# Acknowledgements

First and foremost, I give my deep thanks to Allah for giving me the opportunity and the strength to accomplish this work.

I would like to thank my supervisors Dr. Hatem Mohammed and Dr. Amgad Monir for their help and support during my work for creating a very inspiring research environment. They have been helpful with background information and have continually encouraged me and helped me with comments on my work. They always had time to discuss new ideas and give feedback on early ideas and research problems.

I would like to thank my family who was a constant source of rising and supporting my spirit. Thanks go out especially to my father, my mother, my wife, and my brothers for support and encouragement.

Finally, special thanks to my faculty, department, and my colleagues.

*Thank you everyone,*

Ahmed Madkour

December 2024

# Contents

# List of Figures

# List of Tables

# List of Algorithms

CHAPTER

# 1

# Introduction

Governments and companies are producing vast streams of data and require effective data analytics and machine learning methods to assist in making predictions and decisions promptly. One crucial aspect is the machine learning pipeline, which involves training a prepared dataset to construct a model and subsequently utilizing this model to predict new instance outputs. As depicted in Fig. 1.1, the process entails fetching historical data from the database during the training phase to construct the machine learning model. Then, the system can input new instances from the database to predict the output.

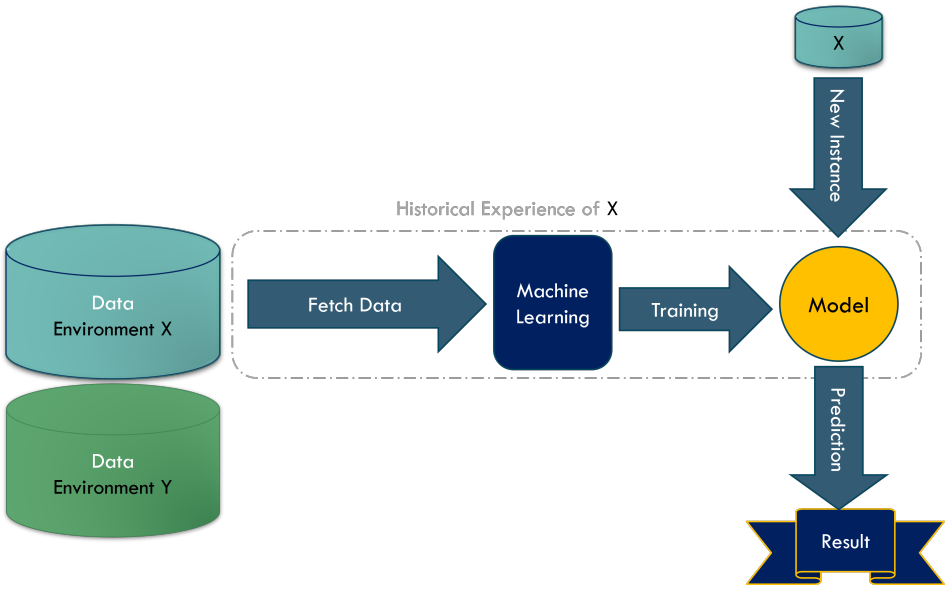


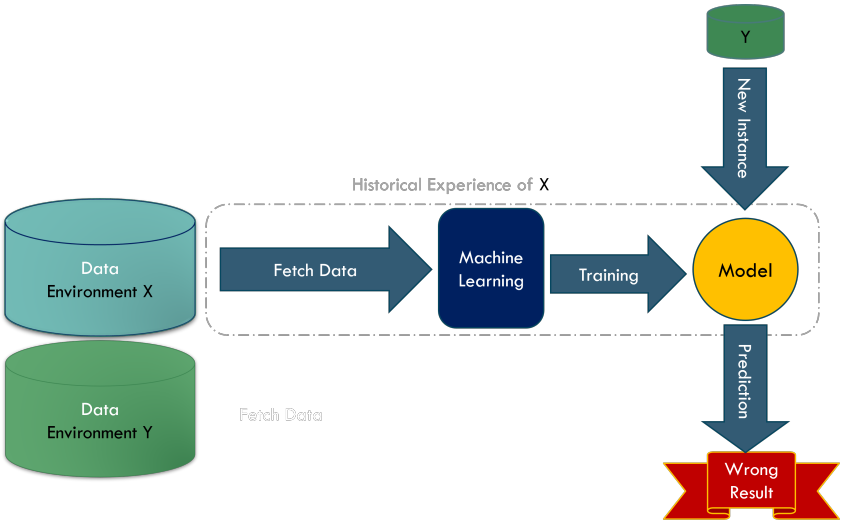

**Figure 1.1:** Machine learning workflow for environment X.

Nevertheless, when endeavoring to forecast outcomes for fresh instances sourced from an alternative database, as illustrated in Fig. 1.2 , there frequently emerges a conspicuous decline in accuracy. This disparity accentuates the imperative for model developers to intervene and rectify the issue. Addressing this, developers must adjust and retrain the