JAISCR, 2025, Vol. 15, No. 2, pp. 147 – 166
10.2478/jaiscr-2025-0008

A NOVEL METHOD FOR DRIFT DETECTION IN STREAMING DATA BASED ON MEASUREMENT OF CHANGES IN FEATURE RANKS

Piotr Porwik*, Tomasz Orczyk, Krzysztof Wrobel, and Benjamin Mensah Dadzie

Institute of Computer Science, Faculty of Science and Technology, University of Silesia, ul. Bedzinska 39, 41-200 Sosnowiec, Poland

**E-mail: piotr.porwik@us.edu.pl*

Submitted: 2nd October 2024; Accepted: 1st December 2024

Abstract

Hidden changes in the data stream are unknown to learning algorithms and are referred to in the literature as drifts of various types. The accuracy of the classifier may degrade due to the occurrence of drift in non-stationary data streams. In such situations, the classifier must detect significant data changes and adjust its predictions. This article aims to present a new method of drift detection based on analyzing changes in feature ranks across adjacent chunks of data. The proposed strategy involves determining the ranking of the most important feature and tracking its fluctuations within the chunks into which the input data stream is divided. Changes in feature rankings between adjacent chunks serve as symptoms of data drift. The Least Absolute Shrinkage and Selection Operator (LASSO) procedure was proposed as an efficient rank pointer. We compared well-known and popular drift detection algorithms, such as the Drift Detection Method (DDM), Early Drift Detection Method (EDDM), ADaptive WINdowing (ADWIN), and Principal Component Analysis Feature Drift Detection (PCA-FDD), with our approach in comparative studies. The tests were conducted on different artificial data streams (sudden, gradual, recurring, and incremental) as well as real data. Comparative studies were performed on both two-class and multi-class datasets. The experiments confirm that the proposed feature drift detection strategy produces valuable results.

Keywords: feature ranking, feature drift, data stream, drift detection, stream classification methods

1 Introduction

Drift occurrence is a common phenomenon in machine learning where the data distribution changes over time, leading to a degradation of a classification model's performance [1, 2]. This can occur in various domains, such as financial forecasting, image recognition, and natural language processing. The underlying data may change due to consumer behavior, technology evolution, sensor degradation, or external events. Detecting various types of drift (concept, data, or features) is essential to maintaining model accuracy, as ignoring it can lead to incorrect predictions, decreased customer satisfaction, and lost revenue [3]. Drifts can change the statistical properties of the data over time, leading to a degradation in model performance. Various methods, such as statistical tests, ensemble-based procedures, and distance-based measures, have been proposed to detect and respond to drift to address this issue [2, 4, 5, 6].

Concept drift and feature drift are critical in maintaining the performance of machine learning models, but their meanings are different. Concept drift occurs when the relationship between input features and the target variable changes over time. Changing the underlying data distribution can make a previously accurate model unreliable even if the features remain stable. In other words, concept drift alters the target variable's relationship with features, invalidating previous patterns.

Feature drift refers to changes in the input features' statistical properties without necessarily changing their relationship to the target variable. While feature drift doesn't always mean the model will perform worse, significant shifts in feature distributions can degrade the model's performance. Sometimes, a model might need to be retrained to account for these changes. It means that the feature drift also affects the input feature distributions, which may impact the performance of the decision model. Feature drift is a term that can refer to any changes in the features used in the model, which may include changes in their distributions, relevance, or predictive power. This term can encompass the covariate drift (distribution changes) as well as shifts in the significance of certain features (for example, a feature that was once relevant becoming less important over time). Feature drift may also be a concept drift, as changes in the features can affect how well the model performs. When features lose relevance or change significantly, it impacts the model's ability to predict the target variable accurately.

The amount of data and features used in machine learning and data mining constantly increases. Such a phenomenon occurs in non-stationary data, where information comes as a data stream. Streaming data from non-stationary distributions often results in data distribution known as drift. Therefore, there is also a problem with the non-stationary training of a classifier. This is an unfavorable phenomenon because, with a changing data stream, the efficiency of each classifier is volatile and must be counteracted. Classification is widely employed in data mining and machine learning. In the traditional approach, we try to learn concepts from a static dataset in which instances fall within a particular statistical distribution. However, this approach is unacceptable for many real-world scenarios, including medical data, biometric data, spam and fraud detection, observation of climate change, etc.

The data stream can be described as follows: a stream S appears incrementally in time as a sequence of examples with labels $\{\mathbf{x}^t, y^t\} \in S$, $t = 1, 2, \ldots$, where $\mathbf{x}$ is a d dimensional features vector. So, $\mathbf{x} = [x_1, \ldots, x_d]$ is one observation (data instance) among thousands of others. Each instance has a corresponding class label $y \in \{1, \ldots, C\}$. Each example $\mathbf{x}^t$ is classified by a classifier that predicts the appropriate class label from the available class pool. The true class labels y^t are available to update the classifier. In practice, drift is hidden in the data stream and is unknown to the learning algorithm. However, drift must be revealed to counteract this phenomenon to ensure correct classification. Concepts are stable if successive data come from the same or similar statistical distribution P. Concept drift primarily refers to the change in the decision model, while data or feature drift refers to the underlying data distribution. However, these two phenomena are related, and data drift may indicate a possible occurrence of a drift [7, 8]. Thus, we may assume that if in time t and $t+1$, we observe the input vector such that $P(\mathbf{x}^t, y^t) \neq P(\mathbf{x}^{t+1}, y^{t+1})$, then drift occurs [9, 10]. In other words, the data stream is non-stationary, so its data distribution might change over time. The distribution and nature of these changes can be very different and challenging to predict because the data flow is continuous, and their prior observation is impossible. Furthermore, the true label must be known to detect a feature drift by detecting degradation of classification accuracy. The true label can be known after a specific time, called verification latency. In some cases, this latency may be very high, or we may never get the true label; thus, detecting a concept drive by observing data drift may have serious advantages, despite being an indirect method.

Feature drift occurs when the rank importance of the features subset changes in the pair of adjacent timestamps. Similarly to the previous, suppose the feature space in an origin data stream consists of d features, $F_{tot} = \{x_1, \ldots, x_d\}$. Let in at timestamp t, the subset of the essential features that significantly influence classification accuracy will be denoted as $F_t = \{x_i, x_{i+1}, x_p\} \subset F_{tot}$. Let in timestamp $t+1$ such subset differ and will be marked as $F_{t+1} = \{x_m, x_n\} \subset F_{tot}$. A feature drift occurs when the relevance of attributes in both subsets changes between two timestamps t and $t+1$, respectively: $F_t \neq F_{t+1}$. The rank of features can be determined using the LASSO procedure, for example. In this way, feature drift is a specific variant of a concept drift [5].

Following the work of [11], we assume that features are relevant (important) in data analysis or machine learning when they significantly impact the target variable we are trying to predict. Relevant features contain helpful information that helps the model make accurate predictions or classifications. Features with little or no correlation with the target variable or introducing noise are generally considered irrelevant.

Feature relevance is often assessed using model performance evaluation, where the model's performance is compared with and without certain features to see how they affect the quality of the predictions. Identifying essential features can improve the model's accuracy and reduce its complexity. This method involves comparing models before and after the selected feature has been reduced. This is time-consuming. The Least Absolute Shrinkage and Selection Operator (LASSO) strategy is a technique that focuses on identifying the most relevant predictors in the context of regression. The LASSO method penalizes the absolute values of the regression coefficients. This penalty tends to reduce some coefficients, effectively selecting the subset of variables with the most vital relationships with the response variable. Each regression coefficient correlates with a given feature, giving natural ranking features from most relevant to least important.

1.1 Overview and main contributions

This paper aims to provide an overview of the feature drift problem and discuss various approaches for detecting and mitigating it. We propose a new method of feature drift detection based on statistical tests and the rank of features in the data stream. This study aims to comprehensively explain the feature drift problem to provide researchers and data scientists with the necessary knowledge and tools to detect drifts effectively in machine-learning models. We will compare various drift detectors, such as the DDM, EDDM, ADWIN, and PCA-FDD algorithms, using our approach to evaluate these detectors regarding their detection accuracy and ability to adapt to different data types.

