C-SHIFT: Efficient Cluster-based Model Fairness Control under Data Drift

Yijie Li
yxl2056@student.bham.ac.uk
University of Birmingham
Birmingham, UK

Huiping Chen
h.chen.13@bham.ac.uk
University of Birmingham
Birmingham, UK

Paolo Missier
p.missier@bham.ac.uk
University of Birmingham
Birmingham, UK

Abstract

Machine learning models deployed in real-world scenarios must contend with data shifts that occur over time, resulting in degraded model performance and potentially exacerbating fairness concerns. Considerable research has focused separately on maintaining either model accuracy or algorithmic fairness under distribution shifts, but not both. Separately, previous results are also available for detecting the *harmful* regions of the training set, where data shifts have the highest impact, with the goal of reducing the cost of retraining. In this work, we propose **C-SHIFT** (**C**luster-based **S**elective **H**armful Shift **I**dentification for **F**airness-aware **T**raining), a framework for efficiently managing model performance and fairness together in general data shift scenarios. Starting from an initial model with a satisfactory accuracy-fairness tradeoff, **C-SHIFT** activates on batches of serving data, using a novel cluster-based algorithm to identify harmful data regions that may appear in some of the clusters. **C-SHIFT** restores fairness and accuracy by either partially retraining or fine-tuning the original model, achieving efficiency by focusing only on the harmful data within a portion of the clusters. **C-SHIFT** works well with multiple fairness adjustment methods, and is not sensitive to the specific type of data shift. Because clusters are also agnostic to the data type (they only require a suitable distance metric), in principle, the approach applies to multiple data modes and is not restricted to tabular data. We evaluate the approach using three real-world datasets as well as synthetic datasets specifically designed to simulate harmful data shift scenarios. Our results indicate that **C-SHIFT** can restore accuracy-fairness balance with quality comparable to a baseline global retraining approach, but using a small fraction of the training/serving data, with similar results for Logistic Regression (LR) as well as nonlinear Neural Network (NN) models.

Keywords

Data Shifts, Algorithm Fairness, Data Bias

1 Introduction

Automatic decision-making systems increasingly impact societal outcomes, such as hiring [7, 14], loans [32], and criminal justice [6], highlighting the challenges of ensuring fairness in Machine Learning (ML) models. Fairness in ML refers to the equitable treatment of individuals and groups, regardless of sensitive or protected attributes such as race, gender, age, or socioeconomic status [25], and is essential for promoting trust, transparency, and accountability in automated systems.

Despite the growing attention to fairness-aware learning methods [11, 25], most existing approaches assume a fixed data distribution and focus on in-distribution fairness guarantees [3, 5]. However, achieving fairness remains a challenge in dynamic, real-world environments where data often arrive from diverse sources and under varying conditions, leading to inevitable drifts between training and deployment data. These shifts might amplify biases, disproportionately disadvantaging certain groups and compromising fairness objectives [31]. On the other hand, while numerous methods have been proposed to detect and adapt to distribution shifts [20, 23, 33, 34], these focus predominantly on maintaining accuracy. Methods that jointly detect data shifts leading to degradation of model predictive abilities and fairness violations remain scarce, limiting the robustness of ML models deployed in real-world settings.

To examine this phenomenon and appreciate the problem, we conduct the following preliminary experiment. Consider ACSIncome [15], an updated version of the widely used 1996 UCI Adult dataset [4]. In addition to providing socioeconomic attributes of individuals over the years, this version also groups individuals by State (US). Making use of this natural partitioning, we train two models, M_b^S, M_f^S, separately for each State S. M_b^S is a baseline, fairness-unaware Logistic Regression (LR) model, while M_f^S is a fairness-aware LR model with a widely-used reweighting strategy (LR+RW) [21] that enforces the demographic parity (DP) fairness metric (defined in Sec. 2.1) on the protected attribute SEX through sample weighting.

We evaluate models M_b^S, M_f^S for each S on two test scenarios. Firstly, we reserve a test dataset from the overall population in State S, as normal, and this provides an in-distribution (ID) evaluation. Then, we also evaluate M_b^S, M_f^S against every other State S' test data, providing multiple out-of-distribution (OOD) test scenarios. Each evaluation produces two metrics: accuracy (Accuracy : $\Pr(\hat{y} = y)$) and absolute demographic parity violation (DP : $|\Pr(\hat{y} = 1|\text{SEX} = 1) - \Pr(\hat{y} = 1|\text{SEX} = 0)|$), where lower DP values indicate better fairness.

Figure 1 presents the results for four sample States, AZ, CO, GA, and IN. Each row shows results for models trained on one State, with the left plot displaying baseline LR and the right showing LR+RW. In each subplot, the orange cross marks the ID evaluation performance, while gray circles represent OOD evaluations on other states. Best overall results are found in the lower-right quadrant (high accuracy, better fairness). In all cases, results improve when using LR+RW, as expected. However, with regards to reacting to shifts, we note the following: (1) Performance changes, but these changes are not obvious. When models are tested under ID or OOD evaluations, their performance can show varying degrees of degradation, and in some cases, even improvement. For example, for the LR+RW model trained on CO data, several OOD results (gray circles) show similar or even improved accuracy compared to ID evaluation (orange cross). (2) There is often a tension between accuracy and fairness, and their patterns differ between ID and OOD evaluations. For instance, for AZ, GA, and IN, accuracy is always higher for ID than OOD, but the same does not hold for fairness. (3) Even when an OOD

EDBT '26, Tampere (Finland)

10.48786/edbt.2026.26

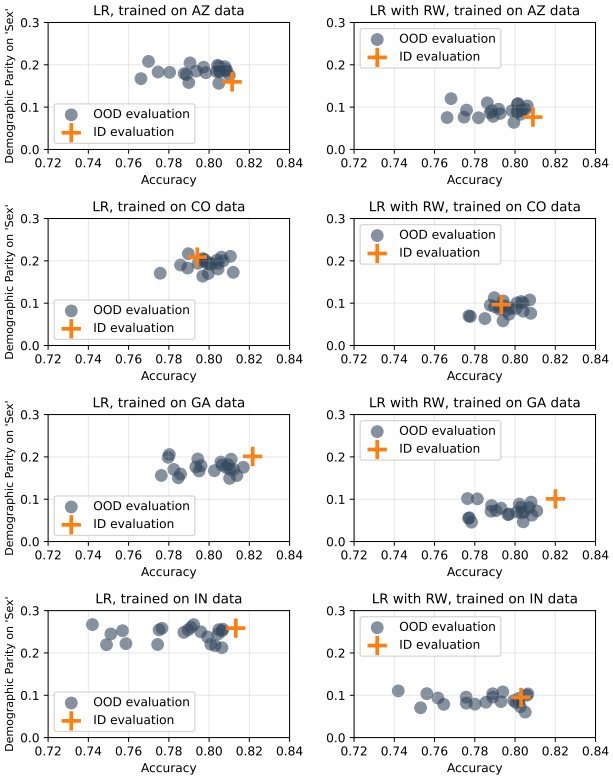

Figure 1: Performance comparison of models under in-distribution (ID) and out-of-distribution (OOD) evaluation on ACSIncome-2018 data [15] across different U.S. states.

dataset results in lower fairness than the ID dataset for a State, a model trained on that State's ID dataset may nevertheless exhibit greater unfairness when evaluated on an OOD dataset.

We will use this example again as part of our evaluation (Sec. 5). For now, we conclude that the effects of distributional changes in real-world data, and their varying impacts on model behaviour, may be complex and somewhat unpredictable. This suggests a number of questions that motivate our work. Firstly, should we always worry about all the data shifts, as it appears they are not equally harmful to the model? Previous work [16] has addressed this question, but with limitations, mainly focusing only on model accuracy and on tabular training data. Secondly, which regions of the OOD dataset are responsible for the observed accuracy-fairness loss? Are there efficient ways to identify those and thus implement restorative interventions that are local and thus more efficient than global retraining? Identifying the sources of fairness and accuracy shifts is essential for developing robust ML systems that maintain their desirable properties across diverse deployment contexts.

Challenges. Addressing these questions presents several challenges. (1) (*Region Discovery*) How to automatically identify cohesive regions in high-dimensional spaces where harmful shifts concentrate, without prior knowledge of shift patterns. (2) (*Multi-objective Assessment*) How to jointly evaluate performance and fairness degradation in a principled manner that captures their interaction, rather than treating them independently. (3) (*Scalability*) How to maintain computational efficiency and save costs for expert relabelling, for instance, using active learning, for instance, in [16]. (4) (*Interpretability*) How to provide human-understandable explanations of detected shifts that can guide intervention strategies.

Contributions. We propose **C-SHIFT**, a cluster-based framework that captures the dual objectives of maintaining predictive performance and ensuring model fairness under data drift.

- **C-SHIFT** identifies data regions that are *harmful* relative to a model trained on the data. These are the regions of the training set that are responsible for both performance and fairness degradation. Our approach leverages clustering in the serving data (i.e., new data that are potentially OOD) to discover cohesive sub-populations, and then quantifies the contribution of each cluster to overall performance and fairness metrics under shift. This contribution extends to a sequence of serving data batches ("chunks"), using a cluster-merging technique;
- **C-SHIFT** leverages the proposed *HarmScore* to provide intuitive and interpretable insights into data shifts. Specifically, the *HarmScore* reveals which subsets of the data have the greatest impact on model accuracy and fairness, as well as the magnitude of their influence.
- We evaluate the approach on both real-world and synthetic data, reporting effectiveness and efficiency results relative to a number of baselines and against two broadly used classes of classification models (Logistic Regression and Neural Network). Our results indicate that using clustering to partition the training space leads to efficient interventions (using a fraction of the data) to restore accuracy and fairness levels comparable to those achieved using global retraining, using either retraining from scratch or a fine-tuning (continued training) approach.

2 Definitions and Problem Statement

We first introduce notations, fairness definitions, and the concept of harmful data shifts (Sec. 2.1), then formally state our problem of efficiently restoring model performance under distribution shift (Sec. 2.2).

2.1 Definitions

Our work is set in the context of standard Machine Learning classification problems, where the feature space includes one *sensitive attribute*. Following common practice, we denote the d-dimensional feature space by $\mathcal{X} \subseteq \mathbb{R}^d$ where the sensitive attribute $Z \in \mathcal{X}$ is binary. Let $\mathcal{Y}$ denote the set of labels (typically, $\mathcal{Y} = \{0, 1\}$). We denote the classifier model as $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ with optimal parameters θ after training on dataset $D_0 = \{(\mathbf{x}_i, y_i)\}_{i=1}^{|D_0|}$ where each value x_{ij} for feature X_j is drawn from distribution $P_0(X_j)$. After deployment, the model makes inferences on data points from a batch of serving data D_s (also referred to as a *chunk* below). We denote the predicted outcome for input feature vector $\mathbf{x}$ by $\hat{y} = h_\theta(\mathbf{x})$ where we assume that the x_{ij} are drawn from a potentially different distribution, $P_s(X_j)$. We refer to the differences between $P_0(X_j)$ and $P_s(X_j)$ generically as *distribution shifts*. To simplify notation, we will refer to models of the form $h_\theta()$ as $f, f_0, f_1 \ldots$.