model utilizing datasets from the new environment to ameliorate accuracy. This iterative process aims to refine the model's precision and ensure its efficacy across diverse contexts, thereby bolstering the reliability of decision-making and predictive capabilities. To confront this challenge, the field of auto machine learning endeavors to facilitate online updates to the model without necessitating direct intervention from developers for modification.

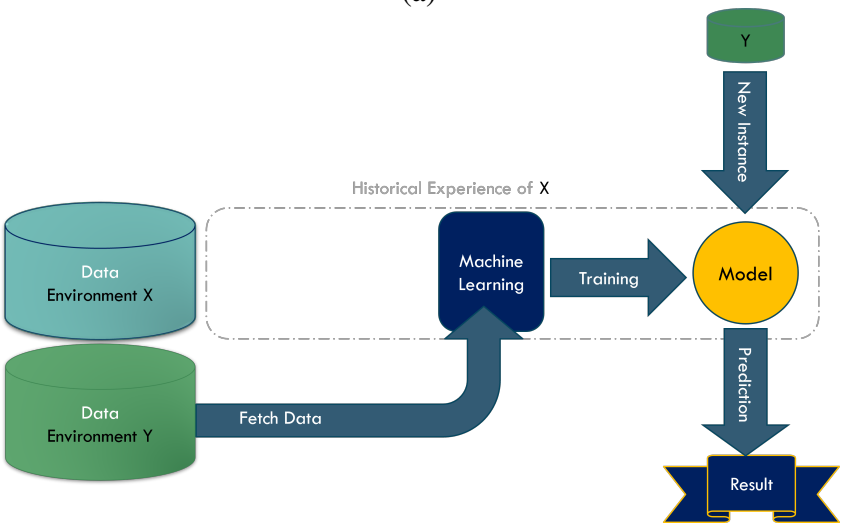
In recent years, the surge in high-speed data streams has posed notable challenges for machine learning models, particularly in the context of streaming data analysis. These data streams, characterized by continuous, dynamic, and high-volume data arrivals, demand adaptive learning algorithms that can effectively cope with their evolving nature [1–3]. Within these evolving data environments, two paramount challenges have emerged: concept drift, class imbalance, Emerging new class, and heterogenous transfer learning.

Concept drift, a phenomenon defined by the evolving statistical properties of a data generation process over time [4, 5]. introduces a dynamic element to the data, necessitating continuous adaptation of machine learning models. This shift can manifest as changes in underlying concepts, relationships between variables, or alterations in data distribution. Traditional models trained on historical data may suffer diminished accuracy or become inadequate when confronted with new data influenced by concept drift, highlighting the need for effective concept drift detection mechanisms. Addressing concept drift involves the utilization of concept drift detectors, which are methods capable of identifying changes in data stream distributions. These detectors rely on information related to classifier performance or incoming data items to signal the need for model updates, retraining, or even replacing the old model with a new one. The dynamic nature of concept drift necessitates ongoing monitoring and adaptation to maintain the model's efficacy.

