The Journal of Supercomputing (2024) 80:7781–7811
https://doi.org/10.1007/s11227-023-05729-8

# An adaptive XGBoost-based optimized sliding window for concept drift handling in non-stationary spatiotemporal data streams classifications

**Ature Angbera**[1,2] · **Huah Yong Chan**[1]

Accepted: 14 October 2023 / Published online: 10 November 2023

## Abstract
In recent years, the popularity of using data science for decision-making has grown significantly. This rise in popularity has led to a significant learning challenge known as concept drifting, primarily due to the increasing use of spatial and temporal data streaming applications. Concept drift can have highly negative consequences, leading to the degradation of models used in these applications. A new model called BOASWIN-XGBoost (Bayesian Optimized Adaptive Sliding Window and XGBoost) has been introduced in this work to handle concept drift. This model is designed explicitly for classifying streaming data and comprises three main procedures: pre-processing, concept drift detection, and classification. The BOASWIN-XGBoost model utilizes a method called Bayesian-Optimized Adaptive Sliding Window (BOASWIN) to identify the presence of concept drift in the streaming data. Additionally, it employs an optimized XGBoost (eXtreme Gradient Boosting) model for classification purposes. The hyperparameter tuning approach known as BO-TPE (Bayesian Optimization with Tree-structured Parzen Estimator) is employed to fine-tune the XGBoost model's parameters, thus enhancing the classifier's performance. Seven streaming datasets were used to evaluate the proposed approach's performance, including Agrawal_a, Agrawal_g, SEA_a, SEA_g, Hyperplane, Phishing, and Weather. The simulation results demonstrate that the suggested model achieves impressive accuracy values of 70.83%, 71.02%, 76.76%, 76.96%, 84.26%, 95.53%, and 78.35% on the corresponding datasets, affirming its superior performance in handling concept drift and classifying streaming data.

**Keywords** Concept drift · Machine learning · Sliding windows · Spatiotemporal data streams · Bayesian optimization

✉ Ature Angbera
angberaature@student.usm.my

[1] School of Computer Sciences, Universiti Sains Malaysia, 11800 Pulau Pinang, Malaysia

[2] Department of Computer Science, Joseph Sarwuan Tarka University, Makurdi, Nigeria

## 1 Introduction

In the modern digital age, numerous applications create enormous spatiotemporal data streams that must be instantly sorted and analyzed. For a variety of systems, the capacity to understand spatiotemporal data streams in real-time is essential. Processing enormous amounts of spatiotemporal data from various sources, including online traffic, social media, sensor networks, and others, is a significant challenge [1]. Because of this, storing a lot of data for analysis is impractical. Currently, the classification of this infinitely evolving data stream is being challenged by concept drifts. Concept drift is the process by which the spread of the data input or the association between the input and the desired label alters over time. Practically, there are three ways that concept drift can happen in a situation of stream learning: (a) "sudden/abrupt drift", where there is a total change in the distribution of data; (b) "gradual drift", where the current concept gradually shifts to another concept over time; and (c) "recurring drift", where the old concept reappears after a specific interval of time [2]. Figure 1 depicts the practical types of concept drifts.

Assuming a sample $(X, Y)$ in the data stream has three potential classes $(Y = \left(y_1, y_2, y_3\right))$, as shown in Fig. 2, and a two-dimensional feature vector $(X = \left(x_1, x_2\right))$ as well. At time *t,* the samples exhibit a specific distribution $P_t(X, Y)$. Concept drift happens when $P_t(X, Y) \neq P_{t+1}(X, Y)$ and the distribution changes at time $t + 1$. Concept drift, as mentioned above, can be caused by the

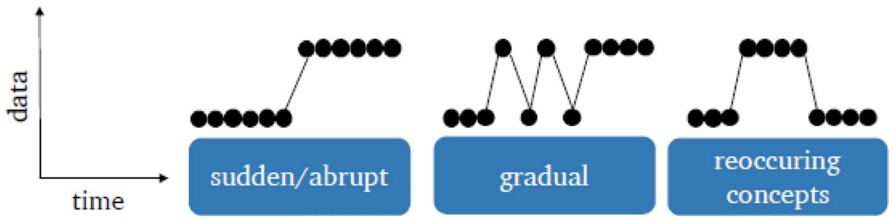

**Fig. 1** Structural types of concept drift

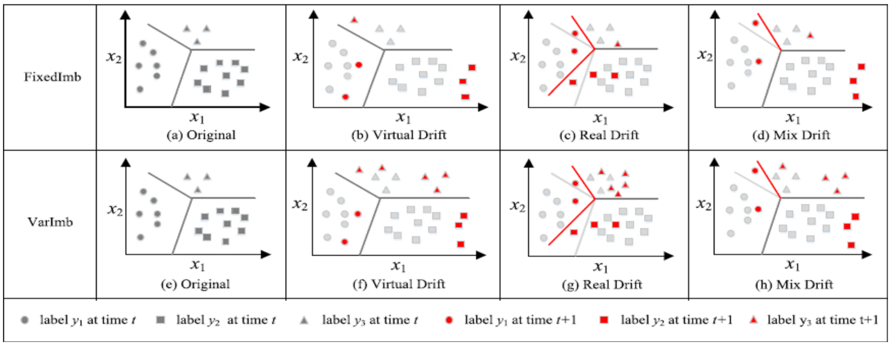
FixedImb
VarImb
$x_2$
$x_1$
(a) Original
(b) Virtual Drift
(c) Real Drift
(d) Mix Drift
(e) Original
(f) Virtual Drift
(g) Real Drift
(h) Mix Drift
● label $y_1$ at time $t$
■ label $y_2$ at time $t$
▲ label $y_3$ at time $t$
● label $y_1$ at time $t+1$
■ label $y_2$ at time $t+1$
▲ label $y_3$ at time t+1

**Fig. 2** Factors causing concept drift [4]

three factors that correspond to the equation $P_t(X, Y) = P_t(X) * P_t(Y|X)$, virtual, real, and mixed drift [3], as shown in Fig. 2. Virtual drift occurs when the decision boundary ($P_t(Y|X) = P_{t+1}(Y|X)$) does not change but the feature vector distribution does, as seen in Fig. 2b, f, that is: $P_t(X) \neq P_{t+1}(X)$. Real drift on the other hand happens when the decision boundary shifts, as in Fig. 2c, g, where $P_t(Y|X) \neq P_{t+1}(Y|X)$, but the feature vector distribution does not change; that is: $P_t(X) = P_{t+1}(X)$. Finally, as seen in Fig. 2d, h, mixed drift happens when both the decision boundary change ($P_t(Y|X) \neq P_{t+1}(Y|X)$) and the feature vector distribution change ($P_t(X) \neq P_{t+1}(X)$) [4]. Figure 2a, e indicates the original nature of the dataset when the concept has not changed. Concept drift, a critical aspect in evolving data analysis, can be categorized in two ways based on alterations in the class prior probability denoted as $P(Y)$. The first category, "FixedImb," as shown in Fig. 2, signifies concept drift characterized by a fixed imbalance ratio. In this scenario, the class prior probability $P(Y)$ remains constant, while variations occur in the class-conditional probability $P(X|Y)$. The second category, "VarImb," represents concept drift with a variable imbalance ratio. In VarImb, the class prior probability $P(Y)$ changes, as visually represented in Fig. 2. It is important to note that the study did not specifically delve into the investigation of class imbalance but focused on the broader concept drift phenomenon.

The learning model performs worse if the drifts are left unattended. Concept drift is the most challenging issue in real-time learning because it significantly affects the consistency of streaming spatiotemporal data classification [5]. As a result, real-time analytics on streaming or non-stationary spatiotemporal data have recently caught the attention of researchers [6]. Spatiotemporal data streams are data collections that flow continuously and alter as they enter a system. According to [7], data streams can be enormous, ordered promptly, changed quickly, and potentially endless in duration. Due to the periodic data changes on the streaming platform, the typical mining method needs to be upgraded [8].

Constructing models that can adjust to the online adaptive analytics for the anticipated and unanticipated variations in the spatiotemporal data is vital. The importance is because the traditional machine learning (ML) models cannot handle concept drift [3]. As a result, this research suggests an ML-based drift adaptive framework for spatiotemporal streaming data analytics, which deals with data that has both spatial and temporal dimensions.

The framework consists of a "Bayesian Optimization with Tree-structured Parzen Estimator (BO-TPE)" approach for model optimization, an eXtreme Gradient Boosting model (XGBoost) for learning spatiotemporal data, and a newly proposed method called Bayesian-Optimized Adaptive and Sliding Windowing (BOASWIN) for adaptation of concept drift. The effectiveness and efficiency of the suggested adaptive framework are assessed using seven open-source datasets. The following can be used to summarize the main article's contributions:

- *Novel Drift Adaptation Approach* The paper introduces a novel method called "BOASWIN" to address the challenge of concept drift in spatiotemporal data.

BOASWIN offers a fresh perspective by combining Bayesian optimization and sliding window techniques. This innovative approach not only detects changes in data distribution but also optimizes model parameters to adapt effectively.

- *Efficient Adaptive Framework* The research presents an adaptive framework that combines "BO-TPE" for model optimization, an "XGBoost" for learning spatiotemporal data, and the BOASWIN method for concept drift adaptation. This framework offers offline and online learning functionalities, enhancing its efficiency for spatiotemporal data categorization use cases.
- *Experimental Evaluation* The proposed approach is empirically evaluated using seven open-source datasets and compared against contemporary techniques. This evaluation provides evidence of the framework's effectiveness and efficiency in handling spatiotemporal data streams with varying patterns and concept drift.

While the proposed framework for spatiotemporal streaming data analytics presents several noteworthy contributions, some limitations deserve attention. Firstly, the size and complexity of the spatiotemporal datasets it encounters might influence the framework's effectiveness. Extensive experimentation on datasets with varying scales and dimensions is imperative to gauge their scalability accurately. Additionally, despite its efficiency, the framework may introduce some computational overhead when dealing with particularly large-scale data streams. Thus, further research should explore strategies to optimize its computational efficiency.

## 2 Related works

Various concept drift learning strategies have been developed recently to adjust to shifting concepts [9–11]. "Concept drift detectors" strive to spot changes in streams by either keeping an eye on the streams' distribution or the performance of a classifier concerning some standard, like accuracy. The Adaptive Sliding Window (ADWIN) [12, 13] is a standard method for assessing a classifier's prediction accuracy, and it works under the presumption that if a change in performance is seen, the concept has been altered [14]. ADWIN breaks a window $W$ into two adaptive subwindows, analyses the underlying statistics, and utilizes $W$ to detect distribution changes. If no change is discovered, the main window enlarges; if a difference in the statistics of the subwindows is discovered, it shrinks. Hoeffding Bound [14] allows for the recognition of the change. The "Drift Detection Method (DDM)" [15, 16], a well-liked model performance-based approach, establishes two thresholds—a warning level and a drift level—to track changes in the standard deviation and error rate of the model for drift detection [15].