Let Z denote a sensitive (or protected) attribute that partitions the population into a protected group and an unprotected group, where $Z \in \{0, 1\}$ indicates group membership. With reference to the sensitive attribute Z, DP states that the probability of receiving a positive outcome $\hat{y} = 1$ for input $\mathbf{x}$ should be conditionally

independent of group membership: $\Pr(\hat{y} = 1 \mid z = 1) = \Pr(\hat{y} = 1 \mid z = 0)$. EO imposes a stronger condition, requiring that the probability of receiving a *true* positive outcome $\hat{y} = 1$ be conditionally independent of group membership given the true label: $\Pr(\hat{y} = 1 \mid z = 1, \hat{y} = y) = \Pr(\hat{y} = 1 \mid z = 0, \hat{y} = y)\Big|$.

Enforcing model fairness. Three main approaches are normally taken to ensure model fairness, including pre-processing, in-processing, and post-processing methods [11, 25]. These aim to address imbalance, e.g., $\Pr(\hat{y} = 1 \mid z = 1) > \Pr(\hat{y} = 1 \mid z = 0)$ where the $z = 1$ group receives more favourable treatment than the $z = 0$ group. Pre-processing methods modify training data before training; in-processing methods incorporate fairness constraints into learning; and post-processing methods adjust model predictions or decision thresholds after training.

Harmful Data Shift. We say that a distribution shift from P_0 to P_s is considered *harmful* to model performance if it causes degradation in either model accuracy or fairness metrics:

$$\begin{cases} \text{Acc}(f, D_s) - \text{Acc}(f, D_0) < 0, \\ \text{Fair}(f, D_s) - \text{Fair}(f, D_0) > 0. \end{cases} \tag{1}$$

where $\text{Acc}(\cdot)$ denotes accuracy, $\text{Fair}(\cdot)$ denotes one of the fairness metrics introduced above. Unlike traditional approaches that treat performance and fairness degradation separately, harmful shifts here usually represent complex distributional changes that compromise both objectives.

2.2 Problem Statement

The challenges we set out initially translate into the following specific problems.

Let D_0 and D_s be the training set and one chunk of serving data, respectively,[1] and let f_0 be a model initially trained on D_0, with corresponding accuracy and fairness measures $\text{Acc}(f_0, D_0)$, $\text{Fair}(f_0, D_0)$, respectively.

Firstly, we aim to identify a partition $\mathcal{R} = \{R_1, R_2, \ldots, R_K\}$ of regions $R_i \subseteq D_0 \cup D_s$, where the local distributions for each R_i will be different, in general, from that of D_0. We then map the differences in such distributions to corresponding differences in accuracy and fairness (see Eq (1)), introduce a measure of joint degradation of fairness and accuracy (*HarmScore*, Sec. 4.1.2), and use it to identify the subset $\mathcal{R}' \subset \mathcal{R}$ of harmful regions.

Finally, we aim to retrain or fine-tune model f_0 using data points in $\mathcal{R}'$ and yielding f_1, with the aim to jointly restore both accuracy and fairness, i.e.,

$$\min(\text{Acc}(f_1, \mathcal{R}') - \text{Acc}(f_0, D_0)) \tag{2}$$

$$\min(\text{Fair}(f_1, \mathcal{R}') - \text{Fair}(f_0, D_0)) \tag{3}$$

while at the same time minimising the size of $\mathcal{R}'$, that is, the volume of data points required to restore the model.

Next, we describe our approach to localising model retraining, and then show experimentally that it achieves accuracy/fairness levels that are comparable to those obtained through global retraining or fine-tuning, at a fraction of the cost.

In summary, we develop a cluster-based framework that enables the partition of the data (see Sec. 4.1.1), localisation of harmful data regions (see Sec. 4.1.2), and adaptation to data drifts in a chunk-wise way (see Sec. 4.2).

3 Related Work

Machine learning models deployed in real-world systems face significant challenges from data distribution shifts, which can degrade both predictive performance and fairness. Prior research addressing these challenges has largely evolved along two parallel tracks: techniques for ensuring algorithmic fairness and methods for detecting as well as adapting to distribution shifts [31]. Our work sits at the intersection of algorithmic fairness, data drift detection, and efficient model adaptation.

Several systems have been developed for multi-objective fairness optimisation. Agarwal et al. [1] formulate fair classification as a sequence of cost-sensitive learning problems. Martinez et al. [24] propose minimax Pareto fairness for multi-objective optimization. OmniFair [38] provides a declarative system for model-agnostic group fairness using unconstrained optimisation with soft constraints. While these methods effectively balance accuracy-fairness trade-offs, they are designed for static datasets and perform global optimisation over all data. In contrast, our work targets dynamic scenarios where data arrives in chunks and aims to identify and intervene only on problematic regions.

A large body of work focuses on detecting when the data distribution changes [20, 23, 34]. These methods typically rely on statistical tests over feature distributions or on monitoring predictive performance metrics such as accuracy, ROC-AUC, or precision. Comprehensive surveys [23] highlight a rich landscape of drift detection and adaptation techniques, but these approaches are predominantly accuracy-centric. A recent study [28] further demonstrates that not all distribution shifts equally affect model performance, motivating the need for more selective adaptation strategies. In this context, the DDLA framework [16] identifies regions of the data space with low accuracy and selectively retrain models on these harmful subsets, achieving efficiency gains over global retraining. While DDLA [16] strongly inspires our work, it focuses exclusively on accuracy degradation and does not consider fairness objectives. More recently, Modyn [8] proposes a data-centric ML pipeline orchestration system that handles concept drift through trigger-based model updates and data versioning. While these methods effectively address accuracy degradation, they do not consider the fairness implications of distribution shifts. As a result, fairness-harming regions that do not manifest as severe accuracy drops may remain undetected. Our work extends the selective retraining paradigm to jointly optimise accuracy and fairness objectives.

Besides, slice finding methods have also been used to identify problematic data subsets for model updating and debugging. Recent work [12] extends slice finding to various debugging scenarios. Chung et al. [13] propose automated data slicing using conjunctions of attribute values to discover underperforming subgroups. SliceLine [29] provides fast slice discovery through linear algebra operations. However, slice finding typically uses axis-aligned attribute conjunctions that create overlapping regions, while our clustering-based approach provides disjoint partitions suitable for retraining. Second, slice finding focuses on error localisation for debugging rather than adaptive retraining under drift. Third, these methods do not explicitly model group fairness violations. Our cluster-based regions provide geometric interpretability (examples see Sec. 6.3) while enabling efficient fairness-aware adaptation.

To our knowledge, C-SHIFT is the first framework that combines drift-aware detection, joint accuracy-fairness optimisation, and localised intervention through clustering. By integrating cluster-level drift localisation with a unified HarmScore that captures both accuracy and fairness degradation, C-SHIFT provides a principled and efficient alternative to global retraining for maintaining fair and accurate models in dynamic environments.

[1] We discuss solving the problem for a sequence of chunks later in the paper.

4 Methodology

In this section, we introduce our **C-SHIFT** framework and algorithms, which address the single-serving data setting (Section 4.1) and the chunk-wise setting (Section 4.2), where serving data arrive as a sequence of chunks. An overview of the framework is shown in Figure 2.

4.1 The C-SHIFT framework and algorithm

We first consider a single-batch serving data setting. Let f_0 be the model trained on D_0, and let D_s be one serving batch. Our goal is to extract a subset of D_s that harms f_0's performance, and use it to update the model so that accuracy and fairness improve.

Overview of the procedure: (i) cluster $D_0 \cup D_s$ using a distance-based hierarchical method to obtain C; (ii) identify a subset of D_s that has a harmful impact on accuracy and the group-fairness metric (see Equation (5)); (iii) update the model on this subset to restore performance, e.g., through retraining or fine-tuning (introduced in Sec. 5.5).

4.1.1 ***Distance-based clustering.*** Clustering forms the basis of our approach, providing a systematic way to measure model fairness across different regions of the data. Model bias often arises from imbalanced data distributions. By clustering the combined dataset $D_0 \cup D_s$ and computing fairness metrics within each cluster, we can assess how the new batch D_s affects fairness. This process localises regions where group-level disparities emerge due to distributional shifts introduced by incoming data. For instance, in a loan application dataset, if the male and female applicants initially show a small approval gap (e.g., within 2%), but the new batch D_s increases it to 10%, this indicates a drift-induced bias. Such behaviour reveals fairness degradation that may require model adaptation.

Clustering $D_0 \cup D_s$ also groups samples with similar features, enabling the framework to detect local subpopulations or patterns that the original model f_0 may have underrepresented. Analysing these clusters allows the framework to identify regions where performance or fairness deterioration is concentrated.

Before applying clustering, we remove the sensitive attribute z and the label y from the feature space to ensure that protected group membership or outcome information does not dominate the clustering process. Note that clustering is purely a geometric partition. The label and sensitive-attribute information are fully preserved for accuracy/fairness evaluation. Removing these two features ensures clusters are formed based on non-sensitive feature similarity rather than sensitive attributes. All features are normalised (if not already done during preprocessing) so that each feature contributes equally to distance computations.

The choice of clustering algorithm is not directly connected to detecting data shifts. Rather, we simply aim to cluster similar data points together, and a hierarchical clustering in combination with simple Euclidean distance seems to serve the purpose. Specifically, we first cluster the training and serving data points separately, and then we merge them. Thus, we believe that using spectral clustering in this context would not provide an additional advantage.

We employ an agglomerative hierarchical clustering algorithm inspired from [10] with a distance threshold r. Unlike partition-based methods such as k-means, which require specification of the number of clusters, we adopt a complete-linkage approach [27]. This approach determines the maximum inter-cluster distance allowed, controlling the number of resulting clusters automatically. At each iteration, the algorithm merges the pair of clusters whose combination yields the smallest cluster diameter, that is, the smallest maximum intra-cluster distance. The merging process continues in a bottom-up manner until any further merge would produce a cluster with a diameter exceeding r. The resulting clustering, denoted as C, ensures that no two points within the same cluster are more than r apart in Euclidean distance. The parameter r thus controls the granularity of clustering: a larger r produces fewer, broader clusters, improving efficiency but potentially merging distinct subpopulations and obscuring local fairness disparities. Conversely, a smaller r yields more fine-grained clusters, increasing sensitivity to local distribution shifts at the expense of higher computational cost.

4.1.2 ***Problematic shift detection.*** In data-driven machine learning systems, complex interactions between accuracy loss and fairness violations are common. Standard drift detection methods usually assess these two aspects in isolation. In practice, a detected shift creates a difficult choice: should we update the model with data that improves accuracy but harms fairness, or with data that preserves fairness but reduces accuracy? There are four outcomes: accuracy increases or decreases, combined with fairness improving or worsening. These combinations make data selection for retraining difficult, and they hinder a consistent selection strategy.

To answer the question above, we propose a unified metric, *HarmScore*, that captures both accuracy degradation and fairness degradation under data shifts. This metric quantifies the potential harm introduced by data drift. The model owner then decides whether to incorporate a batch of new data based on its *HarmScore*. Algorithm 1 presents the overall procedure for detecting *problematic shifts* using *HarmScore*.

The design of the *HarmScore* follows two requirements: (i) a continuous, fine-grained evaluation of the accuracy–fairness trade-off, and (ii) an appropriate cluster-level fairness metric. We discuss both points below.