Data streams also present challenges related to class imbalance, a condition characterized by uneven distribution among different classes [6, 7]. This scenario,




(a)




(b)

**Figure 1.2:** Machine learning workflow for environment Y.

especially prevalent in multi-class settings, poses a significant challenge for traditional classifiers. The risk of misclassifying minority class samples due to their limited representation demands specialized techniques to ensure accurate classification without sacrificing the performance of the majority class [8–11]. To tackle class imbalance, three primary methods are commonly employed: sampling methods, algorithm adaptation methods, and hybrid methods. Sampling methods involve un-

dersampling the majority class or oversampling the minority class to balance class distribution. Algorithm adaptation methods modify existing algorithms to handle imbalanced data [12–14], while hybrid methods combine data preprocessing with classification techniques, often utilizing ensemble classifiers to effectively mitigate class imbalance and enhance overall classifier performance [15–18].

Another challenge arising in the context of class imbalance is class overlap, where instances from different classes share the same region in data space [17, 19]. This overlap complicates the task of distinguishing between representative instances of different classes, leading to performance challenges for traditional classifiers referred to as overlapping problems. Recent research introduces class-overlap undersampling methods to address this issue, leveraging local similarities among minority instances to identify potentially overlapping majority instances.

Therefore, both class imbalance and class overlap present significant hurdles in the realm of data stream analysis. Consequently, addressing class imbalance has become crucial in multi-class learning, leading to research efforts focusing on both concept drift and class imbalance challenges. Researchers have explored dynamic ensemble selection (DES) and multi-class oversampling techniques to tackle these issues. Dynamic classifier ensembles offer a unique ability to adapt their composition based on data characteristics, making them valuable in situations with evolving data conditions [18]. Researchers focus on the overproduce-and-select approach for classifier ensemble selection methods. The objective of classifier ensemble selection is to choose an optimal subset of classifiers from a larger ensemble. The selection process is guided by various criteria, including individual performance measures, diversity metrics, meta-learning techniques, and performance estimation approaches. This optimization is particularly important in scenarios where a balance between accuracy and computational resource constraints is critical. There are two distinct approaches: static and dynamic selection. Static selection involves assigning classifiers to predefined feature partitions, while dynamic selection adaptively selects classifiers based on their competency [20]. Dynamic selection offers two choices: individual models, known as Dynamic Classifier Selection (DCS), and ensemble models, called Dynamic Ensemble Selection (DES). DCS algorithms enable the selection of the most appropriate classifier for each data point based on

its local competencies. In contrast, DES focuses on selecting the optimal classifiers for each instance based on their competence within localized regions [21–23] Competency assessment relies on a dynamic selection dataset (DSEL) containing labeled samples. Moreover, innovative techniques like the Randomized Reference Classifier introduce randomness into class supports to enhance adaptability in addressing challenges related to imbalanced data.

Additionally, transfer learning assumes a pivotal role in addressing the intricate challenges posed by dynamic data streams and inherent concept drift. This domain of research focuses on enhancing a model's learning performance within a target domain by harnessing knowledge gleaned from source domains [4, 6].Techniques in transfer learning include reducing domain gaps through instance re-weighting and feature matching, along with strategies to mitigate negative knowledge transfer by down-weighting irrelevant source data.

Lastly, in the study, the focus extends to the specific scenario of Streams with Emerging New Classes (SENC). This refers to situations where new classes, not present during the initial training of a learning model, emerge in the data stream. Traditional learning approaches, designed for fixed or predefined class distributions, face challenges in effectively recognizing and adapting to these novel classes in real-time. The need for adaptive learning mechanisms that can handle the emergence of new classes underscores the complexity of real-world data stream scenarios.

In this chapter, the challages for this research that naturally arise is discussed in Section 1.1. After this, the the motivation is presented in Section 1.2 . After this, the objectives and research questions are presented in Sections 1.3 and 1.4, respectively. Next, the research contribution is summarised in Section 1.5. Finally, the outline of this thesis is presented in 1.6, respectively.

