Detecting Interpretable Subgroup Drifts

Flavio Giobergia*
Politecnico di Torino
Turin, Italy

Eliana Pastor
Politecnico di Torino
Turin, Italy

Luca de Alfaro
University of California, Santa Cruz
Santa Cruz, USA

Elena Baralis
Politecnico di Torino
Turin, Italy

Abstract

The ability to detect and adapt to changes in data distributions is crucial to maintain the accuracy and reliability of machine learning models. Detection is generally approached by observing the drift of model performance from a global point of view. However, drifts occurring in (fine-grained) data subgroups may go unnoticed when monitoring global drift. We take a different perspective, and introduce methods for observing drift at the finer granularity of subgroups. Relevant data subgroups are identified during training and monitored efficiently throughout the model's life. Performance drifts in any subgroup are detected, quantified and characterized so as to provide an interpretable summary of the model behavior over time. Experimental results confirm that our subgroup-level drift analysis identifies drifts that do not show at the (coarser) global dataset level. The proposed approach provides a valuable tool for monitoring model performance in dynamic real-world applications, offering insights into the evolving nature of data and ultimately contributing to more robust and adaptive models.

CCS Concepts

• **Computing methodologies** → *Supervised learning*; *Machine learning approaches*; • **Information systems** → *Data mining*.

Keywords

Drift detection; Interpretability; Subgroup monitoring

ACM Reference Format:
Flavio Giobergia, Eliana Pastor, Luca de Alfaro, and Elena Baralis. 2025. Detecting Interpretable Subgroup Drifts. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709259

1 Introduction

Given the rapid evolution of data, maintaining machine learning model accuracy and reliability is critical. Detecting and adapting to shifts in performance ensure that models remain effective over time. Traditionally, the approaches for detecting drifts in model performance take a global perspective, focusing on overall performance. However, changes in the data distribution or in the relationship between input and target variables may not be widespread across all data, but may affect specific subpopulations. Hence, traditional methods may overlook finer-grained drifts occurring in specific data subgroups.

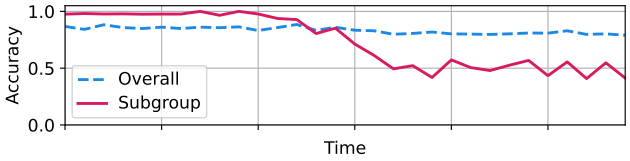

Figure 1: Adult dataset. Drift event affecting the subgroup *women under the age of 36, employed in the private sector* (approx. 10% of the overall population). The overall accuracy is only marginally affected, whereas the drop in performance for the drifting subgroup is more prominent.

Consider as an example the Adult dataset [5]. To predict the income bracket of a person based on their demographic information, a classification model can be trained on the available data. This data represents a snapshot of the socioeconomic conditions at training time. However, these conditions may be subject to changes. Consider, for instance, the (relatively small) subgroup including *women under the age of 36, employed in the private sector*. The subgroup income bracket is initially predicted with near-perfect accuracy by the classification model. Over time, the subgroup exhibits a new behavior: the salary for that segment increases, as the gender pay gap is reduced. This behavior is depicted in Figure 1. While the overall model performance (dashed blue line) is only marginally affected by the drift, the subgroup performance (red line) changes significantly over time. Drift detection methods that only focus on global performance may fail to detect this drift.

Identifying subgroup drifts enables assessing whether the model remains reliable and accurate across subpopulations. Thus, it also allows detecting model unfairness due to drift, as changes in performance may disproportionately affect different demographic groups or other protected subpopulations. Finally, it allows targeted interventions to prevent any subgroup from experiencing disservice or degraded performance over time.

We propose DriftInspector, a methodology that identifies relevant subgroups during model training, on the basis of the interpretable features of the data. It aims to (i) detect drift in data

*Corresponding author – flavio.giobergia@polito.it

KDD '25, Toronto, ON, Canada.

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709259

subgroups, (ii) describe the subgroups affected by drift in an interpretable manner, and (iii) quantify the drift affecting them. We develop algorithms that allow the efficient monitoring of subgroup performance throughout model lifetime, enabling the timely detection and quantification of subgroup-level performance drifts.

We characterize each data instance through interpretable features, possibly extracted from data. For instance, in image data, the interpretable features can include metadata such as the image source, location, creation date, features derived from the caption, or high-level descriptors like whether the photo was taken indoors or outdoors. Similarly, for tabular data, we can use as features either the attribute values themselves, or coarser representations to enhance interpretability (e.g., aggregating precise income values into ranges).

DriftInspector produces an interpretable summary of the subgroup-level drifts that can be used to guide model improvements and interventions. Considering the example in Figure 1, our method detects a significant drop in performance for women under the age of 36 employed in the private sector. Experts could then decide the most appropriate action (e.g., retrain the model). With a global drift detection method, this drop would be hidden by the stable performance at the global level, thus delaying any action to address the performance issue. Our experiments demonstrate the effectiveness of our approach in identifying performance drifts that occur in specific subgroups.

Our approach is performance-based [4] and detects drift by considering changes in performance over time, as new labels become available. This scenario of progressive label availability is common in multiple applications such as spam detection, loan approval, traffic management, online advertising, and social media content moderation. For example, in online advertising, near-real-time feedback on user engagement (e.g., clicks on an ad or purchases following ad views) is typically available.

Our main contributions are as follows.

- **Subgroup-level drift definition.** We introduce the notion of *subgroup-level drift*, as distinct from the global drift that affects a model in its entirety.
- **Fine-grained drift detection and monitoring.** We present a novel approach to model performance monitoring over subgroups, based on sparse matrix multiplication and efficient tests of statistical significance. Our method is able to identify drifts at the subgroup level that would not have been visible via global-level drift monitoring only.
- **Effective drift exploration.** We characterize subgroups via interpretable representations provided by itemset mining techniques. Interpretable subgroups allow the investigation of the factors contributing to performance drifts.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 introduces the background and the problem definition. Section 4 describes our subgroup-based drift approach. Section 5 presents the experimental results. Finally, in Section 6, we draw conclusions and outline future directions.

2 Related work

We analyze related works from two different perspectives: (i) drift detection, and (ii) subgroup analysis.

Drift detection. The main approaches for detecting drift are data distribution-based and performance-based. The former evaluate the similarity of the data distributions over time (e.g., [12, 17, 22]), while the latter focus on tracing differences in performance to detect changes [4]. Our work aligns with performance-based approaches.

Several performance-based approaches, such as DDM [16] and its modification HDDM [15], are based on the statistical process control criterion and adopt statistical tests to identify significant performance degradations, while others directly perform statistical tests such as the Page-Hinkley [31, 33], Chi-squared test [36] and/or Fisher's Exact Test (FET) [14] comparing to a reference distribution. Other approaches adopt a window technique, monitoring performance on different time windows and comparing them with a reference window [6, 13, 32, 37, 38] (e.g., ADWIN [6], KSWIN [38]). We also employ a sliding window approach by comparing the current window under observation with a reference window and applying a statistical test. However, we focus on performance *at the subgroup level* rather than overall. This approach allows us to identify statistically significant drifts for data subgroups that may be undetected when focusing solely on the overall dataset.

Detecting local drift in regions of the feature space is an underexplored area in drift detection [26]. Current methods exploring local drifts mostly focus on unsupervised methods to detect changes in input data and distribution-based approaches. In contrast, we propose a performance-based approach to identify changes in the relationship between input data and a target variable. Mandoline [9] relies on user-specified data slices identified by practitioners. Instead, we automatically detect relevant subgroups (i.e., data slices) and focus on drifts in subgroup performance rather than using subgroups to estimate model performance. [23] proposes a k-means space partitioning for histogram-based distribution change detection. Clustering is also adopted in [46] to partition the data into clusters, followed by learning a discriminator for each cluster to detect drift. Clusters group the data into non-interpretable and non-overlapping instances. Our data subgroups may overlap and are interpretable (e.g., loan applications of young adults and young female adults), allowing for their inspection and analysis. LDD-DSDA [24] monitors differences in the regional density of the data to detect local drifts. Rather than data density, we focus on changes in performance. [19] learns an encoder-decoder model and monitors its loss using an adaptive window size and statistical testing, identifying subspaces by attribute subsets. Instead, we slice the data via attribute-value pairs and do not require additional training.