The proposed method allows for drift detection with similar effectiveness as other drift detectors known from the literature. However, its main advantage is that fewer false positive alerts are generated than other modern drift detectors. This phenomenon can be observed in both artificial and real datasets.

1.2 Type of drifts

If a concept or feature drift exists, the classification model is inconsistent. Therefore, we must continuously track the input data, detect drift, and update the model regularly when drift occurs. The described operation consists of two stages. Drift must be detected first, then the decline in data classification quality must be counteracted. These mechanisms are well known and described in many variations in the extensive literature [4, 5, 12, 13]. In the dynamic context, we can separate feature drifts into sudden, gradual, recurring, and incremental [14, 15]. Drifts can also be described by recurring context – a concept that has arisen in the past may reappear after some time.

Abrupt (sudden) drift. Abrupt drift is a sudden and unexpected shift or change in a particular situation, condition, or trend. It can occur in various contexts, including scientific research, politics, economics, climate, or personal relationships. For example, in scientific research, an abrupt drift may occur when a study's results suddenly and unexpectedly diverge from the expected outcome, leading to a discovery of something. In economics, an abrupt drift could occur when the stock market experiences a sudden and unexpected crash or when a country experiences a sudden and unexpected recession. Overall, an abrupt drift can be disruptive and challenging to detect, with significant negative consequences.

Gradual drift. Gradual drift is a gradual shift or change in a particular situation, condition, or trend. Unlike abrupt drift, which occurs suddenly and unexpectedly, gradual drift happens over an extended period, and its effects may not be immediately noticeable [14]. For example in data science, gradual drift can refer to a gradual shift in the data collected over an extended period. This can sometimes lead to refining theories or models to explain the phenomena under investigation. Overall, gradual drift can be significant and have long-term consequences, particularly in complex systems such as ecosystems, economies, or societies.

Recurring drift. Recurring drift occurs when the underlying distribution of data changes over time in a cyclical or recurring manner. This means that changes in the data distribution are not constant but occur periodically or in cycles. Recurring drift is particularly difficult for machine learning models because models must adapt to the changing data distribution at the right time and in the right way. During this type of drift, the distribution may change either abruptly or gradually. For example, consider a machine learning model trained to predict stock prices based on historical data. When the stock market goes through periodic volatility cycles, the data distribution will change over time, and the model must adapt to these changes to maintain prediction accuracy [16].

Incremental drift. Incremental drift occurs gradually over time. Drift refers to changes in the underlying data distribution that the model tries to learn. Incremental drift occurs when these changes happen slowly and gradually over time rather than suddenly and dramatically. Incremental drift occurs when a series of barely noticeable changes in concept appear in the data stream. For this reason, drift can only be identified after a certain period, assuming a threshold at which it should be noticed [14, 17]. This type of drift is perpetual and infinite.

2 Features ranking

Feature ranking refers to the importance of different features or variables in a dataset. This process is often used in machine learning, data science, and statistics to determine the most relevant or informative features for predicting a target variable. There are various methods for ranking features, but one common approach is to use a statistical measure of features. Feature ranking methods are based on different principles. Therefore, the different ways can highlight the importance of multiple arrangements of features. We can apply, among others, the Fisher score [18], two-sample t-test [19], Kolmogorov-Smirnov test [20], Kruskal-Wallis ranks [21], ReliefF algorithm [22], Neighborhood Component Analysis (NCA) [23], RFE (Recursive Feature Elimination) [24], ANOVA, PCA [25], Random Forest Feature Importance (RF) [26] and LASSO (Least Absolute Shrinkage and Selection Operator) [8, 27].

When comparing different feature ranking methods for machine learning models, each method has strengths and weaknesses depending on the data type, the goal of the analysis, and the computational constraints. In our approach, data can have many features associated with suspected noise. Therefore, our investigations recommend the LASSO algorithm for feature ranking. Other ranking methods have various limitations [8, 28]: Fisher's score does not account for feature interactions. The two-sample t-test is only effective for binary classification tasks. The Kolmogorov-Smirnov Test does not scale well to multiclass problems and doesn't handle high-dimensional data effectively. Kruskal-Wallis Test does not measure feature importance but checks only their statistical significance. ReliefF Algorithm can be computationally expensive for large datasets or high-dimensional data. NCA is sensitive to overfitting, especially in high dimensions. RFE is computationally expensive since it requires multiple model fits. ANOVA procedure does not capture non-linear relationships. PCA doesn't rank features based on their predictive power but based on their variance. It may discard essential and useful features. RF does not generalize well if the dataset is small.

It should be noted that, although using an algorithm primarily applied for feature reduction, our strategy assumes that features are ordered according to their importance, but no features are ever removed. Due to the non-stationary nature of the input data, unimportant features in the current step may suddenly become essential in the next step. In this case, the drift detector focuses on observing a new feature that has become significant in the next chunk. This part of the data now becomes the new reference chunk.

2.1 Feature importance in the chunks

The instability of a concept arises from the variability in the data stream over time, resulting in the possibility of the classifier model being inconsistent for new data chunks. Thus, it is imperative to update the model regularly to prevent errors in the learning process. A detection mechanism should track mistakes online to detect these changes in real time. Over the last two decades, various methods have been proposed to identify concept and feature drift, including similarity and dissimilarity-based techniques, sequential analysis-based approaches, window-based methods, statistical-based methods, significance analysis-based methods, data distribution-based methods, and decision boundary-based methods. However, implementing some of these strategies can be challenging and sophisticated.

Our approach assumes that a data stream is a continuous flow of features that describe a phenomenon's state, device status, or monitored patient parameters. For instance, critical patient parameters in intensive care units, such as body temperature, blood pressure, blood saturation, heart rate, etc., are continuously measured. Doctors receive alerts and therapeutic suggestions based on these parameters. In our study, we focus on identifying changes in the most critical feature of the data stream, which can be treated as feature drift. To determine the validity of all components, we use the LASSO algorithm, as mentioned earlier.

We assume that the data stream is multidimensional and the samples flow continuously. The samples consist of an ordered vector of features $\mathbf{x}$ with a class label y: $(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^t, y^t)$ and the stream can include even thousands of samples. We also assume that the dimensionality of the data can be tens or hundreds of features. In our approach, we want to monitor the individual features x_j of the vector $\mathbf{x}^i$, $x_j \in \mathbf{x}^i$ and assess whether they are a source of potential drift. It should be noted that this strategy differs from those described in the literature, where drift is detected globally based on observing changes in classifier efficiency. In this respect, the proposed method is more precise and allows us to assess the change in the importance of features (informativeness) in adjacent data chunks.

The LASSO procedure identifies the most important features of the model in each chunk based on the β_i coefficient values. Then, without LASSO, the features are ranked according to the relationships:

$$R_i = \sum_{j=1}^{d} \mathbf{I}(x_j \leq x_i), \qquad i = 1, \ldots, d, \tag{1}$$

where R_i is a rank of a feature x_i, $i = 1, \ldots, d$. $\mathbf{I}()$ denotes the indicator function, and d is the number of all features.

In our approach, we set a maximum allowable fluctuation range for the top-ranked feature x_i in the reference part h. This range specifies the degree of change in the position of the feature x_i.

To illustrate this phenomenon, we present Fig. 1, which shows changes in the importance of feature x_i over time for various data streams. Initially, the feature is stable and highly significant for classification efficiency, as evidenced by the high efficiency of the classifier (red line) and the stable rank of the feature (green line) in the first data chunks. However, at some point, the importance of feature x_i decreases, causing a drop in the classification quality, indicating the occurrence of drift (blue line) for this feature. We compare the moment of drift detection with changes in the classification efficiency of the Random Forest (RF) classifier, known for its high accuracy. Fig. 2.1 illustrates that the feature x_i remains stable and stays below the threshold line in the chunks ranging from 0 to 49,999 but becomes unstable in the chunks between 50,000 and 99,999, indicating the occurrence of drift at the point numbered 50,000. The detection of the drift moment and the decline in classification accuracy exhibit a strong correlation. A similar phenomenon can also be observed in the other charts. As shown in Fig. 1, the accuracy of the RF classifier drops drastically when a drift is detected in the input data stream. We observe feature drift (green line) in one of the data stream features, and the moment of drift occurrence is marked in blue. Our approach provides a precise and effective way to detect drift in streaming data.