Continuous evaluation. Following the idea of [16], a naive approach to data selection would partition serving data into four discrete categories based on the direction of accuracy and fairness changes: (1) accuracy ↑, fairness ↑; (2) accuracy ↑, fairness ↓; (3) accuracy ↓, fairness ↑; (4) accuracy ↓, fairness ↓. This categorical approach suffers from several critical limitations: (1) **Coarse granularity**: Binary categorisation ignores the magnitude of changes. For example, a cluster with a minor accuracy drop could be treated identically to one with catastrophic performance loss; (2) **Ambiguous trade-offs**: When accuracy and fairness move in opposite directions (e.g., categories 2 and 3), it is unclear which clusters to prioritise for retraining; (3) **Data inefficiency**: Rigid category-based selection may either retain excessive data (including marginally useful clusters) or discard valuable instances due to arbitrary cutoffs.

By contrast, a continuous metric like *HarmScore* enables fine-grained discrimination among clusters. It encodes the accuracy-fairness trade-offs without requiring explicit case-by-case analysis. Each cluster receives a quantitative score, allowing the selection threshold τ to be tuned based on downstream needs.

Fairness metric. Measuring accuracy for a small group is simple. The main challenge is to choose a fairness metric at the cluster level.

The intuition of our cluster-based approach to harmful shift detection relies on the principle that fairness degradation under distribution shift can be localised to specific data regions. For example, for demographic parity (DP) on data D, the *overall*

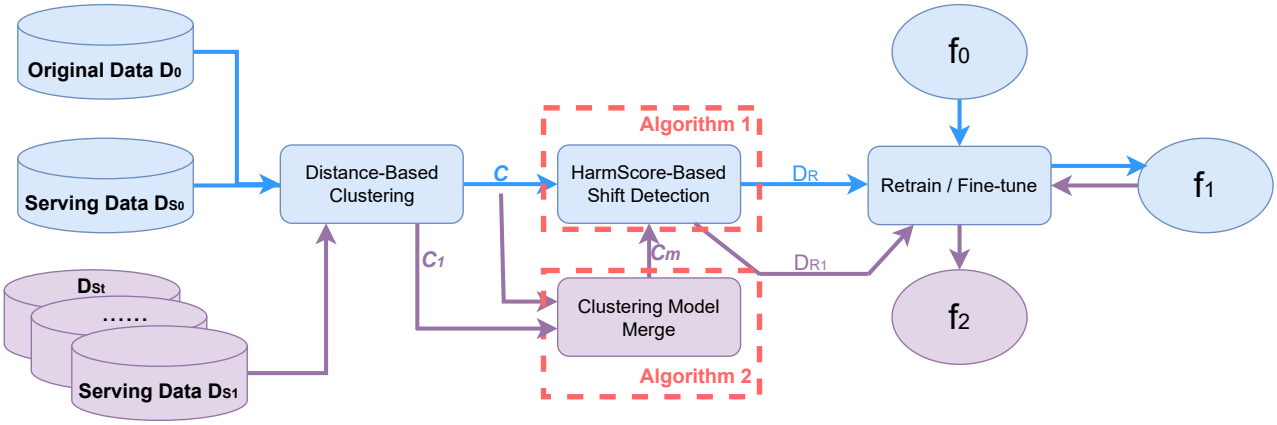

Figure 2: Workflow of the C-SHIFT framework for single-serving data (in blue) and its variation for chunk-wise serving data (in purple).

fairness metric for one group can be decomposed using the law of total probability:

$$\Pr(\hat{y}=1 \mid z=s, D) = \sum_{j=1}^{K} \Pr(\hat{y}=1 \mid z=s, C^{(j)}) \cdot \Pr(C^{(j)} \mid z=s, D) \tag{4}$$

where $C^{(j)}$ denotes cluster j. This decomposition reveals that overall fairness is a weighted combination of cluster-level fairness contributions ($w_s^{(j)} = \Pr(C^{(j)} \mid z=s, D)$). Consequently, degradation in overall fairness $\mathrm{DP}(D_0 \cup D_s) > \mathrm{DP}(D_0)$ must originate from fairness degradation within one or more clusters.

Beyond this, cluster-level analysis also provides enhanced diagnostic power. By partitioning data based on feature similarity (excluding z and y from distance computation), confounding variables that might obscure the relationship between sensitive attributes and predictions are controlled. Within a cohesive cluster $C^{(j)}$ where individuals share similar feature distributions, any observed fairness violation $\mathrm{DP}(C^{(j)}) > 0$ more directly reflects model bias toward the sensitive attribute rather than legitimate feature-based differences. This aligns with the principle of individual fairness [17], which requires similar individuals to receive similar predictions: clusters naturally group similar individuals, making cluster-level fairness assessment a practical bridge between group and individual fairness notions.

As a result of label sparsity due to clustering, a cluster may be dominated by one class; for example, it may contain only points with $y = 0$. In this case, class-conditional fairness metrics, such as equalised odds or equal opportunity, become ill-defined or uninformative. Given this constraint, we adopt a group-level parity measure. For cluster $C^{(j)}$ we define:

$$\mathrm{Fair}(C^{(j)}) = \Pr(\hat{y}=1 \mid f_0, C^{(j)}, z=s) - \Pr(\hat{y}=1 \mid f_0, C^{(j)}, z \neq s), \tag{5}$$

where $Pr(\hat{y}=1 \mid f_0, C^{(j)}, z=s)$ is the probability that model f_0 predicts a positive outcome for instances from favoured sensitive group s in $C^{(j)}$.

***HarmScore* design** We introduce $\lambda \in [0,1]$ as a *fairness sensitivity* parameter that balances accuracy and fairness. A smaller λ (e.g., 0.2–0.4) gives more weight to accuracy, which suits applications such as recommender systems. A larger λ (e.g., 0.8–1.0) prioritises fairness, which suits high-stakes domains such as criminal justice or credit scoring.

Algorithm 1 *HarmScore*-Based Drifts Detection

Input: Clusters C over $D_0 \cup D_s$, threshold τ, Acc_{Base}, f_0, and $\lambda \in [0,1]$.
Output: Selected subset D_R for model update.

1: Initialise arrays ΔAcc[] and ΔFair[] to store accuracy and fairness changes.
2: **for** each cluster $C^{(j)} \in C$ **do**
3: **if** $C^{(j)} \cap D_s == \emptyset$ **then continue**
4: **end if**
5: $\Delta\mathrm{Acc}[j] \leftarrow \mathrm{Acc}_{Base} - \mathrm{Acc}(f_0, D_s \cap C^{(j)})$
6: $\Delta\mathrm{Fair}[j] \leftarrow \mathrm{Fair}(f_0, C^{(j)}) - \mathrm{Fair}(f_0, D_0 \cap C^{(j)})$
7: **end for**
8: Let $\Delta\mathrm{Acc}_{\max}$, and $\Delta\mathrm{Fair}_{\max}$, be the maximum value of $|\Delta\mathrm{Acc}[]|$ and $|\Delta\mathrm{Fair}[]|$ respectively.
9: **for** each j in ΔAcc[] and F[] **do**
10: $\Delta\mathrm{Acc}[j] \leftarrow \Delta\mathrm{Acc}[j]/\Delta\mathrm{Acc}_{\max}$
11: $\Delta\mathrm{Fair}[j] \leftarrow \Delta\mathrm{Fair}[j]/\Delta\mathrm{Fair}_{\max}$
12: **end for**
13: $D_R \leftarrow \emptyset$
14: **for** each cluster $C^{(j)} \in C$ **do**
15: $HarmScore(C^{(j)}) \leftarrow (1-\lambda) \cdot \Delta\mathrm{Acc}[j] + \lambda \cdot \Delta\mathrm{Fair}[j]$
16: **if** $HarmScore_j > \tau$ **then**
17: $D_R \leftarrow D_R \cup (D_s \cap C^{(j)})$
18: **end if**
19: **end for**
20: **return** D_R

Let $\Delta\mathrm{Acc}(C^{(j)})$ denote the change in accuracy on new instances in cluster $C^{(j)}$ relative to the baseline accuracy Acc_{Base} of f_0 (refer to Algorithm 1, line 5). Let $\Delta\mathrm{Fair}(C^{(j)})$ be the change in fairness after adding new instances in $C^{(j)}$; by convention, we set $\Delta\mathrm{Fair}(\emptyset) = 0$ (refer to Algorithm 1, line 6). To bring ΔAcc and ΔFair to the same scale, we normalise them across clusters (see Algorithm 1, line 8 to line 12). Let

$$\Delta\mathrm{Acc}_{\max} = \max_{C^{(j)} \in C} |\Delta\mathrm{Acc}(C^{(j)})|, \Delta\mathrm{Fair}_{\max} = \max_{C^{(j)} \in C} |\Delta\mathrm{Fair}(C^{(j)})|.$$

$$\Delta\mathrm{Acc}_{\mathrm{norm}}(C^{(j)}) = \frac{\Delta\mathrm{Acc}(C^{(j)})}{\Delta\mathrm{Acc}_{\max}}, \Delta\mathrm{Fair}_{\mathrm{norm}}(C^{(j)}) = \frac{\Delta\mathrm{Fair}(C^{(j)})}{\Delta\mathrm{Fair}_{\max}}.$$

From an accuracy perspective, a region is harmful if the deployed model performs worse on serving data than on the original training distribution. For fairness, harmfulness is defined in terms of contribution to overall disparity. We thus identify serving data whose inclusion increases overall fairness disparity relative to the original training data, while regions that reduce or compensate disparity are not considered harmful. *HarmScore* combines normalised accuracy loss and fairness degradation into a single continuous measure.

For a cluster $C^{(j)}$ with instances from D_0 and D_s, *HarmScore* is calculated as:

$$\begin{aligned} \mathit{HarmScore}(C^{(j)}) = (1-\lambda)\,\Delta\mathrm{Acc}_{\mathrm{norm}}(C^{(j)}) \\ + \lambda\,\Delta\mathrm{Fair}_{\mathrm{norm}}(C^{(j)}) \end{aligned} \tag{6}$$

HarmScore increases when $\Delta\mathrm{Acc}(C^{(j)}) > 0$ (accuracy drops) or $\Delta\mathrm{Fair}(C^{(j)}) > 0$ (fairness worsens). High *HarmScore* indicates data regions that introduce harmful shifts and should be prioritised for retraining. Low *HarmScore* indicates stable or beneficial regions. This unified formulation provides a simple and interpretable way to balance accuracy recovery and fairness preservation, without hand-crafted case rules.

We set a threshold τ to control the severity of degradation. After calculating *HarmScore* of each cluster (line 15 in Algorithm 1), clusters with $\mathit{HarmScore}(C^{(j)}) > \tau$ are flagged as harmful shifts (see lines 16 to 18); a smaller τ selects more data from D_s. We collect their new instances into a subset D_R and use it to retrain f_0, improving its performance.

4.2 C-SHIFT Chunk-wise Variation

Our **C-SHIFT** framework extends naturally to a chunk-wise setting, where serving data arrive as a sequence of chunks D_{s_1}, $D_{s_2}, \ldots, D_{s_t}$. We use a lightweight *clustering-merge* procedure to update the model incrementally (see Algorithm 2). At step t, we train a clustering model C_t on the latest chunk D_{s_t} and merge it with the existing model C_{t-1} built from earlier chunks. Our goal is to study harmful data drift in the serving data; clusters provide a natural unit for tracking accuracy and the group-fairness metric in Equation (5) over time. When a cluster in C_{t-1} cannot be matched to any data in D_{s_t} under the conditions detailed in the *Merge Strategy*, we discard it for this update, since our focus is on the new chunk. This approach avoids re-clustering all accumulated data from scratch and reduces computation time and memory usage.