In DDM, concept drift is a frequent phenomenon characterized by a significant increase in the model's overall error rate and standard deviation. Since a learner will only be changed when its performance significantly deteriorates, DDM is easy to use and can prevent unnecessary model modifications. While DDM is good at detecting abrupt drift, it frequently responds slowly to gradual drifts. The disadvantage happens because memory overflows result from storing many data samples to meet the drift level of a long, slow drift [17]. "Early drift detection method (EDDM)", a variation of DDM [18], examines the distance between two successive misclassifications rather than the total number of misclassifications. One benefit of this detector is its lack of an input parameter [6]. An incremental learning system called Online Passive-Aggressive (OPA) [19] adapts to drift by passively responding to accurate predictions and forcefully reacting to errors. The standard $K$-Nearest Neighbors (KNN) model for online data analytics has been improved by "Self-Adjusting Memory with KNN (SAM-KNN)" [20]. The SAM-KNN algorithm uses two memory modules to adjust to concept drift: Short-term memory (STM) for the present concept and long-term memory (LTM) for prior conceptions [20]. Lu et al. presented the chunk-based dynamic weighted majority to analyze data streams with concept drift, in which the chunk size was adaptively chosen using statistical hypothesis testing [5]. To increase the classification accuracy, Zhang et al. suggested a three-layer concept drift detection method [21]. A framework for drifting data stream classification that integrates data pre-processing and the dynamic ensemble selection approach is proposed. It uses stratified bagging to train base classifiers [22]. Concept drift detection is achieved using a cluster-based histogram, and segmentation loss minimization increases the method's sensitivity [11]. In [23], the selective ensemble technique suggests adopting a deep neural network to solve the concept drift problem. Shallow and deep features are merged in the depth unit to improve the convergence of the online deep learning model. A semi-supervised classification system was suggested by Din et al., where the micro-clusters were dynamically maintained to capture idea drift in data streams [24]. The diversified dual ensemble model is built for the drifting data stream, where the weights are updated dynamically and adaptively to identify gradual drift and rapid drift [25]. The aforementioned studies are successful at resolving concept drift, but based on this review, classifier performance received more focus than stream data distributions. As a result, in this research, we examine both classifier performance and streaming data distribution for better classifications.

## 3 Proposed model

This study proposed an optimized adaptive sliding window with the XGBoost model called the BOASWIN-XGBoost model, which monitors the classifier's performance and regulates the streaming distribution of data into the classifier for improved classification. Figure 3 illustrates the three components of the suggested paradigm: preprocessing (stage 1), concept drift detection and classification. An initial XGBoost model is trained using the historical dataset. Additionally, Bayesian Optimization, a hyperparameter optimization (HPO) technique, is used to adjust the XGBoost model's hyperparameters to produce the optimized XGBoost model.

The suggested system will handle the data streams continuously produced throughout time. The next step in this method is processing (stage 3) the data streams using the initial XGBoost model obtained. Suppose concept drift (stage 2) is discovered in the new data streams using the proposed BOASWIN approach, fitting the current concept of the new data streams. In that case, the XGBoost model will

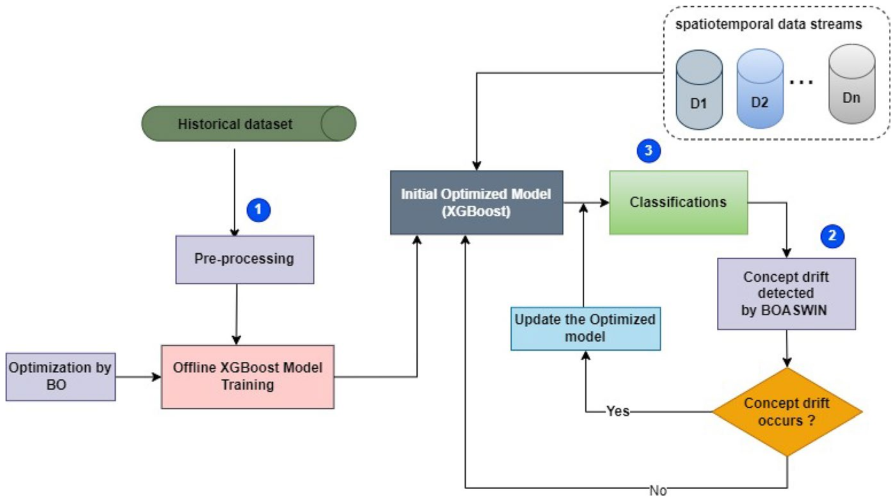
spatiotemporal data streams
D1
D2
...
Dn
Historical dataset
1
Pre-processing
Optimization by BO
Offline XGBoost Model Training
Initial Optimized Model (XGBoost)
3
Classifications
2
Concept drift detected by BOASWIN
Update the Optimized model
Concept drift occurs ?
Yes
No

**Fig. 3** Proposed model framework

be retrained on the new concept data samples obtained by the adaptive window of BOASWIN. The suggested system can adapt to the new data streams' ever-changing patterns to maintain correct classifications. The classifier's performance is monitored, and the distribution of data streams is also controlled.

## 3.1 XGBoost model

A powerful ensemble machine learning model built on decision trees is the eXtreme Gradient Boosting (XGBoost) model [26]. The XGBoost discussed in [27] was created using a GBDT (Gradient Boosting Decision Tree), and it was shown to have excellent convergence and generalization speed [28]. In [28], the XGBoost algorithm's goal function and optimization strategy were introduced. XGBoost's target function is given by Eq. 1 [29].

$$Obj(\theta) = L(\theta) + \Omega(\theta) \tag{1}$$

where $L(\theta) = l\left(\hat{y}_i, y_i\right)$ and $\Omega(\theta) = \gamma \mathrm{T} + \frac{1}{2}\lambda \| \omega \|^2$.

The objective function ($Obj(\theta)$) which is to be optimized is divided into two sections: $L(\theta)$ and $\Omega(\theta)$. $\theta$ corresponds to the formula's numerous parameters. The goal is to find the values of $\theta$ that minimize this function. The difference between the forecast $\hat{y}_i$ and the target $y_i$ is measured by $L(\theta)$, a differentiable convex loss function. The point is to demonstrate how to incorporate the facts into the framework [29]. Convex loss functions frequently employed, such as the mean square loss function in Eq. 2 and the logistic loss function shown in Eq. 3, can be employed in the above equation.

$$l(\widehat{y}_i, y_i) = (\widehat{y}_i - y_i)^2 \tag{2}$$

$$l(\widehat{y}_i, y_i) = y_i \ln\left(1 + e^{-\widehat{y}_i}\right) + (1 + y_i)\ln\left(1 + e^{\widehat{y}_i}\right) \tag{3}$$

Complex models are penalized by the regularized term $\Omega(\theta)$. T is the number of leaves in the tree, and $\gamma$ is the learning rate, which ranges from 0 to 1. When multiplied by T, it equals spanning tree pruning, which prevents overfitting. When compared to the classic GBDT algorithm, the XGBoost algorithm increases the term $\frac{1}{2}\lambda\|w\|^2$. The regularized parameter is, while $w$ is the weight of the leaves. The value of this item can be increased to prevent the model from fitting and to improve its generalization capabilities. On the other hand, including model penalty items with functions as parameters leads to the failure of classical approaches to be optimized by the objective function in Eq. 1. As a result, we must assess if we can learn to obtain the aim $y_i$ as seen in Eq. 4 [29].

$$Obj(\theta) = \sum_{i=1}^{n} l\left(y_i, \widehat{y}_i^{(t-1)} + S_t(\mathrm{T}_i)\right) + \Omega(\theta) \tag{4}$$

where, in the $t$ iteration, $S_t(\mathrm{T}_i)$ denotes the tree produced by instance $i$, and $n$ number of points.

The optimization target in each iteration is to build a tree design that minimizes the aimed function. Hence, when solving the square loss function, the objective function of Eq. 4 is optimal, but it becomes tricky when calculating other loss functions. As a result, Eq. 4 translates Eq. 5 using the two-order Taylor expansion, allowing further loss functions to be solved.

```
Input: $I$, current node's instance set
Input: $d$, dimension of the characteristic
  i:    gain ← 0
  ii:   G ← $\sum_{i \in I} g_i$
  iii:  for $k = 1$ to $m$ do
  iv:       $G_L \leftarrow 0$
  v:        for j in sorted $(I, by\ X_{jk})$ do
  vi:           $G_L \leftarrow G_L + g_j$
  vii:      end for
  viii:     $G_L \leftarrow G_L + g_j$
  ix:       $G_R \leftarrow G - G_L$
  x:        $score \leftarrow max(score, G_L + G_R - G)$
  xi:   end for
  Result: Split with max score
```

**Algorithm 1** Split finding greed algorithm

$$Obj(\theta) = \sum_{i-1}^{n} \left[ l\left(y_i, \hat{y}^{(t-1)}\right) + g_i S_t\left(\mathrm{T}_i\right) + \frac{1}{2} h_i S_t^2\left(\mathrm{T}_i\right) \right] + \Omega(\theta) \tag{5}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ which is the 1st derivative of the error function and $h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ is the 2nd derivative of the error function.

Because the tree model needs to find the best segmentation points and store them in several blocks, the algorithm ranks the eigenvalues based on the realization of XGBoost. This structure is reused in subsequent iterations, resulting in a significant reduction in computing complexity. Furthermore, the information gain of each feature must be determined during the node splitting process, which employs the greed algorithm, as shown in algorithm 1, allowing the calculation of information gain to be parallelized [28].

Algorithm 1 utilizes a greedy approach as its fundamental strategy. Its primary concept involves an initial sorting of the data based on eigenvalues. Subsequently, it proceeds by iterating through each feature. It considers every possible value as a potential splitting point for each feature and computes the corresponding gain and loss. After evaluating all features in this manner, the algorithm identifies the most distinctive value for gain loss as the optimal splitting point. Within the algorithm, '*j*' represents the index used to iterate through all eigen attribute values during the sorting process, while '*k*' is employed to iterate through all samples.

## 3.2 Dynamic hyperparameter tuning

Dynamic hyperparameter tuning strategies are pivotal in ensuring that machine learning models maintain their effectiveness in changing data characteristics. These strategies enable models to adapt and optimize their hyperparameters to match evolving data distributions.

Dynamic hyperparameter tuning, as outlined in the literature [30], involves automatically adjusting hyperparameters during training and inference. One of the primary ways it achieves this is through a learning rate schedule. Learning rate scheduling dynamically adapts the learning rate based on performance metrics or predefined schedules. For instance, the learning rate may be reduced when the loss plateaus, allowing the model to fine-tune its parameters more delicately in response to data changes [31].

Another strategy involves early stopping, a widely recognized technique [32]. By monitoring a validation metric such as validation loss, the model's training can be halted when it begins to deteriorate, thereby preventing overfitting and ensuring that the model remains robust to variations in data characteristics. Adaptive optimizers like Adam and RMSprop [33] are also valuable in dynamic hyperparameter tuning. These optimizers adapt the learning rates for individual model parameters based on their gradients, allowing the model to navigate through varying data landscapes effectively.

Hyperparameter search methods, as discussed in research by Wu et al. [34], can continuously seek optimal hyperparameter configurations as data evolves. Techniques like Bayesian optimization or grid search can be employed to identify the best hyperparameters for the current data distribution, ensuring the model's adaptability. Ensemble models [35] and online learning [19] are further strategies for adapting models to changing data patterns by combining multiple models or incrementally updating the model with new data.

In summary, dynamic hyperparameter tuning strategies, backed by research in the field, provide the means for models to adapt their hyperparameters to continuously changing data characteristics. By incorporating these strategies, machine learning models can maintain their performance and relevance over time. They are well-suited for real-world applications where data is subject to fluctuations and shifts.

