Measuring Dynamic Fairness Metrics in Data Streams

Jinyang Li
University of Michigan
jinyli@umich.edu

H. V. Jagadish
University of Michigan
jag@umich.edu

ABSTRACT

The increasing deployment of machine learning (ML) algorithms in critical decision-making processes has raised concerns regarding their potential biases and the consequent unjust treatment of specific demographic groups. Despite the extensive studies on fairness definitions and measurement techniques to assess different facets of algorithms and datasets, current methods largely overlook the dynamic, ever-changing nature of real-world data which makes static fairness assessments insufficient. This paper introduces a novel approach for measuring dynamic group fairness metrics for classification in streaming data by applying a time decay to traditional fairness metrics to account for changes over time. We propose novel data structures and algorithms to efficiently monitor such dynamic fairness, offering a real-time, continuously evolving view of fairness. Experiments demonstrate the superiority of our approach over static methods in terms of stability, smoothness, and resistance to occasional fluctuations in the measurement results.

PVLDB Reference Format:
Jinyang Li and H. V. Jagadish. Measuring Dynamic Fairness Metrics in Data Streams. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

Recent years have seen increasing applications of machine learning (ML) algorithms in decision-making processes with significant real-life impact. This surge raises concerns about the potential biases these ML models might exhibit, leading to possible unjust treatment of particular demographic groups. Instances of such bias have been documented across multiple domains, including loan approvals [1], hiring processes [2], and recidivism predictions [3]. To address this issue, the research community has proposed various fairness definitions to measure different facets of ML models and datasets [4–8], alongside extensive studies on bias measurement and mitigation techniques [9–13]. However, these fairness measurement approaches do not consider changes in a data set, a factor that is prevalent in the real world.

The world evolves, data change, and there is a great deal of work on detecting and managing data distribution shifts over time. Many learning algorithms support a model that evolves over time to account for non-stationary environments, in varied contexts such as social networks, web mining, financial data, user modeling, etc. As models and data evolve, so does their fairness status. In fact, it has been shown that static fairness approaches are inadequate in situations such as college admission [14], the labor market [15], and credit lending [16], etc. Despite extensive research on developing fairness-aware models in non-stationary environments, there is a notable absence of fairness evaluation methods designed for such contexts. To further motivate the need for dynamic fairness measurements, consider the following example based on the extensively studied COMPAS recidivism prediction algorithm, which faced criticism for alleged discrimination against Black individuals by over-predicting their recidivism rates [3, 17, 18].

		Jan.	Feb.	Mar.	Apr.	May	Jun.	**Total**
Black	number	105	100	95	90	90	80	**560**
	ratio	24%	24%	23%	21%	21%	16%	**21%**
	FP	21	20	17	10	16	14	**98**
	FN	10	10	9	10	10	8	**57**
	TP	35	34	38	37	38	36	**218**
	TN	39	36	31	33	26	22	**187**
	FPR	0.35	0.36	0.35	0.23	0.38	0.39	**0.34**
White	...							
...								
All races	FP	60	58	55	72	65	68	**378**
	TN	170	174	174	216	217	241	**1192**
	number	430	420	410	445	435	500	**2640**
	FPR	0.26	0.25	0.24	0.25	0.23	0.22	**0.24**

Table 1: Monthly recidivism prediction data for a city (January-June), with six-month aggregation in the last column. Positive predictions indicate that a defendant is predicated to re-offend; negative predictions suggest the opposite.

Example 1.1. Table 1 presents the outcomes of a recidivism prediction algorithm applied to defendant data from a city, sampled monthly from January to June. Local authorities seek to understand the demographic representation within this dataset, and monitor which racial group(s) lacks representation. Given past criticism suggesting racial bias against Black defendants in such prediction models, we take this demographic as an example.

To assess whether Black defendants are sufficiently represented, an analyst calculates the percentages of defendants from different racial backgrounds relative to the entire sample pool. A representation below 20% is deemed insufficient. Using the sliding window strategy to examine data from each month reveals that Black defendants maintain adequate representation until a notable decline in June, suggesting a need for close monitoring and possibly corrective actions in subsequent months. However, this under-representation is only flagged when examining the data for June alone or when aggregating data for May and June, which shows a representation

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

of 18%. A three-month window from April to June, or an analysis aggregating all the monthly data, fails to capture this phenomenon. The analyst notes that the recent data is important to reflect the current situation However, exclusively focusing on the most recent data could lead to false alarms or overlook consistent patterns due to sporadic outliers.

To make the discussion more concrete, suppose that local authorities look for any racial group that exhibits an excessively high False Positive Rate (FPR). The FPR is calculated as the ratio of false positives to the sum of false positives and true negatives: $FPR = \frac{FP}{FP+TN}$. An FPR for a group equal to or greater than 0.35 is considered unacceptably high. These high FPR values amongst Black defendants are marked in red, with the remainder in blue. The analyst observes that, with the exception of April, the FPR for Black defendants consistently exceeds the 0.35 threshold. This pattern suggests a persistent bias against Black defendants, with April being an anomaly, whether due to unusual circumstances that month or simply on account of statistical variation. This finding challenges the effectiveness of traditional fairness assessments, which might provide a skewed view by either focusing on a specific month or aggregating data over several months.

The authorities wish to continually monitor the racial fairness of their algorithm in real-time to reflect the current situation, while ignoring isolated, one-off anomalies. From the above analyses, the analyst realizes that focusing solely on specific time frames does not effectively address the ever-shifting nature of fairness, like the sliding window approach or the aggregation method. Inspired by time decay aggregation techniques used in data stream analysis, she decides to use a similar approach, aggregating six-month data while assigning greater weights to recent months. Utilizing the exponential decay, one of the most commonly used decay function, with weights defined as α^t (where $t = 0$ for June and $t = 5$ for January, and $0 < \alpha < 1$), the time decay representation of FPR can be calculated. For instance, with $\alpha = 0.5$, at the end of June, the time decay total count of Black defendants over six months is $80\alpha^0 + 90\alpha^1 + 90\alpha^2 + 95\alpha^3 + 100\alpha^4 + 105\alpha^5 = 168.91$, and similarly, the total for all defendants is 919.69. This results in a Black defendant ratio of 18.37%, below the 20% threshold. This is mainly influenced by June's data which has a large effect on the time decay result. Future results, like those of July, will depend on whether this declining trend continues. Similarly, the time decay FPR for Black defendants is 0.37, which, even with the fluctuation in April, still confirms the disproportionate false positive rate among Black defendants. □

In this paper, we study the problem of measuring dynamic group fairness metrics for classification on streaming data with a time decay. To the best of our knowledge, this is the first work studying fairness measurements in data streams. The literature presents a variety of time decay models for processing streaming data, with distinctions primarily based on how the weight of an item is influenced by its age, such as exponential or polynomial decay mechanisms highlighted in [19]. We apply exponential time decay to fairness measurement in streaming data. Data is chronologically segmented into distinct windows based on its generation or processing timestamp. Each data segment within these windows is assigned a weight using an exponential time decay function, a standard in data stream analysis. This function ensures that the most recent data maintains a weight of 1, while the weight of considerably older data gradually diminishes towards 0. Using this framework, we re-evaluate conventional fairness metrics, for a real-time, continuously evolving perspective of fairness.

The adoption of a time-decay methodology facilitates the generation of smoother fairness value trajectories over time, as opposed to the abrupt changes seen with traditional techniques. This smoothness is essential for their ability to provide consistent interpretations and more accurate conclusions in tracking data evolution or system states at any point in the history, primarily due to two advantages. First, while both smoother and jittery curves can reveal the overarching trend within the data upon inspection, smoother curves offer a clearer representation of the primary trend within the data, making the analysis more straightforward. Second, smoother curves reduces the impact of short-term fluctuations that could otherwise lead to false alarms or incorrect conclusions. When data points show high variability, it's challenging to tell if a point reflects a short-lived anomaly or a significant trend, leading to potential misinterpretations. Thus, smooth curves offers a more reliable and stable representation of data trends.

Given the rapid and frequent change in many real-world applications, it is important to be able to measure fairness in real-time. For instance, in social media, the landscape is highly dynamic with various trending topics emerging and fading within a single day. Each of these trends can reveal different facets of the recommendation algorithm's performance and biases. But real-time calculation of fairness metrics can be computationally expensive, since aggregates may need to be computed over the entire data set each time. We propose to tackle this challenge by incrementally maintaining DF-Monitor, a novel data structure which keeps track of the real-time fairness status of protected groups.

We make the following contributions in this paper:

(1) We define Dynamic Fairness Problem (Section 2), which measures the fairness metrics of ML classifiers for data streams by applying a time decay to traditional fairness definitions. To our knowledge, this is the first paper which studies fairness measurement from a dynamic point of view for data streams.
(2) We develop novel data structures and algorithms (Section 3 and Section 4) to monitor dynamic fairness metrics of protected groups in real time.
(3) We conduct empirical studies on actual streaming data to demonstrate that, compared to traditional static approach, our dynamic approach to fairness assessment yields results that are both stable and smooth, accurately reflecting the present situations without being highly affected by occasional fluctuations (Section 5). The experiments also show that our algorithms are time-efficient and that the optimizations helps enhance the space efficiency.

The rest of the paper is organized as follows. In Section 2, we formally define the Dynamic Fairness Problem, the problem of measuring dynamic time decay fairness and report unfairness in data streams in real time. Our solutions is presented in Section 3 and Section 4 for different applications. We describe our experimental evaluation in Section 5, overview the related work in Section 6, and conclude in Section 7.

Notation	Description
p	pattern/group
t	tuple
P	protected groups
P_t	set of patterns monitored by DFMonitor and satisfied by tuple t
$\overline{P_t}$	set of patterns monitored by DFMonitor but dissatisfied by tuple t
w	window, either time-based or count-based
α	rate of time decay
α^w	time decay weight in window w
D	dataset / data stream
D^w	set of data appearing in window w
D_p^w	set of data appearing in w satisfying p
$C(p), CR(p)$	coverage, and coverage ratio of p
τ	threshold of coverage ratio
FP^w, TP^w, FN^w, TN^w	set of false positives, true positives, false negatives, true negatives in window w
$FP_p^w, TP_p^w, FN_p^w, TN_p^w$	set of false positives, true positives, false negatives, true negatives in window w satisfying p
$FPR(p), TPR(p)$ $FNR(p), TNR(p)$	false positive rate, true positive rate, false negative rate, and true negative rate of p
$UF(p)$	a binary flag indicating unfairness detected w.r.t. p
$\Delta(p)$	how far it is from flipping $UF(p)$
$counter(p)$	the counter of p in data structure

Table 2: Notation table

2 PROBLEM DEFINITION

In this section, we introduce the notion of dynamic fairness in time series. We study the dynamic fairness measurement of a data stream processed by a machine learning classifier. The data stream D consists of tuples, which are fed to an ML classifier. D is segmented into windows $t = 0, 1, 2\dots$ chronologically, depending on when a tuple is processed. As mentioned in Section 1, both the ML classifier and the distribution of D may undergo changes over time. The nature of these changes is not predetermined by the definition of dynamic fairness metrics. Without loss of generality, we assume the tuples are represented using a single relational database, and that the relation's attribute values used for group definitions are categorical. Attributes with continuous values are converted into categories through bucketization, a standard method for data summarization. Groups within the data are defined using patterns, which are specific value assignments to attributes, according to [20].