Merge Strategy. For each cluster in C_t, we find its nearest cluster in C_{t-1} by centroid distance and consider a one-to-one merge, subject to a distance constraint and a label-compatibility check. We precompute pairwise centroid distances

$$D[i,j] \;=\; \big\|\mu_t^{(i)} - \mu_{t-1}^{(j)}\big\|_2, \qquad i \in \mathcal{I}_t,\; j \in \mathcal{J}_{t-1},$$

where $\mu_t^{(i)}$ and $\mu_{t-1}^{(j)}$ are cluster centroids, and $\mathcal{I}_t$ and $\mathcal{J}_{t-1}$ index the clusters of C_t and C_{t-1}, respectively.

Distance constraint. As mentioned in Section 4.1.1, we constrain cluster diameter by a threshold r so that any two points within a cluster are at most r apart; this yields controllable clusters for analysis. We therefore keep only pairs with $D[i,j] \le r$ and process them in ascending order of distance (Algorithm 2, line 3). Because merges are one-to-one, we use lazy invalidation to skip any pair whose indices have already been matched (line 6). To ensure the merged cluster respects this constraint, we add a near-subset check (line 8): for any candidate pair (i,j) with $D[i,j] \le r$, define

$$S \;=\; \big\{\, x \in C_t^{(i)} : \|x - \mu_{t-1}^{(j)}\|_2 \le r/2 \,\big\},$$

which ensures only new points within distance $r/2$ of the previous centroid $\mu_{t-1}^{(j)}$ to be merged into $C_{t-1}^{(j)}$ (lines 10 to 12). The existing cluster $C_{t-1}^{(j)}$ is already confined to radius $r/2$ around the same centroid (see section 4.1.1). Thus, for any merged new point x and any existing point $y \in C_{t-1}^{(j)}$, $\|x-y\|_2 \le \|x-\mu_{t-1}^{(j)}\|_2 + \|\mu_{t-1}^{(j)} - y\|_2 \le r$, so the merged set has diameter at most r. We then merge using only S (subject to label compatibility) and keep $C_t^{(i)} \setminus S$ as a standalone cluster if it is non-empty (see line 12).

Label compatibility. Let the dominant (predicted) class of a set C be

$$y_{\mathrm{dom}}(C) \;=\; \arg\max_{y \in \mathcal{Y}} \big|\{(\mathbf{x}, y') \in C : y' = y\}\big|.$$

We merge S with $C_{t-1}^{(j)}$ only if $y_{\mathrm{dom}}(S) = y_{\mathrm{dom}}(C_{t-1}^{(j)})$ (Algorithm 2, line 9); otherwise, $C_t^{(i)}$ remains standalone. This restriction prevents mixing clusters with different label contexts, which could harm accuracy and fairness, and it preserves more homogeneous groups for interpretation and subsequent fairness analysis. When the merging processes within each cluster finish, the unmatched clusters are directly added to the final cluster list. (lines 16 to 18).

After obtaining the merged model, we apply our problematic-shift detection (Algorithm 1) to the current cluster model C_m to identify harmful samples. Depending on the severity of the detected shift, we either retrain on a curated dataset or fine-tune the model.

Complexity analysis. Let k_t and k_{t-1} be the numbers of clusters in C_t and C_{t-1}, and let d be the data dimension. Computing all pairwise centroid distances costs $O(k_t k_{t-1} d)$ time. Building the ordered list of candidate pairs costs $O(k_t k_{t-1} \log(k_t k_{t-1}))$ in the worst case. Let $q = \max_{i \in \mathcal{I}_t} |C_t^{(i)}|$ be the maximum cluster size in C_t. Because merges are one-to-one, there are at most k_t merges. For each accepted merge, forming the near-subset requires checking distances from points in $C_t^{(i)}$ to $\mu_{t-1}^{(j)}$, which costs $O(qd)$. Overall,

$$O\big(k_t k_{t-1}(d + \log(k_t k_{t-1})) \;+\; k_t\, q\, d\big)$$

time and $O(k_t k_{t-1})$ space to store the distance list.

5 Experimental Setup

When the serving data is even partially OOD, the naive approach to repairing a model with respect to accuracy and fairness is to retrain using both training and serving data. This incurs not only the cost of retraining, but also the cost of acquiring suitable ground truth annotations for the entire serving data. Our evaluation aims to show that **C-SHIFT** drastically reduces this cost, by precisely identifying regions of the serving data that are *harmful*, as defined above, and thus enabling either localised retraining or fine-tuning of the current model. In this section, we corroborate this hypothesis using an extensive experimental space that combines (i) four datasets, (ii) two types of models: Logistic Regression (LR), and Multi-Layer Perceptron (MLP), (iii) three baseline methods for partitioning the data space into regions, and (iv) three established fairness adjustment strategies that operate on those regions.

Algorithm 2 Clustering Model Merge

Input: Cluster models C_t, C_{t-1} with indices sets $\mathcal{I}_t, \mathcal{J}_{t-1}$, class label y, and distance threshold d
Output: Merged model C_m

$C_m \leftarrow \emptyset$; $U_t \leftarrow \emptyset$; $U_{t-1} \leftarrow \emptyset$
// **Precompute distances as key→value**
$Q \leftarrow \{(i,j) : i \in \mathcal{I}_t, j \in \mathcal{J}_{t-1}, \|\mu_t^{(i)} - \mu_{t-1}^{(j)}\|_2 \le r\}$ ascending order by distance
while Q not empty **do**
  $(i,j) \leftarrow \text{popMin}(Q)$
  if $i \in U_t$ **or** $j \in U_{t-1}$ **then continue** *// lazy skip*
  end if
  $S \leftarrow \{x \in C_t^{(i)} : \|x - \mu_{t-1}^{(j)}\|_2 \le r/2\}$ *// near-subset*
  if $S \neq \emptyset$ **and** $y_{\text{dom}}(S) = y_{\text{dom}}(C_{t-1}^{(j)})$ **then**
    C_m.addCluster$(S \cup C_{t-1}^{(j)})$
    $U_t \leftarrow U_t \cup \{i\}$; $U_{t-1} \leftarrow U_{t-1} \cup \{j\}$
    if $C_t^{(i)} \setminus S \neq \emptyset$ **then** C_m.addCluster$(C_t^{(i)} \setminus S)$
    end if
  end if
end while
for each $i \in \mathcal{I}_t \setminus U_t$ **do**
  C_m.addCluster$(C_t^{(i)})$ *// add unmatched clusters*
end for
return C_m

The experiments are implemented using common scikit-learn python libraries [27] as well as the AI Fairness 360 (AIF 360) library [5] for fairness adjustments. All experiments are performed on a Linux server with Intel®Xeon ®Platinum 8570 CPUs.

5.1 Datasets

We use two types of datasets and corresponding distribution shifts: (i) Real and synthetic baselines with simulated shifts and (ii) Real baseline with real shift.

Regarding the former, we follow common practice in research on drift analysis, where specific data shifts are deliberately introduced into baseline datasets. This allows us to test **C-SHIFT** under varying severity of the divergence between the In-Distribution (ID) training and test set, and the Out-Of-Distribution (OOD) test set. Specifically, given a baseline dataset D_0, we use a portion D_0^{tr} to train a corresponding model f_0, such that evaluating f_0 on the ID test set D_0^{test} achieves satisfying performance, i.e., high predictive accuracy and low unfairness. We then modify D_0 to simulate a shifted version, D_s, and this provides the OOD test set D_s^{test} required to assess the effectiveness and efficiency of **C-SHIFT**, as evaluating f_0 against D_s^{test} is expected to produce lower accuracy, worse fairness, or both. We conduct experiments using the three datasets listed in Table 1. Corresponding results are presented in Sec. 6.

For a real baseline using real data shift, we have used the New York Stop-and-Frisk (NYSF) dataset [22], describing policing activities in New York City with racial bias between 2004 and 2024. Assuming that drift continues over time, we have chosen the first and last years, 2004 and 2024, which capture the extreme data drifts, to demonstrate its effect on fairness. We report the results in Appendix (Table 6 and 7).

Table 1: Summary of datasets used in the experiments.

Dataset	Total Size	Selected Data Size	Description
ACSIncome [15]	MO: 41,664 rows NJ: 52,067 rows	Original: Total MO data Serving: 10,000 NJ samples	Predict whether an individual's yearly income exceeds $50K.
COMPAS [2]	5,278 samples	Original: 1,700 samples Serving: 1,910 samples	Assess recidivism risk for pretrial release decisions.
Synthetic [36]	4,000 samples	Original: 2,000 samples Serving: 2,000 samples	Generated under fairness-aware experimental settings for bias analysis.
NYPD [26] (Details see A.2)	685,724 samples	Original: 10,506 samples of year 2017 Serving: 10,000 samples of year 2024	Record New York City stop-and-frisk encounters (2004-2024) with documented racial bias patterns.

5.2 Generating simulated data shifts

ACSIncome. [15], described in Sec. 1, contains US population data demographics, partitioned by State. The classification task is predicting whether an individual's yearly income exceeds $\$50K/year$. Each pair of States provides a natural simulation of data shift, namely a In-Distribution (ID) training and Out-Of-Distribution (OOD) test pair of datasets. To identify the most harmful shifts between two States S_1, S_2, we introduce a simple measure of Performance Distance in accuracy and fairness when using ID and OOD test datasets from S_1 and S_2, respectively. This approach provides the exact metric we need for the experiments, with the advantage of being agnostic to the specific nature of the shift, as we only measure its effect on the model. For simplicity, we have used datasets from 22 States, containing more than 30,000 records. Performance Distance *PD* is defined as:

$$PD = A_{\text{ID}} - A_{\text{OOD}} + F_{\text{OOD}} - F_{\text{ID}} \tag{7}$$

where A_{ID} and A_{OOD} denote the accuracy on ID and OOD, respectively, and F_{ID} and F_{OOD} denote the corresponding fairness metrics (DP). For simplicity, to simulate the changes of underlying fairness patterns in a model, we apply the basic Reweighting [21] method as the fairness adjustments here. After calculating all distances for each pair of States, we find the top-3 pairs:

$$\text{MO} \rightarrow \text{NJ}(0.116), \quad \text{MA} \rightarrow \text{IN}(0.093), \quad \text{TN} \rightarrow \text{NJ}(0.087),$$

in the form $\langle trainingState \rangle \rightarrow \langle testState \rangle (distance)$. Following this analysis, we use the MO → NJ pair for evaluation on the ACSIncome dataset as it provides the most extreme scenario, though others are also available.

COMPAS. [2] is one of the more widely used model fairness benchmark datasets. It holds data about convicted individuals, and the task is to predict the risk of re-offending if an individual is released on parole. To simulate shifts, we split the dataset into k disjoint age groups $\{G_1, G_2, \ldots, G_k\}$ based on age quantiles. Following a pattern similar to ACSIncome, after standard preprocessing on each age group G_i, we train pairs of models using LR and LR+RW models, and calculate a matrix where each model is evaluated on the test set for each other group. The pair that exhibits the max distribution shift is chosen for the experiment. In our experiments, we use $k = 3$.

