Int. J. Appl. Math. Comput. Sci., 2025, Vol. 35, No. 4, 687–702
DOI: 10.61822/amcs-2025-0049

IMPROVING THE PERFORMANCE OF THE FEATURE DRIFT DETECTOR BY LASSO OBSERVATION OF SAMPLE FEATURE FLUCTUATIONS

Piotr PORWIK [a,*], Tomasz ORCZYK [a], Nathalie JAPKOWICZ [b]

[a]Institute of Computer Science
University of Silesia
ul. Będzińska 60, 41-00 Sosnowiec, Poland
e-mail: {piotr.porwik,tomasz.orczyk}@us.edu.pl

[b]Computer Science Department
American University
4400 Massachusetts Avenue, NW, Washington, DC, 20016, USA
e-mail: japkowic@american.edu

Feature drift is a subtype of data distribution drift that occurs when the statistical significance of input features changes over time, despite the overall decision boundary remaining stable. This phenomenon can cause a subtle degradation in model accuracy in streaming environments. In this paper, we propose a new model-aware method called *feature importance-driven drift detection* (FIDD). Rather than relying on classification error signals, FIDD tracks changes in feature importance rankings obtained from LASSO regression across neighbouring data fragments. As it observes the dynamics of feature importance instead of global label shifts, this method is particularly suited to detecting subtle shifts in data distribution. Experimental evaluation on both synthetic and real-world data (including different types of drift, such as abrupt, gradual, incremental and recurrent) shows that FIDD achieves higher accuracy consistently and produces significantly fewer false alarms than standard drift detectors (e.g., DDM, EDDM and ADWIN). Furthermore, FIDD is robust to labelling noise and computationally efficient, which makes it a practical and interpretable solution for adaptive learning in real-time applications.

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

1. Introduction

The occurrence of drift is a common phenomenon when the distribution of data changes over time, which leads to the degradation of the classification model's performance (Porwik and Doroz, 2021; Guo *et al.*, 2022; Japkowicz and Boukouvalas, 2024). These phenomena can occur in many domains, such as financial forecasting, computer vision, natural language processing, and climate change assessment. The underlying data can change due to consumer behavior, technology evolution, sensor degradation, or external events. Detecting different types of drift is essential to maintain the model's accuracy because ignoring it can lead to incorrect predictions (Bartz-Beielstein and Lukas, 2024; Japkowicz and Boukouvalas, 2024; Mielniczuk and Wawrzeńczyk, 2025). This is because drifts can change the data's statistical properties over time, leading to the degradation of the model's performance. There is a rich and constantly updated part of literature where authors have been introducing various drift detection methods for many years, such as statistical tests, procedures based on classifier ensembles, and distance-based measurements to detect and respond to drift (Gonçalves *et al.*, 2014; Gama *et al.*, 2014; Agrahari and Singh, 2022).

This paper proposes a new drift detection method based on statistical tests and feature rank in the data stream. Feature selection and feature relevance analysis are applicable techniques for data preprocessing and analysis in machine learning and data science. Additionally, the paper tries to explain the problem of feature drift to equip researchers and practitioners with the knowledge and tools to effectively detect drifts in

[*]Corresponding author

machine learning models. Feature drift occurs when the rank (importance) of the features' subset changes in the pair of adjacent timestamps.

Let a given data piece consist of a set of samples, where $(\mathbf{x}, y)$ represents each sample in the data. We assume that $\mathbf{x}$ is a feature vector and y is a class label. If concept drift occurs, the distribution of $p(\mathbf{x}, y)$ between the current and next data chunks changes. This means that there are some places where $\exists t, p_t(\mathbf{x}, y) \neq p_{t+1}(\mathbf{x}, y)$. It can be seen, based on Bayesian theory, that the joint probability $p_t(\mathbf{x}, y)$ of events $\mathbf{x}$ and y can be written in the form $p_t(\mathbf{x}, y) = p_t(\mathbf{x}) \cdot p_t(y|\mathbf{x})$. This allows us to consider two types of concept drift (Gama *et al.*, 2014; de Souza *et al.*, 2020): *real drift*, when $p_t(y|\mathbf{x}) \neq p_{t+1}(y|\mathbf{x})$, where $p_t(\mathbf{x}) = p_{t+1}(\mathbf{x})$ is the change in conditional probability, and *virtual drift*, when $p_t(y|\mathbf{x}) = p_{t+1}(y|\mathbf{x})$ and $p_t(\mathbf{x}) \neq p_{t+1}(\mathbf{x})$.

Feature drift is a specific type of virtual drift, as it involves changes in the distribution or importance of input features $\mathbf{x}$, while the conditional relationship $p_t(y|\mathbf{x})$ remains unchanged. In feature drift, although the labels remain consistent for given instances, the statistical properties or relevance of features evolve, signaling a structural change in the input space rather than in the decision function (Žliobaitė, 2010). Feature-level virtual drift may lead to shifts in feature relevance or redundancy, degrading model interpretability, increasing computational cost, and potentially preceding real drift. Ignoring such changes may cause models to rely on outdated or irrelevant input representations.

This paper focuses on the importance (ranking) of input features that change over time, even though the underlying decision function remains stable. To address this challenge, we propose a novel method called FIDD. The approach uses LASSO regression to estimate feature importances in consecutive data chunks and identifies significant rank-based deviations as signals of feature-level drift. Unlike traditional detectors that rely on classification errors, FIDD is classifier-agnostic and focuses on shifts in the internal structure of the data stream.

Although FIDD detects virtual drift, it triggers retraining of the classifier when changes are detected. When the importance of these features changes, even without altering the decision boundary, the original model may no longer be representative of the current data structure. Therefore, updating the model ensures continuous performance and interpretability. Virtual drift can also signal impending real drift, where retraining is necessary.

In the taxonomy of drift, the following types are distinguished:

- *Abrupt (sudden) drift.* In this type of drift, the context change is immediate. The new concept is introduced abruptly, and since the concepts switch over, there is no more data from the previous concept in the data stream. The switching point is clear, but there are no signs of it happening.
- *Gradual drift.* This type of drift is more subtle. Data from the new concept starts to appear in the stream at an increasing frequency over time. The concept switchover lasts for some time, during which data from both concepts interleave in the data stream. The switching point is stretched over time, but subtle changes in data may indicate it is coming.
- *Incremental drift.* This type of drift is similar gradual one, but the switchover takes infinite time. The data stream contains data from both concepts; only their proportions change over time. In finite time, there is no clear concept switchover point.
- *Recurring drift.* This is a special case of abrupt and gradual drift in which switching between concepts occurs multiple times over time. There is a special case of recurring drift, where the same data concept may reappear in the future so that the old classification model can be reused. This also differs from data seasonality, as the concept changes are irregular and unpredictable.

The remainder of the paper is organized as follows. Section 2 discusses feature ranking and the role of LASSO in streaming settings. Section 3 presents the FIDD method. Section 4 describes the datasets used, and Section 5 contains the experimental setup and evaluation. Section 6 analyzes runtime performance, Section 7 outlines algorithmic complexity, and Section 8 concludes the study.

Motivation. The increasing prevalence of non-stationary data streams across application domains, such as finance, medicine, or environmental monitoring, presents major challenges for maintaining classification model accuracy over time. While concept drift detection has been widely studied, a less explored but equally important phenomenon is feature drift-a specific form of virtual drift where the importance of input features changes even if the decision boundary remains stable. Feature drift can silently degrade model interpretability, increase computational cost, and act as an early indicator of real concept drift. Traditional classifiers are often unable to detect such shifts, especially when they rely on static feature selection. This motivates the development of methods capable of tracking changes in feature relevance over time, enabling proactive adaptation to evolving data streams. Our work addresses this gap by proposing a flexible, interpretable approach that responds to fluctuations in feature importance, offering both early drift detection and structural insight into the evolving dataset.

Proposed contribution. This paper presents a comprehensive and repeatable study to evaluate feature drift on a classifier or a drift detector. We introduce the FIDD method, a novel approach to supervised feature drift detection. FIDD identifies feature drift by comparing the rankings of feature relevance across consecutive data chunks. The detector is adaptable, making it versatile for various applications. Its key advantages include the following:

- simultaneous feature ranking and selection in a single pass,
- controlling the number of significant features with a single regularization parameter,
- a comprehensive solution for drift detection, increasing the robustness of classification against false positives.

Our approach belongs to the family of model-aware drift detection methods, as it involves training a predictive model and monitoring the model features over time (Gama *et al*., 2014; Baena-Garcia *et al*., 2006; Duda *et al*., 2020). Our strategy requires training an auxiliary LASSO-based model from which diagnostic knowledge (feature ranking) is extracted. This means that drift is detected not directly in the data, but through changes in the model behavior. Regression-based feature ranking does not require monitoring of classifier errors—it relies more on relative changes between features. While the DDM and EDDM methods rely on classifier error metrics, our technique tracks the relative importance of features over time (Japkowicz and Boukouvalas, 2024). This makes it less sensitive to label noise, especially in multi-class scenarios with redundant or correlated features. Although our method technically falls into the model-aware category, since it tracks changes in an auxiliary predictive model, it functionally resembles feature-based approaches, similar to methods based on PCA or on data distributions such as ADWIN.