Definition 2.1 (Pattern). A pattern p in a dataset D with categorical attributes $\mathcal{A} = A_1, \dots, A_n$ is defined by a set of conditions $A_{i_1} = a_1, \dots, A_{i_k} = a_k$, where each A_{i_j} is an attribute in $\mathcal{A}$ and a_j belongs to its active domain. A tuple t in D satisfies pattern p if it meets all conditions in p. The subset of D that satisfies p, denoted as D_p, is used to calculate p's *size* or *coverage* ($|D_p|$) and its *proportion* or *coverage ratio* ($\frac{|D_p|}{|D|}$).

Example 2.2. Consider a demographic dataset D with attributes including race, gender, and age. A pattern $p = Race = Black$ identifies all Black individuals in D. If a tuple in D is $\{Race = Black, Gender = Female, Age = 60+\}$, it satisfies p. For instance, if in January, $D_{\{Race=Black\}}$ comprises 105 tuples out of a total of 430, the coverage ratio of p is 24%. □

To prioritize recent data in the data stream, we apply a time decay. There can be multiple choices for the decay function. We choose exponential decay [21], the most commonly used. We segment data chronologically into distinct windows, and assign a time decay weight to each window. A window covers items seen within a specified duration (N time units), which is similar to the sliding-window model by [22].

Definition 2.3 (Time decay weight). Consider a data stream D within windows $w = 0, 1, 2\dots$, where $w = 0$ is the current window and larger numbers refer to older windows. The time decay weight for data in window w is α^w, with α being a decay rate parameter where $0 < \alpha < 1$.

By this exponential time decay, data in the current window have a weight of $\alpha^0 = 1$, and as we progress to older windows, the weight decreases exponentially and asymptotically approaches zero, as expressed by $\lim_{w\to+\infty} \alpha^w = 0$.

Example 2.4. Consider Example 1.1 with data in Table 1, where data gets segmented monthly. Each month's data is assigned a weight of α^w, where $w = 0$ is the latest month, June, and $w = 5$ is the oldest, January. □

Based on the above definitions from previous works, we define the fairness metrics with time decay. For the data arriving in window w that satisfies pattern p, denoted as D_p^w, the time decay total size is $C(p) = \sum_{w=0}^{W} |D_p^w|\alpha^w$, with W being the window count and notation D^w representing the entire set of all data appearing in window w. The time decay coverage ratio ($CR(p)$) compares the weighted size of group p to the total weighted data size, offering a temporal perspective on data representation:

$$CR(p) = \frac{C(p)}{C(D)} = \frac{\sum_{w=0}^{W} |D_p^w|\alpha^w}{\sum_{w=0}^{W} |D^w|\alpha^w} \quad (1)$$

Definition 2.5 (Biased coverage ratio). Given a data stream D and coverage threshold τ, where $0 < \tau < 1$, a pattern p is said to exhibit a biased coverage ratio if and only if $CR(p) \le \tau$, indicating potential bias in data representation over time.

Example 2.6. Revisiting our ongoing example, when the analyst applies a time decay with $\alpha = 0.5$, the time decay total number of black defendants in all six months is $80\alpha^0 + 90\alpha^1 + 90\alpha^2 + 95\alpha^3 + 100\alpha^4 + 105\alpha^5 = 168.91$. Similarly, the total for all defendants amounts to 919.69, so the time decay coverage ratio of black defendants is 18.37, below the threshold of $\tau = 20\%$. □

Similarly we account for the other fairness metrics of a ML classifier on streaming data by applying a time decay. In particular, we focus on fairness definitions based on the model's binary predictions and their relationship to binary outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). Each tuple t in the data stream D falls into one of these categories based on its predicted label and ground truth. Many common fairness definitions require some computational results with respect to these four predicative outcomes should be similar across different groups. Here we provide a overview of a few common fairness definitions. For more details, please refer to [4].

- **False positive error rate balance:** False positive rate $FPR = \frac{FP}{FP+TN}$ should be similar for all groups
- **False negative error rate balance:** False negative rate $FNR = \frac{FN}{FN+TP}$ should be similar for all groups
- **Predictive parity**: The fraction of correct positive prediction $\frac{TP}{TP+FP}$ should be similar for all groups
- **Overall accuracy equality**: The prediction accuracy should be similar for all groups
- **Equalized odds**: The ratio of errors that the classifier makes should be similar for all groups

These fairness metrics can also be transitioned to dynamic version by applying a time decay weight α^w. In this paper, we exemplify this approach using the false positive rate (FPR) as a case study, although the methodology can be similarly applied to other definitions. For a group of items, FPR measures the probability of subjects in the actual negative class having a positive predicative value, i.e., $FPR = \frac{FP}{FP+TN}$. Let $FP_p^w, TP_p^w, FN_p^w, TN_p^w$ denote the sets of false positives, true positives, false negatives and true negatives in group p within window w, respectively. Then the false positive rate of pattern p in data stream D over windows $w = 0, 1, 2 \ldots W$ with time decay parameter α $(0 < \alpha < 1)$ is

$$FPR(p) = \frac{\sum_{w=0}^{W} |FP_p^w| \alpha^w}{\sum_{w=0}^{W} |FP_p^w| \alpha^w + \sum_{w=0}^{W} |TN_p^w| \alpha^w} \tag{2}$$

Overloading notations, FPR of D is

$$FPR(D) = \frac{\sum_{w=0}^{W} |FP^w| \alpha^w}{\sum_{w=0}^{W} |FP^w| \alpha^w + \sum_{w=0}^{W} |TN^w| \alpha^w} \tag{3}$$

The concept of Biased False Positive Rate (FPR) can be formally defined as follows:

Definition 2.7 (Biased FPR). Given a data stream D and coverage threshold τ, $(0 < \tau < 1)$. We say pattern p shows a biased FPR if and only if $FPR(p) \geq \tau$.

The interpretation of a high or low FPR as indicative of unfair treatment depends on the specifics of the situation at hand. In this paper, for the sake of clarity and without loss of generality, we assume that a high FPR is indicative of a problem, in line with our running example of recidivism prediction.

With this time decay model, traditional definitions of these fairness metrics, e.g., calculating $FPR = \frac{FP}{FP+TN}$ for a portion of data without considering their age, can be seen as a special case with one time window containing all data weighted by $\alpha^0 = 1$. Calculating FPR window-by-window, in our running example, month-by-month, is equivalent to the special case where $\alpha = 0$, and calculating FPR using all observed time series data relates to $\alpha = 1$.

Our approach to computing fairness metrics with time decay, while similar, differs from the sliding-window model [23]. Sliding-window model only considers a window of the most recent items in the data stream, effectively assigning a weight of zero to items outside this window, as they are considered "aged out". In the context of our running example, this would mean considering only the data from the latest month, June. In contrast, our method includes all historical windows, but applies diminishing weights to the data in older windows. This approach ensures that while recent data is prioritized, older information still contributes to the overall assessment of fairness.

We now formally define the Dynamic Fairness Problem to monitor the fairness status of protected groups P in real time. In this paper, "protected groups" refer to the set of all groups subject to fairness assessment. In other words, any groups not identified as protected will not be queried for fairness status.

Problem 2.8. *[Dynamic Fairness Problem] Given a data stream D applied to a ML classifier with in windows $w = 0, 1, \ldots,$ a time decay parameter α, for a set of given protected groups P, with respect to a given fairness metric $\mathcal{F}$ and a fairness threshold τ, for any given group (pattern) $p \in P$, we aim to answer the following two questions about the current status in real time:*

- *$UF_{\mathcal{F}}(p)$: whether p shows any unfairness with respect to $\mathcal{F}$.*
- *$\mathcal{F}(p)$: the value of the time decay fairness metric $\mathcal{F}$ of group p.*

Example 2.9. Revisiting our example with Table 1, traditional fairness assessment methods might only consider recent statistics, such as data from a single month. Applying a time decay shifts this perspective, potentially uncovering fairness issues not immediately apparent. For instance, the False Positive Rates (FPR) initially seems to improve in April, with a time decay FPR of 29.52%, below the 35% threshold. However, time decay reveals that by May and June, FPRs increase to 35.74% and 37.15%, respectively, both exceeding the fairness threshold and signaling ongoing fairness concerns ($UF_{\mathcal{F}}(Race = Black) = True$). This example illustrates that, despite temporary fluctuations, the time decay approach still identifies the persistent historical trend of fairness. It will take a longer period of fluctuations to overcome this history when considering time decay. □

Assumptions. For simplicity, in this paper, we have the following two assumptions. First, we assume that the groups monitored by DFMonitor comprehensively cover the dataset without overlap. For instance, in our running example, we assume the domain of the Race attribute is $Dom(Race) = \{White, Black, Hispanic\}$, meaning these groups collectively represent all possible racial categories present in the data set D, without overlapping members. Under this assumption, the aggregate size of the dataset over time can be calculated using $C(D) = \sum_{p' \in P} counter(p')$. Should this assumption not hold, an additional counter would be required to accurately measure the total dataset size. Second, without loss of generality, it is assumed that each group has the same fairness threshold τ (where $0 < \tau < 1$). This assumption is viable because fairness evaluation for each group is conducted independently, based on its specific circumstances and the predefined threshold. Therefore, whether the thresholds are same or different across different groups does not influence the methodology for addressing this problem.

Following these assumptions, we outline the method for monitoring the coverage ratio in Section 3 and extend the discussion to other fairness metrics in Section 4. The latter focuses on the distinctions from Section 3 as the foundational concepts are similar.

3 ALGORITHMS

In this section, we introduce the data structure DFMonitor (**D**ynamic **F**airness **Monitor**) and its operations designed for monitoring dynamic fairness metrics within time windows as a solution

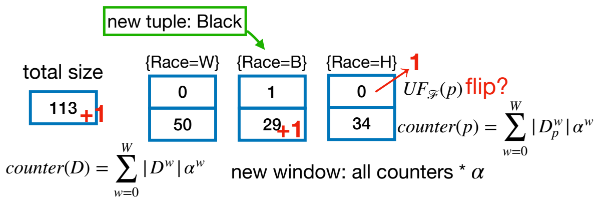

new tuple: Black
total size
113
+1
{Race=W}
0
50
{Race=B}
1
29
+1
{Race=H}
0
34
1
$UF_{\mathcal{F}}(p)$ flip?
$counter(p) = \sum_{w=0}^{W} |D_p^w| \alpha^w$
$counter(D) = \sum_{w=0}^{W} |D^w| \alpha^w$
new window: all counters * α

Figure 1: DFMonitor, basic solution for coverage ratio

to the Dynamic Fairness Problem. Data within a span of N time units is aggregated into a single window. Depending on whether the focus is on monitoring coverage ratio or other fairness metrics like false positive rate, the specific fields of DFMonitor and its operations may vary. We start with the coverage ratio and expand to other metrics in Section 4. This section is structured as follows: We outline DFMonitor's algorithms in Sections 3.1 and 3.2, followed by a correctness proof in Section 3.3.

3.1 Basic Solution

A straightforward method of monitoring if the coverage ratio of any group falls below a predetermined threshold is to keep a counter for each protected group, denoted as $counter(p)$, which reflects its current time decay coverage, i.e., $counter(p) = C(p) = \sum_{w=0}^{W} |D_p^w| \alpha^w$, as shown in Figure 1 where three racial groups are identified as protected: White, Black, and Hispanic. Note that while we use race as an attribute defining protected groups here, the concept of protected groups can extend to different attributes, and the coverage thresholds for these groups are not constrained to uniformity. We use binary indicators $UF_{\mathcal{F}}(p)$ for each group to signal insufficient coverage, and a total dataset counter ($counter(D)$) to record the overall size. Upon the arrival of a new tuple t, both $counter(D)$ and $counter(p)$ are incremented by one for all groups p that t satisfies, as the weight of items in current window is always 1. Then for each group p we need to calculate its coverage ratio to see whether $UF_{\mathcal{F}}(p)$ needs to be flipped. When the system enters a new window, the value in each counter is multiplied by α, while all binary flags remain unchanged. Groups flagged as having insufficient coverage are reported in real time.