Synthetic data. To generate synthetic data, we follow the approach in [36]. Two-dimensional datasets of the form (x_1, x_2) with outcome y are generated using multivariate Gaussian distributions: sample (x_1, x_2, y) is drawn from class-conditional Gaussian distributions:

$$(x_1, x_2)|y = 1 \sim \mathcal{N}((2,2)^T, \Sigma_1), (x_1, x_2)|y = 0 \sim \mathcal{N}((-2,-2)^T, \Sigma_0),$$

where $\Sigma_0 = [[10, 1], [1, 3]]^T$ and $\Sigma_1 = [[5, 1], [1, 5]]^T$.

To introduce bias in the sensitive attribute z, first, a rotation transformation is applied to the feature coordinates: $\mathbf{x}' = R(\pi/k)\mathbf{x}$, where $R(\theta) = [[\cos\theta, -\sin\theta], [\sin\theta, \cos\theta]]^T$, k denotes the rotation factor that controls induced bias.

Then, the sensitive attribute z is generated probabilistically based on the rotated features (x_1', x_2'). For each sample, we compute the probability densities under both class-conditional distributions: $p_1 = \mathcal{N}(x_i' \mid \mu_1, \Sigma_1), \quad p_2 = \mathcal{N}(x_i' \mid \mu_2, \Sigma_2)$ and normalise them to get $\tilde{p}_1$ and $\tilde{p}_2$. The sensitive attribute z is assigned according to $z_i \sim \text{Bernoulli}(\tilde{p}_1)$.

We create synthetic datasets with 2,000 samples each, where the rotation factor k controls the degree of data biases. The original training data uses $k = 4$, while the serving data uses $k = 2$, introducing different systematic biases that affect the relationship between sensitive attributes and class labels. Each dataset contains binary labels (positive/negative) and a binary sensitive attribute $z \in \{0, 1\}$, where $z = 0, z = 1$ represents protected / unprotected groups, respectively.

5.3 Model Training and Model Metrics

We evaluate our method using Logistic Regression (LR) and Multi-Layer Perceptron, abbreviated as NN (Neural Network) in the following. Given an ID, OOD test set pair D_0, D_t, model training on D_0 follows the standard 60%/20%/20% split approach for training/validation/test sets with 5-fold cross-validation to set hyperparameters, and optimising for fairness. The experimental serving data is randomly sampled from the full serving dataset. For experiments involving random selection, we conduct multiple independent runs to account for stochastic variation. We use Demographic Parity (DP) [18] and Equalised Odds (EO) [19], defined earlier, as our fairness objectives. They are calculated as: $\text{DP} = \max_{z\in\mathcal{Z}} |\Pr(\hat{y} = 1 \mid z = z) - \Pr(\hat{y} = 1)|$, and $\text{EO} = \max_{z\in\mathcal{Z}, y\in\mathcal{Y}} |\Pr(\hat{y} = y \mid z = z, y = y) - \Pr(\hat{y} = y \mid y = y)|$.

Models are then evaluated separately on D_s, D_t, the training and test are repeated with different random seeds, and the average Accuracy and Fairness performance pairs for each set of experiments are recorded.

5.4 Baselines

We compare **C-SHIFT** against the following baselines.

- **Fully Retrained** refers to globally retraining using all serving data points in a batch.
- **Random Select** randomly picks a number of serving data points equal to the number of data points selected by **C-SHIFT**. In practice, this is **C-SHIFT** with random selection of data points, as opposed to those identified as harmful by the clustering.
- Data Distributions with Low Accuracy (DDLA) detection [16], a method that provides precise selection of harmful data points in the serving data, but is driven entirely by accuracy and ignores fairness.

To each of the baseline methods, we apply the following fairness-adjustment strategies: (1) Reweighting method according to the expected fair (group, label) combination [21] globally. (2) Adversarial Debiasing [37] optimises a classifier to improve prediction accuracy while limiting the extent to which an adversary can exploit predictions to identify sensitive attributes. (3) Reduction [1] reframes fairness constraints as standard predictive learning tasks by applying techniques such as reweighting or resampling.

5.5 Restoring the model

Once the harmful data samples have been selected, we compare two model update methods aimed at restoring accuracy and fairness. Firstly, **Retrain from scratch** simply involves fully retraining the model from scratch, using the original training dataset along with the selected samples from serving data. In contrast, **Fine-tuning** employs incremental training, which is available for our target models (LR and NN), to efficiently update the original model. To achieve this, we utilise the "warm_start" parameter on the underlying scikit-learn estimator and invoke the fairness-aware algorithms with the new data, allowing the model to incrementally adjust fairness constraints. More specifically, when fine-tuning an LR model, the current coefficients and intercept from the existing model are used as a starting point for training the new model. Then, the model is further trained on the target dataset with a limited number of iterations. This procedure utilises a mechanism with parameter initialisation from the pre-trained model, allowing continued optimisation from previous coefficients and intercepts [9]. Besides, the NN model is fine-tuned in a similar way, starting from the initial weights and biases of each layer. To prevent catastrophic forgetting, the learning rate is reduced, and the number of training iterations is constrained. This strategy allows the MLP to gradually adapt its representations to the target dataset while preserving knowledge from the source domain [35].

6 Results

6.1 Effectiveness and Efficiency

Our main claims are that **C-SHIFT** achieves two key advantages: (1) it maintains both accuracy and fairness under distribution shift; (2) it reduces computational cost by selecting fewer samples while matching the comparable performance under two strategies, namely retraining from scratch, and fine-tuning. Regarding the retraining strategy, the main results supporting these claims are presented in Tables 2 and 3. Due to space constraints, similar results for the synthetic dataset, which simulates only severe fairness problems after data shifts but exhibit only minor accuracy loss, can be found in Appendix A.1. Results for the fine-tuning strategy are in Table 4 for all three datasets. Please note that in these Tables, **Fully Retrained-F** and **Random Select-F** are the fairness-aware versions of the respective baseline methods, with the corresponding Fairness Adjustment method in the first column. Also, we omit the *samples* column as the retained harmful data after applying **C-SHIFT** is the same as retraining scenarios.

We observe similar result patterns when using the retraining strategy. The first three rows without fairness interventions show the highest accuracy, but with poor fairness performance. Inside this, the accuracy with fully retraining is considered the real upper bound on each dataset. Meanwhile, the DDLA method, which can select much less data than the whole batch, does not take fairness into account, again resulting in poor fairness performance. For the results that involve Reweighting (RW) [21], Reduction (RD) [1], and Adversarial Debiasing (AD) [37] approaches, *Random Select-F* always shows worse performance compared to *Fully Retrained-F*. In comparison, our **C-SHIFT** consistently obtains comparable results on effectiveness with *Fully Retrained-F*, while improving efficiency through retraining with much fewer samples to improve efficiency.

Regarding the fine-tuning strategy, we observe that the results in Table 4 are similar to those in Table 2, 3, and 5. Fine-tuning

Table 2: Performance evaluation on ACSIncome-New Jersey (NJ) test data with original data from Missouri (MO) using retraining from scratch w.r.t. efficiency and effectiveness metrics on LR and NN.

Fairness Adjustment	Method	LR					NN				
		Efficiency		Effectiveness			Efficiency		Effectiveness		
		Samples↓	Time (s)↓	Acc↑	DP↓	EO↓	Samples↓	Time (s)↓	Acc↑	DP↓	EO↓
None	Random Select	11.286%	.143±.029	.745±.003	.131±.001	.157±.004	10.010%	3.045±1.007	.759±.005	.112±.007	.123±.009
	Fully Retrained	100.000%	.165±.007	.762±.001	.128±.001	.142±.001	100.000%	3.273±1.223	.769±.006	.104±.004	.108±.010
	DDLA	11.286%	.131±.012	.750±.001	.136±.004	.177±.005	10.010%	2.619±1.278	.754±.006	.119±.003	.140±.004
Reweighting [21]	Random Select-F	35.030%	.137±.018	.731±.004	.076±.002	.055±.004	33.900%	**5.300±1.281**	.747±.006	.058±.007	.020±.010
	Fully Retrained-F	100.000%	.164±.017	**.751±.001**	**.061±.002**	**.036±.002**	100.000%	8.824±4.772	**.762±.004**	.042±.003	**.006±.003**
	C-SHIFT	**35.030%**	**.137±.007**	.745±.005	.063±.002	.049±.005	**33.900%**	5.434±2.841	.758±.009	**.041±.006**	.010±.009
Reduction [1]	Random Select-F	36.047%	**1.805±.155**	.738±.002	.043±.001	.016±.001	33.437%	81.949±9.005	.740±.006	.048±.002	.013±.003
	Fully Retrained-F	100.000%	2.430±.165	**.758±.001**	.031±.001	**.007±.001**	100.000%	110.922±11.021	**.759±.006**	**.031±.003**	.008±.005
	C-SHIFT	**36.047%**	1.836±.136	.753±.002	**.030±.001**	.008±.001	**33.437%**	**77.528±20.192**	.751±.005	.036±.004	**.008±.003**
Adversarial Debiasing [37]	Random Select-F	34.070%	6.273±.531	.737±.005	.070±.007	.047±.004	33.970%	6.185±.672	.738±.005	.070±.007	.051±.012
	Fully Retrained-F	100.000%	7.536±.130	**.760±.003**	.068±.004	.065±.009	100.000%	7.558±.155	**.760±.003**	.068±.005	.065±.009
	C-SHIFT	**34.070%**	**6.084±.415**	.753±.007	**.059±.003**	**.035±.014**	**33.970%**	**6.051±.345**	.755±.005	**.060±.010**	**.035±.012**

Table 3: Performance evaluation on COMPAS-serving test data with original data from COMPAS-original using retraining from scratch w.r.t. efficiency and effectiveness metrics on LR and NN.

Fairness Adjustment	Method	LR					NN				
		Efficiency		Effectiveness			Efficiency		Effectiveness		
		Samples↓	Time (s)↓	Acc↑	DP↓	EO↓	Samples↓	Time (s)↓	Acc↑	DP↓	EO↓
None	Random Select	30.060%	.006±.001	.597±.005	.158±.010	.139±.010	31.070%	.274±.050	.630±.010	.234±.015	.230±.015
	Fully Retrained	100.000%	.006±.001	.622±.005	.145±.010	.148±.010	100.000%	.289±.050	.640±.010	.280±.020	.249±.020
	DDLA	30.060%	.006±.001	.614±.005	.125±.008	.123±.008	31.070%	.274±.050	.627±.010	.205±.015	.196±.015
Reweighting [21]	Random Select-F	32.765%	.060±.030	.595±.006	.079±.016	.153±.012	32.533%	.143±.060	.614±.009	.062±.036	.092±.048
	Fully Retrained-F	100.000%	.075±.041	**.618±.001**	.061±.005	.142±.010	100.000%	.190±.023	**.619±.012**	.074±.051	.107±.007
	C-SHIFT	**32.765%**	**.055±.029**	.617±.006	**.043±.027**	**.123±.033**	**32.533%**	**.197±.139**	.600±.044	**.034±.008**	**.081±.035**
Reduction [1]	Random Select-F	24.622%	.089±.027	.499±.033	.099±.015	.087±.012	21.933%	1.183±.166	.487±.041	.091±.033	.082±.029
	Fully Retrained-F	100.000%	.124±.010	**.597±.003**	.114±.004	.133±.006	100.000%	1.752±.079	**.612±.006**	.110±.011	.088±.015
	C-SHIFT	**24.622%**	**.087±.106**	.579±.069	**.037±.031**	**.036±.023**	**21.933%**	**1.014±.090**	.581±.067	**.053±.040**	**.047±.026**
Adversarial Debiasing [37]	Random Select-F	37.533%	.616±.136	.597±.010	.078±.044	.081±.046	36.232%	.568±.091	.601±.007	.071±.027	.080±.041
	Fully Retrained-F	100.000%	.758±.088	**.623±.008**	**.033±.010**	**.037±.022**	100.000%	.683±.002	**.633±.004**	**.032±.012**	**.033±.017**
	C-SHIFT	**37.533%**	**.571±.052**	.606±.010	.037±.033	.041±.012	**36.232%**	**.562±.050**	.621±.012	.034±.020	.036±.012

models requires less time to update compared to retraining under the same fairness adjustments on the same dataset.

