DESIGN AND ANALYSIS OF CLASSIFICATION FRAMEWORK FOR STREAMING DATA

by Seba Susan

Submission date: 03-May-2026 04:17PM (UTC+0530)
Submission ID: 2950478208
File name: short_thesis.docx (5.84M)
Word count: 39361
Character count: 235852

CHAPTER 1

INTRODUCTION

This chapter provides an overview of the current research landscape, highlighting the key problems and underlying motivations that inspired this study. The primary focus is on understanding the challenges associated with processing and analyzing streaming data in real-time environments, where data arrives continuously and evolves dynamically. We discuss the significance of developing adaptive machine learning models capable of handling concept drift, high-dimensional data, and limited labeled instances, which are common issues in streaming scenarios.

Furthermore, the chapter outlines the scope of the research, giving a clear perspective on the specific areas addressed, including online classification, ensemble learning, and anomaly detection in dynamic data streams. It also identifies existing research gaps, such as the lack of scalable frameworks that can adapt swiftly to changing data distributions while maintaining high accuracy. The unique contributions of this work are emphasized, particularly in designing robust online models, integrating ensemble techniques for improved stability, and devising effective anomaly detection methods tailored for evolving environments.

The chapter continues with a detailed discussion of the experimental setup, describing the datasets used, the methodologies employed in developing online machine learning algorithms, and the strategies implemented for model evaluation. The research involves constructing online classifiers capable of incremental learning, developing an ensemble-based classification framework to enhance robustness, designing anomaly detection mechanisms suited for non-stationary data, and rigorously assessing the performance of these approaches using real-time streaming datasets. The overarching goal of this research is to advance the state-of-the-art in online learning by improving adaptability, computational efficiency, and predictive accuracy in real-world streaming applications. Ultimately, this work aims to contribute valuable insights and practical solutions for deploying intelligent systems in dynamic, data-driven environments.

1.1 Overview

The rapid proliferation of real-time applications across various domains, such as social media platforms, financial trading systems, sensor networks, healthcare monitoring, and the Internet of Things (IoT), has resulted in the continuous flow of massive volumes of streaming data. Unlike traditional static datasets, streaming data is inherently unbounded, arrives sequentially over time, and undergoes constant evolution. These characteristics give rise to the phenomenon known as concept drift, where the underlying data distribution changes unexpectedly or gradually, posing significant challenges for machine learning models (Gama et al., 2014 [5]; Putatunda et al., 2021 [4]).

Conventional batch learning methods, which rely on fixed training datasets, are often ill-suited for such environments because they require repeated retraining and are computationally expensive, making them impractical for real-time applications (Rechard et al., 2022 [1]). In contrast, online learning algorithms are designed to update their models incrementally as new data arrives, enabling continuous adaptation to changing data patterns (Hoi et al., 2021 [236]). Despite this advantage, maintaining high classification accuracy and robustness in the face of evolving data distributions remains an ongoing challenge.

In addition to classification, anomaly detection is crucial for identifying unusual or suspicious patterns in streaming environments, especially in critical applications like network security, financial fraud detection, and IoT device monitoring. Many existing anomaly detection techniques rely on static thresholds or prior domain knowledge, which are often inadequate because anomalies in dynamic systems can be unknown, rare, and constantly changing (Montiel et al., 2021 [2]; Putatunda et al., 2021 [3]). Unsupervised approaches, which do not depend on labeled data, are better suited for such scenarios but require careful design to handle the scale and complexity of real-time data streams effectively.

Given these challenges, this research aims to develop an adaptive, scalable classification framework tailored for streaming data environments. The proposed approach integrates ensemble learning methods with real-time concept drift detection and robust anomaly detection techniques. By combining these strategies, the framework seeks to deliver accurate, reliable, and timely insights in highly dynamic settings. The study also emphasizes the importance of designing models that can adapt swiftly to new patterns, detect anomalies with minimal false alarms, and operate efficiently under resource constraints, ultimately contributing to the advancement of intelligent, real-time decision-making systems.

1.2 Problem Statement

With the exponential growth of real-time data streams, traditional classification systems struggle to keep pace, as they rely on static models that become outdated quickly. These conventional methods face issues such as high memory consumption, frequent retraining, limited adaptability to changing data patterns (concept drift), and inadequate performance in real-time scenarios. Although online machine learning offers a solution by enabling models to learn incrementally, it alone may not fully address the complexities of evolving data environments.

In summary, current data-driven systems generate an endless flow of information, making traditional classification methods inefficient due to their dependence on static learning. While online learning supports continuous adaptation, there remains a need for more advanced, scalable frameworks.

- **Limitations of Traditional Approaches:** Excessive memory use, need for constant retraining, poor adaptability, and lack of real-time responsiveness.
- **Advantages of Online Learning:** Supports incremental updates, reduces memory requirements, and enables real-time predictions.
- **Need for an Improved Framework:** An efficient, adaptive, and scalable system capable of handling concept drift and enhancing anomaly detection in streaming data.
- **Addressing Concept Drift:** As data distributions evolve over time, models must dynamically adapt without losing accuracy, ensuring reliable performance over prolonged periods.

Developing such flexible and scalable systems is essential for maintaining high accuracy in dynamic environments. These models should support continuous learning with minimal resource usage, allowing immediate response to changing data patterns and ensuring robustness against concept drift. Such advancements are vital for improving real-time analytics, anomaly detection, and decision-making across various domains.

1.3 Research Gaps

Based on the literature survey, several key research gaps are identified:

1. Many methods work only for specific concept drift types, like sudden or gradual, and can't handle multiple simultaneously, reducing effectiveness in dynamic real-world scenario environments.

2. Another problem is the scalability issue, as many techniques struggle to meet the demands of high-dimensional and large datasets.
3. Many current methods lack efficient online learning, needing frequent retraining on large datasets, which harms computational efficiency.
4. Few studies address both concept drift and anomaly detection in streaming network traffic.
5. Limited research has explored hybrid ensemble frameworks using River and PySAD for anomaly detection in streaming environments.

1.4 Scope of Work

The scope of this study focuses on developing an adaptive and efficient methodology for classification and anomaly detection within real-time streaming data environments. The proposed approach is capable of handling concept drift and evolving data distributions collectively, ensuring robust performance. Additionally, it leverages online learning techniques to adapt dynamically to changes, supporting real-time data processing and analysis.

1.5 Research Objectives

The classification framework for streaming data developed in this thesis focuses on handling evolving data distributions, detecting anomalies, and improving real-time learning performance. The following are the research objectives, which are described in Chapters 3–6:

- Performance evaluation of online learning strategies on real-time streaming data (Chapter-3)
- Development of ensemble classification framework for streaming data (Chapter-4)
- Design of novel anomaly detection technique for evolving data streams (Chapter-5)
- Ensemble-based framework for concept drift adaptation and network intrusion detection in streaming data (Chapter-6)

Overall, this thesis addresses key challenges in streaming data by proposing adaptive classification and anomaly detection approaches, with a focus on improving performance and reliability in real-time environments.

1.6 Research Motivation

The increasing prevalence of data streams necessitates the development of efficient and accurate learning methods capable of supporting real-time decision-making across various applications. A key challenge in such environments is the occurrence of concept drift, which can significantly impair predictive performance if not properly managed. Additionally, anomaly detection in streaming data is critical for identifying irregular events such as fraud, cyber-attacks, or system malfunctions; however, this task is complicated by the continuously evolving nature of the data Aggarwal, 2017 [237]). Despite numerous proposed approaches, effectively integrating drift-aware ensemble techniques with robust anomaly detection remains an open research problem.

Moreover, the proliferation of real-time data has highlighted the limitations of traditional machine learning algorithms, including high memory consumption, frequent retraining requirements, limited adaptability, and inadequate support for real-time processing. These issues underscore the need for learning paradigms that can process data incrementally without dependence on static datasets. Online learning methods address these needs by enabling continuous model updates, reducing computational resources, and facilitating real-time predictions. Nonetheless, the presence of evolving data distributions and anomalies introduces additional complexities, necessitating the development of resilient, scalable, and adaptive frameworks.

Motivated by these challenges, this research aims to design a comprehensive solution that effectively addresses concept drift, enhances anomaly detection capabilities, and maintains reliable performance in streaming environments. This forms the core motivation and foundation of the work presented in this thesis.

1.7 Research Issues and Proposed Solutions

As highlighted in the previous section, there is a pressing need for advanced learning algorithms tailored for streaming data analysis. These algorithms must effectively perform classification and anomaly detection while accommodating evolving data distributions and concept drift. Addressing these challenges is crucial for developing reliable and efficient systems capable of operating in dynamic, real-time environments.

The structure of this thesis is organized into seven chapters. Chapter 2 provides a comprehensive review of the current state-of-the-art techniques in data stream mining and online learning. Chapters 3 through 6 present innovative solutions aimed at overcoming key challenges related to model adaptability, anomaly detection, and maintaining high performance in real-time, continuously changing scenarios.

The proposed solutions to the four core problems addressed in this thesis are summarized below:

1.7.1 Problem 1: Ensemble-Based Framework for Handling Concept Drift in Evolving Data Streams

In Chapter 4, we propose an Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF) designed to effectively address the challenges posed by concept drift in continuously evolving data streams. Learning from non-stationary data presents significant difficulties because the underlying data distribution changes over time, which can degrade the performance of static models. Traditional batch learning methods often assume data stationarity, limiting their applicability in real-time streaming environments where data dynamics are prevalent. To overcome this limitation, the proposed framework leverages ensemble learning techniques—such as Adaptive Random Forests (ARF) [6], coupled with multiple drift detection algorithms to enhance adaptability and predictive accuracy.

The framework incorporates a suite of state-of-the-art drift detection methods, including the Drift Detection Method (DDM) [9], Reactive Drift Detection Method (RDDM) [10], Early Drift Detection Method (EDDM) [11], ADaptive WINdowing (ADWIN) [12], and Hoeffding-based approaches like HDDM_A and HDDM_W [13], in addition to the Fast-Hoeffding Drift Detection Method (FHDDM) [14]. These detectors enable the system to identify both abrupt and gradual concept drifts effectively. Multiple base learners are constructed by integrating different drift detection techniques with the ARF model, and their outputs are combined using dynamically updated weights based on real-time prediction errors. This adaptive weighting scheme ensures that the ensemble remains responsive to changes in data distribution, maintaining high classification performance over time.

Extensive experiments are conducted using both real-world datasets, such as weather data and LUdata, and synthetic datasets like sea_stream and sea_big. The experiments utilize tools such as Scikit-learn and River libraries [2, 19, 20], enabling robust evaluation of the proposed method. Results indicate that the EW-DA-EF significantly outperforms existing approaches such as PWPAE [16], particularly in

scenarios with multiple types of drift and under constraints on memory and computational resources [5, 15, 17, 18]. These findings demonstrate the framework's effectiveness in delivering accurate, adaptable, and real-time streaming data classification, making it highly suitable for practical applications in dynamic environments.

1.7.2 Problem-2: Anomaly Detection in Evolving Data Streams

In Chapter 5, we introduce a heterogeneous stacked-ensemble framework tailored for anomaly detection in evolving data streams. The core challenge lies in accurately identifying irregular patterns within continuously generated and dynamically changing data, which is critical for applications such as cybersecurity, fraud detection, and system monitoring. Traditional anomaly detection techniques are often designed for static datasets and depend on fixed thresholds or batch processing, which hampers their effectiveness in real-time streaming scenarios where data distributions evolve over time. To address this, our approach leverages online machine learning methods capable of detecting anomalies incrementally, ensuring timely responses to emerging threats or unusual behaviors.

The proposed framework uniquely combines multiple online anomaly detection models that cater to different types of anomalies, point, contextual, and collective, as classified by Chandola et al. [21]. Several online methods are incorporated, including One-Class Support Vector Machine (OCSVM) [22, 23] and ensemble-based approaches [24], which model normal behavior and identify deviations effectively. Despite their individual strengths, existing approaches often lack an integrated mechanism that consolidates diverse models to boost detection accuracy and adaptability in constantly evolving environments.

To fill this gap, our heterogeneous stacked-ensemble framework integrates ten diverse base learners drawn from two prominent online machine learning libraries: River and PySAD. The River models include HST, QF_OCSVM, TF_HST, QF_HST, and KNN, while the PySAD models encompass HBOS, KitNet, RRCF, IForestASD, and KNNCAD. Each model generates outlierness scores for incoming data instances, which are then fed into meta-learners such as Logistic Regression and AdaBoost. This meta-learning strategy intelligently combines the scores, enabling the system to adaptively weigh the contributions of different models and produce a robust final anomaly prediction [25, 26].

Extensive experimental evaluations are performed on nine benchmark streaming datasets, such as HTTP, CreditCard, SMTP, MaliciousURL, Android, covtype_1, cpu4, Falling, and Phishing, using evaluation metrics like ROCAUC,

Recall, and the Geometric Mean to assess detection performance under online learning conditions [20]. The results demonstrate that the proposed heterogeneous stacking ensemble significantly enhances anomaly detection accuracy, effectively handling the challenges of concept drift and evolving data distributions without reliance on fixed thresholds. Its adaptability and robustness make it particularly suitable for real-time applications demanding high reliability and low latency.

1.7.3 Problem-3: Assessment of Online Learning Techniques for Real-Time Data Streams

In Chapter 3, we undertake an in-depth evaluation of various online learning algorithms tailored for processing real-time data streams. The proliferation of sensor networks, Internet of Things (IoT) devices, and urban monitoring systems generates vast amounts of high-velocity data that require immediate analysis. Traditional batch learning methods, which necessitate access to the entire dataset during training, are ill-suited for such dynamic environments due to their computational cost and inability to adapt promptly to changing data distributions. Consequently, online learning approaches have gained prominence for their capability to incrementally update models and provide instant predictions as new data arrives.

This study emphasizes the application of online anomaly detection techniques in the context of real-time air quality monitoring, a critical aspect of environmental management amid rapid urbanization. Monitoring the Air Quality Index (AQI) is vital for safeguarding public health and maintaining ecological balance. The challenge involves detecting anomalies, such as sudden pollution spikes, promptly and accurately, despite the presence of high-dimensional, noisy, and often imbalanced data.

To address these challenges, nine unsupervised online anomaly detection models are evaluated, including Half Space Trees (HST) [28], Quantile Filter One-Class Support Vector Machine (QF_OCSVM), Quantile Filter Half Space Trees (QF_HST), Threshold Filter Half Space Trees (TF_HST), Histogram-Based Outlier Score (HBOS) [29], Kitsune's Core Algorithm (KitNet) [30], Robust Random Cut Forest (RRCF) [31], Isolation Forest for Anomaly Streaming Data (IForestASD) [32], and K-Nearest Neighbor Conformal Anomaly Detection (KNNCAD) [33]. These models are applied to AQI data collected over a year from 20 monitoring stations across Delhi, India, capturing diverse pollution scenarios.

The evaluation employs multiple metrics, including Accuracy, ROCAUC, Recall, and the Geometric Mean, to comprehensively assess detection performance.

Results reveal that models such as QF_OCSVM, QF_HST, and HBOS consistently outperform others across most measures, demonstrating their robustness and adaptability. The study further validates these findings on the IoTID20 streaming dataset, which presents additional complexities like high dimensionality, class imbalance, and limited labeled data.

Implementation leverages the River and PySAD frameworks, which are optimized for real-time data processing and model updating. Key challenges addressed include managing high data velocity, ensuring scalability, handling concept drift, and coping with scarce labels, common issues in real-world IoT environments. The experimental outcomes affirm that online learning strategies are highly effective for scalable, real-time analysis, enabling timely anomaly detection that can inform prompt decision-making in critical applications such as environmental monitoring, smart cities, and industrial IoT systems.

1.7.4 Problem-4: Online Drift-Aware Ensemble Framework for Real-Time Network Security

In Chapter 6, we introduce an innovative online ensemble framework designed to enhance real-time network security by effectively managing concept drift and detecting intrusions amidst evolving network traffic. Modern computer networks generate vast streams of data involving packet exchanges, routing decisions, and traffic management activities. These streams are characterized by rapid changes in traffic patterns and attack vectors, making traditional static machine learning models inadequate for timely threat detection. The increasing sophistication of cyber-attacks, coupled with the dynamic nature of network environments, necessitates adaptive, real-time solutions capable of handling these challenges.

The core challenge addressed by this framework is the seamless integration of online learning with drift detection and anomaly detection techniques to ensure accurate identification of malicious activities as network behaviors change over time. To this end, the proposed methodology combines robust drift detection mechanisms with an advanced streaming anomaly detection model, forming a comprehensive system capable of adapting to new threats without sacrificing detection accuracy.

A key component of the framework is the Incremental Packet-to-Vector (IP2V) streaming mechanism, which efficiently converts raw network packets into meaningful feature vectors suitable for real-time analysis. The ensemble incorporates an Adaptive Random Forest (ARF) classifier, renowned for its incremental learning

capabilities, and is augmented with multiple drift detection algorithms, including Early Drift Detection Method (EDDM), Drift Detection Method (DDM), ADaptive WINdowing (ADWIN), and Hoeffding-based approaches such as HDDM_A and HDDM_W. This dynamic ensemble allows the system to adaptively respond to changes in network traffic and attack patterns. Moreover, an error-sensitive model replacement strategy continuously evaluates individual model performance, replacing underperforming models with more accurate ones to maintain optimal detection capabilities.

The effectiveness of this drift-aware ensemble is validated against several state-of-the-art methods, including AdaBoost, Bagging [7, 8], ENDrift [34], KitNet [30], IForestASD [32], and HELAD [35], using benchmark intrusion detection datasets such as CIC-IDS2017, UNSW_NB15, and IoTID20. Experimental results demonstrate that the framework not only maintains high detection accuracy but also exhibits remarkable adaptability to the evolving nature of network threats. Its ability to process network traffic incrementally and detect emerging attack patterns in real-time makes it a promising solution for securing modern cyber-infrastructures against sophisticated cyber threats.

1.8 Study Area and Experimental Datasets

A diverse set of datasets has been utilized to analyze and assess the proposed research. The details of these datasets are provided below to offer a comprehensive understanding of the data sources and their relevance to the study.

1.8.1 Streaming Datasets for Classification

1. Electricity (Elec2) [87]: The Elec2 electricity dataset is taken from the Australian New South Wales Electricity Market, where prices vary. Electricity prices are affected by market supply and demand and are set every 5 minutes. This dataset contains 45,312 instances, where class labels indicate whether the price changed (two possible classes: up or down) relative to a moving average over the last 24 hours. An important feature of this dataset is its temporal dependencies.

2. Phishing [88]: This dataset includes features extracted from web pages, labeled as either phishing or legitimate. The dataset is binary, consisting of 1,250 samples and 9 features.

3. Bananas [89]: An artificial dataset where instances are related to clusters with a banana shape. There are two attributes, At1 and At2, corresponding to the x-axis and y-axis, respectively. The class label (-1 or 1) represents one of the two banana shapes in the dataset.

4. CreditCard [90]: This dataset contains transactions performed through credit cards in September 2013 by European cardholders. All transactions that occurred over two days are considered. Total samples in this dataset are 284807, and 492 fraud transactions are detected. This dataset is highly imbalanced and has 30 features. There are two classes. The class label 1 indicates fraud detection, and 0 represents a normal transaction.

1.8.2 Descriptions of real-world and synthetic drifting datasets

1. Weather dataset [84]: The National Oceanic and Atmospheric Administration (NOAA), USA, compiled a database of weather measurements from over 7,000 weather stations worldwide. The dataset contains 18159 samples with eight features and two classes (rain and no rain). Class labels are used as a binary indicator, indicating a 31% chance of being positive, while a 69% label indicates a negative case.

2. LUdata dataset [85]: The Luxembourg dataset (LUdata dataset) is based on the European Social Survey 2002-2007. This survey was designed to classify internet usage tasks through questioning, with a total of 31 features and two class levels (high/low). A total of 1,901 samples were selected for the general demographics survey questionnaires. This dataset includes timestamps, as the questionnaires were collected over 5 years.

3. sea_stream and sea_big datasets [36]: The SEA datasets have three numerical attributes, and the values of these attributes lie between 0 to 10. At one time, only two values may be considered for the classification task. There are four possible classification functions; only one function may be selected at a time. Some threshold values are taken and compared with the sum of any two relevant attributes. Based on comparisons of attributes, all four functions will be zero if two attributes are less than or equal to 8, 9, 9.5, and 7, respectively; else, they will be 1 for each function. In the sea_stream and sea_big synthetic datasets, there are 40,000 and 100,000 samples, respectively, with 3 features and a binary target class of 0 or 1.

1.8.3 Details of Anomaly Detection-Based Streaming Datasets

The theoretical explanations related to the dataset are provided below, and Table 1.1 presents comprehensive information on the nine streaming datasets, including their overall sizes, key characteristics, and anomalies.

1. HTTP: HTTP is a network protocol for streaming datasets that may contain unauthorized activities that behave as anomalies in traffic patterns. It continuously flows traffic data and detects anomalies to prevent security attacks.

2. CreditCard: The CreditCard dataset is a real-time, highly imbalanced stream of credit card transactions with a high probability of fraud. An ensemble framework is applied to consider all outlier transactions as fraudulent or anomalous, and efficiently detects the most reliable anomaly scores.

3. SMTP: SMTP datasets are highly sensitive streaming datasets that monitor emails during transmission and reception, where there is a high likelihood of anomalies in the form of spam or phishing emails. Evaluating the performance values of SMTP streaming datasets yields improved email security results.

4. MaliciousURL: The MaliciousURL streaming dataset is commonly used in cybersecurity to spread anomalies via malicious URLs. An effective model monitors and scores the real-time fake URLs for maximum efficiency.

5. Android: The Android anomaly detection dataset includes a vast number of applications for Android devices. However, it is uncertain how many of these apps are legitimate or anomalous. The dataset is primarily used to determine whether an application is malware (1) or not (0). It is equally distributed for apps that require permission (1) and those that do not (0).

6. covtype_1: The covtype_1 streaming dataset is a binary classification dataset that focuses on identifying anomalies present in forest regions, elevation, slope, etc. It contains key features extracted from satellite images for this purpose. The goal is to distinguish between normal and anomalous data.

7. cpu4: The cpu4 dataset is utilized for detecting anomalies and estimating performance based on CPU utilization. Performance scores that fall outside the range of CPU utilization values are considered anomalous and can lead to system failures.

8. Falling: The Falling dataset is designed to identify falls among elderly individuals in a smart home environment. The dataset includes data from four sensors that measure three-dimensional parameters X, Y, and Z, representing the person's chest, ankles, and belt from a fixed sensor position. Labels in the dataset indicate whether an event is a fall or anomalous (1) or a normal living event (0).

9. Phishing legitimate: The Phishing dataset includes labels for legitimate and phishing websites involved in phishing attacks. This dataset improves internet security and enables real-time detection of phishing attacks in URLs.

Table 1.1. Descriptions of anomaly detection based on streaming datasets.

S.No.	Datasets	Total Instances	Features	Anomaly Contributions	Reference
1	HTTP	567498	3	2211 (0.4%)	[92]
2	CreditCard	284807	30	492 (0.172%	[92]
3	SMTP	95156	3	30 (0.03%)	[92]
4	MaliciousURL	2396130	3231961	Benign (66%), Defacement (15%) + Others (19%)	[92]
5	Android	398	331	Benign Apps (50%), Malwares (50%)	[91]
6	covtype_1	286048	54	2747 (0.9%)	[93]
7	cpu4	17568	2	209 (1.18%)	[95]
8	Falling	23907	3	1153 (4.84%)	[94]
9	Phishing	1250	9	103	[92]

1.8.4 Details of Real-Time Streaming Datasets

1. IoTID20: The IoTID20 dataset [102] is widely recognized in IoT research for detecting anomalies. It mimics a real IoT setting, encompassing diverse network traffic data from IoT devices. The dataset encompasses 20 device categories, such as cameras, smart bulbs, and thermostats, generating both normal and anomalous network behavior. Notably, it contains crucial features such as packet size, inter-arrival time, protocol, and destination port. As a result, the IoTID20 dataset proves instrumental in assessing anomaly detection algorithms and crafting resilient solutions to safeguard IoT networks.

2. Delhi Air Pollution AQI Dataset: We collected real-time hourly data of eight gaseous emissions (PM2.5, PM10, NO, NO2, SO2, CO, NH3, and O3) from 20 base stations corresponding to the most polluted sites in Delhi, sourced from the authorized national portal of the Central Pollution Control Board[1] website, from 1st May 2022 to 30th April 2023. The dataset and code are made available online[2]. We computed the average AQI across all 20 sites for ground-truth annotation. AQI > 100 is considered an anomaly (annotated as 1), indicating higher pollution levels, while lower AQI values are considered normal (annotated as 0).

[1] https://airquality.cpcb.gov.in/ccr/#/caaqm-dashboard-all/caaqm-landing/data
[2] https://github.com/Santosh Kumar Ray/Real-time-Air-Pollution-Monitoring-in-Delhi-Using-Online-Anomaly-Detection-Algorithms.

1.8.5 Description of the Benchmarks Intrusion Detection Dataset

1. CIC-IDS2017 [113]: This intrusion detection dataset includes benign traffic and various modern types of attacks. It consists of multiple packet captures and utilizes a CICFlowMeter for network flow analysis. As a labeled dataset, it contains traffic flows defined by source IP, destination IP, timestamps, source and destination ports, network protocols, and a range of cyberattacks. The dataset encompasses a total of 28303 rows and 20 columns. Data was captured over five days, from Monday, July 3, 2017, at 9:00 AM to Friday, July 7, 2017, at 5:00 PM. Monday is characterized by benign network traffic, indicating a typical day without intrusions. However, from Tuesday to Friday, attacks such as brute-force FTP, brute-force SSH, DoS, Heartbleed, and web attacks were recorded as intrusions, labeled as 1, while normal instances were labeled as 0.

2. UNSW_NB15 [114]: This network security dataset encompasses nine types of security breaches: backdoors, fuzzing attacks, data analysis threats, DoS, worm infections, exploits, generalized attacks, reconnaissance, and shellcode. The dataset consists of 95,329 rows and 197 columns. All attacks are classified as anomalous threats and designated as 1, while normal instances are marked as 0.

3. IoTID20 [102]: The IoTID20 dataset focuses on network intrusion and is based on IoT botnet data, featuring network and flow-based characteristics. This dataset serves as a benchmark for identifying unusual activities in IoT networks. It contains 625,783 rows and 86 columns. The labels encompass various categories and subcategories of network attacks, considered intrusions, and labeled as 1. During the preprocessing of the dataset, irrelevant samples that introduced noise were removed. Additionally, optimal feature selection techniques were implemented to retain relevant features while discarding unusual ones that could lead to overfitting during model training.

The attack types within the CIC-IDS2017, UNSW-NB15, and IoTID20 datasets are described in detail, with packet-level information for each category summarized in Table 1.2.

Table 1.2. Description of various attack types present in the CIC-IDS2017, UNSW-NB15, and IoTID20 datasets.

	Traffic Type	Attack Category	Description (High-Level)	No. of Packets
CIC-IDS2017	Normal	Benign	Normal user activity (July 3, 2017)	22,662
	Malicious	Brute Force	FTP and SSH password attacks	—
		DoS / DDoS	Hulk, Slowloris, GoldenEye, and Slowhttptest	—

		Web Attacks	XSS, SQL Injection, Brute Force	—
		Infiltration	Malware/backdoor activity	—
		Botnet & Port Scan	ARES botnet and network scanning	—
	Total	—		28303
UNSW_NB15	Normal	Benign	Normal network traffic	49,997
	Malicious	Generic	General-purpose attacks	18,871
		Exploits	Vulnerability exploitation	11,132
		Fuzzers	Randomized input attacks	6,062
		DoS	Denial-of-service attacks	4,089
		Reconnaissance	Scanning and probing	3,496
		Analysis	Traffic analysis attacks	677
		Backdoor	Unauthorized remote access	583
		Shellcode	Malicious payload injection	378
		Worms	Self-propagating malware	44
	Total	—		95329
IoTID20	Normal	Benign	Normal IoT device traffic	40,073
	Malicious	DoS	Denial-of-service attacks	59,391
		Mirai Botnet	IoT botnet-based attacks	415,677
		MITM (ARP Spoofing)	Man-in-the-middle attacks	35,377
		Scan	Network scanning	75,265
	Total	—		625,783

1.9 Research Contributions

Our work introduces the design and analysis of a classification framework for streaming data, with the primary contributions summarized as follows:

i. We introduce the Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF), which integrates Adaptive Random Forest (ARF) with various drift detection methods. This framework dynamically adjusts the weights of base

learners according to their real-time prediction errors, effectively managing both abrupt and gradual concept drift while ensuring high classification accuracy.

ii. Additionally, we assess the performance of online classifiers, including Logistic Regression (LR), K-Nearest Neighbors (KNN), and Gaussian Naïve Bayes (GNB), utilizing the River and Creme frameworks on benchmark streaming datasets such as Elec2, Banana, Phishing, and CreditCard.

iii. We develop a novel anomaly detection method for evolving data streams by employing a hybrid stacked ensemble framework that combines ten base learners (from River and PySAD) with two meta-learners (Logistic Regression and AdaBoost). This approach effectively merges supervised and unsupervised learning techniques, improves anomaly detection accuracy, eliminates the requirement for predefined thresholds, and is validated across nine benchmark streaming anomaly detection datasets.

iv. Additionally, we provide a comprehensive survey of anomaly detection techniques for streaming data, analyzing their strengths, limitations, and challenges in real-time settings. The study reviews recent methodologies, compares their performance on imbalanced and high-dimensional data, and highlights key research gaps along with potential future directions.

v. We assess the performance of various online learning strategies on real-time streaming data by evaluating multiple online anomaly detection models using real-world datasets. The study demonstrates the effectiveness of these approaches and identifies the top-performing models, validated through real-time AQI data from Delhi.

vi. Additionally, we evaluate the performance of anomaly detection using the River and PySAD streaming frameworks with unsupervised online models on the IoTID20 dataset. The models are assessed using metrics such as Accuracy, Sensitivity, Specificity, and ROC-AUC, with a detailed analysis presented afterward.

vii. By integrating drift detection and anomaly detection within a dedicated chapter, we propose an IP2V-based online drift-adaptive ensemble framework for real-time network intrusion detection. The framework combines incremental learning with drift-adaptive ARF classifiers and employs Isolation Forest for anomaly detection. This unified approach effectively addresses both concept drift and anomalies, thereby improving accuracy and adaptability in streaming environments.

1.10 Reading Roadmap

The thesis is organized into seven chapters in alignment with the three stated objectives. Chapter 2 presents the literature review, while Chapters 3, 4, 5, and 6 provide a detailed discussion of the research objectives along with the experimental studies conducted. Finally, Chapter 7 outlines the conclusions, future scope of work, and the social implications of the research.

Chapter 1 introduces the research by providing a general overview and context of the study. It outlines the motivation, presents the problem statement, and highlights the key contributions and objectives. The chapter also discusses the scope of the work and identifies the relevant research gaps.

Chapter 2 presents a comprehensive review of streaming data classification techniques, with a particular emphasis on drift detection and adaptation in evolving environments. It also covers ensemble methods, anomaly detection approaches, and real-time evaluation strategies, highlighting key techniques, datasets, performance metrics, limitations, and research gaps that guide the proposed work.

Chapters 3, 4, 5, and 6 provide a thorough examination of each research objective, including the formulation of the problem statement, the literature review, contributions, the methodology, the workflow, and implementation details. Each chapter also includes experimental analysis, suitable datasets, performance metrics, results, discussion, and a summary.

Chapter 7 summarizes the key findings and contributions of the thesis, discusses potential directions for future research, and explores the social implications of the study.

CHAPTER 2

LITERATURE REVIEW

Recent growth in real-time applications such as network monitoring, financial transactions, sensor networks, and social media has generated vast amounts of streaming data. Unlike static datasets, data streams are continuous, high-speed, and evolving, making analysis and classification difficult. Challenges such as concept drift, changing data distributions, and anomalies demand adaptive, efficient learning. Researchers have proposed machine learning and online techniques to tackle these issues.

This section presents a comprehensive review of existing research on classification frameworks for streaming data, with particular emphasis on ensemble learning methods, anomaly detection for evolving data streams, and the performance evaluation of online learning strategies in real-time environments. The literature is organized in alignment with the objectives of this research. It highlights key methodologies, datasets used, evaluation metrics, and limitations of existing approaches. The discussion provides insights into how current techniques handle dynamic data characteristics and identifies the research gaps that motivate the development of the proposed classification framework for streaming data.

2.1 Review of Concept Drift Detection and Adaptive Classification in Data Streams

Concept drift poses a major challenge in streaming data, as the distribution of data changes over time, impacting model performance. Several adaptive and ensemble techniques have been proposed. One approach [36] uses chunk-wise sampling to train classifiers in a fixed ensemble for large data streams, but limits the consideration of the full dataset. Another method [37] detects drift with Bagging ADWIN and Adaptive-Size Hoeffding Tree Bagging, allowing scalable, parallel processing and model replacement to adapt to changes. Streaming Random Patches (SRP) [38] uses random subspaces with online bagging, training base learners on random samples and feature subsets, usually with Hoeffding Trees. Likewise, a Random Forest approach in [39] detects concept drift by monitoring feature importance in streaming data. As IoT systems grow, real-time data from sensors, devices, and embedded systems often changes distribution. Several drift detection methods [40, 41, 42, 43] aim to identify these changes and maintain performance, though sensor drift can still reduce sensing accuracy.

Du et al. proposed a selective detector ensemble in [44] based on an early-find-early-report rule to improve drift detection. Online ensemble classifiers combined with drift-detection mechanisms were also explored in [45] for both labeled and unlabeled data streams. However, block-based ensembles struggle with sudden changes, and incremental models may not handle gradual drift effectively. To address

this limitation, the framework in [46] combines block-based and incremental learning strategies through online component evaluation. Cano and Krawczyk [47] improved block-based ensembles by applying the Kappa statistic to base-learner weights and selection, with updates based on the Poisson distribution. Detecting drift in unlabeled streams remains challenging; Kim and Park proposed an approach in [48] that identifies drift using previous predicted class labels. These studies highlight the importance of adaptive, drift-aware learning for reliable classification in evolving data streams. Some major streaming-based drift detection techniques and their performance are presented in Table 2.1.

Table 2.1. Overview of Major Streaming-Based Drift Detection Techniques.

Authors	Methods	Datasets	Performances	Conclusions
Gama, Joao et al. 2004 [9]	DDM .	**Artificial:** SINE1, SINE2, SINIRREL1, SINNIREL2, CIRCLES, GAUSS, STAGGER, MIXED **Real-time:** Elec2	95% to 99% 96%	A new context is created when the warning level is crossed, and it starts learning afresh once the drift level is crossed.
Baena García et al. 2006 [11]	EDDM	**Artificial:** SINE1, SINE2, SINIRREL1, SINNIREL2, CIRCLES, GAUSS, STAGGER, MIXED **Real-time:** Elec2	90-95% 95%	Estimates the distribution of distances between consecutive errors and effectively detects gradual drifts.
Bifet, Albert et al. 2007 [12]	ADWIN	**Real-time:** Elec2 (44,848 instances)	97.4%	Defines a window of adjustable size that automatically changes when the

				distribution shifts.
Blanco et al. 2014 [13]	HDDM_A & HDDM_W	**Artificial:** STAGGER, Agarwal, LED **Real-time:** Elec2, SPAM, USENET1, USENET2, COVERTYPE, NURSERY	97% 99%	Two methods use Hoeffding's bounds to detect drift and trigger alarms for significant changes in the data.
Pesaranghader, Ali, et al. 2016 [14]	FHDDM	**Synthetic:** SINE1, MIXED, CIRCLES	87.7% 83.40% 86.66%	The Fast-Hoeffding Drift Detection Method (FHDDM) lowers false positive and false negative rates by using sliding window analysis.
Barros, Roberto SM et al. 2017 [10]	RDDM	**Artificial:** Agarwal generator Mixed generator Sine generator LED generator **Real-Time:** Airlines, Pokerhand Electricity	63.33% 100% 89.17% 25% 65.73%, 79.80% 84.47%	The Reactive Drift Detection Method (RDDM) addresses this limitation by discarding older instances to enable earlier drift detection.

Yang, Li et al. 2021 [16]	PWPAE	**IoT cyber-security:** IoTID20 and CICIDS2017	99.16% and 99.20%	A drift-adaptive ensemble framework that combines two drift detection methods, DDM and ADWIN, with the online ensemble classifiers ARF and Streaming Random Patches (SRP).
Yang, Abdalah Shami et al. 2021 [40]	OASW	**IoT cyber-security**: IoTID20 and NSL-KDD	99.92% and 98.31%	New adaptation methods effectively track pattern changes in online IoT data streams, showing high accuracy and efficiency.

2.2 Review of Online Machine Learning Approaches for Anomaly Detection in Data Streams

The availability of large-scale, continuously generated data increases the need for effective machine learning methods for anomaly detection in data streams. The high velocity, evolving patterns, and complexity of streaming data render traditional batch techniques ineffective. To address this, studies have examined ensemble and hybrid approaches that adapt to data changes while maintaining

accuracy. Among these, stacking-based ensemble methods [49] are popular because they combine multiple models to improve robustness. In stacking, base models trained on different data subsets or algorithms have their predictions combined via a meta-model to learn each model's contribution. This approach captures diverse patterns and improves anomaly detection in dynamic streams.

Several studies demonstrate the effectiveness of stacking-based approaches with various anomaly detection algorithms. For example, [50] explores stacking Isolation Forest models and uses logistic regression to efficiently identify anomalies. The iNNE framework [50] extends this by detecting outliers in real-time distributed systems via weighted voting among local detectors. Similarly, [51] proposes stacking with Local Outlier Factor (LOF) models, combining multiple LOF detectors with decision trees. This method also uses ensemble classifiers such as AdaBoost, J48, and Random Forests, along with feature selection, to improve prediction accuracy in high-dimensional datasets.

Deep learning-based stacking frameworks have shown promising results. In [35], autoencoder models are combined through a neural network stacking mechanism to detect global and local anomalies in high-dimensional streaming data. The framework uses incremental statistical feature extraction, autoencoder-based anomaly scoring, and LSTM models for better detection. Abro et al. [52] introduced a stacking ensemble that outperformed traditional methods on real datasets. Other studies have combined stacking with multiclass support vector machines for system health monitoring [53, 54], while hierarchical semi-supervised stacking models have been used for anomaly detection in smart home data [55]. These studies show the effectiveness of hybrid ensemble and machine learning techniques in improving anomaly detection in streaming environments. Some major streaming-based anomaly detection techniques and their performance are presented in Table 2.2.

Table 2.2. Overview of Major Streaming-Based Anomaly Detection Techniques.

