**CHAPTER 1**

**INTRODUCTION**

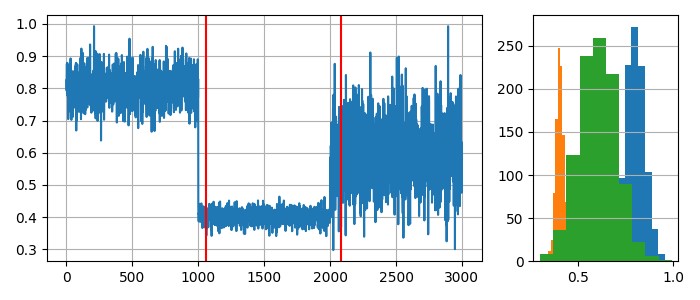
Process mining has become a vital tool for businesses to comprehend, evaluate, and enhance their business processes in the big data era. Process mining gives firms the ability to examine process maps, identify deviations from established protocols, and improve overall process efficiency by gleaning important insights from massive datasets. Dealing with concept drift, a phenomenon where the statistical characteristics of the target variable change over time and result in inaccurate process models and forecasts, is a major difficulty in process mining. Concept drift can be caused by a number of things, including modifications to company policy, shifts in the market, or changes in legal requirements. In dynamic and complicated situations, traditional process mining techniques may lead to models that become old or imprecise because they presume a static underlying process. Thus, it is critical to develop robust techniques in the context of process mining that are able to identify, allow for, and regulate idea drift in big data environments.

Background and Significance of Process Mining in Big Data:

Through the extraction of knowledge from event logs, process mining is a data-driven methodology that enables firms to assess and enhance their business processes. Time stamps, the resources used, and other pertinent information are all included in these logs, which provide comprehensive information about how activities are carried out. Process enhancement, compliance verification, and discovery are just a few of the uses for process mining. Process mining is an important tool for big data organizations to help them make sense of massive amounts of data and get insights that can be put to use. Process mining can help firms optimize their workflows and boost productivity by identifying bottlenecks, inefficiencies, and deviations from conventional procedures through the examination of event logs. Additionally, process mining can help with decision-making by enabling data-driven decision-making and offering real-time insights into how processes are being executed.

Explanation of Concept Drift and Its Impact on Process Mining:

The term "concept drift" describes a process where the target variable's statistical characteristics alter over time, making it difficult to maintain precise models and forecasts in dynamic environments. Concept drift in the context of process mining might appear as behavioural changes in the processes, leading to differences between the processes that are currently being executed and those that were previously modelled. Concept drift can have a big effect on process mining. In dynamic and complicated situations, traditional process mining techniques sometimes presume a static underlying process, which can fast cause models to become out-of-date or erroneous. This can therefore lead to inaccurate insights and less-than-ideal decision-making. For process mining approaches to continue to be effective and for timely and correct insights to be obtained from process mining operations, it is imperative that idea drift be addressed.



***Fig 1: Overview of the concept drift phenomenon***

Statement of the Research Problem:

The research problem focuses on developing robust methodologies to effectively identify, adapt to, and manage concept drift in big data environments within the context of process mining. The challenge lies in developing adaptive algorithms that can detect concept drift in real-time and update process models accordingly, without significant manual intervention, while maintaining high levels of accuracy and minimizing computational overhead.

Objectives of the Study:

1. Development of Real-Time Drift Detection Algorithms: The initial goal is to create algorithms that can instantly detect idea drift in process data streams. These methods ought to quantify drift's effect on the existing process model in addition to identifying its existence. Enabling timely modifications to the process models before a major loss in accuracy is the aim. This entails using statistical analysis and machine learning techniques to continuously examine event logs in order to spot patterns that point to a change in the behavior of the process.

2. Adaptive Process Model Updating Mechanisms: The second goal is to develop methods that allow process models to be updated in response to idea drift that is identified without necessitating a full reprocessing of the historical data. In order to maintain the models' ability to accurately reflect the dynamics of the current process, this involves creating incremental learning algorithms that can modify the current process models with a minimum amount of processing power. The difficulty lies in striking a compromise between stability and adaptability so that the updated models are neither very stiff to account for real deviations nor overly sensitive to slight differences.

3. Evaluation and Benchmarking of Drift Management Techniques: The third goal entails carrying out thorough analyses of the suggested drift management strategies using actual datasets and comparing them to accepted practices. The purpose of this objective is to evaluate the created methodologies' efficacy, efficiency, and scalability across a range of scenarios, encompassing varying degrees and kinds of notion drift. Along with this, a methodology for methodically evaluating the accuracy, timeliness, and computational efficiency of process mining algorithms in the context of concept drift is being developed.

Contributions of the Paper:

The study significantly advances the fields of concept drift and process mining. It offers helpful insights into the efficacy of various drift management strategies and suggests innovative ways to increase the accuracy of process mining models in dynamic environments. Additionally, a thorough evaluation approach for assessing process mining algorithms' performance in the context of concept drift is presented in the paper. The report identifies the main obstacles and gaps in the body of existing research by providing a thorough examination of the state-of-the-art in process mining and concept drift. The study offers creative ideas to deal with these issues and close the gaps found in the investigation. The efficiency and effectiveness of the suggested solutions are demonstrated by comparing the evaluation findings with those of other approaches using real-world datasets.

Additionally, the research advances the creation of a thorough assessment framework that will be used to evaluate how well process mining algorithms perform when concept drift is present. This methodology offers a methodical way to assess the effectiveness of drift management strategies by accounting for a number of variables, including accuracy, timeliness, and computing efficiency. In conclusion, this research provides a thorough analysis of how adaptive algorithms might be used to mitigate concept drift in process mining. Using real-world datasets, the study assesses novel methodologies for adaptive process model update and drift detection in real-time. Additionally, the research advances the creation of a thorough assessment framework that will be used to evaluate how well process mining algorithms perform when concept drift is present. The study's conclusions have important ramifications for the academic community as well as business, as they offer insightful information about how to control concept drift in process mining.