In summary, all three Tables show that **C-SHIFT** consistently requires much less data from the full dataset for both retraining and fine-tuning, substantially reducing the data annotation burden in other application scenarios such as active learning [30]. Compared to the Fully Retrained-F baseline that processes the entire new data chunk, our approach achieves comparable predictive accuracy and fairness metrics while using significantly fewer labeled samples, while it outperforms *Random Select-F* in both accuracy and fairness. The benefits of **C-SHIFT** remain robust across different fairness adjustment techniques and model architectures (both LR and NN). This consistency validates that harmful shift detection can be a general principle for efficient fairness-aware model adaptation.

6.2 Results on a Sequence of Data Chunks

To demonstrate the effectiveness of the proposed method under realistic streaming settings, we design a sequential chunk processing experiment where data arrive over time. Specifically, by leveraging the same method simulating data drifts on ACSIncome [15] in Sec. 5.2, the serving dataset is then randomly shuffled and partitioned into $N = 5$ non-overlapping data chunks without replacement, each containing 5,000 samples. The chunks are processed sequentially to simulate a streaming environment.

At each time step, the newly arrived data chunk is incorporated into the learning process. For simplicity, we conduct the experiments on the LR model with reweighting as the fairness adjustments and retraining from scratch (with the same setting in Table 2). We compare the fully retrained baseline approach with our C-SHIFT method. For each chunk, we evaluate the model performance on the cumulative dataset using three metrics: classification accuracy, demographic parity (DP), and equalized odds (EO). In addition, we report the selected number of samples processed over time.

Figure 3 summarises the experimental results under the sequential data chunk setting. As shown in the first graph, the accuracy of C-SHIFT closely tracks that of the fully retrained baseline across all data chunks. The second graph reports the fairness metrics. Across all chunks, C-SHIFT achieves fairness levels comparable to those of the fully retrained model. These results suggest that the proposed method is able to effectively control bias under streaming data distributions, while maintaining accuracy as new data are incorporated. At last, the third graph illustrates the cumulative number of samples processed over time. As expected, the sample size selected by C-SHIFT is much less than the whole data chunk.

Table 4: Performance evaluation across three datasets with fine-tuning w.r.t. efficiency and effectiveness metrics on LR and NN models.

Dataset	Fairness Adjustment	Method	LR				NN			
			Efficiency	Effectiveness			Efficiency	Effectiveness		
			Time (s)↓	Acc↑	DP↓	EO↓	Time (s)↓	Acc↑	DP↓	EO↓
ACSIncome [15]	Reweighting [21]	Random Select-F	.030±.004	.745±.003	.066±.004	.051±.004	.775±.411	.752±.002	.060±.013	.031±.009
		Fully Retrained-F	.032±.004	**.766±.001**	**.048±.001**	**.029±.002**	2.137±1.097	**.770±.004**	.050±.002	.022±.007
		C-SHIFT	**.028±.008**	.757±.005	.052±.004	.032±.004	**.767±.356**	.769±.004	**.034±.010**	**.022±.006**
	Reduction [1]	Random Select-F	.781±.163	.743±.004	.045±.005	.035±.008	18.305±12.378	.754±.021	.036±.008	.032±.010
		Fully Retrained-F	1.096±.179	**.762±.002**	.036±.003	.021±.005	48.278±8.696	**.775±.002**	.031±.006	.029±.006
		C-SHIFT	**.767±.132**	.760±.003	**.034±.009**	**.018±.004**	**15.542±9.821**	.767±.086	**.029±.012**	**.018±.001**
	Adversarial Debiasing [37]	Random Select-F	6.988±.224	.747±.005	.067±.005	.062±.007	7.607±.689	.751±.005	.067±.005	.062±.007
		Fully Retrained-F	7.638±.132	**.757±.000**	**.047±.003**	**.029±.006**	7.737±.223	**.757±.000**	**.047±.003**	**.029±.006**
		C-SHIFT	**6.973±.167**	.755±.004	.053±.006	.034±.014	**7.364±.837**	.755±.007	.047±.006	.034±.014
COMPAS [2]	Reweighting [21]	Random Select-F	.013±.013	.609±.003	.047±.007	.027±.010	.156±.180	.625±.026	.082±.048	.078±.041
		Fully Retrained-F	.014±.013	**.654±.003**	.038±.002	**.014±.002**	.235±.092	**.643±.006**	**.037±.006**	**.065±.004**
		C-SHIFT	**.012±.015**	.649±.000	**.031±.000**	.015±.000	**.140±.042**	.632±.041	.047±.073	.070±.046
	Reduction [1]	Random Select-F	.104±.066	.652±.017	.087±.017	.072±.018	1.275±.429	.639±.030	.056±.019	.038±.007
		Fully Retrained-F	.129±.002	**.676±.001**	.082±.006	.054±.006	2.155±.311	**.656±.015**	.087±.019	.054±.015
		C-SHIFT	**.064±.003**	.660±.003	**.050±.002**	**.036±.001**	**.905±.168**	.649±.023	**.043±.014**	**.027±.021**
	Adversarial Debiasing [37]	Random Select-F	.729±.119	.615±.004	.071±.020	.069±.042	.724±.114	.617±.005	.078±.019	.062±.034
		Fully Retrained-F	.785±.007	.626±.001	**.056±.010**	.052±.021	.812±.007	.626±.001	**.052±.064**	.057±.009
		C-SHIFT	**.661±.024**	**.627±.006**	.058±.044	**.041±.046**	**.719±.101**	**.627±.006**	.060±.044	**.051±.046**
Synthetic [36]	Reweighting [21]	Random Select-F	.002±.000	.870±.003	.165±.004	.080±.004	.063±.020	.878±.007	.104±.023	.080±.032
		Fully Retrained-F	.002±.000	**.880±.001**	.164±.003	.083±.004	.086±.012	**.888±.001**	.102±.009	.085±.004
		C-SHIFT	**.002±.000**	.873±.000	**.108±.010**	**.070±.034**	**.063±.017**	.878±.016	**.091±.040**	**.073±.031**
	Reduction [1]	Random Select-F	**.057±.002**	.827±.017	.135±.083	.108±.024	2.629±2.175	.850±.027	.137±.019	.151±.027
		Fully Retrained-F	.066±.004	**.830±.000**	**.112±.000**	**.090±.000**	2.190±.120	**.859±.015**	.120±.022	.094±.023
		C-SHIFT	.058±.006	.829±.001	.123±.004	.090±.008	**1.216±.097**	.846±.001	**.113±.006**	**.073±.005**
	Adversarial Debiasing [37]	Random Select-F	.682±.109	.880±.002	.120±.005	.081±.005	.659±.013	.880±.002	.120±.005	.081±.005
		Fully Retrained-F	.771±.014	**.884±.001**	**.106±.001**	**.071±.004**	.774±.009	**.884±.001**	**.106±.001**	.071±.002
		C-SHIFT	**.648±.024**	.880±.002	.110±.004	.071±.005	**.640±.012**	.880±.004	.107±.004	**.069±.005**

Overall, these results demonstrate that C-SHIFT can achieve performance and fairness comparable to full retraining in sequential data settings, while avoiding the computational cost associated with retraining models from scratch at each time step. This highlights the practicality of the proposed method for real-world applications involving continuous data arrival.

6.3 Visualisation of Harmful Data

To intuitively demonstrate which portions of data **C-SHIFT** can identify, we conduct experiments on synthetic datasets following the same simulation described in Sec. 5.2. Figure 4 presents the results across four panels. The first two plots from the left show the distribution of classes and sensitive groups under data generation with $k = 4$ and $k = 2$. Since the sensitive attribute z of different k are all assigned through probabilities on which Gaussian distribution the data instances belong to after simple rotation from the initial data, we can clearly identify which portions of the data exhibit group discrimination when transitioning from $k = 4$ to $k = 2$. These biased instances observed with the ground truth are marked with colored points in the third column (Ground Truth Biased Data), while unbiased instances are shown in gray. The fourth column displays the data identified as harmful shifts by **C-SHIFT** with parameters $\lambda = 0.5$, HarmScore threshold $\tau = 0.3$, and distance threshold $r = 3.5$. These parameters are chosen to effectively detect the most harmful regions. Notably, the dense colored regions in the third column (ground truth) align well with those detected in the fourth column, with our method regarding 72.4% of the instances as severely problematic, concentrating in specific regions of the feature space where distributional shifts most severely impact fairness. This demonstrates **C-SHIFT**'s capability to localise biased data regions in the feature space.

Figure 5 presents the regional harm analysis, where the data feature space is partitioned into spatial regions by **C-SHIFT** and each region's contribution to model performance is quantified by *HarmScore*. *HarmScore* intuitively aggregates the impact degree on model performance (including accuracy and fairness) through color intensity—darker colors indicate higher harm. Each region size is controlled by d, λ defines the tradeoff between accuracy and fairness effects, and τ determines which data points are classified as harmful. Note that the parameters here are selected for visualisation purposes to demonstrate the detection of the most harmful shifts without hyperparameter optimisation.

6.4 Hyperparameter Sensitivity Test

The effectiveness of our approach depends on an appropriate parameter selection. To evaluate the robustness of our framework and provide more practical results for hyperparameter sensitivity, we conduct a sensitivity analysis on the three parameters: distance threshold (r), fairness contribution (λ), and *HarmScore* threshold (H / τ). This analysis examines how variations in each parameter affect fairness metrics, model accuracy, and computational efficiency (measured as retrained samples).

We employ a controlled single-parameter variation methodology to isolate the effects of each hyperparameter, i.e., fixing the other two parameters when testing different configurations on one parameter. The results are shown in Figure 6, 7, and 8.