Authors	Methods	Datasets	Performances	Conclusions
		Real-Time: HTTM, SMTP,		Streaming Half Space-Tree detects

Tan, Swee Chwan et al. 2011 [28]	HS-Tree	COVERTYPE, MULCROSS, SHUTTLE	98.2%, 74.0%, 85.4%, 99.8%, 99.9%	anomalies in continuous streaming data with high accuracy and efficiency, using standard data for single-class labels.
Goldstein et al. 2012 [29]	HBOS	Breast Cancer, Pen Global and Pen Local	99.10%, 92.14%, and 76.51%	An unsupervised outlier detection method that doesn't require prior training, modeling univariate feature densities with histograms with fixed or dynamic bin widths.
Ding, Zhiguo, et al. 2013 [32]	IForestASD	Http, Smtp, ForestCover, Shuttle	AUC: 0.94, 0.86, 0.83, 0.97	A streaming anomaly detection framework by adapting a static-data algorithm and validating its effectiveness.
Guha, Sudipto, et al. 2016 [31]	RRCF	NYC Taxicabs Window Size (256, 512, 1024)	96% 96% 96%	A resilient random-cut data structure produces a compact stream representation that approximately preserves distances for anomaly

				detection and related computations.
Burnaev, Evgeny et al. 2016 [33]	KNNCAD	Numenta Anomaly Benchmark	NAB Scores: 0.9999 0.8093	The study introduces a robust, nonparametric approach for identifying anomalies in one-dimensional time-series data using probabilistic anomaly scoring.
Faria, Eline et al. 2016 [56]	Novelty Detection-based	**Real-time:** KDD Cup-99, Forest cover, SHUTTLE Spambase. NLS KDD **Artificial:** SynC, SynCN, MOA	True, False	A stream-based novelty detection method adapts to recognize new abnormal data for future applications.
Mirsky et al. 2018 [30]	KitNet	Recon, Man in the Middle, Denial of Service, Botnet Malware	Execution Time (Min) 18.9-52.2, 19.2-74.8, 26.4-54.9, 66.9	A new method, "Kitsune's core algorithms," uses artificial neural networks (ANNs) to automatically detect network intrusions.
Cheng, Yongliang et al. 2019 [55]	HS-TCN	Data Collected from Smart Home Reading	100% (accuracy)	An anomaly is detected in smart home communication streams using stacking to remove outliers, demonstrating

				the effectiveness of hybrid ensemble frameworks for streaming data.
Q-T, Doan et al. [2019] [73]	Integration of IoT streaming data based on indexing and storage optimizations	**IoT Source 1:** 368,199 records and 94 MB in size **IoT Source 2:** 2592,200 records of 62.8 MB **IoT Source 3:** 1,472,800 records and 376 MB **IoT Source 4:** 6,480,000 records and 1.5 GB	256 days 3 days 1024 days 75 days	**Source 1, 2 & 3, 4 shows** Compression ratio 3.68% & 3.72% (sprintz), 27.24% & 27.5 % (real number bits blocks), 4.94% & 5.05% (real number bits blocks and byte transfer), 2.12% & 2.12% (real number bits blocks & huffman) **Storage of source 1, 2 & 3, 4:** 96.32% & 96.28% 2.76% & 72.5%, 95.06 %& 4.95% 97.88% & 97.85%

2.2.1 An Overview of Anomaly Detection Taxonomy

1. Unsupervised Anomaly Detection Techniques

a) Distribution-based Anomaly Detection Methods

This section covers distribution-based algorithms used for anomaly detection in data streams. This approach is used when the usual behavior of the sample can be modelled based on statistical properties [125]. An instance is considered an anomaly when it significantly deviates from the data's standard distribution. The distribution-based methods can be categorized into parametric and non-parametric methods, as shown in Fig. 2.1.

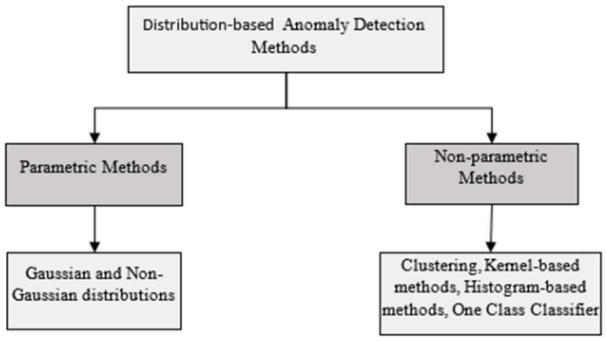


Fig. 2.1. Categories of distribution-based anomaly detection methods.

1. Parametric methods

Parametric distribution-based techniques assume a fixed probability distribution for the sample and then use statistical methods to identify anomalies that fall outside the expected range of values according to that distribution [126].

Gaussian Distribution (GD): One of the easiest and most universally used statistical techniques for discordant observation assumes that the input pattern follows a Gaussian or normal distribution [127]. Anomalies are identified as data points that fall outside of a fixed number of standard deviations above or below the mean value [128]. Shaikh et al. [127] adopted the concept of the Gaussian distribution to identify outliers in data streams, making it a highly sought-after statistical approach due to its ease of handling. Consider an uncertain object o in the storage database (DB) that qualifies as a distance-based outlier. This occurs when the anticipated count of objects $o_i \in DB$

within a certain distance D from object o is no greater than a limit value $\theta = N(1-p)$, where N represents the total number of instances in the database, and p denotes the fraction of objects in the database that are farther from o than distance D. The mathematical representation of the set of distance-based outliers using the Gaussian Distribution can be described as

$$DB_{Outliers} = \{o_i \in DB | \sum_{j=1}^{|DB|} Pr(dist(o_i, o_j) \leq D) \leq \theta\} \quad (2.1)$$

Ahmad et al. [129] successfully adapted the Gaussian distribution for detecting anomalies in data streams. They estimated the anomaly likelihood score from the complement of the Gaussian tail probability.

EXPected Similarity Estimation (EXPOSE): Schneider et al. [130] introduced a robust approach for anomaly detection from massive data streams. This methodology, called EXPOSE, competently calculates the homogeneity between naive data instances and the continuous data distribution. It achieves prediction in fixed time, using the kernel mean map $\mu[p]$ for the distribution p. Anomalies are detected by considering p as the distribution of normal data. Let $x_1, x_2, x_3, \ldots, x_n$ denote instances from P. Empirical distributions of P are then computed for the analysis as

$$P_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} \quad (2.2)$$

In this context, δ represents a probability measure. The empirical distribution P_n is estimated using the kernel mean $\mu[P_n]$ derived from the distribution $\mu[P]$ as

$$\mu[P] \approx \mu[P_n] = \frac{1}{n} \sum_{i=1}^{n} \varphi(x_i) \quad (2.3)$$

Here $\varphi(x_i)$ denotes the feature map. In situations where subtraction of the distributions is not possible, the EXPOSE method uses the dot product $\eta(z) = \langle \varphi(z), \mu[P_n] \rangle$ as an alternative. This substitution allows for the consideration that led to the derivation of EXPOSE as

$$\eta(z) = \langle \varphi(z), \frac{1}{n} \sum_{i=1}^{n} \varphi(x_i) \rangle \quad (2.4)$$

Poisson distribution (PD): This approach considers that the sample follows a Poisson distribution and identifies anomalies based on the rate at which events occur. The authors in [131] use a Poisson distribution with a thresholding mechanism to detect anomalies in high-dimensional data streams.

Linear regression: Another parametric approach for anomaly detection is linear regression [132], in which anomaly is detected when the real value significantly deviates from the expected value predicted by the linear regression model.

2. Non-parametric methods

Non-parametric distribution-based techniques avoid making any presumptions regarding the underlying probability distribution of the sample. Instead, they rely on statistical methods that utilize the data itself. A few non-parametric statistical techniques used for anomaly detection in data streams are discussed next.

Clustering: The process of clustering involves grouping data that shares similarities without using prior labels, by using a metric to measure the homogeneity or the gap separating two instances. Clustering approaches partition the instances into groups based on likeness. An anomaly, in this context, is recognized as a pattern or instance that is irrelevant to any of the clusters [127]. An updated clustering method in [133] identifies unusual patterns in the amplitude and shape of multiple time series. This method first divides the data into smaller groups using a sliding window technique, then uses an advanced form of clustering called fuzzy clustering to identify patterns within these groups. The authors in [134] use a clustering-based approach called AnyOut for detecting anomalies in data streams. This algorithm used cluster features as nodes in a tree-like structure, generating a cluster hierarchy.

Kernel Density Estimation (KDE): This approach estimates the probability density function of the given patterns or instances using the kernel function. Contaminants are identified as data points with a small probability density [135]. Current methods for detecting anomalies in data through local density estimation have limitations, as they are not precise enough for handling large and complex databases, and their execution is influenced by the size of the neighborhood considered. In view of the above, the authors in [136] have adopted a new approach to determining the density of samples in a specific area, and have developed a technique to adjust the density calculation to account for variations in the size of the surrounding area using a new kernel function. A kernel-based method [137] learns a decision function that separates target data from others, with regularization to control model complexity, and optimizes the solution using quadratic programming.

One-Class Classification (OCC): This technique involves training a classifier on a dataset containing only unusual instances, rather than on one containing both typical and atypical instances. Anomalies are then identified as patterns that deviate significantly from normal patterns by the classification methods [21]. Miao et al. [23] propose a distributed online OCSVM for streaming network data. A decentralized cost function is proposed to implement the distributed algorithm. Kurt et al. [138] defined a nonparametric algorithm for both the usual and extraordinary multivariate data distributions, which are completely anonymous. Furthermore, they extracted valuable unidimensional summary metrics and conducted an abnormality investigation in one-dimensional space. One-class SVMs learn a model of normal instances and define anomalies as significant deviations from this model. The study examines the application of a one-class support vector machine technique [139] for identifying the inception of system anomalies and classifying trend outcome probabilities. The output from the one-class support vector model is adjusted to succeeding probabilities by

installing a logistic distribution to the support vector prediction method to reduce inappropriate alarm rates.

The basic concept of OCSVM is presented next in the form of a mathematical formulation. Consider the summary statistics of the data corresponding to x_t as represented by d_t. Consider that the statistical characteristics of x_t are varying over time t. Consequently, the statistical properties of d_t will also change at the time τ. We analyse the probability distribution functions (PDFs) of d_t under both nominal and anomalous conditions, denoted by ${f_0}^d$ and ${f_1}^d \neq {f_0}^d$ respectively. Combining these findings, the conclusion is as follows:

$$d_t \sim \begin{cases} {f_0}^d, & if\ t < \tau \\ {f_1}^d, & if\ t \geq \tau \end{cases} \tag{2.5}$$

where the value of ${f_0}^d$ and ${f_1}^d$ are considered unknown.

Overall, distribution-based anomaly detection techniques are effective in identifying anomalies across a wide range of applications [140]. However, they do have some obstacles. For example, these techniques may not be effective if the usual behavior of the instance is unknown or cannot be accurately modeled [141]. Additionally, these concepts may not be effective at detecting subtle or sparse anomalies. Some distinctive distribution-based anomaly detection approaches are illustrated in Table 2.3.

Table 2.3. A summary of popular distribution-based approaches for anomaly detection in data streams.

Ref.	Method	Highlight	Distinctiveness
Laxhammar et al. 2010 [142]	Streaming Data-based Distribution-Independent Method	This research examines statistical techniques in sea surveillance using recorded data, assumed to be normally distributed. Prediction sets are supported by an empirical error rate slightly below the significance level, validating their reliability.	Superior anomaly detection sensitivity with a simulated instance.
Catania et al. 2012 [143]	Autonomous labelling approach using SVM	This paper proposes an autonomous labeling approach to support vector machines (SVMs) for detecting network traffic anomalies. The aim is to enhance the accuracy of	This approach makes the process more efficient and reliable while maintaining the efficacy of

		abnormality recognition by using an unsupervised learning approach to automatically label network traffic data.	exploring abnormalities using SVM.
Wu et al. 2014 [144]	Single-class semi-supervised RS-Forest model for data streams	This method integrates two key concepts: accurate statistical attribute range estimation with a high probability guarantee and dual node profiles for efficient model updates. This integration effectively handles the dynamic nature of data streams.	The study establishes a theoretical upper bound for the approach and explores its asymptotic characteristics using bias-variance decomposition.
Paulauskas et al. 2019 [145], Goldstein et al. 2012 [29]	Histogram-Based Outlier Score (HBOS)	These studies examine the effectiveness of the HBOS in identifying abnormalities within computer networks. The researchers explore various histogram construction techniques and how the number of bins impacts anomaly detection performance.	This method utilizes both static and dynamic bins, achieving high detection scores and fast execution times.
Gu et al. 2019 [146]	Statistical Nearest Neighbour-based anomaly detection method	At the first level of this approach, the authors employed Nearest Neighbor-based methods for anomaly detection. Through simulations, they found that these methods significantly outperform other state-of-the-art techniques. In the second phase, we examined the theoretical properties of these methods by analyzing a quantity called distance-to-measure.	Computes and analyzes the empirical performances with the sample point score and the degree of anomalous data points.
Talagala et al. 2020 [147]	Streaming nonstationary temporal data framework.	The framework is initiated by establishing the limit for the system's normal behavior using extreme value theory. It subsequently uses a sliding window concept to identify	This method performs effectively even when dealing with noisy nonstationary data across various

		abnormal series among a recently received set of series.	classes of time series.
Li et al. 2021 [133]	Clustering-based anomaly detection from time series data	Tests conducted on multiple artificial and real datasets have demonstrated that this strategy 117 effectively identify deviants in multivariate time series data. The framework is useful for identifying abnormal patterns across domains such as healthcare, weather analysis, finance, and disease tracking by leveraging clusters revealed by extended fuzzy clustering.	A multivariate time series-based concept that deals with the shape and amplitude collectively.
Wang et al. 2021 [148]	k-NN Centroid-based Outlier Detection Method	An optimal method for identifying outliers using k-nearest neighbor centroids is presented. It is simple to implement and yields results comparable to those of existing solutions. Tests on real datasets have shown that this method is effective.	Unique approach that automatically decides the counting of "k" efficiently.
Ryu et al. 2021 [149]	Dynamic Sequential Extreme Studentized Deviate Tests	The study introduces a contextual abnormality detection methodology designed for streaming time-series data. This approach utilizes an online version of generalized extreme studentized deviates tests, enabling efficient outlier detection in the presence of various patterns within the data stream.	A statistical hypothesis testing method tailored for analyzing streaming data exhibiting diverse patterns.
Wu et al. 2021 [150]	Anomaly detection in real applications of data streams	The study presents a real-time abnormality detection model for gas sensor streaming data, accomplished by integrating the support vector regression (SVR) algorithm with the conventional statistical distribution method.	Real-time system updates for precise abnormal situations in streaming sensor data.

Glyn-Davies et al. 2022 [151]	Gaussian Process-based Stochastic Differential Equations in data streams	This research focuses on merging non-parametric estimation techniques for stochastic differential equations with statistical testing principles, creating a versatile algorithm for real-time anomaly detection. It utilizes SDE estimation to capture both the deterministic and stochastic components of the normal data generation process.	Statistical testing is employed to compare new incoming instances with the learned model, identifying anomalies in data streams.

b) Proximity-based Anomaly Detection Methods

Proximity-based anomaly detection techniques identify abnormalities based on their distance from other instances in the dataset. These techniques are often employed when the ordinary behavior of the instance is unknown or cannot be modeled [120, 140]. A popular approach is to categorize an instance as unusual when it is in a region with few other samples nearby [120]. An enhanced version of the distance and density-based proximity algorithm was proposed in [152, 153], which refines DBSCAN to convert it into a hierarchical clustering algorithm, requiring only the minPts parameter inherited from DBSCAN for density-based space transformation. The concept of proximity can be explained in various ways, but they are similar enough to be grouped in one section. The prominent proximity-based techniques used for anomaly detection are categorized in Fig. 2.2.

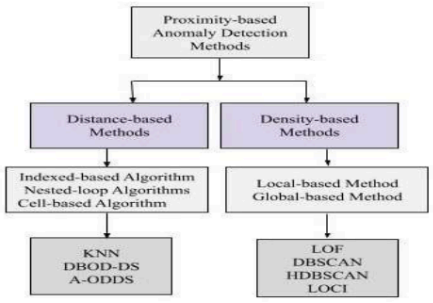


Fig. 2.2. Categories of proximity-based anomaly detection methods.

1. Distance-based anomaly detection methods

K-Nearest Neighbor (KNN): KNN-based techniques identify anomalies as data points that deviate noticeably from the values of their nearest neighbors. A new KNN-based approach is introduced in [154] that utilizes a time series approach called k-Nearest Neighbor (kNN) investigation. This approach has the benefit of handling electrical measurements with fluctuating patterns and can be used in real-time. The method is broken down into two segments: offline modeling and online detection. During the offline stage, a series of anomaly index values is calculated using KNN on past samples, and a threshold is established to detect anomalies. The online stage then applies this threshold to freshly measured data. This technique identifies anomalies as input data patterns that have a low figure of nearest neighbors within a certain range [155]. Finding unusual data points in datasets with multiple dimensions is strenuous. K-Nearest Neighbor-based methods distinguish between truly exceptional discordant versus normal instances by assessing how far they deviate [148].

Yao et al. [156] designed a KNN-based technique for investigating local deviants in streaming data. The scheme involved dynamically evaluating the local outlier status of each data instance. This approach utilizes concepts such as k-nearest neighbors, reverse nearest neighbors, and avails nearest neighbors, which are computed for every instance. To estimate the local density of an object, a distance-based technique considers its composite nearest neighbors (CNN) zone.

Consider a set of D-dimensional samples, denoted as $X = \{x_1, x_2, x_3, \dots, x_n\}$, where $|X| = n$ and each instance $\in R^D$, for $i = 1, 2, 3, \dots, n$. The local density of x_i is then computed as the mean distance from x_i to its k-nearest neighbors $kNN(x_i)$ as

$$p(x_i) = \frac{1}{k} \sum_{x_j \in kNN(x_i),\ j \neq i} \|x_i - x_j\| \tag{2.6}$$

The expression $\|x_i - x_j\|$ represents the Euclidean distance between two instances x_i, and x_j.

Additionally, the Local Outlier Factor (LOF) density is computed based on the composite nearest neighbors (CNN). The density deviation of x_i from the $CNN(x_i)$ is computed as

$$CLOF(x_i) = \frac{p(x_i)}{\frac{1}{|CNN(x_i)|} \sum_{x_k \in CNN(x_i), i \neq k} p(x_k)} \tag{2.7}$$

In this context, $|CNN(x_i)|$ signifies the count of instances in the $CNN(x_i)$, the estimated value of the CLOF is 1, and the distributions for both x_i and CNN neighbors are identical.

K-means-based anomaly detection techniques: K-means is a type of unsupervised machine learning approach used to categorize a dataset into distinct groups or clusters. It is frequently applied in data science and machine learning for solving clustering problems. This method [157] identifies unusual network traffic using flow data and the k-means clustering algorithm. It involves grouping unlabeled flow records from training data into separate clusters for normal and abnormal traffic, an efficient technique for identifying unusual network behavior by combining K-means clustering and the C4.5 decision tree methods. This method, known as "k-Means + C4.5," utilizes the clustering results from k-Means to classify usual and abnormal activities in a computer network [158]. This approach [159] introduces cluster-based local outliers and the CBLOF measure, achieving improved outlier detection compared to existing methods.

Distance-Based Outlier Detection for Data Stream (DBOD-DS): A method for identifying unusual data points, known as DBOD-DS [160], utilizes a unique, adaptable probability density function to address challenges in handling data streams. Results from testing on a meteorology dataset demonstrate the superiority of DBOD-DS in terms of precision when compared to other methodologies.

In this approach, a probability density function $f(x)$ is used, representing the distribution when x is a random attribute. Data samples d with a value v is processed and evaluated for the occurrence probability with the term $p(v,\ r)$, where r is a user-defined radius away from the term $v(p(v,r))$. The integration of the $f(x)$ function is then performed within the limits of $v-r$ to $v+r$. The mathematical description is given below

$$p(v,r) = \int_{v-r}^{v+r} f(x)dx \qquad (2.8)$$

According to this idea, when the occurrence probability $p(v,r)$ is much lower than the assumed occurrence probability q, then the data sample d is confirmed as an anomaly.

2. Density-based outlier detection methods

Local Outlier Factor (LOF): Outlier detection is commonly viewed as a binary concept based on properties. However, LOF methods [161, 162] challenge this notion by considering outliers as dependent on the extent to which objects deviate from their local surroundings, identifying local outliers based on neighboring objects within clusters. Let the distance of an object p from its k nearest neighbours be denoted by $d(p,o)$, where o is an object that is represented as $o \in D$. In this scenario, two conditions will emerge for detecting outlier objects.

(a) For a minimum of k objects $o' \in D|\{p\}$, condition is satisfied when $d(p,o') \leq d(p,o)$, additionally

(b) for a maximum of $k-1$ objects $o' \in D|\{p\}$, condition is met when $d(p,o') < d(p,o)$.

Local Correlation Integral (LOCI): LOCI is a density-based method that computes the local density of a point and compares it to the density of its neighbors to identify outliers. Variation in local densities may cast an ambiguity on the actual definition of an anomalous instance. A method called Local Correlation Integral (LOCI) in [163] utilizes a metric known as Multi Granularity Deviation Factor (MDEF) to assess the similarity between the neighborhood density of a specific data instance and the average neighborhood density. MDEF is robust to local density variations in the feature space.

Density-based spatial clustering of applications with noise (DBSCAN): This technique clusters data points based on their density, and identifies anomalies as data points that are not part of any cluster or are in low-density clusters [164, 153]. Finding outliers in constantly updated data is difficult because it cannot be analyzed repeatedly, and new concepts may appear in future data. Attributes that are not useful for the analysis can be considered noise, making it harder to work with the data streams. A dynamic concept is introduced in [165] that combines density-based and partitioning-based clustering methods, employing both density and distance approaches to detect outliers.

Proximity-based techniques have several advantages. For example, they do not assume that the data adheres to a distinct distribution, and they can handle high-dimensional data effectively [120]. In addition, these techniques can handle noisy and incomplete data and are resistant to the presence of abnormality in the training data [164]. However, proximity-based techniques also have some limitations. For example, these techniques may be subtle in the selection of the distance evaluation and the count of nearest neighbors. Additionally, these methods may not be robust in identifying subtle or sparse anomalies [155].

Overall, proximity-based anomaly detection techniques are effective in identifying abnormalities across a wide range of applications. These techniques are especially helpful when the ordinary behavior of the sample is unknown or cannot be accurately modeled. Proximity-based anomaly detection methods are illustrated in Table 2.4 below.

Table 2.4. A summary of popular proximity-based approaches for anomaly detection in data streams.

Ref.	Method	Highlight	Distinctiveness
Angiulli et al. 2007 [166]	Distance-based method for identifying	This approach comprises two phases: a stream outlier query approach, which demonstrates efficiency in processing time,	Innovative method for identifying deviants in data streams using

	outliers in data streams	and an approximate method that shows effectiveness in precision and recall of the solution.	query-based approaches.
Sadik et al. 2010 [160]	Distance-Based Outlier Detection for Data Stream (DBOD-DS)	The outlier detection technique DBOD-DS employs a versatile adaptive probability density function to address the limitations of data streams. Thorough testing conducted on a real-world meteorology dataset reveals that DBOD-DS outperforms prevalent tactics in connection to fidelity.	Computes the probability density function (PDF) of an instance within a radius of the neighborhood with lower time and space computation.
Niennattrakul et al. 2010 [167]	Data editing outliers' detection approach for streaming data	This study presents data-editing techniques that employ two complementary methods for efficiently detecting distance-based outliers in high-speed streams with resource-constrained sensors. The original algorithm ensures that no false positives occur, while the alternative algorithm ensures that no false negatives occur.	The framework enables efficient distance-based anomaly detection in data streams, intending to reduce the database size.
Tran et al. 2016 [168]	Distance-based Outlier Detection Data Streaming (DODDS)	This study is driven by the need for a comparative assessment of state-of-the-art models using identical datasets and platforms. It aims to systematically assess the newest distance-based outlier detection approach in data streams, taking into account several outlier rates.	Effectively manages continuous outlier detection in a streaming environment.
Toliopoulos et al. 2022 [169]	Distance-based outlier detection approach.	An unsupervised distance-based outlier detection approach that explores subspaces to intuitively label outliers without runtime training, enabling efficient online detection and explanation in data streams using massive parallelism.	Enhanced and rigorously evaluated with respect to both label quality and performance.

Sinha et al. 2018 [170]	Local density anomaly detection technique	The authors conducted numerous experiments with MRLOF and SLOF on various real-life and synthetic datasets, and the algorithms were found to outperform the simplified LOF.	Parallel computation of two anomaly detection algorithms with lower time complexity.
Huo et al. 2019 [171]	Online anomaly detection approach	The new online algorithm for distance-based anomaly detection is designed to tackle abnormalities and their challenges in intelligent operations. It specifically addresses the demand for high accuracy in time series databases.	The idea of online fast anomaly detection is centered around using a sliding window to reduce time complexity.
Chenaghlou et al. 2018 [153]	Clustering-based online anomaly detection techniques for streaming data	This concept for identifying unusual occurrences in real-time uses both the proximity in time and space of observations. It can identify the formation of clusters in noisy data streams, continually update the model, and determine the optimal window size for analyzing the constantly changing data without sacrificing accuracy.	This method identifies anomalies as outlier instances during the temporal development of clusters in real-world situations.

Tran et al. 2020 [152]	Core Point Outlier Detection (CPOD)	An improved algorithm called CPOD utilizes a specialized data organization, known as "core point," with multi-distance indexing to promptly identify normal instances and minimize the search space for potential outliers. The approach helps to increase the cost-effectiveness of outlier detection while minimizing memory usage. Results from testing on numerous datasets demonstrate that CPOD is, on average, 10, 19, and 73 times quicker than M_MCOD, NETS, and MCOD, respectively, while requiring minimal storage.	Memory-saving approach, reduced neighboring search time, and multi-distance pruning techniques.
Isha et al. 2021 [172]	Outlier detection technique for high-dimensional data	This integrated methodology can detect outliers without requiring training datasets or previous knowledge about the fundamental procedure that produced the dataset, making it efficient for computing local density-based outliers.	Detects outliers without requiring training or prior knowledge, thereby boosting accuracy for high-dimensional data streams.

c) Reconstruction-based Anomaly Detection Methods

Reconstruction-based techniques identify anomalies by reconstructing the data and assessing the reconstruction mistakes to a threshold [121]. These techniques are often used in the domain of signal processing and are particularly useful for detecting abnormalities in time series data [21]. Some common reconstruction-based techniques are shown in Fig. 2.3.

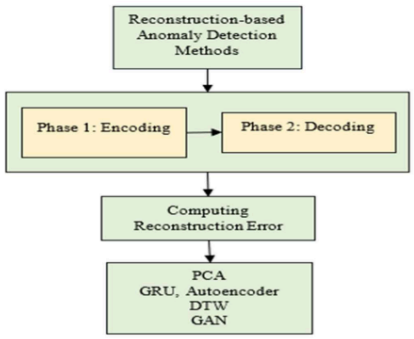


Fig. 2.3. Some examples of reconstruction-based anomaly detection methods.

Principal Component Analysis (PCA): This method forecasts the pattern onto a lower-dimensional space and identifies anomalies as data points with a large reconstruction error [173]. Anomaly detection in networks has been done using Spatial Principal Component Analysis (SPCA) in [174], but it has been found that PCA is sensitive to calibration settings. The authors in [174] aim to address this issue and explain the discrepancies found. PCA is commonly employed to decrease the data's dimensionality, making it easier to explore and analyse. This concept is commonly utilized to identify multivariate outliers by calculating the distance from each data point in the principal component space. In some cases, PCA loses its relevance when dealing with large datasets. In such scenarios, concepts based on the chi-square distribution with q degrees of freedom are used. It is assumed that the eigenvalues for all components are positive and distinct, i.e., $\gamma_1 > \gamma_2 > \gamma_3 > \cdots > \gamma_n > 0$ for a given significance level α. The final criterion for detecting outliers is as follows: an observation x will be considered an outlier if the sum of the squared values of $\sum_{i=1}^{q} \frac{y_i^2}{\gamma_i} > \mathcal{X}_q^2(\alpha)$, where the right-hand term is the upper significance level of the chi-square distribution with respect to q. The worth of α signifies the error or false alarm probability when classifying a usual observation as an outlier, where y_i is a simple principal component for observation x.

Autoencoders: A neural network model that is trained to reconstruct the input instance by learning a compressed depiction of the data; thus, anomalies are identified as data points with a large reconstruction error [177]. A novel anomaly detection method in [175] utilizes the reconstruction probability from a variational autoencoder. This considers the volatility of the distribution of variables to identify anomalies. An autoencoder is incorporated in [176] with the objective of identifying anomalies in event streams in an unsupervised manner. This is expected to boost the results of online

process mining. Some autoencoder-based anomaly detection method [178] learns the normal behavior of high-performance computing systems and identify anomalies as deviations from this pattern. This approach avoids reliance on limited abnormal-condition data used in earlier methods.

Dynamic Time Warping (DTW): The Dynamic Time Warping approach calculates the homogeneity between two sequences of events that may differ in speed. It measures the similarity between two temporal sequences. This technique aligns time series data by warping the time axis and identifies anomalies as data points with a large reconstruction error [179]. It was introduced in the Data Mining field as a useful tool for various tasks related to time series hurdles, covering classification, clustering, and anomaly detection [180]. DTW can be applied to any instance that can be modified into a linear sequence, including audio, video, and graphics data. DTW has been widely used in automatic speech recognition to handle variations in speaking speed.

Reconstruction-based techniques have several advantages. For example, these techniques can handle high-dimensional data effectively [121]. In addition, these techniques can handle noisy and incomplete data and are resistant to the existence of abnormalities in the training data [174]. However, reconstruction-based techniques also have some limitations. For example, these techniques may be susceptible to the choice of the reconstruction model and the threshold for the reconstruction error [21]. Additionally, this method may not be proficient in detecting subtle or sparse anomalies [176]. Reconstruction-based anomaly detection methods are illustrated in Table 2.5 below. These include mention of recent models used for reconstruction, such as Generative Adversarial Network (GAN), along with long short-term memory (LSTM) neural network models.

Table 2.5. A summary of popular reconstruction-based approaches for anomaly detection in data streams.

Ref.	Method	Highlight	Distinctiveness
Jeong et al. 2015 [181], Jeong et al. 2011 [182]	Weighted DTW-based time series classification for anomaly detection	It employs a supervised learning model, specifically the SVM algorithm, along with a similarity measure known as the weighted dynamic time warping kernel function, which considers the temporal alignment of the time series.	The incorporation of DTW can enhance the performance of time series classification in various applications, including speech recognition, healthcare, and finance.

Li et al. 2019 [183]	MAD-GAN: Multivariate anomaly detection using GAN	The method utilizes the generator and discriminator of the GAN to detect anomalies through reconstruction and discrimination.	The combination of GAN and LSTM can effectively capture temporal correlations in time series distributions, thereby enabling more effective anomaly detection.
Geiger et al. 2020 [184]	Generative Adversarial Network (GAN) based anomaly detection method	The method is robust, computing reconstruction errors and employing diverse techniques to combine them with critic outputs. This enables the computation of anomaly scores, facilitating effective analysis of time-series data.	The method proves the effectiveness of GANs in reconstructing time series and detecting anomalies.
Qiao et al. 2020 [185]	Reconstruction-error-based one-class classification method	A new method for reconstructing data that addresses issues of uneven class distribution by combining elements from both positive and negative classes, using the F1 score and cross-entropy loss. Experiments demonstrate that this approach improves the accuracy of abnormality detection models, such as PCA and autoencoders.	This method can be applied when the data is sparse and binary, and traditional reconstruction-based methods are not applicable.
Zhu et al. 2020 [186]	KNN-Based outlier detection over IoT streaming data	The study presents a novel framework with two phases. Initially, a grid-based index is established to manage streaming data within the window. The grid's resolution is dynamically adjusted according to object distances, facilitating efficient approximate neighbor-based outlier detection. Subsequently, an outlier candidate algorithm identifies objects with a higher probability of becoming outliers.	A novel algorithm was developed for approximate K-nearest neighbor (KNN) search, offering both high efficiency and accuracy.

Kwak et al. 2021 [187]	Unsupervised anomaly detection using Convolutional Autoencoder (CAE)	A convolutional autoencoder is used to generate explicit channel-wise reconstruction errors, which are then aggregated as an anomaly score to detect abnormal signals.	Significantly improves detectability when compared to merely averaging the reconstruction errors.
Liu et al. 2022 [188]	Time series-based adversarial reconstruction network model	BeatGAN is an unsupervised reconstruction model designed to detect anomalies primarily from normal data, where the majority of samples are normal. It achieves this by using only a small fraction of anomalous time series during training.	Adversarial learning framework for enhanced accuracy and efficient inference in reconstruction.
Deka et al. 2022 [189]	Semi-supervised dynamic density approach	The approach involves utilizing reconstruction error vectors obtained from long short-term memory networks with an encoder-decoder (LSTM-ED) architecture to identify anomalies in time series data. The effectiveness of this method is demonstrated through experiments conducted on both a testbed and various real-world datasets.	A highly efficient dynamic density approach calculates reconstruction error vectors, which are remarkably effective in detecting performance and security anomalies.

d) Unsupervised Ensemble-based Anomaly Detection Frameworks

Approaches based on ensemble learning utilize multiple models or algorithms to identify abnormalities in data streams. Ensemble learning can enhance the robustness and accuracy of irregularity finding by merging the strengths of different models or algorithms. Le and Zincir-Heywood propose an unsupervised learning-based ensemble method [190] for detecting insider threats by utilizing four different unsupervised anomaly detection methods and exploring various data representation approaches that incorporate temporal information. Furthermore, the authors investigate various methods for combining these models to create ensembles for anomaly detection, aiming to enhance overall performance. A new unsupervised ensemble framework was proposed in [191] for identifying abnormal behavior of nodes in real-time by modeling their temporal behavior. The framework uses a latent space model to assign each node a probability distribution across a minimized set of

roles based on its characteristics. This method helps identify anomalies in node behavior based on their deviation from expected roles.

Zhao et al. [192] focus on enhancing the performance of ensembles for outlier detection by utilizing score and rank aggregation. The authors further examine sequential approaches in which one outlier detection model is followed by another. The research in [124] evaluates the effectiveness of three unsupervised methods for identifying deviants: One-Class SVM, elliptic envelope, and Local Outlier Factor, using real-world datasets. In a similar context, the unsupervised online abnormality detection concept presented in [193] identifies unusual consumption patterns in real-time smart devices by detecting deviations from normal usage behavior. To improve performance, these algorithms are combined using a voting ensemble method. This may be categorized as in Fig. 2.4. The supervised ensemble-based methods will be illustrated in Section 2.

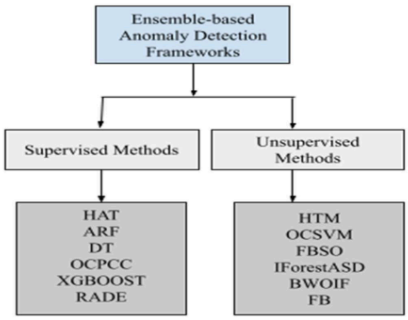


Fig. 2.4. Categories of supervised and unsupervised ensemble-based anomaly detection frameworks.

Several unsupervised ensemble techniques are popularly used as base learners for ensemble learning in outlier detection. Some of them are:

Isolation Forest Algorithms for Streaming Data: Isolation Forests employ a collection of decision trees to identify anomalies that significantly differ from the majority of the data. The unique properties of streaming data, including its rapid generation, high volume, and tendency to change over time, make detecting anomalies

in such data a challenging task. These characteristics, like infinite data and the concept drift phenomenon, highlight the challenges of anomaly detection in streaming data. In this method, a new approach is used to identify anomalies in streaming instances by utilizing the concept of sliding windows and considering the concept of concept drift. This novel anomaly detection methodology modified the iForest algorithm called iForestASD [32] for use with streaming data. A new unsupervised machine learning technique that combines the K-means algorithm and the Isolation Forest algorithm to identify deviations in large datasets, a scheme tailored to industrial settings, is proposed in this approach [194]. LSHiForest is a method specifically developed for identifying abnormalities in both batched and multi-dimensional data. The approach involves constantly updating the data structure in a streaming setting rather than rebuilding it. It can tackle high-dimensional instances and identify anomalies that may be surrounded by other anomalies or local outliers [195].

Feature Bagging (FB): Feature Bagging is an ensemble method that randomly selects a subset of features from the data and trains multiple models on these subsets to identify anomalies.

Some prominent unsupervised anomaly detection techniques, along with their distinctiveness factors, are described in Table 2.6.

Table 2.6. A summary of popular unsupervised ensemble-based approaches for anomaly detection in data streams.

Ref.	Method	Highlight	Distinctiveness
Naguyen et al. 2015 [196]	Unsupervised anomaly detection ensemble for online games	An evaluation method was employed to assess the efficacy of unsupervised anomaly detection techniques, with kernel density and ensemble techniques yielding the best results on game data. The results of this analysis helped detect and remove players who were cheating.	A unique approach for detecting anomalies in online game activity that does not rely on labeled examples and can be adjusted to different categories of game data.

Ghao et al. 2015 [197]	Density and Rank Ensemble Framework for sequential data	The concept of an ensemble framework aims to enhance the performance of individual algorithms by aggregating scores and ranks. This sequential method involves one detection method following another, resulting in improved overall performance.	The sequential methods demonstrate a substantial enhancement in detecting anomalous data points across all datasets.
Tuor et al. 2017 [198]	Unsupervised Deep learning-based anomaly detection employing multiple RNNs	This study employs an online unsupervised deep learning method for real-time detection of anomalous network activity from system logs. The approach utilizes multiple RNNs to model the normal behavior of multiple users.	This study suggests that recurrent neural networks can be used to model "normal" user behavior. Anomaly is proportional to the prediction error of the network.
Bibinbe et al. 2022 [199]	Unsupervised streaming anomaly detection ensemble	In this survey, the author evaluated various methods for identifying abnormal data points in unsupervised data streams, focusing on their performance, runtime, and ability to handle characteristics such as data drifting, seasonality, trends, and cycles in the datasets.	This survey provides an esteemed resource for researchers interested in grasping and applying unsupervised techniques for detecting abnormality in streaming data.
Chaurasia et al. 2020 [200]	Unsupervised autoencoder ensembles for Anomaly Detection	The outlier detection employs a strong ensemble of autoencoders, which are effective in reducing dimensionality. Autoencoders excel at mapping higher-order features to lower-order representations, facilitating the segregation of data points.	The experiments yield exceptional outcomes, establishing the efficacy of autoencoder ensembles as compared to other neural network-based techniques.

Heigl et al. 2021 [80]	Unsupervised Anomaly Detection Using a Performance Counter–Based iForest	Efficient detection on networked embedded devices requires lightweight, memory-efficient, and anomaly-adaptive methods, such as PCB-iForest, a generalized ensemble-based online outlier detection framework for streaming data.	It integrates two variants: an Extended Isolation Forest and a classic Isolation Forest enhanced with feature contribution–based anomaly scoring.
Hannak et al. 2023 [201]	Bilateral weighted online adaptive isolation forest (BWOIF) for streaming data	A new algorithm called BWOIF, which utilizes a bilateral weighting scheme to adapt the Isolation Forest algorithm for streaming data. The algorithm employs two distinct weighting factors, one for the recent data sample and one for the older data points, to evaluate the anomaly scores of data instances.	BWOIF efficiently detects anomalies in streaming data with exceptional accuracy while maintaining low computational demands. The bilateral weighting scheme distinguishes between old and new samples for identifying dynamically changing anomalous behavior.

e) Statistical methods for real-time unsupervised anomaly detection in streaming data

This section introduces some simple statistical indicators or metrics popularly used for real-time anomaly detection from data streams. This approach is valuable because it relies on statistical methods to calculate moving statistics [118, 202, 203], which are essential parameters for streaming data and its models. It does not need previous knowledge of the instance, making it a useful tool for identifying anomalies in real-time data. Unsupervised statistical anomaly detection is a concept applied to identify abnormal or malicious patterns in instances. It builds a model of the normal behavior of the data and then uses this approach to identify instances of data that do not fit this usual pattern. We considered three popular unsupervised statistical techniques (Z-Score Method, IQR Method, and Welford Method). When tackling streaming data, it is preferable to compute the performance of the abnormality detection algorithm in real-time. Below are some popular evaluation metrics used for this purpose.