3 Mathematical foundation

Consider a sample of N instances (observations), each consisting of d-dimensional vector $\mathbf{x}_i = [x_1, \ldots, x_d]$ with $\mathbf{y} = [y_1, \ldots, y_N]^T$ a response vector, and $i = 1, \ldots, N$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in C$. In practice, elements of the vector $\mathbf{x}_i$ are features of the object. Let $y_i \in \{1, -1\}$ be the label of the vector $\mathbf{x}_i$. Given

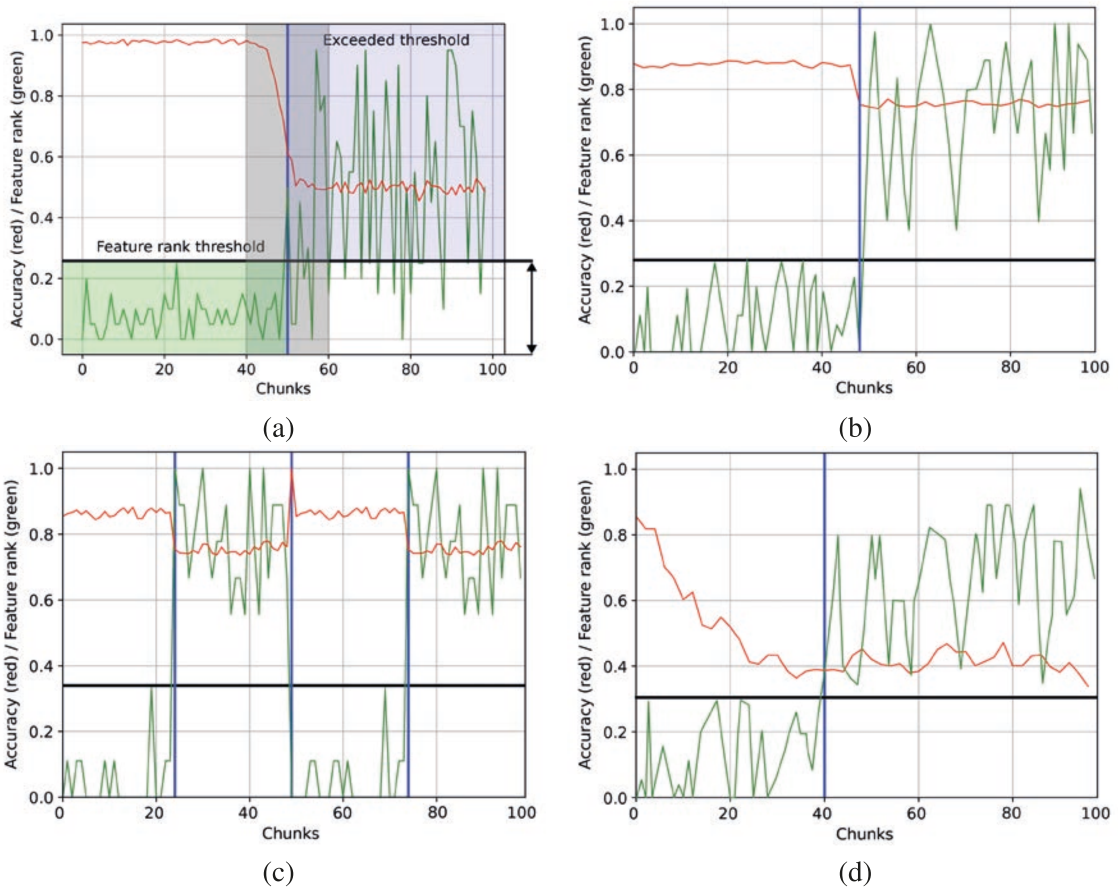


Figure 1. Rank of an essential feature in subsequent chunks of data. Gradual changes (a), Abrupt changes (b), Recurring changes (c), and Incremental changes (d). The gray rectangle in (a) denotes the duration of the drift recognized by the classifier.

a dataset with N observations and d predictors, the linear regression model is described by the formula:

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_d x_{id} + \varepsilon_i, \quad i = 1, \ldots N, \tag{2}$$

where: y_i - is the response variable, x_{ij} - is the j-th predictor for the i-th observation, β_j are the coefficients, ε_i is the error term, assumed to be normally distributed with mean zero.

The Least Absolute Shrinkage and Selection Operator (LASSO) is a regression analysis method that performs variable selection and regularization to enhance statistical models' prediction accuracy and interoperability. The LASSO regression aims to minimize the following function:

$$\min_{\beta} \Big(\frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \Big), \tag{3}$$

where $\beta = [\beta_1, \ldots, \beta_d]$ is the coefficients vector, and λ regularization parameter that controls the penalty term's strength. The parameter β_0 in the LASSO regression equation (3) represents the intercept term of the linear model. The coefficients β_i represent the impact of particular features of the linear numerical model on the data representation. Element x_{ij} is a value of the j-th feature in the i-th observation. Parameter λ is a predetermined free factor determining the regulation degree (penalizing parameter). The term $\lambda \sum_{j=1}^{d} |\beta_j|$ is the L1 penalty. If λ increases from zero, LASSO shrinks the coefficients of less important features toward zero. Some coefficients become zero at specific points along this path, removing those features from the model. Unlike the original LASSO procedure, in the method proposed in this article, features are not removed but are ranked.

From the LASSO point of view, the most crucial feature is in the first place, and the least important feature is in the last place in the formed rank. In our method, we always analyze the drift of the most essential feature. This phenomenon was used in the proposed approach, which analyzed the rank of all features, focusing on the first, most important feature. The LASSO method uses regularization to shrink certain coefficients β_j, effectively selecting a subset of features x_j in observation i. In practice, the LASSO method can come in three variants: L1 Lasso ($\lambda \sum_{j=1}^{d} |\beta_j|$) regularization, L2 Ridge ($\lambda \sum_{j=1}^{d} \beta_j^2$) regularization, and Elastic Net regularization, which combines both L1 and L2 penalties. Ridge regression results can be less interpretable due to including all features, each with a reduced but non-zero coefficient.

Lasso regression can improve interpretability by selecting only the most relevant features, making the model's predictions more explainable. Three variants of the LASSO procedure will be checked for artificial and real-world datasets.

It should be clearly emphasized that the above idea of the LASSO strategy is used in this article only to determine the rank of features, and that is all. No steps are taken to perform regression analysis on the available data set.

4 Proposed method

The Feature-Based Drift Detector is a method used to detect feature drift (and indirectly a concept drift) in data streams by monitoring changes in the distribution or behavior of individual features. It aims to identify whether the statistical properties of features have shifted, indicating a possible change in the underlying data-generating process. The proposed approach helps identify subtle shifts in feature distributions that may not immediately affect overall model performance but could cause issues in the long run. The proposed stream data analysis consists of some steps:

- *Baseline ranking of features:* Calculate the initial Lasso ranking of features in a data reference window (h_0).
- *Monitor incoming data:* Continuously monitor ranks of features from the incoming data stream. This is done in fixed-size windows.
- *Statistical comparison:* Perform statistical tests to compare the distribution of each feature in the new data window with the baseline distribution.
- *Detect drift:* If significant changes are detected in the distribution of the most crucial feature (based on permissible threshold fluctuation), flag a feature-level drift.
- *Adaptation decision:* Determining whether

Algorithm 1 Determining position of the importance of features x_i in the chunk h_0, standard deviation (σ) for such feature, and its average position (μ).

```
1. Take $1^{st}$ chunk of data, name it $h_0$.
2. Divide a chunk $h_0$ into set of 10 sub-chunks denoted as $\{h_0^1, \ldots, h_0^{10}\}$.
3. For each subchunk $h_0^i$ apply *Lasso* to obtain a feature rank vector denoted as $[rank(x_1), \ldots, rank(x_d)]^i$.
4. Compute mean ranks of the features $[AVG\{rank(x_1)\}, \ldots, AVG\{rank(x_d)\}]$ over all 10 sub-chunks rank vectors.
5. Compute standard deviation of ranks of the features $[\sigma_{rank(x_1)}, \ldots, \sigma_{rank(x_d)}]$ over all 10 sub-chunks rank vectors.
6. Find a feature $x_{min}$ with best average rank $x_{min} = \arg\min_{x_j} [AVG\{rank(x_1)\}, \ldots, AVG\{rank(x_d)\}]$.
7. Return a standard deviation of the most discriminating features $\sigma = \sigma_{rank(x_{min})}$ and
   a mean value of the most discriminating features $\mu = AVG\{rank(x_{min})\}$.
```

Algorithm 2 Alarm for a given classifier if the drift was detected.

```
1. Check a standard deviation of the most discriminating features $\sigma = \sigma_{rank(x_{min})}$
   in the current chunk $h_c$.
2. Check a mean value of the most discriminating features $\mu = AVG\{rank(x_{min})\}$.
3. if $\lceil \mu - \sigma < \mu < \mu + \sigma \rceil$ then
   take the next chunk $h_i$ into consideration
   else
   ALARM. A given classifier should be learned from the current data.
   Learn a given classifier on the data from the chunk $h_c$.
```

model retraining is needed based on the relevance of rank changes the monitored feature.