## 1.1 Challenges

Within the swiftly evolving realm of machine learning, the proliferation of high-speed data streams presents a significant hurdle — the proficient management of non-stationary data environments. This challenge is marked by dynamic fluctuations in statistical attributes, fundamental concepts, and data distributions over

time. The crux of the issue emerges as models, initially trained on historical data, experience a decline in accuracy when confronted with new data influenced by concept drift. This includes scenarios such as:

- Multi-class imbalanced data streams
- Overlapping classes
- Emergence of new classes
- Heterogeneous transfer learning

However, addressing these challenges yields substantial benefits. By effectively managing non-stationary data environments, organizations can enhance the adaptability and resilience of their machine learning systems. This enables more robust decision-making processes, improved predictive capabilities, and heightened performance across diverse contexts. Moreover, it fosters innovation and agility, empowering organizations to stay ahead in dynamic markets and evolving scenarios.

## 1.2 Thesis Motivations

The rapid advancements in machine learning, particularly in the realm of real-time data streams, have ushered in a new era of challenges that demand innovative solutions. The primary motivations driving this thesis stem from the necessity to overcome the limitations of traditional models and methods when faced with dynamic, non-stationary data environments. These motivations are outlined as follows:

- **Adapting to Emerging Classes in Real-Time Scenarios:** In today's fast-paced data-driven world, the sudden emergence of new classes within data streams presents a critical challenge. Existing models often falter when confronted with such unforeseen changes, leading to significant declines in predictive accuracy. Our motivation lies in developing robust, real-time adaptive mechanisms that enable models to swiftly and effectively accommodate new classes, ensuring that the system remains relevant and accurate despite the evolving data landscape.

- **Proactive Management of Concept Drift:** As data streams evolve, the underlying concepts and statistical properties often shift, resulting in concept drift. This phenomenon can severely degrade the performance of models if not addressed promptly. Our research is motivated by the need to create sophisticated strategies that proactively detect and manage concept drift, ensuring that models can maintain high accuracy and adapt to changes in the data environment over time.

- **Dynamic Optimization of Classifier Ensembles:** The dynamic nature of non-stationary data streams necessitates the continuous optimization of classifier ensembles to handle emerging classes and shifting data distributions effectively. We are motivated to pioneer techniques that enable the real-time adjustment of classifier ensembles, thereby enhancing the overall performance of classification algorithms in evolving contexts.

- **Addressing Multi-Class Imbalance in Streaming Data:** Imbalanced data distributions, especially in multi-class scenarios, pose significant challenges for accurate classification. Traditional approaches often fail to adequately represent minority classes, leading to biased outcomes. Our motivation is to develop innovative methods that can dynamically address class imbalances, ensuring fair and accurate classification across all classes, even in non-stationary environments.

- **Enhancing Transfer Learning in Non-Stationary Contexts:** Transfer learning has emerged as a powerful technique for leveraging knowledge from diverse source domains. However, in non-stationary environments, the risk of negative knowledge transfer can undermine model performance. Our research is driven by the need to create frameworks that facilitate effective transfer learning, minimizing negative transfer and maximizing the potential for positive knowledge transfer, thereby ensuring robust model performance across diverse and shifting data landscapes

## 1.3 Thesis Objectives

The thesis objectives for our research stems from the imperative need to address the following key motivations:

- **Objective 1**. Handling Imbalanced Multiclass Drifted Data and overlapping classes streams.
- **Objective 2**. Addressing Emerging New Classes in Incremental Streams via Concept Drift and K-means Techniques.
- **Objective 3**. Addressing Heterogeneous Transfer Learning Problem in data streams via Concept Drift and Eigenvector Techniques.

## 1.4 Thesis Questions

- $Q_1$. What is the impact of imbalanced stream on the performance of ML models in non-stationary Environment?
- $Q_2$. What is the impact of reducing overlapping classes stream on the performance of ML models in non-stationary environment?
- $Q_3$.How does the emergence of new classes in data streams affect the stability and performance of ML models?
- $Q_4$. How can concept drift be utilized to address the challenge of emerging new classes?
- $Q_5$. How does the Heterogeneous sources in data streams affect the stability and performance of Transfer Learning Technique?
- $Q_6$. What approaches or techniques can be employed in heterogeneous transfer learning to facilitate knowledge transfer across diverse sources and domains?

