IJCI V10-2 (2023) 16–27
International Journal of Computers and Information (IJCI)
Available online at *https://ijci.journals.ekb.eg/*

Historical Isolated Forest for detecting and adaptation concept drifts in nonstationary data streaming

Ahmed H. Madkour*, Amgad M. Mohammed, Hatem M. Abdelkader

Information Systems Department, Faculty of Computers and Information, Menoufia University, Shebin Elkom, 32511, Egypt
hamdymadkour@ci.menofia.edu.eg, amgad.elsayed@ci.menofia.edu.eg

Abstract

Concept drift refers to sudden changes in the fundamental structure of the streaming data distribution over time. The core objective of concept drift research is to develop techniques and strategies for detect, understand, and adapt data streaming drifts. Data research has shown that if concept drift is not handled properly, machine learning in such an environment would provide subpar learning outcomes. In this paper, a historical Isolated Forest (HIF) is presented that depends on a decision tree, which split the data streaming into chunks and each chuck considers a region in the tree. HIF is employed to detect concept drifts and adapt this region with current changes. Which HIF stores previously generated models and employs the most similar model of each concept drift distribution as the current model until generate the best performance model. HIF doesn't stop the main system model when retraining a new model, which HIF is divided into three primary parallel blocks: detection block, similarity block (online block), and retraining block (offline block). For several authentic data sets (three data sets), our suggested algorithm was verified and contrasted. the accuracy and execution speed were specifically assessed, and memory usage. The experimental results demonstrate that our modifications use fewer resources and have comparable or greater detection accuracy than the original IForestASD.

Keywords: Automatic Machine Learning; Isolation-based; Data Streaming; Drift Detection; Model Ensemble;

1. Introduction

Governments and businesses are producing enormous volumes of streaming data, and they urgently require effective data analysis and machine learning analytics approaches to assist their ability to anticipate the future and make choices. The rapid emergence of new goods, markets, and consumer habits, however, unavoidably leads to the problem of notion drift. Concept drift describes how the target variable's statistical parameters, which the model is attempting to forecast, vary over time in unexpected ways [1]. Concept drift can result in inaccurate forecasts and poor decision-making since the induced pattern of previous data might not apply to the new data. Concept drift is an issue that affects many data-driven information systems, including data-driven decision support systems, early warning systems and, making them less effective overall. How to give more accurate data-driven forecasts and decision tools has become a critical challenge in a big data world that is always evolving. This development gives rise to a brand-new subject: adaptive data-driven prediction/decision systems. Given that the big data age is characterized by the inherent unpredictability of data kinds and distribution, concept drift is a particularly conspicuous and significant problem [2]. Prediction and training/learning are the two primary parts of conventional machine learning. Three additional steps are introduced by learning under the concept drift: concept drift detection, whether drift happens, drift understanding [2] and drift adaptation. Drift detection denotes the approaches and procedures used to define and predict concept drift by detecting changed points or change time periods [3]. Understanding drift involves finding out "When" (the time that concept drift starts and how long it lasts), "How" (the degree or severity of the concept drift), and "Where" information (the drift regions of concept drift). Which used this step as input to

the drift adaptation step. Adaptation or reaction step, sometimes known as updating current learning models in accordance with the drift. There are three main approaches to the adaptation step that try to address various types of drift: Simple Retraining, Ensemble Retraining, and Model Adjusting.
For detecting steps, several tools and algorithms are employed. Which detect drifting zones by comparing old and fresh chunks with several statistical models based on data distribution [4]. Furthermore, some technique employs a constant chunk length and others use a variable length [3] [5]. Drift Understanding step is a crucial step in understanding the shift and making the right decisions in the adaptation step. Which adaptation step requires this knowledge to investigate the start time, finish time, and zone period of each drift. The understanding phase allows you to calculate the number of modifications needed in the trained model to adapt to new changes. If the severity region is large, a new model will be generated, and discard the old one. If the severity region is small, few changes are required the adapting the old model [2]. The adaptation step is used to adapt the current model with the newly arrived chuck which has three ways to adapt the mode. Retraining a new model is the most straightforward solution to concept drift, which using the most recent data to replace the old model. When retraining the model must be determined using an explicit concept drift detector. This approach frequently uses a window strategy to keep the most current data to retrain and/or historical data for distribution change testing [4]. The second way is Model Ensemble, which comes to keeping and reusing existing models and may save a lot of time and work when repeating concept drift [6]. The Third way for adapting step is Model Adjusting, which Rather than retraining a whole model, constructs a model that adapts from changed data in an adaptable manner. Once the original data distribution has changed, such models can partially update themselves [7]. This strategy is likely to be more effective than retraining the model when the drift only happens in limited regions. Because trees may study and adjust to each sub-region individually, several solutions in this area use decision tree approaches.
In recent years, stream data mining research communities have given a lot of attention to ensemble approaches. A group of basic classifiers with varying kinds or parameters make up ensemble approaches. To forecast the newly arriving data, the result of each base classifier is blended using specific voting rules. By expanding traditional ensemble techniques or by developing unique adaptive voting rules, a variety of adaptive ensemble methods have been created with the goal of handling concept drift [2]. Therefore, there are a lot of benefits to the Ensemble technique there are some limitations: (1) Retaining a new model for each drift detection. (2) Stopping the system each concept drift until a new model is generated and merged with the existing models. (3) Repeating the same model or similar models in different regions