The steps mentioned above are implemented by Algorithms 1 and 2.

In the proposed strategy, some statistics are computed for the reference chunk h_0 and in all subsequent chunks h_i where the drift was detected. In these cases, the average position fluctuation (μ) of the most discriminative feature is performed, and in the next stage, sample standard deviation (σ) is also determined. The method of determining these statistical parameters is presented in Algorithm 1.

Algorithm 1 allows us to determine the range of changes in the position of the essential feature in the reference chunk h_0, calculate the mean value of this position, and the sample standard deviation. The most essential (discriminative) feature is also checked in the remaining chunks $h_1, h_2, ...$, and its position is determined in a pool of features, as shown in Algorithm 1. This position should be within the $\lceil \mu - \sigma < \mu < \mu + \sigma \rceil$, then drift is undetected. We assume drift occurs if the place lies outside the mentioned range.

In the proposed method, the detection threshold for drift is determined by utilizing the average position (μ) and sample standard deviation (σ) of the most crucial feature within the chunk. However, in classical statistics, it is recommended to avoid using the average when the nature of the data distribution is unknown, as outliers can significantly influence the average value. In such cases, a median that is immune to outliers is preferred. Despite this, our approach considers the possibility of outliers occurring in the data stream, which enables us to determine the extent of changes in the position of the essential feature in subsequent chunks.

If drift is detected, an alert may be generated to advise the classifier reviewing incoming data to be retrained. Because the concept of data has changed, a new chunk of data must now be treated as reference data. Alerts are generated as long as feature drift is detected. This task is performed by Algorithm 2.

In chunk-based methods, determining the number and size of chunks is crucial. Ideally, the number of detected drifts should be independent of the number of fragments in the source data stream. However, this assumption is unrealistic. In all studies, the rule was that the first chunk did not contain drifts. Additionally, all chunks are the same size, covering 5% of a given dataset.

Example

Let a given chunk h_0 consist of a set of N samples. Each sample consists of 20 features and is written as a feature vector, which means that such vector has a form $\mathbf{x}_i = [x_0, \ldots, x_{19}]$, $i = 1, \ldots, N$. For simplification, let the chunk h_0 be divided into three sub-chunks only (see Table 1). For each sub-chunk, the Lasso procedure is performed.

The first row of Table 1 shows the ordered features of each sample. The following three lines of the Table show the importance of all features indicated by the Lasso in each of the three sub-chunks. For example, feature no. 0 in sub-chunk h_0^1 is ranked by Lasso on position 11 and in sub-chunk h_0^3 on position 13. The most crucial feature (the highest rank) pointed out by Lasso is a feature with the number 8. After a trivial calculation, we have:

- the essential feature in the chunk h_0: feature no. 8,
- the average value of the position of feature no. 8 is $\mu = (4+2+8)/3 = 4.66$,
- the sample standard deviation $\sigma = 3.06$,
- the $threshold_{x_8} = \lceil \mu + \sigma \rceil = 8$.

The threshold value informs the range in which the position of a given feature can change (fluctuate) so that this change is not treated as drift. The threshold value is automatically computed without the need to set additional parameters. The Algorithm always establishes the parameters mentioned above.

5 Data preparation

Tests on the effectiveness of the proposed detector have been carried out on artificial collections, where the drift location can be programmed, and on real-world data taken from repositories. This study considered both two-class and multi-class data, which will assess the method's universality.

The detector described in the article compares favorably with other drift detectors regarding the

Table 1. Short steps of Algorithm 1 for the chunk h_0 which was divided into sub-chunks h_0^1, h_0^2, h_0^3

	0	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	16	17	18	19
h_0^1	11	3	15	14	13	2	12	5	4	1	7	16	8	10	6	19	18	9	17	0
h_0^2	11	13	14	5	15	7	9	18	2	10	19	3	4	6	8	1	12	0	16	17
h_0^3	13	7	11	14	10	16	13	15	8	9	1	3	19	5	4	2	6	18	17	0
Average feature rank (μ)																				
	11.7	7.6	13.3	11.0	12.6	8.3	11.3	12.6	**4.6**	6.6	9.0	7.3	10.3	7.0	6.0	7.3	12.0	9.0	16.6	5.6

number of false alarms generated - this is the main advantage of the proposed method. Experiments were conducted using artificial and real data sets to demonstrate this advantage. The artificial data were obtained using a data stream generator, where the type of drift, the place of its occurrence, and the level of data noise can be programmed. It allows for a reliable, comparative statistical evaluation of the behavior of different drift detectors.

5.1 Synthetic data

Synthetic data were prepared in the MOA environment (https://moa.cms.waikato.ac.nz/) using the command line interface. Synthetic data were produced by a HyperPlane Generator (HPG). The HPG generates the two-class data stream where the number of instances (m), number of features (a) in an instance, as well as the number of drifting features (k), place of drift occurrence (p) and time of drift (w), can be programmed. For each instance, (n)% noise can also be added.

From mathematical point of view, the hyperplane H is a set of multidimensional points such that $H = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} = b\}$, where $\mathbf{w} = [w_1, \ldots, w_a]$, $w_i \in R$, $\mathbf{x} = [x_1, \ldots, x_a]$ $x_i \in R$ and $b \in R$. We assume that $\sum_{i=1}^{d} w_i = b$. Instances are randomly generated, and each instance has the same pre-defined number of features x_i. Instances $\mathbf{x}$ for which $\mathbf{w}^T\mathbf{x} > b$ belong to the first class, whereas instances with condition $\mathbf{w}^T\mathbf{x} < b$ belong to the second class. A drift can be added to each weighted feature $w_i = w_i + \alpha$, where α is the change applied to every example. A sample command calling the MOA environment is shown below.

```
java -cp moa.jar moa.DoTask ”WriteStream-
ToARFFFile -s (ConceptDriftStream -s (genera-
tors.HyperplaneGenerator -a 20)
-d (generators.HyperplaneGenerator -i 10 -k 20
-a 20 -n 5) -p 50000 -w 1) -f Abrupt-HP.arff -m
100000”
```

Running the command generates a file called Abrupt-HP.arff. The file contains 100,000 instances (parameter m). Each instance consists of 20 features (parameter a). Drift occurs in instance no. 50,000 (parameter p). Drift is recorded on 10 features (parameter k). The data is distorted by 5% noise (parameter n). Parameter $-i$ denotes seed for random generation of instances. The reader can refer to the MOA documentation for more detailed information on selecting switches in its command line interface. Experiments have implemented four drifts: abrupt, gradual, recurring, and incremental with various data perturbations with noise.

5.2 Real data

This paper also presents the results of experiments based on real data. The study examined several open and publicly available repositories from where the data was downloaded. The multivariate multi-class datasets have been employed.

As mentioned below, real datasets are helpful for drift detection and adaptation research. Data evolve in these sets, meaning the relationship between the features and the target label changes. This makes the datasets valuable for testing machine learning models that need to adapt to these evolving trends.

For this purpose, seven real data streams have been checked with the following characteristics:

Phishing. The dataset gathers information about phishing websites and is used to identify the phishing attack. It includes 11,055 instances and 46 attributes distributed into two classes. Repository: https://github.com/tegjyotsingh/ConceptDriftInduction.
Electricity. The electricity market dataset represents pricing data collected from New South Wales (Australia), which fluctuates based on the supply and demand components of the market. The dataset consists of 45,312 instances and 8 at-

tributes distributed into two classes. Repository: https://sites.google.com/view/uspdsrepository.