Welford Method: Welford's method is a technique that operates in a one pass for estimating the flowing variance or standard deviation. It involves calculating the variations among the sums of squared dissimilarity for N and $N-1$ instances. No additional data storage is required for a second pass. For detecting anomalies, a modified method is used. Upper and lower limits are defined, and any instance that exceeds these limits or a specified multiple (X) of the online standard deviation is deemed an anomaly. The limits are calculated as

$$Upper\ limit = Online\ mean\ consumption + X \times Online\ standard\ deviation \tag{2.9}$$

$$Lower\ limit = Online\ mean\ consumption - X \times Online\ standard\ deviation \tag{2.10}$$

The value of X affects the trade-off between false positives (FPs) and false negatives (FNs). A higher X leads to more FNs, while a lower X results in more FPs. Therefore, selecting the right value of X that aligns with the problem's requirements is vital. The mean consumption and the boundary are automatically modified when new instances are added, enabling continuous monitoring of anomalies in the streaming data.

Inter Quartiles Method (IQR): A quartiles-based solution is a method for dividing a dataset into four equal parts or quartiles, with each quartile representing 25% of the data. This technique is commonly used to identify the spread of data and locate outliers. For computing the quartiles, the dataset is first sorted in ascending order. The initial quarter $(Q1)$ denotes the value at the 25th percentile, meaning that 25% of the data is below this value. The balance point $(Q2)$ signifies the median or the value at the 50th percentile. The third quartile $(Q3)$ signifies that the value at the 75th percentile is below this value. The interquartile range (IQR) is the boundary between the first quartile $(Q1)$ and the third quartile $(Q3)$ of a dataset. To identify outliers in the dataset, we can compute the maximum and minimum scores using Eqns. 2.11 and 2.12:

$$Maximum = Q3 + 1.5 \times IQR \tag{2.11}$$

$$Minimum = Q1 - 1.5 \times IQR \tag{2.12}$$

Any data points that are above the maximum or below the minimum can be considered outliers.

Z-Score Method: The z-score, also known as the standard score, is a measure indicating how far an instance is from a population's mean. It is calculated by finding the count of standard deviations that a data point is above or below the mean. To calculate the z-score for a specific data point

$$z = \frac{(x-\mu)}{\sigma} \quad (2.13)$$

where x represents the data point, μ constitutes the mean of the population, and σ is the standard deviation of the population. The standard score can also be calculated in real-time using a sliding window approach. In this case, the moving or running z-score is calculated by considering a window size (w) and computing the mean and standard deviation only over the previous w instances. The moving z-score calculation provides an estimate of the deviation of each instance from the average of the previous (w) instances quantified in units of standard deviations.

Some commonly used metrics for assessing unsupervised anomaly detection algorithms include the silhouette score [203], Calinski-Harabasz score [204], Davies-Bouldin score [205], and standard deviation, for which the values are presented for the CreditCard streaming dataset. In the CreditCard streaming dataset, only the "Amount" column contains data on normal and fraudulent transactions, while the remaining columns do not contain original data for security reasons. The complete evaluation determines normal and abnormal instances by referring to the "Amount" column of the dataset. All techniques process instances sequentially and evaluate the required rolling statistics, such as rolling mean and standard deviation. A threshold value is crucial in deciding whether the current instance is usual or anomalous, and it establishes the boundary for all techniques. In the implementation phase, normal instances are represented by 0, and abnormal instances are represented by 1. The performance scores of different methods are recorded in Table 2.7. For testing purposes, 500 instances, including both anomalous and non-anomalous instances, have been randomly selected and visualized using scatter plots for all three methods in Figs. 2.5 to 2.7.

Silhouette Score: The Silhouette Score is typically used as a metric for evaluating the quality of clustering results. However, in the context of abnormality detection, the silhouette score can also be used as a measure of how well the individual data instances fit into their required clusters, and therefore, how anomalous they are. Mathematically, the silhouette score for an individual instance i can be represented as

$$S(i) = \frac{(b(i)-a(i))}{max(a(i),b(i))} \quad (2.14)$$

where $a(i)$ is the average distance between i and all other points in the same cluster, and $b(i)$ is the minimum average distance between i and all points in any other cluster. The denominator is the maximum of $a(i)$ and $b(i)$, which ensures that the score is normalized between -1 and 1.

Calinski-Harabasz Score (Variance Ratio Criterion): The Calinski-Harabasz Score can be used to identify irregularities in streaming data by monitoring the score over time and detecting when it deviates significantly from the expected range. The mathematical representation of the Calinski-Harabasz Score is as follows.

$$CHS = \frac{(B/(k-1))}{(W/(n-k))} \quad (2.15)$$

where CHS is the Calinski-Harabasz Score, B is the between-cluster dispersion, W is the within-cluster dispersion, k is the number of clusters, and n is the total number of observations.

Davies Bouldin Score: The Davies-Bouldin Score is a measure of cluster quality in clustering analysis. It is based on the average homogeneity between each cluster and its most homogeneous cluster, as well as the distance between their centroids. The mathematical details for the Davies-Bouldin Score are as follows:

$$DBS = \left(\frac{1}{k}\right) \times \sum_{i=1}^{k} \frac{(R_i + R_j)}{D(c_i, c_j)} \; max(i \neq j) \quad (2.16)$$

where DBS is the Davies-Bouldin Score, k is the number of clusters, R_i is the average distance between each point in the cluster i and the centroid of the cluster i, c_i is the centroid of the cluster i, $D(c_i, c_j)$ is the distance between the centroids of clusters i and j, and $max(i \neq j) \; (R_i + R_j) / D(c_i, c_j)$ is the maximum homogeneity between clusters i and any other cluster. The Davies-Bouldin Score can be used as a real-time indicator of anomalies in streaming data, enabling prompt action to be taken to address any issues that may arise.

Standard Deviation: Standard deviation is a statistical measure of the spread or variation of a dataset. It is the square root of the average of the squared deviations from the mean. The mathematical description of standard deviation is as follows.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x-\mu)^2}{n}} \quad (2.17)$$

where σ is the standard deviation, x is each data point in the dataset, μ is the mean of the dataset, i is the counting of data point 1 to n, the total count of data points in the dataset, and Σ represents the sum of all the terms inside the parentheses. The performance scores of unsupervised statistical-based anomaly detection using the CreditCard streaming dataset are shown in Table 2.7.

Table 2.7. Performance evaluation scores of unsupervised statistical-based anomaly detection techniques using the CreditCard streaming dataset.

S.No.	Performance Evaluation Metrics	Unsupervised Statistical Techniques for Detecting Anomalies		
		Welford Method	IQR Method	Z-Score Method
1	Silhouette Score (ss)	0.01860	0.01507	0.02005

2	Calinski Harabasz Score (cbs)	26.84829	22.02712	38.76785
3	Davies Bouldin Score (dbs)	10.65653	16.42220	4.72983
4	Standard Deviation (std)	50.60552	33.06014	89.75449

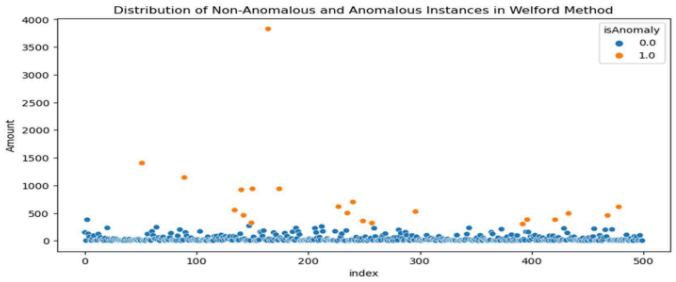


Fig. 2.5. Distribution of Non-Anomalous and Anomalous Instances in CreditCard Streaming Dataset using Welford Method.

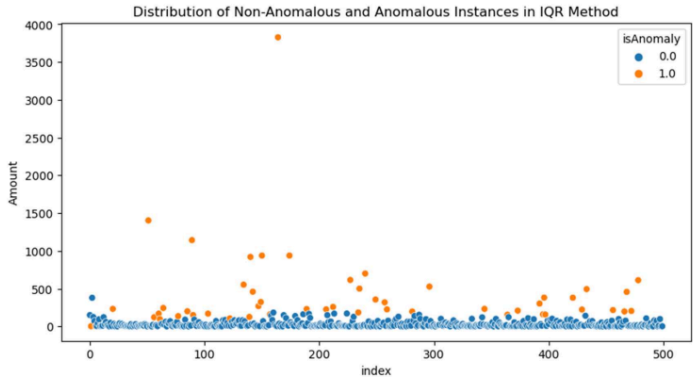


Fig. 2.6. Distribution of Non-Anomalous and Anomalous Instances in CreditCard Streaming Dataset using IQR Method.

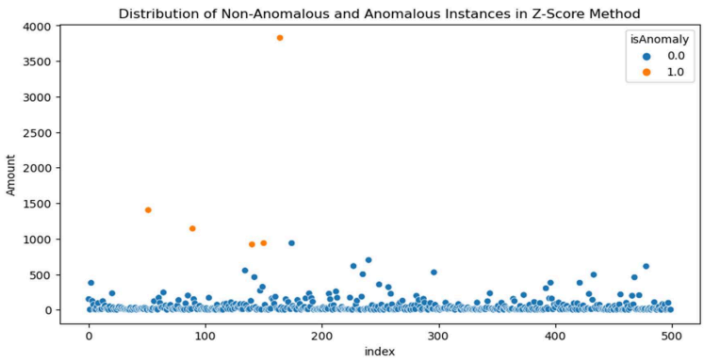


Fig. 2.7. Distribution of Non-Anomalous and Anomalous Instances in CreditCard Streaming Dataset using Z-Score Method.

2. Supervised Techniques

a) Machine learning-based anomaly detection methods

Machine learning-based anomaly detection techniques utilize supervised or semi-supervised machine learning algorithms [206] to automatically learn patterns in the data stream, and then employ these patterns to identify data points that fall outside the expected range. These techniques are often used when the normal and abnormal behavior of the data is known, and class labels are available for instances in the data stream [208]. These techniques can be more effective than traditional statistical methods because they can automatically identify trends in the data and adapt to changing circumstances. Some common machine learning-based techniques have been categorized into supervised, unsupervised, and semi-supervised methods, as shown in Fig. 2.8.

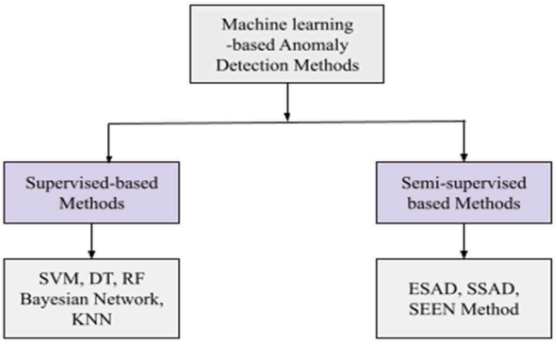


Fig. 2.8. Categories of machine learning-based anomaly detection methods.

1. Supervised-based anomaly detection methods

Support Vector Machines (SVM): A supervised learning approaches that identify anomalies as data points that are difficult to classify [208].

Decision Trees (DT): These are supervised learning models that identify anomalies as data instance that drop into leaf nodes with a less probability of occurring [139], method that uses decision-tree and deep learning to identify abnormalities from traffic cameras and accurately determine the beginning and end time of the abnormal event, such strategy involved developing a detection model, then identifying and analysing the anomalies [209].

Random Forests (RF): These are ensemble learning models that combine multiple decision trees and identify anomalies as data points that are not well-represented by most of the trees [210]. An efficient ensemble-based approach [28] presents a quick one-class anomaly detector for changing data streams. It only needs usual data for training and performs well when discordant data is scarce. The method includes a group of random HS-Trees, and the tree structure is built without any sample, making it highly cost-effective as it does not need any method restructuring when adjusting to dynamic data streams. A reasonable explanation for non-parametric anomalies based on how an unobserved point affects the rest of the data, specifically the impact that point has on the remainder of the data [31].

Neural Networks (NN): These are supervised learning models that can model complex relationships in the instance and identify anomalies as data instances with a

large error [212]. A Probabilistic Neural Network (PNN) [211] is used to identify abnormalities resulting from a cybercrime, employing an anomaly detector.

2. Semi-supervised-based anomaly detection methods

The process combines supervised and unsupervised learning, with the model being trained on both a small labeled dataset and a large unlabeled dataset. The method employs Streaming Half-Space Trees (HS-Trees) [213], a rapid one-class anomaly detector for dynamic data streams, for this purpose. It requires only an ordinary sample for training and performs better when anomalous instances are rare. This method becomes important because of its high adaptability. A novel one-class semi-supervised method has been employed to detect abnormalities in data streams. The model utilizes a quick and precise density estimator called RS-Forest [144], which is made up of multiple completely randomized space trees. The density estimate for each tree node is obtained through the RS-tree and used to estimate the results for each incoming instance in the data stream, which is then averaged over all trees in the forest. A new method called SEEN [214] consists of three main parts: a novel class detector utilizing clustering random trees, a robust classifier for named class predictions, and an efficient modification method that automatically adapts to environmental changes.

Machine learning-based techniques have several advantages. For example, these techniques can handle high-dimensional data effectively [122]. Additionally, these techniques can handle noisy and incomplete data and can learn from numerous examples [207]. However, machine learning-based techniques also have some limitations. For example, these techniques need an enormous number of labeled instances for training, which may not be available in practice. Additionally, these techniques may not be effective in detecting subtle or rare anomalies [212].

Overall, machine learning-based anomaly detection approaches are effective in identifying anomalies across a wide range of applications. A hybrid methodology for anomaly detection in data streams was proposed in [215], which amalgamates machine learning algorithms for both batch and stream processing. The advantage of batch processing is the accuracy it provides, while stream-based algorithms have the advantage of speed, as they respond quickly to real-time anomalies in data streams. These techniques are particularly beneficial when the normal behavior of the instance is known or can be modeled accurately. Some valuable machine learning-based anomaly detection methods are illustrated in Table 2.8.

Table 2.8. A summary of popular supervised and semi-supervised machine learning-based approaches for anomaly detection in data streams.

Ref.	Method	Highlight	Distinctiveness

Chitrakar et al. 2012 [216]	K-Medoids clustering with SVM.	The method aims to enhance the accuracy of anomaly detection by using k-Medoids clustering to pre-process the data and identify clusters of similar instances, and then using SVM classification to classify instances within each cluster as normal or anomalous.	This method effectively handles the high-dimensional data and boosts the accuracy of anomaly detection.
Ma et al. 2016 [217]	Classification-based anomaly detection approach	A novel supervised classification framework for anomaly detection in uncertain data streams utilizes wavelet soft-thresholding to eliminate noise and inaccuracies from streaming data.	Efficient pattern recognition and feature extraction improve computational efficiency in the corrected data stream.
Guha et al. 2016 [31]	Random Cut Forest anomaly detection method for streaming data	The method is designed to be robust against changes in the distribution of the data instance over time, and against various types of noise and outliers. The paper also provides experimental results on several datasets to compute the performance of the proposed method and compare it with other state-of-the-art methods.	The RCF algorithm is based on a randomization technique that generates multiple random cuts through the data, and then uses these cuts to build a forest of DTs.
XU et al. 2019 [218]	Semi-supervised anomaly detection framework in fog computing	This method includes two elements: a fog-based infrastructure and an AI engine with fog assistance. The fog-enabled infrastructure provides affordable computing resources for selecting the best learning approach and determining the optimal sampling fraction. Meanwhile, the fog-assisted AI engine employs semi-supervised learning models for anomaly detection, ensuring effectiveness, reliability, and achieving high accuracy and computational efficiency.	Utilizes the fog computing architecture, where data is processed closer to the source, handling the vast amount of data generated by IoT devices and reducing communication overhead.

Bhati et al. 2020 [219]	Intrusion Detection using Support Vector Machine	The simulated results were evaluated using detection outcomes, AUCROC, and a confusion matrix. The NSL-KDD dataset, commonly used for evaluating intrusion detection techniques and comprising a large number of network addresses, was utilized to evaluate the results of the SVM model. The analysis found that Linear SVM, Quadratic SVM, Fine Gaussian SVM, and Medium Gaussian SVM achieved total detection scores of 96.1%, 98.6%, 98.7%, and 98.5%, respectively.	This research provides a descriptive comparison of the performance of different methods, highlighting the strengths and weaknesses of each approach.
Kim et al. 2021 [212]	Supervised machine learning domain-specific knowledge for anomaly detection	A new IDS using a combination of sequential data preserving log embedding methods and anomaly detection techniques. The experiments for this model, which involve embedding system call logs and detecting anomalies, performed well, achieving an AUROC score of up to 0.8708. With the use of supervised classification and sufficient labeled data on attack logs, this score has the potential to improve further to 0.9745.	It enhances the results of intrusion detection by preserving sequential data in log files and utilizing this information to identify anomalies.
Aboah et al. 2021 [209]	Decision tree-based Deep neural network (DNN) for anomaly detection	The anomaly detection process involves estimating traffic scene backgrounds, extracting road masks, and applying adaptive thresholding. A decision tree is used to evaluate potential anomalies and describe the final abnormalities. Through exploratory validation, this approach achieved an F1 score of 0.8571 and an s4 score of 0.5686.	A vision-based approach to traffic anomaly detection, which allows the system to analyze video footage of traffic and identify abnormal events in real-time.
Iegorov et al.	Semi-supervised	SIM-AD is a semi-supervised method to identify anomalies in	This is a parameter-less

2021 [220]	Anomaly Detection method (SIM AD)	univariate time series instances that does not require any user-specified components. It utilizes three publicly accessible time series datasets from diverse fields, and its cost-effectiveness is compared to that of the PAV and RRA anomaly detection methods. SIM-AD was found to be more accurate than the evaluated methods when applied to trendless time series data.	nature and its ability to handle univariate time series instances, which is unique among the other state-of-the-art methods.

b) Supervised Ensemble-based Anomaly Detection Frameworks

Supervised ensemble methods for anomaly detection are relatively less common than unsupervised methods, as anomalies are often rare and difficult to label. A new type of ensemble, which is made up of a combination of an adaptive random forest and a HAT [24], performed the best in dealing with changes in data (concept drift) while also having a slightly better run time, making it more efficient for online training. This ensemble was not as accurate as a larger adaptive random forest, but it is a trade-off worth making for the improved performance in dealing with concept drift. This study found that two ensembles [221] performed particularly well: one that combined an ARF and the HAT approach, and another that solely used the Hoeffding Adaptive Tree algorithm. In this study, the performance of several classifiers [123], including Bagged Trees, AdaBoost, RUSBoost, LogitBoost, and GentleBoost, was compared. The performance of these classifiers was analyzed, and their ability to learn from the data was evaluated. Ensemble-based methods are known to be more robust and resistant to overfitting compared to single models; however, they may also lead to an increase in computational complexity. This may be categorized as shown above in Fig. 2.8. Some of the supervised ensemble methods that can be used for abnormality detection include:

Boosting-based methods: Boosting algorithms, such as AdaBoost and XGBoost, can be applied for anomaly exploration by training multiple weak classifiers and combining them to form a robust classifier.

Random Forests: Random Forests can also be used for anomaly detection by constructing multiple decision trees and joining their results to identify anomalies.

Gradient Boosted Trees: A supervised learning approach that can be used for anomaly detection by iteratively building decision trees and boosting the system's accuracy.

Support Vector Machines (SVMs): SVMs can also be used for anomaly detection in a supervised setting by training on labeled data and classifying new data instances as either normal or anomalous. The most valuable supervised ensemble-based anomaly detection methods are shown in Table 2.9.

Table 2.9. A summary of popular machine learning-based supervised ensemble approaches for anomaly detection in data streams.

Ref.	Method	Highlight	Distinctiveness
Cheng et al. 1999 [222]	Semi-supervised stacked hierarchical model for anomaly detection	A novel concept for anomaly detection in the Internet of Things (IoT) using a semi-supervised hierarchical stacking temporal convolutional network (HS-TCN). The results show that the model significantly enhances security in smart home communication while achieving substantially improved performance.	A hierarchical stacking of temporal convolutional layers to augment the accuracy of anomaly detection in IoT systems.
Dong et al. 2016 [223]	Supervised a base online ensemble model	The research presents two swift anomaly detection models based on neural networks as ensembles for streaming data. One of the models uses supervised online learning and a group of threaded multilayer perceptrons, while the other employs a one-class learning algorithm and a set of threaded autoencoders	This enables the model to learn from both labeled and unlabeled instances, and to update accordingly as the data changes.
Mwallem et al. 2017 [224]	Hoeffding Tree-based ensemble for streaming data	An outline of the various methods introduced in the related works for using Hoeffding trees for anomaly detection issues in streaming cyber-related data. The aim is to demonstrate how this approach can effectively address this prevalent problem.	The method provides a descriptive overview of the recent state of the art in using Hoeffding trees for anomaly identification in stream-based datasets.
Amutha et al.	Supervised and clustering	The author presents a method for detecting anomalies in streaming Phasor Measurement Unit (PMU)	This approach allows the model to handle the high-

2021 [225]	ensemble method	data. It is a pairing of supervised and clustering ensemble approaches that classify the PMU data into normal and abnormal patterns.	dimensional and dynamic nature of the PMU data.
Vergaftik et al. 2021 [226]	Decision Tree-based ensemble classifier (RADE)	RADE is a novel anomaly detection approach that enhances the performance of decision-tree-based ensemble classifiers under resource constraints. The approach is to first train a small model that can accurately classify most queries, thereby saving resources.	RADE can achieve high detection accuracy while using significantly fewer resources than other state-of-the-art anomaly detection processes.

2.3 Review of Online Learning Strategies for Real-Time Data Streams

Air pollution is a major health concern in urban areas due to toxic gases from industrial and vehicular emissions, as well as dust from construction activities [57]. Industrial exhausts also contain high levels of volatile organic compounds and particulate matter that can harm human health [58]. Traditional machine learning algorithms, such as support vector machines and random forests, are often used for offline air quality classification [58, 59]. However, real-time monitoring in polluted urban areas is crucial, requiring online machine learning to detect toxic gases quickly. Real-time anomaly detection helps authorities take preventive actions, such as closing schools or implementing traffic policies like the odd–even rule. In Delhi, air quality worsens from October to February due to factors like crop burning and festival firecrackers. Continuous monitoring of pollutants such as PM 2.5, PM 10, CO, CO_2, SO_2, SO_3, NO, NO_2, NH_3, and O_3 is needed for effective real-time air quality monitoring [60].

Streaming data mining faces challenges such as class imbalance, which reduces the predictive performance of online learning models. Many existing approaches are not well-suited to handle large-scale streaming data with imbalanced class distributions. To address this, the cost-sensitive regularized dual averaging (CSRDA) model extends the regularized dual averaging method by introducing a

convex optimization function to improve learning from imbalanced streams [61]. Liu et al. proposed an anomaly detection framework for infrequent data streams that uses a monitoring module for dynamic window selection and a tracking module to evaluate irregular observations [62]. Similarly, Boateng et al. developed an unsupervised anomaly-detection approach based on a stacked ensemble with an isolation forest as the meta-learner to detect anomalies in industrial control systems [63]. In IoT streaming environments, class imbalance is addressed using a dynamic ensemble method combined with borderline-SMOTE to generate normal samples, thereby improving anomaly detection performance [64].

2.4 Related Work on Ensemble Learning for Concept Drift and Network Intrusion Detection in the Network Streams

Recent advancements in Network Intrusion Detection Systems (NIDS) highlight the effectiveness of online machine learning and ensemble approaches for analyzing real-time network traffic and identifying evolving cyber threats. Malathy et al. [65] proposed a self-regulating intrusion detection framework that manages network drift with window-based feature pruning and optimization. A sliding-window drift-detection method using an isolation forest variant was introduced for anomaly detection [66]. Togbe et al. [67] created a unified framework for anomaly and drift detection using an isolation forest with ADWIN and KSWIN. Horchulhack et al. [68] developed a three-phase stream-learning approach employing an online classifier and a single-class model to detect novel network activities. Mulimani et al. [69] introduced an ensemble approach for attack detection and concept drift handling. In industrial control systems, anomaly detection using one-class classification analyzes real-time sensor data [70]. LSTM models identify abnormal energy patterns in smart meter data [71]. Recently, a hybrid ensemble framework was proposed to detect anomalies, address drift, and reduce retraining [72].

Table 2.10 highlights research gaps and issues in existing classification models for several popular anomaly detection techniques.

Table 2.10. Existing classification models have research gaps and issues.

Approaches	Descriptions	Research Gaps and Issues

Adaptive online incremental learning for dynamic streams	An online incremental learning approach [75, 76] faces challenges such as drifting, forgetting prior knowledge, and a limited prediction space. It addresses these challenges with adaptive online learning for streaming data, using an autoencoder with memory to extract features and minimize reconstruction loss.	This approach suits single-task streams, but not multi-task streams. Research areas include recurring contexts, feature evolution, detecting novel classes for single- or multi-class problems, and mechanisms of forgetting.
Adaptive learning-based classification model	An adaptive learning-based approach [74] addresses changes in data over time by using concepts of selective training sets. It handles sudden, gradual, and recurring drift with four algorithms, offering faster processing, less memory use, and improved accuracy for real-time stream applications.	This classification method offers good accuracy and adapts well to small real-time streaming data, but is costly and less accurate for large datasets. Our proposed drift detection and adaptation model maintains high accuracy while keeping computation time low across all dataset sizes.
Adaptive learning process	An adaptive learning method [5] addresses changes in input data and target outputs over time to handle drifting concepts. It categorizes techniques for managing drift and emphasizes evaluation processes.	Most adaptive algorithms assume changes occur in a hidden mode, but this isn't always true; drift is unpredictable, making its detection and monitoring crucial challenges for drift-adaptive processes.
A novel adaptive prototype-based kernel classifier for streaming data	The machine learning approach [78] addresses data stream challenges, such as training time, memory, and drift, by using a kernelized nearest-neighbor classifier with a sparse prototype dictionary. It continuously adapts by updating prototypes, achieving lower complexity and better accuracy.	This approach handles complexity and error rate well, but for gradual and abrupt drift, learning becomes unstable, and accuracy drops. It sometimes causes errors during online updates and dictionary pruning.

Online anomaly detection for streaming data	The method [79] detects anomalies in telecommunication traffic using entropy, signatures, and machine learning, combined with multifractal and recurrence analysis. It effectively distinguishes normal and attack traffic and is validated in a virtual network against existing methods.	This method lacks support for distributed online learning, incurs high training and detection time, and underperforms compared to packet learning techniques, leaving a significant research gap.
Outlier detection for streaming data	Real-world streaming anomaly detection is challenging due to high volume and speed. The ensemble-based framework [80], based on isolation forest variants, achieves efficient, low-memory detection with strong AUC, F1, and time performance.	This method effectively detects anomalies but is limited to specific parameters, making it less applicable as a global, real-time framework. A pair of alternating windows is added to improve this model's performance. This approach is inefficient for comparing multiple detections. Not supportive of the multi-dimensional approach.
IoT- based approach for smart sensor devices	The IoT-based approach [81] uses intelligent sensors to automatically connect smart devices, enhancing security and enabling efficient integration of measurements in smart applications.	A key gap in semantic-based approaches is the inability to integrate semantic models with machine learning simultaneously.
IoT-based streaming data for healthcare	The machine learning framework [82] collects IoT data from multiple sources, performs feature selection, and analyzes sensor signals to reduce medical faults, healthcare costs, and improve patient monitoring.	This approach handles only low-sensitivity IoT data effectively, with limited performance on highly sensitive data.
Machine learning-based approach for IoT	With rapid IoT growth, large and diverse data complicates access. A machine learning approach [83] focuses on smart applications by extracting higher-level information from multi-source IoT data.	This approach has limitations in handling multiple IoT data sources and requires an effective processing mechanism.

2.5 Problem Statement

In modern data-driven applications, data streams challenge traditional classifiers that rely on static learning, causing inefficiency in real-time environments. Online machine learning offers adaptive learning, but there's a need for a robust, efficient method for accurate classification and anomaly detection in dynamic streams.

- **Limitations of Traditional Methods:** High memory usage, frequent retraining, poor adaptability, and limited real-time capability.
- **Need for Online Learning:** Supports incremental updates, low memory usage, and real-time prediction.
- **Motivation for Proposed Approach:** Requirement of an efficient, adaptive, and scalable framework to handle concept drift and improve anomaly detection in streaming data.
- **Handling Evolving Data Distributions:** Continuous changes in data patterns (concept drift) require models that can adapt dynamically without degrading performance over time.

CHAPTER 3

Performance evaluation of online learning strategies on real-time streaming data

3.1 OBJECTIVE-3 (a): Real-time Air Pollution Monitoring in Delhi Using Online Anomaly Detection Algorithms for Streaming Data

Citation: Ray, Santosh Kumar, and Seba Susan. "Real-time Air Pollution Monitoring in Delhi Using Online Anomaly Detection Algorithms for Streaming Data." *National Academy Science Letters* (2025):1-3. **https://doi.org/**10.1007/s40009-025-01739-7. **(Springer), (SCIE-indexed, IF: 1.3), (Published).**

3.1.1 Methodology

This chapter presents an unsupervised online machine learning framework for real-time air quality monitoring implemented using the River and PySAD streaming frameworks [2, 86]. A total of nine diverse anomaly detection models: HST, QF_OCSVM, QF_HST, and TF_HST from the River framework, and HBOS, KitNet, RRCF, IForestASD, and KNNCAD from the PySAD framework are employed based on their ability to capture both global anomalies (e.g., stubble burning) and local anomalies (e.g., traffic emissions).

Real-time hourly data of eight air pollutants (PM2.5, PM10, NO, NO2, SO2, CO, NH3, and O3) were collected from 20 monitoring stations representing the most polluted locations in Delhi, sourced from the official Central Pollution Control Board (CPCB)[3] portal, covering the period from May 2022 to April 2023. Real-time air quality monitoring involves measuring the concentration of these pollutants in the atmosphere [60]. The average Air Quality Index (AQI) across all stations was computed for ground-truth labeling, where AQI > 100 indicates anomalous conditions (annotated as 1), while lower values represent normal conditions (annotated as 0**).**

The study applies unsupervised online machine learning to enable real-time detection without prior training, thereby supporting timely decision-making for pollution control measures such as traffic restrictions and institutional closures. This approach establishes a continuous, unified, and adaptive streaming-based anomaly detection framework for real-time air quality assessment, particularly relevant for urban environments like Delhi, where pollution levels significantly worsen during winter months due to stubble burning and festival-related emissions such as Diwali[4].

[3]https://airquality.cpcb.gov.in/ccr/#/caaqm-dashboard-all/caaqm-landing/data.
[4]https://en.wikipedia.org/wiki/Air_pollution_in_Delhi.

3.1.2 Experimental Setup and Result Analysis

All experiments were conducted using Jupyter Notebook (version 8.15.0) on a system configured with an Intel Core i5-8265U processor and 8 GB of RAM. Data points achieving anomaly scores [0.5, 1] are considered as anomalies, while those having low anomaly scores [0, 0.5) are regarded as normal. The descriptions of the Delhi air pollution AQI dataset are provided in Chapter 1, Section 1.5.4.

We configure the model hyperparameters after a grid search from coarser to finer scales. The optimal River model hyperparameter settings [2] are: HST- 10 trees with height of 8 and a threshold value of 0.80; QF_OCSVM-expected anomalies fraction (μ) of 100 and quantile size (q) of 0.69; QF_HST- quantile size (q) of 0.58; TF_HST-threshold value (HST) of 0.99, and final threshold of 0.95.

Similarly, the optimal hyperparameter setting of the PySAD model settings [86] are: HBOS - 10 bins with regularizer value of 0.1; KitNet- 20 autoencoders with learning rate of 0.05, and the ratio of hidden to visible neurons is 0.9; RRCF- 4 trees of size 256 and shingle size of 4; IForestASD- window size of 2048; KNNCAD- probationary period with output 0.0. The performance was evaluated in terms of Accuracy, ROCAUC, Recall, and Geometric Mean. The dataset and code are made available online[5]. The results of the experiment are summarized in Table 3.1.

Table 3.1. Performance scores of nine unsupervised anomaly detection streaming models.

S. NO.	Methods	Real-time Pollution Dataset			
		Accuracy	**ROCAUC**	**Recall**	**Geometric Mean**
1	HST	73.16%	0.4929	0.0095	0.0961
2	QF_OCSVM	87.04%	0.8603	0.8399	0.8601
3	QF_HST	69.71%	0.7931	0.9874	0.7689
4	TF_HST	74.38%	0.4979	0.0009	0.0300
5	HBOS	74.36%	0.7547	0.7771	0.7543
6	KitNet	65.95%	0.6647	0.6752	0.6646
7	RRCF	72.00%	0.6798	0.6062	0.6757
8	IForestASD	61.69%	0.5938	0.5471	0.5919
9	KNNCAD	63.72%	0.4962	0.2106	0.4057

[5]https://github.com/Santosh Kumar Ray/Real-time-Air-Pollution-Monitoring-in Delhi-Using-Online-Anomaly-Detection-Algorithms.

After evaluating the performance of the nine unsupervised online anomaly detection models, in terms of Accuracy, ROCAUC, Recall, and Geometric Mean, the QF_OCSVM model demonstrated the best results, achieving an accuracy of 87.04%, ROCAUC of 0.8603, Recall of 0.8399, and Geometric Mean of 0.8601. QF_HST and HBOS were the second-best and third-best performing models, respectively, implying that Quantile Filters (QF) and univariate feature density histograms are highly useful for the detection of global anomalies in pollution data streams. Recall and Geometric Mean, followed by ROCAUC, are highly useful performance indices to rate model performance when the number of anomalies is much lower than the number of normal data samples. In this context, HST and TF_HST showed the worst performance, implying that they are not suitable for the detection of global anomalies clustered together in a short time frame. As observed from Fig. 3.1, which presents visually annotated anomaly scores across models for the period from May 2022 to April 2023, QF_OCSVM is the most effective online streaming algorithm that provides a distinctive anomaly pattern with pollution peaks observed between October to February, aligning with the duration when crop residue/ stubble is burnt in North India and festivals like Diwali are celebrated with widespread firecracker use.

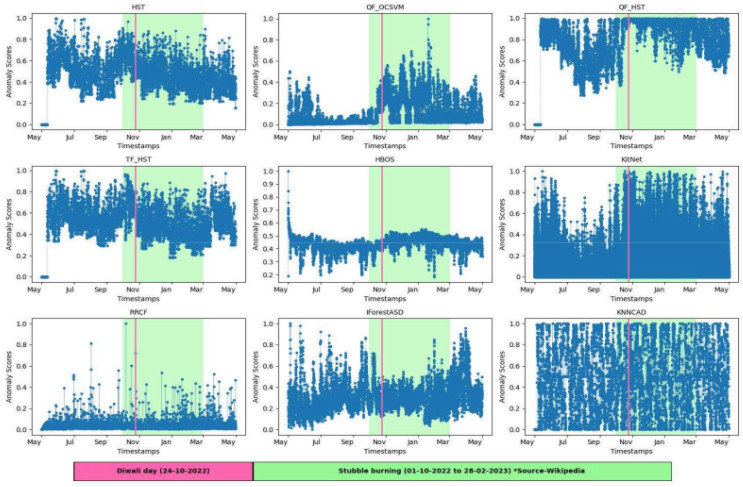


Fig. 3.1. Timestamps versus anomaly scores plot for nine unsupervised anomaly detection methods (HST, QF_OCSVM, QF_HST, TF_HST, HBOS, KitNet, RRCF, IForestASD, and KNNCAD) using real-time pollution data procured in real time from 20 base stations in the duration of one year, using annotations for the festival Diwali and stubble burning.

3.2 OBJECTIVE-3 (b): Performance analysis of online machine learning frameworks for anomaly detection in iot data streams.

Citation: Ray, Santosh Kumar, and Seba Susan. "Performance analysis of online machine learning frameworks for anomaly detection in iot data streams." *In 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1-5. **DOI:**10.1109/ICCCNT61001.2024.10724326. **(IEEE), (Published).**

3.2.1 Methodology

Additionally, this chapter focuses on the process flow for anomaly detection in data streams, which is adapted to evaluate the classification performance of two recently introduced streaming frameworks, River and PySAD. The online machine learning models in both frameworks are trained unsupervised on streaming data. The IoT-based streaming dataset, IoTID20, is used for the evaluation. The performance metrics are Accuracy, Sensitivity, Specificity, and ROCAUC. The details of the River and PySAD models are given below, and the process flow diagram is shown in Fig. 3.2.

1. River Framework

1. Half Space Trees (HST): Tan et al. [28] executed a fast one-pass streaming-based model named HST. The HS-Trees method exclusively utilizes normal data during its training phase because anomalous data is infrequent and might not be accessible for

the training process. Updating the model is both straightforward and rapid since it necessitates no alterations to the tree structure when handling streaming data.

2. Quantile Filter One-Class Support Vector Machine (QF_OCSVM): During the initial training phase, OCSVM focuses on normal data, creating a boundary encompassing these non-anomalous data points in a high-dimensional space. This involves setting a decision boundary to separate normal instances from the rest of the space. The use of One-Class SVM involves integrating a Quantile Filter to enhance and fine-tune the anomaly detection process.

3. Quantile Filter Half Space Trees (QF_HST): The pipeline starts by normalizing features using a min-max scaler and follows up with anomaly detection using Half Space Trees within a Quantile Filter, emphasizing data points within a specific quantile. This configured pipeline is ready for both training on existing data and making predictions on new data instances.

4. Threshold Filter Half Space Trees (TF_HST): The anomaly scores generated by the Half Space Trees are then passed through a Threshold Filter. This filter is configured with a specific threshold value, suggesting that instances with anomaly scores above this threshold will be considered anomalies. This configured pipeline is ready for both training on existing data and making predictions on new data instances.

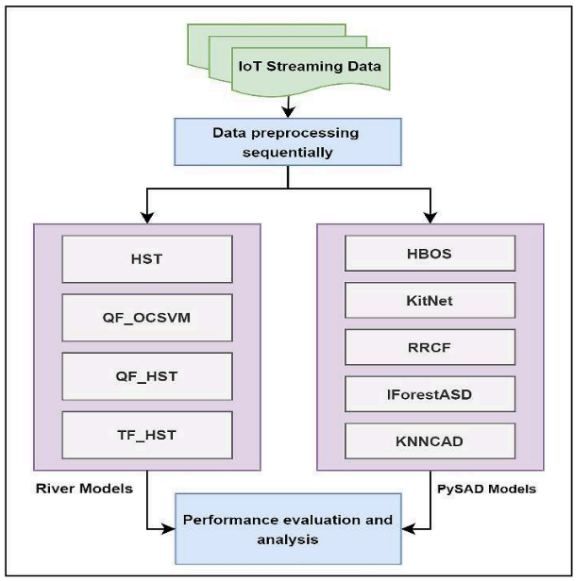


Fig. 3.2. Process flow diagram of the system**.**

2. PySAD Framework

1. Histogram-Based Outlier Scores (HBOS): Goldstein et al. [29] designed an unsupervised outlier detection algorithm called HBOS, which uses histograms to represent the densities of univariate features. During the training phase, both normal and abnormal observations are employed, and the model identifies outliers as part of the fitting process. HBOS utilizes a referential windowing approach; the window size determines the quantity of data points employed for model training, and the sliding size governs how frequently the model is updated.

2. Kitsun's core Algorithm (KitNet): Mirsky et al. [30] designed a network intrusion detection-based methodology known as KitNet. The KitNet model utilizes unsupervised autoencoders, where only normal samples are involved in the training phase. Abnormal samples are excluded from training and are later employed for anomaly detection.

3. Robust Random Cut Forest (RRCF): Guha et al. [31] concentrated on a dynamic streaming-based anomaly detection concept. In this concept, the model learns a favorable score threshold using a training set and evaluates its effectiveness on a separate test set. The training set comprises all points before time t, while the test set comprises all points after time t.