This paper's primary contributions are as follows:

- Use the Ensemble technique to detect and adapt drift regions.
- Save old models' properties in a fixed-length pool.
- Work with any model not restricted to specific models.
- Work with non-stationary environments.
- Don't stop the system each concept drift until a new model is generated and merged with the existing models.
- Use the most similar model until generating another one for each drift.

The remainder of this work is structured as follows. Section II offers notations and official definitions of the learning problem under consideration in this study, as well as a review of previous works. Section III discusses the HIF method. Section IV presents the results of our empirical investigations on the HIF and IForestASD approaches. Section V concludes this paper with the conclusion, drawbacks, and future works.

2. Related Work

In incremental learning, a set of data is collected each period *t* from a stationary or non-stationary environment. Which, in incremental learning, there are multiple methodologies for handling concept drift, including concept drift management approaches, ensemble methods, and IForestASD.

a) Concept Drift Handling Techniques

The sliding window techniques [8] [9] [10], which are commonly used in online learning, the most current data should be saved and updated the existing model with both the saved data and the fresh training chunk. Other techniques [11] [1] explicitly include in the learning process a concept drift detecting module. If no drift occurs, the present model is modified with freshly arriving data (latest chunk). Otherwise, the existing model, which might be a single learner [11] or an ensemble [12], is destroyed and a new model is generated from the scratch. Evidently AI [13] and Cinnamon [14] are the key tools utilized to detect drift zones. And there are several methods used in the detection step for instance: The drift Detection Method (DDM) [11], Early Drift Detection Method (EDDM) [1] [15], Adaptive Window (ADWIN) [5], Kolmogorov-Smirnov windowing method (KSWIN) [15], Hoeffding's bounds with moving average-test (HDDMA) [16], Hoeffding's bounds with moving average test (HDDMW) [17], Page-Hinkley method for concept drift detection [18].

b) Ensemble Methods

The preservation of previously learned information or models is not required by the previous technique [19]. However, there are other scenarios in which retaining historical models might be advantageous. For instance, a past model might be used as the current model if a previously concept reappears. In a broader sense, if two concepts are associated but do not occur in chronological order, adapting a historical model may be easier than continuing to train the present model or developing a new one. As a result, ensemble approaches that maintain historical models have increased in prominence in recent years [20] [5] [10]. The focus of this study is on continuous learning algorithms that maintain past models and use them to construct ensembles. The Streaming Ensemble Algorithm (SEA), the temporal inductive transfer (TIX) approach [18] [21], the dynamic integration of classifiers (DIC) approach [22], the Learn++ algorithm [4] in non-stationary environments (Learn++.NSE [10]), and Accuracy Updated Ensemble (AUE2) [23] are examples of such approaches. Although the notion of ensembles is employed in the Diversity for Dealing with Drifts (DDD) technique [12], DDD does not retain historical models and the ensembles employed in that situation might be viewed as a single model for time step t. Because of concept drift, it is obvious that historical models can have both beneficial and harmful effects on learning the current concept. As a result, a fundamental difficulty for all of the above ensemble methods is determining how to gain from historical models while avoiding undesirable effects.

Assume that a pool of historical models that have been trained over several previous time steps is available and that a fresh set of data has arrived at time step t. SEA, DIC, AUE2, and Learn++.NSE exploit the historical model in a similar way. To draw conclusions about testing instances of the present idea, The output of the fresh model is mixed with the results of the older models, which was produced with D_t. The only difference between both strategies is how the outcomes are combined. Simple majority voting is used in SEA. Through dynamic selection (DS), DV with Selection, or dynamic voting (DV), DIC integrates the past models with the fresh model. The k nearest neighbors in D_t of each testing case are initially identified. The regional performance of each model is then calculated using the closest training samples. DS selects the model with the greatest local classification performance for each testing example. DV does not pick one specific model but instead uses a weighted voting mechanism to integrate the output of all separate models. DVS is equivalent to DV, except it only applies DV to half of the single models with the top regional performance. AUE2 similarly leverages weighted voting as a combination strategy, with the weights allocated to each model determined by the models' mean squared errors. The weight is allocated to an individual model in Learn++. NSE is determined by more than just how well it performs on the most recent training data (as DIC and AUE2), but furthermore by its performance on past data. TIX offers a unique way of leveraging previous models. A fresh set of training data is provided D_t, the outputs of previous models on Dt are utilized as additional attributes of D_t, and a new model is created utilizing the enhanced D_t. TIX may be viewed as a specific weighted voting method if linear models are built throughout the learning phase. Preserving past models incurs expense in terms of both processing and storage, such as re-evaluating historical model performance on fresh training data. As a result, rather than rising