Ozone. There are two sets of data on ground-level ozone in this collection. One is a set of eight-hour peak values, and the other is a one-hour peak value. These data were collected in three US cities. The database comprises a collection of 2,534 examples with 72 features in two classes. Repository: https://archive.ics.uci.edu.

Arrythmia. The dataset describes problems related to human cardiac arrhythmias. The database includes 452 instances with 279 features divided into 16 classes. Repository: https://archive.ics.uci.edu.

Poker-hand. This dataset exhibits minimal feature drift since the classification task is based on the stable rules of poker, which do not change frequently. Given its stability, the poker hand dataset does not face significant drift challenges, and traditional drift detection methods may not be necessary. Overall, this dataset serves as a benchmark for classification tasks.

Each record of this dataset is a poker hand consisting of five playing cards drawn from a standard deck. Each card is described by two attributes (suit and rank). The dataset contains 1,025,010 instances, 11 attributes, and 10 class labels related to a possible poker hand. Repository: https://archive.ics.uci.edu.

Rialto Bridge Timelapse. Ten colorful buildings next to the Rialto Bridge in Venice were photographed at different times of the day. The color of the buildings changes significantly during each of the 20 days recorded. Images are encoded in a normalized 27-dimensional RGB histogram. The dataset contains 82,250 instances, 27 features, and 10 class labels. Repository: https://sites.google.com/view/uspdsrepository.

Outdoor. Forty objects were photographed under varying lighting conditions, affecting the color-based representation. The objects are encoded in a normalized 21-dimensional RG-chromaticity histogram. The database includes 40,000 samples, 21 features, and 40 classes. A repository is accessible from the link: https://sites.google.com/view/uspdsrepository.

6 Experiments and evaluation of the proposed method

All computational experiments were conducted in Python 3.8 with the scikit-learn stream-learn, scikit-multiflow, NumPy, and Frouros 0.6.1 packages. The mentioned libraries allow research on new algorithms for batch processing of data streams.

6.1 Evaluation metrics

The accuracy measure (ACC) and Matthews correlation coefficient (MCC) were considered for measuring the algorithm's performance. In binary and multiclass classification:

$$ACC = \frac{1}{N}\sum_{i=0}^{N-1} I(y_i, y_i^*), \tag{4}$$

and

$$I(y, y^*) = \begin{cases} 0 & \text{if } y \neq y^* \\ 1 & \text{if } y = y^* \end{cases},$$

where, N denotes the number of instances, y is a real instance label, and y^* is a predicted label of class. $I()$ is an indicator function.

The problem with (2) is that it is not an informative metric when the two classes have significantly different sizes. The MCC metric, although not necessarily the best choice in all cases (see, e.g. [29]), often yields a more helpful performance score than ACC [30]. For the binary classification and multiclass (which is automatically recognized by scikit-learn implementation of this metric), we have the formulas:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}, \tag{5}$$

$$MCC = \frac{c \times N - \sum_{i=1}^{C} p_i \times t_i}{\sqrt{(N^2 - \sum_{i=1}^{C} p_i^2)\times(N^2 - \sum_{i=1}^{C} t_i^2)}}, \tag{6}$$

where:
TP - True Positive, TN - True Negative, FP - False Positive, FN - False Negative,
C - number of classes $c_i \in C$,

N - number of samples,
c - number of samples correctly predicted,
t_i - number of times class i truly occurred,
p_i - number of times class i was predicted.

If the data are divided into chunks, then the average values of ACC and MCC are computed according to the formulas:

$$ACC_{Avg} = \frac{1}{W}\sum_{j=1}^{W}\left[\frac{1}{N}\sum_{i=1}^{N} I(y_i = y_i^*)\right]_j, \quad (7)$$

and

$$MCC_{Avg} = \frac{1}{W}\sum_{j=1}^{W}[MCC]_j, \quad (8)$$

where W denotes the total number of chunks, N denotes the number of instances in a current chunk j.

6.2 Evaluation based on synthetic data

We compared the proposed features drift detection method (Algorithm 1) with the Random Forest (RF) classifier results without and with a signal that the classifier should be rebuilt after drift detection. The key idea behind RF is to combine the predictions of multiple decision trees, each trained on a random subset of the training data and a random subset of the features. This randomness helps reduce overfitting and improves the classifier's model. For the reader's convenience, the experiments in the form of graphs show two cases (a) the moment of detecting the drift and the natural decreasing of classifier accuracy, and (b) the reaction of the classifier after the detector information that a drift was detected. Sudden, gradual, recurring, and incremental drifts have been checked. Examples that well illustrate this principle are shown in the form of graphs in Figure 2 – Figure 5.

The blue line indicates the signal generated by Algorithm 1, which detected feature drift. We can observe that the classification accuracy (red line) decreases significantly after drift occurrence. The observation of the graphs shows that the proposed method detects feature drifts correctly and sufficiently early. Classification accuracy improves after classifier re-learning.

One drift can be detected several times, as the concept changes in some drifts, which are fluid and long-lasting. The proposed method is sensitive to transient concepts and can detect an early phase of drifts. It was presented in the middle chart of Figure 3. This is, however, an advantageous phenomenon. In such a case, the classifier can be retrained using the transition patterns between the two concepts, potentially affecting classification accuracy.

A comparative study shows that the proposed strategy of drift detection gives good results - drift is recognized early, and the classifier can be quickly rebuilt. It can be seen that after re-training on new data, the classifier quickly achieves high efficiency, similar to that before the occurrence of the drift. This ultimately means the classifier makes a prediction error in a short time interval after drift detection.

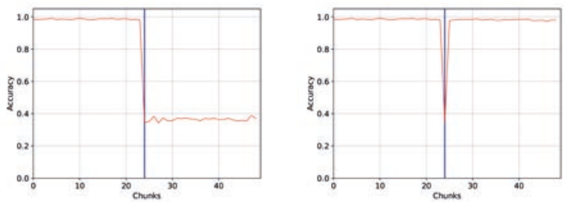


Figure 2. Abrupt drift. Drift detection is without (left) and with a retraining RF classifier (right).

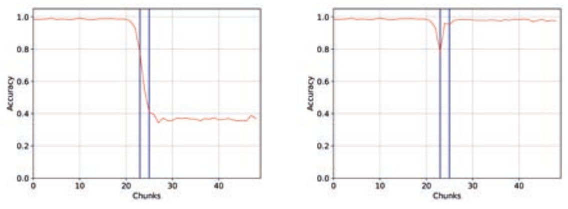


Figure 3. Gradual drift. Drift detection is without (left) and with a retraining RF classifier (right).

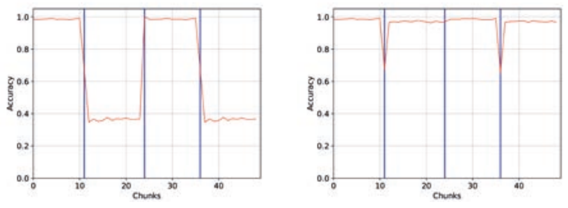


Figure 4. Recurring drift. Drift detection is without (left) and with a retraining RF classifier (right).

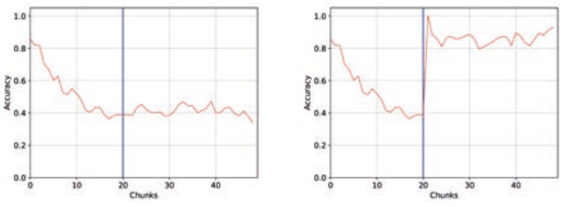


Figure 5. Incremental drift. Drift detection is without (left) and with a retraining RF classifier (right).

Algorithm 1 detects drift directly in the source data, rather than on intermediate indications, after processing by the selected classifier, which must be chosen beforehand. This means that in other methods of detecting drift, the detector cooperates with the selected classifier, unlike our process, in which it is unnecessary. We attach the classifier when classification is required. We can observe that the proposed drift detection method is consistent with changes in the RF classifier accuracy (red line). Illustrating the phenomenon is helpful but is not sufficient in comparative research. For this reason, tests were carried out for data with different noise levels (1%, 3%, 5%) and with a different number of simultaneously drifting features (10 or 20). The artificial streams are balanced, two-class, and have 20 features. A common rule is that the first chunk is a drift-free reference set and contains 5% of all data in the data set. In the artificial data stream generator, one drift was programmed. The drift was implemented regardless of the speed of data changes.

