021.

[28] Mingliang Chen and Min Wu. Towards threshold invariant fair classification. In *Conference on Uncertainty in Artificial Intelligence*, pages 560–569. PMLR, 2020.

[29] Dennis Wei, Karthikeyan Natesan Ramamurthy, and Flavio Calmon. Optimized score transformation for fair classification. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1673–1683. PMLR, 26–28 Aug 2020.

[30] Sen Cui, Weishen Pan, Changshui Zhang, and Fei Wang. Towards model-agnostic post-hoc adjustment for balancing ranking fairness and algorithm utility. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 207–217, New York, NY, USA, 2021. Association for Computing Machinery.

[31] Han Zhao. Fair and optimal prediction via post-processing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 22686–22686, 2024.

[32] Alexandru Tifrea, Preethi Lahoti, Ben Packer, Yoni Halpern, Ahmad Beirami, and Flavien Prost. Frappé: A group fairness framework for post-processing everything. In *Forty-first International Conference on Machine Learning*.

[33] André F Cruz and Moritz Hardt. Unprocessing seven years of algorithmic fairness. *arXiv preprint arXiv:2306.07261*, 2023.

[34] Taeuk Jang, Pengyi Shi, and Xiaoqian Wang. Group-aware threshold adaptation for fair classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6988–6995, 2022.

[35] Alan Mishler, Edward H Kennedy, and Alexandra Chouldechova. Fairness in risk assessment instruments: Post-processing to achieve counterfactual equalized odds. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 386–400, 2021.

[36] Foster J. Provost. Machine learning from imbalanced data sets 101 extended. 2000.

[37] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, 2006.

[38] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.

[39] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019.

[40] Keith Kendig. Is a 2000-year-old formula still keeping some secrets? *The American Mathematical Monthly*, 107(5):402–415, 2000.

[41] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[42] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. propublica, 23 may, 2016.

[43] Wael Alghamdi, Hsiang Hsu, Haewon Jeong, Hao Wang, Peter Michalak, Shahab Asoodeh, and Flavio Calmon. Beyond adult and compas: Fair multi-class prediction via information projection. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 38747–38760. Curran Associates, Inc., 2022.