indefinitely, there should be certain restrictions on the number of retained models. To be more specific, given a set maximum number of saved models, a selection system is required to determine which past models should be retained. This problem is not handled directly in DIC, TIX, or Learn++.NSE.

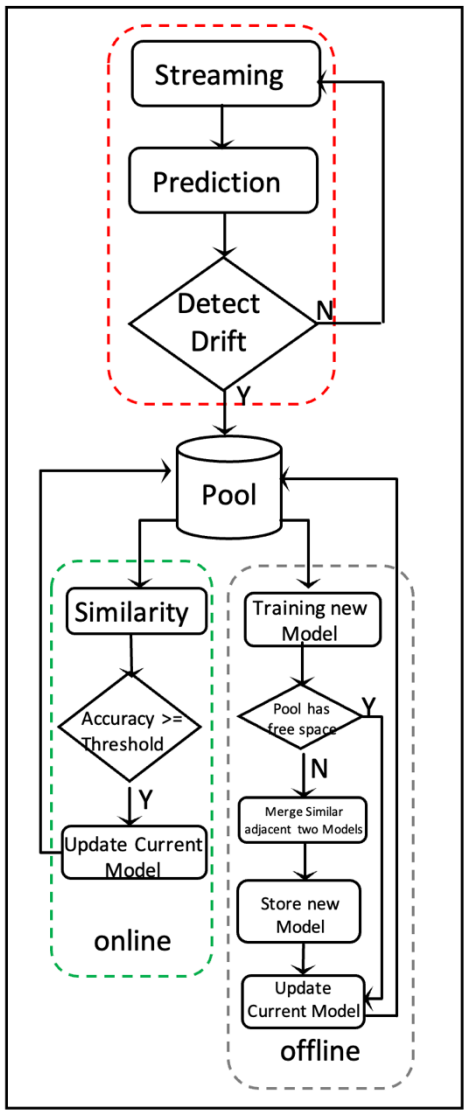

Figure 1. Historical Isolated Forest Algorithm components (HIF)

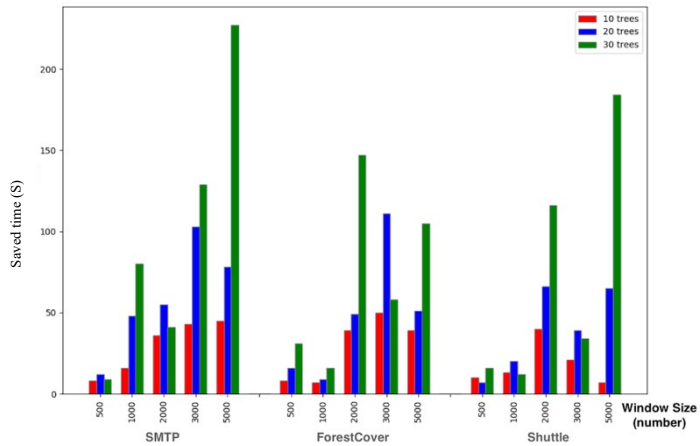

Figure 2. Saved time between HIF and IForestASD with fix length window w (500, 1000, 2000, 3000, 5000) and) and changed the amount of trees T for all datasets (STMP, Forest Cover, Shuttle)

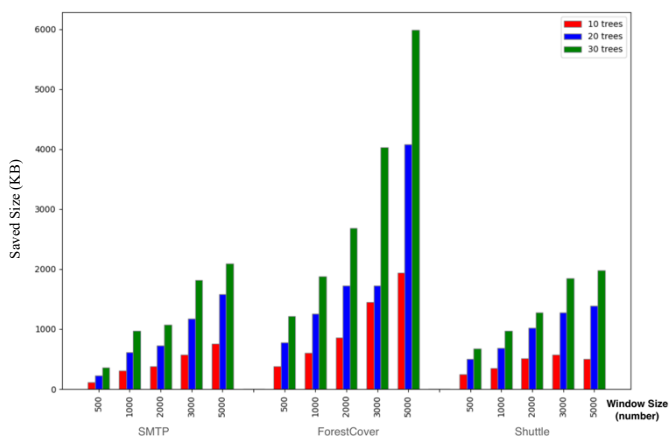

Figure 3. Saved Size between HIF and IForestASD. with fix length window w (500, 1000, 2000, 3000, 5000) and) and changed the amount of trees T for all datasets (STMP, Forest Cover, Shuttle)