Subgroup analysis. Multiple subgroup analysis techniques have been proposed to evaluate subgroup performance. Several approaches rely on the a priori knowledge of the subgroups to inspect [3, 21, 30, 40]. The automatic identification of subgroups has been the focus of recent research [8, 11, 34, 35, 39]. We follow the approach of [11, 35, 39], slicing the dataset and representing subgroups as itemsets, i.e., conjunctions of attribute-value pairs, thus yielding interpretable subgroups. While these works characterize system performance at the subgroup level, we focus on monitoring how the model performance over subgroups changes in time.

3 Problem definition

We consider a model $f : \mathcal{X} \to \mathcal{Y}$ which maps elements $x \in \mathcal{X}$ of the input domain $\mathcal{X}$ to a prediction $\tilde{y} = f(x)$. The problem addressed by f may encompass generic tasks such as classification or regression.

We consider performance metrics that can be measured on collections $X = x_1, \ldots, x_N$ of input instances for f, and that can be expressed as the fraction of positive occurrences of an event of choice. Precisely, we assume that we have two indicator functions $\alpha, \beta : \mathcal{X} \mapsto \{0, 1\}$, such that $\alpha(x) = 1$ if x is a positive instance, and $\beta(x) = 1$ if x is a negative instance. Given a collection $X = x_1, \ldots, x_N$ of inputs, we define a performance metric h via:

$$h(X) = \frac{\sum_{x \in X} \alpha(x)}{\sum_{x \in X} \alpha(x) + \beta(x)} \ . \tag{1}$$

By taking $\alpha(x) = 1$ for correctly classified samples and $\beta(x) = 1$ for incorrectly classified ones in (1), h expresses accuracy. By taking $\alpha(x) = 1$ for false positives and $\beta(x) = 1$ for true negatives, h expresses the false-positive rate. We are interested in detecting whether the value of the performance metric h drifts, in statistically significant ways, when measured over data subgroups. We assume that the ground truth labels become gradually available over time, allowing us to track the performance metric during model lifetime.

To define subgroups, we assume that with each instance x_i is associated an interpretable description, or *metadata*, D_i. For images, the metadata can consist in the image source, creation date, dominant color, camera name, or location. For tabular data, the instance itself can be adopted as its own description. We assume that each piece of metadata in D_i has the form of an *attribute=value* pair. Further, we assume that the metadata can be extracted from the instance via a function e of choice (i.e., $D_i = e(x_i)$), so that while extracting the metadata, we can ensure it is discretized and aggregated in ways that facilitate its interpretation. For example, while x_i can contain a timestamp with 1-second resolution, the metadata $e(x_i)$ may contain the month on which image x_i was taken, or the day of the week. An example of metadata for an image is {*age=50, gender=female, smiling=true, hair=blonde*}.

Subgroups are defined on the basis of the instance metadata. Borrowing terminology from frequent pattern mining [42], each *attribute=value* pair constitutes an *item*; we denote by $\mathcal{I}$ the set of all such items. Subgroups are defined via *itemsets*, or sets of items. For instance, an itemset for the above image dataset can be {*age=30, gender=female*}. An instance $x_i \in \mathcal{X}$ is *covered* by (or satisfies) an itemset S if $S \subseteq e(x_i) = D_i$, which we abbreviate as $S \subseteq x_i$. Given a collection X, we denote with $X(S)$ the subcollection of X that satisfies the itemset S, that is, $X(S) = (x \mid x \in X, S \subseteq x)$. Further, we define the support of S in X, denoted $\text{sup}_X(S)$, as the fraction of instances satisfying S, i.e. $\text{sup}_X(S) = |X(S)|/|X|$.

4 Method

Our goal is to monitor the subgroup evolution and detect any drifts in the chosen performance metric over time. We do so by identifying a reference window of data with the expected behavior. We then consider subsequent windows of data (e.g., batches of data collected after deployment), and we detect statistically significant drifts in performance separately for each subgroup.

Figure 2 summarizes the proposed approach and contextualizes its adoption in a classic training/deployment scenario. The pipeline features the following sub-processes of interest.

- **Training.** As part of the model training phase, we extract the subgroups of interest using itemset mining approaches [35, 39]. We note that the rest of the pipeline is agnostic with respect to the choice of monitored subgroups.
- **Deployment.** The previously built model is applied to new, unseen data. In this phase, we identify a *reference window*, which represents a reference situation with data being sampled from the nominal data distribution (often, the distribution used during training). The *current window* instead represents the latest observed data points. We study the difference in performance, separately for each subgroup, between the current and the reference windows.
- **Monitoring.** The divergences of the subgroups through time, i.e., the changes in the current performance w.r.t. the reference situation, are computed. The final outcome of the monitoring phase is a summary defining the extent to which drift is occurring and which subgroups it is affecting.

In the following, we first discuss the process of identifying the subgroups to monitor (§4.1). Then, we present an efficient representation (§4.2) for the matrix-based computation of subgroup-wise metrics (§4.3). Finally, we define how we compute the drift in performance and its statistical significance at the subgroup level (§4.4).

4.1 Identification of Monitored Subgroups

We focus on monitoring subgroups that are well-represented in the training set and interpretable. Specifically, given a dataset X, consisting for instance of the training data, we monitor the subgroups corresponding to the itemsets S such that $\text{sup}_X(S) \geq s$ for a user-specified minimum support s. Defining subgroups in terms of itemsets ensures their interpretability. Focusing on subgroups that are well-represented in the training set ensures that any performance variation over the subgroups has practical relevance. For a counter-example, the subgroup {*gender=male, height=100-120in*} may not contain any instance during training and evaluation, so monitoring its performance is pointless. We denote by $\mathcal{G} = \{S : sup_X(S) \geq s\}$ the set of *frequent* subgroups in the training set X. Such frequent subgroups can be extracted via frequent pattern mining techniques such as the Apriori [2] or FP-Growth [18] algorithms.

4.2 Subgroup membership extraction

To monitor model performance across subgroups, the naive approach would be to compute the performance for each subgroup S separately by considering the instances $X(S)$ that satisfy S and computing the performance metric h over $X(S)$. This can be very inefficient: the number of monitored subgroups can be very large due to the combinatorial way in which itemsets (and thus subgroups) are defined. To make subgroup performance monitoring practical, we present an approach that leverages efficient sparse matrix multiplications to compute the performance across all subgroups simultaneously.

Since the monitored subgroups $\mathcal{G}$ are chosen at training time (i.e., on the training set), they can be computed once and stored for later use. Specifically, we store the subgroups as an itemset membership

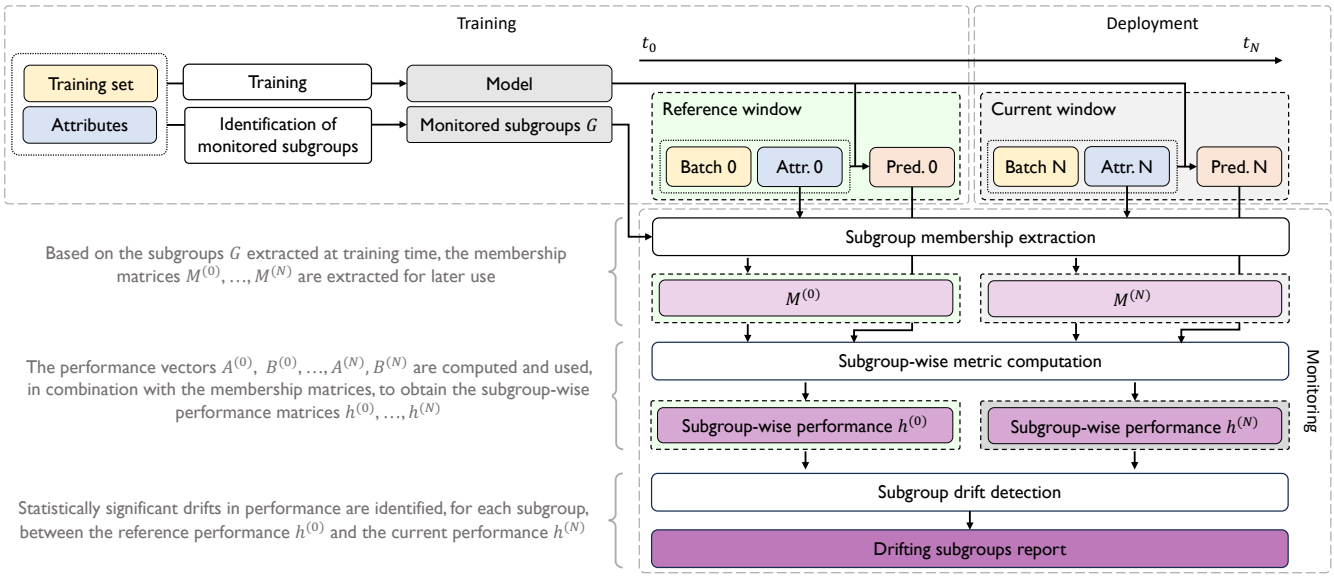

Figure 2: Main steps of the DriftInspector process.

matrix. We define G' as a $|\mathcal{G}| \times |\mathcal{I}|$ sparse matrix representation of $\mathcal{G}$. Each element G'_{ij} is 1 if the item $\alpha_j \in \mathcal{I}$ is contained in subgroup S_i, and 0 otherwise:

$$G'_{ij} = \mathbb{1}(\alpha_j \in S_i) \ . \tag{2}$$

Further, we let G be a normalized version of G', such that each column sums to 1; each element G_{ij} is computed via:

$$G_{ij} = \frac{G'_{ij}}{\sum_k G'_{ik}} \ . \tag{3}$$

We will leverage this normalization in the efficient computation of the membership matrix for data instances, as described below.