### 3.2.1 Bayesian optimization (BO)

BO [36] models were created to solve optimization issues. BO comprises two essential components: surrogate models for simulating the objective function and an acquisition function for measuring the value produced by the objective function's assessment at a new location [37]. These activities, exploration and exploitation occur during BO processes. Exploration is exploring previously unexplored areas, whereas exploitation is analyzing samples in the current zone where the global optimum is most likely. These activities should be balanced according to BO models [38]. "The Gaussian Process (GP)" and "The Tree Parzen Estimator (TPE)" are two popular models used as BO surrogate models [39, 40]. Based on the surrogate model used, BO models can be divided into BO-GP and BO-TPE models [41]. In this study, we adopted the BO-TPE due to the drawback of BO-GP, which restricts parallelizability due to its cubic computational complexity, $O(n^3)$. Among BO surrogate models, the Tree-structured Parzen Estimator (TPE) [40] is well-liked. BO-TPE creates two density functions, $l(x)$ and $g(x)$, that act as generative models for all processed data instead of deriving a prediction distribution for the objective function. As part of BO-TPE, the input data is divided into two groups (good and poor observations) depending on a predetermined threshold * that is modeled using standard Parzen windows (Eq. 6):

$$p(x|y,D) = \begin{cases} l(x), if\ y < y^* \\ g(x), if\ y > y^* \end{cases} \tag{6}$$

where $y=f(x)$ represents the prediction for input data $x$, and $D$ is the configuration search space.

Due to its capacity to optimize complex configurations with low computational complexity of $O(n\log n)$, BO-TPE has demonstrated excellent performance when applied to a variety of machine learning applications [37, 38]. Furthermore, TPE can accurately handle conditional variables because it uses a tree structure to keep conditional dependencies [42]. Hence, we used the BO to optimize the proposed adaptative XGBoost and the adaptive sliding windows for effective concept drift handling in spatiotemporal data streams.

### 3.3 Bayesian optimized adaptive sliding window (BOASWIN)

As seen from the literature [13, 17], the adaptive sliding windows continue to grow large enough to detect a drift; however, this is a drawback. Since drifts like the gradual drift may occur unnoticed, this will give the classifier a wrong classification since the gradual drift is not detected promptly. To handle this challenge and check the distribution of the streams, we proposed BOASWIN. Here, our windows are made to be variables, giving room for close monitoring.

The BOASWIN approach is suggested in this study to provide reliable analytics. It is intended to detect concept drift and adapt to the continually changing data stream. BOASWIN was created based on synthesizing concepts from sliding and adaptable window-based methods, performance-based approaches, and window-based strategies. Two essential functions, "ConceptDriftAdaptation" and "HPO_BO-TPE," comprise the entire BOASWIN technique. With the help of the supplied hyperparameter settings, the "ConceptDriftAdaptation" function seeks to identify concept drift in streaming data and update the XGBoost model with fresh concept samples for drift adaptation. The "ConceptDriftAdaptation" function's hyperparameters are tuned and optimized using BO-TPE by the "HPO_BO-TPE" function. A sliding window ($P$) for concept drift detection and an adaptive window ($P_{\max}$) for storing new concept samples are the two different types of windows in BOASWIN. The concept drift detection method also uses two thresholds to indicate the drift level: $\alpha$ and the warning level: $\beta$.

Four parameters in the BOASWIN proposed method, $\alpha$, $\beta$, $P$, and $P_{\max}$ are the crucial hyperparameters that directly affect how well the BOASWIN model performs. Since BO is successful for both discrete and continuous hyperparameters, to which the hyperparameters of BOASWIN correspond, it is utilized to change these four hyperparameters to provide the optimal adaptive learner. We adopted the adaptive sliding window algorithm proposed in [43] [44] and modified it to achieve our goals. Hence, the algorithm for our proposed BOASWIN is given in algorithm 2.

## 4 Experimental analysis

Our experiment is carried out and analyzed using the River [45] library and Python 3.9. Our suggested approach, BOASWIN-XGBoost, is compared to seven cutting-edge models, including ADWIN, DDM, EDDM, OPA, SAM-KNN, SRP, and XGBoost. Here, we aim to assess a fair comparison between these models regarding how well they perform in the face of various types of concept drift.

### 4.1 Datasets used in the study

Where an actual drift genuinely is must be determined before we can evaluate a drift detector's performance using the various detection criteria. Only synthetic datasets make this possible. The scikit-multiflow framework enables the creation of various types of synthetic data to simulate the occurrence of drifts [46]. Table 1 includes specific details about the seven datasets that were used in this research.

```
Input: $\alpha$: the alert level
       $\beta$: the drift level
       $S$: data streams
       $P$: size of the sliding window
       $P_{max}$: size of the adaptive sliding window
       $K$: classifier
       $Space$: hyperparameter configuration space
       $MaxTime$: the maximum hyperparameter search times.
Output: $HP_{opt}$: the detected optimal hyperparameter values
       $MaxAcc$: the average accuracy of the overall
       $State$: either the warning level or drift/change level
1.  Function: ConceptDriftAdaptation($S, K, \alpha, \beta, P, P_{max}$):
2.      $W' \leftarrow 0$                //Initialise the adaptive window
3.      $State \leftarrow 0$             // An indicator of normal, drift and alert state
4.      for all samples $x_i \in S$ do
5.          $W_i \leftarrow$ a sliding window of the $P$ samples of $x_i$
6.          $AccWIN_i \leftarrow accuracy(W_i)$   //Current window accuracy
7.          $AccWIN_{i-P} \leftarrow accuracy(W_{i-P})$   // Last window accuracy
8.          If (Indicator ==0)&&( $AccWIN_i < \alpha * AccWIN_{i-P}$) Then
9.              $W' \leftarrow W' \cup \{x_i\}$
10.             $State \leftarrow 1$     // Warning occurs
11.         end
12.         If $State == 1$ Then     //The warning state
13.             $P' \leftarrow Size(W_i)$
14.             If $AccWIN_i < \beta * AccWIN_{i-P}$ Then   //New window
                                                  accuracy drops to drift level
15.                 $State \leftarrow 2$;        //Drift occurs
16.                 $f \leftarrow i$  //Obtained the first new concept window acc.
17.                 $K \leftarrow$ Retrain classifier on $W'$   //Retrain K on
                                                        the new concept
18.             elseIf($AccWIN_i \geq \alpha * AccWIN_{i-P}$)||($P' == P'_{max}$)
                    Then
19.                 $W' \leftarrow 0$;    //Release the adaptive window
20.                 $State \leftarrow 0$
21.             else
22.                 $W' \leftarrow W' \cup \{x_i\}$   //W' keep collecting new samples
23.             end
24.         end
25.         Repeat 12 – 24 for $state == 2$
26      return $AvgAcc$       //The average of the accuracy.
27. Function: HPO_BO-TPE ($S, Space, MaxTime$):
28.     $MaxAcc \leftarrow 0$,
29.     for $j \leftarrow 1$ to $MaxTime$ do
30.         $\alpha, \beta, P, P_{max} \leftarrow SelectConfiguration(space)$; //Search best
                                                        HP values by BO-TPE
31.         $Acc$
32.             ConceptDriftAdaptation($S, K, \alpha, \beta, P, P_{max}$);
33.         If $MaxAcc < Acc$ Then
34.             $MaxAcc \leftarrow Acc$;
35.             $HP_{opt} \leftarrow \alpha, \beta, P, P_{max}$  //Update accuracy & optimal
                                                  hyperparameter values.
36.         end
37.     end
38.     return $MaxAcc, HP_{opt}$   //The best accuracy & hyperparameters
```

**Algorithm 2** Bayesian-optimized adaptive sliding window (BOASWIN)

**Table 1** Attributes of the datasets

| Type | Dataset | Instance | Attributes | Class | Noise (%) | Class proportions (%) | Drift type |
|---|---|---|---|---|---|---|---|
| Synthetic | Agrawal_a (ARG_a) | 1,000,000 | 9 | 2 | 0 | 52.83/47.17 | Sudden |
| | Agrawal_g (ARG_g) | 1,000,000 | 9 | 2 | 0 | 52.83/47.17 | Gradual |
| | SEA_a | 40,000 | 3 | 2 | 10 | 50.14/49.86 | Sudden |
| | SEA_g | 41,000 | 3 | 2 | 10 | 50.11/49.89 | Gradual |
| | Hyperplane (HYP) | 200,000 | 10 | 2 | 0 | 50.03/49.97 | Sudden, Recurring |
| Real | Phishing (PHI) | 11,055 | 46 | 2 | – | 44.31/55.69 | Unknown |
| | Weather (WET) | 18,159 | 8 | 2 | – | 68.62/31.38 | Unknown |

*Agrawal generator* [47] has three categorical elements and six numeric attributes to describe the hypothetical loan applications. A perturbation factor for the numeric characteristics offsets the actual value and causes it to shift. It can provide ten functions to assess whether or not the loan should be granted. By altering the functions, the concept drift takes place.

*SEA generator* [48] is comprised of two classes, three numerical attributes produced at random, and noise for the third attribute. In the range [0,10], the numbers are created at random. Each instance is classified as class 1 if $f_1 + f_2 \leq \theta$, where $f_1$ and $f_2$ are the first two characteristics, and is a threshold that generates several contexts, has a value of 8, 9, 7, or 9.5.

*The weather dataset* consists of over 9000 weather stations worldwide and has provided data to the US National Oceanic and Atmospheric Administration. Records go back to the 1930s and offer various weather patterns. Temperature, pressure, wind speed, and other variables are measured every day, along with indications for precipitation and other weather-related phenomena. We used the Offutt Air Force Base in Bellevue, Nebraska, as a representative real-world dataset for this experiment because of its vast period of 50 years (1949–1999) and a variety of weather patterns that make it a long-term precipitation classification/prediction drift challenge [49].

*HyperPlane* In this data set, the ideas that have gradually changed are calculated using the formula $f(x) = \sum_{i=1}^{d-1} a_i * \left(\left(x_i + x_{i+1}\right)/x_i\right)$, where $d = 10$ is the dimension and $a_i$ is utilized to regulate the decision hyperplane [50].

*The phishing dataset*, distinguishing between dangerous and benign web pages, is taken from [51]. A typical classification problem was assumed to be represented by the digits dataset [51].

## 4.2 Results and discussion

BO automatically tunes the hyperparameters of the XGBoost and BOASWIN models to produce optimum versions. Table 2 displays the XGBoost and BOASWIN

**Table 2** Hyperparameters of XGBoost and BOASWIN

| Model | Hyperparameters | Search Range | Optimal values for datasets used | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | AGR_a | AGR_g | SEA_a | SEA_g | HYP | PHI | WET |
| XGBoost | *gamma* | [0,1] | 0.1673 | 0.9530 | 0.3600 | 0.8481 | 0.1494 | 0.4190 | 0.9121 |
| | *Learning rate* | [0,1] | 0.1168 | 0.0720 | 0.2318 | 0.1461 | 0.0758 | 0.2752 | 0.3442 |
| | *max- depth* | [5,50] | 6.0881 | 7.6416 | 4.7145 | 3.2596 | 7.0240 | 9.3369 | 3.8295 |
| | *n-estimators* | [50,500] | 101.70 | 106.82 | 113.87 | 114.65 | 107.99 | 110.54 | 114.90 |
| BOAS-WIN | $\alpha$ | (0.95,1) | 0.952 | 0.957 | 0.969 | 0.967 | 0.987 | 0.972 | 0.975 |
| | $\beta$ | (0.90,1) | 0.905 | 0.917 | 0.92 | 0.943 | 0.945 | 0.933 | 0.938 |
| | $P$ | [100,1000] | 800 | 300 | 650 | 1000 | 900 | 700 | 850 |
| | $P_{\max}$ | [500,50000] | 2100 | 1600 | 1500 | 1600 | 3400 | 1700 | 4800 |

**Table 3** Default and optimized performance accuracies of the XGBoost model

| Status | Datasets used (%) | | | | | | |
|---|---|---|---|---|---|---|---|
| | AGR_a | AGR_g | SEA_a | SEA_g | HYP | PHI | WET |
| Default | 70.13 | 70.00 | 75.57 | 75.57 | 74.08 | 91.92 | 77.71 |
| Optimized | 70.53 | 70.28 | 75.94 | 76.20 | 74.66 | 92.31 | 78.10 |

models' initial hyperparameter search range and discovered hyperparameter values for the seven datasets under consideration. After applying BO to create optimized models for spatiotemporal classifications, the proposed models were given the ideal hyperparameter values. Table 3 shows the default and optimized classification accuracy for the seven datasets used, in which the optimized were higher than the default.