Although the DIC DS/DVS scheme and the Learn++.NSE time-adjust mistakes scheme might be altered to pick past models to maintain, the usefulness of such adaptations has not been evaluated. In contrast, SEA and AUE2 actively restrict preserved models' quantity below a predetermined level. To be more specific, SEA and AUE2 both keep all previous models if their size is smaller than a certain threshold. Alternatively, when a fresh model is trained, a rule is employed to determine whether it should replace an existing preserved model by assessing the quality of both the new model and the preserved model. Both methods evaluate each model's quality in

terms of accuracy. The difference is that whereas AUE2 rates each model under consideration individually on current training data, SEA evaluates ensembles (gained by substituting a maintained model with the fresh model) on their overall accuracy. Even though variation has been shown to be crucial in ensembles, none of the ensemble techniques for incremental learning now in use explicitly address ensemble diversity. [24] [25].

c) IForestASD algorithm

Recently, Isolation Forest Algorithm (IForestASD) [26], was implemented in scikit-multiflow [27], an open-source machine learning library for streaming data, and modified in [26] to effectively manage concept drifts by expanding it with multiple drift detection approaches algorithm [26]. As a result, the authors expanded the ADWIN [5] and KSWIN [15] drift detectors SADWIN/PADWIN and NDKSWIN to reduce the training time and model size of the ensemble. However, several important shortcomings of their proposals, such as changing partially or completely forest each concept and not saving it for use in the subsequent concept.

3. Proposed Approach

As described in Section II, existing ensemble approaches for incremental learning, generate a new model for each concept drift and discard the old without performing any operation to fetch the model's knowledge. This is scenario unsuitable, because the historical drifts might occur with new coming data, and it's mattered to use it instead of generating another one. Also, each technique stops the system until generates another adaptive one. Through the use of the Ensemble approach and other adjustments to the IForestASD technique, our proposal (Historical Isolated Forest) attempted to address the aforementioned problems.

3.1. HIF Algorithm

In this part, HIF (Historical Isolated Forest) algorithm is described, which uses the IForestASD technique to discover and adapt concept drift zones. Data streaming is divided into regions (chunks). Each chunk has a model, standard deviation, and mean. HIF utilizes region properties to discover related areas based on the region's mean and standard deviation. HIF improves the IForestASD process by identifying the most similar model to the current data distribution. As illustrated in equation 1, Euclidean Distance is employed to calculate the distance between the current distribution and all previous chunks distributions. To speed up the HIF procedure, all components carry out concurrently, each component executes separately on its own thread. As shown in Fig.1, HIF is divided into three main components:

a) The first component (red dashed rectangle) is used for normal work, a machine learning algorithm that uses the current model for prediction and is employed to detect any concept drift in the data distribution.
b) The second component (green dashed rectangle) is used to identify the model that best fits the current chunk distribution and has model accuracy that is at least as accurate as the accuracy threshold. And employed this model as the current model until the third component generate a new model, used in the first component.
c) Simultaneously to the second component, the third component (black dashed rectangle) is preformed, which is utilized to generate a new model for the present chunk distribution. Then merge the most similar adjacent model's distribution if the pool doesn't have free space. And replace the current model used in the first component with the new model.

$$similar\ model = min\ dist(\ Euclidean\ Distance(current\ distribution, all\ stream\ distribution)) \quad (1)$$

3.2. Detailed HIF Procedures

In this section, HIF algorithm will be looked through in depth, as well as the IForestASD tree that was used as an input to HIF in algorithm 1. And the HIF algorithm follows these steps:

a) The first component of HIF is implemented from lines 1 to 3: which, line 3, performs the model's essential functionality Line 3 determines if the current chunk drift rate has exceeded the allowable threshold and fires the second and third components if it has exceeded.

b) The second component of HIF is implemented from lines 4 to 11: At line 4, calculate the similarity between the current chunk distribution and IForestASD Trees distribution. At line 5, find the most similar distribution and use her model as the most similar model. from lines 7 to 11, update the existing model if the most similar algorithm accuracy is greater than the drift threshold. and store the current chunk mean and standard deviation, which are used in the similarity between historical models.

c) The third component of HIF is implemented from lines 12 to 25: At line 12, a new model is trained based on the current chunk distribution. At line 13, determine whether the IForestASD instance has free space to store the new algorithm and if so, store it and replace the current algorithm; otherwise, as shown in the algorithm from steps 16 to 25, fetch all IForestASD models and calculate the similarity between each similar adjacent two models, and merge the closest adjacent two models. After that, add the new model to the IForestASD models list, along with the standard deviation and mean of her distribution.