Next, we keep track of the items present in each instance x of a collection $X = x_1, \ldots, x_N$ via a *point matrix* $P^{(X)} \in \{0,1\}^{N \times |\mathcal{I}|}$, defined via:

$$P^{(X)}_{ij} = \mathbb{1}(\alpha_j \in x_i) \ . \tag{4}$$

The point matrix $P^{(X)}$ is sparse, allowing for an efficient representation. For simplicity, we omit the (X) superscript when clear from context. Based on the groups matrix G and the points matrix $P^{(X)}$, we define a membership matrix $M^{(X)} \in \{0,1\}^{N \times |\mathcal{G}|}$, which tracks the membership of instances into subgroups. The entry $M^{(X)}_{ij}$ indicates whether x_i satisfies subgroup S_j:

$$M^{(X)}_{ij} = \mathbb{1}(S_j \subseteq x_i) \ . \tag{5}$$

The matrix $M^{(X)}$ can be efficiently computed from $P^{(X)}$ and $G^\intercal$ via:

$$M = \left\lfloor P^{(X)} G^\intercal \right\rfloor \ , \tag{6}$$

where the floor function $\lfloor \cdot \rfloor$ is applied element-wise. In (6), the normalization of G is used along with the floor function to implement the membership check of input instances into subgroups. Similarly to G and $P^{(X)}$, the matrix $M^{(X)}$ can be efficiently computed and stored as a sparse matrix. We will use the membership matrix $M^{(X)}$ to efficiently compute performance across subgroups.

4.3 Subgroup-wise metric computation

Recall that the performance function h we monitor is expressed via the ratio (1) between positive outcomes and positive-plus-negative outcomes. Letting $\alpha(X) = \sum_{x \in X} \alpha(x)$ and $\beta(X) = \sum_{x \in X} \beta(x)$, we can rewrite (1) as

$$h(X) = \frac{\alpha(X)}{\alpha(X) + \beta(X)} \ . \tag{7}$$

Similarly, we express the performance over a subgroup S in X as:

$$h(X(S)) = \frac{\alpha(X(S))}{\alpha(X(S)) + \beta(X(S))} \ , \tag{8}$$

where $\alpha(X(S))$ and $\beta(X(S))$ are the sums, limited to the instances that belong to the subgroup, or $\alpha(X) = \sum_{x \in X(S)} \alpha(x)$ and $\beta(X) = \sum_{x \in X(S)} \beta(x)$. We can compute these quantities efficiently via the membership matrix $M^{(X)}$. To this end, let

$$A^{(X)} = (\alpha(x_1), \alpha(x_2), \ldots, \alpha(x_N)) \ ,$$
$$B^{(X)} = (\beta(x_1), \beta(x_2), \ldots, \beta(x_N) \ .$$

Further, define $A^{(X(S_j))}$, $B^{(X(S_j))}$ as the length-N vectors with components:

$$A^{(X(S_j))}_i = A^{(X)}_i M^{(X)}_{ij} \ ,$$
$$B^{(X(S_j))}_i = B^{(X)}_i M^{(X)}_{ij} \ .$$

Then,

$$\alpha(X(S_j)) = \sum_{i=1}^N A^{(X(S_j))}_i = \sum_{i=1}^N A^{(X)}_i M^{(X)}_{ij} \ ,$$

$$\beta(X(S_j)) = \sum_{i=1}^N B^{(X(S_j))}_i = \sum_{i=1}^N B^{(X)}_i M^{(X)}_{ij} \ ,$$

allowing the computation of $h(X(S_j))$ via (8). All these computations can be efficiently performed via sparse-matrix computation, yielding an efficient technique for monitoring performance over large numbers of subgroups defined over itemsets.

4.4 Drift detection

To identify drift, we consider a reference window of data X_R, and a current window of data X_C. For each subgroup S, we can compute the drift

$$\Delta h_S(R, C) = h(X_R(S)) - h(X_C(S)) \,, \tag{9}$$

via the methods presented above. The remaining question is: are such drifts statistically significant? If a subgroup S has small representation in R, or especially in C, where its support may not be above the chosen support threshold, $\Delta h_S(R, C)$ may be affected by statistical fluctuations, and the measured drift not indicative of a change in model performance.

To estimate the statistical significance of drift, following [35], we note that we can consider the outcomes of α and β as the outcomes of a Bernoulli process, where $\alpha = 1$ indicates a positive outcome (or "head" in a coin toss), and $\beta = 1$ a negative outcome (or "tail"). If we take $\alpha = 1$ and $\beta = 1$ to be the observations, the distribution of the true performance $h(X(S))$, in a Bayesian sense, is distributed according to the Beta distribution with parameters $\alpha(X(S)) + 1$ and $\beta(X(S)) + 1$, corresponding to $\alpha(X(S))$ observed positive outcomes and $\beta(X(S))$ negative ones. The mean of such a Beta distribution is

$$\mu(h, S \mid X) = \frac{\alpha(X(S)) + 1}{\alpha(X(S)) + \beta(X(S)) + 2} \,,$$

and the variance is

$$v(h, S \mid X) = \frac{(\alpha(X(S)) + 1)(\beta(X(S)) + 1)}{(\alpha(X(S)) + \beta(X(S)) + 2)^2(\alpha(X(S)) + \beta(X(S)) + 3)} \,.$$

Thus, we can use Welch's t-test to compute the statistical significance $t_{h,S}(R, C)$ of the drift $\Delta h_S(R, C)$ given by (9), as:

$$t_{h,S}(R, C) = \frac{|\mu(h, S \mid X_R) - \mu(h, S \mid X_C)|}{\sqrt{v(h, S \mid X_R) + v(h, S \mid X_C)}} \,. \tag{10}$$

This result allows us to define a ranking among all subgroups based on the statistical significance of their divergence. Additionally, we can introduce a significance lower bound τ_t, detecting drift on a subgroup S only if $t_{h,S} > \tau_t$.

Finally, we propose extracting a global detection outcome (i.e., whether, overall, the data is drifting). This binary outcome is helpful in defining whether any action should be taken. We adopt a strict detection policy whereby a dataset is said to be drifting if at least one of the subgroups contained within is drifting. The global detection function $G_t(R, C)$, producing a binary decision regarding whether dataset C has drifted w.r.t. R can be defined as:

$$G_t(R, C) = \begin{cases} \text{T} & \text{if } \exists\ S \ s.t.\ t_{h,S} > \tau_t \\ \text{F} & \text{otherwise} \end{cases} \,. \tag{11}$$

We note that the *global* subgroup (i.e., the subgroup containing the entire population) is one of the monitored subgrups in $\mathcal{G}$, so that the test (11) is also performed at the global level. Thus, DriftInspector generalizes the behavior of global-only drift detection techniques such as DDM [16] and HDDM [15].

5 Experimental evaluation

We evaluate DriftInspector from both the qualitative and quantitative perspectives. For the qualitative evaluation, we discuss the insights provided by our method on performance drift. The quantitative evaluation assesses DriftInspector capability of (1) detecting the occurrence of a drift event, and (2) correctly identifying the subgroups for which a drift has actually occurred. We investigate the ability to detect both *local* (i.e., subgroup-based) and *global* drifts (i.e., uniformly affecting the data) of our approach. The source code of DriftInspector, the code necessary to reproduce our experiments, and further details on the experimental setting are available at https://github.com/fgiobergia/DriftInspector.

5.1 Experimental setting

Table 1: Main features of Adult and CelebA (s = 0.01).