## 1.5 Contributions

Our research endeavors to create advanced frameworks designed for effectively managing non-stationary data streams while prioritizing high accuracy and mini-

mizing computational complexity. The key contributions of our work can be summarized as follows:

- **Incorporation of Concept Drift Detection and Ensemble Classifier:** Our primary innovation involves incorporating a concept drift detection method alongside an ensemble classifier. This integration enables real-time adaptation and refinement of our proposed framework in response to transfer learning in non-stationary environments. This methodology ensures the continuous evolution of our classification model in alignment with the changing data landscape.

- **Dynamic Classifier Ensemble Framework for Emergence Class Problem:** We introduce a classification framework that harnesses dynamic classifier ensemble techniques to address the emergence class problem. This framework enhances the performance of classification algorithms when dealing with non-stationary data streams featuring emerging new classes. By dynamically adjusting the ensemble based on data characteristics, our framework improves the accuracy and effectiveness of classification models, effectively tackling challenges associated with emerging new classes.

- **Introduction of Precise Weighting Method for Local Classifiers of the transfer learning framework:** A significant advancement in our research is the introduction of a precise weighting method to assess the significance of each local classifier within the ultimate classifier. This method enhances the accuracy of the overall classifier by providing a nuanced evaluation of individual classifier contributions.

- **Development of Innovative Framework using Eigenvector Technique for addressing heterogenous transfer learning problem:** A significant stride in our research involves the creation of an innovative framework, harnessing the power of the eigenvector technique. This framework is specifically designed to tackle the challenges posed by heterogeneous transfer learning problems. By leveraging the eigenvector technique, our framework enables the seamless transfer of knowledge from diverse source domains to the target domain, thereby enhancing adaptability and overall performance.

- **Dynamic Adjustment Method to Multi-class Imbalanced Data:** Our classification framework introduces a dynamic adjustment method tailored for multi-class imbalanced data scenarios. It seamlessly incorporates concept drift detection mechanisms and optimizes classifier ensemble selections, aiming to significantly improve classification accuracy in the context of multi-class imbalanced non-stationary streams.
- **Adaptive Method for Class Imbalance:** Additionally, we propose an adaptive method for addressing the class imbalance issue. This method considers data distributions and historical instances of class imbalance, especially in cases where class overlap occurs within multi-class and drifted data streams. The adaptive approach improves classification performance by selecting the most suitable oversampling method based on the unique characteristics of the data stream.

## 1.6 Structure of the Dissertation

The structure of the remainder of this thesis dissertation is outlined below:

- **Chapter 2** reviews background about concept drift, concept drift types, concept drift components, adapting types.
- **Chapter 3** reviews state-of-the-art concept drift, classifier ensemble selection, imbalanced data streams, Streams with Emerging New Classes (SENC), and transfer learning.
- **Chapter 4** presents our first proposal to build an effective proposed approch for handling Imbalanced Multi-class Drifted Data streams.
- **Chapter 5** provides our second proposal to adressing emerging new classes in incremental streams via concept drift techniques.
- **Chapter 6** presents our third proposal to addressing heterogeneous transfer learning problem in incremental streams via concept drift techniques.
- **Chapter 7** revisits the main goal and specific objectives posted earlier. In this chapter, we summarise the main contributions of this thesis and outline possible future research.

CHAPTER

# 2

# Background

Amidst the surge of vast streaming data, governments and businesses find themselves in an urgent need for sophisticated data analysis and machine learning analytics approaches. These tools are indispensable for anticipating future trends and making well-informed decisions. However, the perpetual emergence of new goods, markets, and consumer behaviors introduces a formidable challenge known as concept drift [24]. This phenomenon involves the variation of statistical parameters of the target variable over time in unexpected ways, posing a substantial obstacle to accurate forecasting and optimal decision-making. The patterns derived from historical data may become obsolete when applied to new and evolving datasets. The impact of concept drift extends across data-driven information systems, including decision support and early warning systems, diminishing their overall effectiveness. In the dynamic realm of big data, where data types and distributions are inherently unpredictable, the challenge of concept drift becomes even more pronounced. In response to this challenge, the field introduces a new subject: adaptive data-driven prediction/decision systems.