Algorithm 1: HFE Algorithm

```
Input: Isolated Forest Tree, data streaming
Output: Updated Isolated Forest Tree
1  for stream have data do
2      predict ← prediction phase
3      if predic ≥ driftThrehold then
4          do in parallel
5              calculateSimilarty(current chunk)
6              similarModel ← getSimilarModel()
7              if similarModel.accurcy ≥ driftThrehold then
8                  curentModel ← similar model
9                  new mean ← getMean(newChunk)
10                 new standar divation ← getMean(newChunk)
11                 updateIsolatedForestTree(newModel, new standar divation, new mean)
12         do in parallel
13             newModel ← trainingPhase()
14             if IsolatedForestPool have free space then
15                 curentModel ← new model
16             else
17                 oldModels ← getIsolatedForestModels()
18                 curentModel ← new model
19                 for model index ← oldModels.size() do
20                     calculateSimilarty(oldModels[model index], oldModels[model index + 1],)
21                 similarAdjacentTowModels ← getSimilarAdjacentTwoModels()
22                 merge(similarAdjacentModels)
23             new mean ← getMean(newModel)
24             new standar divation ← getMean(newModel)
25             updateIsolatedForestTree(newModel, new standar divation, new mean)
26 return Updated Isolated Forest Tree
```

4. Experimental Results

This section will explain the outcomes of experiments that support our contribution. The section will be structured as follows:

a) Description of the setup of the tests and the materials used to produce the work.
b) Outcomes of our enhancement along with a thorough comparison to IForestASD.

4.1. Datasets and Evaluation Metrics

In this section the datasets were provided and utilized to demonstrate our experimental findings with various parameters, features, and error rates as shown in the Table 1, as well as our evaluation matrices that performed in these datasets.

4.1.1. Datasets Description

Table 1. Datasets and hyper-parameters set up.

Hyper-parameters \ Datasets	Shuttle	Forest-Cover	SMTP
Drift Rate = Anomaly rate(u)	7.15	0.96	0.03
Number of samples(i)	49,097	0.286,048	95,156
Window size range (W)	[500,1000,2000,3000,5000]	[500,1000,2000,3000,5000]	[500,1000,2000,3000,5000]
Number of Trees (T)	[10,20,30]	[10,20,30]	[10,20,30]

As shown in Table 1, the Shuttle dataset has nine features with 7.15 as an error rate, ten features for Forest-Cover with 0.96 error rate, and three features for SMTP with 0.03 error rate. A benchmark utilizing our suggested modification and the IForestASD technique on a variety of parameters were ran, three parameters for the number of estimators, and five parameters for the window size.

4.1.2. Experimental Setup

To compare IForestASD with our proposal, on the one hand, experiments with the hyper-parameters were ran provided in Table 1. The threshold for anomalies (drifts) is set to 0.5. This is the number utilized to identify drift and reset the anomaly detection model. As performed for IForestASD in [26], the parameter *u* is setup to the true anomalies rate in the datasets.

4.1.3 Evaluation Metrics

The model's performance is evaluated using the (test-then-train technique), which is specifically designed for data streaming settings in that each chunk serves two purposes (test then train) and that chunks are analyzed sequentially, in chronological order, and become unreachable instantly. This strategy assures that the model is assessed on new chunks that have not been used in training earlier. For the implementation of IForestASD) [26] and the suggested proposal, three metrics: accuracy, running time (training + testing), and model size were focused:

a) **Accuracy:** Equation 2 is employed to calculate the model accuracy [28]. Which *FP* relates to the number of expected values as a positive number, but the actual value was negative, therefore the Expected values were wrong. *TP* and *TN* refer to the number of actual values that is consistent with predicted values. *FP* relates to the number of expected values as a negative number, but the actual value was positive.

$$Accuracy = (TN + TP)/(TN + TP + FN + FP) \quad (2)$$

b) **Running Time**: Fig.1 shows the running time's absolute values (in seconds) and how they have changed when hyper-parameter variations have occurred, also shows that our proposal is faster than IForestASD in all hyper-parameters.
c) **Model Size**: when the model size (KB) is analyzed instead of the execution time, our proposal always utilized less memory than IForestASD. Fig.3 reports the effect of parameter selection (number of trees and window length) on resource (SMTP, Forest Cover, and shuttle). for both IForestASD and HIF.