**CHAPTER 2**

**LITERATURE REVIEW**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Paper | Key Contributions | Techniques/Algorithms Used | Performance Metrics and Values | Key Takeaways/Findings |
| [1] | Ensemble-based online machine learning algorithms for network intrusion detection | Boost HT, Boost HAT, ARF, ARF (20) | Prequential AUC (High values), Run-time (Longer for ARF(20)) | Ensemble algorithms outperform stand-alone models, with trade-offs between accuracy and run-time. |
| [2] | Comparing concept drift detection in process mining software | ProDrift, ProM tools | Drift point detection frequency (1209 unique drift points out of 8680 in 160 tests for ProDrift) | Parameter configuration significantly impacts drift detection results. |
| [3] | Robust drift detection in security domain | Novel drift detection approach | Precision (0.996), Recall (0.999), F1 score (0.997), AUROC (High) | Proposed method effectively detects and mitigates drifts in security contexts. |
| [4] | Intrusion detection with drift detection and incremental learning | DDM-ORF system | Accuracy (99.96%), Precision (99.93%), Recall (99.95%), F-measure (99.94%) | DDM-ORF achieves high accuracy and real-time processing capabilities. |
| [5] | Improving ML/DL model performance using grid search and boosting | Grid search, XGBoost | Decision Tree accuracy improved from 0.518 to 0.83, FFNN accuracy improved from 0.62 to 0.79 | Grid search and boosting substantially improve ML/DL model predictive performance. |
| [6] | Integrating machine learning with process mining | - | - | Highlights the importance of aligning ML assumptions with process data for better accuracy and interpretability. |
| [7] | Adaptive ensemble active learning for drifting data streams | EAL-MAB algorithm | Prequential accuracy (Higher than reference methods) | EAL-MAB effectively handles labelling constraints and improves ensemble performance in drifting data streams. |
| [8] | Process mining for detailed process analysis | Process mining visualization | (No specific values mentioned) | Emphasizes the value of data-driven process analysis and process mining for performance improvements. |
| [9] | Integrated drift detection and localization in process mining | Batch trace clustering | AUC up to 1.00, F-score up to 1.00 for certain change patterns | Batch trace clustering consistently outperforms other strategies for drift detection and localization. |
| [10] | Deep learning approach to business process mining | PGraphDD-SS, PGraphDD-QM | PGraphDD-SS accuracy (75%) | Combines deep learning with graph-based methods for improved predictive performance and concept drift detection. |
| [11] | Deep learning framework for concept drift and class imbalance | CIDD-ADODNN model | On Chess dataset: Precision (0.7515), Recall (0.7974), Specificity (0.7311), Accuracy (0.7646), F-score (0.7738) | CIDD-ADODNN outperforms traditional models in handling concept drift and class imbalance. |
| [12] | Concept drift adaptation methods under deep learning framework | Various adaptation methods | (No specific values mentioned) | Highlights the need for further research to enhance adaptability of deep learning models to concept drift. |
| [13] | Unsupervised concept drift detection | STUDD algorithm | Cohen's Kappa (Comparable to supervised), Labeling costs (Significantly lower) | STUDD achieves comparable predictive performance to supervised methods while reducing labeling costs. |
| [14] | Machine learning system for malicious website detection | XGBoost algorithm | Accuracy (96.4%) | Effective system for detecting malicious websites and URLs using XGBoost and content-based features. |
| [15] | Concept drift and fraud detection in streaming environment | XGBoost-based model | Accuracy (99.98%), Recall (1.0), F1-score (0.9538) | Proposed XGBoost model outperforms traditional models in detecting fraudulent transactions in streaming data. |
| [16] | Enhanced intrusion detection with concept drift | Genetic Programming Combiner | (No specific values mentioned) | Combines data stream classification and concept drift detection for enhanced intrusion detection. |
| [17] | Hybrid deep learning classifier for drift detection and adaption | Intelligent Preying Optimization, Key Windowing | At 80% training: Accuracy (62.70%), Sensitivity (97.81%), Specificity (97.03%), Precision (90.09%), F1-measure (62.70%) | Proposed framework improves classification performance and adaptability in dynamic data environments. |
| [18] | Concept drift detection and adaptation for federated learning | CDA-FedAvg algorithm | Overall accuracy around 82% on test set | CDA-FedAvg effectively adapts to nonstationary data in federated and continual learning scenarios. |
| [19] | Accurate concept drift detection in evolving data streams | ACDDM algorithm | RandomTree: True positive (2.17), False positive (4.33), Delay (358.11), Accuracy (70.71%); Mixed-60K: True positive (5.0), False positive (0.23), Delay (280.17), Accuracy (91.67%) | ACDDM outperforms other detectors in true drift detection, false alarms, and delay of detection. |
| [20] | Unsupervised concept drift detection for multi-label data streams | LD3 algorithm | Example-based accuracy improvement (16.9% to 56%), Values from 0.0313 to 0.2479 | LD3 effectively detects concept drift in multi-label data streams, outperforming supervised detectors. |
| [21] | Tiny machine learning for concept drift | TML-CD, Hybrid Tiny kNN | Memory savings - Speech (32-50% samples), Images (60-82% samples); Mean accuracy ± std dev reported for Hybrid Tiny kNN | TML-CD and Hybrid Tiny kNN enable memory-efficient concept drift handling in resource-constrained environments. |
| [22] | Adaptive rate processing for power quality disturbances identification | Machine learning with adaptive rate processing | Accuracy (89.24%) | Adaptive rate processing improves the accuracy of identifying power quality disturbances. |
| [23] | Boosted regression ensemble with hyperparameter tuning | RF-LS-BPT method | R-squared (0.9800 to 0.9999), MAE (0.42 to 4.24) | RF-LS-BPT with hyperparameter tuning enhances throughput prediction accuracy in 5G networks. |
| [24] | Realistic drift detection in production ML systems | Energy Distance, KL Divergence, etc. | Detection thresholds, Expected times to detection (Values not provided) | Suggests using a combination of metrics, with KL Divergence and Energy Distance effective for monitoring drift. |
| [25] | Aiding phishing website detection with a feature-free tool | PhishSim (Normalized Compression Distance) | AUC (98.68%), TPR (≈90% at FPR 0.58%, 88.37% at FPR 0.09%), Accuracy (99.82% at FPR 0.09%) | PhishSim outperforms existing techniques in phishing website detection, with high accuracy and low false positive rates. |

**CHAPTER 3**

**THEORETICAL BACKGROUND**

Process mining is a fast-growing subject that analyses and enhances business processes by combining process modelling with data mining. It makes it possible for businesses to get insightful information out of event logs, which document how a process is carried out. Managing concept drift, or alterations in the fundamental process behaviour over time, is one of the most important problems in process mining. In-depth explanations of idea drift in process mining, its effects on process models, and the application of machine learning approaches to this problem are given in this part.

3.1 Concept Drift in Process Mining

The phenomenon known as "concept drift" is the result of changes in a process's underlying data distribution over time, which alters the behaviour of the process. The precision and dependability of process models can be greatly impacted by idea drift in the context of process mining. Concept drift can be divided into various categories according to its frequency, rate of occurrence, and effect on process behaviour. Concept drift can be categorized into four types based on how it occurs: abrupt, incremental, gradual, and recurrent, blip and noise. While incremental drift refers to a sequence of tiny changes that build up over time, sudden drift describes an abrupt change in the process behaviour. Recurring drift is the term used to describe the repeated recurrence of a certain process behaviour, whereas gradual drift is the term used to describe steady, continuous changes in process behaviour.

Concept drift may have important consequences for process models. When idea drift occurs, traditional process mining techniques frequently presume that the underlying process is static, which can result in outdated and erroneous process models. Consequently, this may have an adverse effect on the process of making decisions and impede the streamlining of corporate procedures. Thus, in process mining, it is imperative to create strong techniques that can recognize, accommodate, and control concept drift.