### 4.2.1 Analysis of varying window size

As part of our goal to monitor the stream data distribution in the sliding windows, Table 4 shows the experiments carried out in this study with varying window sizes to see which sizes of these windows produced good results in the presence of concept drift. It was observed that moderate-sized windows produce good outputs regarding the classifier accuracy on the seven datasets used in this study. The bold values from Table 4 were the best parameters; hence, they were used to detect changes in the datasets used, and the results of the experiments are shown in Fig. 4.

### 4.2.2 Drift points

Drift points are those specific moments or data points where these changes become evident, often leading to the need for model adaptation or retraining. Identifying and monitoring drift points is crucial for maintaining model accuracy and effectiveness in applications that involve evolving data distributions. In Fig. 4, the dots indicate

**Table 4** Performance of the varying window sizes on the seven datasets

| Dataset | Varying windows size [win1, win2] | $\alpha$ | $\beta$ | Accuracy (%) |
|---|---|---|---|---|
| AGR_a | [250, 1100] | 0.955 | 0.958 | 69.83 |
| | [350, 2200] | 0.97 | 0.926 | 70.53 |
| | [400, 1800] | 0.976 | 0.975 | 70.79 |
| | [550, 1400] | 0.975 | 0.954 | 70.52 |
| | [550, 4800] | 0.979 | 0.972 | 70.50 |
| | [500, 3700] | 0.976 | 0.909 | 70.63 |
| | [400, 3300] | 0.974 | 0.947 | 70.58 |
| | **[800, 2100]** | **0.952** | **0.905** | **70.83** |
| | [350, 1800] | 0.987 | 0.932 | 70.06 |
| | [650, 1700] | 0.952 | 0.948 | 70.71 |
| AGR_g | [900, 3300] | 0.983 | 0.961 | 70.48 |
| | [200, 3700] | 0.974 | 0.931 | 70.03 |
| | [500, 4100] | 0.96 | 0.919 | 70.80 |
| | [600, 3200] | 0.963 | 0.912 | 70.10 |
| | **[300, 1600]** | **0.957** | **0.917** | **71.02** |
| | [1000, 2200] | 0.962 | 0.971 | 70.71 |
| | [850, 2500] | 0.963 | 0.972 | 70.34 |
| | [400, 2000] | 0.981 | 0.939 | 69.58 |
| | [250, 1800] | 0.973 | 0.907 | 70.23 |
| | [250, 1900] | 0.965 | 0.969 | 70.56 |
| SEA_a | [350, 1700] | 0.966 | 0.921 | 72.21 |
| | [300, 2300] | 0.982 | 0.923 | 74.73 |
| | [850, 2400] | 0.966 | 0.964 | 76.18 |
| | [250, 1700] | 0.956 | 0.946 | 70.98 |
| | [850, 2000] | 0.961 | 0.940 | 74.60 |
| | [350, 3900] | 0.952 | 0.969 | 71.27 |
| | [400, 4600] | 0.969 | 0.953 | 75.85 |
| | **[650, 1500]** | **0.969** | **0.92** | **76.76** |
| | [300, 1300] | 0.953 | 0.957 | 74.45 |
| | [500, 2300] | 0.974 | 0.967 | 74.35 |
| SEA_g | [350, 1900] | 0.958 | 0.903 | 71.99 |
| | [800, 1300] | 0.973 | 0.965 | 73.87 |
| | **[100, 1600]** | **0.967** | **0.943** | **76.96** |
| | [900, 4700] | 0.96 | 0.903 | 75.57 |
| | [350, 2300] | 0.969 | 0.947 | 75.29 |
| | [350, 3100] | 0.962 | 0.961 | 74.94 |
| | [300, 4400] | 0.957 | 0.91 | 74.76 |
| | [600, 2200] | 0.978 | 0.966 | 73.15 |
| | [300, 3600] | 0.989 | 0.968 | 71.93 |
| | [950, 3200] | 0.986 | 0.969 | 74.88 |

**Table 4** (continued)

| Dataset | Varying windows size [win1, win2] | $\alpha$ | $\beta$ | Accuracy (%) |
|---|---|---|---|---|
| HYP | [900, 2300] | 0.973 | 0.923 | 81.15 |
| | [700, 1800] | 0.972 | 0.933 | 76.54 |
| | [950, 3100] | 0.954 | 0.951 | 75.13 |
| | [450, 3600] | 0.952 | 0.927 | 75.04 |
| | [250, 3900] | 0.950 | 0.922 | 73.88 |
| | [750, 1700] | 0.982 | 0.901 | 77.95 |
| | **[900, 3500]** | **0.969** | **0.923** | **81.26** |
| | [700, 4200] | 0.968 | 0.952 | 75.17 |
| | [900, 3400] | 0.987 | 0.945 | 84.26 |
| | [250, 2900] | 0.96 | 0.973 | 76.51 |
| PHI | [300, 2700] | 0.979 | 0.935 | 94.62 |
| | [700, 1100] | 0.973 | 0.964 | 94.72 |
| | [550, 1600] | 0.961 | 0.958 | 94.48 |
| | [650, 4700] | 0.98 | 0.938 | 95.16 |
| | [500, 4300] | 0.986 | 0.952 | 94.46 |
| | **[700, 1700]** | **0.972** | **0.933** | **95.53** |
| | [900, 2700] | 0.967 | 0.955 | 94.87 |
| | [500, 2100] | 0.968 | 0.922 | 95.02 |
| | [550, 3800] | 0.954 | 0.943 | 95.08 |
| | [500, 1700] | 0.98 | 0.909 | 94.53 |
| WET | [950, 3300] | 0.964 | 0.941 | 78.01 |
| | [300, 4200] | 0.961 | 0.937 | 74.79 |
| | [700, 4900] | 0.981 | 0.977 | 75.72 |
| | [300, 2400] | 0.979 | 0.971 | 75.89 |
| | **[850, 4800]** | **0.975** | **0.938** | **78.35** |
| | [800, 4600] | 0.957 | 0.962 | 77.85 |
| | [350, 2600] | 0.958 | 0.958 | 76.24 |
| | [750, 3000] | 0.988 | 0.906 | 77.71 |
| | [500, 3400] | 0.975 | 0.902 | 78.01 |
| | [350, 2700] | 0.978 | 0.918 | 77.13 |

the change points in the datasets used. The blue lines representing our proposed model could track the point of drifts while maintaining a higher classification accuracy than the offline XGBoost model in red lines. The suggested accuracy of the BOASWIN + XGBoost model is compared in Table 5 to the cutting-edge drift adaptive techniques described in an earlier section.

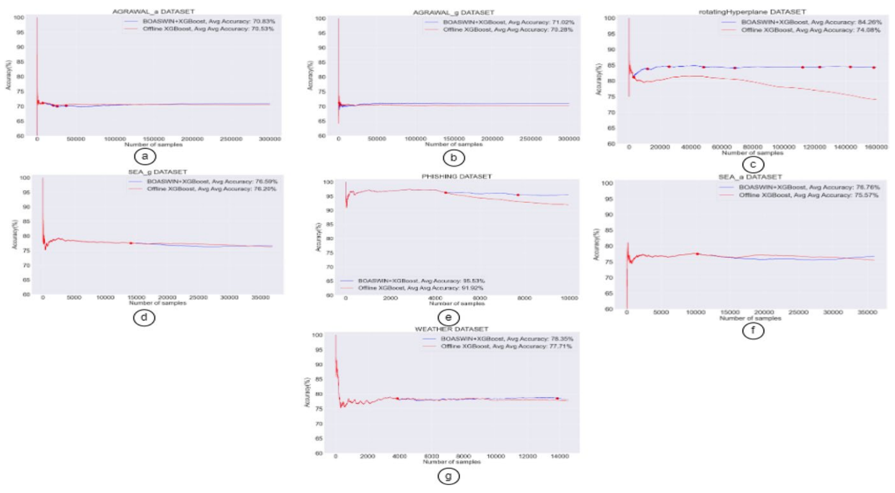

**Fig. 4** Drift points detection graphs

### 4.2.3 Analysis of AGRAWAL dataset

Table 5 shows that the suggested adaptive model performs better than all previous techniques regarding accuracy on the seven datasets used in this study. Bold values show the best outcomes for each dataset in Table 5.

The proposed technique, implemented on the AGRAWAL_a dataset and illustrated in Fig. 5, attained the most excellent accuracy of 70.83% among all implemented models by adjusting to the sudden concept drift found in the dataset. The offline XGBoost model's accuracy is 70.53% without drift adaption, which is slightly less accurate. The accuracy ratings of the other six cutting-edge methods are also less accurate than those of our proposed strategy. Additionally, as shown in Fig. 6, our suggested model (BOASWIN + XGBoost) indicated as "OURs" beat other state-of-the-art models in terms of precision, recall, and *f*1-score (69.61%, 67.87%, and 68.73%, respectively).

There is a gradual drift on the AGRAWAL_g dataset. As seen in Fig. 7 and Table 5, the suggested technique attained the best accuracy of 71.02% by responding to the gradual drift identified. In comparison, the offline XGBoost model's accuracy reduces significantly to only 70.28% without drift adaptation. This places a focus on the advancement of our proposed drift adaption technique. The proposed technique is substantially more accurate than the other six examined methods, OPA, SAM-KNN, SRP, ADWIN, DDM and EDDM, with accuracy values of 50.32%, 53.71%, 67.15%, 67.78%, 67.18%, and 67.03% respectively. Figure 8 depicts our proposed model's precision, recall, and *f*1-score comparison with the models experimented with in this study. However, the proposed model was best in precision and f1-score with values of 69.47% and 69.00%, respectively, while the XGBoost model had the highest recall value of 69.83%.