Table 2. Comparison between IForestASD and Historical Isolated Forest (HIF). The window size *w* was fixed (500, 1000, 2000, 3000, 5000), and changed the amount of trees *T* each iteration for SMTP dataset.

Window size(number)	Estimator	iForestASD				HIF			
		Detection	Accuracy(%)	Total time(s)	Size(KB)	Detection	Accuracy(%)	Total time(s)	Size(KB)
500	10	6	0.81	202.52	433.99	6	0.8474	**195.51**	**316.58**
	20	5	0.83	390.68	572.12	5	0.81	**379.53**	**344.67**
	30	5	0.82	559.55	699.81	5	0.80	**550.07**	**344.70**
1000	10	3	0.81	421.13	933.24	3	0.78	**405.57**	**627.70**
	20	2	0.78	806.75	1300.22	2	0.79	**748.99**	**690.82**
	30	3	0.79	1245.81	1660.90	3	0.79	**1164.65**	**690.84**
2000	10	1	0.730	828.07	1711.03	1	0.72	**792.76**	**1333.14**
	20	2	0.7285	1596.92	2064.98	2	0.73	**1542.94**	**1337.70**
	30	2	0.73	2228.79	2369.29	2	0.73	**2187.05**	**1295.54**
3000	10	2	0.64	1152.04	1624.18	2	0.63	**1109.10**	**1056.68**
	20	2	0.6341	2218.37	2295.33	2	0.62	**2115.85**	**1117.33**
	30	2	0.62	3049.92	2927.87	2	0.63	**2920.26**	**1113.61**
5000	10	1	0.47	1446.40	3821.90	1	0.47	**1401.61**	**3066.41**
	20	1	0.47	2885.24	4685.79	1	0.47	**2807.76**	**3104.33**
	30	1	0.48	3942.60	5159.23	1	0.47	**3715.69**	**3063.79**

4.2. Discussion and Comparison between IForestASD and Our Proposal

Tables 2,3,4 show the results each time a hyper-parameter combination is used (window size and the number of trees "Estimator") and show number of drifts "Detection", accuracy and total time of training and testing phases and the model size of each hyper-parameter, for each dataset (SMTP, Forest Cover, and shuttle). The best values were emphasized by making them bold. Regarding Model Size and running time were discovered, our proposal outperforms IForestASD for both the Shuttle, Forest-Cover, and SMTP datasets, regardless of the number of trees or the size of the windows. Three further results were observed:

a) **Running time:** Our methodology produces the best results for all datasets. As shown in Fig. 3, each column represents the decreased time from the IForestASD method and demonstrates how the decreased time grows as the training time or the number of trees increases. Three regions (three datasets) are shown in Fig. 3, and each region shows how each hyper-parameter's time has decreased.

Our methodology takes the least amount of time since it occasionally utilizes the most similar model instead of generating a new one.

b) **Model Size**: In contrast to IForestASD, the model size in our methodology is smaller. Because it occasionally uses the most similar model instead of generating a new one, resulting in less generated models than IForestASD's models, which take up less space. Additionally, as shown in Fig. 2, the size-decreasing rate increased by increasing the number of trees or window size because the needed size rises depending on increasing the window size or the number of estimators.

c) **Accuracy**: As shown in the preceding section (Proposed Approach), the proposal accuracy may rise or fall depending on training duration, with the proposal algorithm accuracy being greater than or equal to IForestASD when training time for the new model is short. In several circumstances, our proposal's accuracy falls short of IForestASD. Because our proposal uses the most similar previous model instead of sleeping the model until it generates a new model.

Table 3. Comparison between IForestASD and Historical Isolated Forest (HIF). The window size *w* was fixed (500, 1000, 2000, 3000, 5000), and changed the amount of trees *T* each iteration for Forest Cover dataset.

Window size(number)	Estimator	iForestASD				HIF			
		Detection	Accuracy(%)	Total time(s)	Size(KB)	Detection	Accuracy(%)	Total time(s)	Size(KB)
500	10	5	0.56	242.06	1144.70	5	0.82	**234.70**	**762.55**
	20	7	0.57	462.82	1539.39	7	0.81	**446.67**	**762.57**
	30	6	0.60	689.42	1979.21	6	0.74	**658.14**	**762.60**
1000	10	5	0.55	508.53	2038.46	5	0.66	**501.88**	**1435.95**
	20	5	0.55	974.40	2699.34	5	0.66	**965.28**	**1440.48**
	30	4	0.53	1423.07	3318.95	4	0.69	**1407.67**	**1436.01**
2000	10	2	0.54	973.92	3700.88	2	0.52	**934.87**	**2840.18**
	20	2	0.57	1875.39	4566.35	2	0.55	**1826.42**	**2840.59**
	30	2	0.60	2929.35	5525.77	2	0.56	**2782.49**	**2845.08**
3000	10	1	0.53	1365.39	3815.65	1	0.49	**1315.42**	**2368.84**
	20	2	0.55	4392.21	5851.09	2	0.52	**4281.64**	**2368.89**
	30	2	0.56	3850.86	6399.11	2	0.55	**3792.73**	**2373.95**
5000	10	2	0.47	1797.23	9244.56	2	0.478	**1758.45**	**7305.93**
	20	1	0.48	3567.57	11330.27	1	0.49	**3516.71**	**7253.33**
	30	1	0.48	5347.56	13295.79	1	0.49	**5242.19**	**7310.76**