4. Isolation Forest Anomaly Streaming Data (IForestASD) [32]: The IForestASD algorithm is unsupervised and does not require subclass attributes in the anomaly detection process. Anomaly labels are used solely to assess the last anomaly detection performance. When updating the detection model, there is an option to discard old instances while simultaneously incorporating new ones selectively.

5. K Nearest Neighbors Conformal Anomaly Detection (KNNCAD): Burnaev et al. [33] deployed a standardized distance and density-based approach during the training phase. This approach calculates the probability that for each new observation, consideration with a more utmost value of a Non-Conformity Measure (NCM) exists in the training set compared to the NCM value of the new observation.

3.2.2 Experimental Setup and Results

Additionally, in this paper, we utilize the online machine learning concept and employ well-known open-source libraries for streaming data, PySAD [34, 115] and River (ver. 0.20.0) [2], along with the scikit-learn Python library [19], for anomaly detection in IoT data streams. The same experimental setup was used as described in an earlier chapter. To assess the performance of the River and PySAD framework's anomaly detection models, a comparison was made against four River-based anomaly detection models: HST, QF_OCSVM, QF_HST, and TF_HST, and five PySAD-based anomaly detection models: HBOS, KitNet, RRCF, IForestASD, and KNNCAD. The evaluation was based on four reliable metrics, namely Accuracy, Sensitivity, Specificity, and ROCAUC. The performance evaluation was conducted using the widely used IoTID20 streaming dataset, and the results demonstrated that the River framework has more consistent models than the PySAD framework, which outperformed in terms of satisfactory scores. Typically, the hyperparameter configuration defaults to all nine streaming anomaly detection models.

1. Details of IoTID20 Streaming Dataset

The performance evaluation was carried out using the widely used IoTID20 streaming dataset, as detailed in Chapter 1, Section 1.5.4.

2. Performance Metrics

Let t_p depict true positives, t_n depict true negatives, f_p depict false positives, and f_n depict false negatives.

1. Receiver Operating Characteristics Area Under Curve (ROCAUC): AUC suggests measuring the whole two-dimensional space below the total ROC curve. In other words, AUC quantifies the extent of the area covered by the ROC curve, encompassing all its data points.

2. Sensitivity: Sensitivity evaluates the proportion of original positive instances correctly detected by a method. Mathematical descriptions are shown as

$$Sensitivity = \frac{t_p}{[t_p + f_n]} \quad (3.1)$$

where t_p depicts the count of true positives, f_n signifies the count of false negatives.

3. Specificity: Specificity evaluates the proportion of original negative instances correctly detected by a method. Mathematical descriptions are shown as

$$Specificity = \frac{t_n}{[t_n + f_p]} \quad (3.2)$$

where t_n depicts the count of true negatives, f_p is the count of false positives.

4. Accuracy: Accuracy assesses the entire correctness of the model's predictions by computing the ratio of correct predictions to the total number of predictions made. Mathematical descriptions are shown as

$$Accuracy = \frac{[t_p + t_n]}{[t_p + t_n + f_p + f_n]} \quad (3.3)$$

3. Discussion on Results

The nine online machine learning models from the River and PySAD streaming frameworks are evaluated for their performance in anomaly detection on IoT data streams using reliable metrics such as Accuracy, Sensitivity, Specificity, and ROCAUC. The performance scores are summarized in Table 3.2.

Table 3.2. Performance Metrics of River and PYSAD Framework's Models.

Frameworks	Methods	IoTID20 Streaming Dataset			
		Performance Metrics			
		Accuracy	**Sensitivity**	**Specificity**	**ROCAUC**
River Models	HST	0.8608	1.0000	0.9194	0.4426
	QF_OCSVM	0.6386	0.8485	0.6242	0.6837
	QF_HST	0.9257	1.0000	0.9890	0.7810
	TF_HST	0.8306	1.0000	0.8874	0.4432
PySAD Models	HBOS	0.8393	0.8839	0.8847	0.5040
	KitNet	0.9360	1.0000	1.0000	0.5000
	RRCF	0.8254	0.8791	0.8606	0.4954
	IForestASD	0.5019	0.8346	0.8562	0.7937
	KNNCAD	0.8430	0.8950	0.8937	0.5000

Sensitivity quantifies the correct identification of true anomalies; among River models, HST, QF_HST, and TF_HST show higher sensitivity scores of 1.00, while QF_OCSVM shows an average score of 0.8485. Similarly, among the PySAD models, KitNet shows a higher sensitivity score of 1.00, and the remaining models, HBOS, RRCF, IForestASD, and KNNCAD, achieve average scores of 0.8839, 0.8791, 0.8346, and 0.8950, respectively. Thus, the River models are more consistent than the PySAD models. Specificity measures the correct identification of normal instances. The specificity of River models: QF_HST and HST show higher scores of 0.9890 and 0.9194, while the remaining two models show average scores. Among PySAD models, the KitNet model shows a higher score of 1.00, and the remaining models show satisfactory scores. ROCAUC gauges the model's ability to distinguish between anomalies and normal instances, with a higher ROCAUC indicating superior performance. QF_HST shows a higher ROCAUC score of 0.7810 for the River framework, while the other models demonstrate average performance. Similarly, most of the PySAD models show average ROCAUC scores, except for IForestASD, which shows a higher score of 0.7930. Accuracy assesses the overall correctness of the anomaly detection model. Both framework models show good accuracy, but IForestASD shows an average accuracy score. These performance metrics are valuable in evaluating the performance of online anomaly detection methods and facilitating comparisons between different approaches. The selection of metrics is contingent upon the specific requirements of the application and the significance of different error types in the context of anomaly detection.

Figs. 3.3 and 3.4 visually depict the overall performance comparison of the models in the two frameworks, River and PySAD.

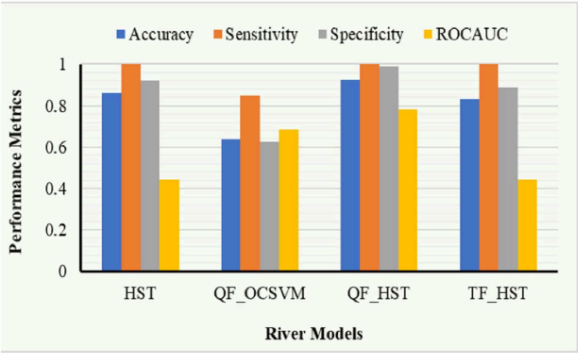


Fig. 3.3. Performance comparison of River framework's anomaly detection models on the IoTID20 streaming dataset.

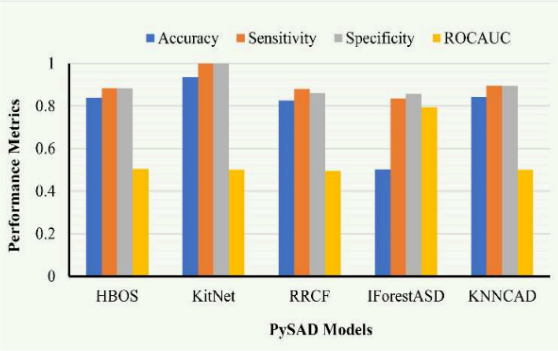


Fig. 1.4. Performance comparison of PySAD framework's anomaly detection models on the IoTID20 streaming dataset.

3.3 Conclusion

This chapter presented an unsupervised online machine learning framework for real-time data stream analysis using the River and PySAD streaming frameworks. The study explored anomaly detection in both environmental (air quality) and IoT data streams, demonstrating the applicability of streaming-based models in dynamic and real-time scenarios. Experimental results indicate that the evaluated models perform effectively in identifying anomalies, with River-based approaches generally showing better performance across key evaluation metrics such as Accuracy, Sensitivity, Specificity, and ROC-AUC.

The findings highlight the potential of online learning techniques for deployment in real-world applications, including air quality monitoring and IoT-based systems. However, challenges such as concept drift, class imbalance, and memory constraints remain significant concerns in streaming environments. Future work will focus on addressing these limitations by incorporating drift-aware mechanisms, handling imbalanced data distributions, and exploring adaptive window-based techniques for improved performance.

CHAPTER 4

Development of ensemble classification framework for streaming data

4.1 OBJECTIVE-1 (a): Error-weighted drift-adaptive ensemble framework for classification of streaming data

Citation: Ray, Santosh Kumar, and Seba Susan. "Error-weighted drift-adaptive ensemble framework for classification of streaming data." *Evolving Systems* 17, no. 2 (2026): 45. **https://doi.org/**10.1007/s12530-026-09810-4**. (Springer), (SCIE Indexed, IF: 2.8), (Published).**

4.1.1 Drift Detection Methodologies

1. Drift Detection Method

The Drift Detection Method (DDM) proposed by Gama et al. [9] is an online learning approach that takes a decision only when the samples are sequentially available for the process. For evaluating the error rate of the i^{th} random point in a sequence of n samples, the Binomial distribution is used. The sequence of samples is denoted by $(\boldsymbol{x_i}, y_i)$, where $\boldsymbol{x_i}$ is the feature vector and y_i is the label; the model-predicted value $\hat{y}_i$ may be either true or false. The rate of error may be considered by using the false observed probability p_i and the standard deviation s_i that is calculated as

$$s_i = \sqrt{\frac{p_i(1-p_i)}{i}} \tag{4.1}$$

This drift detection approach uses two registers p_{min} and s_{min}, that are updated by the values p_i and s_i respectively (corresponding to the i^{th} sample in the incoming data stream), when p_i+s_i is lower than $p_{min} + s_{min}$. This method declares a new context when the warning level k_w is crossed. When the Out-control level k_d is crossed, a change in distribution is declared, and the model starts learning afresh from the incoming data. Conditions for checking the warning level (k_w) is p_i+$s_i \geq p_{min}$+2× s_{min} and for the drifting level (k_d) is $p_i + s_i \geq p_{min}$+3 × s_{min}.

2. Early Drift Detection Method

The Early Drift Detection Method (EDDM), proposed by Baena-Garcia et al. [11], primarily estimates the distribution of distances between consecutive classification errors. DDM outperforms in the case of abrupt drifts, but it cannot accurately detect slow changes. EDDM searches for concept drift after 30 consecutive errors, allowing it to handle gradual drifts more effectively. EDDM can be incorporated into an online learning algorithm to enhance the model's predictions. EDDM considers the distance between two errors, rather than the number of errors. The average distance between two errors is denoted by p'_i and the standard deviation by s'_i, the maximum values for which are denoted by p'_{max} and s'_{max}, respectively. EDDM establishes two threshold values as indicated in Eqn. (4.2) and Eqn. (4.3).

$$\frac{P'_i+2s'_i}{p'_{max}+2s'_{max}} < \alpha \qquad (4.2)$$

regarding the warning level, the values are pre-captured to account for potential context changes.

$$\frac{P'_i+2s_i}{p'_{max}+2s'_{max}} < \beta \qquad (4.3)$$

in the case of the drift level, when concept drift occurs, the learning model resets, and a new model is trained using the values stored after the warning level. The maximum values p'_{max} and s'_{max} are also clear.

3. Adaptive Windowing

ADaptive WINdowing (ADWIN) proposed by Bifet and Gavalda [12] maintains a window of variable size. In this approach, the window size automatically increases when no change in the distribution is detected. When a change in the distribution is detected, ADWIN reduces the window size to ensure the expected value remains stable across all splits of the window. The overall performance of ADWIN is inefficient in terms of time and memory. An updated version of ADWIN2 addresses these issues as explained below.

Consider a sliding window W of length n comprising a sequence of data points $x1, x2, x3, \ldots, xt, \ldots, xn$, where xt is available at time t, and xt is generated as per a distribution Dt. μ'_W is the observed average value of W. For all possible partitions of the window W=W0.W1, where W0 and W1 are sub-windows, tail elements from the older portion are deleted if the expected value of W0 and W1 differ by an amount exceeding $\in_{cut}$. The threshold $\in_{cut}$ is computed as

$$\in_{cut} = \sqrt{\frac{1}{2m} \ln\left(\frac{4}{\delta'}\right)} \tag{4.4}$$

Here, m denotes the harmonic mean of the lengths of W0 and W1. δ' is computed as $\delta' = \frac{\delta}{n}$, where δ is a value between 0 and 1 that represents confidence.

4. Hoeffding's Drift Detection Method using Moving Average Test

Hoeffding's Drift Detection Method using Moving Average test (HDDM_A) proposed by Frias Blanco et al. [13] focuses on detecting significant variations. It uses Chernoff's and Hoeffding's bounds for mathematical considerations. A three-step process is used to detect significant changes in the moving average test for streaming data.

Step 1: Hoeffding's inequality [36]. Let $X_1, X_2, X_3, \ldots, X_n$ be independent random variables such that $X_i \in [a_i, b_i]$ where $i \in \{1, 2, 3, \ldots, n\}$. The average of all random variables may be considered as $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. For the maximum error $\varepsilon > 0$, Hoeffding's inequality may be defined as

$$pr\{\overline{X} - E[\overline{X}] \geq \varepsilon\} \leq e - \left[\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right] \tag{4.5}$$

From the average value $\overline{X}$ and Hoeffding's inequality, we can estimate the error bound ε_δ which can be easily computed for a significance value δ as $\varepsilon_\delta = \sqrt{\frac{1}{2n}\ln\left(\frac{1}{\delta}\right)}$.

Step 2: Let $X_1, X_2, X_3, \ldots, X_n$ and $Y_1, Y_2, Y_3, \ldots, Y_m$ be independent random variables, such that $X_i \in [a, b]$ with $i \in \{1, 2, 3, \ldots, n\}$; $Y_i \in [a, b]$ with $i \in \{1, 2, 3, \ldots, m\}$. The average of the random variables may be considered as $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\overline{Y} = \frac{1}{m}\sum_{i=1}^{m} Y_i$. For maximum error $\varepsilon > 0$, Hoeffding's inequality may now be defined as

$$pr\{\overline{X} - \overline{Y} - (E[\overline{X}] - E[\overline{Y}]) \geq \varepsilon\} \leq e - \left[\frac{2\varepsilon^2}{(n^{-1}+m^{-1})(b-a)^2}\right] \quad (4.6)$$

From the average values $\overline{X}$ and $\overline{Y}$, and Hoeffding's inequality, we can estimate the error bound ε_α which can be easily computed for a significance value α as $\varepsilon_\alpha = (b-a)\sqrt{\frac{n^{-1}+m^{-1}}{2}\ln\left(\frac{1}{\alpha}\right)}$.

Step 3: Average-test (A-test). The hypothesis test determines the acceptance and rejection of the maximum estimated error (ε_α) based on two prominent conditions.

A. Bound checking for type-I error: if $E[\overline{X}] \leq E[\overline{Y}]$, then the inequality will be $pr\{\overline{X} - \overline{Y} \geq \varepsilon_\alpha\} \leq \alpha$.

B. Bound checking for type-II error: if $E[X] \geq E[\overline{Y}] + \xi$ and $\xi > \varepsilon^\alpha$, then the inequality will be $\Pr\{\overline{X} - \overline{Y} \leq \varepsilon^\alpha\} < e^{-\left[\frac{2(\xi-\varepsilon_\alpha)^2}{(n^{-1}+m^{-1})(b-a)^2}\right]}$.

5. Hoeffding's Drift Detection Method using Weighted Moving Average Test

Hoeffding's Drift Detection Method using Weighted Moving Average Test (HDDM_W) was proposed by Frias Blanco et al. [13]. The weighted moving average test (W-test) is stated by the authors to be more efficient than the average test (A-test) because its evaluation is based on weights for detecting changes in the probability distribution. Recently, the samples have more weight than the previous ones. A modification of Hoeffding's inequality is done by using McDiarmid's inequality as explained next.

Step 1: McDiarmid's inequality [116]. Let $X_1, X_2, X_3, \ldots, X_n$ be an independent random variable, where $X_i \in A_i \subseteq [a_i, b_i]$ $i \in \{1, 2, 3, \ldots, n\}$. Let us assume that g is a computable function that maps $A_1 \times A_2 \times \ldots \times A_n \rightarrow R$ and satisfies the independent bounded difference conditions represented by $\vec{X}$ and $\vec{X'}$, such that for i, $\left|g\left(\vec{X}\right) - g\left(\vec{X'}\right)\right| \leq d_i$. $\overline{Y}$ is another random variable which may be represented as $\overline{Y} = g(X_1, X_2, X_3, \ldots, X_n)$ where $\varepsilon > 0$. McDiarmid's inequality may be written as

$$Pr\{\overline{Y} - E[\overline{Y}] \geq \varepsilon\} \leq e^{\frac{-2\varepsilon^2}{\sum_{i=1}^{n} {d_i}^2}} \quad (4.7)$$

This inequality pertains to a weighted total that is adjusted as each new sample is received. When dealing with streaming data, it is not possible to receive all samples at once for processing. Instead, they arrive gradually over time, making it necessary to use a weighted moving average value $\overline{X_n} = \sum_{i=1}^{n} V_i X_i$ where V_i are weighted values that lie in the range of [0, 1] and $X_i \in [a, b]$.

Step 2: A similar approach to the A-test. We compute the average weighted difference by considering the inequality as

$$Pr\{\overline{X_n} - \overline{Y_m} - (E[\overline{X_n}] - E[\overline{Y_m}]) \geq \varepsilon\} \leq e^{\frac{-2\varepsilon^2}{D_{n,m}}} \quad (4.8)$$

where $D_{n,m}$ represents the disparity between the weights of the moving average $D_{n,m} = (b-a)^2 \left[\sum_{i=1}^{n}(V_i)^2 + \sum_{i=1}^{m}(V'_i)^2\right]$. We can estimate the error bound ε_α which can be easily computed for a significance value α as $\varepsilon_\alpha = (b-a)\sqrt{\frac{D_{n,m}}{2}\ln\left(\frac{1}{\alpha}\right)}$. The W-test is the extension of the A-test. A hypothesis test is used to determine the bounds of the probability, considering both Type I and Type II errors. This is known as the weighted average (W-test).

Step 3: Weighted Average Test (W-test). The hypothesis test determines the acceptance and rejection of the maximum estimated error (ε_α) which is based on two main conditions.

A. Bound weighted average checking for type-I error should be satisfied if $E[\overline{X_n}] \leq E[\overline{Y_m}]$, then the inequality will be $Pr\{\overline{X_n} - \overline{Y_m} \geq \widehat{\varepsilon_\alpha}\} \leq \alpha$.

B. Bound weighted average checking for type-II errors should be satisfied if $E[\overline{X_n}] \geq E[\overline{Y_m}] + \epsilon$ and $\epsilon > \varepsilon_\alpha$, then inequality will be $Pr\{\overline{X_n} - \overline{Y_m} < \widehat{\varepsilon_\alpha}\} \leq e^{\frac{-2(\epsilon-\varepsilon_\alpha)^2}{D_{n,m}}}$.

4.1.2 Proposed Ensemble Framework

1. Motivation

A drift-adaptive IoT anomaly detection framework called PWPAE was developed by Yang et al. [16]. It combines two advanced online ensemble models—Adaptive Random Forest (ARF) and Streaming Random Patches (SRP)—with drift detection techniques, namely DDM and ADWIN. The framework achieves robustness by integrating the weighted outputs of four base learners: ARF-DDM, ARF-ADWIN,

SRP-DDM, and SRP-ADWIN. In this approach, the weights of the four base learners are assigned dynamically, as shown below.

Let the data stream be denoted by $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where x is the input data, and y is the associated class label given by $y \in \{1, 2, \ldots, c\}$. The final target class is evaluated as

$$\hat{y} = \underset{k \in \{1,2,\ldots,c\}}{argmax} \frac{\sum_{i=1}^{b} W_i p_i (y=k| L_i,x)}{b} \quad (4.9)$$

where L_i is a base learner, and b is the number of base learners; in this case, b=4. $p_i\ (y = k|\ L_i, x)$ is the prediction probability of class k for data sample x from the base learner; w_i is the weight of the base learner L_i that is computed from the real-time error rate as

$$\mathrm{w_i} = \frac{1}{\mathrm{Error\ rate}+} \quad (4.10)$$

where ε is a small constant value. There are some limitations of the PWPAE framework.

1. In this drift adaptation framework, the Drift detection method (DDM) can detect sudden types of drift efficiently, but in the case of gradual drift, it shows a slower response time for the larger size IoT data streams, which causes issues related to the memory overflow and degrades the framework's performance.
2. In this framework, the Adaptive WINdowing (ADWIN) method detects the gradual types of drift efficiently, but it suffers from continuous changes in the mean value, so exact performance evaluation becomes difficult.
3. PWPAE uses ARF and SRP, which are state-of-the-art online ensemble models for creating the base learners, but both approaches may show limited drift reaction speed and prediction accuracy, which decreases the overall performance of the model.

2. Proposed Error-Weighted Drift-Adaptive Ensemble Framework

In this paper, we propose a drift-adaptive online ensemble-based framework for streaming data, known as the Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF), which combines five base learners (ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, and ARF-HDDM_W). To construct the base learners, the online ensemble model ARF is combined with five popular drift detection techniques (EDDM, DDM, ADWIN, HDDM_A, and HDDM_W). The AdaBoost, Bagging, and PWPAE ensembles have been used to compare the performance of the

proposed EW-DA-EF. Accuracy, precision, recall, F1-score, and processing time are the performance metrics. Both real-world and synthetic drifting datasets are used for experimentation.

Generally, in traditional ensemble learning, the weights assigned to base learners remain stable throughout the entire process. However, in EW-DA-EF, the base learners are assigned a dynamic weight based on their real-time prediction error, which varies until the last sample is processed. Mathematically, the weight is inversely proportional to the real-time error rate of the base learners, i.e.

$$W_i \propto \frac{1}{E_i+\varepsilon} \tag{4.11}$$

where E_i and W_i, with $i = 1, 2, \ldots, b$, are the error rate and weight of the i^{th} base learner, respectively; ε is a small constant value added to prevent the denominator from being zero. This model operates on the principle of an accuracy-updated ensemble, which dynamically updates the weights; the higher the weight, the better the base learners are in terms of drift adaptability.

In the proposed ensemble framework, EW-DA-EF, the total error E_{total} is defined as the sum of the inverse of error rates of all base learners, as shown in Eqn. (4.12). The weights of the individual base learners W_i , $i = 1, 2, \ldots, b$, are defined as the ratio of the inverse of the real-time error rate of the i^{th} base learner to the total error rate E_{total}, as shown in Eqn. (4.13). Here, the number of base learners is b=5.

As observed in Eqn. (4.12) and Eqn. (4.13), if the rate of error is higher, since the weight value is inversely proportional to the error rate, the updated weight will be lower. This ensures a lower contribution of the base learner in the ensemble output. As compared to the PWPAE equations in Eqn. (4.9) and Eqn. (4.10), we consider the relative or fractional error rate (due to the division by E_{total}) instead of the absolute error rate in PWPAE.

$$E_{total} = \sum_{i=1}^{b} \frac{1}{E_i+\varepsilon} \tag{4.12}$$

$$W_i = \frac{1}{\left[{(E_i+\varepsilon)}/{E_{total}}\right]} \tag{4.13}$$

As compared to block-based ensembles, the updated ensemble method presented in our work is more suitable for adapting quickly to drifting data streams. The efficiency of the suggested framework is determined by the time it takes for the base learners (ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, and ARF-HDDM_W) to compute results, whereas the time complexity of EW-DA-EF can be expressed as O(nbk), with n being the total number of samples, b representing the

number of base learners, and k denoting the total weight computed. The time complexity of each base learner can vary, and this directly impacts the overall time complexity of the proposed framework. The detailed block diagram of the EW-DA-EF framework is shown in Fig. 4.1.

Advancement in the proposed EW-DA-EF

1. EW-DA-EF uses the online ensemble -Adaptive Random Forest (ARF) in combination with five popular drift detection techniques (EDDM, DDM, ADWIN, HDDM_A, and HDDM_W), to obtain the five base learners (ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, and ARF-HDDM_W).
2. This proposed framework is expected to deal with all types of drift (abrupt, gradual, recurring, incremental, etc.) efficiently due to the variety of drift detection techniques used in our ensemble.
3. In this framework, the weights of all five base learners are computed in real-time and dynamically updated over time.

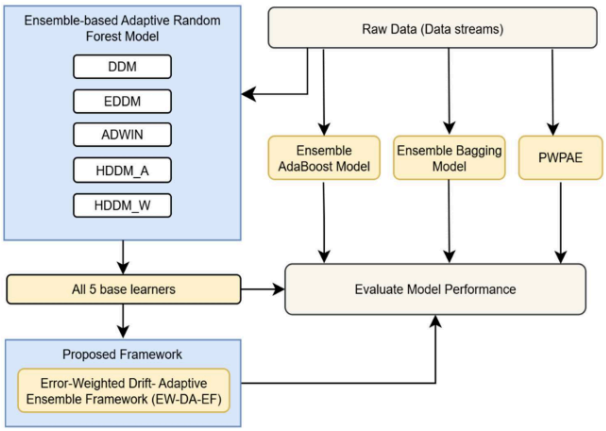


Fig. 4.1. Process flow of the experimentation.

4.1.3 Results and Discussions

1. Descriptions of Real-World and Synthetic Datasets

In this research, the performance of all drift detection techniques is evaluated using two real-world drifting datasets (weather and LUdata) and two synthetic drifting datasets (sea_stream and sea_big). The descriptions of these datasets are presented in Chapter 1, Section 1.5.2.

2. Results

We utilized online machine learning concepts to implement all the ensemble models. The recently introduced online machine learning package River and the traditional machine learning package Scikit-learn have been used to evaluate the performance of different online models in terms of accuracy, precision, recall, F1-score, and processing time for two real-world drifting datasets, weather and LUdata, and two synthetic drifting datasets: sea_stream and sea_big. The AdaBoost, Bagging, and PWPAE ensemble-based models have been used to compare the performance of the proposed EW-DA-EF.

The results are compiled in Table 4.1. In the implementation part, the same experimental setup was used as described in an earlier chapter. The code used for our implementation can be accessed from the GitHub repository[6]. If drift is detected during model training, the model's performance continues to decrease. To overcome this situation, the proposed EW-DA-EF utilizes concepts from drift detection techniques, enabling the online ensemble to rapidly adapt to all changes, thereby boosting its performance. The traditional machine learning package Scikit-learn was used to train and test the model. Specifically, 10% of the data was used for training, and 90% for testing. Training is conducted in a dynamic data environment; the concept of streaming data involves training models sequentially from feature/label pairs using the learn_one method. Predictions are then made using predict_one, and the average predictions of

[6] https://github.com/SantoshKumarRay/Error-Weighted-Drift-Adaptive-Ensemble-Framework-for-Classification-of-Streaming-Data

each classifier are calculated as a probability distribution using predict_prob_one for every sample.

In the proposed ensemble, all models are set to their default hyperparameters as suggested by the River and Scikit-learn online machine learning libraries. The hyperparameters for each base model are specified as follows. For ADWIN, the significance value is 0.002, the minimum length of the sliding window is 5, and the maximum number of buckets used to store statistics for windows is 5. The number of trees for ARF is set to 3. For EDDM, the warning level is 0.95, the drift level is 0.90, and the minimum number of instances before EDDM starts detecting drifts is 30. For DDM, the warning level is set at 2.0, the drift level at 3.0, and the minimum number of instances before DDM starts detecting drifts is 30. HDDM_A and HDDM_W share similar hyperparameters; the warning confidence is 0.005, and the drift confidence is 0.001. Our proposed EW-DA-EF consistently delivers good performance across the four drifting datasets, as evident from the scores in Table 4.1 and the visualization graphs in Figs. 4.2, 4.3, 4.4, and 4.5.

Table 4.1. Performance measures and processing time of ensemble-based classification models using two real-world drifting datasets (weather and LUdata) and two synthetic drifting datasets (sea_stream and sea_big).

weather dataset						
S. No.	**Methods**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Processing Time**
1	Proposed Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF)	0.7823	0.7298	0.4906	0.5868	1 min 10 s
2	ARF-EDDM	0.7204	0.5769	0.4220	0.4874	9.79 s
3	ARF-DDM	0.7639	0.6540	0.5319	0.5867	16.4 s
4	ARF-ADWIN	0.7543	0.6348	0.5180	0.5705	11.7 s
5	ARF-HDDM_A	0.7601	0.6513	0.5133	0.5741	13.6 s
6	ARF-HDDM_W	0.7305	0.5957	0.4496	0.5125	11.7 s
7	PWPAE [16]	0.7584	0.6440	0.4871	0.5547	8.81 s
8	AdaBoost [7]	0.7545	0.6266	0.5461	0.5836	9.03 s
9	Bagging [8]	0.7638	0.6679	0.4976	0.5703	3.32 s
LUdata dataset						
S. No.	**Methods**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Processing**

						Time
1	Proposed Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF)	1.0000	1.0000	1.0000	1.0000	8.11 s
2	ARF-EDDM	0.9953	0.9988	0.9916	0.9952	1.89 s
3	ARF-DDM	0.9971	0.9940	1.0000	0.9970	1.83 s
4	ARF-ADWIN	0.9860	0.9845	0.9868	0.9856	1.62 s
5	ARF-HDDM_A	0.9696	0.9827	0.9545	0.9684	2.14 s
6	ARF-HDDM_W	0.9848	0.9844	0.9844	0.9844	2.12 s
7	PWPAE [16]	1.0000	1.0000	1.0000	1.0000	1.77 s
8	AdaBoost [7]	0.9994	1.0000	0.9988	0.9994	2.22 s
9	Bagging [8]	0.9568	0.9646	0.9461	0.9553	1.09 s
sea_stream dataset						
S. No.	**Methods**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Processing Time**
1	Proposed Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF)	0.8839	0.8824	0.9258	0.9036	12 s
2	ARF-EDDM	0.8614	0.8617	0.9102	0.8853	2.39 s
3	ARF-DDM	0.8631	0.8808	0.8870	0.8839	3.18 s
4	ARF-ADWIN	0.8556	0.8568	0.9054	0.8805	2.21 s
5	ARF-HDDM_A	0.8672	0.8705	0.9092	0.8895	2.22 s
6	ARF-HDDM_W	0.8475	0.8575	0.8879	0.8725	2.17 s
7	PWPAE [16]	0.8819	0.8834	0.9206	0.9016	13.9 s
8	AdaBoost [7]	0.8689	0.8679	0.9177	0.8916	1.34 s
9	Bagging [8]	0.8700	0.8617	0.9277	0.8934	661 ms
sea_big dataset						
S. No.	Methods	**Accuracy**	**Precision**	**Recall**	**F1-score**	**Processing Time**
1	Proposed Error-Weighted Drift-Adaptive Ensemble Framework (EW-DA-EF)	0.8943	0.8936	0.9268	0.9099	7 min 21 s
2	ARF-EDDM	0.8865	0.8880	0.9190	0.9032	1 min 33 s
3	ARF-DDM	0.8870	0.8879	0.9201	0.9037	1 min 37 s
4	ARF-ADWIN	0.8839	0.8849	0.9179	0.9011	1 min 1 s
5	ARF-HDDM_A	0.8842	0.8850	0.9183	0.9014	1 min 28 s
6	ARF-HDDM_W	0.8866	0.8878	0.9194	0.9033	1 min 37 s
7	PWPAE [16]	0.8788	0.8765	0.9204	0.8979	36.6 s
8	AdaBoost [7]	0.8816	0.8804	0.9193	0.8994	31.9 s
9	Bagging [8]	0.8836	0.8831	0.9197	0.9011	13 s

3. Discussions

Firstly, we analyzed the performance scores and processing time of the weather real-world drifting dataset and stored the results in Table 4.1. ARF-ADWIN, AdaBoost, and PWPAE show very close accuracies of 0.7543, 0.7545, and 0.7584, respectively, while ARF-HDDM_A, Bagging, and ARF-DDM show slightly better performance than the above three approaches with scores of 0.7601, 0.7638, and 0.7639, respectively, but ARF-EDDM and ARF-HDDM_W are poor performers than the others with scores of 0.7204 and 0.7305, respectively. Our proposed ensemble (EW-DA-EF) achieves the highest accuracy of 0.7823 on the weather dataset, as shown in Table 4.1. The precision, recall, and F1-score values show a similar trend. The processing time is higher, though, due to the ensemble learning process. As observed from Fig. 4.2, EW-DA-EF exhibits higher accuracy over time, indicating good drift adaptability, compared to the five base learners and the state-of-the-art ensembles comprising AdaBoost, Bagging, and PWPAE. Similarly, for another dataset (LUdata), the proposed EW-DA-EF adapts efficiently to the drift in the LUdata, as observed from the accuracy graph in Fig. 4.3. Similar to the performance scores of the weather dataset, the proposed EW-DA-EF achieves the highest accuracy of 1.0000 for LUdata, a feat paralleled by the PWPAE and AdaBoost ensembles. Processing times vary for all models and depend on the system's configuration. In the case of synthetic datasets (sea_stream and sea_big), the performances of all other classifiers are very close; however, our proposed framework, EW-DA-EF, shows the highest accuracy of 0.8839 for the sea_stream dataset and 0.8943 for the sea_big dataset. The size of the synthetic dataset is very large. Hence, the processing time is longer, and it takes a longer time to compute the ensemble outputs, especially for EW-DA-EF and PWPAE. The performance scores are summarized in Table 4.1 for the two synthetic datasets. The performance visualization over time is shown in Fig. 4.4 for sea_stream and in Fig. 4.5 for sea_big.

Our proposed EW-DA-EF consistently delivers good performance across the four drifting datasets, as evident from the scores in Table 4.1 and the visualization graphs in Figs. 4.2, 4.3, 4.4, and 4.5. Our method outperforms the state of the art in online learning for real-world drifting datasets. The reason for the quick adaptability to drift is the dynamic weight updation rule given in Eq. (4.12) and Eq. (4.13), which is used for adjusting the weights of the base learners in real-time based on their prediction performance. Our method also outperforms the related research, PWPAE, due to the substitution of the absolute error rate by the relative or fractional error rate for the weight updation of the base learners. The dynamic weight updates for the five base learners can be visualized in Figs. 4.6, 4.7, 4.8, and 4.9 for the four datasets, respectively. EDDM is effective at identifying gradual drift, but it operates at a slower

pace and may not be suitable for processing data with high arrival rates. On the other hand, DDM is well-suited for detecting sudden and relatively rapid changes, but struggles with detecting slow, gradual changes. ADWIN is versatile and capable of handling both abrupt drifts in the case of infrequent changes and slow, gradual changes. When it comes to specifically detecting abrupt changes, HDDM_A is more appropriate, while HDDM_W is better equipped to manage gradual changes.

Table 4.1 presents the results on the weather dataset, where EDDM, ADWIN, and HDDM_W exhibit decreasing performance scores, resulting in lower scores compared to other techniques. Conversely, DDM and HDDM_A perform well in handling the sporadic abrupt drift type found in the weather dataset. In these cases, abrupt drift does not negatively impact performance scores, leading to superior performance. Our proposed framework excels in managing both abrupt and gradual drift types, yielding higher performance scores when compared to other ensembles such as AdaBoost, Bagging, and PWPAE.

The LUdata dataset comprises social survey information, with individuals serving as data points. The objective is to anticipate whether these individuals are frequent internet users, primarily for marketing applications. This dataset is noteworthy due to its gradual drift [117]. LUdata, in particular, displays a minimal performance variance among various methods, making it challenging to identify drift due to its gradual nature. When dealing with gradual drift, the proposed ensemble exhibits significantly superior performance scores as compared to others. EDDM and HDDM_W exhibit improved performance in the case of the LUdata dataset, as they are more receptive to gradual drifts.

Table 4.1 presents two synthetic datasets, namely sea_stream and sea_big, designed for data streams containing abrupt and gradual drifts, respectively. EDDM, ADWIN, and HDDM_W detect gradual drift effectively for the sea_big dataset. DDM and HDDM_A, on the other hand, are effective at identifying infrequent abrupt drifts in the sea_stream dataset, though DDM has performed consistently well for all datasets. The proposed method effectively adapts to changes in the data stream, resulting in a high classification score in the presence of both gradual and abrupt types of drift.

The primary advantage of the proposed framework lies in its ability to integrate both abrupt and gradual drift detection methods concurrently within the classification process, eliminating the requirement for additional strategies to accommodate evolving data distributions in streaming environments. This model addresses the need for retraining using the accumulated dataset, ensuring the model remains continuously updated. The ensemble demonstrates stable performance across various datasets, including real-time, synthetic, random, and live source data streams. The disadvantage of the proposed ensemble framework is its higher computational

time complexity, which can be mitigated by optimizing the base learners or utilizing high-performance distributed processing systems that facilitate parallel processing.

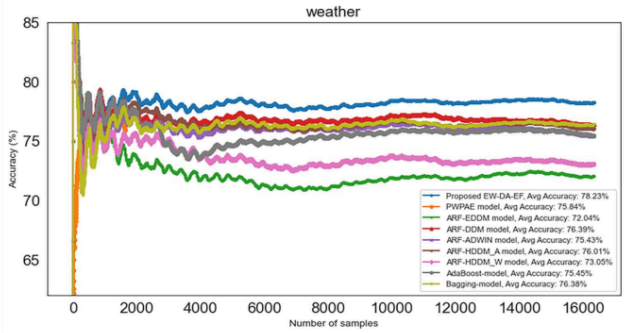


Fig. 4.2. Performance accuracy plot of all ensemble-based classification models for weather real-world drifting dataset.

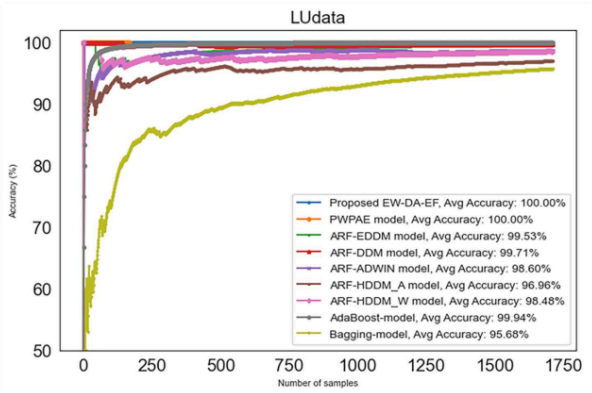


Fig. 4.3. Performance accuracy plot of all ensemble-based classification models for the LUdata real-world drifting dataset.

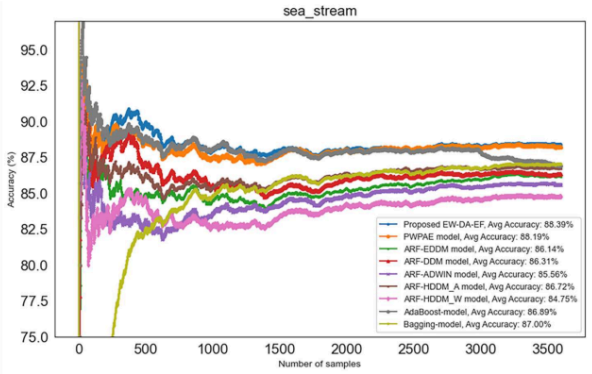


Fig. 4.4. Performance accuracy plot of all ensemble-based classification models for sea_stream synthetic drifting dataset.

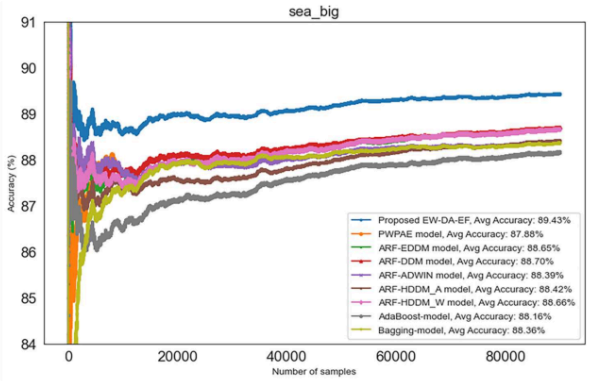


Fig. 4.5. Performance accuracy plot of all ensemble-based classification models for sea_big synthetic drifting dataset.