## 2.1 Concept Drift Sources

Concept drift, illustrated comprehensively in Fig. 2.1, unfolds through three distinct scenarios. First, (a) portrays a shift in data distribution, signifying changes in the underlying patterns and characteristics of the incoming data. This type of drift challenges the model's ability to adapt to new trends and patterns [25–28].

Second, (b) showcases a change in function output, leading to the need for adjustments in the position of the class delimiter. Here, the relationship between input features and output classes undergoes transformations, demanding the model to realign its decision boundaries accordingly. Third, (c) presents a scenario involving a dual shift—both in data distribution and function output. This complex manifestation of concept drift requires the model to address simultaneous changes in underlying patterns and output relationships. Effectively navigating these shifts is crucial for maintaining the model's predictive accuracy and decision-making capabilities in dynamic environments. Understanding these nuanced aspects of concept drift is foundational for devising adaptive strategies in machine learning models.

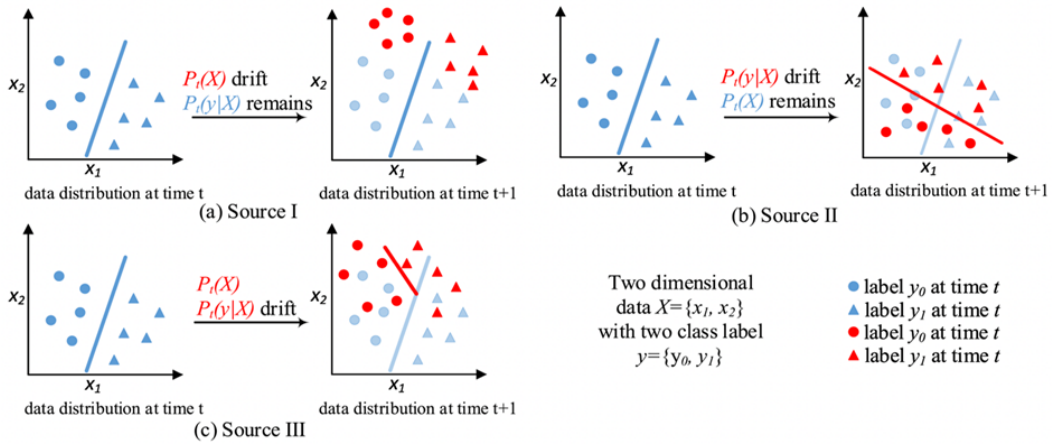



**Figure 2.1:** Sources of concept drift.

DOI: 10.1109/TKDE.2018.2876857

## 2.2 Concept Drift Types

Concept drift is commonly categorized into four types, as illustrated in Fig. 2.2. In Types 1-3, the focus of research on concept drift adaptation revolves around minimizing the drop in accuracy and achieving the fastest recovery rate during the transformation process of concepts. Conversely, Type 4 drift delves into leveraging historical concepts, emphasizing the identification of the best-matched historical concepts within the shortest time frame when a new concept emerges—whether suddenly, incrementally, or gradually. To illuminate the distinctions between these

types, [27] introduces the term ”intermediate concept” to describe the transitional phases between concepts. As noted by [11], concept drift may not only occur at a precise timestamp but may also extend over a prolonged period. Consequently, intermediate concepts may emerge during the transformation, representing a blend of the starting and ending concepts in the case of incremental drift or embodying one of the starting or ending concepts, as observed in gradual drift. Understanding these intermediate concepts is essential for comprehending the nuanced dynamics of concept drift during transitions from one state to another.