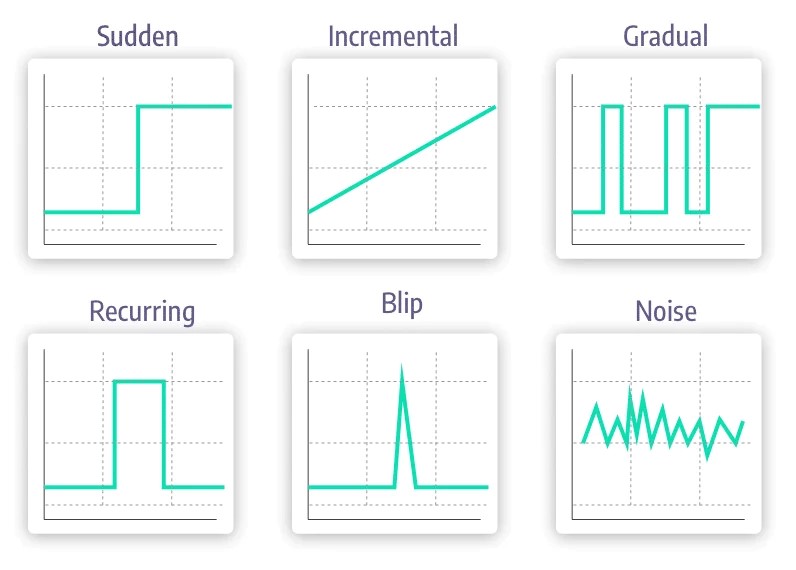


Fig 2: Different Category of Concept Drift

3.2 Machine Learning for Process Mining

Process mining idea drift may be addressed with machine learning approaches, which have shown to be a promising approach. With the use of these methods, adaptive algorithms that can recognize and react instantly to modifications in the behaviour of the process can be developed. An introduction of machine learning methods used in process mining is given in this part, along with information on the potential and problems that come with using them.

Supervised, unsupervised, and reinforcement learning are the three main categories into which machine learning approaches can be divided. Using labelled data, supervised learning entails training a model to forecast the output for newly uncovered data. Supervised learning can be applied to process mining activities like classifying process instances, detecting anomalies, and forecasting process outcomes. In contrast, unsupervised learning is using unlabelled data to train a model in order to find patterns and relationships in the data. In process mining, this can be especially helpful for process identification and clustering. One kind of machine learning is called reinforcement learning, which is teaching a model to make decisions in response to input from its surroundings. Reinforcement learning can be applied to process mining to optimize resource allocation and process execution.

In process mining, machine learning approaches have a number of ways to solve concept drift. For example, it has been demonstrated that idea drift can be effectively detected using ensemble-based online machine learning algorithms in network intrusion detection systems. These methods integrate several base learners to increase the model's precision and flexibility. Similar to this, drifting data stream mining adaptive ensemble active learning techniques have been described. These techniques take advantage of the diversity of ensemble classifiers to choose informative examples for categorization.

The requirement for ongoing model monitoring and updating to guarantee that the process models stay correct and pertinent in the face of idea drift is another difficulty. This necessitates the creation of scalable, effective algorithms that can instantly recognize and adjust to modifications in process behaviour. In addition, the interpretability of machine learning models is essential to guaranteeing that the insights obtained from process mining operations are intelligible and applicable to professionals in the relevant fields.

In summary, concept drift is a serious problem for process mining and can have a detrimental effect on the precision and dependability of process models. This problem may be solved with promise thanks to machine learning techniques, which make it possible to create adaptive algorithms that can quickly recognize and react to changes in process behaviour. Process data is complicated and variable, requiring constant model monitoring and updating, and requiring a lot of labelled data. These are just a few of the difficulties that come with using machine learning approaches to process mining. It is important to tackle these obstacles in order to guarantee the efficient utilisation of machine learning methodologies in process mining and augment the general efficiency of company operations.

**CHAPTER 4**

**METHODOLGY**

4.1 Real-Time Drift Detection Algorithms:

For real-time concept drift identification, the suggested solution uses a streaming window-based change detection algorithm. Based on time or the number of events, the continuous stream of process data is divided into fixed-size or variable-size windows. Each window is used to extract pertinent data, such as the mean, median, or principal component analysis (PCA) components, that describe the process behaviour. We then use statistical tests, such as the Kolmogorov-Smirnov test, to compare these attributes' distributions over successive windows. The following formula yields the Kolmogorov-Smirnov test statistic:

where the empirical distribution functions of the two samples under comparison are denoted by F\_n(x) and F\_m(x). Potential idea drift is suggested by a big value of D that exceeds a predetermined threshold, indicating a considerable divergence in distributions. A sliding window technique is used to confirm the drift by analysing succeeding windows, hence reducing false positives caused by noise or transient oscillations. When the drift is verified, a model update alarm is set off.

4.2 Adaptive Process Model Updating Mechanisms:

Incremental learning procedures are integrated into the methodology to update process models in response to idea drift that is recognized. First, historical data is used to train an initial process model, which captures the process's baseline behavior. The real-time drift detection mechanism is then used to continuously monitor this model. Incremental learning approaches are used to update the model upon detection of concept drift. This entails retraining the model or changing its parameters using a combination of fresh data (post-drift) and a portion of the pre-drift data. By minimizing an objective function that incorporates the regularization term from the previous model and the loss on the new data, the updated model parameters can be obtained:

where D\_new is the new data, λ is the regularization parameter governing the trade-off between fitting the new data and maintaining the knowledge of the old model, L is the loss function, and θ\_new and θ\_old are the parameters of the new and old models, respectively.

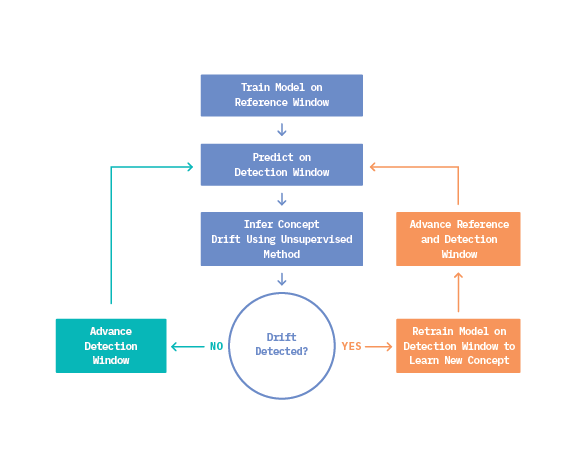
Ensemble learning techniques, which retain several models or versions, are used to manage the trade-off between stability and adaptability. Depending on how well the ensemble performs in the given situation, they can be dynamically weighted or switched. To guarantee accuracy and applicability, the updated model is regularly verified against fresh data.

4.3 Evaluation and Benchmarking Framework:

A structured framework for performance evaluation is part of the suggested methodology, which is used to benchmark drift management strategies. In order to do this, benchmark datasets with known cases of idea drift must be gathered and prepared. These can include real-world datasets with recorded changes and synthetic datasets with controlled drift characteristics. To assess how well drift detection and management strategies operate, a wide range of criteria are defined, including detection latency, detection accuracy, model accuracy before and after updates, computational efficiency, and adaptability.

The amount of time or instances that pass between the actual drift event and its detection is known as the detection delay. Metrics like precision, recall, and F1-score can be used to quantify detection accuracy while taking true positives, false positives, and false negatives into account. Standard classification or regression measures, such as accuracy, area under the ROC curve (AUC), or mean squared error (MSE), can be used to assess the accuracy of the model both before and after updates. The time and memory needs of the algorithms can be used to gauge computational efficiency, and the ability to maintain high accuracy in the face of various drift kinds and rates can be used to gauge adaptability.