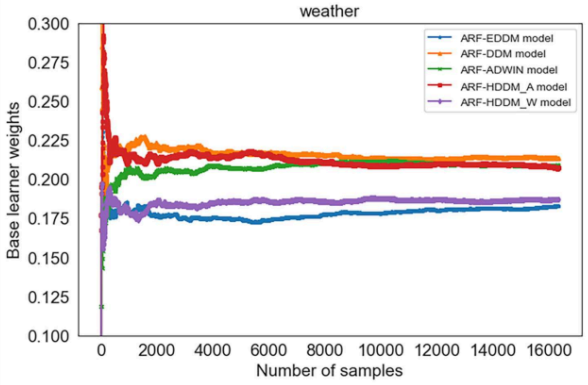


Fig. 4.6. Dynamic weight updation for the five base learners for the weather real-world dataset.

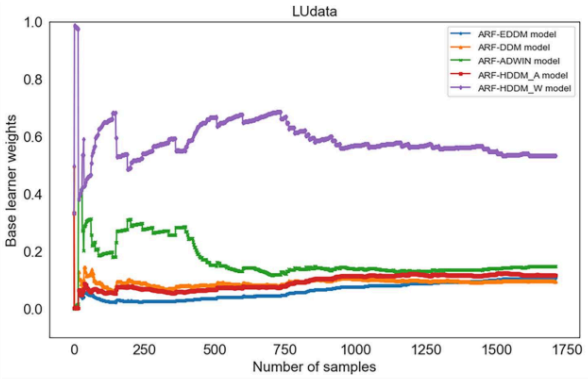


Fig. 4.7. Dynamic weight updation for the five base learners for the LUdata real-world dataset.

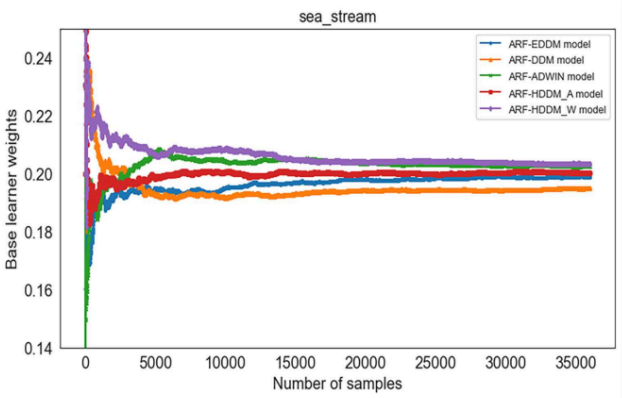


Fig. 4.8. Dynamic weight updation for the five base learners for the sea_stream synthetic dataset.

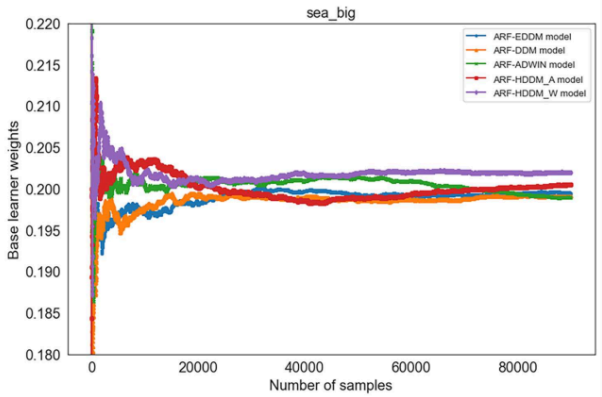


Fig. 4.9. Dynamic weight updation for the five base learners for the sea_big synthetic dataset.

4.2 OBJECTIVE-1 (b): Performance evaluation using online machine learning packages for streaming data.

Citation: Ray, Santosh Kumar, and Seba Susan. "Performance evaluation using online machine learning packages for streaming data." *In 2022 international conference on computer communication and informatics (ICCCI), pp. 1-6. IEEE, 2022*. **DOI:** 10.1109/ICCCI54379.2022.9741068. **IEEE, (Published).**

4.2.1 Process Flow Framework

In this chapter, the main challenge in the online machine learning approach is to find a model that yields good accuracy for the given streaming dataset. Patterns of the stream are unknown beforehand, which can affect the efficiency and accuracy of the system.

We proposed a framework, where the input data is processed and transferred to a data stream that is the concatenation of *n* samples $D_1, D_2, D_3, \ldots, D_n$.

$$D = \text{Concatenate}(D_1, D_2, D_3, \ldots\ldots, D_n) \quad (4.14)$$

The streaming data is processed by three online classifier models - Logistic Regression (LR), K-Nearest Neighbor (KNN), and Gaussian Naive Bayes (GNB). For complete online machine learning functionality and its evaluation, the Creme and River machine learning packages are used. The workflow is illustrated in Fig. 4.10.

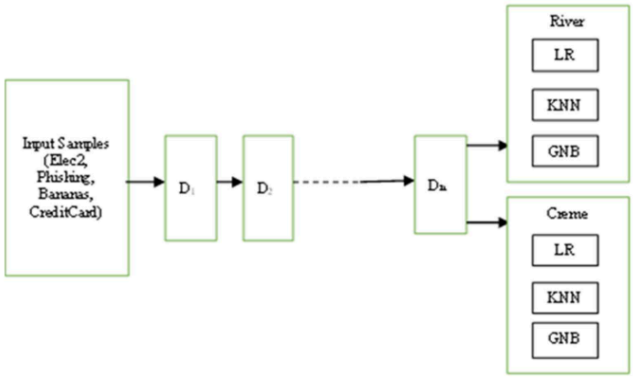


Fig. 4.10. Process flow framework for the performance analysis using River and Creme online machine learning packages.

4.2.2 Experimental Setup and Results

Online machine learning packages Creme and River are used to perform the evaluations. The following classifiers are used to perform online or incremental learning: Logistic Regression (LR), K-Nearest Neighbors (KNN), and Gaussian Naive Bayes (GNB) classifiers. The experiments are performed using Python software version 3.8 on a 1.8 GHz Intel Core PC. The code took a few seconds to execute. In this research, real-time streaming datasets are used for the experiments, namely, Elec2, Phishing, Bananas, and CreditCard, the details of which are given below. The accuracy, ROC-AUC, precision, recall, and F1-score performance metrics are used to evaluate the predictions made by the online classifiers. The results are shown in Table 4.2.

1. Description of Datasets

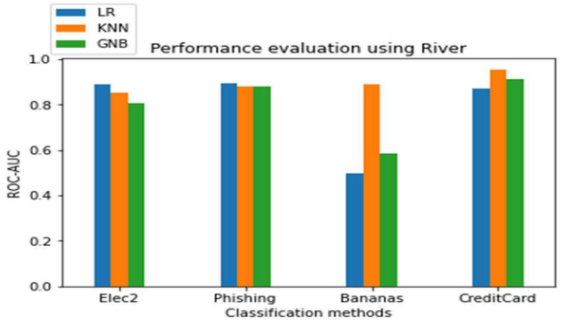
In this research, four benchmark real-time streaming datasets, Elec2, Phishing, Bananas, and CreditCard, are used for the experiments. Details of these datasets are provided in Chapter 1, Section 1.5.1.

Table 4.2. Performance of the four streaming datasets evaluated using River and Creme online machine learning packages (shown for LR, KNN, and GNB classifiers).

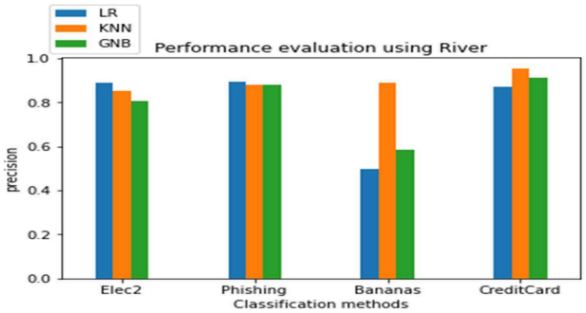
S. No.	Dataset	Metrics	River			Creme		
			LR	KNN	GNB	LR	KNN	GNB
1	Elec2	accuracy	89.5%	85.8%	76.14%	83.73%	87.08%	76.13%
		ROC-AUC	0.8909	0.8519	0.8052	0.8275	0.8710	0.7357
		precision	0.885	0.845	0.8155	0.8398	0.8580	0.8155
		recall	0.865	0.814	0.5659	0.7623	0.8330	0.5658
		F1-score	0.875	0.829	0.6681	0.7992	0.8460	0.6681
2	Phishing	accuracy	89.0%	88.01%	88.0%	89.20%	75.92%	87.92%
		ROC-AUC	0.8919	0.8809	0.8814	0.8922	0.7590	0.8814
		precision	0.849	0.850	0.842	0.8642	0.7400	0.8417
		recall	0.911	0.883	0.8940	0.8942	0.6950	0.8923
		F1-score	0.879	0.866	0.867	0.8789	0.7170	0.8663
3	Bananas	accuracy	51.9%	88.5%	61.2%	53.74%	51.32%	61.15%
		ROC-AUC	0.4966	0.8905	0.5830	0.5007	0.5130	0.5830
		precision	0.443	0.877	0.640	0.4508	0.4580	0.6397
		recall	0.279	0.865	0.306	0.1465	0.4700	0.3064
		F1-score	0.342	0.871	0.414	0.2211	0.4640	0.4143
4	CreditCard	accuracy	99.9%	99.9%	98.1%	99.94%	99.86%	98.14%
		ROC-AUC	0.8707	0.9551	0.9129	0.8607	0.9990	0.9095
		precision	0.783	0.959	0.073	0.8853	0.9690	0.0732
		recall	0.742	0.427	0.837	0.7215	0.1890	0.8374
		F1-score	0.762	0.591	0.134	0.7951	0.3160	0.1346

2. Performance visualizations using River

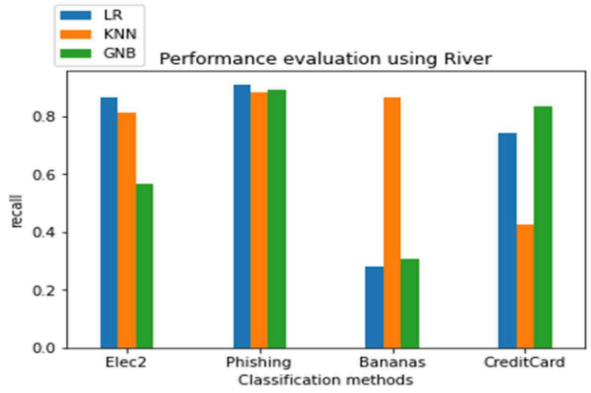
Performance evaluation of online machine learning models tests the consistency of models. In Fig. 4.11 (a), Fig. 4.11 (b), Fig. 4.11 (c), and Fig. 4.11 (d), the bar plots visualize the ROC-AUC, precision, recall, and F1-score performance metrics using the River online machine learning package. The metrics are computed for four streaming datasets (Elec2, Phishing, Bananas, CreditCard). For each dataset, the results of logistic regression (LR), K-nearest neighbor (KNN), and Gaussian Naive Bayes (GNB) online classifiers are shown. For the Elec2 dataset, LR and KNN show better ROC-AUC value while the GNB classifier yields a low value. For the other metrics, such as precision, recall, and F1-score, the performance of the KNN classifier is better, while LR shows average performance, and the overall performance of GNB is better in the case of the Elec2 dataset. In the case of the Phishing dataset, the performances of all three classifiers are similar with respect to all four-performance metrics. In the Bananas dataset, the performance of the KNN classifier is significantly better for all four metrics, but the other classifiers show very low performance. CreditCard is a large-sized, imbalanced streaming dataset. The LR classifier performs best for this dataset. Performance scores using the River online machine learning package are summarized in Table 4.2 for all datasets.


LR
KNN
GNB
Performance evaluation using River
ROC-AUC
1.0
0.8
0.6
0.4
0.2
0.0
Elec2
Phishing
Bananas
CreditCard
Classification methods

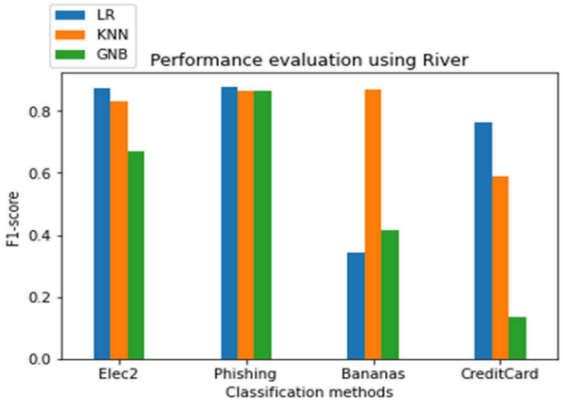
(a)


LR
KNN
GNB
Performance evaluation using River
precision
1.0
0.8
0.6
0.4
0.2
0.0
Elec2
Phishing
Bananas
CreditCard
Classification methods

(b)



(c)

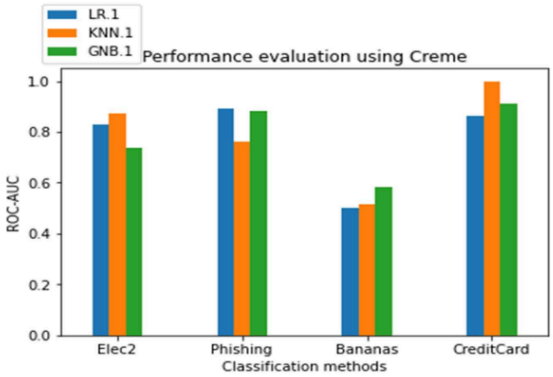


(d)

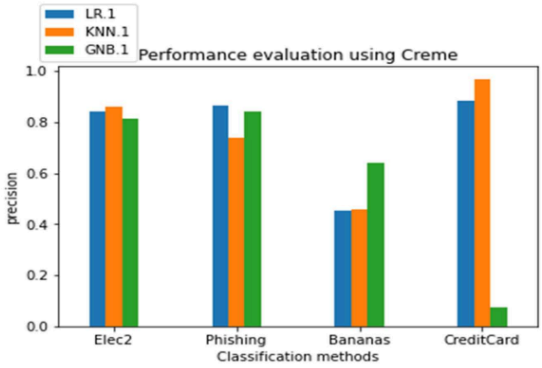
Fig. 4.11. Bar plots of classifiers (LR, KNN, GNB) for the four datasets (Elec2, Phishing, Bananas, CreditCard) using River for the performance metrics **(a)** ROC-AUC, **(b)** precision, **(c)** recall, and **(d)** F1-score.

3. Performance visualizations using Creme

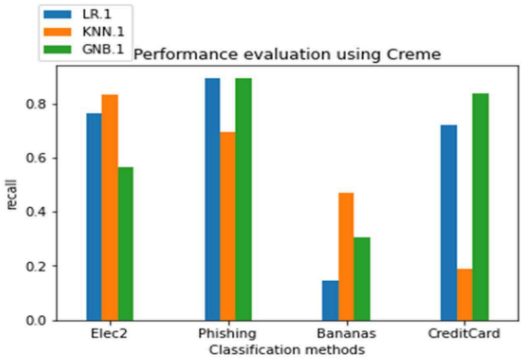
Similar experimental conditions are applied for the Creme online machine learning package, for the same datasets, and the same online classifiers. The ROC-AUC, F1-score, precision, and recall performance metrics are plotted in Fig. 4.12 (a), Fig. 4.12 (b), Fig. 4.12 (c), and Fig. 4.12 (d), respectively, and summarized in Table I. The average performance scores of Creme are lower in comparison to River. For the Creme online machine learning package, the performance of the KNN classifier is better, but the overall performance of the LR and GNB classifiers is not as high as that of the River package. The performance of the Bananas dataset, however, is low even with the KNN classifier. Online machine learning shows an average performance for small-sized streaming datasets, and when the size of the dataset is large, it shows better performance. Creme takes more processing time than River. Creme supports the mini-batch concept, but the outcome is not more efficient. It can extend the functionality of machine learning techniques at an industrial scale, but it is time-consuming and requires more memory.



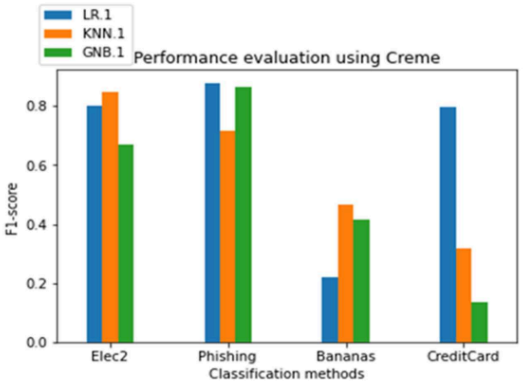
(a)


LR.1
KNN.1
GNB.1
Performance evaluation using Creme
precision
1.0
0.8
0.6
0.4
0.2
0.0
Elec2
Phishing
Bananas
CreditCard
Classification methods

(b)


LR.1
KNN.1
GNB.1
Performance evaluation using Creme
recall
0.8
0.6
0.4
0.2
0.0
Elec2
Phishing
Bananas
CreditCard
Classification methods

(c)



(d)

Fig. 4.12. Bar plots of classifiers (LR, KNN, GNB) for the four datasets (Elec2, Phishing, Bananas, CreditCard) using Creme for the performance metrics **(a)** ROC-AUC, **(b)** precision, **(c)** recall, and **(d)** F1-score.

4. Comparison of model accuracies using River and Creme

Accuracy is a key parameter for deciding the efficacy of a model. In Table 4.2, the accuracies for all three classifiers using the River and Creme online machine learning packages are calculated for all four streaming datasets. The results are visualized in the form of a bar plot shown in Fig. 4.13. A stable performance is observed in the bar plot in the case of Elec2, Phishing, and CreditCard streaming datasets for both River and Creme; however, the accuracy is low for the Bananas dataset in both cases. The accuracy obtained using River is higher than that of Creme. It is specially noted that for the highly imbalanced CreditCard dataset, accuracy is higher than all other datasets, unlike the other performance metrics shown in Fig. 4.11 and Fig. 4.12. This would imply that accuracy alone may not be a reliable metric for an imbalanced dataset.

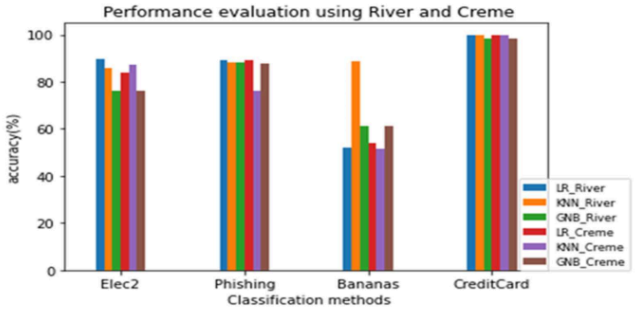


Fig. 4.13. Bar plot of classifiers (LR, KNN, GNB) accuracy (%) for the four datasets (Elec2, Phishing, Bananas, CreditCard) using River and Creme.

4.3 Conclusion

This chapter focused on enhancing the performance of online learning in streaming environments through drift-aware ensemble methods and comparative analysis of streaming frameworks. The proposed EW-DA-EF model demonstrated superior performance on both real-world and synthetic drifting datasets, achieving higher accuracy, precision, recall, and F1-score than existing ensemble approaches. Its ability to incorporate multiple drift-detection techniques enabled effective adaptation to both gradual and abrupt concept drift, resulting in more stable and reliable predictions.

In addition, a comparative evaluation of online machine learning frameworks highlighted the effectiveness of River over Creme across multiple datasets and classifiers, with KNN showing consistent performance. The study also emphasized the importance of using multiple evaluation metrics, especially in imbalanced data scenarios where accuracy alone may be misleading.

Despite these improvements, challenges such as increased computational cost, latency, memory constraints, and handling high-throughput streaming data remain critical. Future work will focus on improving computational efficiency through optimized learning strategies, parallel and distributed processing, and developing more robust techniques for real-time drift handling and large-scale dynamic data streams.

CHAPTER 5

Design of novel anomaly detection technique for evolving data streams

5.1 OBJECTIVE-2 (a): Multi-Framework Stacked Ensemble for Quantifying Outlierness in Data Streams.

Santosh Kumar Ray and Seba Susan's research paper title "Multi-Framework Stacked Ensemble for Quantifying Outlierness in Data Streams" is currently under review at ***The Journal of Supercomputing (Springer),*** **(SCIE-indexed, IF: 2.7), (Communicated).**

5.1.1 Proposed Methodology

Motivated by the success of heterogeneous ensembles for online/real-time anomaly detection, we propose a hybrid approach that advocates for unsupervised learning in the initial phase, followed by supervised meta-learning. The heterogeneous models are selected from two different Python frameworks for streaming data, namely River and PySAD, which offer distinct machine learning functionalities. To the best of our knowledge, this is the first time the two frameworks have been combined for the same streaming application. The proposed multi-framework stacked ensemble for detecting anomalies in data streams operates in two tiers. The initial level consists of the ten base learners; the second level comprises the meta-learner stage. The proposed stacking-based ensemble framework will be explained in detail in a separate section. An introduction to the River and PySAD machine learning frameworks is also given. The detailed block diagram of the proposed stacking framework is shown in Fig. 5.1. The description of the components in the unified framework, as illustrated, is provided in the subsections below.

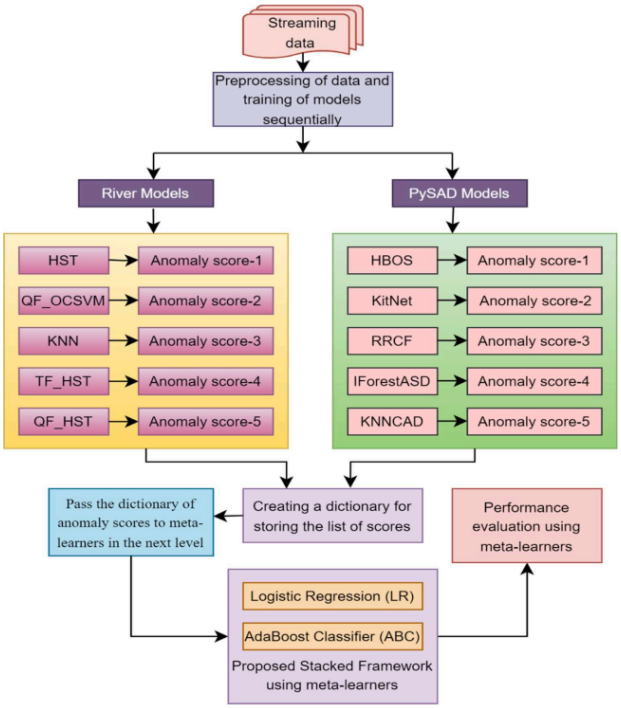


Fig. 5.2. Block diagram of the proposed multi-framework stacked ensemble for anomaly detection from data streams.

1. Contributions of base learners

In our approach, we utilize five River models and five PySAD models to identify anomalies in data streams, which serve as the base learners at the first level. These techniques include HST, Quantile Filter One Class Support Vector Machine (QF_OCSVM), TF_HST, QF_HST, KNN, HBOS, KitNet, RRCF, IForestASD, and KNNCAD. Each base learner continuously takes streaming data, trains the model online, and outputs an outlierness score, also known as the anomaly score, which is stored in a dictionary with keys and passed as input to the second level, the meta-learning stage. To identify anomalies, the default threshold of the logistic regression classifier, which serves as the meta-learner in our case, is set to 0.5; a score above this threshold indicates an anomaly. A higher outlierness score indicates a higher degree of anomaly, while a lower score suggests that the data is closer to the true value.

a) River models for anomaly detection

Montiel et al. (2021) [2] introduced a dynamic online machine learning library, River, designed to continuously learn from streaming data, one instance at a time. Each River-based anomaly detection model generates a pipeline consisting of a list of estimators applied sequentially. During online training, these models utilize the learn_one method to train or fit one instance at a time. Additionally, the model offers predictions of anomaly scores through the score_one function. We next describe the River models used in our experiments.

Half Space Trees (HST) [28]: Half-space trees are a variation of isolation forests that can be used online. They perform effectively in scenarios where anomalies are distributed randomly. However, they are not suitable for detecting anomalies clustered within a specific time window. During implementation, it is presumed that every feature has values ranging from 0 to 1. The HST algorithm calculates the anomalous outcomes for a given input by using the mass distribution of each partition in an HST. The distribution may be uniform or non-uniform, depending on different conditions. The data point x travels through a single tree T and returns an anomaly score. Finally, the anomaly score of x is calculated by summing the scores across all trees in the HST, as shown in Eq. (5.1).

$$Anomaly\ Score\ of\ HST = \sum_{T \in HST} Score(x, T) \quad (5.3)$$

Quantile Filter-based One-Class Support Vector Machine (QF_OCSVM): The One-class SVM method used for anomaly detection is a stochastic variant of the original batch formulation and may not be an exact representation of it. During the initial training phase, QF_OCSVM focuses on normal data, creating a boundary that encompasses these points in high-dimensional space. This involves setting a decision boundary to separate normal instances from the rest of the space. The One-Class SVM approach integrates a Quantile Filter to enhance and fine-tune the anomaly detection process.

Quantile Filter_HST (QF_HST): This anomaly detector utilizes quantile levels to determine whether an anomaly score is anomalous or normal. The pipeline begins by normalizing features using MinMaxScaler and subsequently employs Half Space Trees within a Quantile Filter to emphasize data points within a specific quantile. This configured pipeline is ready for both training on existing data and for predicting new data instances. It also decides whether to update the anomaly detector when the anomaly score is deemed anomalous. For sporadic anomalies, it is not advisable to update the anomaly detector. This is because if it learns the anomaly score, it may begin to recognize anomalous anomaly scores as normal over time.

Threshold Filter_HST (TF_HST): This anomaly detector utilizes a threshold to differentiate between an anomalous and a normal anomaly score. The anomaly scores generated by the Half-Space Trees are then passed through a Threshold Filter. This filter is configured with a specific threshold value. The configured pipeline is ready for both training on existing data and for predicting new data instances. It also specifies whether the anomaly detector should be updated when an anomaly score is detected. For infrequent anomalies, it is advisable not to update the anomaly detector, as it may eventually treat anomalous scores as normal, leading to inaccuracies.

K-Nearest Neighbor (KNN): This supervised learning approach works by creating a window of a specified size to make instant predictions, using the nearest neighbors from the observations to make the required predictions. This is essential for identifying anomalies because a normal instance occurs in a dense area, where the anomaly is far from its neighbors, and this process requires no assumptions about the data distribution model. During training, the KNN model is trained on a dataset that includes both normal and anomalous instances. An anomaly score is assigned to incoming instances by evaluating their distance to the k-nearest neighbors. Instances showing notable deviations from their neighbors are identified as anomalies.

b) PySAD models for anomaly detection

PySAD [86] is an open-source Python framework for detecting anomalies in streaming multivariate data. It offers tools for online or sequential anomaly detection, adapting to new instances as they arrive in streaming data. The framework comprises various components, including stream evaluators, preprocessors, postprocessors, probability calibrators, and others. We next describe the PySAD models used in our experiments.

Histogram-Based Outlier Score (HBOS): Goldstein et al. [29] developed an unsupervised method for identifying outliers in datasets that does not require prior training. This technique models univariate feature densities using histograms, employing either fixed or dynamically determined bin widths. Subsequently, these histograms are utilized to calculate an anomaly score for each data point. While

evaluation demonstrates that HBOS excels in detecting global anomalies, it may not effectively identify local outliers. The HBOS score for each instance p is then calculated from the height of the corresponding bin in the histogram as

$$HBOS(p) = \sum_{i=0}^{d} log\left(\frac{1}{his_{\ i}(p)}\right) \quad (5.2)$$

where i represents the count of dimensions, and d represents a dimension.

Kitsune's Core Algorithm (KitNet): Mirsky et al. [30] introduced a new technique called "Kitsune's core algorithms," an innovative artificial neural network (ANN)-based system for detecting network intrusions. This system operates autonomously, eliminating the need for human supervision, and is highly effective in identifying attacks on local networks. KitNet, their unsupervised ANN, is specifically designed for online anomaly detection and can distinguish between typical and anomalous network traffic patterns. It accomplishes this by effectively tracking the patterns across all network channels through a structure comprising two autoencoder layers: the ensemble layer and the output layer. The ensemble layer evaluates the individual abnormality of each subset within the ordered collection of k sub-instances, while the output layer calculates the final anomaly score by considering the interplay between subspace abnormalities and the inherent noise in the network traffic.

Robust Random Cut Forest (RRCF): Guha et al. [31] tackled anomaly detection in dynamic data streams using a technique rooted in random cut forests. They proposed an effective method that explores a resilient random cut data structure, serving as a condensed representation of the input stream. Importantly, they demonstrate that this structure roughly maintains the distances between data points. Preserving these distances is vital for various computations, extending beyond anomaly detection. Moreover, they employed a model-based definition of anomalies that accounts for the impact of adding or removing data points on the size of this condensed representation.

Isolation Forest Algorithm for Anomaly Detection (IForestASD): Ding et al. [32] introduced a framework for effectively identifying anomalies within streaming data. This method involves adapting an anomaly detection algorithm, IForestASD, which is tailored for streaming data and built upon the iForest algorithm. It is worth noting that the iForest algorithm is an unsupervised approach, meaning it does not rely on category attributes during the anomaly detection process; anomaly labels are solely utilized to evaluate the overall outcome of anomaly detection. The primary contributions of this study are threefold: first, the proposal of an anomaly detection framework tailored for streaming data; second, the adaptation of an algorithm originally designed for static datasets to suit streaming data; and third, the conduct of experiments to demonstrate the practicality and effectiveness of the proposed algorithm.

K-Nearest Neighbor Conformal Anomaly Detection (KNNCAD): Burnaev et al. [33] have introduced cutting-edge anomaly detection techniques tailored explicitly for one-dimensional time series data. These algorithms encompass a variety of methods, including extracting features, assessing notable deviations in new observations relative

to past data, and providing a probabilistic interpretation of these deviations within the framework of the conformal paradigm. Detecting anomalies within one-dimensional time-series data poses a significant challenge because it frequently does not adhere to the assumptions of conventional change-point models. The primary objective of this study is to propose a robust, non-parametric method for identifying anomalies in one-dimensional time-series data, while also providing a probabilistic understanding of anomaly scores.

2. Contributions of meta-learner

In our stacking-based ensemble framework, the meta-learning classifier, specifically Logistic Regression (LR) and AdaBoost Classifier (ABC), is employed at level two for merging the anomaly scores of all base learners, as shown in Fig. 5.1, and creating a new input for the meta-learner by stacking all River and PySAD anomaly detection techniques. Meta-learners learn the optimal way to explore and merge the base learner's outputs in an ensemble to generate more precise forecasts. The output of base learners can be used to train meta-learners utilizing online machine learning models. By combining the strengths of various base learners and learning from their prediction scores, the meta-learners can enhance the ensemble's performance, generate more accurate forecasts, and store their final prediction scores. The two meta-learners used in our experiments are described next.

Logistic Regression (LR): Logistic Regression is a linear classification method that is functionally related to one or more predicting values and a categorical value. It supports both types of responses, binomial or multinomial, which are either 1 or 0. This is the easiest method to identify anomalies because the anomalous conditions are binary (a normal instance is 0 and an anomalous instance is 1).

AdaBoost Classifier (ABC): Oza et al. [7] developed an online iteration of the algorithm that processes each training example as it arrives, eliminating the need to store and reprocess data, thereby maintaining a current hypothesis based on all previously seen instances. These algorithms offer benefits over conventional batch algorithms when dealing with continuously arriving data. With each new observation, the "learn_one" method of each model is invoked a variable number of times, k determined by sampling from a Poisson distribution with parameter *lambda*. The *lambda* parameter is updated as weak learners sequentially fit the same observation. AdaBoost is effective for anomaly detection in streaming data because it adapts to changing data patterns over time. By assigning more weight to misclassified data points, it focuses on detecting anomalies that might be challenging to identify with a fixed model.

3. Proposed ensemble framework for anomaly detection in data streams

The proposed stacking-based ensemble framework is composed of five robust anomaly detection River models (HST, QF_OCSVM, TF_HST, QF_HST, and KNN) and five anomaly detection PySAD models (HBOS, KitNet, RRCF, IForestASD, and KNNCAD) as base learners, and the logistic regression classifier and AdaBoost classifier as the meta-learners, respectively. Meta-learners operate by stacking and learning the predictions of the base learners. The proposed ensemble framework utilizes a predefined threshold of 0.5 for the meta-learner, which is employed to identify anomalies when the score exceeds this threshold. The ensemble's performance is evaluated using various metrics, including ROCAUC, Recall, and Geometric Mean [103].

In anomaly detection, the data is often imbalanced because the contribution of anomalies is very small relative to normal instances. Generally, imbalanced datasets refer to situations where one class has significantly fewer contributions compared to the other class in a two-class problem. This imbalance can lead to significant variations in the performance scores of standard supervised anomaly detection methods. It has been observed that the performance of anomaly detection classifiers in class-imbalanced applications should not be assessed solely by average accuracy, as this can be misleading. For example, in a domain where only 5% of examples are positive, labeling all new samples as negative would result in an accuracy of 95%, but would fail to detect any positive cases. To address this issue, key metrics such as ROCAUC, Geometric Mean, and Recall are used to measure and optimize the performance of both positive and negative classes while maintaining balance; these scores are explained below.

Geometric Mean (GM): The use of geometric mean score can effectively evaluate the performance of a classifier even when there is an imbalance in the distribution of examples between classes. This is because the geometric mean is not affected by the distribution. The geometric mean is calculated by combining the recall scores for each class.

$$Geometric\ Mean\ (GM) = \sqrt[n]{(g_1 \times g_2 \times g_3 \times \cdots \times g_n)} \tag{5.3}$$

where g_i is the sensitivity or recall score of the class label i and n is the count of classes.

Receiving Operating Characteristic-Area Under the Curve (ROCAUC): The metric presented is an estimate of the actual ROCAUC. To calculate the true ROC AUC, all the predictions and ground truths would need to be saved, which is not ideal. However, the approximation error is negligible if the predicted probabilities are

accurate. Moreover, this metric can still be employed to compare models with confidence. This also highlights the importance of determining the correct positive value to be considered. The ROCAUC scores are determined by using the values of True Positive Rate (TPR) and False Positive Rate (FPR).

Recall: In an anomaly detection system, recall measures the successfully detected anomaly when a true positive value is anomalous. The mathematical representation of recall is given by

$$Recall = \frac{True\ Positive}{[True\ Positive + False\ Negative]} \tag{5.4}$$

The methodology steps followed in our experiments are described below.

Step 1: To begin with, we stack five popular River based anomaly detection methods (HST, QF_OCSVM, TF_HST, QF_HST, and KNN) and five PySAD anomaly detection methods (HBOS, KitNet, RRCF, IForestASD, and KNNCAD) that operate by sequentially processing incoming data instances to train the model.

Step 2: The learning process occurs at two levels. The first level consists of the ten base learners (HST, QF_OCSVM, TF_HST, QF_HST, KNN, HBOS, KitNet, RRCF, IForestASD, and KNNCAD) that compute individual anomaly scores. These scores are submitted as input to the next level, where a meta-learner (a Logistic regression classifier and an AdaBoost classifier) is trained on all model scores to predict the final anomaly score. The proposed unified framework thus presents a novel approach for quantifying the outlierness of each data point in a sequential manner.

Step 3: To identify anomalies from the meta-learner's output, we use a threshold of 0.5 for the logistic regression and AdaBoost classifiers, which corresponds to a class posterior probability between 0 and 1. Scores lower than the threshold indicate a normal value, while higher scores indicate anomalous values. For each instance, the computed score is checked to determine if it exceeds the threshold or not.

Step 4: By combining the base learners with meta-learners in the same settings, we developed a final framework that outperformed all other anomaly detection techniques, as demonstrated by a comparison of classification performance using ROCAUC, Recall, and Geometric Mean metrics. The entire process is summarized in Algorithms 5.1 and 5.2 below. Fig. 5.2 further illustrates the process flow described in the two algorithms. Table 5.1 summarizes the abbreviations used in the overall study.

4. Proposed algorithm for stacked framework

The proposed framework employs two primary algorithms: Algorithm 5.1 for predicting the base learner's outcomes and Algorithm 5.2 for predicting the meta learner's outcome.

Algorithm 5.1: Estimating the base learner's output

Input: *Steaming data (da)*

Output: *Class posterior probability scores of base learners (b_i)*

Initializing Symbols:

da: streaming dataset

x: features, y: labels

z: total base learners

x0: dictionary

LR: Logistic Regression (meta-learner)

ABC: AdaBoost Classifier (meta-learner)

isAnomaly: List of anomalous and non-anomalous scores

$Z \leftarrow b_i$ *where* $i=1,2, \ldots, n$ *(where n is the number of base learners)*

for each base learner model (b_i) do

    pipeline is established to handle the data (da) in a streaming manner

    Outlierness score (Anomaly score) is determined for each instance of x and y in the data stream. This is passed as input to meta-learner stage

    if (score > threshold)

        return isAnomaly = 1 (anomalous)

    else

        return isAnomaly = 0 (non-anomalous)

    end if

end for

for each instance of x and y of (da), model trained dynamically in streaming manner to each base learner of b_i *, process continuing till end of the instances*

Algorithm 5.2: Estimating Meta-learner's output

Input: *Anomaly scores of base learners (b_i)*

Output: *Performance metrics (ROCAUC, Recall, Geometric Mean)*

LR ← meta-learner, AdaBoost ← meta-learner

x0 ← dictionary which collectively stores the scores of all base learners (b_i)

meta-learner (LR, AdaBoost) ← takes input from (x0), computes class posterior probability scores

retraining the meta-learner in streaming manner for next instance and continuing until the last instance of x0 passed

set threshold ← 0.5

set default threshold value of LR and AdaBoost meta-learner as 0.5 for detecting anomalies

if (class-posterior probability > 0.5)

    return anomalous value

else

    return non-anomalous value

end if

predictions of the meta-learner serve as the basis for evaluating the performance metrics of the framework

end

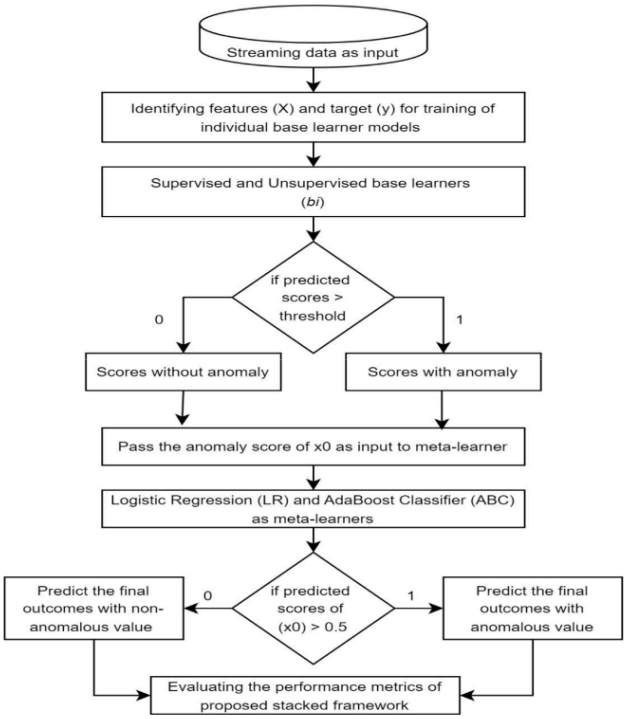


Fig. 5.3. Process flow as per the procedure described in Algorithm 5.1 and Algorithm 5.2.