The success of "BOASWIN + XGBoost" on the AGRAWAL datasets can be attributed to the synergistic combination of Bayesian optimization (BOASWIN)

**Table 5** Comparison of the effectiveness of drift adaption techniques

| Model | Accuracy (%) | Precision (%) | Recall (%) | $F1$-score (%) | Average time (min) |
|---|---|---|---|---|---|
| *AGRAWAL_a Dataset* | | | | | |
| OPA | 50.34 | 47.41 | 47.08 | 49.94 | **0.36** |
| SAM-KNN | 53.56 | 50.86 | 49.06 | 49.94 | 2.67 |
| SRP | 65.01 | 63.68 | 60.30 | 61.95 | 47.07 |
| ADWIN | 67.62 | 65.94 | 65.02 | 65.48 | 37.7 |
| DDM | 66.87 | 64.89 | 65.04 | 64.97 | 48.22 |
| EDDM | 67.00 | 65.23 | 64.54 | 64.88 | 53.10 |
| XGBoost | 70.53 | 69.60 | 66.77 | 68.15 | 40.00 |
| BOASWIN + XGBoost | **70.83** | **69.61** | **67.87** | **68.73** | 40.03 |
| *AGRAWAL_g Dataset* | | | | | |
| OPA | 50.32 | 47.21 | 46.89 | 47.05 | **0.36** |
| SAM-KNN | 53.71 | 50.87 | 48.92 | 49.87 | 3.13 |
| SRP | 67.15 | 66.45 | 61.05 | 63.63 | 59.02 |
| ADWIN | 67.78 | 66.08 | 64.83 | 65.45 | 58.78 |
| DDM | 67.18 | 65.44 | 64.17 | 64.80 | 61.15 |
| EDDM | 67.03 | 64.98 | 64.98 | 64.98 | 63.06 |
| XGBoost | 70.28 | 67.92 | **69.83** | 68.87 | 60.88 |
| BOASWIN + XGBoost | **71.02** | **69.47** | 68.54 | **69.00** | 65.43 |
| *SEA_a Dataset* | | | | | |
| OPA | 58.44 | 59.23 | 53.06 | 55.97 | **0.02** |
| SAM-KNN | 70.24 | 70.84 | 68.42 | 69.61 | 0.12 |
| SRP | 73.93 | 73.58 | 74.35 | 73.96 | 0.55 |
| ADWIN | 76.26 | 75.82 | 76.83 | 76.32 | 0.29 |
| DDM | 76.20 | 75.53 | 77.21 | 76.36 | 0.32 |
| EDDM | 76.07 | 75.29 | 77.33 | 76.30 | 0.38 |
| XGBoost | 75.94 | 73.84 | **80.05** | **76.82** | 8.63 |
| BOASWIN + XGBoost | **76.76** | **76.81** | 76.40 | 76.60 | 5.83 |
| *SEA_g Dataset* | | | | | |
| OPA | 58.55 | 59.37 | 53.31 | 56.18 | **0.02** |
| SAM-KNN | 70.20 | 70.81 | 68.39 | 69.58 | 0.14 |
| SRP | 73.58 | 73.08 | 74.39 | 73.73 | 0.57 |
| ADWIN | 76.17 | 75.51 | 77.24 | 73.73 | 0.27 |
| DDM | 76.23 | 75.72 | 76.98 | 76.35 | 0.33 |
| EDDM | 76.27 | 75.58 | 77.37 | 76.46 | 0.36 |
| XGBoost | 76.20 | 74.58 | **79.25** | 76.84 | 1.32 |
| BOASWIN + XGBoost | **76.96** | **76.93** | 76.80 | **76.86** | 7.22 |
| *HYPERPLANE Dataset* | | | | | |
| OPA | 81.95 | 82.30 | 81.47 | 81.88 | **0.07** |
| SAM-KNN | 75.59 | 75.56 | 75.72 | 75.64 | 1.42 |
| SRP | 76.62 | 76.55 | 76.82 | 76.69 | 7.58 |
| ADWIN | 79.59 | 79.65 | 79.55 | 79.60 | 2.78 |
| DDM | 78.26 | 78.03 | 78.74 | 78.38 | 4.28 |

**Table 5** (continued)

| Model | Accuracy (%) | Precision (%) | Recall (%) | *F*1-score (%) | Average time (min) |
|---|---|---|---|---|---|
| EDDM | 77.44 | 77.36 | 77.64 | 77.50 | 5.37 |
| XGBoost | 74.66 | 74.91 | 74.25 | 74.58 | 0.7 |
| BOASWIN + XGBoost | **84.26** | **84.02** | **84.64** | **84.33** | 25.73 |
| *PHISHING Dataset* | | | | | |
| OPA | 92.54 | 93.10 | 93.54 | 93.32 | **0.007** |
| SAM-KNN | 90.78 | 90.52 | 93.22 | 91.85 | 0.22 |
| SRP | 92.73 | 93.08 | 93.94 | 93.51 | 0.60 |
| ADWIN | 92.88 | 92.92 | 94.43 | 93.66 | 0.10 |
| DDM | 92.49 | 93.39 | 93.11 | 93.25 | 0.14 |
| EDDM | 92.31 | 91.45 | 95.09 | 93.23 | 0.13 |
| XGBoost | 92.31 | 90.60 | 96.18 | 93.31 | 2.25 |
| BOASWIN + XGBoost | **95.53** | **94.99** | **97.10** | **96.03** | 1.05 |
| *WEATHER Dataset* | | | | | |
| OPA | 67.88 | 49.71 | 49.46 | 49.58 | **0.004** |
| SAM-KNN | 74.69 | 62.51 | 51.85 | 56.69 | 0.11 |
| SRP | 75.33 | 64.18 | 51.51 | 57.15 | 0.40 |
| ADWIN | 75.09 | 63.94 | 50.45 | 56.40 | 0.18 |
| DDM | 75.58 | 64.94 | 51.14 | 57.22 | 0.13 |
| EDDM | 71.73 | 57.53 | 43.88 | 49.79 | 0.12 |
| XGBoost | 78.10 | **73.69** | 48.90 | 58.79 | 0.53 |
| BOASWIN + XGBoost | **78.35** | 68.93 | **58.66** | **63.38** | 2.32 |

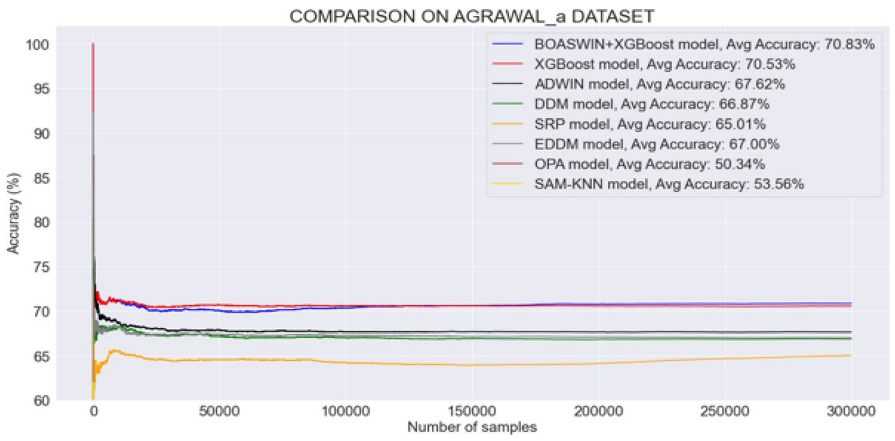

**Fig. 5** Comparison of the accuracy of the AGR_a dataset using different drift adaption techniques

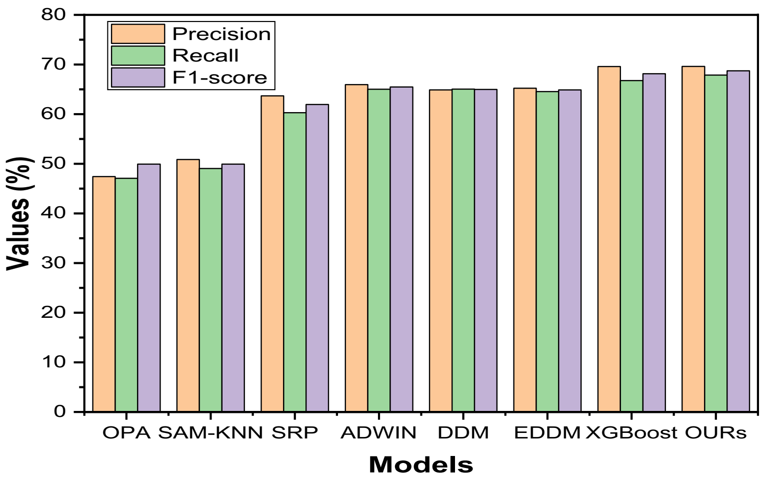

**Fig. 6** Comparison of the proposed model's precision, recall, and *f*1-score and other state-of-the-art models on the Agrawal_a dataset

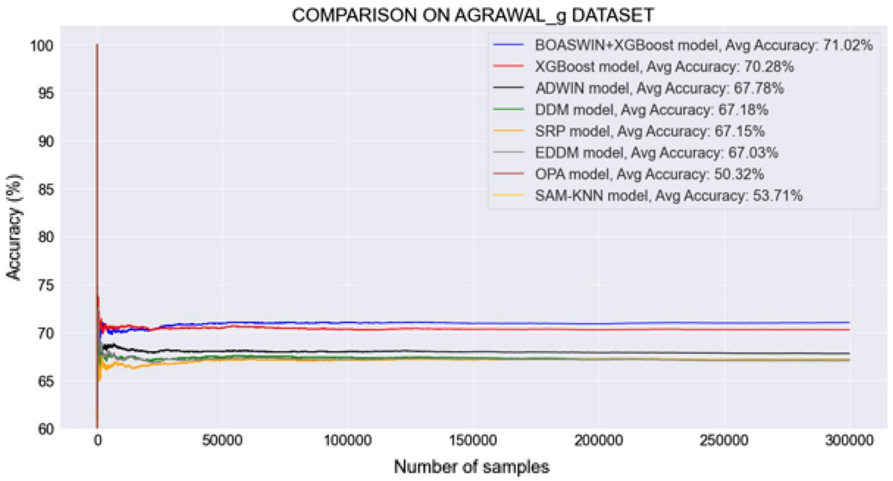

**Fig. 7** Comparison of the accuracy of the AGRAWAL_g dataset using different drift adaption techniques

and the XGBoost model. While other methods struggle to adapt to concept drift adequately, the proposed approach optimizes model hyperparameters dynamically, ensuring robust performance even in changing data distributions.

### 4.2.4 Analysis of the SEA datasets

The suggested BOASWIN + XGBoost model's accuracy is compared on the SEA_a dataset, which contained sudden drift, and on the SEA_g dataset, which

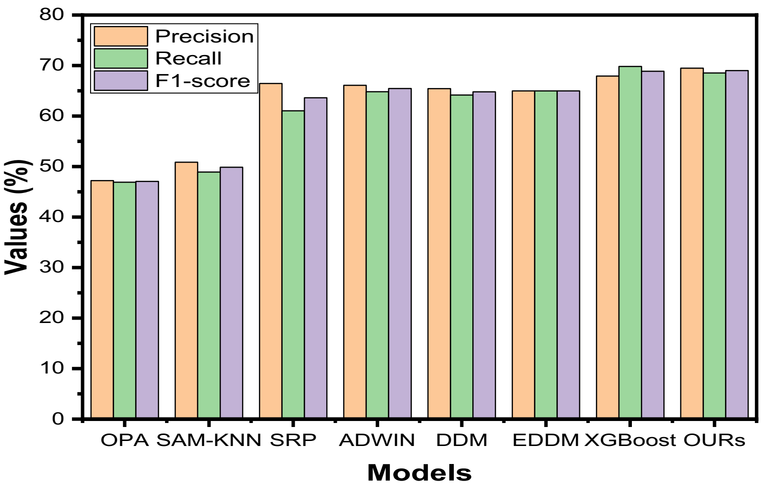

**Fig. 8** Comparison of the proposed model's precision, recall, and *f*1-score and other state-of-the-art models on the Agrawal_g dataset