The proposed drift detector was compared with other of-the-art detectors. Results are gathered in Table 2. In Table 2, in the column "Proposed," are the results for the Lasso (L1) procedure. For the same data, calculations for Ridge (L2) and Elastic Net (L1+L2) have also been carried out, getting the same results. For these reasons, L2 and Elastic Net results have been omitted. The nature of artificially generated data means that features are random and uncorrelated, which means that the order of features determined by LASSO will be the same for L1, L2, and Elastic Net versions. This means the FBDD detector will obtain the same results regardless of the LASSO version. This is due to the properties of the LASSO method. If the features are uncorrelated, the differences between L1, L2, and Elastic Net regularizations are low or non-existent.

For our purposes, the $\lambda = 0.001$ factor was set in all variants of the LASSO procedure (see eq. 3). For the Elastic Net variant, the $ratio = 0.5$ was set. The *ratio* factor defines the mix of Lasso (L1) and Ridge (L2) penalties.

The result was also confirmed using the Wilcoxon test, as shown in Table 3, on the α level equal to 0.05. Statistical tables are used to calculate p-values for small samples (< 20), which is the case here. For p-value $< \alpha$, we can conclude that statistical evidence at $\alpha = 0.05$ shows that the average ACC for detectors differs. The Wilcoxon test shows statistical differences between the results of the proposed detector and the results of all other modern detectors listed in Table 3.

This test shows statistically significant differences for Abrupt changes in the datastream between the proposed method and DDM, EDDM, ADWIN, and PCA-FDD drift detectors. Our method has a clear advantage here. For Gradual and Incremental changes in input data, our method statistically gives the same ACC results as DDM, ADWIN, and PCA-FDD detectors. The proposed new method shows a statistically significant difference in the cyclical changes of Recurring-type data compared to all other strategies. This is beneficial from the point of view of the usability of the method for this type of data. The ones presented in Table 2 and Table 3 should be interpreted more subtly. Drift detectors cooperate with the RF classifier. This classifier shows high efficiency (ACC) for all types of drift for EDDM detectors. However, this happens at the cost of many false alarms (False Positive). Each false alarm of the EDDM detector causes the classifier to be retrained on new data, artificially increasing its momentary efficiency. This happens regardless of the number of drifting features and the noise level. This means that EDDM shows high sensitivity to any changes. In contrast, other detectors showed lower false alert values, and as promoted in this paper, our strategy was the best in this respect.

It should be noted that only one drift was programmed in the artificial data from Table 2. Thus, according to any detector's indications, the RF classifier should be rebuilt only once for ideal drift detection. However, this does not happen, and the detectors, in addition to correctly indicating, incorrectly detect drifts in places where they do not exist. Our method is the exception, where the reported

Table 2. The accuracy ACC of drift recognition by RF classifier for different detectors with the number of detected drifts. "feature-noise" means a number of drifting features and % of noise in data.

Drift	feature-noise	Proposed (Lasso)	DDM	EDDM	ADWIN	PCA-FDD
Abrupt	10-1	**0.867/1**	0.851/7	0.861/18	0.861/1	0.802/4
	10-3	**0.858/1**	0.842/1	0.852/18	0.853/1	0.796/4
	10-5	**0.851/1**	0.835/3	0.844/18	0.845/1	0.790/4
	20-1	**0.868/1**	0.850/3	0.862/12	0.860/1	0.794/3
	20-3	**0.861/1**	0.843/3	0.854/18	0.850/1	0.789/3
	20-5	**0.854/1**	0.836/4	0.843/10	0.844/1	0.784/3
	Avg. drifts	1	3.5	11.3	1	3.5
Gradual	10-1	0.828/2	0.822/3	**0.841/16**	0.823/3	0.760/0
	10-3	0.823/2	0.812/3	**0.833/16**	0.814/3	0.756/0
	10-5	0.819/2	0.813/4	**0.828/18**	0.806/3	0.752/0
	20-1	0.776/2	0.803/5	**0.818/18**	0.796/4	0.787/2
	20-3	0.758/2	0.790/4	**0.811/19**	0.794/4	0.781/2
	20-5	0.755/2	0.784/5	**0.802/17**	0.784/4	0.715/2
	Avg. drifts	2	4	17.3	3.5	1
Recurring	10-1	**0.878/3**	0.836/8	0.860/18	0.855/3	0.832/8
	10-3	**0.865/3**	0.838/6	0.843/18	0.842/3	0.818/8
	10-5	**0.851/3**	0.814/6	0.828/18	0.823/3	0.801/8
	20-1	**0.880/3**	0.857/4	0.863/18	0.857/3	0.805/3
	20-3	**0.867/3**	0.835/3	0.845/18	0.842/3	0.794/3
	20-5	**0.854/3**	0.816/7	0.832/19	0.824/3	0.779/3
	Avg. drifts	3	5.6	18.2	3	5.5
Incremental	10-1	0.797/2	0.795/5	**0.815/19**	0.803/3	0.795/2
	10-3	0.780/1	0.787/2	**0.806/16**	0.788/3	0.788/2
	10-5	0.798/2	0.787/2	**0.806/16**	0.788/3	0.782/2
	20-1	0.754/2	0.751/5	**0.764/18**	0.751/2	0.720/4
	20-3	0.749/2	0.728/3	**0.758/18**	0.747/3	0.716/4
	20-5	0.744/2	0.722/8	**0.750/19**	0.736/2	0.713/4
	Avg. drifts	1.8	4.2	17.7	2.7	3

Table 3. Wilcoxon Signed-Rank-test, using W+ distribution (two-tailed) for $\alpha = 0.05$. Synthetic data for ACC values. Values inside Table are p-values of the statistic.

Proposed vs.	DDM	EDDM	ADWIN	PCA-FDD
Abrupt	0.031	0.033	0.035	0.031
Gradual	**0.400**	0.031	**0.441**	**0.141**
Recurring	0.031	0.034	0.034	0.035
Incremental	**0.156**	0.031	**0.600**	**0.094**

number of false alarms is the lowest compared to other drift detectors. This is the most crucial advantage of our method. The average results of true drift (TP) and false alert (FP) for different artificial data sets are summarized in Table 4. It is visible that the proposed strategy realized by the new detector generates fewer additional alerts in comparison with other detectors.

6.3 Evaluation based on real data

Similar studies, as in the previous section, were performed for the case of real data [31]. The characteristics of these data are given above. Real data also contains multi-label examples, which will better enable the evaluation of the proposed method against others.

The research was conducted similarly to synthetic data for the identical drift detectors. The training data was constructed from the initial 5% of the specified dataset. In contrast to artificial data pipelines, real data streams do not have defined drift points. Hence, the occurrence of drift and its type in such data are unknown. For this reason, the assessment of the usefulness of our method against the background of others known from the literature was carried out by analyzing the ACC and MCC indicators. Results are gathered in Table 5.

The datasets used in the study have more or less correlated features. These are real-world data, and even the names of these sets suggest that the data are at least partially correlated. The exact characteristics of these sets have been presented earlier. For more highly correlated features, expect different results in the ”Proposed” column of Table 5 for Lasso, Ridge, and Elastic Net versions.

While both ACC and MCC measure classification performance, they do not always correlate directly, especially in imbalanced datasets. High ACC, Low MCC: A model might achieve high accuracy in imbalanced datasets by predominantly predicting the majority class. However, MCC would be low because it penalizes incorrect minority class predictions more heavily. High MCC, High ACC: It generally indicates that the model performs well in both classes, with a balanced rate of true positives and true negatives. Low ACC, High MCC: This indicates issues with imbalance or misclassification that accuracy alone does not reveal. When analyzing the results in Table 5, these relationships were taken into account.