5.1.2 Experimental Setup and Results Discussions

The proposed framework has been fully designed and implemented using online machine learning concepts to detect anomalies in real-time streaming data, utilizing the River and PySAD streaming frameworks. In the implementation part, the same experimental setup was used as described in an earlier chapter. The performance of the proposed stacked ensemble meta-learning is compared with that of nine popular

unsupervised anomaly detection techniques (HST, QF_OCSVM, TF_HST, QF_HST, HBOS, KitNet, RRCF, IForestASD, and KNNCAD) and one supervised classification technique, KNN, using three reliable performance metrics - ROCAUC, Recall, and Geometric Mean. The evaluation is done on nine popular streaming datasets (HTTP, CreditCard, SMTP, MaliciousURL, Android, covtype_1, cpu4, Falling, and Phishing). ROCAUC is computed by plotting TPR and FPR. Higher ROCAUC scores indicate better performance of the classifiers in identifying anomalous data.

1. Details of anomaly detection based streaming datasets

In this research, nine benchmark anomaly detection streaming datasets, namely HTTP, CreditCard, SMTP, MaliciousURL, Android, covtype_1, cpu4, Falling, and Phishing, are used for the experiments. Their overall sizes, key characteristics, anomalies present, and details of anomaly contributions (as provided in Table 1.1) are described in Chapter 1, Section 1.5.3.

2. Hyperparameter settings of the base learner models

HST: HST performance is optimal with the following parameters: 10 trees, a height per tree of 8, and a window size of 250 observations for calculating mass at every node. An anomaly threshold of 0.95 is set in HST, where higher scores indicate anomalies, and lower scores indicate normal observations.

QF_OCSVM: QF_OCSVM sets efficient outcome parameters by limiting the training faults and increasing the support vectors; the expected fraction of anomalies μ is 0.2. A quantile level of 0.95 signifies anomalous scores.

TF_HST: TF_HST is a detection model that combines the Threshold Filter (TF) with the Half Space Tree (HST). HST determines the threshold for identifying anomalies based on a value of 0.99, and TF_HST sets its own threshold at 0.95 for optimal performance scores.

QF_HST: QF_HST combines the Quantile Filter (QF) with the Half Space Tree (HST) algorithm. HST determines the threshold point for anomalous or non-anomalous data by setting it to a threshold of 0.95. This threshold point is then used by QF_HST to optimize its scores.

KNN: KNN is set to search for 10 nearest neighbors, and the larger window size for storing the latest observed samples is set to 500.

HBOS: An effective unsupervised approach, HBOS achieves its best performance when configured with 10 bins, a regularizer (alpha) of 0.1 to prevent overflow, and a dataset contamination level of 0.5.

KitNet: By setting the hyperparameters for the ensemble of autoencoders, such as a maximum autoencoder size of 10, a default gradient descent learning rate of 0.1 for all autoencoders, and a default hidden layer of 0.75, the KitNet anomaly detection algorithm achieves its optimal performance.

IForestASD: A default sliding window size of 2048 is configured for this unsupervised anomaly detection method that utilizes the isolation forest algorithm to analyze streaming data.

RRCF: The RRCF model is configured with 4 trees, a default tree size of 256, and a default shingle size of 4.

KNNCAD: In the KNNCAD method, when the number of instances in the probationary period is less than the specified number, the model assigns an anomaly score of 0.0.

Tables 5.1 to 5.9 and Figs. 5.3 to 5.20 showcase the results of all methods across the nine datasets for ROCAUC, recall, and Geometric Mean scores. The proposed unified framework outperforms all 10 methods across various streaming datasets for both stacked meta-learners. When using Logistic Regression (LR) for meta-learning on datasets such as HTTP, covtype_1, Android, and Falling, it achieves the highest ROC AUC scores of 0.9977, 0.9735, 0.9698, and 0.8999, respectively. Meanwhile, other datasets, such as Phishing, MaliciousURL, SMTP, cpu4, and CreditCard, exhibit average performance with ROCAUC scores ranging from 0.8753 to 0.7295. On the other hand, when utilizing the AdaBoost Classifier (ABC) for meta-learning, it attains the highest ROCAUC scores for HTTP, Android, covtype_1, Falling, and SMTP datasets, with scores of 0.9975, 0.9874, 0.9607, 0.9531, and 0.9000 while other datasets like Phishing, MaliciousURL, cpu4, and CreditCard demonstrate average performance with ROCAUC scores ranging from 0.8536 to 0.7254. Overall, the AdaBoost Classifier performs better across most datasets than Logistic Regression.

The evaluation also considers the Geometric Mean, an essential metric for imbalanced data. In this case, with meta-learner Logistic Regression (LR), the framework achieves higher Geometric Mean scores for HTTP, covtype_1, Android, and Falling datasets, reaching 0.9977, 0.9733, 0.9694, and 0.8948, while other datasets like Phishing, MaliciousURL, SMTP, cpu4, and CreditCard show satisfactory performance with Geometric Mean scores ranging from 0.8750 to 0.6777. When applying the AdaBoost Classifier for meta-learning, it achieves higher Geometric Mean scores for the HTTP, covtype_1, Android, Falling, and SMTP datasets, with scores of 0.9975, 0.9847, 0.9603, 0.9521, and 0.8944, respectively. For the remaining datasets, such as Phishing, MaliciousURL, cpu4, and CreditCard, the Geometric Mean

values range from 0.8524 to 0.6716. AdaBoost Classifier generally performs better across most datasets than Logistic Regression.

Recall scores are computed for both meta-learners, LR and ABC, and both achieve higher recall, with the Android dataset achieving 1.0000 and 0.9950, respectively. Other datasets show average performance for both stacked meta-learners. However, the KitNet model demonstrates superior performance across all datasets, with equal recall scores of 1.0000. In some datasets, HBOS, KNNCAD, and IForestASD also compete with our framework, all achieving equal recall scores of 1.000. The results are presented in Tables 5.2-5.10, and ROCAUC and Geometric Mean (GM) plots are shown in Figs. 5.3-5.11 and Figs. 5.12-5.20.

Table 5.1. Performance scores for anomaly-detection-based base learners and the proposed stacking framework on the HTTP streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	HTTP	HST	0.5180	0.0172	0.1307
2			QF_OCSVM	0.9014	0.9955	0.8965
3			KNN	0.4989	0.9941	0.0000
4			QF_HST	0.5038	0.0502	0.2192
5			TF_HST	0.5185	0.0172	0.1307
6	PySAD		HBOS	0.4428	0.3618	0.4353
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.5672	0.6562	0.5601
9			IForestASD	0.3088	0.0922	0.2201
10			KNNCAD	0.7090	0.9000	0.6828
11	Proposed Stacked_LR		Meta-learner-1	0.9977	0.9955	0.9977
12	Proposed Stacked_ABC		Meta-learner-2	0.9975	0.9950	0.9975

Table 5.2. Performance scores for anomaly-detection-based base learners and the proposed stacking framework on the CreditCard streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	CreditCard	HST	0.6906	0.6972	0.7586
2			QF_OCSVM	0.4895	0.9512	0.1628
3			KNN	0.3386	0.5061	0.0000
4			QF_HST	0.6981	0.4492	0.6522
5			TF_HST	0.6507	0.6565	0.6603

6	PySAD		HBOS	0.4853	0.6402	0.4599
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.4679	0.4392	0.4682
9			IForestASD	0.4550	0.4085	0.4297
10			KNNCAD	0.4849	0.4573	0.4841
11	Proposed Stacked_LR		Meta-learner-1	0.7295	0.4593	0.6777
12	Proposed Stacked_ABC		Meta-learner-2	0.7254	0.4512	0.6716

Table 5.3. Performance scores for anomaly-detection-based base learners and the proposed stacking framework on the SMTP streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	SMTP	HST	0.7415	0.4333	0.6517
2			QF_OCSVM	0.4442	0.3333	0.4302
3			KNN	0.3834	0.6667	0.0000
4			QF_HST	0.4937	0.0333	0.1783
5			TF_HST	0.7376	0.4333	0.6532
6	PySAD		HBOS	0.4335	0.3333	0.4217
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.2974	0.1333	0.2480
9			IForestASD	0.2689	0.3666	0.1296
10			KNNCAD	0.5103	0.5000	0.5102
11	Proposed Stacked_LR		Meta-learner-1	0.8333	0.6667	0.8165
12	Proposed Stacked_ABC		Meta-learner-2	0.9000	0.8000	0.8944

Table 5.4. Performance scores for anomaly-detection-based base learners and the proposed stacking framework on the MaliciousURL streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	MaliciousURL	HST	0.5690	0.0669	0.2558
2			QF_OCSVM	0.4439	0.4055	0.4422
3			KNN	0.5091	0.4324	0.0000
4			QF_HST	0.4869	0.0259	0.1568
5			TF_HST	0.5732	0.0837	0.2808
6	PySAD		HBOS	0.4563	0.4654	0.4561
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.4933	0.5176	0.4926
9			IForestASD	0.4827	0.5672	0.4752
10			KNNCAD	0.4649	0.3890	0.4586
11	Proposed Stacked_LR		Meta-learner-1	0.8487	0.7895	0.8466
12	Proposed Stacked_ABC		Meta-learner-2	0.8457	0.7953	0.8442

Table 5.5. Performance scores for anomaly-detection-based base learners and the proposed stacking framework on the Android streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	Android	HST	0.1357	0.0000	0.0000
2			QF_OCSVM	0.5930	0.2412	0.4774
3			KNN	0.7025	0.9950	0.0000
4			QF_HST	0.3819	0.0000	0.0000
5			TF_HST	0.1281	0.0000	0.0000
6	PySAD		HBOS	0.5000	1.0000	0.0000
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.5025	0.0050	0.0708
9			IForestASD	0.4573	0.9145	0.0000
10			KNNCAD	0.5000	1.0000	0.0000
11	Proposed Stacked_LR		Meta-learner-1	0.9698	1.0000	0.9694
12	Proposed Stacked_ABC		Meta-learner-2	0.9874	0.9950	0.9847

Table 5.6. Performance scores of anomaly detection-based base learners and the proposed stacking framework on the covtype_1 streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	covtype_1	HST	0.2390	0.0000	0.0000
2			QF_OCSVM	0.5016	0.4200	0.0648
3			KNN	0.5363	0.9850	0.0000
4			QF_HST	0.5282	0.0714	0.2652
5			TF_HST	0.2645	0.0000	0.0000
6	PySAD		HBOS	0.4998	0.9796	0.1400
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.4894	0.1328	0.3351
9			IForestASD	0.4989	0.5858	0.4912
10			KNNCAD	0.4995	0.4839	0.2446
11	Proposed Stacked_LR		Meta-learner-1	0.9735	0.9920	0.9733
12	Proposed Stacked_ABC		Meta-learner-2	0.9607	0.9894	0.9603

Table 5.7. Performance scores of anomaly detection-based base learners and the proposed stacking framework on the cpu4 streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	cpu4	HST	0.4182	0.0287	0.1584
2			QF_OCSVM	0.4999	0.0000	0.0000
3			KNN	0.4955	0.9713	0.0000
4			QF_HST	0.5162	0.0957	0.2994
5			TF_HST	0.4127	0.2775	0.1600
6	PySAD		HBOS	0.4687	0.3636	0.4567
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.3139	0.4210	0.2502
9			IForestASD	0.5000	1.0000	0.0000
10			KNNCAD	0.5000	1.0000	0.0000
11	Proposed Stacked_LR		Meta-learner-1	0.7976	0.5981	0.7723
12	Proposed Stacked_ABC		Meta-learner-2	0.8408	0.6842	0.8261

Table 5.8. Performance scores for anomaly-detection-based base learners and the proposed stacking framework on the Falling streaming dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	Falling	HST	0.4932	0.0000	0.0000
2			QF_OCSVM	0.3712	0.1239	0.2768
3			KNN	0.3411	0.4558	0.0000
4			QF_HST	0.5091	0.0563	0.2328
5			TF_HST	0.4842	0.0000	0.0000
6	PySAD		HBOS	0.4929	0.3474	0.4709
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.2976	0.0996	0.1877
9			IForestASD	0.3743	0.2305	0.3456
10			KNNCAD	0.4956	0.4350	0.4919
11	Proposed Stacked_LR		Meta-learner-1	0.8999	0.8050	0.8948
12	Proposed Stacked_ABC		Meta-learner-2	0.9531	0.9090	0.9521

Table 5.9. Performance Scores of Anomaly Detection-Based Base Learners and Stacking Ensemble Framework on Phishing Streaming Dataset.

S.No.	Models	Dataset	Anomaly Detection Techniques	ROCAUC	Recall	Geometric Mean (GM)
1	River	Phishing	HST	0.4896	0.7500	0.4826
2			QF_OCSVM	0.3146	0.0693	0.1970
3			KNN	0.4800	0.6113	0.0000
4			QF_HST	0.4876	0.0748	0.2595
5			TF_HST	0.5081	0.6843	0.6059
6	PySAD		HBOS	0.5048	0.0766	0.2674
7			KitNet	0.5000	1.0000	0.0000
8			RRCF	0.5009	0.0018	0.0420
9			IForestASD	0.4891	0.8357	0.3450
10			KNNCAD	0.5075	0.0875	0.2850
11	Proposed Stacked_LR		Meta-learner-1	0.8753	0.8504	0.8750
12	Proposed Stacked_ABC		Meta-learner-2	0.8536	0.8084	0.8524

5.1.3 Performance Visualizations

1. Visualization plots of ROCAUC for all models.

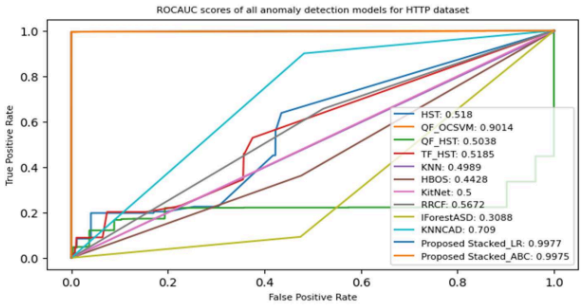


Fig. 5.3. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the HTTP dataset.

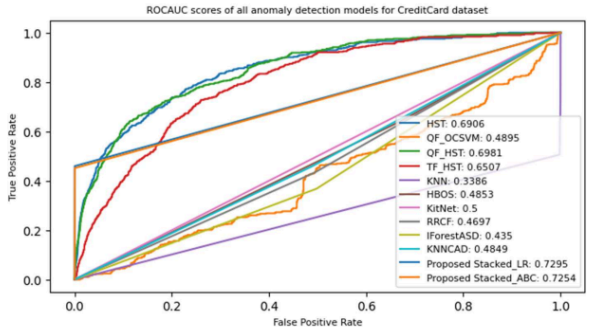


Fig. 5.4. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the CreditCard dataset.

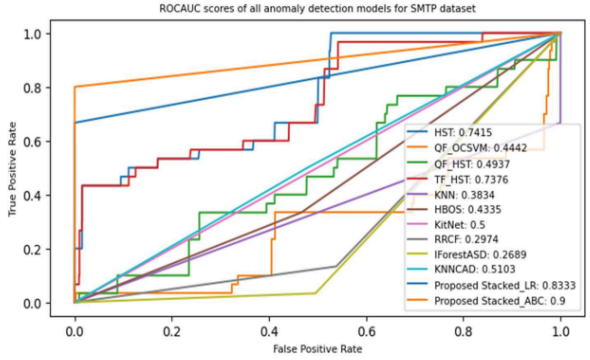


Fig. 5.5. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the SMTP dataset.

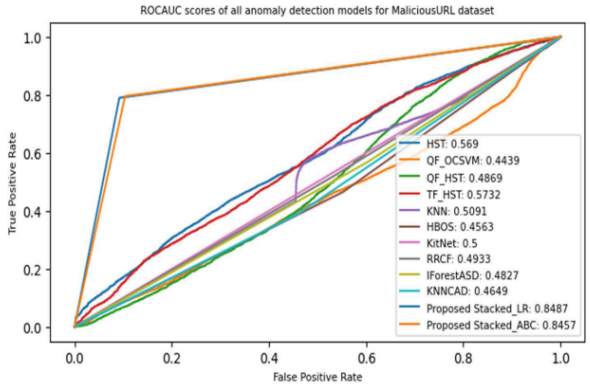


Fig. 5.6. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the MaliciousURL dataset.

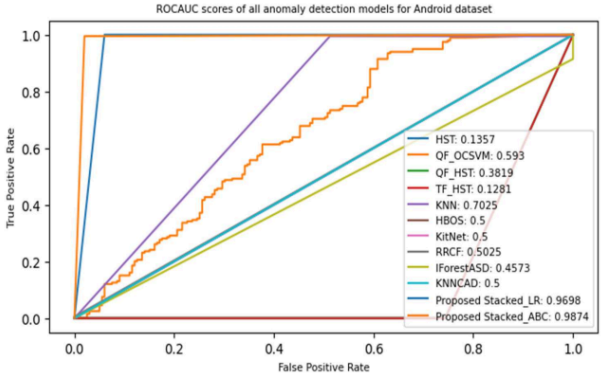


Fig. 5.7. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the Android dataset.

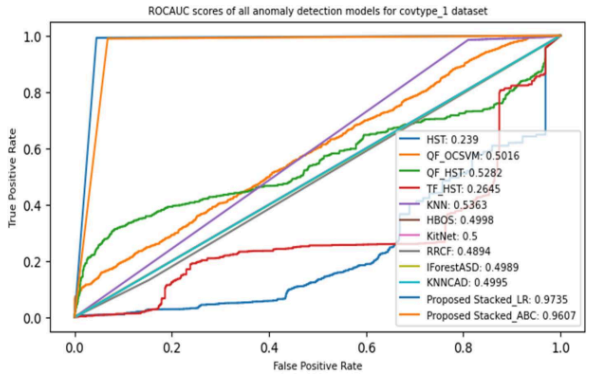


Fig. 5.8. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the covtype_1 dataset.

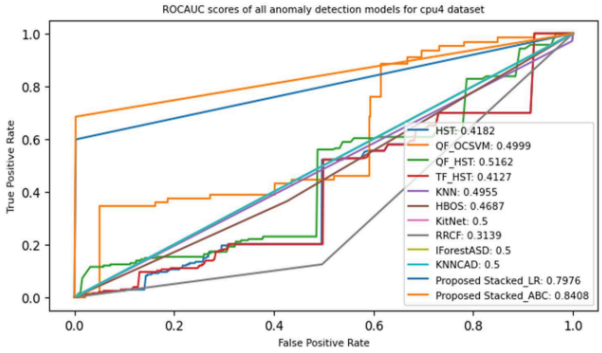


Fig. 5.9. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the cpu4 dataset.

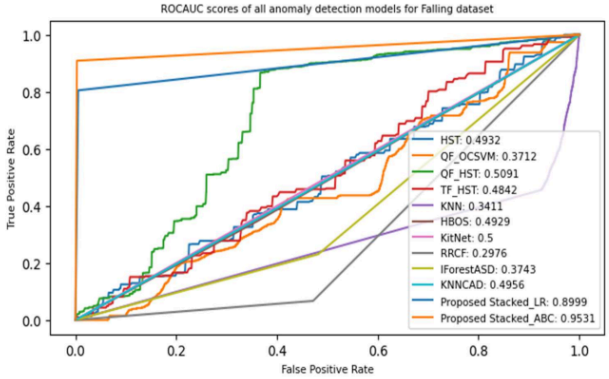


Fig. 5.10. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the Falling dataset.

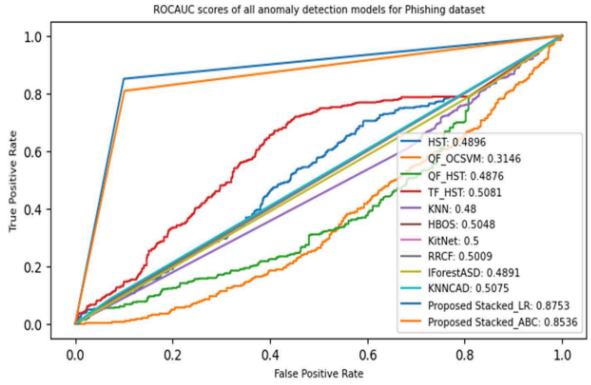


Fig. 5.11. ROCAUC plot of all anomaly detection methods with the proposed stacked framework using the Phishing dataset.

2. Visualizing plots of Geometric Mean (GM) for all models with the proposed stacked model

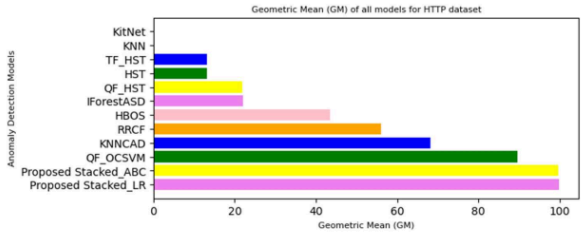


Fig. 5.12. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the HTTP dataset.

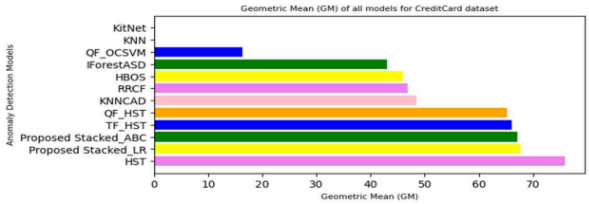


Fig. 5.13. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the CreditCard dataset.

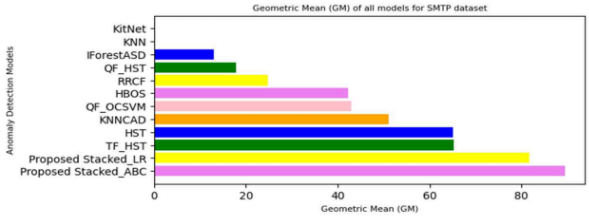


Fig. 5.14. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the SMTP dataset.

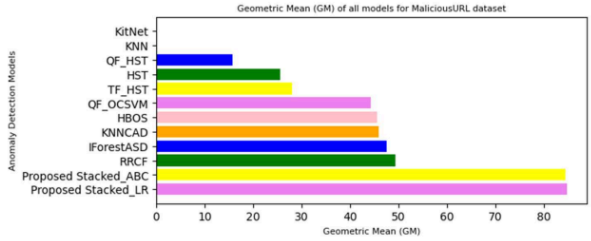


Fig. 5.15. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the MaliciousURL dataset.

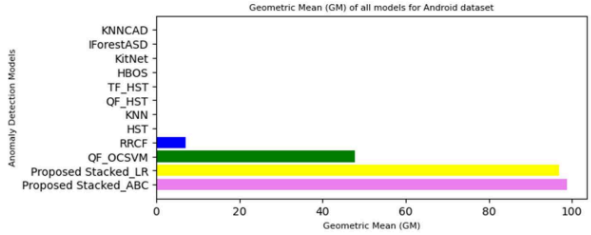


Fig. 5.16. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the Android dataset.

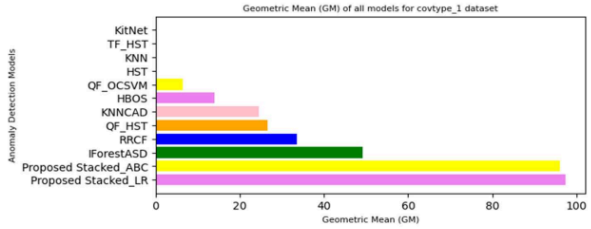


Fig. 5.17. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the covtype_1 dataset.

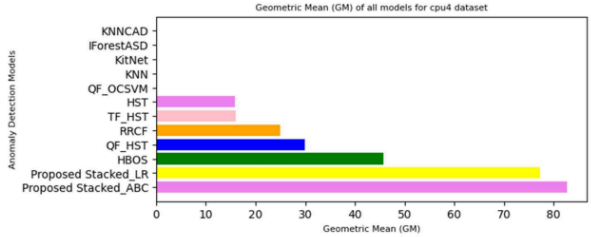


Fig. 5.18. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the cpu4 dataset.

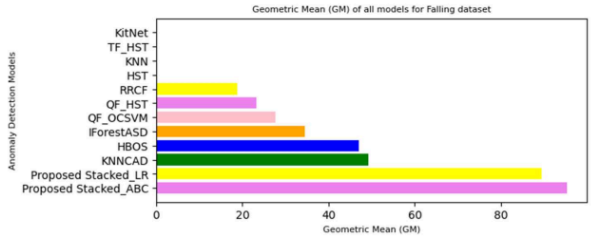


Fig. 5.19. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the Falling dataset.

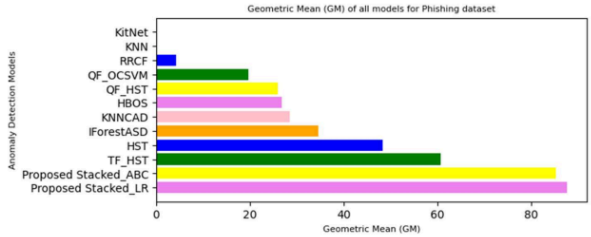


Fig. 5.20. Visualizing the Geometric Mean (GM) of all anomaly detection methods, including the proposed stacked framework, using the Phishing dataset.

5.2 OBJECTIVE-2 (b): Anomaly Detection Methodologies for Streaming Data: Trends and Perspectives.

Santosh Kumar Ray and Seba Susan's survey paper title "Anomaly Detection Methodologies for Streaming Data: Trends and Perspectives "is currently under review in the ***SN Computer Science (Springer), (SCIE-indexed, IF: 2.3), (Communicated).***

5.2.1 Overview of Supervised vs Unsupervised Methods

The principal objective of this survey is to provide an overview of anomaly detection methods for streaming data. Datasets in the real world often contain unusual or abnormal data points, which may be caused by factors such as data corruption, experimental errors, or human mistakes. These anomalies can degrade the method's accuracy; therefore, it is crucial to ensure the dataset is free of them to train a strong, reliable model. We first categorize the various techniques available in the literature into the following categories: statistical techniques [118], distribution-based techniques [119], proximity-based techniques [120], reconstruction-based techniques [121], machine learning-based techniques [122], and ensemble-based techniques [123, 124]. These are broadly classified into unsupervised and supervised approaches for anomaly detection for streaming data. All these methods have their own strengths and weaknesses, and the characteristics of the dataset determine which method to choose, the type of anomaly, and the desired level of accuracy. We then provide a detailed review of each technique, discussing its strengths and limitations. Finally, we conclude our survey with a deliberation on open research challenges and prospects in this domain. The main contributions of this review paper are listed below:

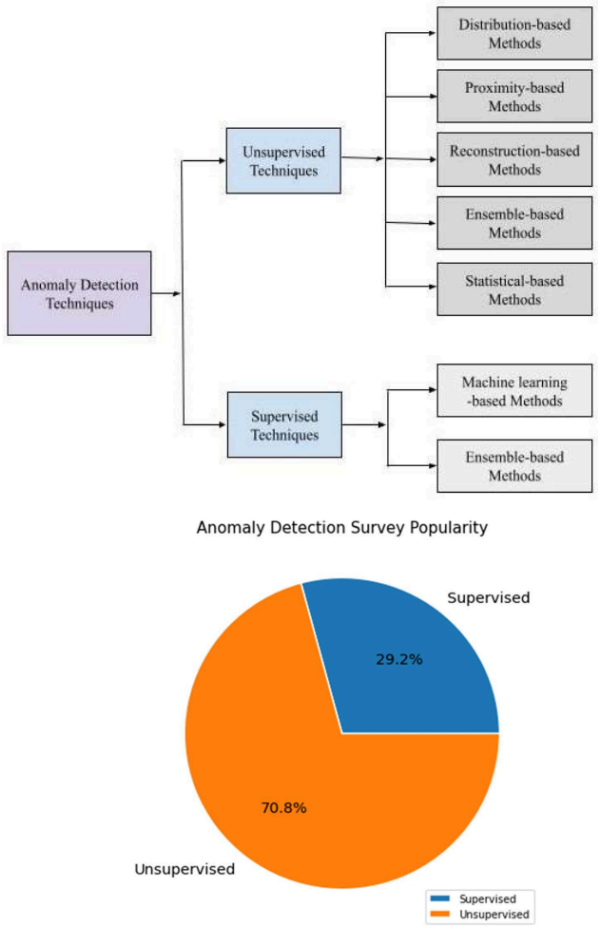
1. The investigation of contemporary research focused on handling real-time anomaly detection in data streams.
2. Analysis of anomaly detection techniques based on their strengths and limitations, with a focus on identifying effective solutions.
3. Demonstration of the robustness of various methods in detecting anomalies in unbalanced streaming datasets.
4. Examination of the tools, applications, and methods applied for detecting anomalies in high-dimensional data streams.

5. Emphasizing the research gaps and guiding future researchers.

5.2.2 Popularity of Anomaly Detection Techniques

The popularity of unsupervised and supervised methods for detecting abnormalities varies depending on the specific method and its applications. However, unsupervised techniques are more commonly used in general due to their simplicity and potential to identify unknown anomalies. Among unsupervised techniques, statistical-based, distribution-based, and proximity-based methods are more widely used than reconstruction-based and unsupervised ensemble-based methods. Supervised techniques are less popular because they require labeled data and are typically only used when specific anomalies need to be detected with high accuracy. However, the popularity of each technique ultimately depends on the characteristics of the dataset, the type of anomaly present, and the desired level of accuracy. The popularity plot in Fig. 5.21 (b) has been constructed using a diverse range of technical and scientific sources. These include literature surveys, insights from online forums and communities, web search analysis, publication rates, quantitative analysis, and current trends in the field of anomaly detection for streaming data.

The taxonomy of anomaly detection methods for streaming data and their corresponding popularity chart are shown in Fig. 5.21 (a) and Fig. 5.21 (b), respectively.



Figs. 5.21 (a). Different categories of anomaly detection methods **(b).** Popularity of unsupervised and supervised anomaly detection methods in (%).

5.2.3 Evaluation of Anomaly Detection Techniques through Software Tools and Visualizing the Anomaly Positioning in Data

Anomaly detection, also known as irregularity detection, involves identifying uncommon items, events, or observations that deviate from the typical distribution of a population. These tasks often lack a clear definition of what constitutes a correct answer, which makes specialized outlier detection algorithms particularly useful in fields that deal with huge quantities of unlabeled instances and require a dependable method of recognizing patterns. Several datasets have been introduced specifically for anomaly detection in data streams. Lavin et al. 2015 [227] proposed the Numenta Anomaly Benchmark (NAB) dataset to evaluate and compare real-time anomaly detection algorithms using labeled time series data. This research also explored improvements, including the utilization of diverse and complex time series data from additional domains, the investigation of various hyperparameters and feature selection methods, and the provision of more detailed information about the dataset. Ahmad et al. (2017) [129] introduced UNSAD, a novel unsupervised anomaly detection procedure that can identify abnormalities in streaming data without requiring manual labeling or prior knowledge of the system's normal behavior. UNSAD outperformed existing algorithms regarding precision, velocity, and memory utilization when tested on real-world datasets. Singh et al. (2017) [228] critically evaluated the Numenta Anomaly Benchmark (NAB), a widely used dataset for evaluating anomaly detection methods, and identified its limitations and potential biases. They found that NAB had a lack of diversity in the kinds of anomalies, making it difficult to compare different algorithms. To address these limitations, they proposed a new benchmark dataset called "Streaming Time Series Anomaly Detection", which includes a more diverse set of anomalies and a streaming data format, better representing real-world scenarios.

Lee et al. 2020 [229] introduced a novel algorithm named RePAD that can identify anomalies in streaming time series data in real-time. The algorithm employs Long Short-Term Memory (LSTM) and utilizes recent data points to predict the likelihood of an anomaly occurrence soon. The algorithm can dynamically adjust its detection threshold over time, making it resilient to slight alterations in the pattern of the time series and enabling it to identify irregularities either beforehand or promptly. Tkach et al. (2023) [230] introduced and utilized the reduced description length principle to develop a framework for identifying the simplest model that can explain the data. Data instances that do not fit this model are considered anomalies or outliers. PyOD [115] is a Python-based toolkit that provides extensive support for detecting outliers in a scalable manner. The PyOD anomaly detection benchmark utilizes various approaches for detecting anomalies, including statistical, proximity, reconstruction, machine learning, and ensemble methods. This benchmark is useful for evaluating the

performance of anomaly detection models on both real-world and artificial streaming datasets.

To compare these methods effectively, it is crucial to identify the types and limits of discordant or outlier data points, as well as the inliers. Figs 5.22, 5.23, 5.24, and 5.25 provide visual representations of the locations of anomalies identified by different methods. We observe from Fig. 5.22 that the statistical techniques HBOS, KNN, and OCSVM show similar performance for anomaly detection. Among the proximity-based methods from Fig. 5.23, LOF performs the best compared to CBLOF and ABOD. Among the reconstruction methods shown in Fig. 5.24, PCA detects a wider range of inliers than MCD. Among the machine learning methods shown in Fig. 5.25, Isolation Forest performs better in comparison to Feature Bagging.

a) Statistical-based Methods

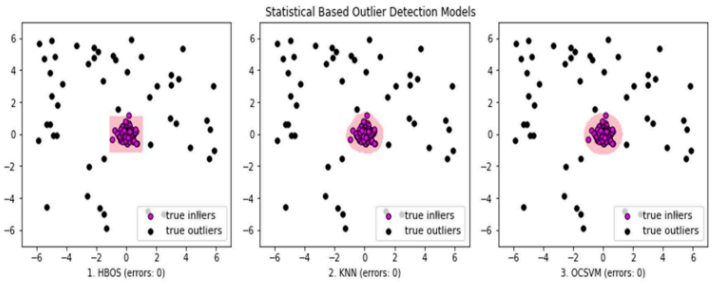


Fig. 5.22. Visualization of Anomaly Positions in Data using Statistical-based Outlier Detection Models.

b) Proximity-based Methods

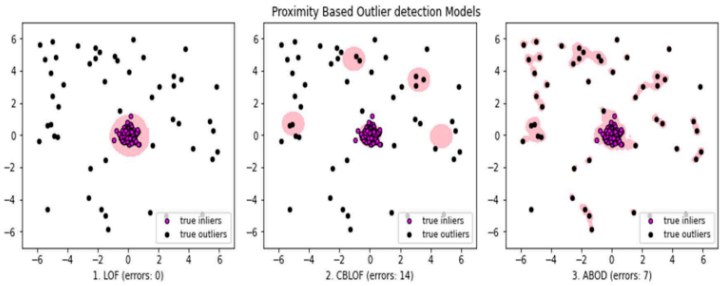


Fig. 5.23. Visualization of Anomaly Positions in Data using Proximity-based Outlier Detection Models.

c) Reconstruction-based Methods



Fig. 5.24. Visualization of Anomaly Positions in Data using Reconstruction-based Outlier Detection Models.

d) Ensemble-based Machine Learning Methods

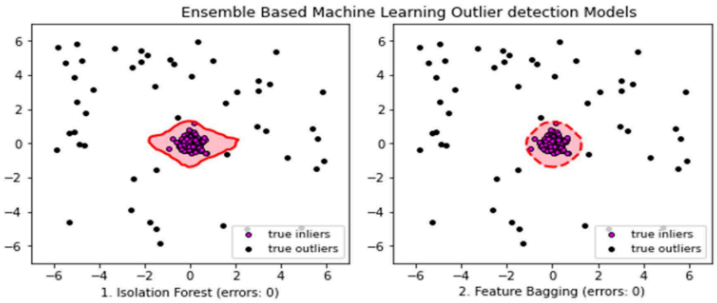


Fig. 5.25. Visualization of Anomaly Positions in Data using Ensemble-based Machine Learning Outlier Detection Models.

5.2.4 Limitations and Strengths of Anomaly Detection Techniques

1. Limitations

Anomaly detection methods for streaming data suffer from various limitations that present significant obstacles to their efficacy. The need for real-time processing poses a major challenge, requiring substantial computational resources and algorithms that can handle high-velocity data streams to promptly detect anomalies. The scarcity of historical data in streaming environments hinders the development of accurate models that often rely on comprehensive learning histories. Additionally, challenges such as imbalanced data, high dimensionality, and the complication of identifying labeling anomalies further contribute to these limitations. Overcoming these issues calls for innovative approaches and robust strategies to achieve reliable anomaly detection in data streams. In this context, the illustrated workflow relies on batch-wise static data analysis with predefined models and thresholds, making it unsuitable for dynamic streaming environments. Such approaches lack adaptability to evolving data distributions and require high computational resources for real-time anomaly detection. The process diagram for conventional anomaly detection methods is shown in Fig. 5.26. Several notable drawbacks of anomaly detection techniques are discussed as follows:

- **Normal behavior is not known or cannot be modelled accurately:** Some techniques, such as statistical-based and machine learning approaches, require the

assumption that the usual behavior of the data instance is known or can be modelled accurately. If this assumption is not met, these methods may not be effective at detecting anomalies.

- **Sensitivity to hyperparameters**: Some techniques, such as proximity-based and machine learning-based methods, may be intuitive for choosing hyperparameters, such as the distance measure or the count of nearest neighbors. This can make it difficult to tune these techniques for a particular dataset.
- **Limited to detecting only a specific type of anomaly**: Some techniques are designed to detect only a specific kind of anomaly, such as point anomalies or contextual anomalies. These techniques may not be valuable for detecting other kinds of anomalies.
- **Difficulty in detecting subtle or rare anomalies:** Some techniques may not be valuable at detecting subtle or rare abnormalities, as these anomalies may not be significantly different from the normal data.
- **High computational complexity:** Some techniques, such as proximity-based techniques, may have great computational difficulty and may not be appropriate for large datasets.
- **Limited to univariate or low-dimensional data:** Some techniques, such as statistical-based techniques, may be Ineffective at handling vast amounts of data or data with complex relationships.
- Classification approach, particularly multi-class methods, utilizes robust algorithms capable of effectively distinguishing instances belonging to various classes.
- Classification-based methods have a fast-testing stage since they only need to compare each test instance with the pre-calculated model.

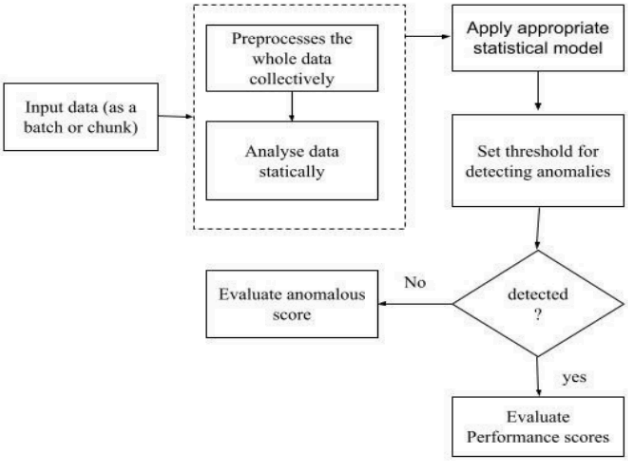


Fig. 5.26. Process diagram for conventional anomaly detection methods.

2. Strengths

Anomaly detection techniques utilizing streaming data offer significant advantages in various applications. Key advantages include real-time processing for timely anomaly identification and intervention. They can operate on continuously arriving data streams, ensuring up-to-date anomaly detection in dynamic systems. The continuous nature of streaming data enables ongoing model refinement, enhancing detection accuracy and reliability over time. Leveraging these strengths, anomaly detection techniques in streaming data enable early anomaly detection, improve decision-making processes, and enhance system performance and security in real-world scenarios. In the illustrated workflow, data is processed incrementally in a one-to-one manner, incoming samples are dynamically analyzed, and the model is continuously updated based on previous observations, making it well-suited for evolving streaming environments. The process diagram for streaming-based anomaly detection methods is shown in Fig. 5.27. Certain noteworthy advancements of anomaly detection methods are discussed as follows:

- Strategies for anomaly detection can distinguish unusual patterns in data instances that are not immediately distinguishable by simply looking at the data.
- Anomaly detection approach can be applied to find fraudulent or malicious activity in datasets, such as cyber-attacks or credit card fraud.
- Anomaly detection process can be applied to discover equipment failures in manufacturing or other industries, allowing for proactive maintenance and repair.
- Anomaly detection models can be used to identify issues with quality control in manufacturing or other industries, allowing for corrective action to be taken.
- Anomaly detection models can be used to find changes in input sample patterns over time, allowing for early detection of trends or changes in the data.
- A multi-class classification approach requires accurate labels for different normal classes, which can be difficult or impractical to acquire for the minority classes in an imbalanced class scenario.
- The use of a classification-based approach assigns labels to test instances, which may present a limitation when seeking a meaningful anomaly score for those test instances that deviate from the training samples.