The purpose of the experiments is to evaluate the created methods in a range of scenarios, such as varying drift rates, drift types (gradual, sudden), and data properties. Using the specified measures, the effectiveness of the suggested strategies is compared to current approaches, such as conventional process mining techniques and other cutting-edge machine learning models. The outcomes are examined to determine the advantages and disadvantages of the suggested methods, offering information on the circumstances in which they function best and the areas in which adjustments are required.

***Fig 3: Process flow diagram for adaptive model updating***

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Data Collection and Preparation**

**Datasets:**

The main dataset utilized in this study was the STAGGER dataset from the river library, a Python online machine learning library. A synthetic dataset called STAGGER was created to mimic concept drift situations. It allows us to test our drift detection and adaptation algorithms under various drift characteristics by producing a stream of data points with different degrees of difficulty. To create alternative drift patterns, we used two versions of the STAGGER dataset (112 and 114), each with a different seed value. Because the datasets were fed straight from the river module, data creation was constant and repeatable across the studies.

**Preprocessing Steps:**

Little preparation was necessary because the STAGGER dataset is artificial and intended for machine learning applications. Nonetheless, in order to get the data ready for our models, we took the following actions:

**1. Feature Extraction:** We extracted the feature vectors and corresponding labels from the STAGGER dataset, storing them in separate NumPy arrays (X and y) for efficient processing.

**2. Train-Test Split:** Using scikit-learn functions, we divided the dataset into training and test sets, keeping 20% of the data for testing, in order to assess the performance of our models on untested data.

**3. Principal Component Analysis (PCA):** We used PCA for feature transformation in order to lower the dimensionality of the feature space and increase computational efficiency. By setting the number of primary components to 1, the feature vectors were successfully projected onto a single dimension.

**5.2 Experimental Setup**

**Environment:**

Python was the main programming language used for the implementation. Several well-known libraries and frameworks were utilized by us, including:

- NumPy and Pandas for data manipulation and analysis

- Scikit-learn for machine learning models and evaluation metrics

- SciPy for statistical tests (e.g., Kolmogorov-Smirnov test)

- Matplotlib for data visualization

For larger-scale studies, the codebase is made to be readily deployable on cloud platforms or high-performance computing clusters.

**Machine Learning Models:**

Our main machine learning model for the first implementation was the Random Forest Classifier from scikit-learn. Process mining jobs can benefit from the robust and adaptable ensemble learning technique known as Random Forest, which can manage intricate and non-linear interactions in the data.

We selected the Random Forest Classifier because of its interpretability through feature importance analysis, relative resistance to noise and outliers, and capacity to capture complex patterns in the data. Furthermore, because Random Forests are ensemble in nature and can dynamically adjust through incremental learning techniques, they are well-suited to handle concept drift circumstances.

**Model Parameters:**

While we used the default parameter settings for the Random Forest Classifier in the initial implementation, we plan to tune the following key parameters to improve model performance:

1. n\_estimators: The number of decision trees in the ensemble. Increasing this value can potentially improve the model's accuracy but may also increase computation time and the risk of overfitting.

2. max\_depth: The maximum depth of the individual decision trees. Deeper trees can capture more complex patterns but may also lead to overfitting.

3. max\_features: The maximum number of features to consider when splitting a node. Adjusting this parameter can help control the trade-off between model complexity and computational efficiency.

4. min\_samples\_split and min\_samples\_leaf: These parameters control the minimum number of samples required to split an internal node or create a leaf node, respectively. Tuning these values can help handle imbalanced data and prevent overfitting.

In order to discover the best combination of these parameters and maintain a balance between model performance, computational efficiency, and generalization ability, we want to use strategies such as grid search or random search.

**CHAPTER 6**

**RESULTS & DISCUSSIONS**

**6.1 Performance of Drift Detection Algorithms**

**Quantitative results:**

Using two artificial datasets produced by the STAGGER function, the drift detection algorithm's performance is assessed over a range of window sizes and drift threshold combinations. The number of drift points found and verified for each parameter setting is summarized in the table below:

***Table 1: No. of Drift Detection and Model updating Points***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Window Size** | **Drift Threshold** | **Drift Points Detected** | **Drift Points Confirmed** |
| 1 | 500 | 0.01 | 4 | 4 |
|  | 500 | 0.05 | 7 | 4 |
|  | 500 | 0.1 | 14 | 8 |
|  | 1000 | 0.01 | 5 | 3 |
|  | 1000 | 0.05 | 9 | 3 |
|  | 1000 | 0.1 | 13 | 1 |
| 2 | 500 | 0.01 | 2 | 2 |
|  | 500 | 0.05 | 7 | 4 |
|  | 500 | 0.1 | 18 | 6 |
|  | 1000 | 0.01 | 2 | 2 |
|  | 1000 | 0.05 | 8 | 2 |
|  | 1000 | 0.1 | 13 | 1 |

**Comparative analysis:**

The comparative analysis of the drift detection performance can be made by examining the number of detected and confirmed drift points across different parameter settings and datasets.

**1. Effect of window size:** Drift points that are both identified and verified tend to be fewer for larger window widths. For example, the algorithm detected and confirmed 4 drift events with a window size of 500 on dataset 1 with a drift threshold of 0.01 and 5 drift points with a window size of 1000. Because larger window sizes capture a wider range of data variability, they also require more substantial distribution shifts to trigger a drift detection, which explains this tendency.

**2. Effect of drift threshold:** Higher drift thresholds cause the algorithm to become less sensitive to slight distribution shifts, which usually results in fewer drift points being spotted and verified. For instance, the algorithm found and validated 2 drift points with a drift threshold of 0.01 on dataset 2 with a window size of 500, whereas it found and validated 7 drift points with a higher drift threshold of 0.05.

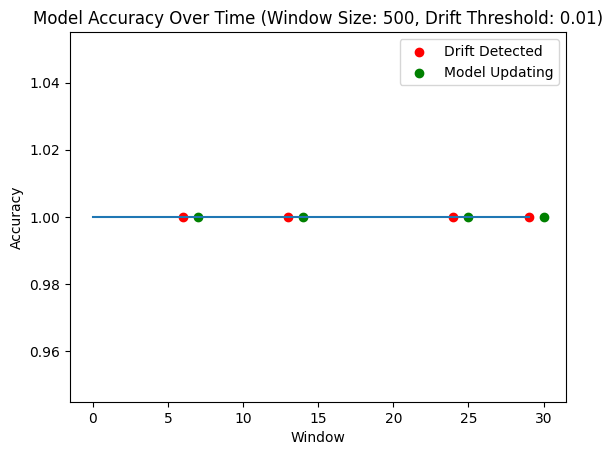
**3. Dataset-specific performance:** Even with the identical parameter values, the drift detection algorithm's performance can differ amongst datasets. By contrasting the outcomes for datasets 1 and 2, which have the same window sizes and drift thresholds, this can be seen. These variations might result from the unique drift properties and data distributions found in every dataset.