2. Features ranking

Feature ranking involves assigning a value (importance measure) to each feature to determine its relative importance. Ranking does not eliminate features but prioritizes them, allowing a better understanding of their impact. Ranking can be performed using different strategies, such as calculating correlation coefficients, mutual information, or weights in linear models. The essential criterion for selecting the ranking procedure algorithm must be its time complexity due to the requirement to analyze the data stream. In general, feature ranking can be done using filtering, wrapping, or embedding methods (Saeys *et al*., 2007; Dhal and Alad, 2022; Usman *et al*., 2023).

Filter-based methods work fast, but unfortunately, filtering methods (e.g., Pearson correlation) do not recognize mutually correlated features (Usman *et al*., 2023). They may retain multiple features conveying similar information, which increases dimensionality without real benefit. Feature selection is solely based on statistical measures, which may lead to discarding features with a subtle but essential value to the model.

Wrapper-based methods are slow (Dhal and Alad, 2022). They consider the evaluated model and assess its performance on different subsets of features. This makes such methods impractical even for average data sizes due to computational time.

Some embedded methods are also fast, e.g., LASSO regularization models (Efron *et al*., 2004). Unlike the ones mentioned earlier, the LASSO strategy also considers the interactions between features, which is an advantage of this approach.

Our strategy assumes that features are ranked according to importance, but none are removed. Unimportant features in the current step may suddenly become essential in the next one due to the non-stationary nature of the input data. In this case, the drift detector focuses on observing a new feature that has become significant in the next chunk of data.

2.1. Feature importance: LASSO. To determine the importance of all features, we use the LASSO algorithm with the L1 norm. While, to our knowledge, the LASSO strategy has not been previously applied in feature drift detection procedures, we leverage its unique properties. LASSO is well-suited for high-dimensional settings where feature redundancy and noise may hinder drift detection. It enforces sparsity by introducing an L1 regularization penalty, effectively selecting a subset of the most relevant features while eliminating irrelevant or less informative ones. This property is crucial in drift detection, as changes in data distribution often manifest themselves through a subset of features rather than uniformly across all dimensions. Additionally, LASSO aids in mitigating overfitting, improving model generalizability when detecting shifts in feature importance over time. Its capability to handle collinearity among features further enhances its suitability for identifying structural changes in evolving data streams.

We will 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 concept 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 (informativeness) of features in adjacent data chunks. Each chunk's best feature (highest rank) is established using the LASSO procedure. The algorithm is emploed for regression analysis in statistics and machine learning, particularly for feature selection (Guyon and Elisseeff, 2003; Efron *et al.*, 2004; Yamada *et al.*, 2014). The LASSO algorithm has three variants: ElasticNet ($\lambda_1 \neq 0, \lambda_2 \neq 0$), Ridge ($\lambda_1 = 0$), and Lasso ($\lambda_2 = 0$).

Consider a sample with N instances (observations) and $\mathbf{x}_i \in \mathbb{R}^d$. The LASSO ElasticNet linear regression aims to minimize the following cost function:

$$\text{Cost Function} = \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{d}\beta_j x_{ij}\right)^2 + \lambda_1\sum_{j=1}^{d}|\beta_j| + \lambda_2\sum_{j=1}^{d}\beta_j^2 \tag{1}$$

where $\beta = [\beta_1, \ldots, \beta_d]$ is the coefficient (weight) vector, λ_1, λ_2 are predetermined free factor regularization parameters controlling the strength of the L1 and L2 penalties, vand variable y_i is the actual target value for observation i. The parameter β_0 in the LASSO regression equation (1) represents the intercept term of the linear model, the coefficients β_i denote the impact of particular features of the linear numerical model on the data representation, vand element x_{ij} is a value of the j-th feature in the i-th observation. This means relevant features are inside the set $\{1 \leq i \leq d : \beta_i \neq 0\}$.

Unfortunately, ElasticNet (1) introduces the complexity of two regularization parameters λ_1, λ_2, which is not beneficial from the point of view of the speed at which data is tracked in the stream. The Ridge version ($\lambda_1 = 0$) is not very interesting, either, because all the weights β_i are different from zero. Weights with small values are not very relevant in practice, but they need to be analyzed. Ultimately, in our study, we use Lasso, which employs L1 norm regularization. Ridge regression (L2 norm) and ElasticNet (combination of L1 and L2) were discarded for the reasons we gave above. The obtained feature rankings are compared at time steps to detect changes. From the point of view of the needs of the proposed feature drift detector, the Lasso version ($\lambda_2 = 0$) is the most useful, as it causes zeroing of all irrelevant feature weights, so they do not need to be analyzed. This significantly speeds up the feature drift detection process.

Determination of the values of the minimizing weights Lasso criterion is not possible by analytical means and requires an iterative algorithm. In the optimization process, weights of the β_i coefficients are established for features x_i, so we obtain the desired order of importance of features after arranging the coefficients β_i in descending order. In Lasso with L1 regularization ($\lambda_1 > 0, \lambda_2 = 0$), correlated features are grouped and reduced to a single feature, which is impossible in other selection and ranking methods. For example, for source data with samples with eight features, after the Lasso procedure, the features can be arranged as follows: $x_1 > x_2 > x_7 > x_4 > x_3 = x_5 = x_6 = x_8 = 0$. In our approach, we set a maximum allowable fluctuation range for the top-ranked feature x_i in the reference part. This range specifies the degree of change in the position of the feature x_i.

For simplicity, instead of λ_1, we will denote this parameter as λ in regard to Lasso. The regularization parameter λ controls the number of selected features, offering flexibility in drift detection. For Lasso with the L1 regularization parameter, feature x_j importance $FI(x_j)$ will be calculated from the formula

$$FI(x_j) = \frac{|\beta_j|}{\sum_{i=1}^{d}|\beta_i|}e^{-\lambda}, \qquad j = 1, \ldots, d, \tag{2}$$

where d is the number of features and λ is a regularization parameter that controls the strength of feature elimination.

The optimal value of λ is the value of the Lasso regularization parameter that gives the best prediction results of a given classifier after feature selection in the Lasso procedure. Small values of the λ parameter (weak regularization) lead to many features and good accuracy. Too large values of λ result in eliminating too many features, which reduces accuracy. The optimization of the λ parameter, considering the prediction quality of the RF classifier, was performed in the simulation procedure. The simulation data were selected to approximately match the characteristics of the benchmark data used in the article. Although this example is static, the same Lasso-based feature ranking procedure is later applied to each data chunk in a stream to track changes in feature importance over time.

The simulation was performed on artificial data with parameters similar to real streaming data, on which detailed studies were later used: samples = 100 K, classes = 10, features = 50, informative features = 20, redundant features = 5.

The synthetic data was generated using the make_classification() function from the scikit-learn library, which allowed controling the number of samples, classes, informative and redundant features, as well as simulating the structure of the real dataset. These data were used to analyze the behavior of Lasso feature importance in a controlled, stationary setting before applying the method to streaming benchmark datasets.

In the next step, 5-fold cross-validation was implemented. For each value of the regularization parameter λ, we performed full cross-validation, and the Lasso model selected features for which $\beta_i \neq 0$. The RF

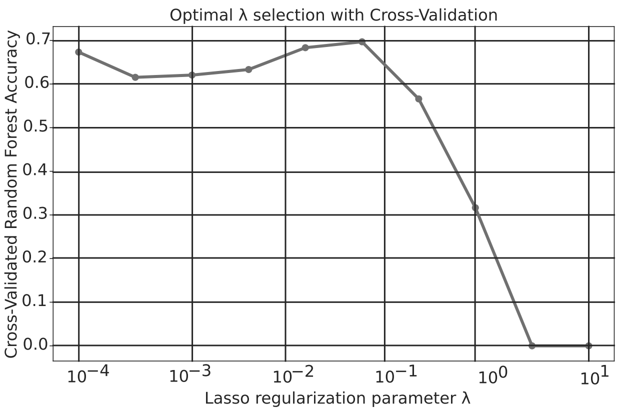

Fig. 1. Classifier prediction depends on λ parameter selection in the Lasso procedure: the best value $\lambda = 0.05$ for the highest classifier accuracy. The OX axis is in log scale.

classifier was trained on these features. After completing the cross-validation, we calculated the average accuracy for each value of λ. The simulation results are depicted in Fig. 1.

As a result of the experiment, it was observed that, for small values of λ, Lasso preserves most of the features, which translates into stable, high-accuracy for the random forest (RF) classifier, while for large λ, there is a strong reduction in dimension and a decrease in efficiency. The optimal compromise was found in the range $\lambda \approx 10^{-2} \ldots 10^{-1}$. In the experiments, we assume that $\lambda = 0.05$.