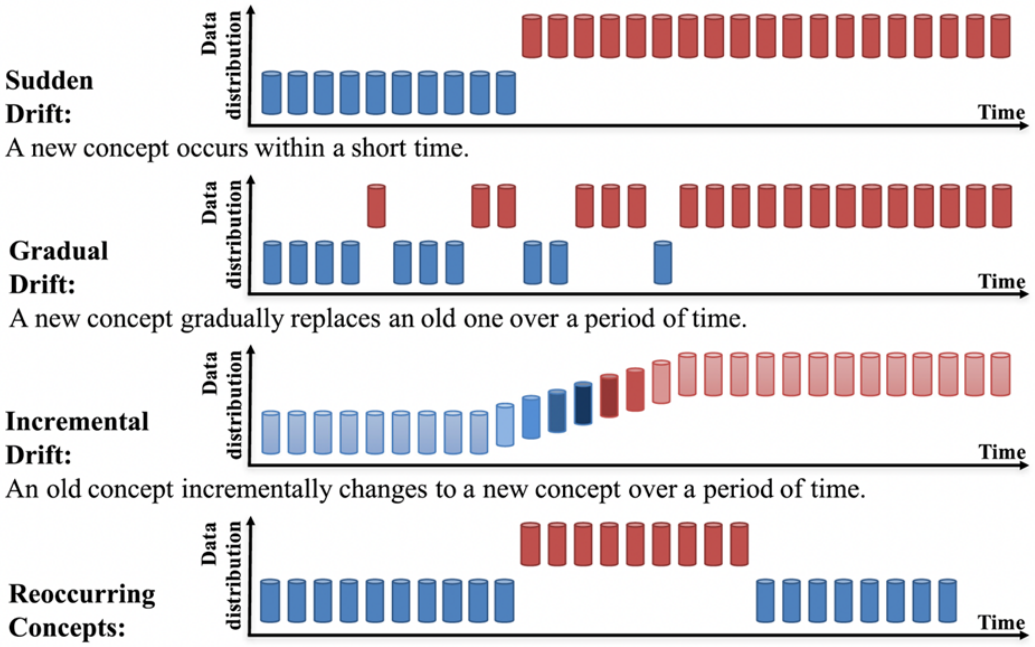



**Figure 2.2:** Types of concept drift.
DOI: 10.1109/TKDE.2018.2876857

## 2.3 Concept Drift Components

Conventional machine learning primarily involves prediction and training/learning. However, learning under the concept drift paradigm introduces three additional critical steps as show in Fig. 2.3: concept drift detection, drift understanding, and drift adaptation. Concept drift detection involves identifying changed points or

change time periods to define and predict drift. Drift understanding delves into crucial aspects such as when the drift starts, how long it lasts, and where it occurs, providing indispensable insights for the subsequent adaptation step. The adaptation step, also referred to as the reaction step, plays a pivotal role in updating current learning models in response to concept drift. Three main approaches address various types of drift: Simple Retraining, Ensemble Retraining, and Model Adjusting. Drift detection employs various tools and algorithms, comparing old and fresh data chunks with statistical models based on data distribution. Techniques vary, with some utilizing a constant chunk length and others employing a variable length. Drift understanding is essential for making well-informed decisions during the adaptation step. This involves calculating the necessary modifications in the trained model to adapt to new changes as shown in Fig. 2.4. The severity region determines whether to generate a completely new model or make minimal adjustments to the existing one.

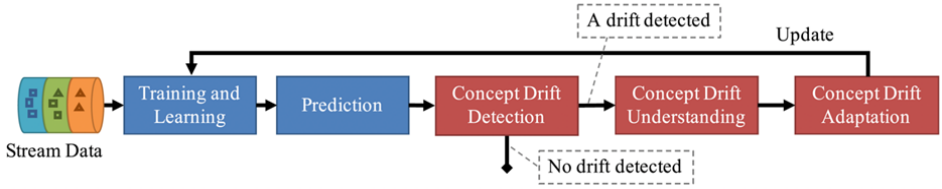



**Figure 2.3:** Main components of concept drift.

DOI: 10.1109/TKDE.2018.2876857

### 2.3.1 Concept Drift Detection

Drift detection involves techniques and mechanisms to characterize and quantify concept drift by identifying change points or intervals [11]. The general framework for drift detection consists of four stages:

- **Stage 1 (Data Retrieval):** This stage focuses on retrieving data chunks from data streams. Given that a single data instance lacks sufficient information to infer the overall distribution [25], organizing data chunks meaningfully is crucial for effective data stream analysis [29].

- **Stage 2 (Data Modeling):** This optional stage abstracts the retrieved data, extracting key features that contain sensitive information impacting a system in case of drift. This stage may involve dimensionality reduction or sample size reduction to meet storage and online speed requirements [11].