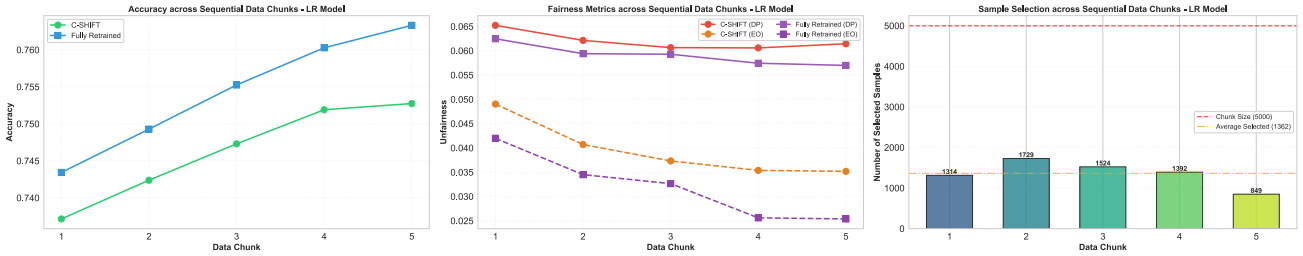

Figure 3: Experimental results under a sequential data chunk setting. Data arrive in five non-overlapping chunks processed sequentially. We report (left) classification accuracy, (middle) fairness metrics including demographic parity (DP) and equalised odds (EO), and (right) the number of samples with colors arranged from darker to lighter as the data sequence progresses.

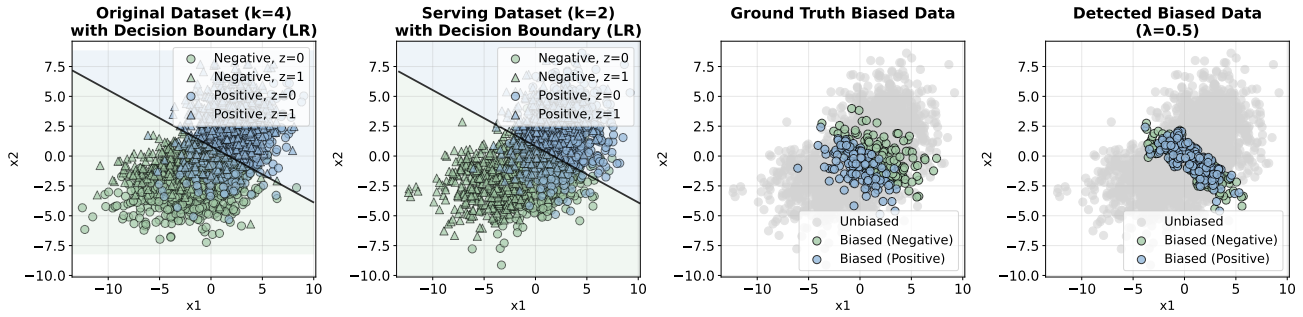

Figure 4: Visualisation of our harmful shift detection on synthetic data, showing that C-SHIFT can localise the problematic data intuitively. Detected harmful data points are marked in blue or green.

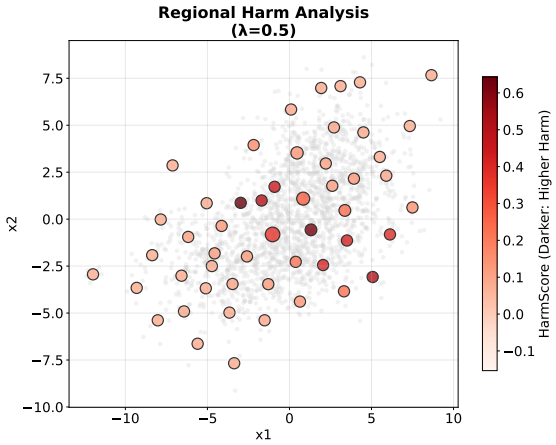

Figure 5: Visualisation of the impact of harmful shift (balance fairness and performance) of each data region. HarmScore aggregates these impacts according to application-specific priorities, where colours indicate regions that disproportionately harm model behaviours under data shifts.

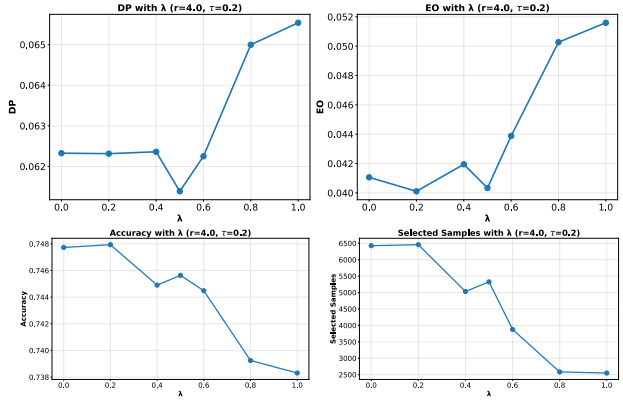

Figure 6: Sensitivity analysis of the fairness trade-off hyperparameter λ with fixed $r = 4.0$, $\tau = 0.2$, showing how varying λ affects model accuracy, fairness metrics, and the number of selected samples.

In Figure 6, we examine the performance changes based on different λ values that define how much the fairness effects contribute to the overall *HarmScore*. We notice that when $\lambda > 0.5$, DP and EP show obvious increases. Additionally, λ serves as an effective control for balancing fairness and performance objectives according to application-specific requirements.

Figure 7 illustrates the impact of the HarmScore threshold τ on performances. As the HarmScore threshold increases, less data will be selected, thus causing the degradation of performance (i.e., lower accuracy and higher biases). This phenomenon can indicate that aggressive filtering with high thresholds can substantially reduce the serving data size, leading to insufficient data for model learning. Notably, with the threshold higher than 0.3, the accuracy and fairness start showing obvious degradation. Additionally, different λ values, denoted by three colours, are also plotted in Figure 7, indicating that higher λ does focus more on fairness problems.

From Figure 8, we can tell that **C-SHIFT** is mostly not sensitive to the distance threshold when the other two parameters are fixed,

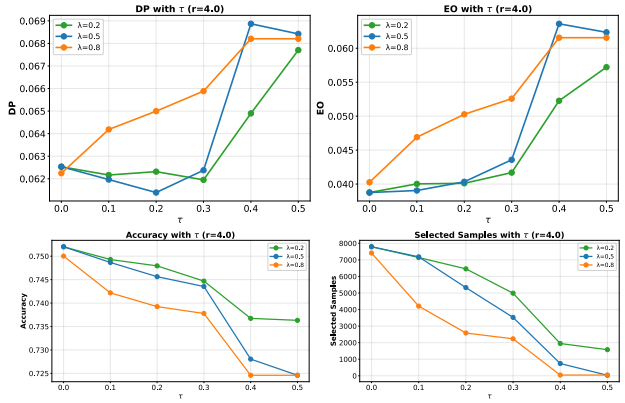

Figure 7: Sensitivity analysis of the hyperparameter HarmScore threshold τ with fixed $r = 4.0$, $\lambda = 0.5$, showing how varying τ affects the metrics.

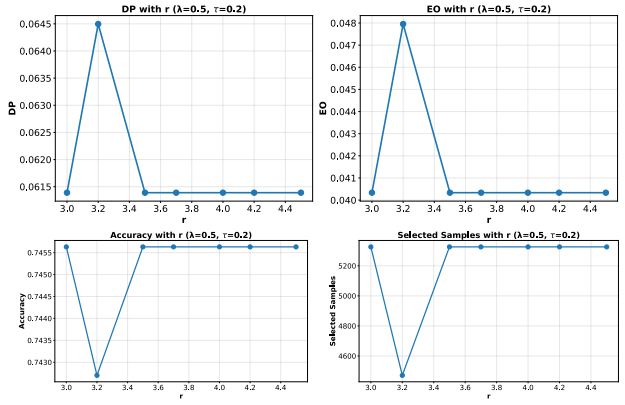

Figure 8: Sensitivity analysis of the hyperparameter distance threshold r with fixed $\lambda = 0.5, \tau = 0.2$, showing how varying r affects the metrics.

indicating that it can refine the sample selection by controlling the sizes of data groups without drastically altering the fairness-accuracy landscape. Consequently, practitioners can tune λ and the HarmScore threshold as primary hyperparameters, while using the distance threshold as a secondary control for fine-grained adjustments.

7 Conclusions

In this work, we have addressed the problem of simultaneously and efficiently restoring model accuracy and fairness in response to data distribution shifts that occur between training and serving data. A naive approach to the problem entails either retraining the model from scratch, or "fine-tuning" the model using the entire serving data. In a deployment setting, such interventions will be recurring with each new batch of serving data, making them inefficient. In contrast, we propose a local approach, **C-SHIFT**, based on the observation that not all of the serving data is equally detrimental to either accuracy or fairness. **C-SHIFT**, a cluster-based method, accurately identifies *harmful* data regions in a batch of serving data and only uses those for retraining or fine-tuning. Experimental results obtained using real-world as well as synthetic datasets indicate that, compared with multiple baselines, substantial savings in the number of data points required to restore the model can be achieved, without sacrificing much accuracy and performance. Finally, our current study mainly focuses on binary classification with a single protected attribute. Extending C-SHIFT to multi-class settings and intersectional fairness definitions is a natural direction for future work. This can be achieved by adopting standard multi-class performance measures and by computing cluster-level fairness over intersectional sensitive groups, without changing the core clustering and HarmScore-based selection mechanism.

Artifacts

The data, implementation code, and test examples are available at: https://github.com/yijieli-cs/harmful_data_shifts.

References

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. In International conference on machine learning. PMLR, Proceedings of Machine Learning Research, Stockholm, Sweden, 60–69.

[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2022. Machine bias. In Ethics of data and analytics. Auerbach Publications, Boca Raton, FL, 254–264.

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2023. Fairness and machine learning: Limitations and opportunities. MIT press, Cambridge, MA.

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

[5] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. https://arxiv.org/abs/1810.01943

[6] Richard Berk. 2012. Criminal justice forecasts of risk: A machine learning approach. Springer Science & Business Media, New York, NY.

[7] Miranda Bogen and Aaron Rieke. 2018. Help Wanted: An Examination of Hiring Algorithms, Equity, and Bias.

[8] Maximilian Böther, Ties Robroek, Viktor Gsteiger, Robin Holzinger, Xianzhe Ma, Pınar Tözün, and Ana Klimovic. 2025. Modyn: Data-centric machine learning pipeline orchestration. Proceedings of the ACM on Management of Data 3, 1 (2025), 1–30.

[9] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers. Springer, Springer, Heidelberg, Germany, 177–186.

[10] Athman Bouguettaya, Qi Yu, Xumin Liu, Xiangmin Zhou, and Andy Song. 2015. Efficient agglomerative hierarchical clustering. Expert Systems with Applications 42, 5 (2015), 2785–2797.

[11] Simon Caton and Christian Haas. 2024. Fairness in Machine Learning: A Survey. ACM Comput. Surv. 56, 7, Article 166 (April 2024), 38 pages. doi:10.1145/3616865

[12] Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. 2019. Slice-based learning: A programming model for residual learning in critical data slices. Advances in neural information processing systems 32 (2019), 13885–13896.

[13] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2019. Automated data slicing for model validation: A big data-ai integration approach. IEEE Transactions on Knowledge and Data Engineering 32, 12 (2019), 2284–2296.

[14] Lee Cohen, Zachary C. Lipton, and Yishay Mansour. 2019. Efficient Candidate Screening under Multiple Tests and Implications for Fairness. arXiv:1905.11361 [cs.LG] https://arxiv.org/abs/1905.11361

[15] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. Advances in Neural Information Processing Systems 34 (2021), 16608–16620.

[16] Sijie Dong, Qitong Wang, Soror Sahri, Themis Palpanas, and Divesh Srivastava. 2024. Efficiently mitigating the impact of data drift on machine learning pipelines. Proceedings of the VLDB Endowment 17, 11 (2024), 3072–3081.