	Adult	CelebA
# Samples	48,842	205,599
# Metadata	14	39
# Subgroups	192,099	26,597
Dataset type	Tabular	Images & annotations
Features type	Categorical & continuous	Binary

Local drift datasets. We tested DriftInspector on two commonly adopted datasets, Adult [5] and CelebA [25], which we adapted by injecting drift (as explained below). Table 1 summarizes the main features of the two datasets. For Adult, we used the income as target. For CelebA, we identified one of the 40 binary variables as the target, and, specifically, the attribute "Attractive". This attribute has been chosen as it represents a non-trivial and subjective task, that may possibly be subject to drifts over time. We note that we do not endorse training models for this target as something useful or desirable beyond our experiments here.

Both datasets have been split into a 50/50 train/test split. The test split has been further split into 30 equally-sized batches. In this way, we can simulate the application of the final model to a sequence of batches through time (as is generally the case for deployed models).

To obtain statistically representative results, we conducted multiple experiments. For each experiment, the data is shuffled, and the occurring drift varies (as detailed below). Each of these executions is considered as a self-contained experiment.

Local drift injection. We introduced a controlled drift that can then be detected and quantified. To this end, we randomly selected a subgroup for each experiment. We refer to this subgroup as the *target subgroup*. Noise that affects a subset of the target subgroup is then introduced. In this way, a ground truth regarding the points affected by drift can be established.

Noise is introduced so as to represent two main kinds of drifts.

- *Concept drift.* The label for the injected points is changed (i.e., flipped in a binary classification problem). In this way, we simulate a change in behavior in the outcome of specific population subgroups, as is the case when concept drift occurs. We inject this drift in the Adult dataset. The degree of the noise can be modulated by varying the fraction of points that are affected by the label flip.

- *Domain drift.* Noise is applied directly to the input (e.g., Gaussian blur for images). In this way, the distribution of the input space changes, as is generally the case with domain shift. We adopt this approach for the CelebA dataset. The entity of the noise can be modulated by varying the intensity of the introduced blur.

We distribute the injection of noise in the 30 batches as follows.

- *Normal batches (batches 1-10).* No noise is injected, thus representing the original, expected behavior.
- *Transition batches (batches 11-20).* A transitory phase where noise is increasingly injected in the batches. These batches represent the transitory where the drift starts occurring.
- *Drift batches (batches 21-30).* The noise saturates to a fixed value. It is the situation in which the drift already occurred.

For the presented results, we consider a reference window and a current window, both containing 5 batches, taken from the normal and drift batch groups, respectively. We empirically verified that the reported results are stable for larger and smaller batch sizes. Finally, we note that, although injected noise alters the points of a single subgroup, this effect may also propagate to other subgroups because each point typically belongs to multiple (overlapping) subgroups.

Global drift datasets. We additionally studied the behavior of DriftInspector when the drift occurs at a global level, i.e., no specific subgroup is drifting. This has been the focus of existing drift detection algorithms. As such, various commonly used datasets exist in the literature to study global drifts in streaming data. We included the following synthetic datasets: *Agrawal* [1], with demographic data and a binary outcome on whether or not a loan is approved, *SEA* [41], with a classification target that only depends on a subset of the input features, *LED* [7], with noisy data from a 7-segments display, and *HP* (Hyperplane) [20], where the problem is to predict the class of a rotation hyperplane. For these datasets, several concepts already exist between inputs and outputs. We adopt a concept drift that gradually shifts from one concept to the other by means of a sigmoid function, a commonly adopted choice [28]. We use 5,000 training points from the nominal concept and then produce 50 batches of 200 points each. We add noise to the labels of 10% of the points to prevent the problems from being trivial.

Experiment structure. We identified three main experiments of interest: (1) the detection of a local (subgroup) drift event, (2) the identification of the most drifting subgroups, i.e. the subgroups for which the drifting event is occurring more extensively, and (3) the detection of a global drift event. For all experiments, we monitor the accuracy of the model and detect a drift based on this metric (i.e., $h(\cdot)$). The drift in performance, as defined in (9), is referred to as Δ_{acc}, with its respective t-statistic. We defined two types of drift detection experiments:

- *Positive experiments*: each experiment represents a sequence of batches for which drift has been injected, as detailed above. For local drifts, the target subgroup injected with noise varies in each experiment.
- *Negative experiments*: each experiment represents a sequence of batches for which no drift has been injected.

For a given experiment, we considered the outcome to be positive (i.e., a drift was detected) if the algorithm detected a drift in at least one of the batches in the sequence, and negative otherwise. We evaluated the algorithms in terms of their accuracy and F_1 scores; additionally, we report their false positive rate (FPR) and false negative rate (FNR).[1]

We framed the identification of the subgroups that drift most as a ranking problem. DriftInspector lets us rank the subgroups according to the detected drift. We take the true fraction of points that we modified via noise as the *relevance* rel_k of the k-th ranked subgroup. We then measure the correlation between rank position and relevance, via Discounted Cumulative Gain (nDCG) [44], a metric that is commonly adopted to evaluate rankings, as well as Pearson and Spearman correlations. We also compute the *nDCG@K* to evaluate the quality of top-K results. Specifically, we evaluated nDCG@10, nDCG@100 (i.e., for the top 10 and 100 results), and nDCG for the full ranking. Additionally, we measured the Pearson correlation between the relevance of the subgroups and the respective t value, as well as the Spearman's correlation on rankings.

We studied the behavior of DriftInspector for affected subpopulations of varying size. The experimental results are presented as a function of the support of the target subgroup. We produced up to 50 positive experiments for each considered support. [2] The average across all experiments is provided when reporting aggregated results over all supports (e.g., in Tables 3 and 4).

Classification models. For the Adult dataset, we adopted XGBoost [10] with 100 estimators as it has been shown to perform well in terms of accuracy [5]. For CelebA, we used a pre-trained ResNet50 model with a binary classification head added to address the introduced binary problem. For the synthetic datasets, we used decision trees, given the simple nature of these benchmarks.

Baselines. We compared DriftInspector with seven global-level drift detection techniques. We considered techniques that monitor model performance over time, specifically DDM [16], HDDM [15], and Page-Hinkley [31, 33], statistical tests, specifically the Chi-squared test χ^2 [36] and the Fisher's Exact Test (FET) [14] and windowing-base techniques, specifically ADWIN [6] and Kolmogorov-Smirnov Windowing (KSWIN) [38], as benchmarks, as these are among the best known and most widely used techniques. For DDM, we fine-tuned the minimum number of samples required before detecting a change W. For HDDM, we varied the confidence level for the drift ϵ, as defined in [15]. For Page-Hinkley [31, 33], we varied the minimum number of samples λ required before detecting a change. For χ^2 and FET, we varied the p-value for the significance test. For ADWIN, we varied the significance threshold δ, which determines the difference between the average performance statistics of the two sub-windows before a drift is identified as such. For KSWIN, we varied the size of the sliding window W.

5.2 Experimental results

Qualitative analysis. To explore the insights provided by DriftInspector, we performed an experiment on the CelebA

[1] We introduced the same number of positive and negative experiments, thus producing a balanced problem.

[2] If less than 50 subgroups exist for a given support, all subgroups are considered, producing less than 50 experiments.

Table 2: Example of subgroup drift detection. Top-4 subgroups by t statistic and 17th subgroup (target subgroup of the experiment). CelebA dataset.

Subgroup	Δ_{acc}	t
Big_Lips , Wearing_Lipstick , Wavy_Hair , Young , Heavy_Makeup, Oval_Face, Arched_Eyebrows	-0.91	43.8
Big_Lips , Wearing_Lipstick , Wavy_Hair , Young , Heavy_Makeup, Oval_Face	-0.88	43.2
Big_Lips , Wearing_Lipstick , Wavy_Hair , Young Heavy_Makeup, Oval_Face, No_Beard	-0.88	43.1
Big_Lips , Wearing_Lipstick , Wavy_Hair , Young , Heavy_Makeup	-0.69	42.9
Big_Lips , Wearing_Lipstick , Wavy_Hair , Young	-0.62	39.8

dataset. We injected noise in the subgroup {Big_Lips, Wearing_Lipstick, Wavy_Hair, Young}. This subgroup has a support of 0.075 in the test set. DriftInspector detects this subgroup as experiencing drift, with a divergence of -0.62% and a t-value of 39.8. This subgroup is among the top 20 with the highest t-statistic (specifically, the 17^{th}). Many of the subgroups with higher t-value are indeed subsets of the considered subgroup. This effect can be observed in Table 2 that lists the four subgroups with the highest t-value, while the bottom line reports the target subgroup. Hence, DriftInspector is able to identify the data subgroups affected by drift. We include further qualitative analyses in the appendix.