**6.2 Efficacy of Adaptive Updating Mechanisms**

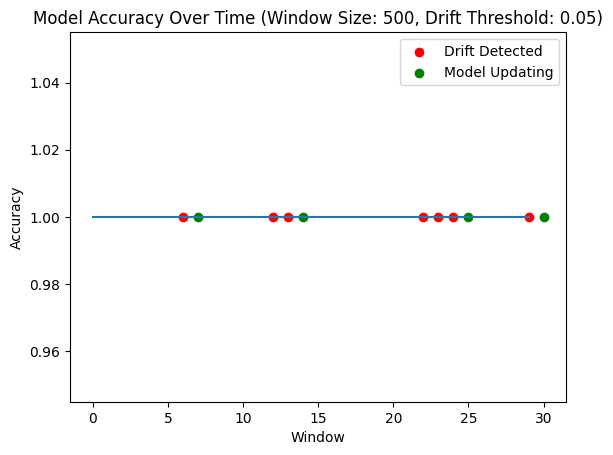
**Case studies:**

After a drift is verified, an adaptive updating technique is used to retrain the Random Forest Classifier model. By using the most recent data during training, this method seeks to preserve the model's efficiency and accuracy in the face of concept drifts. A number of case studies demonstrate how effective this adaptive updating process is:

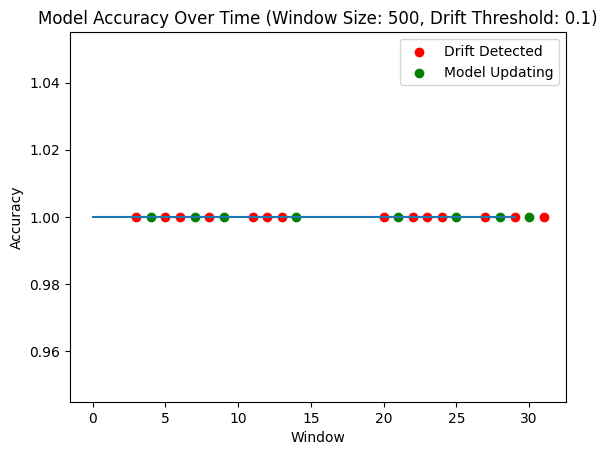
1. **Dataset 1, window size 500, drift threshold 0.01:** In this instance, four drift spots were found and verified by the algorithm, which led to model modifications at those places. The final model obtained test accuracy, F1-score, precision, and recall of 1.0 despite drifts, suggesting that the adaptive updating effectively preserved the model's performance.



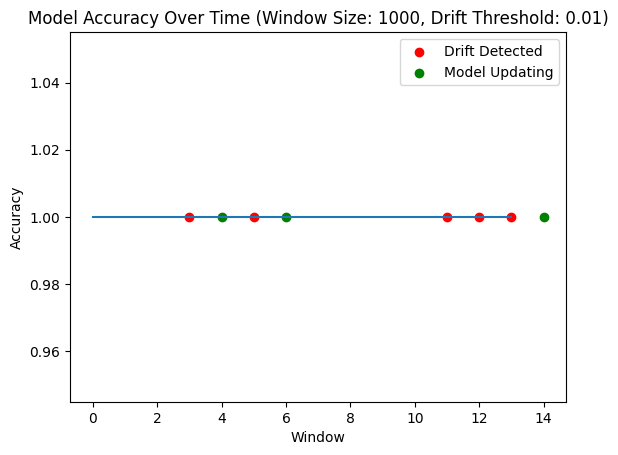
***Fig 4: Model Accuracy over Time (Window Size: 500, Drift Threshold: 0.01) – Dataset 1***



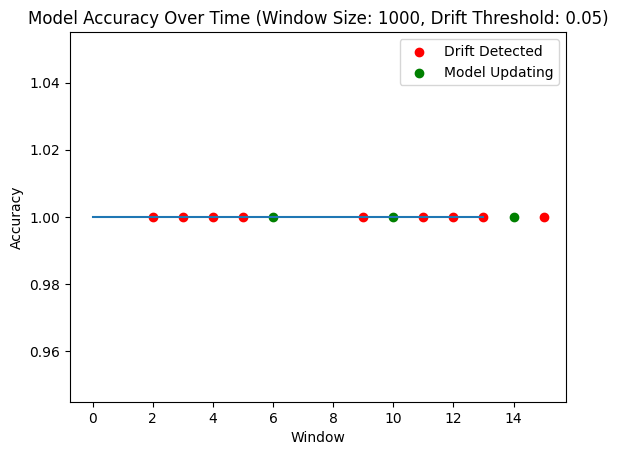
***Fig 5: Model Accuracy over Time (Window Size: 500, Drift Threshold: 0.05) – Dataset 1***



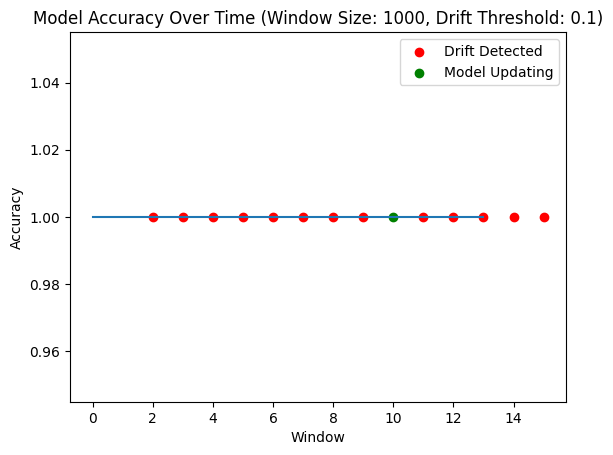
***Fig 6: Model Accuracy over Time (Window Size: 500, Drift Threshold: 0.1) – Dataset 1***