[17] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In Proceedings of the 3rd innovations in theoretical computer science conference. ACM, New York, NY, 214–226.

[18] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, NY, 259–268.

[19] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. Advances in neural information processing systems 29 (2016), 3315–3323.

[20] T Ryan Hoens, Robi Polikar, and Nitesh V Chawla. 2012. Learning from streaming data with concept drift and imbalance: an overview. Progress in Artificial Intelligence 1, 1 (2012), 89–101.
[21] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. Knowledge and information systems 33, 1 (2012), 1–33.
[22] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In International conference on machine learning. PMLR, PMLR, Virtual, 5637–5664.
[23] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. IEEE transactions on knowledge and data engineering 31, 12 (2018), 2346–2363.
[24] Natalia Martinez, Martin Bertran, and Guillermo Sapiro. 2020. Minimax pareto fairness: A multi objective perspective. In International conference on machine learning. PMLR, PMLR, Virtual, 6755–6764.
[25] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. ACM computing surveys (CSUR) 54, 6 (2021), 1–35.
[26] New York Police Department. 2024. Stop, Question and Frisk Data. New York Police Department. https://www.nyc.gov/site/nypd/stats/reports-analysis/reports-landing.page
[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.
[28] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. 2019. Failing loudly: An empirical study of methods for detecting dataset shift. In Advances in Neural Information Processing Systems, Vol. 32. Curran Associates, Inc., Red Hook, NY, 1396–1406.
[29] Svetlana Sagadeeva and Matthias Boehm. 2021. Sliceline: Fast, linear-algebra-based slice finding for ml model debugging. In Proceedings of the 2021 international conference on management of data. ACM, New York, NY, 2290–2299.
[30] Burr Settles. 2009. Active Learning Literature Survey. Technical Report 1648. University of Wisconsin–Madison.
[31] Minglai Shao, Dong Li, Chen Zhao, Xintao Wu, Yujie Lin, and Qin Tian. 2024. Supervised Algorithmic Fairness in Distribution Shifts: A Survey. arXiv:2402.01327 [cs.LG] https://arxiv.org/abs/2402.01327
[32] Mohammad Ahmad Sheikh, Amit Kumar Goel, and Tapas Kumar. 2020. An approach for prediction of loan approval using machine learning algorithm. In 2020 international conference on electronics and sustainable communication systems (ICESC). IEEE, IEEE, Coimbatore, India, 490–494.
[33] Heng Wang and Zubin Abraham. 2015. Concept drift detection for streaming data. In 2015 international joint conference on neural networks (IJCNN). IEEE, IEEE, Killarney, Ireland, 1–9.
[34] Scott Wares, John Isaacs, and Eyad Elyan. 2019. Data stream mining: methods and challenges for handling concept drift. SN Applied Sciences 1 (2019), 1–19.
[35] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? Advances in neural information processing systems 27 (2014), 3320–3328.
[36] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. 2017. Fairness constraints: Mechanisms for fair classification. In Artificial intelligence and statistics. PMLR, PMLR, Fort Lauderdale, FL, 962–970.
[37] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society. ACM, New Orleans, LA, 335–340.
[38] Hantian Zhang, Xu Chu, Abolfazl Asudeh, and Shamkant B Navathe. 2021. Omnifair: A declarative system for model-agnostic group fairness in machine learning. In Proceedings of the 2021 international conference on management of data. ACM, New York, NY, 2076–2088.

A Appendix: Experiments

A.1 Experimental Result on Synthetic Data

Due to space constraints, in Sec. 6.1 we reported results on the synthetic dataset when using fine-tuning. The corresponding results when retraining the model from scratch are presented here in Table 5.

A.2 Experimental Results on New York Police Department (NYPD)

We report additional results on the New York Police Department (NYPD) stop and frisk dataset [26], which is a real dataset from the policing activities in New York City with racial bias, following the same evaluation protocol described in Sec. 6.1. The most recent datasets that contain the same features cover the range from 2003 through 2024. Assuming that drift continues over time, we have chosen the first and last years, 2003 and 2024, which capture the extreme data drifts, to demonstrate its effect on accuracy and fairness. We report the results in Table 6 and 7.

Table 5: Performance evaluation on Synthetic-serving test data with original data from Synthetic-original using retraining from scratch w.r.t. efficiency and effectiveness metrics on LR and NN.

Fairness Adjustment	Method	LR					NN				
		Efficiency		Effectiveness			Efficiency		Effectiveness		
		Samples↓	Time (s)↓	Acc↑	DP↓	EO↓	Samples↓	Time (s)↓	Acc↑	DP↓	EO↓
None	Random Select	5.400%	.007±.001	.880±.005	.246±.010	.126±.010	8.000%	.147±.020	.868±.010	.188±.015	.115±.015
	Fully Retrained	100.000%	.007±.001	.885±.005	.244±.010	.115±.010	100.000%	.151±.020	.887±.010	.183±.015	.130±.012
	DDLA	5.400%	.007±.001	.883±.005	.241±.010	.071±.008	8.000%	.147±.020	.875±.010	.181±.012	.121±.010
Reweighting [21]	Random Select-F	35.933%	.004±.003	.869±.002	.145±.005	.120±.007	38.667%	.132±.063	.878±.005	.150±.010	.117±.014
	Fully Retrained-F	100.000%	.019±.019	**.875±.000**	.155±.000	**.075±.000**	100.000%	.196±.010	**.879±.003**	.144±.010	.108±.001
	C-SHIFT	**35.933%**	**.004±.003**	.870±.002	**.113±.002**	.080±.003	**38.667%**	**.132±.057**	.876±.003	**.139±.001**	**.107±.003**
Reduction [1]	Random Select-F	26.444%	.061±.044	.815±.011	.214±.010	.192±.012	20.156%	2.321±.333	.868±.011	.157±.014	.128±.020
	Fully Retrained-F	100.000%	.102±.020	**.881±.001**	**.142±.001**	**.116±.000**	100.000%	3.629±.602	**.878±.003**	**.134±.007**	**.111±.008**
	C-SHIFT	**26.444%**	**.056±.006**	.864±.006	.154±.006	.122±.001	**20.156%**	**2.159±.290**	.876±.005	.139±.004	.113±.003
Adversarial Debiasing [37]	Random Select-F	13.533%	**.647±.030**	.879±.002	.131±.004	.085±.006	13.533%	.554±.086	.880±.002	.121±.004	.091±.006
	Fully Retrained-F	100.000%	.737±.013	**.883±.002**	**.110±.003**	.075±.003	100.000%	.743±.003	**.883±.002**	**.109±.005**	.073±.002
	C-SHIFT	**13.533%**	.648±.237	.879±.001	.112±.003	**.071±.005**	**13.533%**	**.523±.040**	.879±.001	.112±.003	**.071±.006**

Table 6: Performance evaluation on NYPD test data with original data from year 2003 and serving data from year 2024 using retraining from scratch w.r.t. efficiency and effectiveness metrics on LR and NN.

Fairness Adjustment	Method	LR					NN				
		Efficiency		Effectiveness			Efficiency		Effectiveness		
		Samples↓	Time (s)↓	Acc↑	DP↓	EO↓	Samples↓	Time (s)↓	Acc↑	DP↓	EO↓
None	Random Select	19.820%	.214±.031	.781±.004	.073±.006	.062±.007	19.430%	5.612±1.204	.763±.006	.051±.010	.048±.012
	Fully Retrained	100.000%	.246±.012	.789±.002	.068±.004	.054±.005	100.000%	6.088±1.010	.791±.004	.043±.007	.036±.010
	DDLA	19.820%	.198±.020	.785±.003	.081±.008	.076±.009	19.430%	4.903±1.332	.784±.005	.064±.009	.055±.013
Reweighting [21]	Random Select-F	33.740%	.227±.024	.751±.005	.052±.004	.051±.005	32.980%	9.744±2.110	.756±.007	.054±.010	.058±.012
	Fully Retrained-F	100.000%	.261±.015	**.759±.003**	**.033±.003**	**.030±.004**	100.000%	12.381±2.944	**.766±.005**	**.036±.006**	**.016±.008**
	C-SHIFT	**33.740%**	**.219±.011**	.757±.004	.035±.004	.041±.006	**32.980%**	**9.318±1.982**	.763±.006	.067±.007	.018±.009
Reduction [1]	Random Select-F	35.910%	**2.914±.206**	.744±.004	.060±.003	.021±.003	34.260%	48.352±12.905	.746±.008	.064±.004	.020±.004
	Fully Retrained-F	100.000%	3.302±.241	**.762±.003**	.053±.002	**.015±.002**	100.000%	66.408±15.771	**.763±.007**	**.045±.003**	**.016±.004**
	C-SHIFT	**35.910%**	3.006±.173	.750±.003	**.052±.002**	.016±.002	**34.260%**	**41.227±18.340**	.761±.007	.047±.004	.017±.004
Adversarial Debiasing [37]	Random Select-F	34.120%	8.216±.604	.742±.006	.081±.008	.066±.009	33.840%	8.108±.821	.744±.007	.063±.009	.067±.012
	Fully Retrained-F	100.000%	8.904±.231	**.760±.004**	.056±.006	.049±.010	100.000%	8.975±.266	**.761±.005**	.048±.008	.046±.011
	C-SHIFT	**34.120%**	**7.988±.417**	.758±.005	**.051±.005**	**.049±.012**	**33.840%**	**7.901±.392**	.753±.006	**.043±.007**	**.042±.012**

Table 7: Performance evaluation on NYPD with fine-tuning w.r.t. efficiency and effectiveness metrics on LR and NN models.

Dataset	Fairness Adjustment	Method	LR				NN			
			Efficiency	Effectiveness			Efficiency	Effectiveness		
			Time (s)↓	Acc↑	DP↓	EO↓	Time (s)↓	Acc↑	DP↓	EO↓
NYPD	Reweighting [21]	Random Select-F	.061±.010	.749±.004	.060±.004	.051±.005	2.944±0.910	.745±.006	.055±.010	.029±.010
		Fully Retrained-F	.066±.009	**.764±.003**	**.032±.003**	**.029±.004**	3.701±1.034	**.765±.005**	**.037±.006**	**.020±.008**
		C-SHIFT	**.058±.008**	.758±.004	.035±.004	.033±.006	**2.631±0.802**	.762±.006	.038±.007	.021±.009
	Reduction [1]	Random Select-F	1.183±.240	.755±.004	.060±.003	.021±.003	38.462±10.188	.747±.009	.064±.004	.020±.004
		Fully Retrained-F	1.329±.215	**.762±.003**	.044±.002	**.015±.002**	44.019±11.220	**.764±.008**	**.045±.003**	**.016±.004**
		C-SHIFT	**1.092±.188**	.760±.003	**.042±.002**	.016±.002	**35.908±9.761**	.761±.008	.057±.004	.017±.004
	Adversarial Debiasing [37]	Random Select-F	2.901±.311	.733±.006	.052±.008	.046±.009	3.021±.352	.734±.007	.053±.009	.059±.012
		Fully Retrained-F	3.114±.204	**.740±.004**	.046±.006	.038±.010	3.182±.233	**.741±.005**	.048±.008	.031±.011
		C-SHIFT	**2.736±.271**	.738±.005	**.032±.005**	**.030±.012**	**2.892±.260**	.740±.006	**.043±.007**	**.022±.012**