Empirical research has been carried out to evaluate the performance of HIF. The experiment involves several varieties of concept drift. and compares IForestASD. HIF and IForestASD are primarily assessed in three ways: each chunk's model size, total performance for a full data streaming, and time saving. some notes were observed when HIF and IForestASD were applied on each dataset (SMTP, Forest Cover, and shuttle):

a. The best HIF performance is appeared when the number of trees or the window size is increased as shown in Fig.1 and Fig.2, because training time is the largest and needs more space in memory so in these cases.
b. The number of drifts is increased by increasing the number of windows.
c. HIF's accuracy depends on the training time for the new algorithm, which uses the most similar algorithm until generating a new one.

5. Conclusion and Future Work

This study introduces a new ensemble technique for continuous learning with concept drift (Historical Isolated Forest). Instead of stopping the system until a new model is generated, HIF depends on a decision tree, which split the data streaming into chunks and each chuck and stores previously generated models and employs the most similar model of each concept drift as the current model until generate the suitable model. HIF is divided into three components, one for drift detection, one for finding the most similar model, and one for generating the current suitable model. Empirical investigations on real datasets streams (STMP, forest cover, shuttle) demonstrate the advantages of HIF over the IForestASD technique in model size and time. The main potential drawback of HIF is that the model accuracy may be decreased: because it was used as an unsuitable model for some time until finding the most similar model for the presented environment. To improve model performance of our work, the next drift is needed to predict and predict the suitable model for this drift, Consequently, the model's accuracy will increase.

Table 4. Comparison between IForestASD and Historical Isolated Forest (HIF). The window size *w* was fixed (500, 1000, 2000, 3000, 5000), and changed the amount of trees *T* each iteration for Forest shuttle dataset.

Window size(number)	Estimator	iForestASD				HIF			
		Detection	Accuracy(%)	Total time(s)	Size(KB)	Detection	Accuracy(%)	Total time(s)	Size(KB)
500	10	2	0.92	248.17	910.06	2	0.91	**238.99**	**662.59**
	20	2	0.9182	492.44	1200.84	2	0.93	**485.38**	**701.21**
	30	2	0.93	709.09	1340.20	2	0.92	**693.57**	**667.43**
1000	10	2	0.87	517.41	1742.79	2	0.88	**504.62**	**1396.19**
	20	2	0.88	1006.76	1996.18	2	0.89	**986.95**	**1315.78**
	30	2	0.88	1526.04	2280.65	2	0.88	**1514.25**	**1314.05**
2000	10	2	0.77	1042.23	3118.03	2	0.77	**1002.52**	**2612.53**
	20	2	0.77	1923.95	3637.31	2	0.76	**1857.02**	**2612.55**
	30	2	0.77	2881.75	3893.58	2	0.77	**2765.72**	**2621.45**
3000	10	1	0.68	1339.17	2744.83	1	0.66	**1318.25**	**2177.00**
	20	2	0.67	2662.54	3461.49	2	0.67	**2623.94**	**2183.65**
	30	2	0.68	3853.56	4027.02	2	0.67	**3819.76**	**2178.90**
5000	10	2	0.78	1042.74	3118.06	2	0.77	**1035.62**	**2618.31**
	20	1	0.48	3434.79	7893.02	1	0.47	**3369.67**	**6507.09**
	30	1	0.47	5163.02	8802.85	2	0.47	**4979.00**	**6820.08**

REFRENCES

[1] M. Baena-Garcıa, J. del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, 'Early drift detection method', in Fourth international workshop on knowledge discovery from data streams, 2006, vol. 6, pp. 77–86.

[2] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, 'Learning under concept drift: A review', IEEE transactions on knowledge and data engineering, vol. 31, no. 12, pp. 2346–2363, 2018.

[3] M. Basseville, I. V. Nikiforov, and Others, Detection of abrupt changes: theory and application, vol. 104. prentice Hall Englewood Cliffs, 1993.

[4] S. H. Bach and M. A. Maloof, 'Paired learners for concept drift', in 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008.