Subgroup drift detection. We compare the performance of DriftInspector against DDM, HDDM, and ADWIN in terms of F_1 score when varying the support of the target subgroup. We adopt DriftInspector using $\tau_t = 5$. We investigate the impact of the τ_t threshold in the last part of the section. In terms of execution time, we compare DriftInspector against other drift detection techniques. Under the same conditions (i.e., applying the other detection techniques on each subgroup separately), DriftInspector is at least 100 times faster than then next fastest technique. More details on execution time are reported in Appendix E.

Figure 3 reports the experimental comparison of the above methods. The methods achieve near-perfect performance in identifying drifts when the target subgroup is sufficiently large (e.g., 10% or more), because a large fraction of the entire dataset is being affected by the drift. However, DriftInspector outperforms the other techniques for drift affecting smaller subgroups. This improvement in drift detection is the result of the finer granularity of the subgroup-level analysis. Figure 1 illustrates this: when observing global level performance, local drifting effects may be lost, as in the case with the global competitors.

A summary of the performance in terms of accuracy, F_1 score, FPR and FNR is reported in Table 3, across injected subgroups of different size (i.e., with different support). We see that most algorithms (including DriftInspector) have low FPR: in other words, the detectors rarely report a drift when none occurred. The situation is different for FNR. Global algorithms generally have large FNR, as they fail to detect subgroup-level drifts; in contrast, DriftInspector achieves distinctly lower FNR thanks to its finer-grained analysis (we note that, although KSWIN and Page-Hinkley

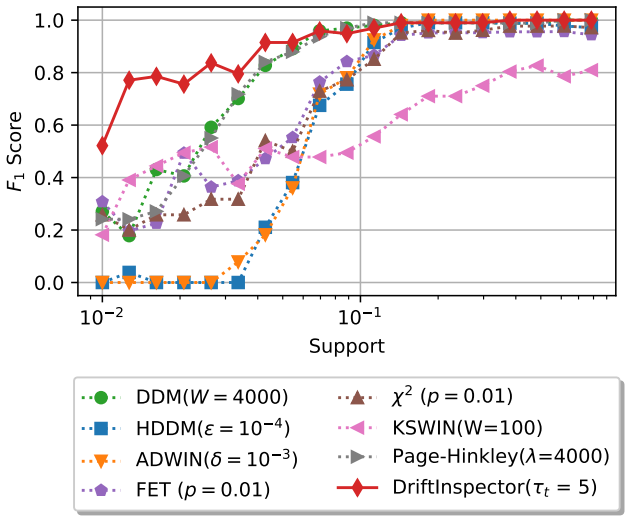

Figure 3: Drift detection performance as the target subgroup support varies; Adult. DriftInspector achieves satisfactory results even when small portions of the population drift. Each point obtained from up to 50 positive experiments (i.e., with injected drift) and the same number of negative ones.

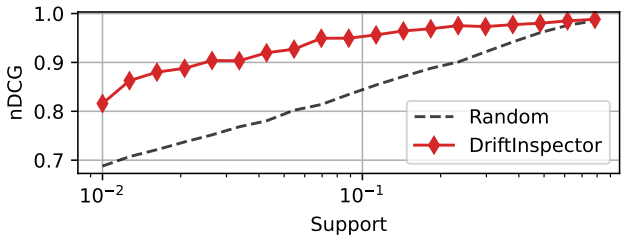

Figure 4: nDCG varying the target subgroup support; Adult.

have lower FNR on CelebA, both are affected by a much larger FPR). This indicates, as expected, that the proposed approach can detect a wider variety of drifts w.r.t. established techniques.

Drifting subgroups identification. Based on the fraction of each subgroup that has been altered, Table 4 quantifies the quality of the ranking produced by ordering the subgroups according to the t-statistic computed. We compare this performance with the one achieved with a random guess, since the previously considered competitors do not produce subgroup-level detections.

The nDCG@10 and nDCG@100 metrics show that, even for small values of K, DriftInspector already highlights the most relevant results. Since manual inspection of the most drifting subgroups is generally expected, this is a result of particular interest.

In Figure 4, we explore the quality of the ranking as a function of the size of the target subgroup. For drifts with support below 10%, the proposed approach already produces meaningful rankings, as highlighted by the difference with respect to the random baseline. When the support of the affected subgroup increases, more

Table 3: Accuracy (Acc.), F_1 score, FPR, and FNR results on Adult and CelebA for the binary drift detection task. Best results for each metric. Results reported as mean ± standard deviation.

		χ^2 ($p = 0.01$)	ADWIN ($\delta = 10^{-3}$)	DDM ($W = 4000$)	FET ($p = 0.01$)	HDDM ($\epsilon = 10^{-4}$)	KSWIN ($W = 100$)	Page-Hinkley ($\lambda = 4000$)	DriftInspector ($\tau_t = 5$)
Acc.	**Adult**	0.774 ± 0.185	0.773 ± 0.223	0.865 ± 0.163	0.776 ± 0.175	0.760 ± 0.221	0.596 ± 0.119	0.861 ± 0.168	**0.919 ± 0.092**
	CelebA	0.812 ± 0.144	0.879 ± 0.170	0.906 ± 0.090	0.828 ± 0.119	0.854 ± 0.152	0.561 ± 0.022	0.709 ± 0.074	**0.991 ± 0.014**
F_1	**Adult**	0.670 ± 0.305	0.580 ± 0.442	0.797 ± 0.272	0.689 ± 0.282	0.571 ± 0.435	0.577 ± 0.173	0.789 ± 0.282	**0.902 ± 0.124**
	CelebA	0.786 ± 0.201	0.801 ± 0.316	0.892 ± 0.123	0.811 ± 0.161	0.804 ± 0.257	0.691 ± 0.013	0.775 ± 0.058	**0.991 ± 0.015**
FPR	**Adult**	0.074 ± 0.013	**0.000 ± 0.000**	0.018 ± 0.006	0.100 ± 0.005	0.024 ± 0.010	0.415 ± 0.035	0.018 ± 0.006	0.016 ± 0.008
	CelebA	0.178 ± 0.058	**0.000 ± 0.000**	0.067 ± 0.029	0.176 ± 0.046	0.083 ± 0.028	0.861 ± 0.041	0.566 ± 0.142	**0.000 ± 0.000**
FNR	**Adult**	0.377 ± 0.363	0.454 ± 0.446	0.253 ± 0.326	0.348 ± 0.349	0.456 ± 0.445	0.394 ± 0.256	0.260 ± 0.334	**0.146 ± 0.181**
	CelebA	0.198 ± 0.266	0.243 ± 0.340	0.121 ± 0.182	0.168 ± 0.231	0.208 ± 0.304	0.017 ± 0.019	**0.015 ± 0.025**	0.018 ± 0.028

Table 4: Evaluation of the rankings, based on t, against a random baseline. Note that the theoretical random guess with Pearson and Spearman correlations is 0 – the reported results are obtained from empirical evidence and are consistent with the expected value. Results reported as mean ± standard deviation.

		DriftInspector	Random
nDCG@10	**Adult**	**0.707 ± 0.235**	0.151 ± 0.167
	CelebA	**0.975 ± 0.112**	0.305 ± 0.235
nDCG@100	**Adult**	**0.697 ± 0.219**	0.156 ± 0.161
	CelebA	**0.952 ± 0.124**	0.308 ± 0.225
nDCG	**Adult**	**0.931 ± 0.068**	0.825 ± 0.088
	CelebA	**0.973 ± 0.050**	0.884 ± 0.073
Pearson	**Adult**	**0.555 ± 0.236**	0.000 ± 0.002
	CelebA	**0.736 ± 0.174**	0.000 ± 0.006
Spearman	**Adult**	**0.457 ± 0.225**	0.000 ± 0.002
	CelebA	**0.676 ± 0.217**	0.000 ± 0.006

instances are perturbed and more subgroups are affected by noise on average. Hence, the fraction of affected points in most subgroups will become larger. In this case, as can be expected, DriftInspector produces near-perfect rankings. Note that, for support values close to 100%, almost all instances are affected by noise and random selection also reaches an "almost perfect" result.

Global drift detection. We additionally studied the behavior of DriftInspector when the drift occurs at a global level, i.e., there is no one specific subgroup drifting. This has been the focus of existing drift detection algorithms. As such, we compared DriftInspector with existing techniques on four benchmark datasets. Table 5 presents the main results obtained in terms of accuracy, F_1 score, FPR and FNR. DriftInspector generally obtains comparable, or better performance w.r.t. other techniques when global drift occurs. This same behavior can be observed on the right-hand side of Figure 3, when the injected subgroup approximately encompasses the entire dataset (i.e., a global drift occurs).