In our experiments, the RF classifier was selected as the predictive model due to its robustness to noisy and redundant features and its strong performance in multi-class classification tasks. Additionally, it is widely adopted as a reference classifier in the concept drift literature, providing a stable and interpretable standard baseline (Breiman, 2001; Gama *et al.*, 2014; Bifet *et al.*, 2010). The RF classifier was configured with 100 trees, max depth = 10, and a default Gini criterion (Bifet and Gavaldà,, 2009; Fernández-Delgado *et al.*, 2014).

Although the following analysis is based on a single, synthetic dataset for illustrative purposes, the same LASSO-based ranking procedure is later applied independently to each data chunk in a streaming scenario to monitor the evolution of feature importance over time. It should be noted that a similar solution, based on feature drift analysis, was described by Zhao and Koh (2020). This solution is fundamentally different from the proposed FIDD strategy, both in terms of the definition of feature drift and the drift detection mechanism. Zhao and Koh (2020) focus on detecting changes in the distributions of input features observed in data blocks, measured using the Wasserstein or energy statistical distance. Their approach is completely unsupervised. FIDD does not analyze the input data itself, but estimates the relationship between features and labels based on the regression coefficients β_i. This makes FIDD a supervised method.

3. Proposed method

This section introduces the proposed method, *feature importance driven drift* (FIDD), which uses Lasso-based feature importance variation to detect drift in streaming data. FIDD is a streaming drift detection algorithm based on tracking changes in feature importance rankings obtained from Lasso regression applied to consecutive data chunks.

The most important part of the feature drift detection method is identifying the most relevant features of the data stream using the Lasso procedure. The data are observed in shifting, non-overlapping windows, within which Lasso operates. The proposed drift detector analyses all features for which the weights $\beta_i \neq 0$ and ignores all other weights where $\beta_i = 0$. Further on, a self-describing Algorithm 1 (see Fig. 2) for detecting feature drift and refitting the base classifier is presented.

In the first step, the features are classified using the average ranks obtained on the subdivisions of the first fragment using the Lasso procedure. Next, the features are sorted from the most to the least important based on their mean ranks. The process is repeated for each new data chunk: the chunk is analyzed, and a new feature ranking is determined. The stability of feature importance is assessed by comparing the latest ranking with the reference ranking.

If a feature's position deviates beyond the range $[\mu - \sigma, \mu + \sigma]$ in a given chunk, feature drift is detected, and this chunk becomes the new reference one. The process continues until all relevant features established by Lasso are exhausted. It is important to clarify that, in FIDD, the threshold σ_i is not a user-defined hyperparameter but a dynamically computed standard deviation of the importance of feature i over a recent data window. This makes the algorithm self-adaptive and statistically grounded, similar to DDM (Baena-Garcia *et al.*, 2006), where thresholds are derived from empirical estimates of classifier error and variance. Therefore, traditional hyperparameter optimization is not applicable in the same sense, as FIDD operates without requiring externally tuned thresholds.

The algorithm terminates when no further chunks remain for analysis. The mentioned interval covers about 68% of the normally distributed feature values. Exceeding this range indicates that the new feature value is unusualy relative to previous observations and may signal a change in the data distribution or feature drift. This is a simple but effective method for detecting changes in distribution, which makes the algorithm work well in data stream

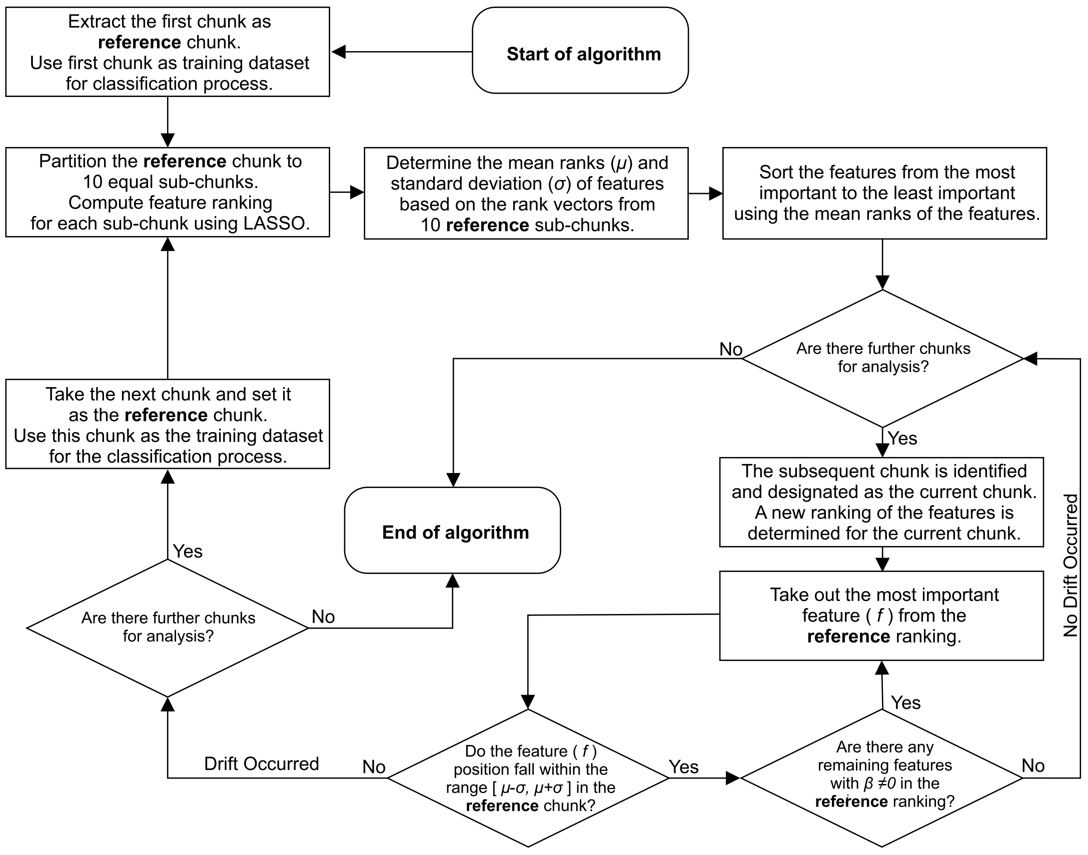
Start of algorithm
Extract the first chunk as reference chunk. Use first chunk as training dataset for classification process.
Partition the reference chunk to 10 equal sub-chunks. Compute feature ranking for each sub-chunk using LASSO.
Determine the mean ranks (μ) and standard deviation (σ) of features based on the rank vectors from 10 reference sub-chunks.
Sort the features from the most important to the least important using the mean ranks of the features.
Are there further chunks for analysis?
No
Yes
The subsequent chunk is identified and designated as the current chunk. A new ranking of the features is determined for the current chunk.
Take out the most important feature (f) from the reference ranking.
Do the feature (f) position fall within the range [μ-σ, μ+σ] in the reference chunk?
Yes
No
Drift Occurred
Are there any remaining features with β ≠0 in the reference ranking?
Yes
No
No Drift Occurred
Are there further chunks for analysis?
Yes
No
Take the next chunk and set it as the reference chunk. Use this chunk as the training dataset for the classification process.
End of algorithm

Fig. 2. Workflow diagram of Algorithm 1 for feature importance-driven drift detection (FIDD). The Python source of the algorithm is available in the GitHub repository at `https://github.com/ZSKPP/feature-drift`.

analysis or environments where variability is natural (e.g., user behavior, sensory data). The advantages of the proposed approach can be expressed as follows:

1. The FIDD algorithm automatically updates the feature ranking when drift is detected. This ensures the classifier always operates on the most up-to-date features, increasing efficiency.

2. Classic drift detection methods, such as DDM or EDDM, detect drift globally based on classification errors. Our approach evaluates changes in the importance of individual features, which allows more precise adjustments to the model.

3. Many drift detection methods require storing and analyzing large sets of historical data, such as sliding window methods. The FIDD algorithm relies on the current data fragment and a simple standard deviation criterion, remarkably reducing the computational cost.

4. Comparing feature positions in rankings rather than raw values avoids false alarms caused by local fluctuations in the data. This approach is more robust to short-term anomalies and random changes, unlike methods based on statistical significance tests that can detect apparent drifts.

5. The FIDD strategy improves the interpretability of drift by identifying which features change in importance over time, helping explain why and where the model's behavior changes.

4. Data preparation

Tests on the effectiveness of the proposed feature drift detector were carried out on artificial collections, where the drift location can be programmed, and on real-world data taken from repositories, where places of drift occurrence are known or unknown. Both two-class and multi-class data are considered in this study (Table 1). This will assess the universality of the method. Some benchmarks used in the experiments have no marked drift moments. Although there are no indicated concept change points in the file, there are changes in the distribution of features, and the relationships between variables change over time. This is a challenge for

Table 1. Characteristic of real datasets.