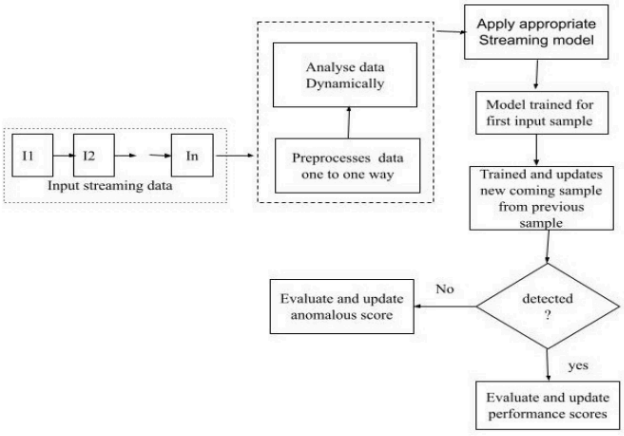


Fig. 5.27. Process diagram for streaming anomaly detection methods.

5.2.5 Identifying Opportunities for Improvement in Anomaly Detection Techniques

The text discusses the process of assessing current anomaly detection concepts and identifying areas for enhancement. Anomaly detection involves identifying unusual instances in a sample that may indicate deviations from the norm or potential obstacles. Improving these techniques can lead to more accurate and effective anomaly detection, benefiting applications like fraud detection, network security, and healthcare. Identifying opportunities for improvement may entail researching and comparing various techniques, understanding their strengths and weaknesses, and proposing new approaches or adjustments to existing methods listed in Table 5.10. Table 5.11 illustrates the limitations of the existing crop of anomaly detection methods for data streams. Table 5.12 outlines the primary challenges and key suggestions for overcoming these challenges in anomaly detection methods.

Table 5.10. Research gaps in existing anomaly detection techniques.

Research Gap	Description
Scalability	Many existing anomaly detection techniques are not scalable to large data sets, making them impractical for use on big data.
Adaptability	Anomaly detection techniques may encounter challenges in adapting to changes in data patterns over time, leading to a decline in their effectiveness.
Explainability	Many anomaly detection techniques do not provide clear explanations for why a particular data point was identified as an anomaly, making it difficult to understand the decision-making process.
Human-machine interaction	There is a lack of research on how to effectively integrate anomaly detection models into human procedures, making it hard to employ the output of these techniques in a practical manner.
High dimensionality	Anomaly detection techniques may strive to identify anomalies in data sets with high dimensionality, as it becomes increasingly difficult to identify.

Table 5.11 Limitations highlighted by other surveys on anomaly detection.

Techniques	Highlight	Limitations

Rassam et al. 2013 [96]	This survey paper highlights the latest advancements in data anomaly detection in wireless sensor networks and critically assesses each method's merits and demerits through comparative analysis.	**1.** An area needing improvement is testing these models in real-world settings to assess their effectiveness across domains. **2.** Exploring innovative techniques to address current limitations in anomaly detection would improve the field and sensor data accuracy.
Hasani et al. 2019 [97]	This approach can significantly improve anomaly detection in periodic data streams, enhancing its value in the field. It adapts instantly to changes in data patterns.	In the future, this method aims to improve its anomaly detection process by including a Harmonic Wavelet Genetic Algorithm (HW GA) and optimizing the Time-Dependent Harmonic Wavelet (TDHW). This will strengthen the model's ability to identify anomalies in large data streams.
Thudumu et al. 2020 [98]	**1.** This survey focuses on techniques addressing high-dimensional data challenges, like the curse of dimensionality, data complexity, and computational costs, highlighting their strengths and limitations. **2.** This survey guides on techniques for various big data scenarios and examines recent trends in anomaly detection. It provides an overview of cutting-edge models for high-dimensional big data.	1. Evaluation of the persuasion of unbalanced datasets on the efficiency of anomaly-identifying concepts, which is a prevalent issue in practical applications. **2.** Discusses how different data distributions affect anomaly detection performance and output authenticity. **3.** Consider the role of human experts in anomaly detection, emphasizing interpretability and human-in-the-loop methods.
Yu et al. 2020 [99]	**1.** The research's novel aspect is its incremental approach, enabling efficient data processing as received, instead of waiting for the complete dataset. **2.** This approach reduces computational needs and allows real-time processing, suitable for large, dynamic data sets.	**1.** One possible further research area is exploring the scalability and robustness of the method with larger, more complex data sets, and its ability to handle noise and irrelevant data. **2.** To extend the algorithm's applicability across domains and scenarios, including cases with missing or incomplete data.
Samara et al. 2022 [100]	This survey evaluates the challenges of current techniques, guiding future IoT and outlier-detection work. It categorizes deviation detection by data type, method, and application.	**1.** Develop new outlier detection methods that overcome existing limitations and account for IoT data's unique attributes. **2.** Investigation of outlier detection in IoT applications like health care, industrial IoT, and smart cities.

Chatterjee et al. 2022 [101]	The survey paper provides an overview of methods for detecting anomalies in IoT, covering anomaly types, detection models, and their applications.	Possible future fields in IoT abnormality detection include: • Deep learning-based methods • Federated learning, • Multi-modal data fusion, • transfer learning • Edge computing.

Table 5.12 A summary of difficulties encountered, solutions offered, accuracies achieved, and future recommendations for some popular anomaly detection techniques.

Ref.	Anomaly detection techniques for streaming data	Traditional methods difficulties	Resolution of the problem	Datasets and Accuracy scores	Future recommendations
Ding et al. 2013 [32]	IForestASD	Concept drift challenges continuous anomaly detection.	Streaming concepts address drift issues, are suitable for fixed-size windows, and support score updates.	HTTP (93%), SMTP (83%), Forest Cover (96%), Shuttle (71.5%).	This research aims to track variations in data distribution and adjust the model to maintain performance. Fixed window sizes are impractical in real-time; therefore, variable window sizes are important.
Salehi et al. 2014 [105]	Weighted Ensemble Framework for Detecting Anomaly with Switching Data Streams	High error rate with false positives in a static monitoring environment.	Dynamic environment, computationally efficient, and high adaptability	IBRL (98.23%), TAO (90.81%), Synthetic (98.17%)	This is the first look at alternative clustering evaluation methods. In the future, it will reduce memory usage and explore techniques for finding optimal window boundaries
Ahmad et al. 2016 [106]	Hierarchical Temporal Memory (HTM)	Capable of spotting spatial deviations only, computations are much slower	The base system frequently changes, requiring detectors to continually adapt and learn from evolving statistics while making predictions.	Numenta Anomaly Benchmark (NAB) dataset (NAB Score 65.3)	The algorithm's accuracy can be improved by adding extensions, such as ensemble methods or different distributions for anomaly scores. Error analysis shows errors aren't always linked to other algorithms, and assuming a Gaussian distribution may be inappropriate.

Veerawali et al. 2017 [107]	Anomaly detection model for healthcare	Existing anomaly detection methods need manual training per patient due to individual differences and variations in physiological signals.	This research aims to develop a real-time, personalized anomaly detection system for wearable healthcare devices.	Arrhythmia (0.989-1.0), Supraventricular (0.960-0.992), Sinus rhythm (1.0), Long-term (1.0), ST-change (0.994-1.0), ST-T (0.992-1.0).	This method enhances detection and classification accuracy and lessens processing burdens in wearable healthcare systems when combined with existing methods.
Choudhary et al. 2017 [108]	Real-time anomaly detection techniques	Choosing the right model is challenging because it involves balancing real-time performance and accuracy, and there is limited characterization.	This method aims to assist in selecting the most suitable anomaly detection method, accounting for changes in variance and seasonal levels.	NAB YAD HFT OPs IoT Health	The future aim is to broaden the evaluation to include other data types, such as real-time streams from Facebook Live, Twitter Periscope, and similar platforms.
Munir et al. 2019 [109]	FuseAD	Suffers from large restrictions and a lack of scalability.	Fusing two models (Statistical and DL) in a residual fashion.	Yahoo Web scope Dataset (60%)	In the future, this concept aims to broaden the assessment to include multivariate time-series datasets by combining it with other conventional distance-based anomaly detection models, such as LOF and k-NN.
Jain et al. 2021 [110]	Preprocessed Isolation Forest (PiForest)	Improper drift handling in streaming data, along with the cost of tree-based methods, strains resources.	Reducing data size speeds up model updates, making them more adaptable and lowering concept drift.	http (KDDCUP99) (0.99), Satellite (0.67), Mulcross (0.87), Forest cover (0.70), Breastw (0.88), Shuttle (0.99).	This work aims to improve performance in constrained and unconstrained Wireless Sensor Networks.
Bomoil et al. 2021 [111]	Streaming Subsequence Anomaly Detection (SAND)	Current methods require expertise or become costly with recurring anomalies.	A new streaming model enables continuous updates, adapting to changing data and discarding outdated information.	MBA 803 (0.97), MBA 805 (0.98), MBA 806 (0.80), MBA 820 (0.96), SED (0.96).	Future improvements could include increasing scalability through parallel processing or deep learning.

Xiaolan et al. 2022 [112]	Evolving Anomaly Detection for Network Streaming Data (EADNSD)	Large, rapidly changing datasets arrive, hindering the creation of accurate network behavior patterns.	This method detects anomalies in network data streams by continuously updating clusters as new data arrives.	KDDCUP'99 (100%), NSL_KDD (100%), CIDDS-001(98.86%).	Future research may focus on reducing memory use and on adapting thresholds for different NID applications. Expert experience is needed to set system parameters, such as local and global outlier thresholds.

5.3 Conclusion and Future Scope

This chapter presented a hybrid stacked ensemble framework for anomaly detection in streaming data, along with a comprehensive survey of existing techniques in the domain. The proposed multi-framework model, which integrates River and PySAD with meta-learning, demonstrated strong performance across multiple real-world datasets, using robust evaluation metrics such as ROC-AUC, Recall, and Geometric Mean. Combining unsupervised and supervised methods within a stacked ensemble proved effective at improving detection accuracy and robustness while maintaining efficient processing suitable for real-time applications.

In parallel, the survey provided a structured overview of anomaly detection approaches, including statistical, proximity-based, reconstruction-based, machine learning, and ensemble techniques. It highlighted the strengths, limitations, and adaptability of these methods across different streaming scenarios, while also identifying key challenges such as the need for context-aware, adaptive, and reliable detection mechanisms.

Overall, the chapter emphasizes that while significant progress has been made in designing efficient anomaly detection frameworks, challenges such as detecting rare anomalies, improving model adaptability, and handling diverse data streams persist. Future work will focus on enhancing detection capabilities through advanced learning techniques, better parameter optimization, and integration with other data analytics approaches to support more scalable and intelligent streaming systems.

CHAPTER 6

Ensemble-Based Framework for Concept Drift Adaptation and Network Intrusion Detection in Streaming Data

6.1 OBJECTIVE 1, 2 & 3: Real-Time Intrusion Detection via Drift-Sensitive and Error-Driven Dynamic Ensemble Adaptation in Evolving Network Streams.

Santosh Kumar Ray and Seba Susan's research paper title "Real-Time Intrusion Detection via Drift-Sensitive and Error-Driven Dynamic Ensemble Adaptation in Evolving Network Streams" is currently under review at the ***Computer Network Journal*** **(Elsevier), (SCIE-indexed, IF: 4.6), (Communicated).**

6.1.1 Methodology

Addressing gaps in earlier research, we propose a novel online machine learning framework for real-time intrusion detection in network streams, designed to handle concept drift and detect anomalies effectively. The system is tailored for continuous, high-throughput packet streams and supports dynamic model updates to adapt to evolving cybersecurity threats. To achieve this, we develop an error-driven dynamic ensemble framework for real-time network traffic data, utilizing an Incremental Packet-to-Vector (IP2V) approach, for drift-sensitive anomaly detection using online drift-adaptive machine learning classifiers. This framework processes network packets as they arrive, converting them incrementally into feature vectors suitable for machine learning analysis without the need for batch processing. The model is designed to adapt dynamically to changes in the network traffic patterns, which can be caused by concept drift or evolving attack methods. It employs five drift-adaptive online classifiers as base learners: ARF-EDDM (Adaptive Random Forest with Early Drift Detection Method), ARF-DDM (Adaptive Random Forest with Drift Detection Method), ARF-ADWIN (Adaptive Random Forest with Adaptive Windowing), ARF-HDDM_A (Adaptive Random Forest with Hoeffding Drift

Detection Method - Average), ARF-HDDM_W (Adaptive Random Forest with Hoeffding Drift Detection Method- Weighted). These classifiers form a dynamic ensemble, where both the weights assigned to each classifier and the ensemble's composition change over time in response to the streaming data's drift and error characteristics. This allows the intrusion detection system to maintain high accuracy and robustness, as the models adapt continuously to new network traffic conditions, detect anomalies early, and reduce false alarms.

The end-to-end workflow of the proposed NIDS method and the comparative methods (AdaBoost, Bagging, ENIDrift, KitNET, IForestASD, and HELAD) is illustrated in Fig. 6.1. In the initial phase, the preprocessed network data is segmented into fixed-size packets, incrementally fed to the five base models. Each model is trained and tested sequentially through instance-wise incremental learning. Based on performance, model errors are evaluated; if a model's error exceeds a predefined threshold ($\theta > 0.5$), it is dynamically replaced with a randomly selected ARF model employing a different drift detector. In the final phase, an Isolation Forest (IForest) algorithm is applied to the accuracy scores of the models to distinguish between normal and anomalous patterns in the data stream. The Isolation Forest was applied to identify abnormal accuracy variations, enabling assessment of each drift detection model's reliability and stability over time. The proposed framework combines drift-sensitive adaptive learning, error-based dynamic model replacement, and anomaly detection to ensure robust and reliable performance of NIDS in dynamic data environments. The subsequent part of this section details the design and implementation of the proposed dynamic ensemble.

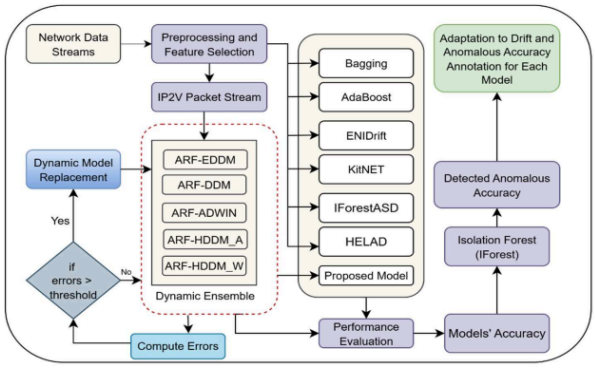


Fig. 6.4. End-to-end workflow of the proposed and comparative NIDS approaches using streaming network data.

1. Drift-Adaptive Online Machine Learning for Evolving Data Streams

The most critical aspect of data analysis in evolving datasets is its time dependence and real-time processing; data distributions may change over time, a phenomenon known as concept drift [104]. These drifts can be sudden, gradual, recurring, or incremental. The primary challenge for a classification model in dynamic environments is adapting to evolving data distributions through one-to-one data analysis in real-time; otherwise, the model's performance will progressively decline. We utilize the Adaptive Random Forest (ARF) [6] online classifier, which dynamically adjusts to concept drift as identified by the drift detection methods in network data streams. Below is a brief overview of the drift-adaptive online machine learning algorithms used in this study. Each algorithm integrates a distinct drift detection method as its core component, while employing ARF as the base classifier.

ARF-EDDM: A practical approach for detecting gradual drift based on the distribution of misclassification errors in evolving data streams. This method identifies slow, gradual changes without increasing the rate of false positives. Adaptive learning principles allow these changes to be processed sequentially, as gradual drift becomes tricky to detect due to its slow manifestation.

ARF-DDM: A prominent technique for rapidly detecting both gradual and sudden drift. Occasionally, the detection rate for gradual changes may rise, leading to longer detection times. When samples in the stream are unavailable for an extended period, it is considered an error. Adaptive learning algorithms can instantly adapt to both types of change in evolving data.

ARF-ADWIN: A drift detection model based on the adaptive windowing concept. It detects drift when the distribution becomes unstable, disrupting the distributions of the sub-windows. Adaptability is the key feature of this model; it learns from every instance in the streams and adjusts accordingly.

ARF-HDDM_A and ARF-HDDM_W: Based on probability inequalities, this resilient, universal, and bounded random variable concept offers theoretical guarantees regarding distributional changes. Thus, it provides two robust approaches: one is efficient on a moving average and suitable for detecting sudden drift, referred to as HDDM_A, while the second is a weighted average that effectively identifies gradual changes, known as HDDM_W. Both methods are computationally robust and quickly adapt to change detection while demonstrating strong performance evaluations.

Algorithm 1 details the steps of the adaptive learning algorithm, which utilizes incremental instance-by-instance learning through the learn_one functionality.

Algorithm 1: adaptive_learning (model, X_train, y_train, X_test, y_test)

Step 1. Initialize variables:
initialize accuracy metric
set metric = Accuracy ()
Counter for evaluated data points
set i = 0
List to record the number of evaluated data points and real-time accuracy
set t = [], and m = []
List to store true labels and predicted labels of the test set
set yt = [], yp = []

Step 2. Train the model on the training set:
for each (xi1, yi1) in stream (X_train, y_train) do
where i represents the number of features and target
model.learn_one (xi1, yi1)
end for
Initialize test sample counter
set n = 0

Step 3. Predict and update the model on test data
for each (xi, yi) in stream (X_test, y_test) do
set y_pred = model.predict_one (xi)
model.learn_one (xi, yi)
Update real-time accuracy
metric = metric.update (yi, y_pred)
Store evaluation count and accuracy
append i to t
append (metric.get () * 100) to m
Store true and predicted labels
append yi to yt
append y_pred to yp
Increment evaluated data point counter
i = i + 1
end for

Step 4. Compute performance metrics as
set accuracy, Precision, Recall, F1-score, ROC-AUC

Step 5. Return result
return t, m, performance metrics
end function definition

Step 6. Define a list of drifting models
set models = [ARF-EDDM, ARF-DDM, ARF-ADWIN,
ARF-HDDM_A, ARF-HDDM_W]

Step 7. Initialize a dictionary to store results
set results = {}

Step 8. Loop through each model and call the adaptive_learning function
for each model in models, do
print "Evaluating model: " + model. name
call adaptive_learning function for the current model
set t, m, metrics = adaptive_learning (model, X_train, y_train, X_test, y_test)
set results[model.name] = {
t: t (evaluated data points)
m: m (real-time accuracy)
metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC}
end for

Step 9. Print final results for all models
for each model in results, do
print "metrics": Accuracy, Precision, Recall, F1-score, ROC-AUC
end for

2. Proposed Error-Sensitive Dynamic Ensemble Adaptation with IP2V for Real-Time Intrusion Detection

The proposed model employs the Variance Threshold technique[7] an effective online feature selection method that removes low-variance features, retaining only those with variance above a specified threshold. This technique is used for feature selection in online machine learning tasks such as network intrusion detection [231]. A higher variance value indicates that a feature has a stronger association with the occurrence of intrusions, while a lower value suggests that the feature contributes little to distinguishing between normal and malicious traffic.

The proposed algorithm integrates five drift-adaptive online machine learning classifiers (ARF_EDDM, ARF_DDM, ARF_ADWIN, ARF_HDDM_A, and ARF_HDDM_W) to enable threat detection from real-time network traffic under dynamic streaming conditions. The model's workflow begins by capturing raw packets from external network streams, which are continuously transmitted to the system in the IP2V embedding format [34]. The system integrates IP2V, an incremental packet-to-vector mechanism, to extract features from raw network packets. It uses network

[7] https://riverml.xyz/dev/api/feature-selection/VarianceThreshold/.

fields as input to progressively transform discrete attributes into vector formats, enabling the packets to be processed effectively in the following detection stage. All models process each incoming instance to generate anomaly predictions along with corresponding probabilistic scores, which are then weighted inversely proportional to each model's error rate. A dynamically aggregated ensemble prediction is generated based on these weights. A self-adaptive model replacement mechanism monitors error thresholds (θ) and replaces the worst-performing model on the fly to maintain robustness. The error threshold θ is set to 0.5 in our implementation to enable efficient model adaptation. After processing the entire stream, the algorithm tracks performance evolution over time and computes baseline assessment metrics (Accuracy, Precision, Recall, F1-score, and ROC-AUC), ensuring real-time adaptability and resilience to concept drift.

The step-by-step procedure of the proposed drift-sensitive dynamic model adaptation with IP2V for real-time intrusion detection is presented in Algorithm 2.

Algorithm 2: Drift-sensitive dynamic model adaptation with IP2V (X_train, y_train, X_test, y_test, p, n, θ)

Symbols: ARF ← Adaptive Random Forest
IP2V ← Incremental Packet to Vector

Input: X_train, y_train (training data)
X_test, y_test (testing data)
p ← packet size
n ← number of ensemble models
θ ← model replacement threshold

Output: model's performance

Begin procedure:

Step 1. Initialize:
models ← []
for i ← 1 to n do
ARF_i with drift detector $D_i \in$ {EDDM, DDM, ADWIN, HDDM_A, HDDM_W}
append model. models
end for
metrics_list [] ← performance scores () for each model
global_performance scores ← performance scores ()
ε ← 0.001(small constant to avoid division by zero)
yt ← [], yp ← []

Step 2. All model's initial training over data stream:
for each instance (xi, yi) in (X_train, y_train) do
for each model in models, do
model.learn_one (xi, yi)

```
    end for
  end for
```

Step 3. Initialize incremental evaluation using IP2V:

```
for each (X_packet, y_packet) in IP2V_packet_stream (X_test, y_test,
p) do
   for each (xi, yi) in Stream (X_packet, y_packet) do
      predict_and_update (xi, yi)
       preds ← []
       probs ← []
       for i ← 1 to n do
          y_pred ← models [i]. predict_one (xi)
          y_prob ← models [i]. predict_proba_one (xi)
          model's [i]. learn_one (xi, yi)
          metrics_list [i]. update (yi, y_pred)
          append y_pred. preds
           append y_prob. probs
        end for
Evaluate errors and weights:
        errors ← [1 - metrics_list [i] for i = 1 to n]
        tot_weight ← Σ (1 / (error [i] + ε))
        weights ← [(1 / (error [i] + ε)) / tot_weight for each i]
Aggregate weighted probabilities for class 0 and 1
         y_pred_final = argmax (weighted_probs)
Dynamic model replacement concepts for week model:
         worst_idx ← index of max (errors)
          if errors [worst_idx] > θ then
             replace models [worst_idx] with a new model (ARF with
random Di)
             reset metrics_list [worst_idx] ← performance scores ()
           end if
     end for
   end for
```

Step 4. After processing all IP2V packet streams

```
return (performance scores (Accuracy, ROC-AUC, Precision, Recall,
F1 scores))
End procedure
```

6.1.2 Experimental Results

1. Description of the Benchmarks Dataset

In this research, the performance of all drift detection techniques is evaluated using three real-time benchmark datasets—CIC-IDS2017, UNSW-NB15, and IoTID20. Descriptions of these datasets, including details of attack types and packet-level characteristics for each category (as provided in Table 1.2), are presented in Chapter 1, Section 1.5.5.

2. Experimental Setup

In implementing the proposed methodology, we integrated robust drift detection models with the online adaptive learner, Adaptive Random Forest (ARF). The online drift-adaptive methods—ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, and ARF-HDDM_W—operate as an ensemble within the drift-sensitive dynamic model adaptation framework, incorporating IP2V for real-time intrusion detection. To evaluate the effectiveness of the proposed approach, we compare its performance against a range of established methods, including the ensemble models Bagging and AdaBoost; ENIDrift, a drift-aware ensemble for intrusion detection; KitNET, an unsupervised, autoencoder-based network intrusion detection system; IForestASD, an adaptive version of Isolation Forest for anomaly stream detection; HELAD, a lightweight online anomaly detection framework; and five base classifiers serving as individual benchmarks.

The experiments demonstrate that significant feature reduction using a variance threshold enhances the speed of network attack detection while maintaining high accuracy. For the CIC-IDS2017 dataset, the proposed model uses a variance threshold (p) of 0.1; for the UNSW-NB15 dataset, a threshold of 0.01 is applied; and for the IoTID20 dataset, a threshold of 0.1 is used to demonstrate enhanced performance. The performance of the models was evaluated through performance metrics such as Accuracy, ROC-AUC, Precision, Recall, and F1-score, using network intrusion detection datasets (CIC-IDS2017, UNSW_NB15, and IoTID20). To address the deviating accuracies, we employed the Isolation Forest anomaly detection model [232], which incorporates the evaluated accuracies of all online drift detection models. We aggregated the anomalous accuracy for various types of attacks across all datasets, as shown in Table 1.2 in Chapter 1. The overall experiments were conducted using the popular online machine learning library, River (ver. 0.9.0) [2], alongside the Anaconda

Python Jupyter platform version 8.15.0. Our PC specifications were used as described in an earlier chapter.

During the process, drifting models are classified as 0 for normal instances and 1 for non-anomalous cases. The models were trained with 10% of the samples and tested with 90% of the data. The model hyperparameters are set to their default values, with DDM configured to a warning level of 2.0 and an out-of-control level of 3.0. EDDM has a threshold warning zone of 0.95 and a drift zone of 0.90, while ADWIN is established with a confidence level of 0.002. HDDM_A and HDDM_W utilize drift confidence and warning confidence levels of 0.001 and 0.003, respectively, with both two-sided test values set to 'False' by default. The proposed methodology employs a dynamic model replacement threshold (θ) set to 0.5, a criterion for identifying underperforming models. A model is replaced only when its error exceeds this threshold value.

The ensemble techniques are configured with their default hyperparameters: ENIDrift uses a learning rate of 0.001, three hidden layers, and a hidden ratio of 'None'; KitNET uses a learning rate of 0.001, a hidden ratio of 0.9, and 50 autoencoders; IForestASD operates with a default window size of 1024; and HELAD uses a maximum autoencoder size of 10, a learning rate of 0.01, and a hidden ratio of 0.75. The Isolation Forest anomaly detection model sets the contamination rate range from (0.0 to 0.5)[8], depending on the proportion of outliers present in the datasets; in this proposed methodology, we set the contamination rate to 0.05 to effectively differentiate between anomalous and normal accuracies. The results of all three datasets (CIC-IDS2017, UNSW_NB15, and IoTID20) are presented in Tables 6.1 to 6.3. The overall performance is visualized through three categories of plots for all three datasets. The first category illustrates the visualizations of accuracy for all twelve models, including drift and anomaly detection. The second category features a histogram plot that displays the count of anomalous accuracies for various types of attacks for all models. In contrast, the third category presents a radar chart of performance metrics for all models. Plots for each dataset for all models are illustrated in Figs. 6.2 to 6.10.

[8] https://pyod.readthedocs.io/en/latest/pyod.models.html#pyod.models.iforest.IForest.

Table 6.1. Performance metrics of drift detection methods using the CIC-IDS2017 networking intrusion detection dataset.

Models	CIC-IDS2017 Dataset				
	Model's Performance Scores				
	Precision	**Recall**	**F1-score**	**ROC-AUC**	**Accuracy**
ARF-EDDM	0.9688	0.9586	0.9637	0.9761	98.76%
ARF-DDM	0.9570	0.9507	0.9538	0.9709	98.42%
ARF-ADWIN	0.9606	0.9582	0.9594	0.9750	98.61%
ARF-HDDM_A	0.9684	0.9532	0.9607	0.9734	98.66%
ARF-HDDM_W	0.9558	0.9593	0.9576	0.9751	98.54%
AdaBoost	0.8372	0.8682	0.8524	0.9166	94.83%
Bagging	0.9253	0.8775	0.9008	0.9314	96.68%
ENIDrift [34]	0.6656	0.2988	0.4125	0.6338	85.37%
KitNET [30]	0.8300	0.9300	0.8800	0.5800	78.00%
IForestASD [32]	0.9100	0.7700	0.8300	0.7000	74.00%
HELAD [35]	0.8256	0.9539	0.8851	0.5646	79.50%
Proposed Model	**0.9870**	**0.9698**	**0.9783**	**0.9836**	**99.26%**

Table 6.2. Performance metrics of drift detection methods using UNSW_NB15 networking intrusion detection dataset.

Models	UNSW_NB15 Dataset				
	Model's Performance Scores				
	Precision	**Recall**	**F1-score**	**ROC-AUC**	**Accuracy**
ARF-EDDM	0.9269	0.8737	0.8995	0.9359	99.48%
ARF-DDM	0.9685	0.8890	0.9270	0.9441	99.63%
ARF-ADWIN	0.9553	0.8991	0.9263	0.9490	99.62%
ARF-HDDM_A	0.9639	0.8793	0.9197	0.9392	99.59%
ARF-HDDM_W	0.9610	0.9175	0.9387	0.9583	99.68%
AdaBoost	0.9271	0.9149	0.9210	0.9565	99.58%
Bagging	**0.9964**	0.8535	0.9194	0.9267	99.60%
ENIDrift [34]	0.9722	0.7061	0.8181	0.8528	99.17%
KitNET [30]	0.9700	0.6700	0.7900	0.6800	66.00%
IForestASD [32]	0.9900	0.8400	0.9100	0.8400	84.00%
HELAD [35]	0.9942	0.8897	0.9390	0.8889	88.76%
Proposed Model	0.9887	**0.9237**	**0.9551**	**0.9617**	**99.77%**

Table 6.3. Performance metrics of drift detection methods using IoTID20 networking intrusion detection dataset.

Models	IoTID20 Dataset				
	Model's Performance Scores				
	Precision	**Recall**	**F1-score**	**ROC-AUC**	**Accuracy**
ARF-EDDM	0.9932	0.9990	0.9961	0.9495	99.27%
ARF-DDM	0.9906	0.9991	0.9948	0.9305	99.03%
ARF-ADWIN	0.9878	0.9973	0.9925	0.9085	98.59%
ARF-HDDM_A	0.9924	0.9992	0.9958	0.9434	99.20%
ARF-HDDM_W	0.9924	0.9992	0.9958	0.9434	99.20%
AdaBoost	0.9950	0.9914	0.9932	0.9595	98.73%
Bagging	0.9620	0.9900	0.9758	0.7092	95.40%
ENIDrift [34]	0.9965	0.9890	0.9927	0.9692	98.65%
KitNET [30]	0.9400	0.9500	0.9500	0.5400	90.00%
IForestASD [32]	0.9900	0.9000	0.9400	0.9000	90.00%
HELAD [35]	**1.0000**	0.9821	0.9910	0.7600	98.21%
Proposed Model	0.9959	**0.9998**	**0.9979**	**0.9697**	**99.60%**

3. Discussions on Results

First, we assessed the performance of various models on the CIC-IDS2017 network intrusion detection dataset using standard evaluation metrics: Accuracy, ROC-AUC, Precision, Recall, and F1-score. The results are summarized in Table 6.1. The proposed model exhibited strong accuracy by effectively adapting to concept drifts typically present in real-time network traffic, achieving a peak intrusion detection accuracy of 99.26%. Other models also produced effective results, though lower than the proposed model: ARF-EDDM (98.76%), ARF-DDM (98.42%), ARF-ADWIN (98.61%), ARF-HDDM_A (98.66%), ARF-HDDM_W (98.54%), AdaBoost (95.54%), Bagging (96.69%), ENIDrift (85.20%), KitNET (78.00%), IForestASD (74.00%), and HELAD (79.50%). In addition to accuracy, the proposed model attained the top scores in other critical metrics for intrusion detection: ROC-AUC (0.9836), Precision (0.9870), Recall (0.9698), and F1-score (0.9783). ENIDrift showed comparatively lower performance with Precision (0.6656), Recall (0.2988), F1-score (0.4125), and ROC-AUC (0.6338). The remaining eleven state-of-the-art models delivered performance metrics close to those of the proposed model, reinforcing their effectiveness in detecting intrusions in dynamic network environments.

Table 6.2 presents the evaluation of all models on the UNSW-NB15 intrusion detection dataset, which represents a broader spectrum of modern attack types. The proposed model again outperformed others, achieving the highest Accuracy (99.77%), ROC-AUC (0.9617), Recall (0.9237), and F1-score (0.9551). Notably, the

Bagging model recorded the highest Precision (0.9964), indicating its reliability in reducing false positives in a network security context. The remaining models produced comparable results, reflecting consistent performance across different intrusion detection scenarios.

Table 6.3 illustrates the model performance on the IoTID20 dataset, specifically targeting intrusion detection in IoT-based networks. The proposed model led across all evaluation metrics: Accuracy (99.60%), ROC-AUC (0.9697), Precision (0.9961), Recall (0.9998), and F1-score (0.9979). Notably, the HELAD model recorded the highest Precision (1.0000). The proximity of the performance scores of other models further validates their effectiveness in handling network threats, especially within the IoT domain, where timely detection is critical.

5. Performance Analysis and Threat Visualizations

In the context of intrusion detection, maintaining high model performance is challenging due to different types of drift (sudden, gradual, incremental, and recurring shifts) in data distribution, and real-time drift commonly arises from concept drift, data imbalance, and sophisticated evolving network attacks, which can significantly degrade detection accuracy over time. These types of drift, along with corresponding accuracy fluctuations and anomaly patterns, are illustrated in Figs. 6.2–6.4 across the three benchmark network datasets: CIC-IDS2017, UNSW_NB15, and IoTID20. To address this issue, the proposed error-driven dynamic ensemble model dynamically adapts to drift patterns and sustains high accuracy in real-time detection scenarios. It outperforms state-of-the-art models with Accuracy scores of 99.26%, 99.77%, and 99.60% on the respective datasets. Furthermore, it achieves the highest ROC-AUC, F1 score, Precision, and Recall when tested against the adversarial attacks listed in Table 1.2 referred to in Chapter 1, highlighting its robustness and effectiveness in dynamic network environments.

The distribution of anomalies in network intrusion detection datasets is crucial, as it reflects various types of network attacks within the data streams. A higher anomaly count suggests increased intrusion activity, which may signal irregularities during model training. Since streaming-based drift adaptation models rely on a consistent, one-to-one flow of data samples for practical training, such irregularities can degrade detection performance. The corresponding anomalous accuracy is visualized using histogram plots in Figs. 6.5–6.7.

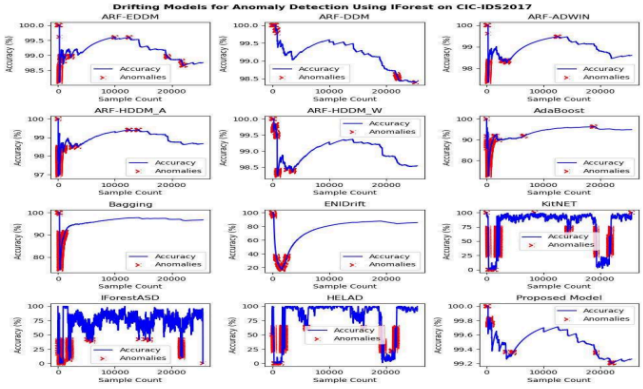


Fig. 6.2. Visualization of each model's accuracy, along with drift and anomalous values, for the CIC-IDS2017 dataset.

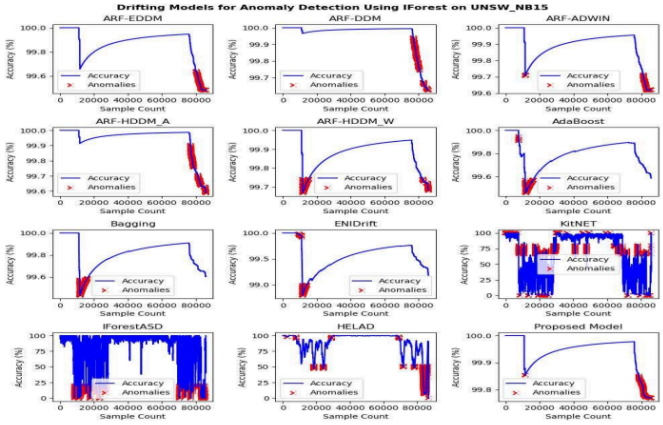


Fig. 6.3. Visualization of each model's accuracy, along with drift and anomalous values, for the UNSW_NB15 dataset.

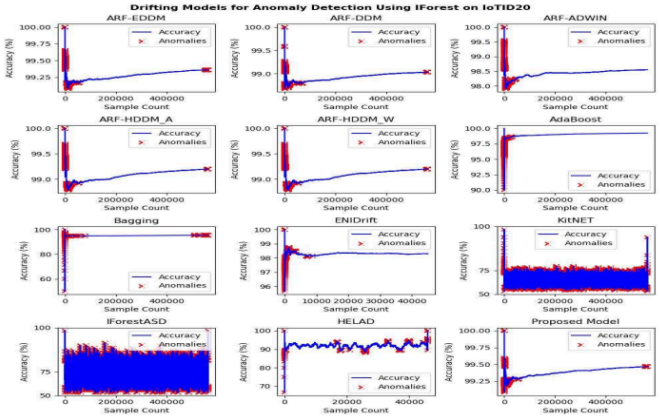


Fig. 6.4. Visualization of each model's accuracy, along with drift and anomalous values, for the IoTID20 dataset.

Figs. 6.2–6.4 presents the performance of various drift-handling models integrated with an isolation forest on the CIC-IDS2017, UNSW-NB15, and IoTID20 datasets for real-time network intrusion detection. The blue lines represent the evolving classification accuracy as the models process streaming network traffic, while the red markers indicate anomalies typically corresponding to malicious or suspicious activities detected by the isolation forest. These anomalies reflect deviations in model prediction behavior, often caused by cyber-attacks or shifts in network patterns. The proposed model consistently demonstrates superior performance by maintaining high detection accuracy and promptly identifying attack instances during both abrupt and gradual concept drifts. Other adaptive models, such as ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, and ARF-HDDM_W, also show competitive performance in detecting intrusions across changing traffic profiles. Notably, the proposed model exhibits strong drift resilience, quickly adapting to evolving threat landscapes while sustaining accuracies of 99.26% on CIC-IDS2017, 99.77% on UNSW-NB15, and 99.60% on IoTID20. In contrast, models like AdaBoost, Bagging, ENIDrift, KitNET, IForestASD, and HELAD respond slightly slowly to changes in network behavior for the CIC-IDS2017 dataset, resulting in higher false negatives and less effective intrusion detection. Overall, integrating the isolation forest enhances the transparency and reliability of the system by explicitly identifying anomalous prediction patterns linked to attack behaviors, while the proposed model offers robust, accurate, and real-time detection of threats in dynamic network environments.

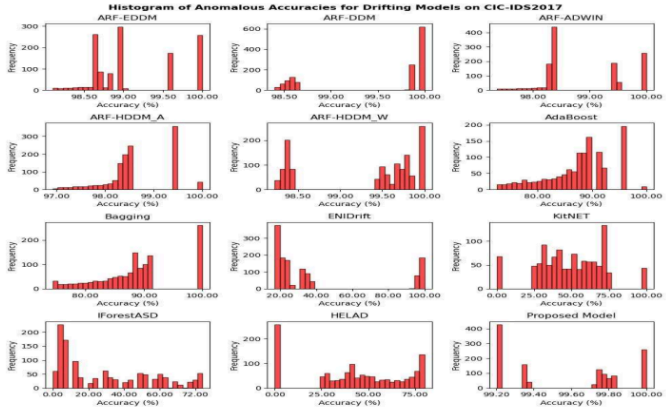


Fig. 6.5. Histogram showing the count of anomalous accuracy for the CIC-IDS2017 dataset.