Table 5: Performance of various drift detection techniques on 4 synthetic datasets, with global drift applied.

		χ^2	ADWIN	DDM	FET	HDDM	KSWIN	Page-Hinkley	DriftInspector
Accuracy	**Agrawal**	0.986	0.986	0.943	0.986	0.986	0.886	**1.000**	**1.000**
	LED	0.486	0.500	0.471	0.457	0.486	**0.514**	0.471	0.500
	SEA	0.700	0.843	0.529	0.543	0.543	0.443	0.571	**1.000**
	HP	0.757	0.914	0.871	0.771	0.914	0.614	**0.929**	0.886
F_1 **score**	**Agrawal**	0.986	0.986	0.943	0.986	0.986	0.885	**1.000**	**1.000**
	LED	0.327	0.357	0.320	0.385	0.470	0.508	0.394	**0.545**
	SEA	0.689	0.840	0.431	0.440	0.510	0.327	0.490	**1.000**
	HP	0.755	0.914	0.871	0.771	0.914	0.590	**0.928**	0.897
FPR	**Agrawal**	0.029	0.029	0.114	0.029	0.029	0.029	**0.000**	**0.000**
	LED	**0.029**	**0.029**	0.057	0.886	0.686	0.371	0.171	0.447
	SEA	0.114	0.029	0.057	0.029	0.714	0.143	0.029	**0.000**
	HP	0.143	**0.000**	0.057	0.171	0.114	0.629	0.143	0.103
FNR	**Agrawal**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.200	**0.000**	**0.000**
	LED	1.000	0.971	1.000	**0.200**	0.343	0.600	0.886	0.562
	SEA	0.486	0.286	0.886	0.886	0.200	0.971	0.829	**0.000**
	HP	0.343	0.171	0.200	0.286	0.057	0.143	**0.000**	0.129

Support sensitivity analysis. We analyze the impact of the τ_t threshold on the performance of the drift detection task. More specifically, we study how the optimal τ_t changes with the support of the injected subgroup. We define the optimal threshold as the one maximizing Youden's J statistic [45] ($J = TPR + TNR - 1$, where TPR and TNR are the true positive and negative rates). Figure 5 shows the value of the optimal threshold when varying subgroup support. The optimal τ_t value, corresponding to $\tau_t \approx 5$, is stable for supports < 10%. This range, corresponding to smaller subgroups, is the focus of this work. Still, Figure 3 shows that $\tau_t = 5$ allows achieving good results also for larger supports, even though it is not the optimal threshold. Larger supports correspond to prominent drifts, for which there is tolerance in the setting of the threshold.

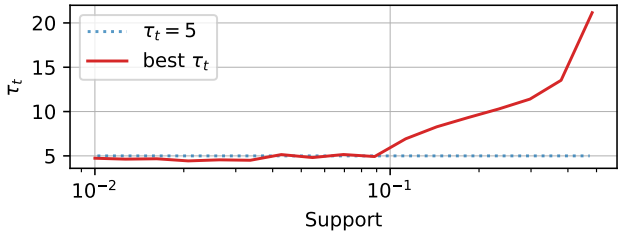

Figure 5: Evolution of the best threshold value for τ_t on Adult, when varying the target subgroup support. The threshold value $\tau_t = 5$ used in the experiments is shown in blue.

6 Conclusions

We propose a novel approach to explore drift phenomena at the finer granularity of subgroups. We automatically identify relevant subgroups during model training and efficiently monitor their drift during model deployment. Our approach allows detecting finer-grained, but highly drifted, data subsets that are not detected by methods that observe drift as a global phenomenon. DriftInspector also allows us to explore the collection of (interpretable) drifting subgroups, thereby serving as a tool for drift understanding and enabling actively working on model improvement. In future work, we plan to explore subgroup-level detection of data distribution shifts when labeled data is unavailable.

Acknowledgments

This work is partially supported by FAIR - Future Artificial Intelligence Research (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013) and the spoke "FutureHPC & BigData" of the ICSC - Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing, both funded by the European Union - NextGenerationEU. This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. 1993. Database mining: A performance perspective. *IEEE transactions on knowledge and data engineering* 5, 6 (1993), 914–925.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215. Santiago, Chile, 487–499.

[3] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. Tfx: A tensorflow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1387–1395.

[4] Firas Bayram, Bestoun S Ahmed, and Andreas Kassler. 2022. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems* 245 (2022), 108632.

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

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

[7] Leo Breiman. 1984. *Classification and regression trees*. Routledge.

[8] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. FairVis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 46–56.

[9] Mayee Chen, Karan Goel, Nimit S Sohoni, Fait Poms, Kayvon Fatahalian, and Christopher Ré. 2021. Mandoline: Model evaluation under distribution shift. In *International conference on machine learning*. PMLR, 1617–1629.

[10] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[11] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2019. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1550–1553.

[12] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. 2006. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *Proc. Symposium on the Interface of Statistics, Computing Science, and Applications (Interface)*.

[13] Roberto Souto Maior de Barros, Juan Isidro González Hidalgo, and Danilo Rafael de Lima Cabral. 2018. Wilcoxon rank sum test drift detector. *Neurocomputing* 275 (2018), 1954–1963.

[14] R. A. Fisher. 1922. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94. http://www.jstor.org/stable/2340521

[15] Isvani Frias-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. 2014. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering* 27, 3 (2014), 810–823.

[16] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. 2004. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*. Springer, 286–295.

[17] Ömer Gözüaçık and Fazli Can. 2021. Concept learning using one-class classifiers for implicit drift detection in evolving data streams. *Artificial Intelligence Review* 54, 5 (2021), 3725–3747.

[18] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM sigmod record* 29, 2 (2000), 1–12.

[19] Marco Heyden, Edouard Fouché, Vadim Arzamasov, Tanja Fenn, Florian Kalinke, and Klemens Böhm. 2024. Adaptive Bernstein change detector for high-dimensional data streams. *Data Mining and Knowledge Discovery* 38, 3 (2024), 1334–1363.

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

[21] Minsuk Kahng, Dezhi Fang, and Duen Horng Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. 1–6.

[22] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *VLDB*, Vol. 4. Toronto, Canada, 180–191.

[23] Anjin Liu, Jie Lu, and Guangquan Zhang. 2020. Concept drift detection via equal intensity k-means space partitioning. *IEEE transactions on cybernetics* 51, 6 (2020), 3198–3211.

[24] Anjin Liu, Yiliao Song, Guangquan Zhang, and Jie Lu. 2017. Regional concept drift detection and density synchronized drift adaptation. In *IJCAI International Joint Conference on Artificial Intelligence*.

[25] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[26] 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.

[27] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes-which naive bayes?. In *CEAS*, Vol. 17. Mountain View, CA, 28–69.

[28] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. 2021. River: machine learning for streaming data in Python. (2021).

[29] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. 2018. Scikit-Multiflow: A Multi-output Streaming Framework. *Journal of Machine Learning Research* 19, 72 (2018), 1–5. http://jmlr.org/papers/v19/18-251.html

[30] Giulio Morina, Viktoriia Oliinyk, Julian Waton, Ines Marusic, and Konstantinos Georgatzis. 2019. Auditing and achieving intersectional fairness in classification problems. *arXiv preprint arXiv:1911.01468* (2019).

[31] Hayet Mouss, D Mouss, N Mouss, and L Sefouhi. 2004. Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In *2004 5th Asian control conference (IEEE Cat. No. 04EX904)*, Vol. 2. IEEE, 815–818.

[32] Kyosuke Nishida and Koichiro Yamauchi. 2007. Detecting Concept Drift Using Statistical Testing. In *Discovery Science*, Vincent Corruble, Masayuki Takeda, and Einoshin Suzuki (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 264–269.

[33] E. S. Page. 1954. Continuous Inspection Schemes. *Biometrika* 41, 1/2 (1954), 100–115. http://www.jstor.org/stable/2333009

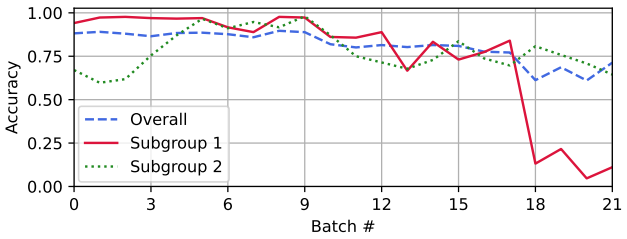

Figure 6: Accuracy, as measured at the global level (continuous blue line), and at the subgroup level. Subgroup 1, $\{schedule\}$, exhibit a drift behavior over time. Subgroup 2, $\{enron, com\}$, is not as significantly affected by a drift.