Example 3.1. Figure 1 shows a snapshot of basic DFMonitor monitoring coverage ratio of three racial groups: White, Black and Hispanic. With a threshold $\tau = 30\%$, initially, only Black has an insufficient coverage ratio. Then in the same window, upon the arrival of a new data tuple belonging to the Black racial group, both the $counter(Race = Black)$ and $counter(D)$ are incremented by one. Then we need to recalculate the coverage ratio of all protected groups to decide whether to flip the bit, which reveals that the coverage ratio of Hispanic becomes $29.82\% < \tau$, so $UF_{\mathcal{F}}(\{Race = Hispanic\})$ ($\mathcal{F}$ denotes the coverage ratio) should be set to True. □

The algorithm's simplicity and clarity are notable advantages. It not only reports groups with insufficient coverage but also provides the precise values of coverage or coverage ratios for each group, due to the maintenance of time decay counters. However, it is important to recognize that in scenarios where the exact numerical value of coverage is not necessary, both the data structure and algorithm can be refined to enhance efficiency. This becomes especially pertinent for other fairness metrics, such as the False Positive Rate (FPR), which will be discussed in subsequent sections.

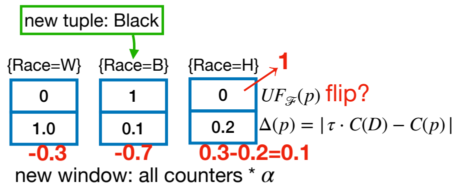

new tuple: Black
{Race=W}
0
1.0
-0.3
{Race=B}
1
0.1
-0.7
{Race=H}
0
0.2
0.3-0.2=0.1
1
$UF_{\mathcal{F}}(p)$ flip?
$\Delta(p) = |\tau \cdot C(D) - C(p)|$
new window: all counters * α

Figure 2: DFMonitor, optimized solution for coverage ratio

3.2 Optimized Solution

In many real-world applications, especially within systems that assess or predict outcomes affecting individuals or groups, stakeholders may only need a binary determination of fairness, for example, fair/unfair or biased/unbiased, rather than the precise quantitative measure of fairness.

For example, in social media content moderation, the fairness of algorithms filtering, promoting and recommending the content is a critical concern [24, 25], particularly because the content visibility is important and can transition to money and opportunity for the creators [26, 27]. The equitable distribution and visibility of content across varied political, cultural, or social spectra are important to ensuring a fair digital environment. Users and regulatory bodies are concerned with whether the moderation process is fair to all viewpoints, requiring a binary understanding of bias, without a focus on the quantitative fairness measurements. For instance, European Union's initiatives like the Digital Services Act [28, 29] and the Digital Markets Act [30] aim to ensure that digital platforms operate transparently and fairly, so that all users, irrespective of their demographics or the nature of their content, have equal opportunities to reach their audiences. Similarly, in healthcare, algorithms that prioritize patient treatment or resource allocation need to ensure fairness across diverse patient demographics, such as age, race, or gender. The healthcare providers and patients are concerned with whether the algorithms are equitable in its prioritization and allocation, rather than the exact score. This binary perspective also extends into other critical areas such as college admission decisions, loan approval systems, recruitment and hiring platforms, etc.

Acknowledging this need, we propose an optimized version of DFMonitor designed for enhanced space efficiency by tracking each group's distance from flipping the binary status of fairness rather than the exact statistic values. This approach provides a simple yes/no answer about the fairness status at a lower cost, but is not able to provide the exact value of any quantitative fairness metric.

Data structure: Figure 2 shows DFMonitor monitoring coverage ratios for protected groups, i.e., groups monitored by DFMonitor: $P = \{Race = White, Race = Black, Race = Hispanic\}$. Unlike the basic model depicted in Figure 1, where each group has a counter, in this optimized model, each group p has a binary indicator $UF_{\mathcal{F}}(p)$ for fairness status and a field $\Delta(p)$. Here, $\mathcal{F}$ represents the coverage ratio, and $\Delta(p)$ calculates the weighted distance needed to alter $UF_{\mathcal{F}}(p)$'s status, using $\Delta(p) = |\tau \cdot C(D) - C(p)|$. In this calculation, τ is a predefined threshold, $C(D)$ is the total coverage of the total dataset, and $C(p)$ is the coverage of group p. This optimized approach eliminates the need for additional counters, offering a more space-efficient solution, because in data streams with relatively balanced distributions, $\Delta(p)$ is potentially much smaller than $counter(p)$. In the most extreme cases, the $\Delta(p)$ value will not exceed $counter(p)$.

Operations: DFMonitor is configured to (1) initialize after the first window, (2) update its fields upon receiving a new tuple, (3) refresh all weighted distances $\Delta(p)$ at the start of each new time window, and (4) provide the real-time fairness status for protected groups. The operations are detailed as follows:

(1) Initialization: In data stream analysis, the initial coverage ratio of a group within early data is not adequately representative until a substantial amount of data is observed in the stream. Therefore, we need to initialize DFMonitor using the data from early stages. For simplicity, we initialize DFMonitor using the statistics collected from the items in the first window. We keep track of the group sizes $C(p)$ during this initialization period by temporarily maintaining counters similar to those in Figure 1. Initially, these counters (as described in the basic model) track the size of each group, starting from zero and increasing by one with each new item, assuming all initial items have a time decay weight of 1. Upon the completion of the first window, DFMonitor transitions to the optimized model in Figure 2 based on these counter values. The coverage ratio $CR(p)$ for any protected group p is derived as $CR(p) = \frac{C(p)}{C(D)} = \frac{counter(p)}{\sum_{p' \in P} counter(p')}$, with $UF_{\mathcal{F}}(p)$ set based on whether $CR(p)$ exceeds a threshold τ. Then the weighted distance $\Delta(p)$ is calculated as $|\tau \cdot C(D) - C(p)|$. This distance intuitively depends on the weighted difference between the group size of p and the size of the entire dataset D, showing how far it is from flipping the binary indicator. After these initial steps, these temporary counters are discarded. DFMonitor only has the components shown in Figure 2, and is maintained in accordance with the operations detailed in the subsequent paragraphs for continuous fairness assessment and data monitoring.

(2) Insert(t): Upon the arrival of a new tuple t in D, DFMonitor updates its relevant fields as outlined in Algorithm 1. For each monitored group p, DFMonitor evaluates whether the bit $UF_{\mathcal{F}}(p)$ requires flipping (line 6, 9, 15, 18). If a flip is necessary, DFMonitor proceeds to invert $UF_{\mathcal{F}}(p)$ and recalculates $\Delta(p)$ (line 12 - 13, line 21 - 22), otherwise, only the update of $\Delta(p)$ is performed. Here we explain the update process for fields of protected groups p satisfying t in detail, outlined from line 5 to line 13 in Algorithm 1. First, if $UF_{\mathcal{F}}(p) = 0$ (indicating previously adequate coverage), the coverage remains sufficient post the addition of the new tuple, moving p further from needing a status change. Given $\Delta(p) = |\tau \cdot C(D) - C(p)| = C(p) - \tau \cdot C(D)$, and both $C(D)$ and $C(p)$ increase by 1 due to the arrival of t, $\Delta(p)$ should be increased by $(1 - \tau)$ (line 6 - 7). Second, if $UF_{\mathcal{F}}(p) = 1$ (signifying prior insufficient coverage), the distance $\Delta(p)$ decreases by $(1 - \tau)$. Should $\Delta(p)$ be too small for the decrease, i.e., $\Delta(p) \leq (1 - \tau)$, indicating the group is close to or at the threshold, $UF_{\mathcal{F}}(p)$ is flipped to reflect the updated status. For groups not satisfying t, $\Delta(p)$'s adjustment is based solely on the increment of $C(D)$ by 1, thus $\Delta(p)$ is changed by τ. Detailed procedures are similar to the aforementioned steps and are omitted here for brevity.

(3) Updating the Time Window: As the system transitions into a new window, the $\Delta(p)$ for each protected group is multiplied by α, in order to account for the time decay of data relevance. This operation adjusts the weighted distance to the fairness threshold

Algorithm 1: Insert(t): insert tuple t to DFMonitor for monitoring coverage ratio

```
1  $P \leftarrow$ the set of protected groups;
2  $P_t \leftarrow \{p \in P | t$ satisfies $p\}$;
3  $\overline{P_t} \leftarrow \{p \in P | t$ dissatisfies $p\}$;
4  $\mathcal{F} \leftarrow$ coverage ratio;
5  for $p \in P_t$ do
6      if $UF_{\mathcal{F}}(p) = 0$ then
7          $\Delta(p) \leftarrow \Delta(p) + (1 - \tau)$;
8      else
9          if $\Delta(p) \geq 1 - \tau$ then
10             $\Delta(p) \leftarrow \Delta(p) - (1 - \tau)$;
11         else
12             $\Delta(p) = (1 - \tau) - \Delta(p)$;
13             $UF_{\mathcal{F}}(p) \leftarrow 0$;
14 for $p \in \overline{P_t}$ do
15     if $UF_{\mathcal{F}} = 1$ then
16         $\Delta(p) \leftarrow \Delta(p) + \tau$;
17     else
18         if $\Delta(p) \geq \tau$ then
19             $\Delta(p) \leftarrow \Delta(p) - \tau$;
20         else
21             $\Delta(p) \leftarrow \tau - \Delta(p)$;
22             $UF_{\mathcal{F}} \leftarrow 1$;
```

in accordance with the passage of time, ensuring that the fairness metrics remain dynamically aligned with the current time window. Other fields within DFMonitor, the binary indicators, remain unchanged during this transition.

(4) Query(p): DFMonitor can instantly respond to inquiries regarding whether a group's coverage ratio meets the established fairness threshold, as per Problem 2.8. One can determine if a group is flagged for insufficient coverage, indicated by $UF_{\mathcal{F}}(p) = True$.

Example 3.2. Figure 2 shows an snapshot of optimized version DFMonitor monitor coverage ratio White, Black, and Hispanic demographic groups. When a new tuple arrives whose race is Black, we need to check and modify some fields according to Algorithm 1. Since $\Delta(\{Race = Black\}) = 1$ which means Black is lacking coverage. According to line 9 - line 10, we need to decrease $\Delta(\{Race = Black\})$ by $1 - \tau = 0.7$, making it 0.3. For Hispanic, $\Delta(\{Race = Hispanic\}) < \tau$ which means Hispanic is at the edge of falling off the coverage ratio threshold and this new tuple makes it flip the flag. Based on line 21 - line 22, $UF_{\mathcal{F}}(\{Race = Hispanic\})$ should be set to True and $\Delta(\{Race = Hispanic\})$ be modified to 0.1. Similarly, for White, according to line 18 - line 19, only $Delta(\{Race = White\})$ needs to decrease 0.3, and the binary indicator remain unchanged. □

3.3 Algorithm Correctness Proof

Now we prove that the above algorithm gives us the correct answer about fairness status. We discuss the three variables $UF_{\mathcal{F}}(p)$, $\Delta(p)$, and $counter(p)$ together so that the correctness of both basic algorithm in Section 3.1 and optimized version in Section 3.2 can be proved. To prove the correctness of the algorithm in question, we first mathematically show why in the insertion algorithm, $\Delta(p)$ is adjusted by $(1-\tau)$ when the new tuple satisfies p, and by τ otherwise. This principle also underpins the logic of our insertion algorithm, whose correctness, and that of window update, we prove next, both by induction. Combining these proofs, we establish the overall correctness of the algorithm.