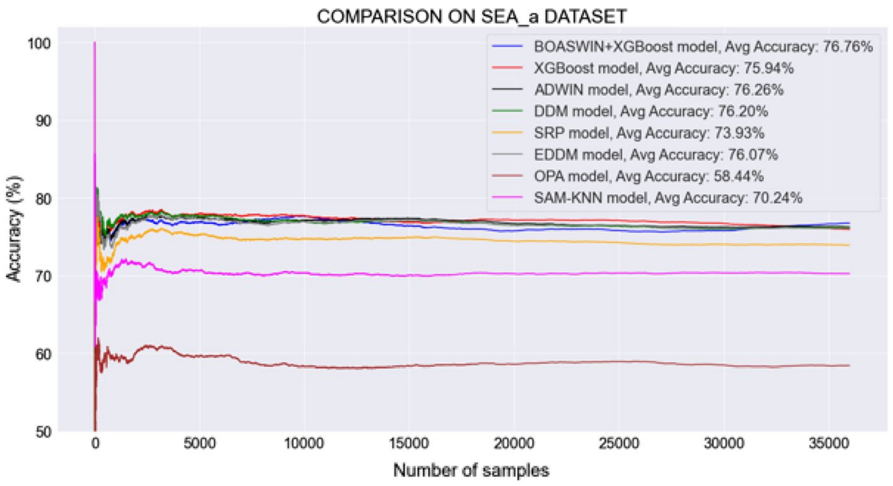

**Fig. 9** Comparison of the accuracy of the SEA_a dataset using different drift adaption techniques

contained gradual drift. Figure 9 compares the proposed model outperforming the other models by adjusting to the sudden concept drift found in the dataset. Figure 10 compares the proposed model outperforming the others by adjusting to gradual drift in the SEA_g dataset. The proposed model attained the most remarkable accuracy of 76.76% on the SEA_a dataset in Fig. 9 and 76.96% on the SEA_g dataset in Fig. 10 among all implemented models. Figure 11 depicts our proposed model's precision, recall, and *f*1-score comparison with the models

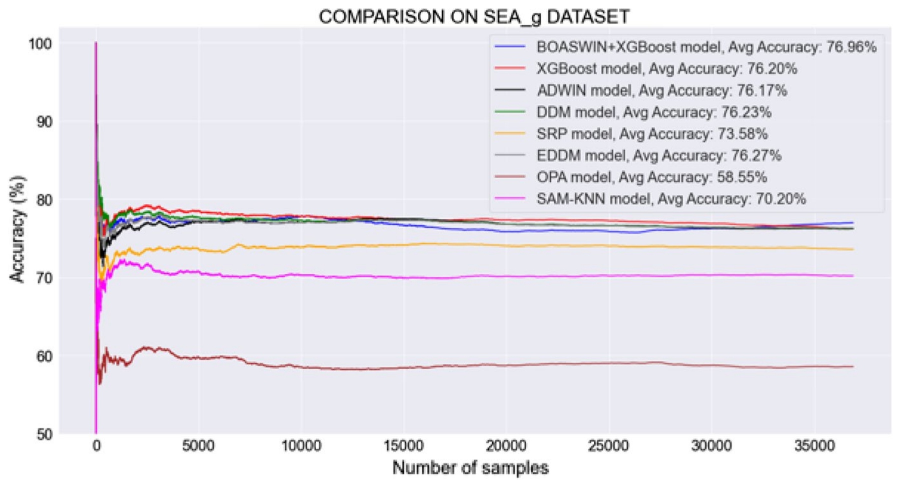

**Fig. 10** Comparison of the accuracy of the SEA_g dataset using different drift adaption techniques

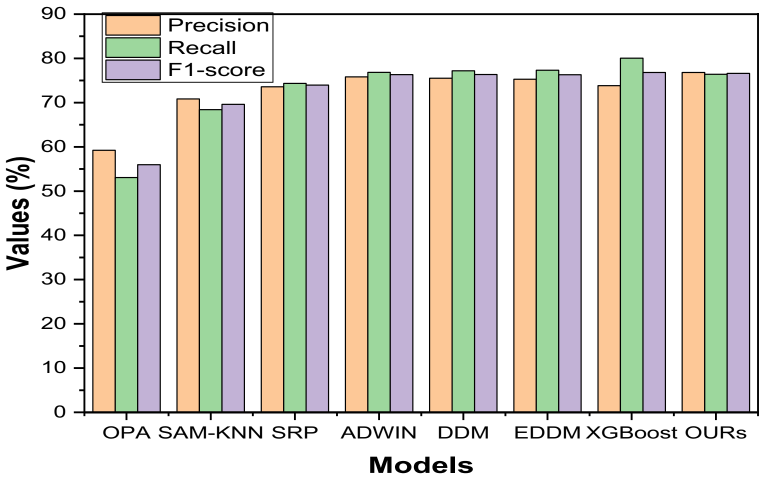

**Fig. 11** Comparison of the proposed model's precision, recall, and $f1$-score and other state-of-the-art models on the SEA_a dataset

experimented with in this study. However, the proposed model was best in precision with a value of 76.81%, while the XGBoost model had the highest values of recall and $f1$-score of 80.05% and 76.82%, respectively.

Figure 12 depicts our proposed model's precision, recall, and $f1$-score comparison with the other models experimented with in this study. The proposed BOASWIN + XGBoost model outperformed all the other models concerning precision and $f1$-score with the highest values of 76.93% and 76.86%, respectively, while the XGBoost model has the best recall value of 79.25%.

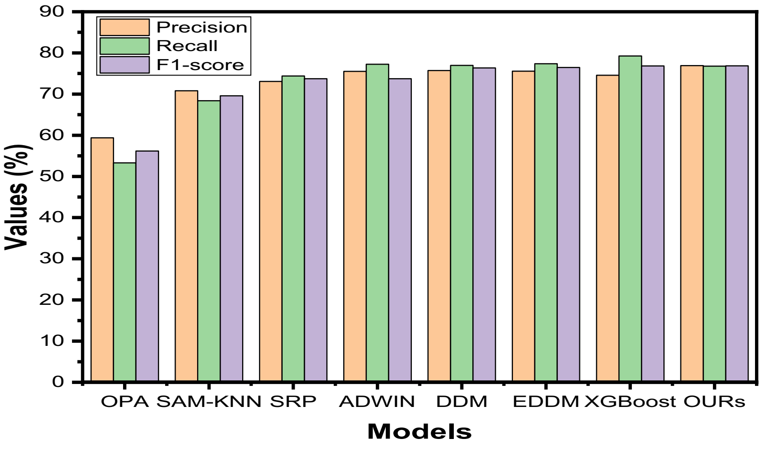

**Fig. 12** Comparison of the proposed model's precision, recall, and *f*1-score and other state-of-the-art models on the SEA_g dataset

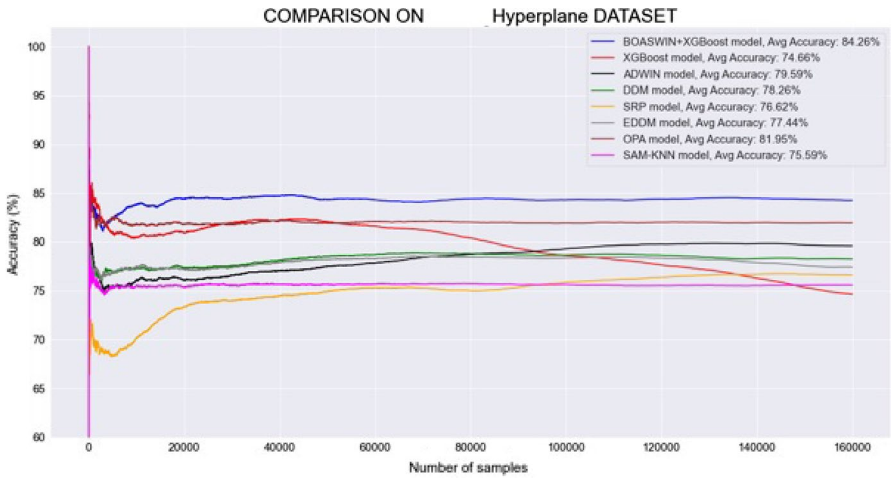

**Fig. 13** Comparison of the accuracy of the HYP dataset using different drift adaption techniques

### 4.2.5 Analysis of the HYPERPLANE dataset

A significant drift at the start of the HYP dataset test set contained sudden and reoccurring concept drift. As seen in Fig. 13, the suggested technique attained the best accuracy of 84.26% by responding to both the sudden and reoccurring drifts identified. In comparison, the offline XGBoost model's accuracy reduces significantly to only 74.66% without drift adaptation. The proposed technique is substantially more accurate than the other six examined methods, OPA, SAM-KNN, SRP, ADWIN, DDM and EDDM, with accuracy values of 81.95%, 75.59%, 76.62%, 79.59%, 78.26%, and 77.44% respectively. Figure 14 compares the precision, recall,

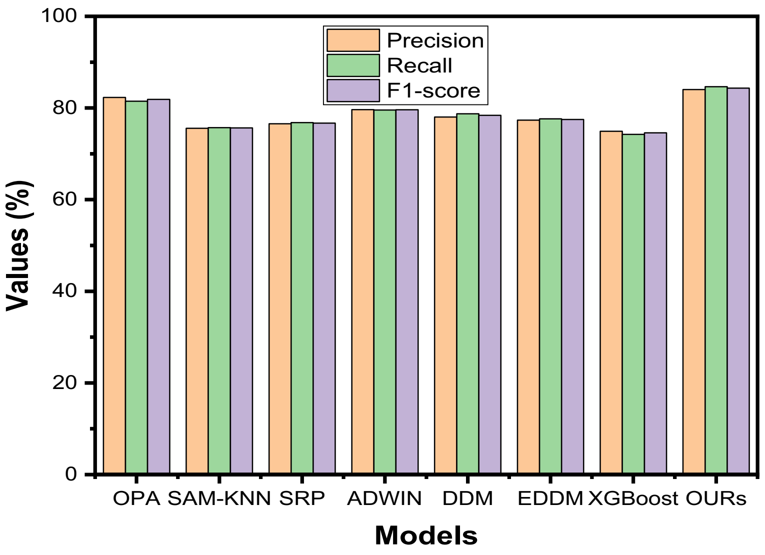

**Fig. 14** Comparison of the proposed model's precision, recall, and *f*1-score and other state-of-the-art models on the HYP dataset

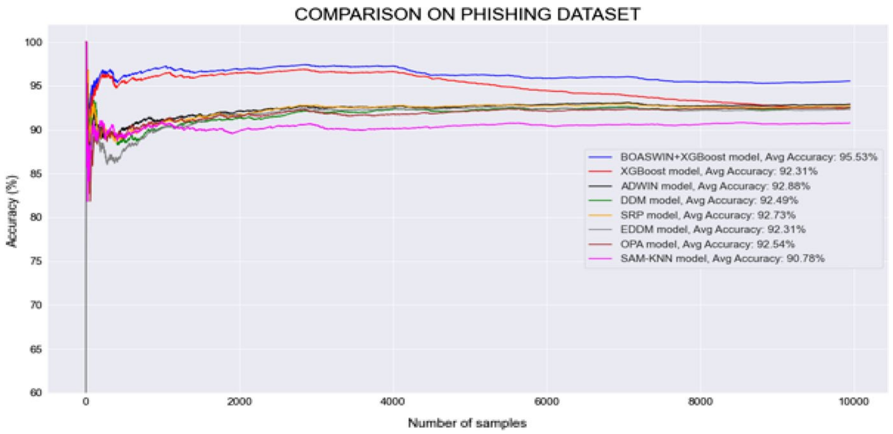

**Fig. 15** Comparison of the accuracy of the PHI dataset using different drift adaption techniques

and *f*1-score of our proposed model on the HYP dataset with the other models in this study. The proposed BOASWIN + XGBoost model outperformed all the other models concerning precision, recall, and f1-score with the highest values of 84.02%, 84.64%, and 84.33%, respectively.