[34] Eliana Pastor, Elena Baralis, and Luca de Alfaro. 2023. A Hierarchical Approach to Anomalous Subgroup Discovery. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 2647–2659. https://doi.org/10.1109/ICDE55515.2023.00203

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

[36] Karl Pearson. 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (July 1900), 157–175. https://doi.org/10.1080/14786440009463897

[37] Ali Pesaranghader and Herna L Viktor. 2016. Fast hoeffding drift detection method for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*. Springer, 96–111.

[38] Christoph Raab, Moritz Heusinger, and Frank-Michael Schleif. 2020. Reactive soft prototype computing for concept drift streams. *Neurocomputing* 416 (2020), 340–351.

[39] 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*. 2290–2299.

[40] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. 2018. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577* (2018).

[41] W Nick Street and YongSeog Kim. 2001. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 377–382.

[42] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. 2018. *Introduction to Data Mining (2nd Edition)* (2nd ed.). Pearson.

[43] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, Ashley Scillitoe, Robert Samoilescu, and Alex Athorne. 2019. *Alibi Detect: Algorithms for outlier, adversarial and drift detection.* https://github.com/SeldonIO/alibi-detect

[44] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG type ranking measures. In *Conference on learning theory*. PMLR, 25–54.

[45] William J Youden. 1950. Index for rating diagnostic tests. *Cancer* 3, 1 (1950), 32–35.

[46] Mingjie Zhao, Yiqun Zhang, Yuzhu Ji, and Yang Lu. 2023. Unsupervised concept drift detection via imbalanced cluster discriminator learning. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 31–43.

A Qualitative results: Enron Spam Dataset

We further tested DriftInspector on a spam detection problem, using the Enron Spam Dataset [27]. This dataset contains information about 33,716 emails, their outcome (50% spam, 50% ham), and a timestamp (from December 1999 to September 2005).

A simple decision tree model is trained on earlier data (1/3 of the dataset) and applied to newer data (2/3 of the dataset). This simulates a situation where a model is trained on the data that is initially available, and then deployed in production for later use.

We use bags of words to represent email contents, both for the classification model and for the definition of the subgroups.

Upon applying various drift detection techniques, we observe two interesting facts: the first one, as can be expected, is that the False Negative Rate of the model increases over time. Spammers change their behavior so as to circumvent spam detectors. This behavior occurs at the global level and can be detected by traditional detectors, as well as DriftInspector.

The second behavior instead occurs only at the subgroup level for the False Positive Rate. At the global level, the model does not produce more False Positives as time progresses. However, the subgroup-level analysis performed by DriftInspector identifies subgroups for which there is a significant increase in FPR. For instance, the emails containing the words "date" or "schedule" go from an FPR ≈ 0 to an FPR > 0.8.

A manual inspection shows that at one point in time, after training, some automated internal emails with those keywords started being sent. These are not spam emails, but are flagged as such by the outdated model. The phenomenon only affects a limited number of emails, so it is small enough to go undetected by traditional global detectors, but is easily identified by DriftInspector.

In Figure 6 we reported an example of the global and local accuracy for two subgroups. Subgroup 1, including emails that contain the word "schedule", has a significant drift over time (due to internal emails containing it). Subgroup 2, including emails that contain the words "enron" and "com", is not as heavily affected (since those keywords are part of the name of the company under analysis).

Since there is no ground truth information on the nature of the occurring drifts, we cannot quantify the effectiveness of DriftInspectorin this scenario.

B Further qualitative analysis

We conduct further qualitative analyses of DriftInspector results to illustrate its effectiveness as a tool for drift inspection and model understanding. We again consider the example discussed in Section 5.2. In this example, we inject noise for the subgroup {Big_Lips, Wearing_Lipstick, Wavy_Hair, Young}. We analyze (i) a summary of the subgroups, (ii) the contribution to the drift of specific subgroups (local contribution), and (iii) the contribution of each metadata to the drift (global contribution).

Subgroup summary. We can generate a summarized view of subgroup divergence to analyze subgroups more easily. We follow an approach similar to redundancy pruning [35]. The rationale is the following: given an itemset I and an itemset $I \cup \alpha$, where α represents an additional item, we keep the more general I if the difference in their t-value falls below a predefined threshold. Essentially, I already represents the divergence of $I \cup \alpha$, with the additional term α having only minimal impact. Different from [35], we prune based on the t-value as a measure of the strength of the difference between groups rather than the divergence. Table 6 shows the subgroups with the highest t-values after applying a redundancy pruning threshold of 5. These pruned subgroups contain fewer terms. The target subgroup is the second by t-value. The other subgroups include metadata from the target subgroup and others associated with them, such as 'Heavy_Makeup' to 'Wearing_Lipstick'.

Global contribution to the drift. To investigate the drift phenomenon, we can explore the contribution of each item (i.e., metadata) to

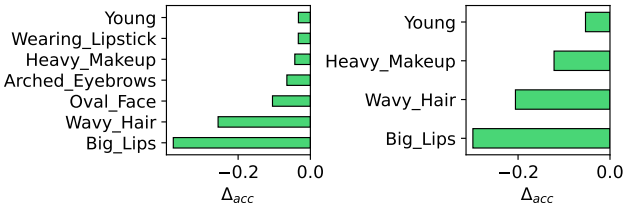

Figure 8: Contribution to the drift in accuracy for the subgroup with the highest t-value and for the target dataset. CelebA dataset.

Table 6: Summary of the top-5 subgroups by t statistic via redundancy pruning of 5 in t-value. CelebA dataset.

Subgroup	Δ_{acc}	t
Big_Lips , Wavy_Hair , Young , Heavy_Makeup	-0.68	42.33
Big_Lips , Wearing_Lipstick , Wavy_Hair , Young	-0.62	39.76
Big_Lips , Wavy_Hair , Oval_Face, Arched_Eyebrows	-0.88	36.93
Big_Lips , Wavy_Hair , Heavy_Makeup	-0.60	36.32
Big_Lips , Wearing_Lipstick , Wavy_Hair	-0.54	34.65

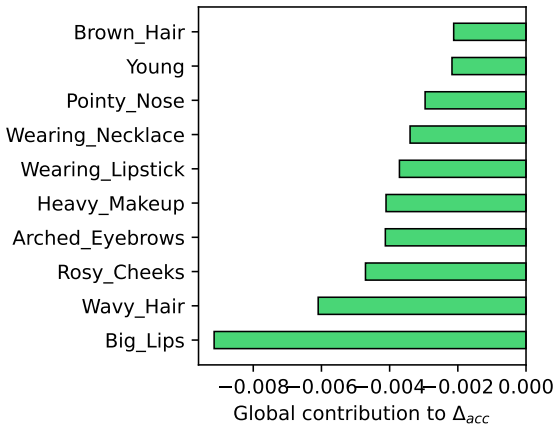

Figure 7: Global contribution to the drift in accuracy. Top 10 terms with the highest contributions to drift. CelebA dataset.

the drift. Using the Global Shapley Value [35], we can estimate the average contribution of each item to the drift, considering its effect on all explored subgroups. Figure 7 shows the top 10 terms with the highest contribution. The higher the magnitude of the value, the more the item impacts the performance reduction. 'Wavy_Hair' and 'Big_Lips' are the most influential metadata contributing to drift. 'Wearing_Lipstick' also stands out as a predominant term. 'Young' instead exhibits a lower contribution, likely due to its high frequency in the dataset (80%). An interesting observation is the prevalence of multiple terms with high negative contribution among images labeled as female in the dataset (e.g., 'Rosy_Cheeks', 'Arched_Eyebrows'). This impact of gender-associated metadata (for this dataset) aligns with observing that almost all images in the target subgroup are labeled as women (99.78%). Practitioners can inspect the subgroups and derive a summary of the drifting behavior. Further, the interpretable definition of subgroups facilitates the inspection of the different factors associated with drift.

Local contribution to the drift. The subgroup with the highest t-value in the change in performance is {Big_Lips, Wearing_Lipstick, Oval_ Face, Wavy_Hair, Arched_Eyebrows, Young, Heavy_Makeup} (see Table 2). The subgroup includes all metadata from the target one plus additional terms. As a result, the instances of this subgroup are all included in the target one and are affected by the noise injection.