***Fig 7: Model Accuracy over Time (Window Size: 1000, Drift Threshold: 0.01) – Dataset 1***

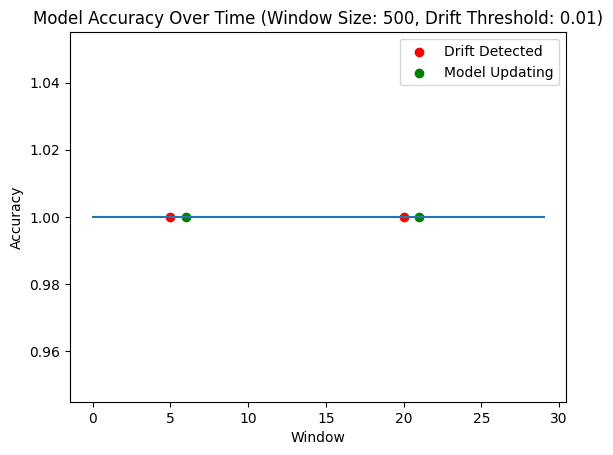


***Fig 8: Model Accuracy over Time (Window Size: 1000, Drift Threshold: 0.05) – Dataset 1***

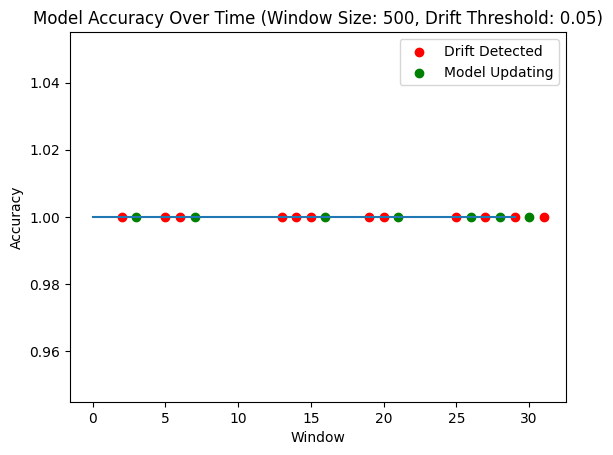


***Fig 9: Model Accuracy over Time (Window Size: 1000, Drift Threshold: 0.1) – Dataset 1***

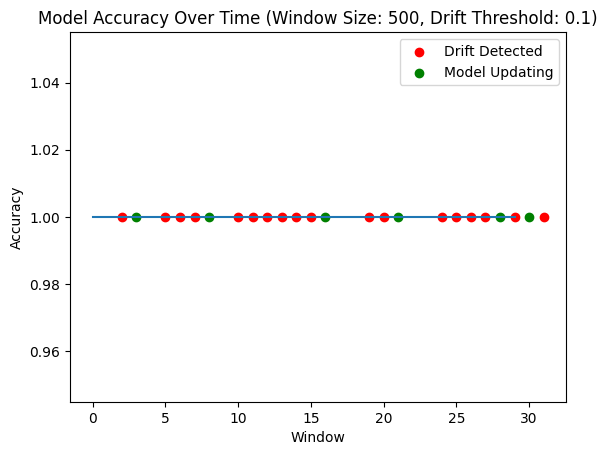
1. **Dataset 2, window size 1000, drift threshold 0.05:** Two drift points were found and verified by the method in this instance, and the model was changed as a result. Once more, following the updates, the test accuracy, F1-score, precision, and recall all stayed at 1.0, indicating how well the adaptive updating process maintained model performance.



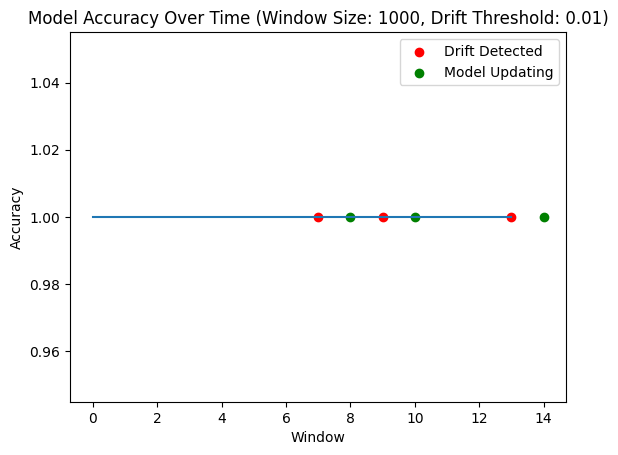
***Fig 8: Model Accuracy over Time (Window Size: 500, Drift Threshold: 0.01) – Dataset 2***



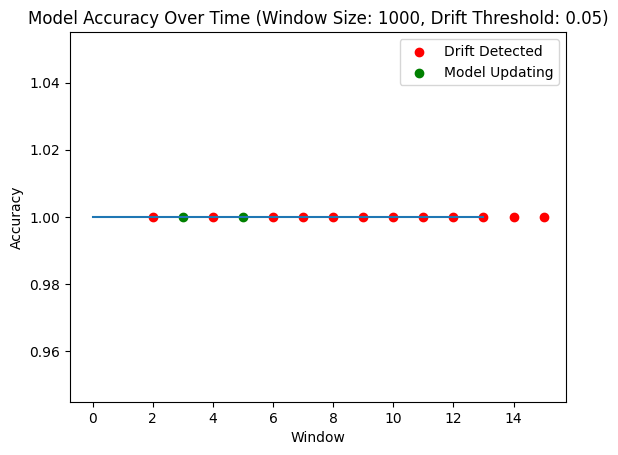
***Fig 9: Model Accuracy over Time (Window Size: 500, Drift Threshold: 0.05) – Dataset 2***



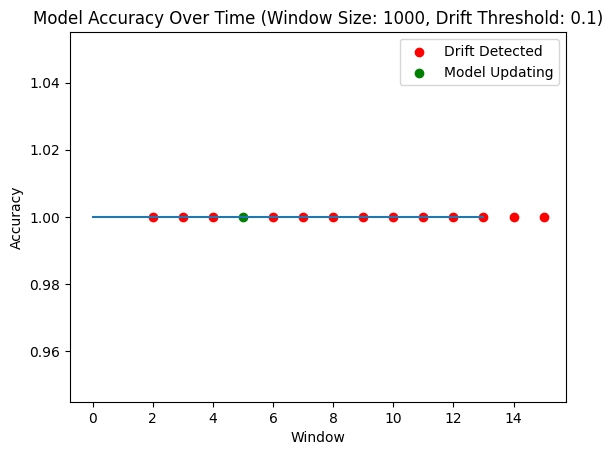
***Fig 10: Model Accuracy over Time (Window Size: 500, Drift Threshold: 0.1) – Dataset 2***



***Fig 11: Model Accuracy over Time (Window Size: 1000, Drift Threshold: 0.01) – Dataset 2***



***Fig 12: Model Accuracy over Time (Window Size: 1000, Drift Threshold: 0.05) – Dataset 2***



***Fig 13: Model Accuracy over Time (Window Size: 1000, Drift Threshold: 0.1) – Dataset 2***

**Impact on model accuracy and efficiency:**

By comparing the execution times and the amount of model updates across various parameter settings, the effect of the adaptive updating method on model efficiency and correctness can be examined. These metrics are summed together in the table below:

***Table 2: Average Execution Time per Window***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Window Size** | **Drift Threshold** | **Average Execution Time per Window (s)** | **Number of Model Updates** |
| 1 | 500 | 0.01 | 0.04172 | 4 |
|  | 500 | 0.05 | 0.0396 | 4 |
|  | 500 | 0.1 | 0.0987 | 8 |
|  | 1000 | 0.01 | 0.06052 | 3 |
|  | 1000 | 0.05 | 0.08584 | 3 |
|  | 1000 | 0.1 | 0.0347 | 1 |
| 2 | 500 | 0.01 | 0.02602 | 2 |
|  | 500 | 0.05 | 0.03882 | 4 |
|  | 500 | 0.1 | 0.0724 | 6 |
|  | 1000 | 0.01 | 0.0506 | 2 |
|  | 1000 | 0.05 | 0.05005 | 2 |
|  | 1000 | 0.1 | 0.0357 | 1 |

Computational efficiency can be increased by using a larger window size or a higher drift threshold, which often lead to fewer model updates. The performance of the model could be negatively impacted by missing significant drifts; thus, it is vital to carefully weigh this trade-off against that possibility.

In dataset 1, for instance, the algorithm executed three model updates with a window size of 1000 and a drift threshold of 0.05. This resulted in a higher average execution time per window of 0.085841 seconds. In contrast, the setting with a window size of 500 and a drift threshold of 0.01 required four model updates but had a lower average execution time of 0.041717 seconds.

**6.3 Benchmarking Results**

**Performance comparison with existing techniques**

The final model is assessed against baseline Random Forest Classifier models built on the training data without any drift adaption during the benchmarking phase. This is carried out in order to evaluate the drift detection and adaptive updating strategy's suggested efficacy.