[5] A. Bifet and R. Gavalda, 'Learning from time-changing data with adaptive windowing', in Proceedings of the 2007 SIAM international conference on data mining, 2007, pp. 443–448.

[6] Y. Sun, K. Tang, Z. Zhu, and X. Yao, 'Concept drift adaptation by exploiting historical knowledge', IEEE transactions on neural networks and learning systems, vol. 29, no. 10, pp. 4822–4832, 2018.

[7] P. Domingos and G. Hulten, 'Mining high-speed data streams', in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 71–80.

[8] G. Widmer and M. Kubat, 'Learning in the presence of concept drift and hidden contexts', Machine learning, vol. 23, pp. 69–101, 1996.

[9] G. Hulten, L. Spencer, and P. Domingos, 'Mining time-changing data streams', in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001, pp. 97–106.

[10] R. Elwell and R. Polikar, 'Incremental learning of concept drift in nonstationary environments', IEEE Transactions on Neural Networks, vol. 22, no. 10, pp. 1517–1531, 2011.

[11] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, 'Learning with drift detection', in Advances in Artificial Intelligence--SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17, 2004, pp. 286– 295

[12] L. L. Minku and X. Yao, 'DDD: A new ensemble approach for dealing with concept drift', IEEE transactions on knowledge and data engineering, vol. 24, no. 4, pp. 619–633, 2011.

[13] 'Evidently AI - Open-Source Machine Learning Monitoring'. [Online]. Available: https://www.evidentlyai.com/. [Accessed: 06-Feb-2023]

[14] 'CinnaMon - Python library which allows to monitor data drift on a machine learning system' .[Online]. Available: https://cinnamon.readthedocs.io/en/latest/quickstart.html. [Accessed: 06-Feb-2023].

[15] C. Raab, M. Heusinger, and F.-M. Schleif, 'Reactive soft prototype computing for concept drift streams', Neurocomputing, vol. 416, pp. 340– 351, 2020.

[16] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, 'Online and non-parametric drift detection methods based on Hoeffding's bounds', IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 3, pp. 810–823, 2014.

[17] Chikushi, Rohgi Toshio Meneses, Roberto Souto Maior de Barros, Marilu Gomes N. Monte da Silva, and Bruno Iran Ferreira Maciel. "Using spectral entropy and bernoulli map to handle concept drift." *Expert Systems with Applications* val. 167, pp.114-148, 2021

[18] Baier, Lucas, Josua Reimold, and Niklas Kühl. "Handling concept drift for predictions in business process mining." In *2020 IEEE 22nd Conference on Business Informatics (CBI)*, vol. 1, pp. 76-83, 2020.

[19] de Barros, Roberto Souto Maior, and Silas Garrido T. de Carvalho Santos. 'An overview and comprehensive comparison of ensembles for concept drift.' *Information Fusion, vol.* 52, pp. 213-244.,2019

[20] W. N. Street and Y. Kim, 'A streaming ensemble algorithm (SEA) for large-scale classification', in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001, pp. 377–382.

[21] G. Forman, 'Tackling concept drift by temporal inductive transfer', in Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006, pp. 252–259.

[22] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, 'Dynamic integration of classifiers for handling concept drift', Information fusion, vol. 9, no. 1, pp. 56–68, 2008.

[23] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, 'Learn++: An incremental learning algorithm for supervised neural networks', IEEE transactions on systems, man, and cybernetics, part C (applications and reviews), vol. 31, no. 4, pp. 497–508, 2001.

[24] G. Brown, J. Wyatt, R. Harris, and X. Yao, 'Diversity creation methods: a survey and categorisation', Information fusion, vol. 6, no. 1, pp. 5– 20, 2005.

[25] E. K. Tang, P. N. Suganthan, and X. Yao, 'An analysis of diversity measures', Machine learning, vol. 65, pp. 247–271, 2006.

[26] M. U. Togbe, Y. Chabchoub, A. Boly, M. Barry, R. Chiky, and M. Bahri, 'Anomalies detection using isolation in concept-drifting data streams', Computers, vol. 10, no. 1, p. 13, 2021.

[27] M. U. Togbe et al., 'Anomaly detection for data streams based on isolation forest using scikit-multiflow', in Computational Science and Its Applications--ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1--4, 2020, Proceedings, Part IV 20, 2020, pp. 15–30.

[28] J. Davis and M. Goadrich, 'The relationship between Precision-Recall and ROC curves', in Proceedings of the 23rd international conference on Machine learning, 2006, pp. 233–240.

[29] D. Brzezinski and J. Stefanowski, 'Reacting to different types of concept drift: The accuracy updated ensemble algorithm', IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 1, pp. 81–94, 2013.