To begin with, we formally establish that $\Delta(p)$ is adjusted by $(1-\tau)$ upon the arrival of a tuple satisfying p, and by τ otherwise.

Proposition 3.3. *Given $\mathcal{F}$ as the coverage ratio with threshold τ, consider any protected group $p \in P$, with the current fields $UF_{\mathcal{F}}(p), \Delta(p)$, and $counter(p)$. Within the same window, if after processing x tuples satisfying p and y tuples not satisfying p, the bit $UF_{\mathcal{F}}(p)$ needs to flip from 1 to 0, then it must hold that $x(1-\tau) - y\tau > \Delta(p)$. Conversely, if $UF_{\mathcal{F}}(p)$ needs to flip from 0 to 1, then $x(1-\tau) - y\tau \le \Delta(p)$.*

Proof. Because $UF_{\mathcal{F}}(p) = 1$ and $UF_{\mathcal{F}}(p) = 0$ are two symmetric scenarios, we can, without loss of generality, assume that $UF_{\mathcal{F}}(p) = 1$, which implies $CR(p) = \frac{C(p)}{C(D)} \le \tau$. We analyze a scenario within the same time window where, after the arrival of x tuples satisfying p and y not satisfying p, $UF_{\mathcal{F}}(p)$ is due for a flip. This leads to $CR'(p) = \frac{C(p)+x}{C(D)+x+y} > \tau \implies x(1-\tau) - y\tau > \tau \cdot C(D) - C(p)$, confirming the proposition given $\Delta(p) = |\tau \cdot C(D) - C(p)|$. □

To prove the correctness of the insertion algorithm of DFMonitor, i.e., it preserves the integrity of the unfairness status, weight distance, and time decay coverage for each group p, we examine the updates made to $UF_{\mathcal{F}}(p)$, $\Delta(p)$, and $counter(p)$. When a new tuple t arrives, the algorithm 1 processes both the protected groups t satisfies (P_t) and those it does not ($\overline{P_t}$) using the same logic. This implies that proving the algorithm's effectiveness in one case implicitly proves it for the other. Specifically, we aim to show that the updated $\Delta'(p)$ aligns with $|\tau C'(D) - C'(p)|$, which can be confirmed by setting $C'(D) = C(D) + 1$ and $C'(p) = C(p) + 1$. So we have the following proposition.

Proposition 3.4. *Let $\mathcal{F}$ denote the coverage ratio metric. At any time after the initialization of DFMonitor, for any protected group p monitored by DFMonitor, assuming the fields $UF_{\mathcal{F}}(p)$, $\Delta(p)$, and $counter(p)$ accurately reflect the group's unfairness status, weighted distance, and coverage value, the arrival of a new tuple t leads to updated fields $UF'_{\mathcal{F}}(p)$, $\Delta'(p)$, and $counter'(p)$, in accordance with the insertion algorithm. These updated fields continue to accurately represent the respective values for p.*

Proof. Prior to the arrival of tuple t, for group p, the fields hold the values:

$$counter(p) = C(p) = \sum_{w=0}^{W} |D_p^w|\alpha^w,$$

$$\Delta(p) = |\alpha \cdot C(D) - C(p)| = \left|\alpha \cdot \sum_{w=0}^{W-1} |D^w|\alpha^w - \sum_{w=0}^{W-1} |D_p^w|\alpha^w\right|,$$

$$UF_{\mathcal{F}}(p) = \begin{cases} 1 & \text{if } CR(p) = \frac{C(p)}{C(D)} > \tau, \\ 0 & \text{otherwise.} \end{cases}$$

Upon t's arrival, protected groups satisfied by t (P_t) and those not ($\overline{P_t}$) are processed distinctly in Algorithm 1. We will analyze them separately.

(1) For any $\mathrm{p} \in \mathrm{P_t}$ (line 5 - 13):

First we need to increment $counter(p)$ by 1 if the counters are maintained. Then, when $UF_{\mathcal{F}}(p) = 0$ (line 6 - 7), indicating p's coverage without t is already above the threshold, $UF'_{\mathcal{F}}(p)$ still equals to 0. We only need to increase the weighted distance. This implies $CR(p) > \tau$, so $C(p) > \tau C(D)$ and thus $\Delta(p) = C(p) - \tau C(D)$. With t's addition, $C(p)$ and the time decay total size $C(D)$ both increment by 1. The new weighted distance is

$$\begin{aligned}\Delta'(p) &= \left|\tau C'(D) - C'(p)\right| = |\tau(C(D)+1) - (C(p)+1)| \\ &= |\tau C(D) - C(p) + \tau - 1| = C(p) - \tau C(D) + 1 - \tau \\ &= \Delta(p) + (1-\tau)\end{aligned}$$

Conversely, when $UF_{\mathcal{F}}(p) = 1$ (line 11), indicating insufficient coverage without t, we must determine if t's addition sufficiently increases the coverage ratio to flip the bit $UF_{\mathcal{F}}(p) = 1$. If it is not enough to flip the bit, $CR'(p) = \frac{C(p)+1}{C(D)+1} \le \tau$, then $\tau \cdot C(D) - C(p) = \Delta(p) \ge 1 - \tau$ (line 9 - 10). We update the weighted distance to

$$\begin{aligned}\Delta'(p) &= \left|\tau \cdot C'(D) - C'(p)\right| = |\tau(C(D)+1) - (C(p)+1)| \\ &= |\tau C(D) - C(p) - (1-\tau)| = \Delta(p) - (1-\tau)\end{aligned}$$

Similarly, if $\Delta(p) < 1 - \tau$ (line 11 - 13), the coverage ratio becomes sufficient, then $UC'_{\mathcal{F}}(p) = 0$ and $\Delta'(p) = (1-\tau) - \Delta(p)$.

(2) For any $\mathrm{p} \in \overline{\mathrm{P_t}}$ (line 14 - 22):

For groups that t does not satisfy, they are handled in a similar manner to those that t does satisfy, except that $counter(p)$ remains unchanged. The update to $\Delta(p)$ and the assessment for potentially flipping $UF_{\mathcal{F}}(p)$ proceed as previously described, taking into account that a previously sufficient coverage ratio may become insufficient due to the increase of $C(D)$. As the underlying logic and operations closely mirror those for groups satisfied by t, we omit the detailed proof here for brevity. □

Then we need to demonstrate that as DFMonitor transitions into a new time window, the applied operations (multiplying counter value $counter(p)$ and weighted distance $\Delta(p)$ for any protected group p by the factor α, while keeping the unfairness status unchanged) ensure all fields within DFMonitor accurately reflect the values corresponding to the protected groups. We begin by expressing all variables in the form of $C(p) = \sum_{w=0}^{W-1} |D_p^w|\alpha^w$, using the definition of time decay coverage. This approach allows us to analyze the dynamics before (stage 1) and after (stage 2) entering a new window. Since the data associated with window w during stage 2 are essentially the same data in window $w-1$ in stage 1, in stage 2, before the arrival of any new data, the variables $C'(p)$, $UF'_{\mathcal{F}}(p)$, and $\Delta'(p)$ can be expressed by the variables in stage 1, which helps us prove the following proposition.

Proposition 3.5. *Let $\mathcal{F}$ denote the coverage ratio. As the system transitions into a new window, before any new items arrive, for any protected group p, the coverage $C(p)$ and weighted distance $\Delta(p)$ are adjusted by α, while the unfairness status indicator remains unchanged. In other words, at the start of a new window, the values* *of coverage, unfairness status, and weighted distance for p are updated to $C'(p) = \alpha \cdot C(p)$, $UF'_{\mathcal{F}}(p)$ maintaining as $UF_{\mathcal{F}}(p)$, and $\Delta'(p) = \alpha \cdot \Delta(p)$, respectively.*

Proof. Consider any protected group p in *stage 1* with a total number of W windows. The coverage $C(p)$, the weighted distance $\Delta(p)$, and the coverage ratio $CR(p)$ are

$$C(p) = \sum_{w=0}^{W-1} |D_p^w| \alpha^w$$

$$\Delta(p) = \alpha \cdot C(D) - C(p) = \alpha \cdot \sum_{w=0}^{W-1} |D^w| \alpha^w - \sum_{w=0}^{W-1} |D_p^w| \alpha^w$$

$$CR(p) = \frac{C(p)}{C(D)} = \frac{\sum_{w=0}^{W-1} |D_p^w| \alpha^w}{\sum_{w=0}^{W-1} |D^w| \alpha^w}$$

Upon entering a new window (*stage 2*), the equations become

$$C'(p) = \sum_{w=0}^{W} |D'^w_p| \alpha^w$$

$$\Delta'(p) = \alpha \cdot C'(D) - C'(p) = \alpha \cdot \sum_{w=0}^{W} |D'^w| \alpha^w - \sum_{w=0}^{W} |D'^w_p| \alpha^w$$

$$CR'(p) = \frac{C'(p)}{C'(D)} = \frac{\sum_{w=0}^{W} |D'^w_p| \alpha^w}{\sum_{w=0}^{W} |D'^w| \alpha^w}$$

Since the data in window w in *stage 2* are the same as those in window $w-1$ in *stage 1*, for all w such that $0 \le w \le W-1$, we have $D'^w_p = D_p^{w-1}$ and $D'^w = D^{w-1}$. No items have arrived in the more recent window in *stage 2*. Therefore,

$$C'(p) = \left(\sum_{w=1}^{W} |D'^w_p| \alpha^w \right) + |D'^0_p| \alpha^0 = \left(\sum_{w=0}^{W-1} |D'^{w+1}_p| \alpha^{w+1} \right) + 0$$
$$= \alpha \cdot \sum_{w=0}^{W-1} |D_p^w| \alpha^w = \alpha \cdot C(p)$$

Similarly, we have $C'(D) = \alpha \cdot C(D)$. Thus,

$$\Delta'(p) = \alpha^2 \cdot C(D) - \alpha \cdot C(p) = \alpha \cdot \Delta(p)$$

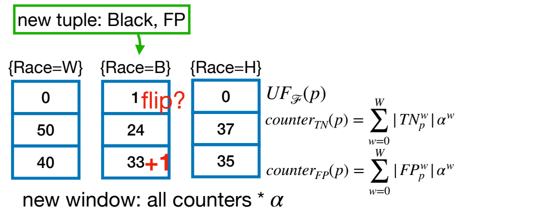
$$CR'(p) = \frac{C'(p)}{C'(D)} = \frac{\alpha \cdot C(p)}{\alpha \cdot C(D)} = CR(p)$$

This proves that the bit $UF_{\mathcal{F}}(p)$ remains constant when transitioning into a new window without the addition of new data, but $C(p)$ and $\Delta(p)$ should be multiplied by a factor α. □

Based on the above propositions, we can prove the correctness of DFMonitor as follows.