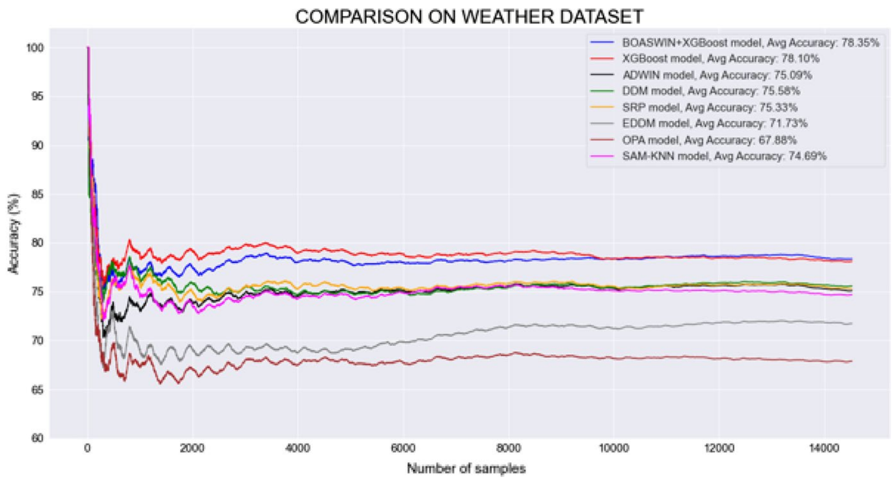

**Fig. 16** Comparison of the accuracy of the WET dataset using different drift adaption techniques

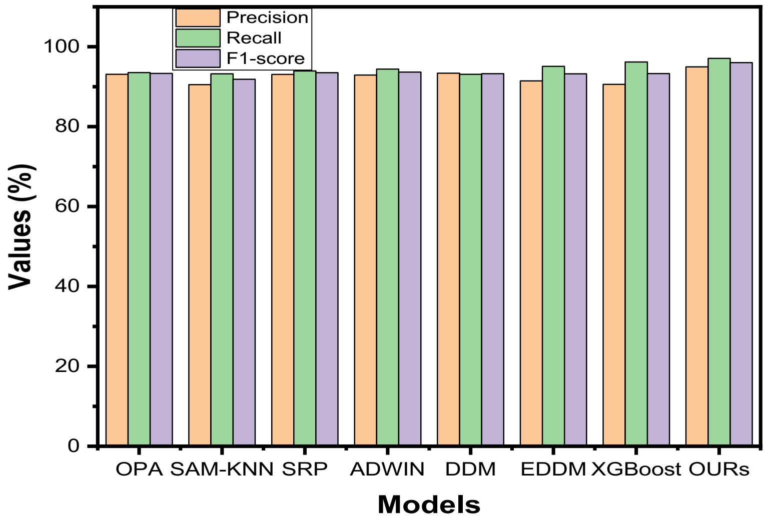

**Fig. 17** Comparison of the proposed model's precision, recall, and *f*1-score and other state-of-the-art models on the PHI dataset

### 4.2.6 Analysis of PHISHING and WEATHER datasets

The proposed BOASWIN + XGBoost model's accuracy is evaluated using the real-world datasets PHI and WET, which contain drifts. Using the PHI data set, Fig. 15 compares the performance of the suggested model with that of the competing models. Figure 16 compares the performance of the proposed model with

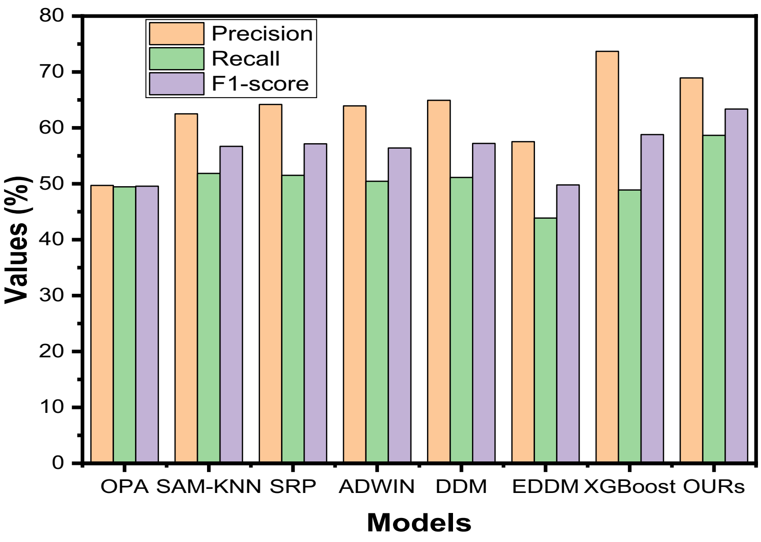

**Fig. 18** Comparison of the proposed model's precision, recall, and *f*1-score and other state-of-the-art models on the WET dataset

that of the other models using the WET data set. The proposed model outperformed the other models by responding to changes detected in the PHI data set with an accuracy of 95.53%. The proposed model's accuracy score of 78.35% was the highest for the WET data set. Figure 17 compares the precision, recall, and *f*1-score of our suggested model on the PHI data set with the other models tested in this work. The proposed BOASWIN + XGBoost model outperformed all the other models concerning precision, recall, and *f*1-score with the highest values of 94.99%, 97.10%, and 96.03%, respectively.

Figure 18 compares our suggested model's precision, recall, and *f*1-score with the models tested in this study on the WET data set. However, the XGBoost model had the highest precision value of 73.69%, while the proposed model had the best recall and *f*1-score with values of 58.66% and 63.38%, respectively.

The results obtained from the proposed "BOASWIN + XGBoost" model exhibit significant implications for both false positives and false negatives in classification tasks. These implications stem from the model's performance in key metrics such as precision, recall, *F*1-score, and accuracy, directly influencing its ability to handle concept drift effectively.

Firstly, let's consider the effect of these results on false positives (Type I Errors). Precision, a critical metric, represents the ratio of true positives to the total predicted positives. When "BOASWIN + XGBoost" achieves higher precision than other models, it implies that the model correctly classifies positive instances while minimizing false positives. This outcome is paramount when false positives can have substantial consequences, such as in medical diagnosis or fraud detection. The model's superior precision suggests that it reduces the risk of falsely flagging instances as positive when they are, in fact, harmful.

On the other hand, the results also have a significant impact on false negatives (Type II Errors). Recall, another essential metric, quantifies the ratio of true positives to the total actual positives. When "BOASWIN + XGBoost" achieves higher recall, the model is proficient at capturing actual positive instances while reducing false negatives. In practical terms, the model is less likely to miss positive cases, leading to a lower rate of false negatives. This characteristic is especially critical in applications where missing positive instances can have severe consequences, such as in medical screenings or cybersecurity, where failing to detect diseases or security breaches can be detrimental.

Moreover, the $F1$-score, a metric that balances precision and recall, plays a pivotal role. A higher $F1$-score achieved by "BOASWIN + XGBoost" suggests an effective trade-off between reducing false positives and false negatives. This balance is crucial in real-world scenarios where both types of errors can have significant implications. The model's ability to maintain high precision and recall implies that it can adapt to concept drift without disproportionately increasing either false positives or false negatives, making it an invaluable choice for applications where balanced performance is paramount.

In conclusion, the performance results of "BOASWIN + XGBoost" on precision, recall, $F1$-score, and accuracy collectively indicate its capability to achieve a harmonious equilibrium between minimizing false positives and false negatives. This equilibrium is essential in diverse real-world settings where the consequences of classification errors can vary widely. The model's ability to maintain this balance while handling concept drift positions it as a robust and adaptable solution for applications demanding accurate and well-balanced classifications.

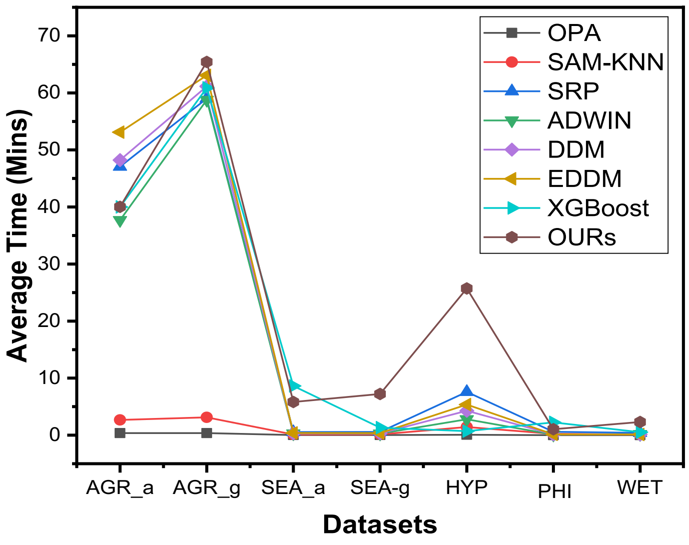

**Fig. 19** Comparison of the average execution time of all the models used on the seven experimented datasets

### 4.2.7 Analysis of the average time

As shown in Fig. 19 and Table 5, the suggested method for real-time learning is evaluated by calculating the average prediction time for each occurrence while considering time in spatiotemporal systems. OPA, SAM-KNN, SRP, ADWIN, DDM, and EDDM all have prediction times that are less than the suggested model, but their accuracy is substantially worse. Regarding the trade-off between accuracy and efficiency, the proposed method continues to outperform the methods in the presence of concept drift. The experimental findings demonstrate the potency and reliability of the suggested BOASWIN + XGBoost model for spatiotemporal streaming data analytics.

In the AGR_a and AGR_g datasets, most models exhibit relatively short processing times, ranging from 0.36 to 65.43 min. While OPA operates swiftly, SRP, DDM, and EDDM require the most extensive processing durations. However, BOASWIN-XGBoost stands out for its relatively longer processing times in these datasets, with 40.03 and 65.43 min, respectively.

Across the SEA_a and SEA_g datasets, OPA, SAM-KNN, and ADWIN consistently demonstrate minimal processing times within the 0.02–0.33 min range. Here, too, BOASWIN-XGBoost exhibits longer processing times, which can be attributed to its intricate algorithm and comprehensive approach to handling concept drift.

However, XGBoost and BOASWIN-XGBoost stand out for their relatively longer processing times, especially in the case of SEA_a, where XGBoost's duration is notably higher. OPA is the quickest in the HYPERPLANE dataset, taking only 0.07 min. In contrast, BOASWIN-XGBoost significantly extends the processing time to 25.73 min, suggesting it may not be ideal for real-time applications within this dataset. This extended processing time for BOASWIN-XGBoost can be attributed to the complexity of its algorithm, which likely involves advanced techniques to maintain high predictive accuracy in the face of concept drift.

In the PHISHING dataset, OPA boasts the fastest processing time at 0.007 min, while both XGBoost and BOASWIN-XGBoost require more extensive processing periods, with XGBoost notably exceeding the duration of OPA. Again, this longer processing time for BOASWIN-XGBoost reflects its thorough approach to concept drift handling.

Lastly, within the WEATHER dataset, OPA maintains its reputation as the swiftest, with a mere 0.004 min. BOASWIN-XGBoost necessitates more processing time but remains within reasonable limits at 2.32 min. The extended processing time for BOASWIN-XGBoost in various datasets can be attributed to its algorithm's complexity and the thoroughness with which it tackles concept drift, resulting in higher predictive accuracy but longer processing durations.

BOASWIN-XGBoost appears to excel for several reasons in the context of drift adaption techniques. First and foremost, it is crucial to recognize that its average time consumption does not solely determine the effectiveness of a drift adaption technique. Instead, it balances time efficiency and maintaining high predictive accuracy in the face of concept drift. BOASWIN-XGBoost appears to strike this balance effectively, as it consistently achieves competitive or even superior performance compared to other techniques.