Dataset	#Inst	#A	#C	Drift points
Electricity (Frank, 2010)	45,312	8	2	unknown
Ozone (Frank, 2010)	2,534	72	2	unknown
Arrythmia (Frank, 2010)	459	279	16	unknown
Poker-hand (Frank, 2010)	1,025,010	10	10	Modified: drift from 50000 to 99 999 with class k permutation $k->(k+1)mod10$
Rialto bridge (Frank, 2010)	82,250	27	10	unknown
Outdoor (Frank, 2010)	40,000	21	40	unknown
INSECTS (de Souza *et al.*, 2020)				
Abrupt balanced	52,848	33	6	14,352; 19,500; 33,240; 38,682; 39,510
Abrupt imbalanced	355,275	33	6	83,859; 128,651; 182,320; 242,883; 268,380
Gradual balanced	24,150	33	6	14,028
Gradual imbalanced	143,323	33	6	58,159
Incremental balanced	79,986	33	6	26,568; 53,364
Incremental imbalanced	452,044	33	6	150,683; 301,365
Reoccurring balanced	79,986	33	6	26,568; 53,364
Reoccurring imbalanced	452,044	33	6	150,683; 301,365

#Inst: number of instances, #A: number of attributes, #C: number of classes.

drift detection algorithms because the changes are not known in advance, and the algorithm should detect them (Stefanowski *et al.*, 2017).

4.1. Synthetic data. The first part of synthetic data was prepared using the command line interface in the MOA environment, a popular JAVA open-source machine learning framework for data streams. Synthetic data were produced by a hyperplane generator (HPG) (Bifet *et al.*, 2010). It generates a data stream where the number of instances, the number of features in an instance, the number of drifting features, the place of drift occurrence, and the time of drift can be programmed. The hyperplane H is a set of multidimensional points such that $H = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} = b\}$, where $\mathbf{w} = [w_1, \dots, w_a]$, $w_i \in R$, $\mathbf{x} = [x_1, \dots, x_a]$ $x_i \in R$ and $b \in R$. We assume that $\sum_{i=1}^{d} w_i = b$. Instances are randomly generated, and each 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 those with condition $\mathbf{w}^T\mathbf{x} < b$ belong to the second one. Classes are balanced. Drift is randomly recorded on 10/15 features with 1%, 3%, or 5% label noise. The noise was introduced by inverting the class labels with probability $n\%$. Each synthetic dataset consists of 100 K instances.

4.2. Real data. The study examined several repositories (Frank, 2010; de Souza *et al.*, 2020), shown in Table 1.

The selected datasets represent a diverse set of stream-like scenarios, encompassing both real-world and synthetic sources, as well as various types of concept drift. The Electricity dataset, although not annotated with explicit drift points, contains well-documented market-driven changes, including both abrupt and gradual real drifts, making it a standard testbed for adaptive algorithms. The Ozone and Arrhythmia datasets exhibit implicit and subtle drift phenomena, mostly of virtual or gradual nature, due to environmental fluctuations and patient variability, respectively. Although the Ozone and Arrhythmia datasets are not among the most commonly used concept drift benchmarks, such as KDDCup99 (Tavallaee *et al.*, 2009) or Forest Cover Type (Blackard, 1998), they were deliberately selected to reflect real-world data complexity and irregular drift behavior.

Ozone, derived from atmospheric monitoring, exhibits natural fluctuations in class distributions over time, while Arrhythmia captures high-dimensional, clinical decision scenarios with underlying gradual or latent drift. These settings are representative of practical applications where drift is neither synthetic nor annotated, and the drift detection task becomes inherently more challenging.

In contrast, Poker-hand is a static synthetic dataset with no inherent drift, but was artificially permuted by class permutation. The Rialto Bridge and Outdoor datasets are sensor-based streams with naturally occurring gradual drifts, such as seasonal environmental variations; these drifts are real but unlabeled, thus implicit. Finally, INSECTS is a hybrid benchmark in which explicitly defined drifts are introduced in both features and class distributions, supporting the evaluation of drift detectors under controlled, repeatable conditions. This collection of datasets enables a comprehensive and realistic assessment of drift detection performance across a range of drift types.

5. Experiments and evaluation of the proposed method

The objective of the experimental study is to answer the following research questions:

- *RQ1*: Does the proposed FIDD method effectively detect various types of concept drift (abrupt, gradual, incremental, recurring) in data streams?
- *RQ2*: How does the FIDD detector compare to established drift detection methods (DDM, EDDM, ADWIN, PCA-FDD) in terms of accuracy, false positives, and true positives?
- *RQ3*: Is FIDD robust to label noise and applicable in both binary and multi-class scenarios?
- *RQ4*: What is the computational cost of the FIDD method?

To address these questions, a series of comparative experiments was conducted using both synthetic and real-world benchmark datasets. The evaluation involves accuracy-based metrics, false positive (FP) and true positive (TP) rates, MCC correlation, and runtime analysis.

All computational experiments were conducted in Python 3.8 with the scikit-learn stream-learn, scikit-multiflow, NumPy and Frouros 0.6.1 packages. In the experiments, three drift detection methods will be applied from skmultiflow, a machine learning package for streaming data in Python (Montiel *et al.*, 2018).

Table 2 lists the hyperparameters and configuration settings used for each of the compared drift detection algorithms in the experimental evaluation. The warning and drift threshold parameters are α and β. To ensure fairness and reproducibility, the parameters from Table 2 were used in configuring each drift detection method and were implemented in Python packages. The PCA-FDD method implements the algorithm by Agrahari and Singh (2022). The parameter values for drift detection algorithms were selected based on the original publications and default settings from widely used implementations in the Python scikit-multiflow and Frouros libraries. These settings enable a fair comparison and preserve reproducibility. The ADWIN detector parameter was chosen based on the settings proposed by Bifet and Gavaldà, (2009), whereas parameters of DDM and EDDM were chosen based on the selection made by Baena-Garcia *et al.* (2006).

Table 2. Drift detector parameters description.

Detector	Parameters
ADWIN	$\delta = 0.002$
DDM	$\alpha = 0.01$, p-value $= 0.001$
EDDM	$\alpha = 0.9, \beta = 0.95$
PCA-FDD	$\delta = 0.5$, no. of components $= 2$

5.1. Evaluation metrics. In some datasets, the data stream is characterized by known drift points applied to all drift detectors. For such assumptions, the evaluation process of drift detectors can be organized similarly to classification tasks, which means that, for a given detector, true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) can be measured (Stapor *et al.*, 2021). Accuracy is $ACC = (TP + TN)/(TP + TN + FP + FN)$. The problem with it is that it is not an informative metric when the two classes are of significantly different sizes. The Matthews correlation coefficient (MCC) metric, although not necessarily the best choice in all cases (see, e.g., Zhu, 2020), often yields a more helpful performance score than ACC (Chicco *et al.*, 2021). For the binary classification and multiclass (which is automatically recognized by scikit-learn implementation of this metric), we have the formula

$$\text{MCC} = \frac{c \times N - \sum_{i=1}^{C} p_i \times t_i}{\sqrt{(N^2 - \sum_{i=1}^{C} p_i^2) \times (N^2 - \sum_{i=1}^{C} t_i^2)}}, \quad (3)$$

where C is the number of classes, N is the number of samples, c is the number of samples correctly predicted, t_i is the number of times class i truly occurred, and p_i is the number of times class i was predicted. This measure can be used even if the classes are of very different sizes. The Matthews correlation coefficient can be considered a good candidate for a standard metric role for assessing classifications, especially when datasets are imbalanced. It generates a high score only if most of the data elements were predicted correctly in the data classes. From the definition of the MCC measure, it follows that, for MCC $= +1$, we have a perfect prediction, so the classifier well predicts across all classes. For MCC $= 0$, predictions are no better than random guessing. If MCC $= -1$, it means that forecasts are completely incorrect. This is the worst possible solution.

Although the formula (3) for a multiclass MCC is more complex than the binary version, it retains the same strengths, making it a valuable metric for evaluating multiclass classifiers, especially in situations where precision, recall, or accuracy might give a skewed view of performance due to class imbalance or other issues. If the data are divided into chunks, the average values of ACC and the MCC are computed.

5.2. Evaluation based on synthetic data. We connect Algorithm 1 with the random forest (RF) classifier, both with and without a signal, to trigger retraining after drift detection. The experiments (Figs. 3–6) illustrate two cases: (a) natural accuracy decline after drift and (b) classifier response upon receiving a drift signal. Various drift types (sudden, gradual, recurring, incremental) were tested. The top graphs show the accuracy of a classifier trained only once, highlighting post-drift performance drops. The bottom graphs depict retraining upon drift detection, demonstrating timely drift identification and improved accuracy.

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 is presented in the middle chart of Fig. 4. This is, however, an advantageous phenomenon. In such a case, the classifier can be re-trained using the transition patterns between the two concepts, potentially affecting classification accuracy.

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 concept drift detection.

Algorithm 1 detects drift directly in the source data and does not use the results of an intermediate classifier that needs to be selected in advance. Our approach eliminates the errors that such a classifier induces. Tests were performed on synthetic datasets. The characteristics of these datasets were presented earlier in Section 3.1. The proposed FIDD detector was compared with the state-of-the-art methods, and the results were averaged over 10 runs with different random seeds.