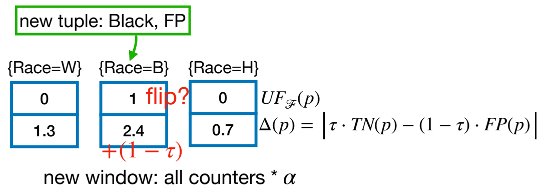
Theorem 3.6. *For any protected group $p \in P$ monitored by DFMonitor, at any point after initialization, the counter value $counter(p)$ accurately represents the time decay coverage of p, the bit $UF_{\mathcal{F}}(p)$ correctly indicates whether the coverage ratio of p is below the threshold τ, and $\Delta(p)$ equals $|\tau \cdot C(D) - C(p)|$.*

Proof. The theorem is proven by induction. Right after the initialization, DFMonitor satisfies the conditions by design. Assuming the conditions hold at any given moment, we demonstrate that they continue to be met after any subsequent operation or event.



new tuple: Black, FP
{Race=W}
{Race=B}
{Race=H}
0
50
40
1
24
33
flip?
+1
0
37
35
$UF_{\mathcal{F}}(p)$
$counter_{TN}(p) = \sum_{w=0}^{W} |TN_p^w| \alpha^w$
$counter_{FP}(p) = \sum_{w=0}^{W} |FP_p^w| \alpha^w$
new window: all counters * α

Figure 3: DFMonitor, basic version for false positive rate



new tuple: Black, FP
{Race=W}
{Race=B}
{Race=H}
0
1.3
1
2.4
flip?
$+(1-\tau)$
0
0.7
$UF_{\mathcal{F}}(p)$
$\Delta(p) = \left|\tau \cdot TN(p) - (1-\tau) \cdot FP(p)\right|$
new window: all counters * α

Figure 4: DFMonitor, optimized version for false positive rate

Following an insertion, Proposition 3.4 ensures the conditions are maintained. As the system enters a new window, Proposition 3.5 ensures the conditions are preserved. Query operations do not alter any fields within DFMonitor, thus maintaining the status. This inductive approach confirms the theorem's correctness. □

4 ALGORITHM FOR OTHER APPLICATIONS

The algorithm for monitoring coverage ratio in Section 3 can be adapted to assess many other fairness metrics of a Machine Learning classifier on streaming data, such as false positive rate, false negative rate, and accuracy. Our analysis primarily focuses on fairness definitions grounded in the model's binary predictions and their correspondence to binary outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). We use false positive error rate, denoted as $FPR = \frac{FP}{FP+TN}$, as an example to illustrate how to apply our dynamic algorithm. Other fairness metrics analyzing different aspects of the classifier's predictions in relation to actual outcomes can be managed in a similar manner, but we do not cover all these metrics in detail within this paper due to constraints on space.

The data structure and operations of DFMonitor for FPR monitoring share similarities and differences with those for coverage ratio. The calculation of FPR for a group is influenced solely by its FP and TN counts, unlike the coverage ratio that is affected by the arrival of any tuple across all groups. We outline key differences specific to handling FPR below.

(1) **Data structure:** As shown in Figure 3 and Figure 4, DFMonitor for monitoring FPR resembles that for the coverage ratio but with notable modifications. The basic version associates each group with two time decay counters: $counter_{TN}(p)$ for true negatives and $counter_{FP}(p)$ for false positives. The optimized version calculates the weighted distance as $\Delta(p) = |\tau \cdot TN(p) - (1-\tau) \cdot FP(p)|$.
(2) **Initialization:** The initial setup requires $counter_{TN}(p)$ and $counter_{FP}(p)$ to track the counts of negatives and false positives, respectively. After the first window, the FPR for each group is computed, setting $\Delta(p)$ accordingly.
(3) **Insertion:** The dynamics of DFMonitor are only influenced by the occurrence of false positives or true negatives. Then, upon the arrival of a false positive or true negative tuple t, only the fields of groups satisfying t are updated. Specifically, the arrival

Algorithm 2: Insert(t): insert tuple t to DFMonitor for monitoring FPR

```
$P \leftarrow$ the set of protected groups;
$P_t \leftarrow \{p \in P | t$ satisfies $p\}$;
$\mathcal{F} \leftarrow$ false positive rate;
if $t$ is a false positive or true negative then
    for $p \in P_t$ do
        if $t$ is a true negative then
            if $UF_{\mathcal{F}}(p) == False$ then
                $\Delta(p) \leftarrow \Delta(p) + \tau$;
            else
                /* $UF_{\mathcal{F}}(p) == True$ */
                if $\Delta(p) \geq \tau$ then
                    $\Delta(p) \leftarrow \Delta(p) - \tau$;
                else
                    $\Delta(p) \leftarrow \tau - \Delta(p)$;
                    $UF_{\mathcal{F}}(p) \leftarrow False$;
        else
            /* $t$ is a false positive */
            if $UF_{\mathcal{F}}(p) == False$ then
                if $\Delta(p) \geq 1 - \tau$ then
                    $\Delta(p) \leftarrow \Delta(p) - (1 - \tau)$;
                else
                    $\Delta(p) \leftarrow 1 - \tau - \Delta(p)$;
                    $UF_{\mathcal{F}}(p) \leftarrow True$;
            else
                /* $UF_{\mathcal{F}}(p) == True$ */
                $\Delta(p) \leftarrow \Delta(p) + 1 - \tau$;
```

of a true negative adjusts the delta value by τ, whereas a false positive's arrival modifies it by $(1 - \tau)$.

Algorithm Extension

The capability of DFMonitor extends to evaluating the false positive rate (FPR) for the union of protected groups.

Consider p_1 and p_2 as two distinct protected groups, and p as their union (for instance, Black males, Black females, and all Black individuals, respectively). For the basic DFMonitor model, calculating the FPR is straightforward due to the availability of all necessary counters. For the optimized model, we have three distinct scenarios:
(1) **Both groups below threshold:** If both p_1 and p_2 exhibit FPRs below a certain threshold, p will also fall below this threshold. This conclusion is drawn from the premise that for both groups, the condition $FP < \tau \cdot (FP + TN)$ holds true. Thus, we can deduce: $FPR(p) = \frac{FP(p_1)+FP(p_2)}{FP(p_1)+TN(p_1)+FP(p_2)+TN(p_2)} < \tau$.
(2) **Both groups above threshold:** Similarly, if p_1 and p_2 both have high FPRs, p will also exhibit a high FPR, following the same reasoning as the first scenario.
(3) **Groups with opposite fairness statuses:** When p_1 and p_2 have different fairness statuses (one below and one above the threshold), the overall fairness status of p ($UF_{\mathcal{F}}(p)$) depends on the comparison of their delta values. Assuming p_1 has a lower FPR and p_2 a higher one, we calculate: $TN(p_1) = \frac{\Delta(p_1)}{\tau} + \frac{FP(p_1)}{\tau} - FP(p_1)$ and $TN(p_2) = \frac{\Delta(p_2)}{\tau} + \frac{FP(p_2)}{\tau} - FP(p_2)$. This leads to $FPR(p) = \tau \cdot \frac{FP(p_1)+FP(p_2)}{FP(p_1)+FP(p_2)+\Delta(p_1)-\Delta(p_2)}$. So the group with the smaller delta value has a FPR below τ.

5 EXPERIMENTS

We empirically evaluate our proposed solutions to demonstrate the importance of our proposed problem and the usefulness and effectiveness of our solution. We describe the experimental setup and provide some real-world case studies to compare the our dynamic fairness measurements with the traditional methods, showing the benefits of our dynamic solution. Then we analyze the impact of time decay factor and time window size on the results of fairness metrics calculation using various datasets, fairness metrics, and protected groups. Moreover, we validate the effectiveness of our algorithms and optimization in Section 3.2. The experimental results demonstrate that our dynamic method for assessing fairness excels in terms of both time and space efficiency, while also outperforming traditional static approaches by delivering more stable, consistent, and fluctuation-resistant results.

5.1 Experiment Setup

To the best of our knowledge, this is the first paper studying dynamic fairness measurement, so there is no standard benchmark or previous work for testing this problem. For our evaluation, we need to use datasets that satisfy several criteria: they 1) must arrive over time, 2) be used for classification tasks with accessible prediction results and ground truth, 3) relevant in scenarios where fairness matters such as demographic studies, and 4) ideally have evolving data distribution or prediction algorithm over time to track shifts in fairness metrics. We use 1) the COMPAS dataset, which is widely used for fairness research but has never been treated as a data stream although every individual has a screening time associated with them; and 2) Baby names dataset, which illustrates the shifts in the association between names and genders over several decades. Protected groups are defined using gender and/or race, as these attributes are commonly employed as sensitive attributes. Additionally, we ensured the protected group's size is at least 10% of the total data or more, to ensure the fairness metrics for this group are meaningful and representative. In terms of fairness metrics, we choose metrics that could be extracted from the data, were likely to change over time, and/or traditionally used for specific datasets.
Datasets:

- **COMPAS dataset**, sourced by ProPublica from Broward County, Florida, comprises data on 7214 defendants for examining racial bias in criminal risk evaluations. It includes demographics, criminal history, COMPAS scores, and two-year recidivism outcomes. It reveals significant racial disparities in False Positive Rates (FPR), especially higher incorrect high-risk assessments for Black defendants compared to their counterparts, making this dataset widely used for studying algorithmic fairness. We use the *"compas_screening_date"* field as the time, since it represents the date on which defendants were evaluated by this tool.
- **Baby name dataset** contains the top 1000 most popular girl and boy names in the US from 1880 to 2020, aggregated from

the US Social Security Administration [31]. It includes the (first) name, year of birth, and sex (male/female) of the infants registered with these names each year, facilitating gender prediction (male/female) from first names over time. This task is crucial for various applications, such as identifying the gender of authors in biographies when gender information is not explicitly provided, or understanding customer demographics for marketing strategies. The dataset is underscored by the fluidity of name-gender associations over time [32]; for instance, the name Leslie was predominantly male in 1900 but became largely female by 2000. Names like Madison, Morgan, Sydney, and Kendall have similarly shifted genders over the century, highlighting the importance of incorporating temporal dynamics into gender prediction models to maintain accuracy. To predict the gender based on first names, we employ the genderize.io API [33], with country specified as US. This API connects to a database of over 100,000 first names from around half a million social network profiles as of May 2014 [34]. This database is regularly updated by scanning public profiles across major social networks, guaranteeing that its data remains current and accurate.

Protected groups: We use traditional sensitive attributes gender and race where applicable, focusing on demographic groups that constitute at least 10% of the data.

- **For COMPAS dataset:** We identify Black, White, and Hispanic individuals as protected groups.
- **For Baby name dataset:** We use gender male and female in this dataset to define the two protected groups.

Metrics: We choose accessible, non-static, widely-used fairness metrics, and use the smoothness scores for a curves that depict fairness values to evaluate the temporal stability of fairness measurement.

- **Fairness metrics:** We use the *coverage ratio (CR)* and *false positive rate (FPR)* for the COMPAS dataset, and use the prediction *accuracy* for the Baby name dataset. The coverage ratio is not considered for the Baby Name dataset since it maintains an equal number of names for boys and girls annually.
- **Smoothness scores:** To assess the temporal consistency in our fairness evaluations,we calculate smoothness scores for the fairness metric trajectories of each protected group over time. This is done by calculating the standard deviation of the first derivatives of these trajectories to assess their variability, i.e., roughness. We then get smoothness scores by invert and normalize these values to a $0 \sim 1$ scale based on the maximum value found in the derivatives, ensuring that higher scores indicate greater smoothness. This method assigns the smoothest curve a score of 1, providing an intuitive and clear interpretation of temporal fairness stability.

Time decay factor and window size: In our experiments, we default to using $\alpha = 0.5$ for the time decay factor, within the range of $0 < \alpha \leq 1$. Regarding the choice of window size, we use 1 month for the COMPAS dataset and 10 years for the Baby name dataset. This decision is guided by the desire to achieve a balance: we aim for a sufficient number of windows to accurately capture the evolution of fairness metrics, yet we also seek to keep the quantity manageable for ease of visualization and analysis. Practically, having 10 to 30 windows keeps this balance, offering both clarity and detailed insight in graphical representations. Given the COMPAS dataset spans 24 months, a 1-month window size is utilized. For the Baby name dataset, which covers a period from 1880 to 2020 (141 years in total), we use a 10-year window as the default size, aligning with our criteria for an effective analysis framework.