One key factor contributing to the strong performance of BOASWIN-XGBoost is its adaptability. Concept drift, which occurs when the underlying data distribution changes over time, is a common challenge in many machine learning applications. BOASWIN-XGBoost possesses a robust mechanism for detecting and adapting to these changes efficiently, reflected in its high accuracy, precision, recall, and $F1$-score across diverse datasets.

In conclusion, the BOASWIN + XGBoost model's suitability in real-time or resource-constrained scenarios depends on the task's context and requirements. While it can be computationally expensive, its accuracy benefits should be balanced against available resources and decision urgency. Careful model selection, deployment optimizations, and hardware choices can make it viable in various applications.

## 5 Conclusion

In this research endeavor, we have delved into handling concept drift within non-stationary spatiotemporal data streams. This challenge has grown exponentially in significance with the proliferation of data-rich environments. Our novel approach, BOASWIN, which marries an adaptive XGBoost-based model with the BO-TPE hyperparameter optimization strategy, has emerged as a potent tool for spatiotemporal data analytics. The outcomes of our extensive experimentation, involving seven diverse datasets (AGR_a, AGR_g, SEA_a, SEA_g, HYP, PHI, and WET), have yielded insights of paramount importance. One of the paramount findings of our research is the remarkable and consistent superiority of our model's classification performance over a spectrum of state-of-the-art drift adaptation techniques. On dataset AGR_a, BOASWIN + XGBoost achieved an accuracy rate of 70.83%.

Similarly, on dataset AGR_g, the model demonstrated an accuracy rate of 71.02%. This trend of outperforming other techniques was maintained across datasets SEA_a (76.76%), SEA_g (76.96%), HYP (84.26%), PHI (95.5%), and WET (78.35%). These results underscore the model's effectiveness in maintaining high classification accuracy rates across all datasets examined. The adaptability of BOASWIN + XGBoost, which enables it to respond autonomously to evolving data patterns, emerges as a critical asset in this research. Not only does it enhance classification accuracy, but it also ensures that models remain pertinent in scenarios where data distributions undergo continuous and unpredictable changes. This adaptability is a testament to the model's practicality in real-world applications where dynamic data streams are the norm. The implications of our research extend far beyond academia, carrying profound significance for a wide array of practical applications. Fields such as environmental monitoring, urban planning, and disaster management stand to gain immensely from the availability of reliable and adaptive classification models. Our work represents a significant step in ensuring these domains can make informed decisions despite rapidly changing spatiotemporal data. As we chart our course into the future of spatiotemporal data analytics, we anticipate that our findings and limitations, as presented in section one, will catalyze the development of more resilient and effective solutions in handling concept drift within spatiotemporal data streams, thereby benefiting a multitude of applications and domains.

**Author contributions** AA wrote the manuscript and did all the analysis, while HYC supervised the study, and both went through the manuscript.

**Funding** No funding.

**Availability of data and materials** The datasets generated during and analyzed during the current study are available from the corresponding author upon reasonable request.

## Declarations

**Conflict of interest** No conflict of interest.

## References

1. Angbera A, Chan HY (2022) A novel true-real-time spatiotemporal data stream processing framework. Jordan J Comput Inf Technol (JJCIT) 8(3):256–270
2. Tanha J, Samadi N, Abdi Y, Razzaghi-asl N (2022) CPSSDS: conformal prediction for semi-supervised classification on data streams. Inf Sci 584:212–234. https://doi.org/10.1016/j.ins.2021.10.068
3. Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2019) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31(12):2346–2363. https://doi.org/10.1109/TKDE.2018.2876857
4. Liu W, Zhang H, Ding Z, Liu Q, Zhu C (2021) A comprehensive active learning method for multiclass imbalanced data streams with concept drift. Knowl-Based Syst 215:106778. https://doi.org/10.1016/j.knosys.2021.106778
5. Yang L, Cheung Y-M, YanTang Y (2020) Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift. IEEE Trans Neural Netw Learn Syst 31(8):2764–2778. https://doi.org/10.1109/TNNLS.2019.2951814
6. Suárez-Cetrulo AL, Quintana D, Cervantes A (2022) A survey on machine learning for recurring concept drifting data streams. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2022.118934
7. Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2013) A survey on concept drift adaptation. ACM Comput Surv 1(1):35. https://doi.org/10.1145/0000000.0000000
8. Priya S, Uthra RA (2020) Comprehensive analysis for class imbalance data with concept drift using ensemble based classification. J Ambient Intell Humaniz Comput 12(5):4943–4956. https://doi.org/10.1007/s12652-020-01934-y
9. Liu A, Lu J, Liu F, Zhang G (2018) Accumulating regional density dissimilarity for concept drift detection in data streams. Pattern Recogn 76:256–272. https://doi.org/10.1016/j.patcog.2017.11.009
10. Liao G et al (2022) A novel semi-supervised classification approach for evolving data streams. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2022.119273
11. Liu A, Lu J, Zhang G (2021) Concept drift detection via equal intensity k-means space partitioning. IEEE Trans Cybern 51(6):3198–3211. https://doi.org/10.1109/TCYB.2020.2983962
12. Bifet A, Gavaldà R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the 7th SIAM International Conference on Data Mining, pp 443–448. https://doi.org/10.1137/1.9781611972771.42
13. Santos SGTC, Barros RSM, Gonçalves PM (2019) A differential evolution based method for tuning concept drift detectors in data streams. Inf Sci J 485:376–393
14. Raab C, Heusinger M, Schleif FM (2020) Reactive soft prototype computing for concept drift streams. Neurocomputing 416:340–351. https://doi.org/10.1016/j.neucom.2019.11.111

15. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 3171, pp 286–295. https://doi.org/10.1007/978-3-540-28645-5_29
16. Yang L, Manias DM, Shami A (2021) PWPAE: an ensemble framework for concept drift adaptation in IoT data streams. In: 2021 IEEE Global Communications Conference, GLOBECOM 2021—Proceedings, pp 1–6. https://doi.org/10.1109/GLOBECOM46510.2021.9685338
17. Wares S, Isaacs J, Elyan E (2019) Data stream mining: methods and challenges for handling concept drift. SN Appl Sci 1(11):1–19. https://doi.org/10.1007/s42452-019-1433-0
18. Baena-Garcia M, Del Campo-Avila J, Fidalgo R, Bifet A, Gavalda R, Morales-bueno R (2006) Early drift detection method. In: 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams, vol 6, pp 77–86
19. Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y (2006) Online passive-aggressive algorithms. J Mach Learn Res 7:551–585
20. Losing V, Hammer B, Wersing H (2017) Self-adjusting memory: how to deal with diverse drift types. In: IJCAI International Joint Conference on Artificial Intelligence, October, pp 4899–4903. https://doi.org/10.24963/ijcai.2017/690
21. Zhang Y, Chu G, Li P, Hu X, Wu X (2017) Three-layer concept drifting detection in text data streams. Neurocomputing 260:393–403. https://doi.org/10.1016/j.neucom.2017.04.047
22. Zyblewski P, Sabourin R, Woźniak M (2021) Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. Inf Fus 66:138–154. https://doi.org/10.1016/j.inffus.2020.09.004
23. Guo H, Zhang S, Wang W (2021) Selective ensemble-based online adaptive deep neural networks for streaming data with concept drift. Neural Netw 142:437–456. https://doi.org/10.1016/j.neunet.2021.06.027
24. UdDin S, Shao J, Kumar J, Ali W, Liu J, Ye Y (2020) Online reliable semi-supervised learning on evolving data streams. Inf Sci 525:153–171. https://doi.org/10.1016/j.ins.2020.03.052
25. Goel K, Batra S (2022) Dynamically adaptive and diverse dual ensemble learning approach for handling concept drift in data streams. Comput Intell 38(2):463–505
26. Yang L, Moubayed A, Hamieh I, Shami A (2019) Tree-based intelligent intrusion detection system in internet of vehicles. In: 2019 IEEE Global Communications Conference, GLOBECOM 2019—Proceedings, no. Ml. https://doi.org/10.1109/GLOBECOM38437.2019.9013892
27. Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, vol 13–17-August, pp 785–794
28. Jiang Y, Tong G, Yin H, Xiong N (2019) A pedestrian detection method based on genetic algorithm for optimise XGBoost training parameters. IEEE Access 7:118310–118321. https://doi.org/10.1109/ACCESS.2019.2936454
29. Chen Z, Jiang F, ChengY, Gu X, Liu W, Peng J (2018) XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud. In: Proceedings—2018 IEEE International Conference on Big Data and Smart Computing, BigComp 2018, pp 251–256. https://doi.org/10.1109/BigComp.2018.00044
30. Smith LN (2018) A disciplined approach to neural network hyper-parameters: part 1—learning rate, batch size, momentum, and weight decay, pp 1–21. http://arxiv.org/abs/1803.09820
31. Ruder S (2016) An overview of gradient descent optimization algorithms, pp 1–14. http://arxiv.org/abs/1609.04747
32. Prechelt L (1998) Early stopping—but when? Early stopping is not quite as simple, pp 55–69
33. Kingma DP, Ba JL (2015) Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, pp 1–15
34. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH (2019) Hyperparameter optimization for machine learning models based on Bayesian optimization. J Electron Sci Technol 17(1):26–40. https://doi.org/10.11989/JEST.1674-862X.80904120
35. Caruana R, Ksikes A, Crew G (2014) Ensemble selection from libraries of models. In: Proceedings of the Twenty-First International Conference on Machine Learning. https://doi.org/10.1145/1015330.1015432
36. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst 4:2951–2959
37. El Shawi R, Sakr S (2020) Automated machine learning: techniques and frameworks. In: Kutsche R-D, Zimanyi E (eds) Big Data Management and Analytics. Springer, Cham, pp 40–69

38. Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316. https://doi.org/10.1016/j.neucom.2020.07.061
39. Seeger M (2004) Gaussian processes for machine learning University of California at Berkeley. Int J Neural Syst 14:69–109
40. Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, vol 24. https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
41. Injadat M, Moubayed A, Nassif AB, Shami A (2021) Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans Netw Serv Manag 18(2):1803–1816. https://doi.org/10.1109/TNSM.2020.3014929
42. Yang L, Shami A (2022) IoT data analytics in dynamic environments: from an automated machine learning perspective. Eng Appl Artif Intell. https://doi.org/10.1016/j.engappai.2022.105366
43. Sun Y, Wang Z, Liu H, Du C, Yuan J (2016) Online ensemble using adaptive windowing for data streams with concept drift. Int J Distrib Sensor Netw. https://doi.org/10.1155/2016/4218973
44. Yang L, Shami A (2021) A lightweight concept drift detection and adaptation framework for IoT data streams. IEEE Internet Things Mag 4(2):96–101. https://doi.org/10.1109/iotm.0001.2100012
45. Montiel J et al (2021) River: machine learning for streaming data in python. J Mach Learn Res 22:1–8
46. López Lobo J (2020) Synthetic datasets for concept drift detection purposes. Harvard Dataverse. https://doi.org/10.7910/DVN/5OWRGB
47. Agrawal R, Swami A, Imielinski T (1993) Database mining: a performance perspective. IEEE Trans Knowl Data Eng 5(6):914–925. https://doi.org/10.1109/69.250074
48. Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R (2009) New ensemble methods for evolving data streams. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 139–147. https://doi.org/10.1145/1557019.1557041
49. Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw 22(10):1517–1531. https://doi.org/10.1109/TNN.2011.2160459
50. Zhu X (2010) Stream Data Mining Repository. http://www.cse.fau.edu/~xqzhu/stream.html
51. Dua D, Graff C (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.