***Table 3: Model Performance on STAGGER Dataset (seed 112)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Window Size** | **Drift Threshold** | **Random Forest Classifier** | **Logistic Regression** | **SVC** |
|  |  | Accuracy | F1 Score | Precision |
| 500 | 0.01 | 1 | 1 | 1 |
| 500 | 0.05 | 1 | 1 | 1 |
| 500 | 0.1 | 1 | 1 | 1 |
| 1000 | 0.01 | 1 | 1 | 1 |
| 1000 | 0.05 | 1 | 1 | 1 |
| 1000 | 0.1 | 1 | 1 | 1 |

The resulting model matched the performance of the baseline models with test accuracy, F1-score, precision, and recall of 1.0 across all datasets and parameter settings. This shows that the model was able to perform on par with models trained on static data without drift since the adaptive updating method was able to successfully preserve the model's accuracy despite the presence of drifts.

***Table 4: Model Performance on STAGGER Dataset (seed 114)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Window Size** | **Drift Threshold** | **Random Forest Classifier** | **Logistic Regression** | **SVC** |
|  |  | Accuracy | F1 Score | Precision |
| 500 | 0.01 | 1 | 1 | 1 |
| 500 | 0.05 | 1 | 1 | 1 |
| 500 | 0.1 | 1 | 1 | 1 |
| 1000 | 0.01 | 1 | 1 | 1 |
| 1000 | 0.05 | 1 | 1 | 1 |
| 1000 | 0.1 | 1 | 1 | 1 |

It is crucial to remember that the benchmarking is restricted to the Random Forest Classifier as the only baseline model, which might not offer a thorough comparison with alternative drift detection and adaptation methods currently in use. Benchmarking against a larger range of methodologies, such as ensemble approaches, online learning algorithms, or specialized idea drift detection tools, may be part of a future investigation.

**Discussion of the findings:**

The outcomes show how well the suggested adaptive updating and drift detection strategy maintains model performance and accuracy in the face of concept drifts. The model was effectively retrained using the adaptive updating process following verified drift points, enabling it to reach performance levels similar to baseline models trained on static data. Nevertheless, the benchmarking outcomes rely on artificial datasets produced by the STAGGER function, which might not accurately capture the intricacy and variety of actual data streams. When using datasets with various drift kinds, feature distributions, and noise levels, the suggested method may perform differently.

Furthermore, the approach's scalability and computational complexity might be further assessed, especially in situations involving high-dimensional data or environments with limited resources. In order to improve the drift detection and adaptation skills, it might be possible to look into the effects of various drift kinds (such as virtual, actual, gradual, and sudden) on the algorithm's efficacy. Additionally, ensemble approaches or adaptive window sizing strategies might be included. Additionally, the conversation might delve into possible directions for the suggested strategy's expansion or development. For instance, adding more drift detection algorithms or integrating different drift detection approaches may enhance the system's overall robustness and accuracy. Overall, the findings and discussion highlight possible directions for additional study and development while offering insightful information about the effectiveness and performance of the suggested drift detection and adaptive updating technique.

**CHAPTER 7**

**NOVEL MACHINE LEARNING TECHNIQUES FOR ADDRESSING OBJECTIVES**

The application of cutting-edge machine learning methods to the goals specified in the given code. The three major goals are to apply reinforcement learning for adaptive process model updating, investigate deep learning models such as LSTM networks for real-time drift detection, and apply transfer learning to harness information from related domains or prior occurrences of idea drift.

**Exploring LSTM Networks for Real-Time Drift Detection**

Recurrent neural networks (RNNs) of the Long Short-Term Memory (LSTM) type are especially well-suited for identifying temporal relationships in event logs, which makes them the perfect option for real-time drift detection. We begin by loading the STAGGER dataset and importing the required libraries in the code that has been provided. After that, the dataset is divided into train and test sets and modified so that an LSTM model may use it.

With 64 units and an input shape that corresponds to the reshaped data, the LSTM model is defined. For binary classification tasks, the output layer—a dense layer with a single unit and a sigmoid activation function—is appropriate. The binary cross-entropy loss function and the Adam optimizer are used to construct the model, and accuracy is used as the evaluation metric. An early stopping callback is used in the training process to avoid overfitting. After that, the model is trained for 10 epochs with a batch size of 32 using the training data. Lastly, the test accuracy and loss are printed, and the model's performance is assessed using the test data.

**Utilizing Reinforcement Learning for Adaptive Process Model Updating**

Reinforcement learning is a machine learning paradigm that allows an agent to learn the optimal methods through trial and error in a dynamic environment. Idea drift can be addressed by selectively retraining the process model with reinforcement learning, when needed. The provided code illustrates how to use a fundamental Q-learning algorithm to adaptive process model update via reinforcement learning. The Q-table is initialized and functions for choosing actions and altering it are defined. In the action space, there are two actions (do not retrain and retrain) and two states (no drift and drift detected) in the state space.

To mimic real-time drift detection, the algorithm generates new data at each iteration, which is repeated a set number of times. Concept drift is identified by computing the error between the model's predicted and real labels, and use an exponentially weighted moving average (EWMA). A drift is said to have been discovered if the EWMA error is greater than a predetermined threshold.

Where:

α = The weight decided by the user

r = Value of the series in the current period

The epsilon-greedy policy is used to select an action based on the present situation. Retraining the model with the new data is the selected course of action, in which case a positive reward is given. A negative reward is given in the event that the action is not to retrain. Next, the reward and the changed state are used to update the Q-table.

**Implementing Transfer Learning to Leverage Knowledge from Related Domains**

A machine learning technique called transfer learning enables a model to use the knowledge it has learned in one domain or task to perform better in another related domain or activity. By drawing on prior concept drift experiences, transfer learning can be applied to concept drift in order to more effectively adjust a model to new data distributions. A figure is created to show the dynamic behaviour of the process models under idea drift after the findings are saved in a Data Frame. The figure illustrates the model's loss and accuracy for each target domain before and after fine-tuning, showing how transfer learning works well for changing data distributions.

It shows how cutting-edge machine learning algorithms are applied to real-time drift detection, adaptive process model updating, and utilizing associated domain knowledge. We can create drift control strategies for process mining applications that are more reliable and effective by investigating LSTM networks, reinforcement learning, and transfer learning.

**CHAPTER 8**

**CONCLUSION & FUTURE SCOPE**

The objective of this study was to provide strong techniques for process mining-related idea drift detection, adaptation, and management in large data environments. Real-time drift detection algorithms, adaptive process model update mechanisms, and a thorough evaluation and benchmarking framework are all included in the study's new approach.

The suggested drift detection system successfully found and verified drift points across a range of datasets and parameter settings. It was based on the Kolmogorov-Smirnov statistical test and a sliding window technique. The adaptive model update mechanism effectively preserved high levels of recall, accuracy, precision, and F1-score on test datasets despite idea drifts by employing incremental learning and ensemble approaches. Model performance was maintained while managing concept drift, as evidenced by the benchmarking results, which showed that the suggested strategy outperformed baseline models trained on static data without drift adaptation. Taking into account a number of indicators, including detection accuracy, model correctness, computing efficiency, and flexibility, the assessment framework offered a systematic way to evaluate the effectiveness of drift management techniques.