Compared algorithms: In our experiments, we compare the following approaches:

- **Traditional approach:** Fairness metrics are calculated per window by exclusively using data from that specific period.
- **Dynamic approach:** Fairness metrics are calculated with an exponential time decay, implemented in both the basic and optimized versions of our proposed method. Consequently, they produce identical outcomes. For the purpose of fairness result analysis, we primarily utilize the basic DFMonitor version for its detailed output, including both the fairness value and status. The distinction between these two versions of DFMonitor is made explicit only when they are directly compared to each other.

Each method operates on in-memory data. We begin with case studies comparing the different fairness interpretation of the same prediction results generated by the traditional approach and our dynamic approach. Then we investigate the impact of various parameters on the results of the dynamic approach. Next we evaluate and compare the basic and optimized versions of DFMonitor.

Platform: All experiments were performed on a macOS Ventura 13.2 machine with an Apple M2 Max chip, 64 GB memory. The algorithms were implemented using Python3.

5.2 Dynamic vs. Traditional Approach

We compare the outcomes of our dynamic approach for measuring fairness, which considers these temporal variations, against the traditional method which does not.

Fairness values comparison:

To illustrate the superior interpretative power of our dynamic approach in fairness measurement within streaming data compared to the static approach, we examine its application to the COMPAS and Baby name datasets, as shown in Figures 5, 6, 7 and 8. In these figures, the x-axis of the figures represent the time we calculate the fairness metrics, and the y-axis represent the measurement results. The dynamic method is denoted by solid lines, whereas the traditional method is indicated by dashed lines. We assess the false positive rate (FPR) and coverage ratio (CR) for different racial groups in the COMPAS dataset over a 1-month window and prediction accuracy for genders in the Baby Name dataset over a decade.

While both methods generally yield similar trends and values at each measurement point, the dynamic method, by accounting for historical data with a focus on recent information, produces smoother, more stable curves than the traditional method, indicating its effectiveness in handling abrupt data fluctuations. For example, in the COMPAS dataset, as the 6th and the 11th measurement points in Figure 5(a) (highlighted with bold grids), the dynamic approach shows less pronounced declines in FPR for both Black and White groups, while the traditional method shows significant drops. This results in a clearer, more consistent curve that better represents long-term trends. Similarly, Figure 5(b) displays the measurement of coverage ratio across racial groups over time, with the dynamic method smoothing the fluctuations while the traditional approach appearing more erratic.

The second advantage of the dynamic method is its consistent identification of bias, such as the ongoing bias against Black defendants in the COMPAS dataset, which can be obscured by the considerable fluctuations shown by the traditional method and result in potentially misleading outcomes at certain intervals. For example, the dynamic method consistently shows the FPR for Black defendants exceeding a predefined threshold of $\tau = 0.3$ at the end of each month, indicating sustained bias, despite observable dips at the 6th and 12th measurement points. Conversely, the traditional method (represented by the dotted light blue line) exhibited some monthly variability, with FPR readings below the threshold at the 6th and 12th points, potentially leading to erroneous interpretations if analyses were primarily based on data from these intervals.

Additionally, the dynamic method is adept at managing missing data by maintaining the last available measurement as the current result, ensuring continuous fairness assessment even when new data for a group is absent. This approach contrasts with the traditional method, which fails to provide any assessment in such cases, as presented in Figure 5(a). In this figure, from the 16th measurement point onwards, the absence of false positives or true negatives in the data makes the FPR calculation inapplicable. Nevertheless, the dynamic method continues to deliver results using the last available data point, while the traditional method stops offering any insights. Specifically, at the 16th measurement point, the traditional method records an FPR of 1 for Whites and 0 for Black and Hispanic groups due to data absence, leading to erratic increases and drops in their graphical representation. In contrast, the dynamic method's consistency ensures reliable and robust longitudinal fairness assessments, demonstrating its advantage in analyzing datasets with time-sensitive variations.

Quantifying curve smoothness:

In addition to visually assessing the smoothness of the curves, we quantitatively evaluate each curve's smoothness by calculating their smoothness scores. Recall that we normalize the scores of curves in the same figure to a $0 \sim 1$ scale where 1 represents the smoothness score of the smoothest curve. The computed smoothness values are depicted in Figure 6 for the COMPAS dataset curves and in Figure 8 for the baby names dataset curves, with the colors corresponding to those in Figures 5 and 7, respectively.

It is evident that visually smoother curves also score higher on the smoothness scale. For example, in the COMPAS dataset, the curve representing Black individuals's FPR (the solid blue line in Figure 5(a)) achieves the highest smoothness score (in Figure 6(a)), reflecting its visual smoothness. Similarly, the curves for Hispanic individuals' CR in the COMPAS dataset (the solid red line in Figure 5(b)) and for female names in the Baby names dataset (the solid green line in Figure 7) also receive the highest smoothness scores, indicating their smoothness in their respective figures.

5.3 Ablation Study

We conducted ablation studies on the COMPAS and Baby Names datasets to explore how adjustments in the time decay factor and time window length influence fairness outcomes.

Time decay factor:

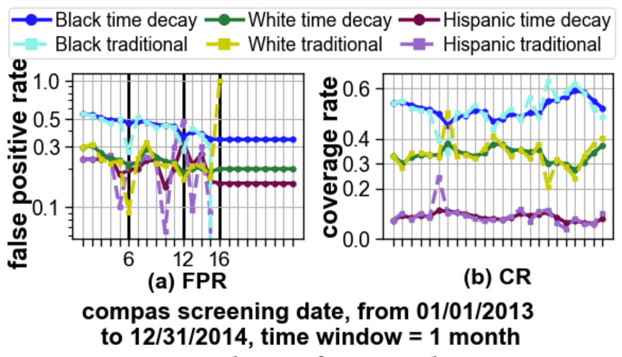


Figure 5: COMPAS dataset, fairness values comparison

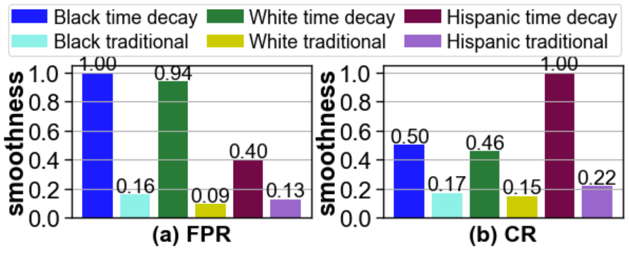


Figure 6: COMPAS dataset, smoothness comparison

Adjusting the time decay factor α from 0.1 to 1 impacts the smoothness of the fairness outcome curves, with solid curves representing changes in α (from dark blue to dark green) and dotted magenta curves illustrating traditional method outcomes without time decay, as shown in Figures 9 and 10. The x-axis of these figures represents the timeline of measurement, and the y-axis shows the values of the fairness metrics. An increase in α smooths the curves by giving greater weight to older data, thereby enhancing the influence of historical data and reducing the impact of recent data on the fairness metrics. At $\alpha = 1$, all historical data receive equal weighting of 1. In contrast, the traditional method, which relies solely on data from the current window, produces the most erratic curves, due to sensitivity to recent data changes. Essentially, this traditional method can be seen as setting the time decay factor to 0, thereby excluding historical data from its calculations.

To quantify these observations, we calculated smoothness scores for these curves, depicted in Figure 11 for the COMPAS dataset and Figure 12 for the Baby Names dataset. In these figures, each column represents a specific subfigure in Figure 9 and 10, with annotations like "B, F" indicating the figure that examines the False Positive Rate (FPR) for black individuals. Rows are organized by the time decay factor, with the colors matching those in Figure 9 and 10. Consistent with expectations, higher α values correlate with higher smoothness scores, confirming that the dynamic approach offers a more stable and reliable assessment of fairness over time.

Time window size:

The effect of different time window sizes on fairness outcomes for the COMPAS and Baby Name datasets is depicted in Figure 13 and 14, where each curve's color represents a distinct window size. The y-axis shows fairness metric values, and the x-axis represents normalized measurement times for uniform comparison. For example, in Figure 13, the COMPAS dataset analysis spans 24 months with the smallest window set at 2 weeks, resulting in 48 distinct

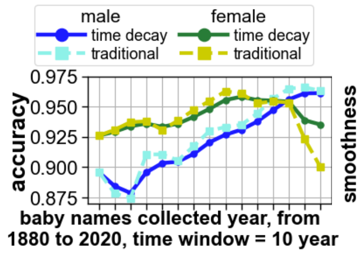


Figure 7: Baby name dataset, fairness values comparison

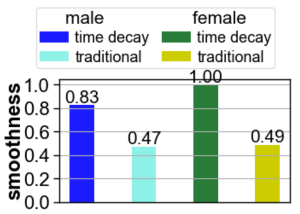


Figure 8: Baby name dataset, smoothness comparison

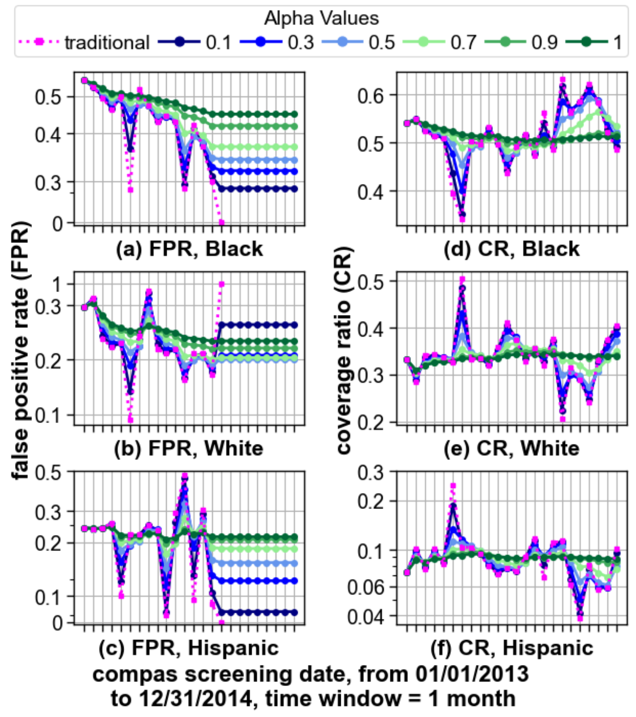


Figure 9: COMPAS dataset, change time decay factor α

measurement intervals. The spacing for curves of larger window sizes is adjusted to align with this 48-point scale.

The trends across these figures are consistent: the dark blue curve, indicative of the smallest window size, exhibits the most variability. Curves transition to lighter shades of blue and green as window sizes increase, becoming progressively smoother. This is because larger windows cover fewer measurement points but aggregate more data over longer periods, naturally smoothing out short-term fluctuations and resulting in more stable variations in fairness metrics over time. Detailed calculations of smoothness scores, similar to previous analyses, are not included here due to space constraints.

5.4 Algorithm Execution Time and Space Consumption

We conducted a comparative analysis of the execution time and memory consumption between the basic and optimized iterations of DFMonitor. Findings indicate that both versions operate at a comparable and time-efficient insertion speed, with the optimized variant of DFMonitor demonstrating faster speed to transition into a new time window, and reduced memory usage due to the optimization detailed in Section 3.2.