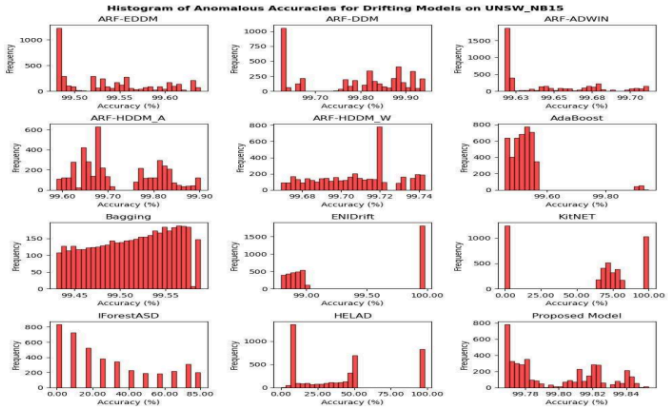


Fig. 6.6. Histogram showing the count of anomalous accuracy for the UNSW_NB15 dataset.

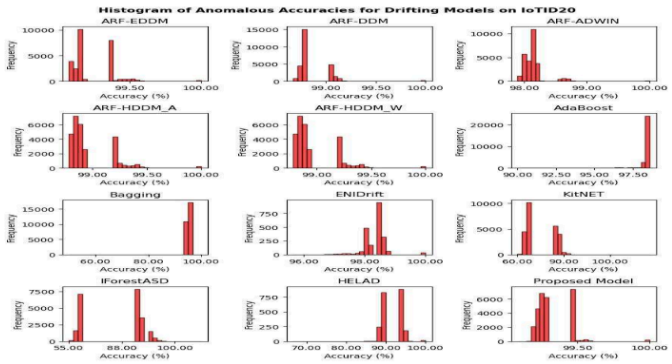


Fig. 6.7. Histogram showing the count of anomalous accuracy for the IoTID20 dataset.

Figs. 6.5–6.7 illustrate histograms of classification accuracies specifically for network attack instances under various drift-handling models on the CIC-IDS2017, UNSW_NB15, and IoTID20 datasets. Each histogram bin reflects how frequently attack samples were classified with a given accuracy level, thus revealing the stability and effectiveness of each model during adversarial conditions. The proposed model consistently shows concentrated and high-frequency peaks near the top accuracy ranges across all datasets, indicating highly precise detection of intrusions even during drift periods. In contrast, models like ENIDrift, Bagging, AdaBoost, KitNET, IForestASD, and HELAD show dispersed or bimodal distributions for the CIC-IDS2017 dataset but appear more stable for the UNSW_NB15 and IoTID20 datasets, with a substantial number of attack samples classified at lower accuracy intervals, implying delayed adaptation and vulnerability to drift-induced degradation. Models such as ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, and ARF-HDDM_W exhibit moderate performance with fragmented accuracy clusters. These findings underscore the superior resilience of the proposed model in maintaining accurate detection of malicious traffic under dynamic streaming conditions. The radar charts presented in Figs. 8 through 10 comprehensively illustrate the performance of each model across multiple evaluation metrics. These metrics include Accuracy, ROC-AUC, Precision, Recall, and F1 score, providing a multi-dimensional assessment of model effectiveness. The results are reported for three widely used network intrusion detection datasets: CIC-IDS2017, UNSW_NB15, and IoTID20. Together, these visualizations offer an in-depth comparison of how each model performs in detecting intrusions under different network traffic conditions and threat scenarios, highlighting strengths and potential areas for improvement.

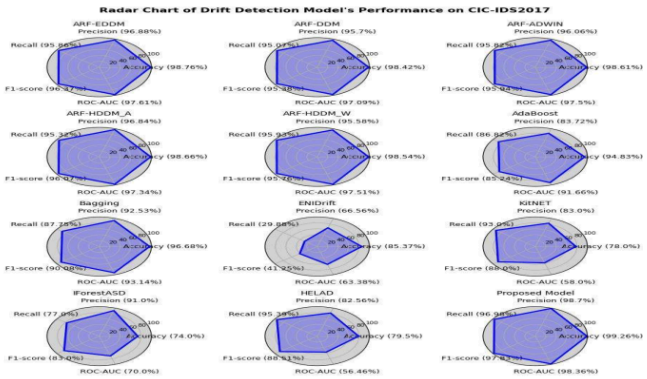


Fig. 6.8. Radar chart depicting the overall model-wise performance for the CIC-IDS2017 dataset.

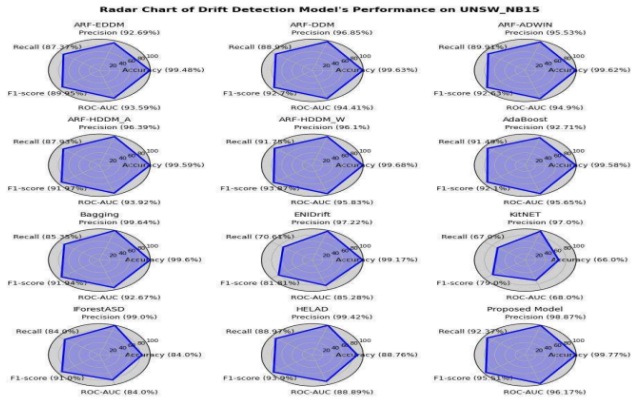


Fig. 6.9. Radar chart depicting the overall model-wise performance for the UNSW_NB15 dataset.

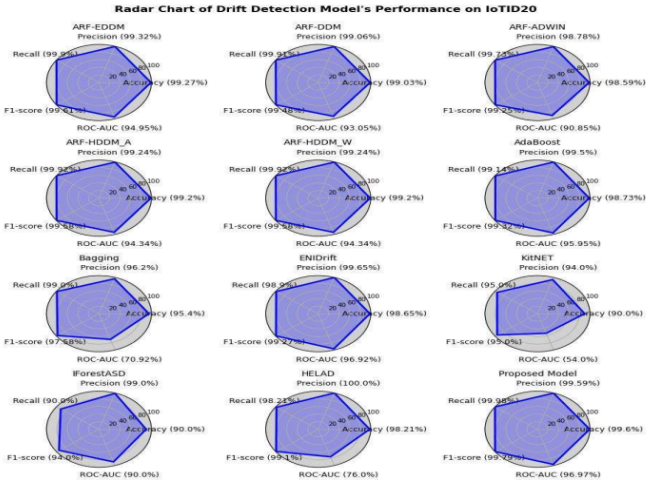


Fig. 6.10. Radar chart depicting the overall model-wise performance for the IoTID20 dataset.

6.2 Conclusion

In this chapter, we propose a methodology that employs five state-of-the-art online machine learning drift adaptive approaches (ARF-EDDM, ARF-DDM, ARF-ADWIN, ARF-HDDM_A, ARF-HDDM_W), which present a drift-sensitive dynamic model adaptation approach, and six renowned drift adaptive ensemble approaches (AdaBoost, Bagging, ENIDrift, KitNET, IForestASD, and HELAD) to address drift detection and adaptation alongside intrusions in network data streams. Generally, real-time drift commonly arises from concept drift, data imbalance, and sophisticated evolving network attacks. Three widely used benchmark network intrusion detection datasets (CIC-IDS2017, UNSW_NB15, and IoTID20) were employed in the implementation.

To assess performance, we evaluated five primary metrics: Accuracy, ROC-AUC, Precision, Recall, and F1-score. Accuracy serves as the key metric, representing the overall balanced performance rate. The Isolation Forest was used to detect anomalous accuracy fluctuations in each drift detection model's performance.

The proposed drift-sensitive dynamic model adaptation outperforms state-of-the-art models with accuracy improvements of up to 99.26%, 99.77%, and 99.60% on the CIC-IDS2017, UNSW_NB15, and IoTID20 datasets, respectively. Furthermore, it consistently outperforms existing approaches in terms of ROC-AUC, F1 score, Precision, and Recall, particularly in the presence of adversarial attacks across all three datasets. Future enhancements will integrate hybrid deep learning [234] and clustering-based models [235] with streaming NIDS to mitigate false positives and address encrypted traffic limitations [141, 233].

CHAPTER 7

CONCLUSION, FUTURE SCOPE, AND SOCIAL IMPACT

7.1 Conclusion

This thesis explores the key challenges associated with mining data streams in real-time environments and presents a series of systematic contributions aligned with the three primary objectives. These focus on ensemble learning, anomaly detection, and the evaluation of online learning strategies within dynamic and evolving datasets.

A core issue addressed in this research is concept drift adaptation and anomaly detection, which are critical for effective real-time data analysis. To tackle these challenges, an innovative error-weighted drift-adaptive ensemble framework was proposed. This framework integrates multiple drift detection algorithms with the Adaptive Random Forest (ARF), enabling it to adapt efficiently to both sudden and gradual changes in data distributions. Experimental results demonstrate that this proposed model surpasses other ensemble-based approaches in key performance metrics such as accuracy, precision, recall, and F1-score, despite incurring higher computational costs.

Another significant aspect of this work involved a comparative analysis of various online learning frameworks. The findings confirm the superiority of incremental learning solutions in handling the complexities of dynamic data streams. Notably, models built on the River framework consistently outperformed alternative implementations. Additionally, the study highlights the importance of evaluation

metrics like ROC-AUC, precision, recall, and F1-score over accuracy, particularly when dealing with imbalanced datasets.

The research also delved into anomaly detection within streaming data, proposing a stacked hybrid ensemble method that combines both supervised and unsupervised techniques. This approach demonstrated robust performance across multiple datasets, achieving high evaluation scores in identifying anomalies. A comprehensive survey conducted during the study identified several limitations in existing methods, emphasizing the need for more adaptable, accurate, and scalable solutions for anomaly detection in dynamic environments.

Furthermore, this work introduced an unsupervised online learning approach tailored for anomaly detection in IoT and environmental data streams. The results validated the effectiveness of streaming-based methods for identifying irregularities, though challenges such as concept drift, class imbalance, and resource constraints remain to be addressed.

Finally, a comprehensive framework was proposed for detecting concept drifts and anomalies in evolving network environments. This framework leverages drift-adaptive ensembles, the Isolation Forest algorithm, and incremental feature transformation techniques to enhance the detection of malicious activities. The proposed solution demonstrated high accuracy in identifying cyber threats across three major datasets: CIC-IDS2017, UNSW-NB15, and IoTID20, contributing valuable insights toward improving intrusion detection in streaming contexts.

7.2 Future Scope

The future research agenda aims to develop efficient and scalable computational techniques that leverage distributed and parallel processing technologies, alongside advanced intelligent modeling approaches capable of handling large-scale and high-velocity data streams. Despite the promising results achieved in this study, several areas offer opportunities for further enhancement:

- Optimization of computational processes within drift-adaptive ensemble models to minimize processing time and reduce memory usage, thereby improving their efficiency in handling rapid data streams.
- Development of novel methods for detecting subtle, rare, and evolving outliers within highly imbalanced data streams.

- Integration of deep learning architectures and hybrid techniques, such as combining neural networks with ensemble frameworks, to enhance detection accuracy and adaptability
- Automated parameter tuning using evolutionary algorithms to streamline model configuration and improve performance.
- Creation of frameworks capable of managing multivariate and high-dimensional data streams effectively.
- Advancement of multidimensional stream analysis techniques for comprehensive data interpretation.
- Improvement of concept drift detection and adaptation mechanisms to reduce system response latency when responding to changes in data distributions.
- Design of context-aware and adaptive anomaly detection systems that can dynamically adjust based on environmental and operational conditions.
- Incorporation of models within edge computing architectures to facilitate real-time processing at the network periphery.
- Enhancement of network intrusion detection systems' efficiency in working with encrypted traffic without compromising detection capabilities.

Exploring these avenues can significantly contribute to the ongoing evolution of streaming data analysis and intrusion detection technologies, paving the way for more robust, responsive, and scalable solutions.

7.3 Social Impact

The findings of this research hold significant practical implications across diverse sectors, including but not limited to:

- **Cybersecurity:** The proposed models can enhance existing intrusion detection systems, enable earlier identification of cyber threats, and strengthen overall network security.
- **Environmental Management:** These models facilitate real-time monitoring and analysis of environmental data, such as air quality metrics, supporting informed decision-making for environmental protection and sustainability initiatives.
- **Financial Sector:** Improved anomaly detection capabilities can help identify suspicious transactions and activities, thereby mitigating potential financial losses and preventing fraud.

- **Internet of Things (IoT):** The developed models are instrumental in processing streaming data generated by IoT devices, enabling automated decision-making and operational efficiency.
- **Industry Applications:** The adaptive AI systems emerging from this research could be deployed across various industries, fostering innovation and operational resilience through intelligent, real-time data analysis.

While the deployment of such models involves addressing challenges like data privacy and security, the importance of advancing intelligent, adaptive systems for real-time decision-making is undeniable. These systems have the potential to transform numerous domains by providing timely, accurate insights.

In conclusion, this research significantly contributes to the development of real-time, intelligent, and adaptive algorithms, paving the way for more responsive and autonomous systems capable of making informed decisions based on dynamic data analysis.

DESIGN AND ANALYSIS OF CLASSIFICATION FRAMEWORK FOR STREAMING DATA

ORIGINALITY REPORT

10%	5%	9%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	1%
2	Yang, Li. "Optimized and Automated Machine Learning Techniques Towards iot Data Analytics and Cybersecurity", The University of Western Ontario (Canada), 2022 Publication	1%
3	text-id.123dok.com Internet Source	<1%
4	B. G. Nagaraja, S. Kannadhasan. "Information and Communication Systems", CRC Press, 2026 Publication	<1%
5	Farah Jemili, Khaled Jouini, Ouajdi Korbaa. "Intrusion detection based on concept drift detection and online incremental learning", International Journal of Pervasive Computing and Communications, 2024 Publication	<1%
6	www.science.gov Internet Source	<1%
7	www.esrapublications.com Internet Source	<1%
8	cttewc.edu.in Internet Source	<1%
	download.bibis.ir	

9	Internet Source	<1%
10	"Machine Learning and Knowledge Discovery in Databases", Springer Science and Business Media LLC, 2021 Publication	<1%
11	arxiv.org Internet Source	<1%
12	Arvind Dagur, Sohit Agarwal, Dhirendra Kumar Shukla, Shabir Ali, Sandhya Sharma. "Artificial Intelligence and Sustainable Innovation - Volume 1", CRC Press, 2026 Publication	<1%
13	Arvind Dagur, Sohit Agarwal, Dhirendra Kumar Shukla, Shabir Ali, Sandhya Sharma. "Artificial Intelligence and Sustainable Innovation - Volume 3", CRC Press, 2026 Publication	<1%
14	"Machine Learning and Knowledge Discovery in Databases", Springer Science and Business Media LLC, 2016 Publication	<1%
15	Yogita, N.A., and Durga Toshniwal. "Variance and density-based anomaly identification and ranking for evolving data streams", International Journal of Computational Intelligence Studies, 2014. Publication	<1%
16	"Proceedings of 20th Iberian Conference on Information Systems and Technologies (CISTI 2025)", Springer Science and Business Media LLC, 2026 Publication	<1%

17	Alex Glyn-Davies, Mark Girolami. "Anomaly detection in streaming data with Gaussian process based stochastic differential equations", Pattern Recognition Letters, 2021 Publication	<1 %
18	Haiqing Yao, Xiuwen Fu, Yongsheng Yang, Octavian Postolache. "An Incremental Local Outlier Detection Method in the Data Stream", Applied Sciences, 2018 Publication	<1 %
19	Harvinder Singh, Priyanka Kaushal, Sarabjeet Kaur. "Advanced Computing and AI for Sustainable Environment, Energy, and Smart Manufacturing - AI-Driven Sustainability Across Environment, Energy, and Industry", CRC Press, 2026 Publication	<1 %
20	"Proceedings of the 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020)", Springer Science and Business Media LLC, 2021 Publication	<1 %
21	Suresh Ukarande, Shailesh R. Nikam, Makarand G. Kulkarni, Nandana A. Prabhu et al. "Technologies for Energy, Agriculture, and Healthcare - 2 International Conference on Technologies for Energy, Agriculture, and Healthcare (ICTEAH 2025)", CRC Press, 2026 Publication	<1 %
22	Huang, Chaoqin. "Unsupervised Deep Anomaly Detection and Its Application", National University of Singapore (Singapore), 2025 Publication	<1 %

23	Vithya Ganesan, Indu Vadhani Suresh, Subrata Chowdhury, Souvik Pal, Vishnu S. Pendyala. "Cognitive Fairness-Aware Techniques for Human–Machine Interface", CRC Press, 2025 Publication	<1 %
24	Oumayma Berraadi, Hicham Gibet Tani, Mohamed Ben Ahmed. "Enhancing Security and Privacy in IoT Data Streams: Real-Time Anomaly Detection for Threat Mitigation in Traffic Management", International Conference on Sustainable Computing and Green Technologies (SCGT'2025), 2025 Publication	<1 %
25	ojs.ninetyjournal.com Internet Source	<1 %
26	Luan Tran, Min Y. Mun, Cyrus Shahabi. "Real-time distance-based outlier detection in data streams", Proceedings of the VLDB Endowment, 2020 Publication	<1 %
27	Yam Sharon, David Berend, Yang Liu, Asaf Shabtai, Yuval Elovici. "TANTRA: Timing-Based Adversarial Network Traffic Reshaping Attack", IEEE Transactions on Information Forensics and Security, 2022 Publication	<1 %
28	shogun-toolbox.org Internet Source	<1 %
29	Mudassir Khan, Fahimuddin Shaik, Shaik Karimullah, Suman Lata Tripathi. "Integration of Internet of Things and Machine Learning - Design Models, Architectures and Application", Routledge, 2026 Publication	<1 %

30	www.coursehero.com Internet Source	<1 %
31	Boubiche Djallel Eddine, Sedat Akleylek. "Generative AI for Cybersecurity", CRC Press, 2026 Publication	<1 %
32	Nidhi Sehgal, Balamurugan Balusamy, Rachel Sheffield, Umer Zaman. "The Future of Business and Society - Integrating Sustainable Technologies with Data-Driven Solutions", CRC Press, 2026 Publication	<1 %
33	Srikanth Thudumu, Philip Branch, Jiong Jin, Jugdutt Singh. "A comprehensive survey of anomaly detection techniques for high dimensional big data", Journal of Big Data, 2020 Publication	<1 %
34	cwejournal.org Internet Source	<1 %
35	pmc.ncbi.nlm.nih.gov Internet Source	<1 %
36	Christoph Raab, Moritz Heusinger, Frank-Michael Schleif. "Reactive Soft Prototype Computing for Concept Drift Streams", Neurocomputing, 2020 Publication	<1 %
37	openaccess.city.ac.uk Internet Source	<1 %
38	Ashok N. Srivastava, Jiawei Han. "Machine Learning and Knowledge Discovery for Engineering Systems Health Management", CRC Press, 2016	<1 %

Publication

39	Attarha, Shadi. "A Framework for Sensor Fault Detection and Management in Low-Power IoT Edge Devices", Universitaet Bremen (Germany) Publication	<1 %
40	Ragini Gupta, Beitong Tian, Yaohui Wang, Klara Nahrstedt. "TWIN-ADAPT: Continuous Learning for Digital Twin-Enabled Online Anomaly Classification in IoT-Driven Smart Labs", Future Internet, 2024 Publication	<1 %
41	Markus Schneider, Wolfgang Ertel, Fabio Ramos. "Expected similarity estimation for large-scale batch and streaming anomaly detection", Machine Learning, 2016 Publication	<1 %
42	journalofcloudcomputing.springeropen.com Internet Source	<1 %
43	Asier Diaz-Iglesias, Antonio Nappa, Carmine Delle Femine, Piero Rivera et al. "Robust Concept Drift Handling in Dynamic Industrial Systems: A Multi-Objective Optimization Approach", Knowledge-Based Systems, 2025 Publication	<1 %
44	Mohamed Lahby, Al-Sakib Khan Pathan, Yassine Maleh. "Combatting Cyberbullying in Digital Media with Artificial Intelligence", CRC Press, 2023 Publication	<1 %
45	Qawasmeh, Saif Al-Dean Rushdi. "Using Machine Learning to Detect Insider Threats; Real-Time Behavior Analysis and Risk	<1 %

	Assessment.", North Carolina Agricultural and Technical State University Publication	
46	businessdocbox.com Internet Source	<1 %
47	dspace.mist.ac.bd Internet Source	<1 %
48	webthesis.biblio.polito.it Internet Source	<1 %
49	Pramod R. Gunjal, Satish R. Jondhale, Jaime Lloret, Karishma Agrawal. "Internet of Things - Theory to Practice", CRC Press, 2024 Publication	<1 %
50	Samir Fenanir, Fouzi Semchedine. "Smart Intrusion Detection in IoT Edge Computing Using Federated Learning", Revue d'Intelligence Artificielle, 2023 Publication	<1 %
51	Sushil Kumar Singh, Rajendrasinh B. Jadeja, Ashish Khanna, Pushan Kumar Dutta, Bal Virdee. "Security Paradigms in 6G Smart Cities and IoT Ecosystems - Navigating the Future", CRC Press, 2025 Publication	<1 %
52	Yu Weng, Lei Liu. "A Collective Anomaly Detection Approach for Multidimensional Streams in Mobile Service Security", IEEE Access, 2019 Publication	<1 %
53	ia803409.us.archive.org Internet Source	<1 %
54	repository.iutoic-dhaka.edu Internet Source	<1 %

55	"Database and Expert Systems Applications", Springer Science and Business Media LLC, 2010 Publication	<1 %
56	Baby Syla L., Salim A.. "Sign language recognition of Malayalam alphabets using transfer learning", 2023 International Conference on Power, Instrumentation, Control and Computing (PICC), 2023 Publication	<1 %
57	Farah Jemili, Khaled Jouini, Ouajdi Korbaa. "Intrusion Detection based on Concept Drift Detection & Online Incremental Learning", Research Square Platform LLC, 2023 Publication	<1 %
58	Rui Wang, Jiayao Li. "VAHMSE: an efficient anomaly detection model based on variational autoencoder and heterogeneous multi-stacking ensemble learning", Applied Intelligence, 2025 Publication	<1 %
59	brightideas.houstontx.gov Internet Source	<1 %
60	dos Santos Conceição, Luís Pereira. "Using DINOv2 for Road Damage Anomaly Detection", Universidade do Porto (Portugal), 2025 Publication	<1 %
61	mafiadoc.com Internet Source	<1 %
62	Vivek Sunnapwar, Hariram Chavan, Namrata F. Ansari. "Advances in Science and Technology", CRC Press, 2026 Publication	<1 %

63	Wang, Jing Hua, Xin Xiang Zhao, Peng Jin, and Guo Yan Zhang. "An Optimized Pruning-Based Outlier Detecting Algorithm", Applied Mechanics and Materials, 2013. Publication	<1 %
64	"Pattern Recognition and Machine Intelligence", Springer Science and Business Media LLC, 2024 Publication	<1 %
65	Lecture Notes in Computer Science, 2010. Publication	<1 %
66	Sugandh Seth, Kuljit Kaur Chahal, Gurvinder Singh. "Concept Drift–Based Intrusion Detection For Evolving Data Stream Classification In IDS: Approaches And Comparative Study", The Computer Journal, 2024 Publication	<1 %
67	oa.upm.es Internet Source	<1 %
68	Liangchen Chen, Shu Gao, Xiufeng Cao. "Research on real-time outlier detection over big data streams", International Journal of Computers and Applications, 2017 Publication	<1 %
69	S.P. Jani, M. Adam Khan. "Applications of AI in Smart Technologies and Manufacturing", CRC Press, 2025 Publication	<1 %
70	Varun Chandola. "Anomaly detection", ACM Computing Surveys, 07/01/2009 Publication	<1 %
71	Zhibo Zhou, Michael Walther, Alexander Verl. "Learning digital twin: a case study on chatter	<1 %

suppression based on a time-varying stability lobe diagram", The International Journal of Advanced Manufacturing Technology, 2025
Publication

72	kth.diva-portal.org Internet Source	<1 %
73	"Brain Storm Optimization Algorithms", Springer Science and Business Media LLC, 2019 Publication	<1 %
74	Ali Pesaranghader, Herna Viktor, Eric Paquet. "Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams", Machine Learning, 2018 Publication	<1 %
75	Lecture Notes in Computer Science, 2012. Publication	<1 %
76	Martin Fejrskov, Jens Myrup Pedersen, Emmanouil Vasilomanolakis. "Detecting DNS hijacking by using NetFlow data", 2022 IEEE Conference on Communications and Network Security (CNS), 2022 Publication	<1 %
77	Vikash Maheshwari, Nurul Aida Bt. Osman, Hanita Daud, Angelina Prima Kurniati, Wan Nur Syahidah Bt. Wan Yusoff. "A Drift-Oriented Adaptive Framework for Concept Drift Detection in Large-Scale Internet-of-Medical-Things Data Streams", Cold Spring Harbor Laboratory, 2024 Publication	<1 %
78	eprints.utm.my Internet Source	<1 %

79	medworm.com Internet Source	<1 %
80	sol.sbc.org.br Internet Source	<1 %
81	Quanyou Zhang, Yong Feng, A-Gen Qiu, Meng Yin, Yaohui Li, Delan Xiong, Chengshui Guo, Fangtao Qin. "Dynamic decision-making for inspecting the quality of treated sewage", Urban Climate, 2024 Publication	<1 %
82	Sixiang Wang, Fumio Machida. "A Robustness Evaluation of Concept Drift Detectors against Unreliable Data Streams", 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), 2021 Publication	<1 %
83	Color Image and Video Enhancement, 2015. Publication	<1 %
84	Jafseer KT, Shailesh S, Sreekumar A. "A Feature Evolution Aware Classification Framework for Streaming Data using Dynamic Autoencoder and Ensembled Learning.", JUCS - Journal of Universal Computer Science, 2025 Publication	<1 %
85	Rizwana Yasmeen, Lal Khan, Ahyoung Choi. "Heart disease prediction using hybrid TabNet architecture with stacked ensemble learning", Frontiers in Physiology, 2025 Publication	<1 %
86	Scott Wares, John Isaacs, Eyad Elyan. "Data stream mining: methods and challenges for handling concept drift", SN Applied Sciences, 2019 Publication	<1 %

87	Sotiris, Vasilis A., Peter W. Tse, and Michael G. Pecht. "Anomaly Detection Through a Bayesian Support Vector Machine", IEEE Transactions on Reliability, 2010. Publication	<1 %
88	escholarship.org Internet Source	<1 %
89	moldstud.com Internet Source	<1 %
90	research.vit.ac.in Internet Source	<1 %
91	www.ijraset.com Internet Source	<1 %
92	Advances in Intelligent Systems and Computing, 2014. Publication	<1 %
93	John Longshore, Angela Cheatham. "Managing the Aerospace System with Lean Six Sigma - A Roadmap to Success", Routledge, 2025 Publication	<1 %
94	Martí, Luis, Nayat Sanchez-Pi, José Molina, and Ana Garcia. "Anomaly Detection Based on Sensor Data in Petroleum Industry Applications", Sensors, 2015. Publication	<1 %
95	Shengjie Xu, Yi Qian, Rose Qingyang Hu. "A Semi-Supervised Learning Approach for Network Anomaly Detection in Fog Computing", ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019 Publication	<1 %

96	Siddhartha Roy, Soumya Sen, Agostino Cortesi. "Intelligent Systems - Emerging Trends and Challenges in Deep Neuro-Fuzzy Models and Explainable Artificial Intelligence", CRC Press, 2026 Publication	<1 %
97	Ying Zhong, Wenqi Chen, Zhiliang Wang, Yifan Chen, Kai Wang, Yahui Li, Xia Yin, Xingang Shi, Jiahai Yang, Keqin Li. "HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning", Computer Networks, 2020 Publication	<1 %
98	huggingface.co Internet Source	<1 %
99	services.phaidra.univie.ac.at Internet Source	<1 %
100	www.proceedings.com Internet Source	<1 %
101	Ahmed A. Abd El-Latif, Mohammed A ElAffendi, Mohamed Ali AlShara, Yassine Maleh. "Cybersecurity, Cybercrimes, and Smart Emerging Technologies", CRC Press, 2025 Publication	<1 %
102	Paul Okafor, Talayeh Razzaghi. "Cost-sensitive online learning for control chart pattern recognition", Computers & Industrial Engineering, 2026 Publication	<1 %
103	download.atlantis-press.com Internet Source	<1 %
104	knowledge.lancashire.ac.uk Internet Source	

		<1 %
105	"Encyclopedia of Big Data Technologies", Springer Science and Business Media LLC, 2019 Publication	<1 %
106	Al-Agha, Ibrahim Khaled. "A Human-in-the-Loop Framework for Scalable and Interpretable Event Triaging in Large-Scale Systems", Louisiana Tech University Publication	<1 %
107	Lecture Notes in Computer Science, 2013. Publication	<1 %
108	Linjin Sun, Yangjian Ji, Mingrui Zhu, Fu Gu, Feng Dai, Ke Li. "A new predictive method supporting streaming data with hybrid recurring concept drifts in process industry", Computers & Industrial Engineering, 2021 Publication	<1 %
109	Sanjeev Kumar, Ravendra Singh, Mohammad Zubair Khan, Abdulfattah Noorwali. "Design of adaptive ensemble classifier for online sentiment analysis and opinion mining", PeerJ Computer Science, 2021 Publication	<1 %
110	Tamás Miseta, Attila Fodor, Ágnes Vathy-Fogarassy. "Surpassing early stopping: A novel correlation-based stopping criterion for neural networks", Neurocomputing, 2024 Publication	<1 %
111	Wenjie Cao, Xinjian Wang, Yaqing Shu, Huanhuan Li, Jingen Zhou, Zaili Yang. "An integrated method of advanced optimisation and adaptive ensemble learning for ship fuel	<1 %

consumption prediction", Transportation Research Part C: Emerging Technologies, 2026
Publication

112	da Costa Brito, Francisco. "Anomaly Detection in Cybersecurity Through Semi-Supervision", Universidade do Porto (Portugal), 2024 Publication	<1 %
113	patents.google.com Internet Source	<1 %
114	Ahmed Shoyeb Raihan, Farzana Islam, Zhichao Liu, Srinjoy Das, Imtiaz Ahmed. "Hybrid Temporal Modeling and Generative Augmentation for Imbalanced Multivariate Time Series Anomaly Detection in Engineering Systems", Journal of Computing and Information Science in Engineering, 2026 Publication	<1 %
115	Ditzler, Gregory, and Robi Polikar. "Incremental Learning of Concept Drift from Streaming Imbalanced Data", IEEE Transactions on Knowledge and Data Engineering, 2012. Publication	<1 %
116	Fei Hu, Qi Hao. "Intelligent Sensor Networks - The Integration of Sensor Networks, Signal Processing and Machine Learning", CRC Press, 2012 Publication	<1 %
117	Haq, Ijaz Ul. "From AutoML to Self-Supervised AI Foundation Models: Deep Learning for Time Series Anomaly Detection With Applications to Hydrological Monitoring and Quality Control.", The University of Vermont and State Agricultural College Publication	<1 %

118	Peng Zhao, Le-Wen Cai, Zhi-Hua Zhou. "Handling concept drift via model reuse", Machine Learning, 2019 Publication	<1 %
119	cdn.intechopen.com Internet Source	<1 %
120	opentransportationjournal.com Internet Source	<1 %
121	www.lsi.upc.edu Internet Source	<1 %
122	Danlei Li, Nirmal Nair, Kevin I-Kai Wang. "Chapter 6 Unsupervised Time Series Anomaly Detection for Edge Computing Applications: A Review", Springer Science and Business Media LLC, 2024 Publication	<1 %
123	Priyanga Dilini Talagala, Rob J. Hyndman, Kate Smith-Miles, Sevvandi Kandanaarachchi, Mario A. Muñoz. "Anomaly Detection in Streaming Nonstationary Temporal Data", Journal of Computational and Graphical Statistics, 2019 Publication	<1 %
124	Quang-Tu Doan, A. S. M. Kayes, Wenny Rahayu, Kinh Nguyen. "Integration of IoT Streaming Data With Efficient Indexing and Storage Optimization", IEEE Access, 2020 Publication	<1 %
125	S. Elamathi, K. Subrahmaniyan, M. Raju, C. Umamaheswari, P. Anandhi. "Performance of Non-Puddled Machine Transplanted Rice over Puddled Machine Transplanted Rice in Thanjavur District of Tamil Nadu", National Academy Science Letters, 2025	<1 %

Publication

126	dokumen.pub Internet Source	<1 %
127	etd.aau.edu.et Internet Source	<1 %
128	ir.library.nitw.ac.in:8080 Internet Source	<1 %
129	rdw.rowan.edu Internet Source	<1 %
130	"Data Mining and Machine Learning Applications", Wiley, 2022 Publication	<1 %
131	Lidia Kidane, Paul Townend, Thijs Metsch, Erik Elmroth. "Efficient retraining of machine learning algorithms in cloud management systems", Future Generation Computer Systems, 2026 Publication	<1 %
132	Mahmood Shakir Hammoodi, Frederic Stahl, Atta Badii. "Real-Time Feature Selection Technique with Concept Drift Detection using Adaptive Micro-Clusters for Data Stream Mining", Knowledge-Based Systems, 2018 Publication	<1 %
133	Parasteh, Sirvan. "New Probabilistic Approaches for Detecting and Evaluating Concept Drift in Data Streams", The University of Regina (Canada), 2025 Publication	<1 %
134	Yinan Guo, Zhiji Zheng, Jiayang Pu, Botao Jiao, Dunwei Gong, Shengxiang Yang. "Robust online active learning with cluster-based local	<1 %

drift detection for unbalanced imperfect data", Applied Soft Computing, 2024
Publication

135	arc.cct.ie Internet Source	<1 %
136	citizenside.com Internet Source	<1 %
137	dspace.ut.ee Internet Source	<1 %
138	escholarship.mcgill.ca Internet Source	<1 %
139	www.mowhs.gov.bt Internet Source	<1 %
140	www.omicsdi.org Internet Source	<1 %
141	Doan Quang Tu, A. S. M. Kayes, Wenny Rahayu, Kinh Nguyen. "Integration of IoT Streaming Data with Efficient Indexing and Storage Optimization", IEEE Access, 2020 Publication	<1 %
142	Kotsis, Alexios Iosif. "Improving Network Anomaly Detection Using Packet Symmetry Analysis with Machine Learning and Deep Learning Algorithms", The George Washington University, 2025 Publication	<1 %
143	Mingu Kwak, Seoung Bum Kim. "Unsupervised Abnormal Sensor Signal Detection with Channelwise Reconstruction Errors", IEEE Access, 2021 Publication	<1 %
144	Zhong Chen, Victor Sheng, Andrea Edwards, Kun Zhang. "An effective cost-sensitive sparse	<1 %

	online learning framework for imbalanced streaming data classification and its application to online anomaly detection", Knowledge and Information Systems, 2022 Publication	
145	api.deepai.org Internet Source	<1 %
146	dspace.library.uvic.ca Internet Source	<1 %
147	ebin.pub Internet Source	<1 %
148	iferp-in-docs.s3.ap-south-1.amazonaws.com Internet Source	<1 %
149	www.numberanalytics.com Internet Source	<1 %
150	www.researchgate.net Internet Source	<1 %
151	Abhishek Kumar, Ramesh C. Bansal, Deng Yan, Praveen Kumar. "Microgrid Handbook - Planning to Practices", Routledge, 2025 Publication	<1 %
152	Ali Pesaranghader, Herna L. Viktor, Eric Paquet. "McDiarmid Drift Detection Methods for Evolving Data Streams", 2018 International Joint Conference on Neural Networks (IJCNN), 2018 Publication	<1 %
153	Deepak Adhikari, Wei Jiang, Jinyu Zhan, Danda B. Rawat, Asmita Bhattarai. "Recent advances in anomaly detection in Internet of Things: Status, challenges, and perspectives", Computer Science Review, 2024 Publication	<1 %

154	Georgios Kambourakis, Marios Anagnostopoulos, Weizhi Meng, Peng Zhou. "Botnets - Architectures, Countermeasures, and Challenges", Routledge, 2019 Publication	<1 %
155	H L Gururaj, Francesco Flammini, V Ravi Kumar, N S Prema. "Recent Trends in Healthcare Innovation", CRC Press, 2025 Publication	<1 %
156	Ness, Steven. "The Orchive : A system for semi-automatic annotation and analysis of a large collection of bioacoustic recordings.", Proquest, 2014. Publication	<1 %
157	Roberto Souto Maior de Barros, Silas Garrido T. de Carvalho Santos. "A Large-scale Comparison of Concept Drift Detectors", Information Sciences, 2018 Publication	<1 %
158	Shuai Lyu, Dongmei Mo, Wai keung Wong. "REB: Reducing biases in representation for industrial anomaly detection", Knowledge-Based Systems, 2024 Publication	<1 %
159	allrounder.ai Internet Source	<1 %
160	apps.dtic.mil Internet Source	<1 %
161	etd.repository.ugm.ac.id Internet Source	<1 %
162	eudoxuspress.com Internet Source	<1 %
163	iris.cnr.it	

	Internet Source	<1%
164	link.springer.com Internet Source	<1%
165	nep.rea.gov.ng Internet Source	<1%
166	orca.cardiff.ac.uk Internet Source	<1%
167	utopia.duth.gr Internet Source	<1%

Exclude quotes	Off	Exclude matches	< 10 words
Exclude bibliography	On		

DESIGN AND ANALYSIS OF CLASSIFICATION FRAMEWORK FOR STREAMING DATA

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56

PAGE 57

PAGE 58

PAGE 59

PAGE 60

PAGE 61

PAGE 62

PAGE 63

PAGE 64

PAGE 65

PAGE 66

PAGE 67

PAGE 68

PAGE 69

PAGE 70

PAGE 71

PAGE 72

PAGE 73

PAGE 74

PAGE 75

PAGE 76

PAGE 77

PAGE 78

PAGE 79

PAGE 80

PAGE 81

PAGE 82

PAGE 83

PAGE 84

PAGE 85

PAGE 86

PAGE 87

PAGE 88

PAGE 89

PAGE 90

PAGE 91

PAGE 92

PAGE 93

PAGE 94

PAGE 95

PAGE 96

PAGE 97

PAGE 98

PAGE 99

PAGE 100

PAGE 101

PAGE 102

PAGE 103

PAGE 104

PAGE 105

PAGE 106

PAGE 107

PAGE 108

PAGE 109

PAGE 110

PAGE 111

PAGE 112

PAGE 113

PAGE 114

PAGE 115

PAGE 116

PAGE 117

PAGE 118

PAGE 119

PAGE 120

PAGE 121

PAGE 122

PAGE 123

PAGE 124

PAGE 125

PAGE 126

PAGE 127

PAGE 128

PAGE 129

PAGE 130

PAGE 131

PAGE 132

PAGE 133

PAGE 134

PAGE 135

PAGE 136

PAGE 137

PAGE 138

PAGE 139

PAGE 140

PAGE 141

PAGE 142

PAGE 143

PAGE 144

PAGE 145

PAGE 146

PAGE 147

PAGE 148

PAGE 149

PAGE 150

PAGE 151

PAGE 152

PAGE 153

PAGE 154

PAGE 155

PAGE 156

PAGE 157

PAGE 158

PAGE 159

PAGE 160

PAGE 161

PAGE 162