We are interested in analyzing how much each term contributes to the drift. For this purpose, we can use the Shapley value, as adopted in [35]. The Shapley value is a notion from game theory that estimates the contribution of each team player to the total score. In our context, we consider the divergence in the performance of a subgroup as the total score, the subgroup as the team, and each item (i.e., each metadata) as the players. Figure 8 (left) shows the Shapley values for the drifting subgroups with the highest t-value. The terms 'Big_Lips' and 'Wavy_Hair' are the terms that mostly contribute to the drift in performance for the subgroup. We then have the terms 'Oval_Face', 'Arched_Eyebrows', 'Heavy_Makeup'. As we analyzed, these terms are prevalent among images labeled as 'woman'. Consequently, their relevance is justified by the fact that almost all images in target subgroups are labeled as 'woman' as well (98.8%).

Figure 8 (right) shows the contributions for the target subgroups. The most important terms are again 'Big_Lips' and 'Wavy_Hair'. The term 'Young' has the lowest contribution in absolute terms. Given that most of the images of the entire dataset (80%) are labeled as 'Young', it is not a distinctive characteristic of the subgroups affected by noise. This justifies the higher contributions of the other metadata.

By injecting noise for images of the target subgroup, we are consequently modifying instances of other overlapping subgroups as well. We study the relationship between the fraction of affected instances in each subgroup and its divergence and t-value. Figure 9 shows the divergence scores (top) and t-values (bottom) of the explored subgroups with respect to the fraction of altered instances in each subgroup. We observe a clear trend. The higher the fraction of altered points, the lower the divergence (and higher in absolute terms), indicating a drop in accuracy for the subgroups. Similarly, higher fraction correspond to higher t-values. In the figures, we highlight the target subgroup. As also discussed in Section 5.2, there are subgroups that experience even higher drops. These subgroups are subsets of the target one (e.g., Big_Lips, Wearing_Lipstick, Wavy_Hair, Young, Heavy_Makeup, Oval_Face, see Table 2).

C Complexity analysis

This section presents a complexity analysis quantifying the number of operations required for each iteration (batch).

The initial cost for the extraction of the subgroups of interest (at training time) depends on the algorithm of choice. We consider this cost as separate from the computational cost required by DriftInspector for the extraction of the subgroup-wise metrics.

We assume that a total of $|\mathcal{G}|$ subgroups are being monitored, and that a batch of size N is being considered. We additionally refer to

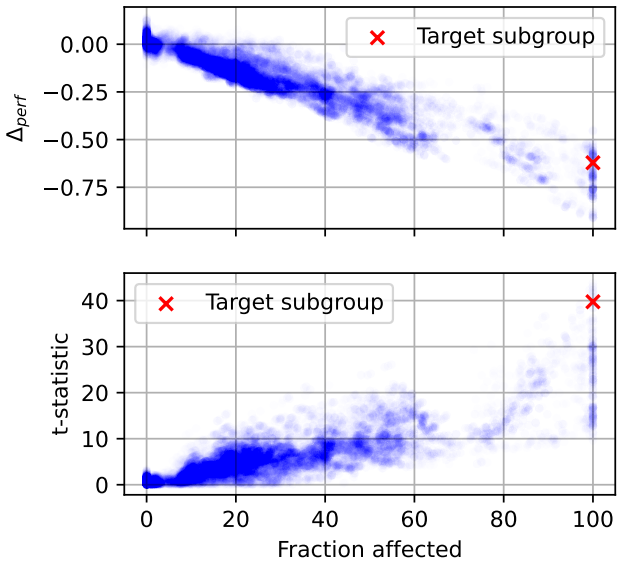

Figure 9: Correlation between the fraction of points that have been altered for each subgroup, and the detection of drift, in terms of Δ_{acc} and t-statistic.

u as the average number of items (metadata) used to describe each point, whereas l represents the average length of each subgroup, in terms of items. In the general case, we expect $u, l << |\mathcal{I}|$.

Based on this notation, we report below the time and space complexities of each step of DriftInspector.

The computation of the sparse matrix G is $O(|\mathcal{G}|\, l)$ in both space (to store the matrix in sparse format) and time. Similarly, the computation of the sparse matrix P is $O(N\, u)$ in space and time.

The computation of the membership matrix M is the result of the multiplication between sparse matrices. For each of the dot products between rows of P and columns of $G^\intercal$, only non-zero pairs of values need to be computed. To compute the total number of products that need to be computed, we need to estimate the number of non-zero pairs of values. We make the simplifying assumption of independence between P and $G^\intercal$ (we note, however, that this assumption is not always applicable, depending on the technique used for the extraction of the subgroups). For a given row of P and column of $G^\intercal$, the expected number of non-zero pairs is of $\frac{u}{|\mathcal{I}|}\ \frac{l}{|\mathcal{I}|}\ |\mathcal{I}|$ non-zero pairs. Thus, the time required for the sparse multiplication if $O(N\,|\mathcal{G}|\ \frac{u\,l}{|\mathcal{I}|})$. Based on the sparsity of the problem (as defined by u, l), this is generally much lower than the cost required for the non-sparse multiplication $O(N\,|\mathcal{G}|\,|\mathcal{I}|)$. It is more difficult to compute the space complexity, since it requires considerations on the relationship between points and subgroups. The lower bound can be established (assuming using a frequent pattern mining with a support threshold s) as $N\,|\mathcal{G}|\,s$. The upper bound is $N\,|\mathcal{G}|$, where all points belong to all subgroups (i.e., $s = 1$). We argue that this is, however, an extreme case where subgroups do not convey any useful information. The α, β vectors require $O(|\mathcal{G}|)$ in space and between $N\,|\mathcal{G}|\,s$ and $N\,|\mathcal{G}|$ in time for their computation. Finally, the computation of the t-statistics for each subgroup is $O(|\mathcal{G}|)$ in time and space.

The overall time cost in the worst case will be $N\,|\mathcal{G}|$, whereas in the best case it will be $N\,|\mathcal{G}|\,(\frac{u\,l}{|\mathcal{I}|}\,s)$. The overall space complexity will be between $N\,|\mathcal{G}|$ (worst case scenario) and $N\,|\mathcal{G}|\,s$ (best case).

D Experimental setting details

This section outlines the performance-based approaches at the overall level against which we compare our model. Specifically, we adopt DDM [16], HDDM [15], Page-Hinkley [31, 33], two algorithm based on sliding windows, ADWIN [6] and KSWIN [38] and the statistical tests χ^2 and FET. For the first 5 methods, we use the implementation available in the *scikit-multiflow* [29] library; for χ^2 and FET, we use the *alibi-detect* [43] library. DDM [16] detects a change if there is a significant increase in the error rate, specifically if the sum of the error rate and standard deviation at time t is greater or equal to the sum of the minimum error rate and three times the minimum standard deviation. DDM also includes a parameter, the minimum number of samples, to prevent false detections. This parameter determines the minimum number of samples that need to be analyzed before a change can be detected. We consider the values [500, 1000, 2000, 4000, 8000].

E Time comparison

In this Section we compare the execution time of various detection algorithms, when computed separately for each subgroup, against that of DriftInspector.

For comparability of the results, we take each detector and apply it to each subgroup separately (i.e., a separate detector is created for each subgroup). This approach would constitute a baseline method to be used as a competitor for DriftInspector . However, the execution times of this naive technique makes the performance comparison unfeasible. As such, we only report the test times, as measured on a limited number of batches.

The results are reported in Table 7. Thanks to the intrinsic support of subgroups, DriftInspector is approximately 100 times faster than the next fastest algorithms (χ^2 and Fisher's Exact Test), and as much as 2,000 faster than KSWIN.

Table 7: Execution times, normalized by the sample size, for the subgroup-wise detection of drifts. Each column represents a separate value for the minimum support sof the subgroups (i.e., a different number of subgroups being monitored).

	Execution time (s)		
	s= 0.01	s= 0.05	s= 0.1
χ^2	1.2867 ± 0.0251	0.1249 ± 0.0026	0.0359 ± 0.0005
ADWIN	2.1340 ± 0.0140	0.2095 ± 0.0033	0.0610 ± 0.0020
DDM	1.8865 ± 0.0242	0.1804 ± 0.0053	0.0519 ± 0.0031
FET	1.2876 ± 0.0406	0.1224 ± 0.0021	0.0350 ± 0.0003
HDDM	5.2180 ± 0.2554	0.4991 ± 0.0066	0.1423 ± 0.0037
KSWIN	25.1528 ± 0.9003	2.5874 ± 0.0184	0.7769 ± 0.0255
Page-Hinkley	2.3472 ± 0.0847	0.2204 ± 0.0025	0.0657 ± 0.0023
DriftInspector	**0.0120 ± 0.0001**	**0.0012 ± 0.0000**	**0.0004 ± 0.0000**