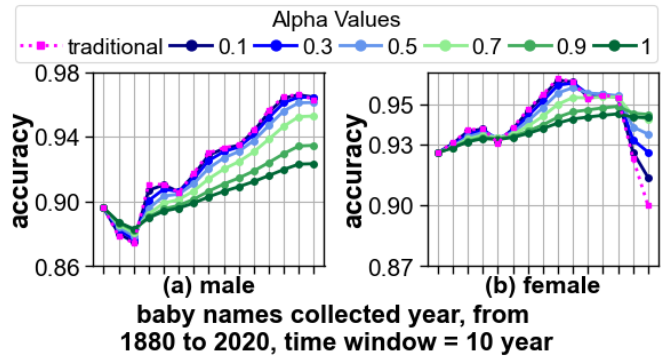


Figure 10: Baby names dataset, change time decay factor α

	B, F	W, F	H, F	B, C	W, C	H, C
🟪	0.04	0.03	0.05	0.05	0.07	0.06
🟦	0.09	0.14	0.07	0.06	0.09	0.10
🟦	0.15	0.21	0.10	0.08	0.13	0.18
🟦	0.25	0.30	0.16	0.14	0.23	0.29
🟩	0.41	0.49	0.28	0.27	0.43	0.49
🟩	0.75	0.81	0.64	0.66	0.83	0.85
🟩	1.00	1.00	1.00	1.00	1.00	1.00

Figure 11: COMPAS dataset smoothness, change time decay factor α

	male	female
🟪	0.32	0.11
🟦	0.36	0.13
🟦	0.45	0.17
🟦	0.56	0.23
🟩	0.69	0.35
🟩	0.87	0.69
🟩	1.00	1.00

Figure 12: Baby name dataset smoothness, change time decay factor α

Execution Time Analysis

Our evaluation of DFMonitor shows that both the basic and optimized versions of DFMonitor exhibit identical linear time complexity. During each insertion process, they both undergo the same procedural checks and modifications across an equivalent number of fields. Thus, it is anticipated that their insertion speeds would be similar. To validate this assumption, we tested both versions on COMPAS and Baby names datasets, treating them as continuous data streams where entries are sequentially added according to their timestamps. Protected groups within these datasets were randomly generated based on race and sex, where applicable. Considering the potential impact of the number of monitored protected groups on performance, we varied this number from 1 to 13 for COMPAS dataset and 1 to 3 for the Baby names dataset, creating five instances for each group set, and performed ten rounds of testing on each stream to derive an average performance metric.

The results show a negligible difference in the execution times for data insertion between the basic and optimized version of DFMonitor: with times of approximately 2.56×10^{-5} seconds and 1.64×10^{-5} seconds on the COMPAS and Baby names datasets, respectively. The discrepancy was often less than 1% of the basic version's time in over 85% of cases. Similarly, when evaluating metrics such as the False Positive Rate (FPR) on the COMPAS dataset and accuracy on the Baby names dataset, the execution times were roughly 1.23×10^{-4} seconds and 4.14×10^{-5} seconds, respectively. Additionally, the transition to a new time window demonstrated that the basic version required about 15% more time for coverage ratio analysis and 30% more time for FPR/accuracy analysis compared to the optimized version, attributable to the basic version's need to scale more than one list by a factor of α, unlike the refined version, which only scales a single list.

Memory Usage Analysis

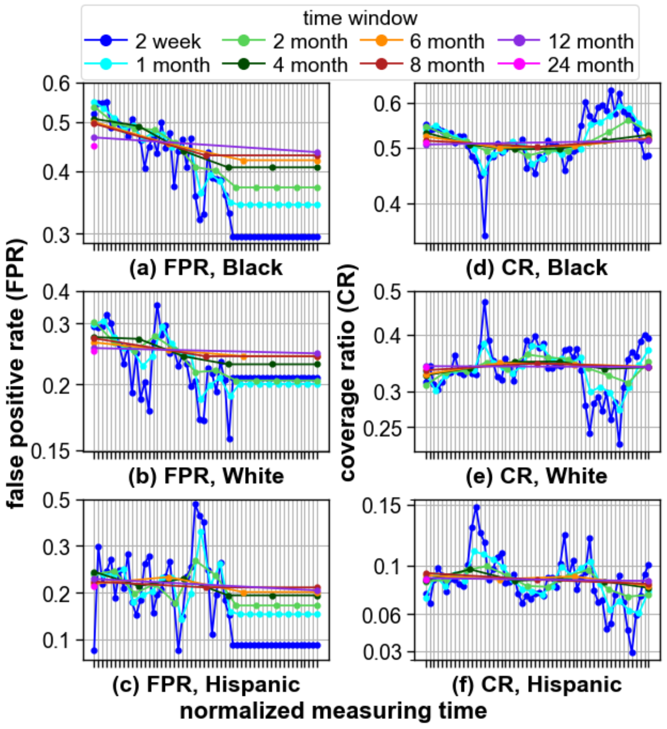


Figure 13: COMPAS dataset, change window size

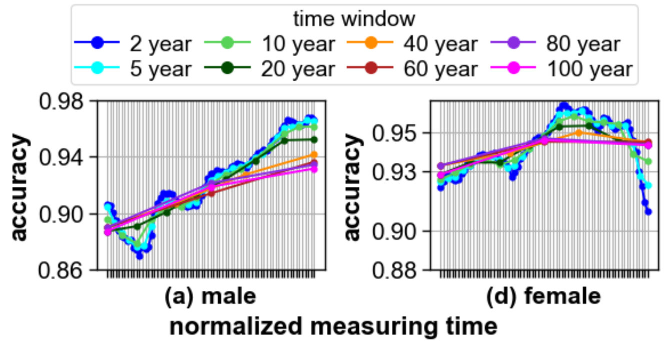


Figure 14: Baby names dataset, change window size

Our examination of memory requirements employed a similar approach, without the need for repeated measurements to determine averages, as the space consumed by the data structure remains constant under the same configurations. For simplicity, in these experiments, we use 64-bit floating-point numbers for both counters in the basic version of DFMonitor and delta values in the optimized version, respectively, although delta values could, in practice, require less space. The comparison on coverage ratio revealed that both iterations of DFMonitor generally consumed similar amounts of memory, with the optimized version showing a slight reduction of about 2% in both datasets. This is because the data structure's design maintains a nearly identical number of fields across both versions, with the primary difference being the basic version's additional global counter.

The situation changes when examining other fairness metrics, such as FPR on the COMPAS dataset and accuracy on the Baby names dataset. Specifically, for the Baby names dataset, the optimized version achieved a reduction in memory usage of approximately 15.35% to 16.31% as the number of protected groups increased. For the COMPAS dataset, memory savings ranged from 17.6% to 26.04% as the number of protected groups expanded from 1 to 13. This improvement is largely due to the optimized version's efficient use of a single delta value list for recording weighted distances, contrasting with the basic version's reliance on two separate counter lists. Further, our findings indicate an increment in space savings with an increase in the number of protected groups, as longer lists for more groups predominantly occupy space. Consequently, the proportion of space saved aligns closely with the eliminated list's space. For metrics like false positive rates and false negative rates, with a large number of protected groups, the space savings could be near 50%. This dynamic approach not only reduces space consumption for DFMonitor but is also advantageous for a range of other fairness metrics where the weighted distance is expressible through linear functions.

6 RELATED WORK

We next overview multiple lines of related work.

Algorithms for time series computation. Time series is a sequence of data points recorded at regular time intervals. It is pervasive in our daily life and usually requires real-time data processing for tasks like recent data statistics maintenance and data stream machine learning. However, such non-stationary environments present challenges in data stream analysis like class imbalance [35, 36] and concept drift [37–39], which can impact performance. To mitigate these issues, various methods have been developed, focusing on prioritizing new over old information. Some algorithms use time-decaying functions with a forgetting factor to weaken the influence of older data [19, 21, 36, 40, 41]. Other algorithms emphasize recent data by sliding window strategies, which only maintains the most recent data [23, 38, 39, 42–47], sampling techniques which summarize data stream characteristics over longer periods [48–50] and gradual forgetting [51–55] which assigns decaying weights to data over time. These approaches ensure that the most recent data is given priority for retraining, adjustment, and evaluation.

Fairness-aware algorithms for dynamic environments. In numerous real-world applications where fairness matters, like recommendation systems and demographic predictions, it is important to maintain fairness amidst environmental changes such as the shifting user interests [56–59] and evolving demographic profiles [60–62]. While some research has explored fairness in the context of shifting environment [14, 63, 64], many of them have primarily focused on fairness intervention strategies to ensure certain levels of fairness under specific types of data shifts. Some adapt to particular deployment environments [61, 65–72], while others create robust models capable of performing accurately in any unknown environments [73–80]. Additionally, fairness in sequential decision-making [14, 16, 81, 82] explores situations where past decisions impact future data populations and subsequent decisions. None of these approaches propose fairness assessment in dynamic setting.

Fairness definitions and evaluations. Fairness in decision-making systems is broadly categorized into individual and group fairness. Individual fairness advocates for similar treatment of individuals with comparable qualifications regardless of demographics, but it is usually challenging to specify suitable distance metrics to measure the similarity between individuals. Group fairness requires all demographic groups to be treated equally without bias on any particular demographic group, with various fairness definitions proposed. However, the pursuit of multiple fairness definitions often leads to incompatibilities [18, 83, 84] and trade-offs between overall predictive performance and fairness [85]. In ranking scenarios, fairness requires proportional representation of groups in top-ranking positions relative to their overall data presence [12, 86, 87].

This body of work highlights the critical need for fairness measurement methods adept at navigating the complexities of time-evolving data analysis.

7 CONCLUSION AND FUTURE WORK

In this paper, we study the problem of measuring real-time dynamic group fairness metrics for classification in the context of streaming data. Despite extensive research on developing fairness aware models in non-stationary environments and measuring fairness metrics, previous studies have not addressed the issues of fairness evaluation in a changing environment. To the best of our knowledge, this is the first work studying fairness measurements in time series. Our novel approach introduces a method of applying a time-decaying weight to traditional fairness metrics, prioritizing recent data while forgetting older data by segmenting the data chronologically and employing an exponential decay function α^t for weighting. This method allows for the re-evaluation of traditional fairness metrics, offering a real-time, dynamic perspective on fairness. We have developed novel data structures and algorithms that enable the real-time monitoring of time-decaying fairness metrics for different protected groups, with optimization techniques for improved space efficiency without sacrificing the accuracy of fairness assessments. Our empirical analysis on real-world time series datasets illustrates the advantage of our dynamic method over traditional static approaches, demonstrating its ability to produce stable, smooth results that more accurately reflect current situations without being highly affected by sporadic data fluctuations. Furthermore, our experiments confirm the time and space efficiency of our proposed solutions.

REFERENCES

[1] Manny Fernandez. Study finds disparities in mortgages by race. *New York Times*, 15, 2007.

[2] Jeffrey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women. In *Ethics of data and analytics*, pages 296–299. Auerbach Publications, 2022.

[3] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580, 2018.

[4] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the international workshop on software fairness*, pages 1–7, 2018.

[5] Abolfazl Asudeh, Zhongjun Jin, and HV Jagadish. Assessing and remedying coverage for a given dataset. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 554–565. IEEE, 2019.

[6] Jurek Leonhardt, Avishek Anand, and Megha Khosla. User fairness in recommender systems. In *Companion Proceedings of the The Web Conference 2018*, pages 101–102, 2018.

[7] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. A survey on the fairness of recommender systems. *ACM Transactions on Information Systems*, 41(3):1–43, 2023.