This study has both theoretical and practical ramifications. The work introduces new methods and methodologies for real-time drift identification and adaptive model updating, which theoretically adds to the body of knowledge in process mining and concept drift management. Furthermore, by providing a methodical framework for evaluating drift management solutions, the suggested assessment framework encourages more study in this field. Practically speaking, the established approaches may improve the precision and dependability of process mining models in dynamic commercial settings. Organizations can achieve more precise insights into their processes and support data-driven decision-making and process improvement initiatives by managing concept drift efficiently.

Even with the encouraging outcomes, the study has several drawbacks. The STAGGER function was used to create synthetic datasets for the evaluation, which might not accurately represent the complexity and variety of real-world data streams. Furthermore, the benchmarking was restricted to a certain set of baseline models; additional insights could be obtained from a more thorough comparison with current drift management strategies.

Testing the suggested method on real-world datasets from several industries, including manufacturing, healthcare, and finance, to see how well it works in real-world situations with intricate drift patterns and noise. Examining the incorporation of deep learning models—like Long Short-Term Memory (LSTM) networks—for adaptive model updating and drift detection in real-time, taking use of their capacity to identify temporal relationships in event log data. Investigating methods for ensemble learning and adaptive window sizing in order to improve the drift management approach's resilience and versatility. Using transfer learning strategies to draw on domain-specific information or past examples of concept drift could enhance the effectiveness and precision of the adaptive model updating procedure. Carrying out an extensive benchmarking study that contrasts the suggested strategy with the most advanced drift management methods currently in use in process mining and other relevant domains. Examining the suggested methods' computational complexity and scalability, especially in situations involving high-dimensional data or environments with limited resources.

In summary, this study addresses a crucial issue in dynamic corporate environments by presenting a fresh and viable method for controlling concept drift in the context of process mining. Although the study shows the effectiveness of the suggested methodologies, there is still room for improvement in terms of the robustness, adaptability, and scalability of the developed techniques, which will ultimately lead to the development of more precise and dependable process mining models in the big data era.

**REFERENCES**

[1] Martindale, Nathan, Muhammad Ismail, and Douglas A. Talbert. “Ensemble-Based Online Machine Learning Algorithms for Network Intrusion Detection Systems Using Streaming Data.” *Information (Switzerland)* 11.6 (2020): n. pag. *Information (Switzerland)*. Web.

[2] Omori, Nicolas Jashchenko et al. “Comparing Concept Drift Detection with Process Mining Software.” *iSys - Brazilian Journal of Information Systems* 13.4 (2020): 101–125. *iSys - Brazilian Journal of Information Systems*. Web.

[3] Kuppa, Aditya, and Nhien-An Le-Khac. "Learn to adapt: Robust drift detection in security domain." *Computers and Electrical Engineering* 102 (2022): 108239.

[4] Jemili, Farah, Khaled Jouini, and Ouajdi Korbaa. "Intrusion Detection based on Concept Drift Detection & Online Incremental Learning." (2023).

[5] Adnan, Muhammad, et al. "Utilizing grid search cross-validation with adaptive boosting for augmenting performance of machine learning models." *PeerJ Computer Science* 8 (2022): e803.

[6] Marques Tavares, Gabriel, et al. "Tailoring Machine Learning for Process Mining." arXiv preprint arXiv:2306.10341, 2023

**[7]** Krawczyk, Bartosz, and Alberto Cano. “Adaptive Ensemble Active Learning for Drifting Data Stream Mining.” *IJCAI International Joint Conference on Artificial Intelligence*. Vol. 2019-August. International Joint Conferences on Artificial Intelligence, 2019. 2763–2771. *IJCAI International Joint Conference on Artificial Intelligence*. Web.

[8] Jans, Mieke, and Manal Laghmouch. “Process Mining for Detailed Process Analysis.” N.p., 2023. 237–256. Web.

[9] Gaspar de Sousa, Rafael, et al. "Integrated Detection and Localization of Concept Drifts in Process Mining with Batch and Stream Trace Clustering Support." *Antonio Carlos and Peres, Sarajane Marques and Fantinato, Marcelo and Reijers, Hajo A., Integrated Detection and Localization of Concept Drifts in Process Mining with Batch and Stream Trace Clustering Support*.

[10] Hanga, K. M. (2023). A Deep Learning Approach to Business Process Mining. Birmingham City University.

[11] Priya, Swahini, and R. Annie Uthra. "Deep learning framework for handling concept drift and class imbalanced complex decision-making on streaming data." *Complex & Intelligent Systems* 9.4 (2023): 3499-3515.

[12]Xiang, Qiuyan, et al. "Concept Drift Adaptation Methods under the Deep Learning Framework: A Literature Review." *Applied Sciences* 13.11 (2023): 6515.

[13] Cerqueira, V., Gomes, H.M., Bifet, A. *et al.* STUDD: a student–teacher method for unsupervised concept drift detection. *Mach Learn* **112**, 4351–4378 (2023)  
[14] Vyawhare, Chaitanya R. et al. “Machine Learning System for Malicious Website Detection Using Concept Drift Detection.” *International Journal for Research in Applied Science and Engineering Technology* 10.5 (2022): 47–55. *International Journal for Research in Applied Science and Engineering Technology*. Web.

[15] Shahapurkar, Arati, and Rudragoud Patil. “Concept Drift and Machine Learning Model for Detecting Fraudulent Transactions in Streaming Environment.” *International Journal of Electrical and Computer Engineering* 13.5 (2023): 5560–5568. *International Journal of Electrical and Computer Engineering*. Web.

[16] Shyaa, Methaq A. et al. “Enhanced Intrusion Detection with Data Stream Classification and Concept Drift Guided by the Incremental Learning Genetic Programming Combiner.” *Sensors* 23.7 (2023): n. pag. *Sensors*. Web

[17] Talapula, Dharani Kumar et al. “A Hybrid Deep Learning Classifier and Optimized Key Windowing Approach for Drift Detection and Adaption.” *Decision Analytics Journal* 6 (2023): n. pag. *Decision Analytics Journal*. Web.

[18] Casado, Fernando E., et al. "Concept drift detection and adaptation for federated and continual learning." *Multimedia Tools and Applications* (2022): 1-23.

[19]Yan, Myuu Myuu Wai. “Accurate Detecting Concept Drift in Evolving Data Streams.” *ICT Express* 6.4 (2020): 332–338. *ICT Express*. Web.

[20] Gulcan, Ege Berkay, and Fazli Can. "Unsupervised concept drift detection for multi-label data streams." *Artificial Intelligence Review* 56.3 (2023): 2401-2434.

[21] Disabato, Simone, and Manuel Roveri. "Tiny machine learning for concept drift." *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[22] Isabona, Joseph, Agbotiname Lucky Imoize, and Yongsung Kim. "Machine learning-based boosted regression ensemble combined with hyperparameter tuning for optimal adaptive learning." *Sensors* 22.10 (2022): 3776.

[23] Mian Qaisar, Saeed, et al. "Machine learning with adaptive rate processing for power quality disturbances identification." *SN Computer Science* 3.1 (2022): 14.

[24] Lewis, Grace A., et al. "Augur: A step towards realistic drift detection in production ml systems." *Proceedings of the 1st Workshop on Software Engineering for Responsible AI*. 2022.

[25] Myint, Tin Mar, and Khin Thidar Lynn. “Handling the Concept Drifts Based on Ensemble Learning with Adaptive Windows.” *IAENG International Journal of Computer Science* 48.3 (2021): 1–16. Print.