Additionally, observation of the MCC coefficient is very valuable because from eqs. (5) and (6) follows that MCC decreases when FP increases. This reflects worse performance, as more false positives indicate that the model incorrectly predicts positives. The analysis of the MCC index changes in Table 5 leads to the conclusion that all real data streams in our method give the highest MCC values. This suggests that the proposed detector and the RF classifier generate the fewest false alarms among all the other detectors. Therefore, the number of true drifts true positives (TP) is the most reliable in our method. Accuracy (ACC) also decreases when false positives (FP) increase. Thus, the accuracy decreases when FP increases, indicating that the model makes more incorrect positive predictions, leading to worse overall performance.

The classification efficiency results were also checked here using the Wilcoxon test. The test results are summarized in Table 6. The Wilcoxon test shows statistically significant differences for MCC between the proposed drift detector and all others used in the experiment. This means there are statistically significant differences in false alarms (FP) occurrence in different drift detectors, which was already discussed above. There are no statistically significant differences in the effectiveness of the proposed detector and the ADWIN and PCA-FDD detectors. However, it should be noted that ADWIN and PCA-FDD generate more false alarms (FP) than the proposed method because the MCC coefficient is lower than that of the proposed method.

7 Evaluation of the operating time of drift detectors

Tests were performed on a computer with an AMD Ryzen 7500 processor, 3.7 GHz, 64 GB main memory, and Windows 10 operating system. Evaluation times (running time in seconds) are presented separately in synthetic and real data graphs. The best methods depend on both the dataset and base detector. It is a trivial conclusion but allows showing these dependencies for analyzed datasets and selected drift detectors.

Table 4. Average TP and FP indices value for different drift detectors based on artificial datasets from Table 2.

Detector	Abrupt		Gradual		Recurr.		Incre.		Avg.	
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
Reference	1	0	1	0	1	0	1	0	1	0.00
Proposed	1	0	1	1	1	2	1	0.8	1	0.95
DDM	1	2.5	1	3	1	4.7	1	3.2	1	3.35
EDDM	1	14.6	1	16.3	1	17.1	1	16.7	1	16.18
ADWIN	1	0	1	2.5	1	2	1	1.7	1	1.55
PCA-FDD	1	2.5	0.5	0.5	1	4.5	1	2	1	2.38

Table 5. The accuracy (ACC) and Matthews coeff. (MCC) of drift recognition.
The RF classifier for seven Real datasets.

	Ridge	Elastic	Proposed (Lasso)	DDM	EDDM	ADWIN	PCA-FDD	coeff.
Phishing	0.924/0	0.943/1	0.943/1	0.942/6	0.940/9	0.937/1	0.940/15	ACC
(2 classes)	0.846	**0.886**	**0.886**	0.762	0.879	0.873	0.879	MCC
Electricity	0.779/3	0.796/4	0.796/4	0.762/18	0.759/18	0.745/13	0.724/2	ACC
(2 classes)	0.544	**0.601**	**0.601**	0.509	0.504	0.482	0.441	MCC
Ozone	0.929/11	0.941/12	0.943/13	0.789/3	0.893/4	0.897/2	0.892/13	ACC
(2 classes)	0.538	0.651	**0.704**	0.174	0.121	0.072	0.005	MCC
Arrhythmia	0.748/10	0.945/18	0.945/18	0.550/0	0.583/3	0.550/0	0.563/17	ACC
(16 classes)	0.556	**0.905**	**0.905**	0.114	0.261	0.114	0.155	MCC
Poker-hand	0.718/5	0.718/5	0.642/6	0.642/0	0.640/6	0.642/0	0.642/0	ACC
(10 classes)	0.472	0.472	**0.503**	0.332	0.327	0.332	0.332	MCC
Rialto Bridge	0.561/2	0.633/0	0.645/1	0.619/18	0.628/19	0.648/17	0.649/10	ACC
(10 classes)	**0.615**	0.595	0.609	0.577	0.587	0.609	0.614	MCC
Outdoor	0.230/0	0.497/7	0.637/11	0.230/0	0.255/12	0.234/4	0.247/14	ACC
(40 classes)	0.222	0.474	**0.634**	0.220	0.236	0.218	0.238	MCC

Table 6. Wilcoxon Signed-Rank-test, using W+ distribution (two-tailed) for $\alpha = 0.05$. Seven Real data sets. Values inside Table are p-values of the statistic for ACC and MCC values.

	DDM	EDDM	ADWIN	PCA-FDD
Proposed (ACC) vs.	0.031	0.015	**0.063**	**0.094**
Proposed (MCC) vs.	0.016	0.016	0.031	0.031

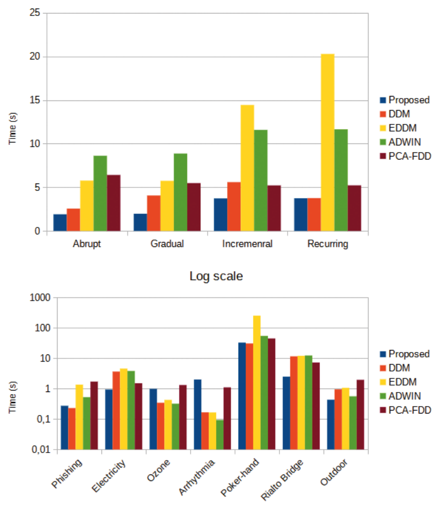


Figure 6. Drift detectors' operating time depending on the input data type.

The charts show that the proposed strategy was the fastest method on all synthetic datasets. Results for real data are presented on a logarithmic scale due to the Poker-hand file. It is a huge file (over 1,000,000 instances, 11 attributes, and 10 classes), and its processing time is disproportionately long (for the EDDM detector, 251 seconds) compared to the other data. For real data, the proposed approach gave the shortest computation time for the Phishing, Electricity, Poker-hand, Rialto-Bridge, and Outdoor sets. The favorable time results of the proposed method result from the fact that it does not generate warning levels like other detectors and immediately indicates the actual drift of features.

8 Complexity of the method

The proposed feature drift detector employs statistical calculations. Only feature ranking methods furnish the requisite data for these calculations. The statistics possess a constant complexity and thus can be excluded from the procedure for determining the complexity of the proposed detector. After each drift detection, the detector utilizes the LASSO feature ranking procedure, depending on the version. Therefore, the computational complexity of ranking is crucial.

8.1 Time complexity

Subsequent chunks are divided into L_N subchunks if a drift is identified. The LASSO feature ranking method is employed for each L_N subchunk. The ranking is calculated once over the entire chunk without drift being detected.

Let the number of drifts be denoted as L_d. Let the total number of chunks be denoted as L_c. Let the computational complexity of the LASSO procedure be denoted as $O(A)$. For such an assumption, computational complexity is as follows: $O(L_d \cdot L_N \cdot O(A)) + O((L_c - L_d) \cdot O(A))$. After simplification, the computational complexity of the algorithm is $O((L_N + L_c) \cdot O(A))$. The number of subchunks is constant, so the algorithm's complexity is approximately $L_c \cdot O(A)$. According to (3), we have N instances and d features. Consider LASSO implementation using LARS algorithm [32], then the computational complexity of LASSO-LARS is $O(d^3 + d^2 \cdot N)$. In our datasets, we assume that $d \leq N$. In consequence $d^3 \leq d^2 \cdot N$, hence computational complexity of LASSO is $O(A) = O(d^2 \cdot N)$.

8.2 Memory complexity

As before, the LARS version of the LASSO procedure will be considered with the same notations. LARS is an efficient algorithm particularly suited for Lasso when the number of features d is large. It should be noted that several algorithms can solve the memory LASSO optimization problem [32]. The memory complexity varies based on the algorithm's implementation.

The data matrix X is an $N \times d$ dimensional matrix representing the dataset. Matrix X requires $O(N \times d)$ space. Coefficients vector β requires $O(d)$ space. The total memory complexity of the LASSO procedure is $O((N \times d) + d) = O(N \times d)$. We also assume that each entry is a 4-byte float.

9 Conclusions

The paper presents a new approach for detecting feature drifts in source data based on ranking feature changes. The method has been validated through empirical evidence, which sets the groundwork for future research extension. The developed strategy is competitive with other methods, as shown in synthetic and real-world data comparative studies.

It is worth noting that feature drift detection is performed directly on the input (raw) data using the feature ranking of instances. Therefore, there is no fear that the classifier constantly assesses the data and drift detection is only based on its indications.