[8] Simon Caton and Christian Haas. Fairness in machine learning: A survey. *ACM Computing Surveys*, 2020.

[9] Abigail Z Jacobs and Hanna Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 375–385, 2021.

[10] Enrique Amigó, Yashar Deldjoo, Stefano Mizzaro, and Alejandro Bellogín. A unifying and general account of fairness measurement in recommender systems. *Information Processing & Management*, 60(1):103115, 2023.

[11] Eliana Pastor, Luca De Alfaro, and Elena Baralis. Looking for trouble: Analyzing classifier behavior via pattern divergence. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1400–1412, 2021.

[12] Jinyang Li, Yuval Moskovitch, and HV Jagadish. Detection of groups with biased representation in ranking. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 2167–2179. IEEE, 2023.

[13] Raphael Sonabend, Florian Pfisterer, Alan Mishler, Moritz Schauer, Lukas Burk, Sumantrak Mukherjee, and Sebastian Vollmer. Flexible group fairness metrics for survival analysis. *arXiv preprint arXiv:2206.03256*, 2022.

[14] Alexander D'Amour, Hansa Srinivasan, James Atwood, Pallavi Baljekar, David Sculley, and Yoni Halpern. Fairness is not static: deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 525–534, 2020.

[15] Lily Hu and Yiling Chen. A short-term intervention for long-term fairness in the labor market. In *Proceedings of the 2018 World Wide Web Conference*, pages 1389–1398, 2018.

[16] Lydia T Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. Delayed impact of fair machine learning. In *International Conference on Machine Learning*, pages 3150–3158. PMLR, 2018.

[17] Melissa Hamilton. The sexist algorithm. *Behavioral sciences & the law*, 37(2):145–157, 2019.

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

[19] Edith Cohen and Martin Strauss. Maintaining time-decaying stream aggregates. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–233, 2003.

[20] Abolfazl Asudeh, HV Jagadish, Julia Stoyanovich, and Gautam Das. Designing fair ranking schemes. In *Proceedings of the 2019 international conference on management of data*, pages 1259–1276, 2019.

[21] Graham Cormode, Flip Korn, and Srikanta Tirthapura. Exponentially decayed aggregates on data streams. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1379–1381. IEEE, 2008.

[22] Odysseas Papapetrou, Minos Garofalakis, and Antonios Deligiannakis. Sketch-based querying of distributed sliding-window data streams. *arXiv preprint arXiv:1207.0139*, 2012.

[23] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.

[24] Black artists say a.i. shows bias, with algorithms erasing their history. https://www.nytimes.com/2023/07/04/arts/design/black-artists-bias-ai.html.

[25] Achieving instagram growth in the age of ai and algorithmic bias. https://www.forbes.com/sites/anniebrown/2021/10/18/achieving-instagram-growth-in-the-age-of-ai-and-algorithmic-bias/?sh=7d54c74d764d.

[26] How social media monetization is evolving in the face of algorithmic bias: A discussion with nick mccandless. https://www.forbes.com/sites/anniebrown/2021/11/14/how-social-media-monetization-is-evolving-in-the-face-of-algorithmic-bias-a-discussion-with-nick-mccandless/?sh=5158418f739e.

[27] Monetization at a glance: Tiktok vs. youtube shorts vs. instagram reels. https://www.theleap.co/blog/tiktok-instagram-reels-youtube-shorts-monetization-comparison/.

[28] Commission adopts rules on independent audits under the digital services act. https://digital-strategy.ec.europa.eu/en/news/commission-adopts-rules-independent-audits-under-digital-services-act.

[29] Cutting through the jargon - independent audits in the digital services act. https://foundation.mozilla.org/en/blog/cutting-through-the-jargon-independent-audits-in-the-digital-services-act/.

[30] The digital markets act: ensuring fair and open digital markets. https://digital-markets-act.ec.europa.eu/about-dma_en.

[31] Popular baby names. https://www.ssa.gov/OACT/babynames/.

[32] Cameron Blevins and Lincoln Mullen. Jane, john... leslie? a historical method for algorithmic gender prediction. *DHQ: Digital Humanities Quarterly*, 9(3), 2015.

[33] Determine the gender from a name. https://genderize.io/.

[34] Kamil Wais. Gender prediction methods based on first names with genderizer. *R J.*, 8(1):17, 2016.

[35] Shuo Wang, Leandro L Minku, Davide Ghezzi, Daniele Caltabiano, Peter Tino, and Xin Yao. Concept drift detection for online class imbalance learning. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2013.

[36] Shuo Wang, Leandro L Minku, and Xin Yao. A learning framework for online class imbalance learning. In *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 36–45. IEEE, 2013.

[37] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[38] Jeffrey C Schlimmer and Richard H Granger. Incremental learning from noisy data. *Machine learning*, 1:317–354, 1986.

[39] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101, 1996.

[40] Shuo Wang, Leandro L Minku, and Xin Yao. Online class imbalance learning and its applications in fault detection. *International Journal of Computational Intelligence and Applications*, 12(04):1340001, 2013.

[41] Shuo Wang, Leandro L Minku, and Xin Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, 2014.

[42] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, 2002.

[43] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

[44] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[45] Lap-Kei Lee and HF Ting. Maintaining significant stream statistics over sliding windows. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 724–732. Citeseer, 2006.

[46] Lukasz Golab, Shaveen Garg, and M Tamer Özsu. On indexing sliding windows over online data streams. In *International Conference on Extending Database Technology*, pages 712–729. Springer, 2004.

[47] Phillip B Gibbons and Srikanta Tirthapura. Distributed streams algorithms for sliding windows. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 63–72, 2002.

[48] Mohammed Al-Kateb, Byung Suk Lee, and X Sean Wang. Adaptive-size reservoir sampling over data streams. In *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, pages 22–22. IEEE, 2007.

[49] Pavlos S Efraimidis and Paul G Spirakis. Weighted random sampling with a reservoir. *Information processing letters*, 97(5):181–185, 2006.

[50] Charu C Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases*, pages 607–618, 2006.

[51] David P Helmbold and Philip M Long. Tracking drifting concepts by minimizing disagreements. *Machine learning*, 14:27–45, 1994.

[52] Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent data analysis*, 8(3):281–300, 2004.

[53] Ralf Klinkenberg and Stefan Rüping. Concept drift and the importance of examples. In *Text mining–theoretical aspects and applications*. Citeseer, 2002.

[54] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.

[55] Sanmin Liu, Shan Xue, Jia Wu, Chuan Zhou, Jian Yang, Zhao Li, and Jie Cao. Online active learning for drifting data streams. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[56] Yunxiao Chen, Xiaoou Li, Jingchen Liu, and Zhiliang Ying. Recommendation system for adaptive learning. *Applied psychological measurement*, 42(1):24–41, 2018.

[57] Aminu Da'u, Naomie Salim, and Rabiu Idris. An adaptive deep learning method for item recommendation system. *Knowledge-Based Systems*, 213:106681, 2021.

[58] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[59] Simon Dooms. Dynamic generation of personalized hybrid recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 443–446, 2013.

[60] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in neural information processing systems*, 34:6478–6490, 2021.

[61] Stephen Giguere, Blossom Metevier, Yuriy Brun, Bruno Castro da Silva, Philip S Thomas, and Scott Niekum. Fairness guarantees under demographic shift. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[62] Alessandro Castelnovo, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Andrea Cosentini. Towards fairness through time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 647–663. Springer, 2021.

[63] Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. The disparate effects of strategic manipulation. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 259–268, 2019.

[64] Smitha Milli, John Miller, Anca D Dragan, and Moritz Hardt. The social cost of strategic classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 230–239, 2019.

[65] Vasileios Iosifidis and Eirini Ntoutsi. -online fairness-aware learning under class imbalance. In *International Conference on Discovery Science*, pages 159–174. Springer, 2020.

[66] Vasileios Iosifidis, Thi Ngoc Han Tran, and Eirini Ntoutsi. Fairness-enhancing interventions in stream classification. In *Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30*, pages 261–276. Springer, 2019.

[67] Wenbin Zhang and Albert Bifet. Feat: A fairness-enhancing and concept-adapting decision tree classifier. In *Discovery Science: 23rd International Conference, DS 2020, Thessaloniki, Greece, October 19–21, 2020, Proceedings 23*, pages 175–189. Springer, 2020.

[68] Wenbin Zhang, Albert Bifet, Xiangliang Zhang, Jeremy C Weiss, and Wolfgang Nejdl. Farf: A fair and adaptive random forests classifier. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 245–256. Springer, 2021.

[69] Wenbin Zhang, Mingli Zhang, Ji Zhang, Zhen Liu, Zhiyuan Chen, Jianwu Wang, Edward Raff, and Enza Messina. Flexible and adaptive fairness-aware learning in non-stationary data streams. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 399–406. IEEE, 2020.

[70] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[71] Amanda Coston, Karthikeyan Natesan Ramamurthy, Dennis Wei, Kush R Varshney, Skyler Speakman, Zairah Mustahsan, and Supriyo Chakraborty. Fair transfer learning with missing protected attributes. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 91–98, 2019.

[72] Hantian Zhang, Xu Chu, Abolfazl Asudeh, and Shamkant B Navathe. Omnifair: A declarative system for model-agnostic group fairness in machine learning. In *Proceedings of the 2021 international conference on management of data*, pages 2076–2088, 2021.

[73] Wei Du and Xintao Wu. Fair and robust classification under sample selection bias. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2999–3003, 2021.

[74] Wei Du, Depeng Xu, Xintao Wu, and Hanghang Tong. Fairness-aware agnostic federated learning. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 181–189. SIAM, 2021.

[75] Bahar Taskesen, Viet Anh Nguyen, Daniel Kuhn, and Jose Blanchet. A distributionally robust approach to fair classification. *arXiv preprint arXiv:2007.09530*, 2020.

[76] Debmalya Mandal, Samuel Deng, Suman Jana, Jeannette Wing, and Daniel J Hsu. Ensuring fairness beyond the training data. *Advances in neural information processing systems*, 33:18445–18456, 2020.

[77] Ashkan Rezaei, Rizal Fathony, Omid Memarrast, and Brian Ziebart. Fairness for robust log loss classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5511–5518, 2020.

[78] Ashkan Rezaei, Anqi Liu, Omid Memarrast, and Brian D Ziebart. Robust fairness under covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9419–9427, 2021.

[79] Haotao Wang, Junyuan Hong, Jiayu Zhou, and Zhangyang Wang. How robust is your fairness? evaluating and sustaining fairness under unseen distribution shifts. *Transactions on machine learning research*, 2023, 2023.

[80] Arpita Biswas and Suvam Mukherjee. Ensuring fairness under prior probability shifts. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 414–424, 2021.

[81] Min Wen, Osbert Bastani, and Ufuk Topcu. Algorithms for fairness in sequential decision making. In *International Conference on Artificial Intelligence and Statistics*, pages 1144–1152. PMLR, 2021.

[82] Samer B Nashed, Justin Svegliato, and Su Lin Blodgett. Fairness and sequential decision making: Limits, lessons, and opportunities. *arXiv preprint arXiv:2301.05753*, 2023.

[83] 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*, pages 214–226, 2012.

[84] Ben Green. The false promise of risk assessments: epistemic reform and the limits of fairness. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 594–606, 2020.

[85] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, pages 797–806, 2017.

[86] L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. Ranking with fairness constraints. *arXiv preprint arXiv:1704.06840*, 2017.

[87] Meike Zehlike, Ke Yang, and Julia Stoyanovich. Fairness in ranking: A survey. *arXiv preprint arXiv:2103.14000*, 2021.