In the study, three levels of label noise (1%, 3%, 5%) were used for each of the four drift types and two numbers of drifting features (10 and 15). The study was repeated for 10 data sets generated with different, randomly generated seeds. This gives 60 data sets for each drift type. This is a compromise between the complexity of the experiment and its readability. The results are summarized in Table 3. The proposed FIDD detector always achieved the highest ACC index, which points out its effectiveness.

The quality of detection is understood as the number of detections of an actual drift (TP) or a falsely indicated one (FP), as presented in Table 4. The row labeled ‘Oracle result’ reflects the expected detection result based on the concept drift injected in the synthetic stream: the number of true positives and false positives. It serves as an ideal basis for interpreting the results of real detectors. The results were averaged over 10 runs of data from Table 3 in each category with different random seeds. The experiments show that the FIDD detector generates significantly fewer false alarms (FP) compared to other methods. This phenomenon is an advantage of the proposed solution.

The result was also confirmed using the Wilcoxon test, as shown in Table 5, on the α level equal to 0.05. The Wilcoxon test does not assume the normality of the data but requires paired data. The conditions were met because each pair of results is from the same experimental condition (noise level + drift type).

The Wilcoxon signed-rank test was applied in its two-tailed form. The null hypothesis assumes no systematic difference in paired results (median difference $= 0$). 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 ACC for detectors differs. At the 5% significance level, the null hypothesis of equal efficiency is rejected ($p \approx 0.03$). However, due to the proximity of the p-value to the threshold, the result should be considered statistically significant but treated with caution in practice. Thus, we acknowledge that, in the future, a more comprehensive evaluation of the cost-effectiveness and stability of our method is required, especially under real-world deployment constraints. The proposed FIDD method shows a statistically significant difference from all other strategies.

To evaluate the robustness of our method, we conducted experiments on 10 synthetic data streams. Each stream was generated with a different initial random seed to ensure variability. Exactly one drift event (denoted as $P = 1$) was embedded in each stream. For each drift detection method, the average number of true positives (avgTP) and false positives (avgFP) was computed as the mean across the ten independent data streams. Based on Table 4, the next coefficients were computed: True Positive Rate (sensitivity) TPR = avgTP/P and Precision Prec = avgTP/(avgTP + avgFP). These two metrics directly quantify the sensitivity of the method and its resistance to false alarms. High precision confirms that the detector rarely responds to short-term fluctuations with false alarms, while a high TPR shows that true drifts are consistently detected. Together, they provide strong empirical evidence supporting the validity of the proposed ranking-based approach. It is presented in Fig. 7, a radar plot comparing the performance of five concept drift detectors (FIDD, DDM, EDDM, ADWIN, and PCA-FDD) across evaluation metrics: True Positive Rate (TPR) and Precision (Prec), for four types of drift (abrupt, gradual, incremental, and recurring). The values were averaged over 10 synthetic data streams with a single programmed drift per stream. The FIDD detector consistently demonstrates a balanced and superior performance across most drift types. Notably, it achieves high precision in all scenarios, particularly under abrupt (Prec $= 0.85$) and recurring drift (Prec $= 1.00$), while also maintaining high detection rates (e.g., TPR $= 1.00$ for abrupt drift and TPR $= 3.00$ for recurring

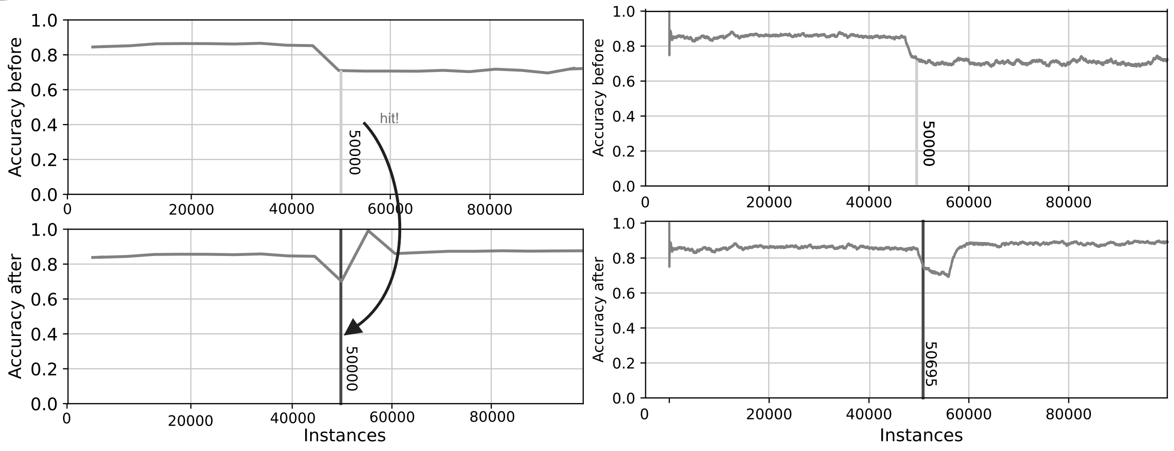

Fig. 3. Abrupt drift: drift detection by the FIDD chunk-based detector (left) and the ADWIN instance-based detector (right) for the RF classifier.

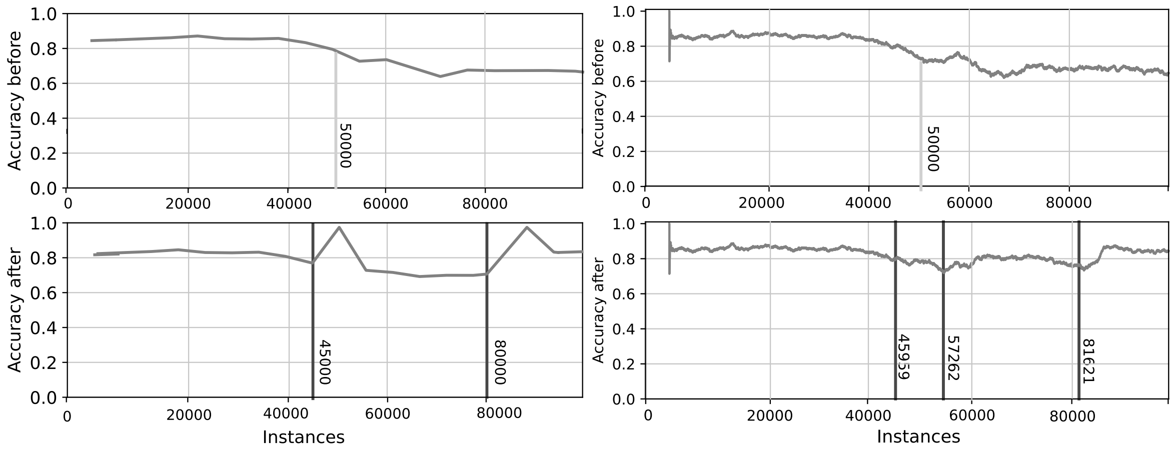

Fig. 4. Gradual drift: drift detection by the FIDD chunk-based detector (left) and the ADWIN instance-based detector (right) for the RF classifier.

drift). These results indicate that FIDD is not only effective in identifying true drifts but also resistant to false positives. In contrast, the DDM and PCA-FDD detectors show limited effectiveness, with a low TPR and nearly zero precision across most drift scenarios, indicating poor sensitivity and high false alarm rates. EDDM demonstrates a strong TPR in all drift types (1.00 for both abrupt and incremental drifts), but its low precision values (as low as 0.05) suggest a tendency to overreact to short-term fluctuations, resulting in frequent false positives. ADWIN exhibits intermediate performance, detecting recurring drifts relatively well (TPR $= 2.17$), but with moderate precision (Prec $= 0.20$). For other drift types, its performance remains limited. The radar plot illustrates that FIDD achieves the most favorable trade-off between sensitivity (TPR) and specificity (Prec), thus empirically supporting the claim that monitoring feature importance rankings offers robustness against transient fluctuations and false alarms, which is an advantage over other detectors.

5.3. Evaluation based on real data. Studies similar, as in the previous section were performed for the case of real data (de Souza *et al.*, 2020). The characteristics of these data are given in Table 1. 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 that for synthetic data for the identical drift detectors. Now, instead of ACC, the MCC is applied. The training data was constructed from the initial 5% of the specified dataset. Results in Table 6 show the advantage of the new drift detection method over other reference methods.