In most cases, the proposed feature drift detection method achieves better results than other detectors. The proposed approach also provides favorable data processing times compared to other detectors, which results from the adopted method of ranking the features of the original, not yet processed data. Future research will include testing alternative feature ranking methods where class labels are not required. These two approaches will be compared. The proposed method can improve the accuracy and efficiency of data stream processing in various domains by solving feature drift problems.

References

[1] Husheng Guo, Hai Li, Qiaoyan Ren, and Wenjian Wang. Concept drift type identification based on multi-sliding windows. Information Sciences, 585:1–23, 2022.

[2] Piotr Porwik and Rafal Doroz. Adaptation of the idea of concept drift to some behavioral biometrics: Preliminary studies. Engineering Applications of Artificial Intelligence, 99:104135, 2021.

[3] Thomas Bartz-Beielstein and Lukas Hans. Drift detection and handling. In Eva Bartz and Thomas Bartz-Beielstein, editors, Online Machine Learning: A Practical Guide with Examples in Python, pages 23–39, Singapore, 2024. Springer Nature Singapore.

[4] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. ACM Computing Surveys, 46:1 – 37, 2014.

[5] Supriya Agrahari and Anil Kumar Singh. Adaptive pca-based feature drift detection using statistical measure. Cluster Computing, 25(6):4481–4494, 2022.

[6] Paulo M. Gonçalves, Silas G.T. de Carvalho Santos, Roberto S.M. Barros, and Davi C.L. Vieira. A comparative study on concept drift detectors. Expert Systems with Applications, 41(18):8144–8156, 2014.

[7] Ruba Abu Khurma, Ibrahim Aljarah, Ahmad Sharieh, Mohamed Abd Elaziz, Robertas Damaševičius, and Tomas Krilavičius. A review of the modification strategies of the nature inspired algorithms for feature selection problem. Mathematics, 10(3):464, 2022.

[8] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. J. Mach. Learn. Res., 3:1157–1182, 2003.

[9] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. Data Mining and Knowledge Discovery, 30(4):964–994, 2016.

[10] Hang Yu, Qingyong Zhang, Tianyu Liu, Jie Lu, Yimin Wen, and Guangquan Zhang. Meta-add: A meta-learning based pre-trained model for concept drift active detection. Information Sciences, 608:996–1009, 2022.

[11] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research, 5:1205–1224, 2004.

[12] Jan Niklas Adams, Sebastiaan J. van Zelst, Thomas Rose, and Wil M.P. van der Aalst. Explainable concept drift in process mining. Information Systems, 114:102177, 2023.

[13] Hang Yu, Weixu Liu, Jie Lu, Yimin Wen, Xiangfeng Luo, and Guangquan Zhang. Detecting group concept drift from multiple data streams. Pattern Recognition, 134:109113, 2023.

[14] Supriya Agrahari and Anil Kumar Singh. Concept drift detection in data stream mining: A literature review. Journal of King Saud University - Computer and Information Sciences, 34(10, Part B):9523–9540, 2022.

[15] Mahmood Karimian and Hamid Beigy. Concept drift handling: A domain adaptation perspective. Expert Systems with Applications, 224:119946, 2023.

[16] Andrés L. Suárez-Cetrulo, David Quintana, and Alejandro Cervantes. A survey on machine learning for recurring concept drifting data streams. Expert Systems with Applications, 213:118934, 2023.

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

[18] Lin Sun, Tianxiang Wang, Weiping Ding, Jiucheng Xu, and Yaojin Lin. Feature selection using fisher score and multilabel neighborhood rough sets for multilabel classification. Information Sciences, 578:887–912, 2021.

[19] Frank S. Corotto. Chapter nine - the two-sample t test and the importance of pooled variance. In Frank S. Corotto, editor, Wise Use of Null Hypothesis Tests, pages 95–98. Academic Press, 2023.

[20] Piotr Porwik and Benjamin Mensah Dadzie. Detection of data drift in a two-dimensional stream using the Kolmogorov-Smirnov test. Procedia Computer Science, 207:168–175, 2022. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022.

[21] Toshiyuki Sueyoshi and Shingo Aoki. A use of a nonparametric statistic for dea frontier shift: the Kruskal and Wallis rank test. Omega, 29(1):1–18, 2001.

[22] Baoshuang Zhang, Yanying Li, and Zheng Chai. A novel random multi-subspace based relieff for feature selection. Knowledge-Based Systems, 252:109400, 2022.

[23] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In Proceedings of the 17th International Conference on Neural Information Processing Systems, NIPS'04, page 513–520, Cambridge, MA, USA, 2004. MIT Press.

[24] Xue-wen Chen and Jong Cheol Jeong. Enhanced recursive feature elimination. In Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pages 429–435, 2007.

[25] Nickolay Trendafilov and Michele Gallo. Pca and other dimensionality-reduction techniques. In Robert J Tierney, Fazal Rizvi, and Kadriye Ercikan, editors, International Encyclopedia of Education (Fourth Edition), pages 590–599. Elsevier, Oxford, fourth edition edition, 2023.

[26] Leo Breiman. Random forests. Mach. Learn., 45(1):5–32, 2001.

[27] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. The Annals of Statistics, 32(2):407–451, 2004.

[28] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. Bioinformatics, 23(19):2507–2517, 08 2007.

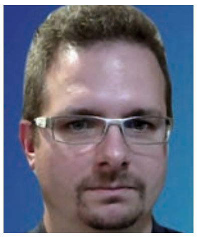
[29] Qiuming Zhu. On the performance of matthews correlation coefficient (mcc) for imbalanced dataset. Pattern Recognition Letters, 136:71–80, 2020.

[30] Davide Chicco, Niklas Tötsch, and Giuseppe Jurman. The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. BioData Mining, 14, 2021.

[31] Vinícius M. A. de Souza, Denis Moreira dos Reis, André Gustavo Maletzke, and Gustavo E. A. P. A. Batista. Challenges in benchmarking stream learning algorithms with real-world data. Data Mining and Knowledge Discovery, 34:1805–1858, 2020.

[32] Yaohui Zeng and Patrick Breheny. The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in r. The R Journal, 12, 01 2017.

Prof. **Piotr Porwik** works at the Institute of informatics, University of Silesia, Poland. His research interests relate to applied spectral Boolean function analysis, machine learning, classifiers, pattern classification and biometrics. Prof. P. Porwik is a member of the Editorial Board of the International Journal of Biometrics and International Journal of Applied Mathematics and Computer Science and Mathematics. He presented invited papers, tutorials and keynotes in Japan, India, Czech Republic, Greece, and Poland. He takes part in many national and international conferences where worked as a chairman or member of scientific and organizing committees. P. Porwik is also a reviewer in many prestigious international scientific journals.
https://orcid.org/0000-0001-8989-9478



Tomasz Orczyk has been a research and teaching worker at the Faculty of Science and Technology at the University of Silesia in Katowice since 2009. In 2018 he earned a Ph.D. in computer science. He specializes in machine learning methods and their applications in medicine and biometry. During his work at the university, he published over 40 research papers, and participated in numerous international scientific conferences. Aside from research work, he is teaching subjects like machine learning in biometry, computer networks, and digital electronics. In his spare time he maintains and repairs his collection of retro computers.
https://orcid.org/0000-0002-4664-8369

Krzysztof Wrobel received his M.Sc. and Ph.D. degrees in computer science from the University of Silesia in 1999 and 2006, respectively. Currently, he works in the Institute of Computer Science, University of Silesia. His research interest areas are biometrics, machine learning, digital image processing and computer graphics. Dr Wrobel is an author more than 80 scientific papers, published in international journals and conference materials. He also been a visiting professor at the University of Calgary in Canada.
https://orcid.org/0000-0001-7339-1100

Benjamin Mensah Dadzie is a Ph.D. candidate specializing in concept drift detection and adaptation in dynamic (evolving) data streams. His research aims to develop feature-based approaches for drift detection, integrating various classifiers and comparing with other drift detectors to improve predictive performance in evolving datasets using deep learning techniques, with applications across various domains. Benjamin Mensah Dadzie is currently working on advancing algorithms that improve predictive performance in evolving datasets, with a focus on feature-based drift detectors. His research aims to develop innovative methods for drift detection and adaptation, ensuring better model accuracy and performance in dynamic data streams.
https://orcid.org/0000-0002-1323-5891