Observation of the results from Table 6 suggests that the proposed feature drift detection method in conjunction with the RF classifier is better than other drift detectors

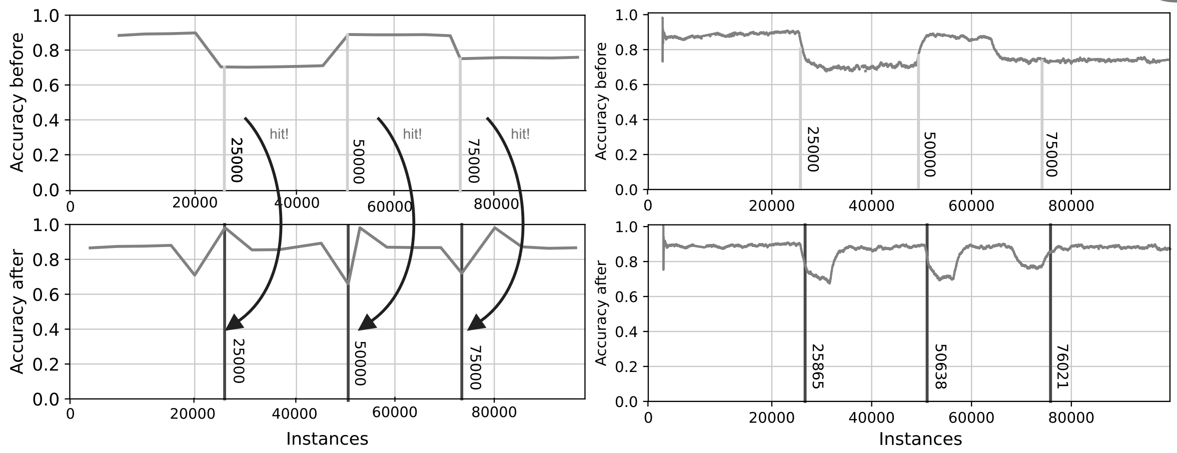

Fig. 5. Recurring drift: drift detection by FIDD chunk-based detector (left) and the ADWIN instance-based detector (right) for the RF classifier.

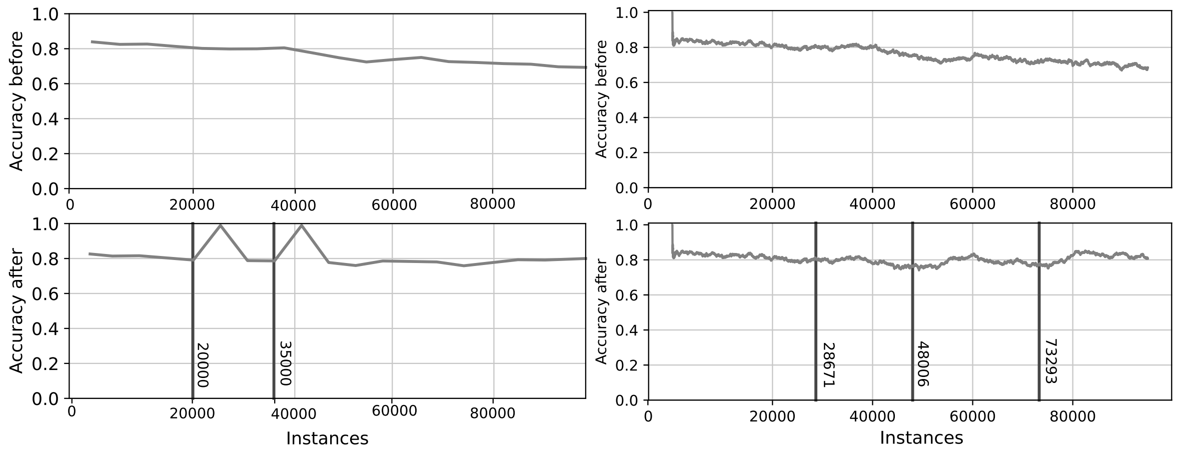

Fig. 6. Incremental drift: drift detection by the FIDD chunk-based detector (left) and the ADWIN instance-based detector (right) for the RF classifier

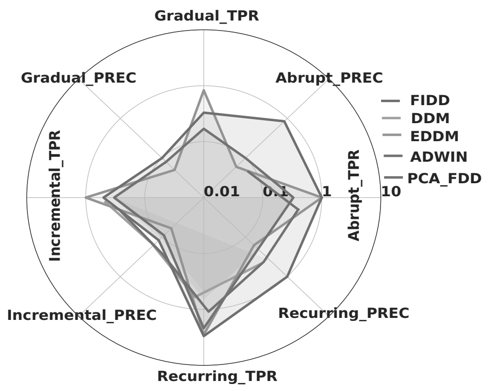

Fig. 7. Radar plot of TPR and Precision (logarithmic scale).

cooperating with the same classifier. The FIDD approach gives MCC values closer to $+1$ than other methods.

The classification efficiency results were checked using the Wilcoxon test. The Wilcoxon signed-rank-test, using W± distribution (two-tailed) for $\alpha = 0.05$ for real data from Tables 6 and 7, is as follows: FIDD vs DDM: $p = 0.016$, vs DDM: $p = 0.016$, vs ADWIN: $p = 0.016$, vs PCA-FDD: $p = 0.016$, for both the ACC and MCC factors, so there are significant differences between the proposed FIDD and other drift detection methods.

Similar studies were conducted on real data sets where drift points were defined. The data set (INSECTS) is presented by de Souza *et al*. (2020) and in Table 1.

All pairwise comparisons between FIDD and other detectors (DDM, EDDM, ADWIN, PCA-FDD) on the real INSECTS dataset (Table 8) yielded $p = 0.008$ using the two-tailed Wilcoxon signed-rank test ($\alpha = 0.05$). All statistical comparisons were performed employing the Wilcoxon signed-rank test on matched pairs under identical experimental conditions. The Wilcoxon test does not assume normality of the data but requires paired data. The conditions were met because each pair of results is from the same experimental condition (noise

Table 3. Average accuracy ACC of drift recognition by the RF classifier: 10 runs with different seeds each.

Drift	feature-noise[†]	FIDD	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**	0.842/4	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
	15-1	**0.875/2**	0.851/5	0.864/18	0.863/1	0.832/4
	15-3	**0.863/1**	0.846/4	0.855/18	0.854/1	0.825/4
	15-5	**0.855/1**	0.837/3	0.847/18	0.846/1	0.819/4
Gradual	10-1	0.828/2	0.822/3	**0.841/16**	0.823/3	0.760/0
	10-3	**0.836/2**	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
	15-1	**0.874/5**	0.804/4	0.817/15	0.789/4	0.744/2
	15-3	**0.855/4**	0.765/2	0.810/14	0.775/4	0.740/2
	15-5	**0.848/4**	0.755/1	0.804/15	0.768/4	0.736/2
Incremental	10-1	**0.820/2**	0.795/5	0.815/19	0.803/3	0.795/2
	10-3	**0.814/2**	0.787/2	0.806/16	0.788/3	0.788/2
	10-5	**0.807/2**	0.764/3	0.798/16	0.784/2	0.782/2
	15-1	**0.840/6**	0.746/1	0.755/19	0.757/4	0.746/4
	15-3	**0.824/6**	0.761/4	0.766/18	0.730/3	0.741/4
	15-5	**0.847/7**	0.736/2	0.762/19	0.732/3	0.739/4
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
	15-1	**0.880/3**	0.858/7	0.865/3	0.857/3	0.784/3
	15-3	**0.868/3**	0.829/5	0.842/13	0.842/3	0.773/3
	15-5	**0.855/3**	0.824/8	0.829/16	0.824/3	0.761/3

*: <ACC>/<number of drifts detected by the given method>.

[†]: <number of drifting features>-<% of label noise> in synthetic data.

level + drift type). Thus, FIDD shows statistically significant differences in performance compared to all baseline detectors.

The MCC results from Table 9 for FIDD are higher than those of other methods, but not high enough to consider this advantage statistically significant. This is due to the small number of samples. For this reason, Wilcoxon may not detect differences because it relies on ranks rather than actual MCC values. The Student t-paired test compares means and their deviations, so it will be better when the sample size is small, with null hypothesis $H_0 : \mu_1 = \mu_2$. The normality of samples was verified earlier using the Shapiro–Wilk test ($\alpha = 0.05$). The obtained p-values for all compared columns of Table 9 ranged between 0.45 and 0.63, indicating no significant discrepancy from normality.

For the t-paired Student test, we obtain FIDD vs DDM: $p = 0.0051$, vs EDDM: $p = 0.0196$, vs ADWIN: $p = 0.0051$, vs PCA-FDD: $p = 0.0068$. In the mentioned cases, $p < 0.05$; hence, significant differences exist between the proposed FIDD and other methods.

Retraining the classifier for each identified drift can lead to redundancy due to the high false alarm rate, even for state-of-the-art drift detectors. As shown in Table 10, the proposed FIDD detector has, in most cases, a low false positive error (FP) rate compared to other detectors, which is a very positive phenomenon. Drift detectors based on error rates frequently produce numerous false alarms, which can mislead the classifier and increase data analysis time.

6. Evaluation of the operating time of drift detectors

Tests were performed on a computer with an AMD Ryzen 5 7500 processor, 3.7 GHz, 64 GB main memory, and a 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 the base detector. This is a trivial conclusion, but it allows showing these dependencies for the analyzed datasets and selected drift detectors.

Figures 8 and 9 show the data analysis time of different detectors as a percentage. The results refer to synthetic data and various real-world data types discussed in the article. The presented methods for synthetic datasets (upper chart) show the average run time separately for the Abrupt, Gradual, Incremental, and Recurring sets. The color shade legend on the first chart defines the appropriate types of detectors.

Table 4. Average TP and FP indices values for drift detectors based on synthetic datasets from Table 3.

Detector	Abrupt		Gradual		Incremental		Recurring	
	TP	FP	TP	FP	TP	FP	TP	FP
Oracle result	1.00	0.00	1.00	0.00	1.00	0.00	3.00	0.00
FIDD	1.00	0.17	0.33	**2.83**	0.50	3.67	3.00	**0.00**
DDM	0.00	4.33	0.00	**2.83**	0.33	2.50	2.50	4.17
EDDM	1.00	17.00	0.83	14.83	1.00	16.83	2.33	14.50
ADWIN	1.00	**0.00**	0.00	3.50	0.00	3.00	2.83	0.17
PCA-FDD	0.33	3.17	0.00	4.67	0.33	**2.17**	0.67	2.50

Table 5. Wilcoxon signed-rank-test using W± distribution (two-tailed) for $\alpha = 0.05$: synthetic data for ACC values. Values inside the table are p-values of the statistic.

FIDD vs.	DDM	EDDM	ADWIN	PCA-FDD
Abrupt	0.034	0.035	0.035	0.031
Gradual	0.035	0.031	0.036	0.031
Incremental	0.031	0.035	0.031	0.031
Recurring	0.031	0.035	0.034	0.034

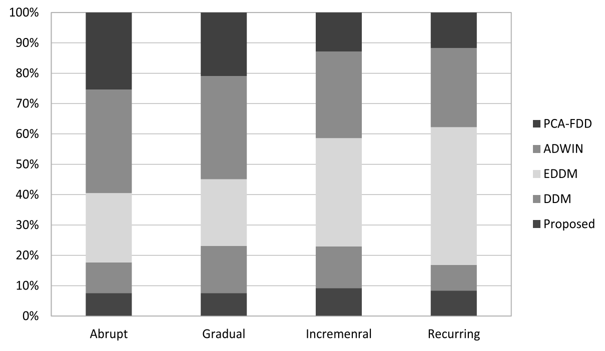

Fig. 8. Drift detector operation time versus the drift rate.

For real data, the proposed approach resulted in the shortest computation time for all datasets except the Arrhythmia and Ozone files. 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. Execution time for the INSECTS real-world dataset contains drifts of a very different nature, and its processing by different detectors is also relatively long due to the need for frequent retraining of the classifier. The observation of all the data sets used shows a favorably shorter feature drift detection time for the FIDD detector than for the others included in the study.

7. Complexity of the proposed method

For detecting drift, the FIDD detector applies the LASSO feature ranking method, so the computational complexity of the ranking process is a critical factor. When drift is detected, the subsequent data chunk is divided into L_N sub-chunks, with the LASSO method applied to each one. If no drift is detected, ranking is performed once over the entire chunk. This concept is presented by Algorithm 1 (Fig. 2).

Let L_d denote the number of detected drifts, L_c be the total number of chunks, and $\mathcal{O}(A)$ be the computational complexity of the LASSO procedure. Under these assumptions, the computational complexity is given by $\mathcal{O}(L_d \cdot L_N \cdot \mathcal{O}(A)) + \mathcal{O}((L_c - L_d) \cdot \mathcal{O}(A))$. Simplifying this expression, the complexity of the FIDD algorithm becomes $\mathcal{O}((L_N + L_c) \cdot \mathcal{O}(A))$. Since the number of sub-chunks is constant, the overall complexity of the FIDD algorithm is approximately $L_c \cdot \mathcal{O}(A)$. Given N instances and d features, and considering the LASSO implementation using the LARS algorithm (Efron *et al.*, 2004), the computational complexity of LASSO-LARS is $\mathcal{O}(d^3 + d^2 \cdot N)$. For our datasets, where $d \leq N$, this simplifies to $\mathcal{O}(A) = \mathcal{O}(d^2 \cdot N)$.

8. Discussion and conclusions

This section discusses the answers to the research questions formulated earlier in this study and summarizes the main conclusions regarding the proposed FIDD method.

- *RQ1*: Experiments conducted on both synthetic data—simulating different types of concept drift (sudden, gradual, incremental, and recurrent)—and real-world benchmarks show that FIDD effectively identifies changes in feature saliency even in the presence of label noise. With an adaptive threshold based on the standard deviation of feature saliency scores, FIDD adjusts its sensitivity to local fluctuations, enabling reliable detection of both sudden and subtle shifts in the data stream.

- *RQ2*: As shown in Tables 4 and 9, FIDD achieves a favorable trade-off between true positives (TPs) and false positives (FPs). The TP/FP ratio highlights its ability to detect significant drift points while minimizing false alarms, outperforming established detectors such as DDM, EDDM, ADWIN, and

Table 6. Accuracy (ACC) for the RF classifier. Drift points are unknown.

	FIDD	DDM	EDDM	ADWIN	PCA-FDD
Phishing	**0.964/6**	0.942/6	0.940/9	0.937/1	0.940/15
Electricity	**0.802/4**	0.762/18	0.759/18	0.745/13	0.724/2
Ozone	**0.954/18**	0.789/3	0.893/4	0.897/2	0.892/13
Arrhythmia	**0.977/19**	0.550/0	0.583/3	0.550/0	0.563/17
Poker-hand	**0.754/7**	0.642/0	0.640/6	0.642/0	0.642/0
Rialto	**0.732/5**	0.619/18	0.628/19	0.648/17	0.649/10
Outdoor	**0.915/18**	0.230/0	0.255/12	0.234/4	0.247/14

Table 7. Mathews correlation coefficient (MCC) for the RF classifier. Drift points are unknown.

	FIDD	DDM	EDDM	ADWIN	PCA-FDD
Phishing	**0.926**	0.883	0.879	0.873	0.879
Electricity	**0.611**	0.509	0.504	0.482	0.441
Ozone	**0.842**	0.174	0.121	0.072	0.050
Arrhythmia	**0.952**	0.550	0.261	0.114	0.155
Poker-hand	**0.539**	0.332	0.327	0.332	0.332
Rialto	**0.704**	0.577	0.587	0.609	0.614
Outdoor	**0.914**	0.220	0.236	0.218	0.238

Table 8. Accuracy (ACC) for the RF classifier and the INSECTS-type real datasets. Drift points are known.

	FIDD	DDM	EDDM	ADWIN	PCA-FDD
Abrupt balanced	**0.728/7**	0.552/8	0.583/17	0.555/7	0.593/15
Abrupt imbalanced	**0.809/6**	0.682/11	0.713/17	0.682/8	0.673/9
Gradual balanced	**0.737/5**	0.650/4	0.698/16	0.679/4	0.715/15
Gradual imbalanced	**0.806/5**	0.735/5	0.768/19	0.735/6	0.749/10
Incremental balanced	**0.789/10**	0.458/10	0.517/16	0.472/10	0.442/13
Incremental imbalanced	**0.822/7**	0.718/9	0.749/19	0.728/9	0.712/8
Reoccurring balanced	**0.764/9**	0.437/8	0.551/16	0.505/11	0.458/12
Reoccurring imbalanced	**0.812/6**	0.692/10	0.707/14	0.700/12	0.685/15

PCA-FDD, especially in environments with high noise or jitter.

- *RQ3*: The method was validated on both binary and multiclass classification tasks, confirming its versatility. The robustness to labeling noise was assessed by controlled noise injection and evaluation on the Electricity dataset, which naturally contains noisy labels. The results indicate that FIDD's adaptive thresholds reduce the impact of random fluctuations, leading to more stable drift detection and fewer false alarms.

- *RQ4*: In terms of computational efficiency, FIDD exhibits low memory and time complexity. Unlike detectors that require retraining classifiers or maintaining complex structures (e.g., decision trees), FIDD works with lightweight, piecewise LASSO regressions. The total execution time does not exceed that of standard reference methods, confirming its suitability for processing real-time data streams.

This study presented a new feature drift detection method based on observing changes in feature importance rankings, measured using LASSO regression. The proposed detector exploits feature importance fluctuations to identify changes in the data distribution, especially in the presence of virtual drifts, where traditional detectors may fail due to the lack of change in the decision boundary.

Extensive experiments were conducted on synthetic datasets with different types of drifts (sudden, gradual, incremental, recurrent) and multiple levels of label noise, as well as on real-world data streams. The results confirm that FIDD effectively captures change points in the feature distribution with minimal false positives. Remarkably, the method shows better performance in maintaining a balance between high true positive rates and reduced false positive rates, outperforming classical drift detectors such as DDM, EDDM, ADWIN, and PCA-FDD. FIDD offers the possibility of explanation through the analysis of feature rankings, which provides insight into which attributes are responsible for drift. Future work will focus on extending the method to unsupervised or semi-supervised learning scenarios, as well as exploring alternative ranking mechanisms based on examining the distances of conditional distributions in data blocks.

Table 9. Accuracy (MCC) for the RF classifier and the INSECTS-type real datasets. Drift points are known.

	FIDD	DDM	EDDM	ADWIN	PCA-FDD
Abrupt imbalanced	**0.745**	0.587	0.626	0.586	0.573
Gradual imbalanced	**0.740**	0.660	0.702	0.660	0.660
Incremental imbalanced	**0.770**	0.635	0.675	0.648	0.624
Reoccurring imbalanced	**0.751**	0.600	0.617	0.609	0.585

Table 10. Average TP and FP indices value based on the real-world INSECTS datasets from Table 8.

Detector	Abr-bal		Abr-imb		Gra-bal		Gra-imb		Inc-bal		Inc-imb		Rec-bal		Rec-imb	
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
Oracle result	5	0	5	0	1	0	1	0	2	0	2	0	2	0	2	0
FIDD	5	**2**	2	**4**	2	**3**	1	**4**	2	**8**	1	**6**	2	**7**	0	**6**
DDM	5	3	2	9	1	**3**	1	**4**	2	**8**	0	9	1	**7**	0	10
EDDM	4	13	1	16	2	14	1	18	2	14	0	19	2	14	0	14
ADWIN	4	3	3	5	1	**3**	0	6	2	**8**	1	8	2	9	0	12
PCA-FDD	7	8	0	9	4	11	0	10	1	12	0	8	1	11	0	15

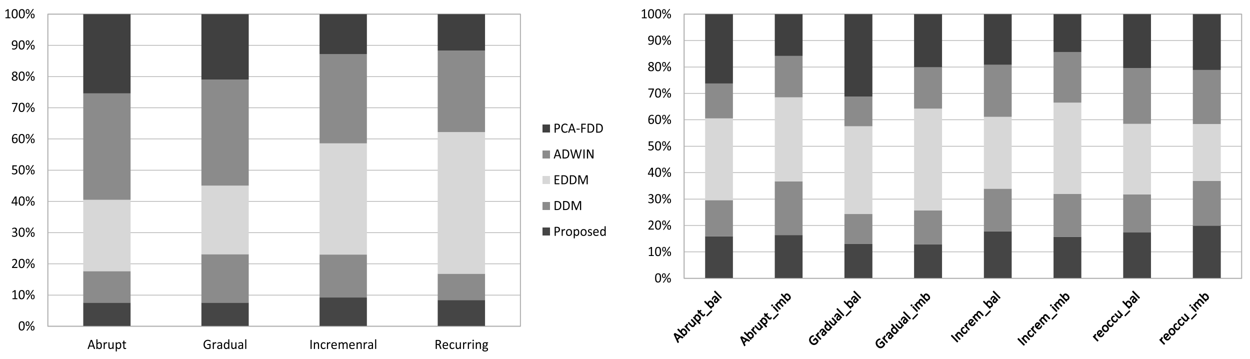

Fig. 9. Drift detectors operating time depending on the input data type: from artificial to real.

All algorithms and datasets used in this study are available in the GitHub repository at `https://github.com/ZSKPP/feature-drift`.

References

Agrahari, S. and Singh, A.K. (2022). Adaptive PCA-based feature drift detection using statistical measure, *Cluster Computing* **25**(6): 4481–4494.

Baena-Garcia, M., Campo-Avila, J., Bifet, A., Gavald, R. and Morales-Bueno, R. (2006). Early drift detection, *in* A.L.C. Bazzan and S. Labidi (Eds), *Advances in Artificial Intelligence,* Lecture Notes Artificial Intelligence, Vol. 3171, Springer, Berlin/Handelberg, pp. 286–295.

Bartz-Beielstein, T. and Lukas, H. (2024). Drift detection and handling, *in* E. Bartz and T. Bartz-Beielstein (Eds), *Online Machine Learning: A Practical Guide with Examples in Python*, Springer Nature Singapore, Singapore, pp. 23–39.

Bifet, A. and Gavaldà, R. (2009). Adaptive learning from evolving data streams, *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII (IDA'09), Lyon, France*, pp. 249–260.

Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B. (2010). MOA: Massive online analysis, *Journal of Machine Learning Research* **11**(2010): 1601–1604.

Blackard, J. (1998). Covertype, Dataset, *UCI Machine Learning Repository*, DOI: 10.24432/C50K5N.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(2001): 5–32.

Chicco, D., Tötsch, N. and Jurman, G. (2021). The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation, *BioData Mining* **14**(13): 1–22.

Dhal, P. and Alad, C. (2022). A comprehensive survey on feature selection in the various fields of machine learning, *Applied Intelligence* **52**(2022): 4543–4581.

de Souza, V.M.A., dos Reis, D.M., Maletzke, A.G. and Batista, G.E.A.P.A. (2020). Challenges in benchmarking stream learning algorithms with real-world data, *Data Mining and Knowledge Discovery* **34**(2020): 1805–1858.

Duda, P., Przybyszewski, K. and Wang, L. (2020). A novel drift detection algorithm based on features' importance analysis in a data streams environment, *Journal of Artificial Intelligence and Soft Computing Research* **10**(4): 287–298.

Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression, *The Annals of Statistics* **32**(2): 407–451.

Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?, *Journal of Machine Learning Research* **15**(1): 3133–3181.

Frank, A. (2010). *UCI Machine Learning Repository*, University of California, Irvine, `http://archive.ics.uci.edu/ml`.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A. (2014). A survey on concept drift adaptation, *ACM Computing Surveys* **46**(2014): 1–37.

Gonçalves, P.M., de Carvalho Santos, S.G., Barros, R.S. and Vieira, D.C. (2014). A comparative study on concept drift detectors, *Expert Systems with Applications* **41**(18): 8144–8156.

Guo, H., Li, H., Ren, Q. and Wang, W. (2022). Concept drift type identification based on multi-sliding windows, *Information Sciences* **585**(2022): 1–23.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection, *Journal of Machine Learning Research* **3**: 1157–1182.

Japkowicz, N. and Boukouvalas, Z. (2024). *Machine Learning Evaluation: Towards Reliable and Responsible AI*, Cambridge University Press, Cambridge.

Mielniczuk, J. and Wawrzeńczyk, A. (2025). Accounting for label shift of positive unlabeled data under selection bias, *International Journal of Applied Mathematics and Computer Science* **35**(3): 507–517, DOI: 10.61822/amcs-2025-0036.

Montiel, J., Read, J., Bifet, A. and Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework, *Journal of Machine Learning Research* **19**(72): 1–5.

Porwik, P. and Doroz, R. (2021). Adaptation of the idea of concept drift to some behavioral biometrics: Preliminary studies, *Engineering Applications of Artificial Intelligence* **99**(2021): 104135.

Saeys, S., Inza, I. and Larrañaga, P. (2007). A review of feature selection methods for machine learning, *Bioinformatics* **23**(19): 2507–2517.

Stapor, K., Ksieniewicz, P., García, S. and Woźniak, M. (2021). How to design the fair experimental classifier evaluation, *Applied Soft Computing* **104**(2021): 107219.

Stefanowski, J., Krawiec, K. and Wrembel, R. (2017). Exploring complex and big data, *International Journal of Applied Mathematics and Computer Science* **27**(4): 669–679, DOI: 10.1515/amcs-2017-0046.

Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A. (2009). A detailed analysis of the KDD CUP 99 data set, *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, Canada*, pp. 1–6.

Usman, M., Sher, M. and Khan, M.N.A. (2023). A survey on feature selection techniques based on filtering criteria, *Information* **14**(3): 191.

Yamada, M., Jitkrittum, W., Sigal, L., Xing, E.P. and Sugiyama, M. (2014). High-dimensional feature selection by feature-wise kernelized Lasso, *Neural Computation* **26**(1): 185–207.

Zhao, D. and Koh, Y.-S. (2020). Feature drift detection in evolving data streams, *in* S. Hartmann *et al.* (Eds), *Database and Expert Systems Applications (DEXA 2020)*, Lecture Notes in Computer Science, Vol. 12392, Springer, Cham, pp. 319–328.

Zhu, Q. (2020). On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset, *Pattern Recognition Letters* **136**(2020): 71–80.

Žliobaitė, I. (2010). Learning under concept drift: An overview, *arXiv* 1010.4784.

Piotr Porwik is a professor of computer science at the Computer Science Institute, University of Silesia, Poland. His research focuses on machine learning, classification methods, data stream mining, and biometrics. Professor Porwik has published over 200 conference and journal papers. He has also been awarded several best paper awards at prestigious conferences. He serves as a program committee chair and a member of numerous scientific events. Professor Porwik is also a member of editorial boards of high-ranked journals.

Tomasz Orczyk is a research scientist and academic teacher. He received his doctoral degree in computer science from the University of Silesia in Katowice in 2018, where he continues his academic career. His research primarily focuses on machine learning, particularly the classification of incomplete data, biometrics, and methods for handling concept drift. He has published over 40 conference and journal papers.

Nathalie Japkowicz has been a full professor of computer science at American University, Washington DC, since August 2016. Prior to that, she had directed the Laboratory for Research on Machine Learning Applied to Defense and Security at the University of Ottawa in Canada, and trained over 30 graduate students and collaborated with many Canadian governmental agencies and private industry. Her publications include two co-authored books by Cambridge University Press, one edited book in Springer, as well as over 150 book chapters, journal articles, and conference or workshop papers. Her main research interests are in the area of machine learning.

Received: 18 April 2025
Revised: 7 July 2025
Accepted: 17 September 2025