- **Stage 3 (Test Statistics Calculation):** This stage involves measuring dissimilarity or estimating distance to quantify drift severity and generate test statistics for hypothesis testing. Defining an accurate and robust dissimilarity measurement remains a challenging aspect of concept drift detection. Test statistics can also be used for clustering evaluation [30] and to determine dissimilarity between sample sets [31].

- **Stage 4 (Hypothesis Test):** This stage employs a specific hypothesis test to assess the statistical significance of the change observed in Stage 3, such as the p-value. These tests determine drift detection accuracy by establishing statistical bounds for the test statistics from Stage 3. Without Stage 4, the acquired test statistics are meaningless for drift detection, as they cannot establish the drift confidence interval. Commonly used hypothesis tests include estimating the distribution of test statistics [32, 33], bootstrapping [34, 35], the permutation test [25], and Hoeffding's inequality-based bound identification [36].

It is crucial to note that without Stage 1, the concept drift detection problem can be regarded as a two-sample test problem, examining whether the populations of two given sample sets are from the same distribution [31]. In other words, any multivariate two-sample test can be adopted in Stages 2-4 for detecting concept drift [31]. However, in cases where distribution drift may not be included in the target features, the selection of the target feature becomes critical for the overall performance of a learning system and poses a significant challenge in concept drift detection [37].

### 2.3.2 Understanding Phase

The extent of concept drift severity serves as a valuable criterion for selecting appropriate drift adaptation strategies. As shown in Fig. 2.4, in a classification task

where the drift's severity is minimal, resulting in only a marginal shift in the decision boundary within the new concept, adjusting the current learner through incremental learning proves sufficient. Conversely, when confronted with high severity in concept drift, wherein the decision boundary undergoes substantial changes, it might be more effective to discard the old learner and opt for retraining a new one rather than incrementally updating the existing learner. It's noteworthy to mention that despite some researchers highlighting the capability to articulate and quantify the severity of detected drift, this information is not yet widely integrated into drift adaptation practices. The adaptation step offers three distinct ways to adapt the model. Simple Retraining involves training a new model using the most recent data, replacing the old model. Model Ensemble, the second approach, entails keeping and reusing existing models, which proves efficient when dealing with repeated instances of concept drift. The third approach, Model Adjusting, constructs a model that adapts flexibly from changed data, allowing partial updates when the original data distribution undergoes significant changes.

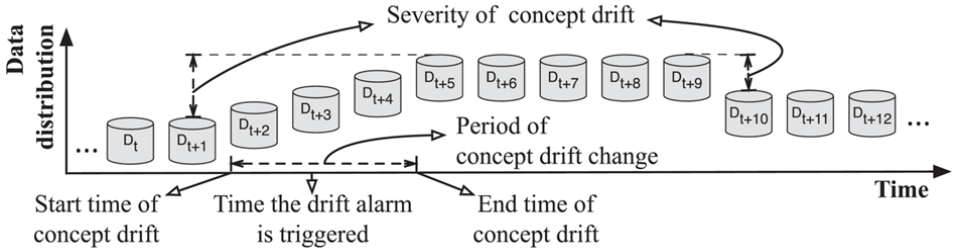



**Figure 2.4:** Understanding phase of concept drift.

DOI: 10.1109/TKDE.2018.2876857

### 2.3.3 Adaptation Phase

This section delves into strategies for updating existing learning models in response to drift, referred to as drift adaptation or reaction. The three primary categories of drift adaptation methods are simple retraining, ensemble retraining, and model adjusting, each tailored to address specific types of drift.

A. **Simple Retraining**

One way to respond to concept drift is by retraining a new model with the

latest data, replacing the outdated model as shown in Fig.2.5. This approach requires an explicit concept drift detector to determine when to retrain the current model. A window strategy is commonly used, preserving recent data for retraining and/or utilizing old data for distribution change tests. An example of this strategy is Paired Learners [38], which employs two learners: the stable learner and the reactive learner. If the stable learner consistently misclassifies instances correctly classified by the reactive learner, indicating a new concept, the stable learner is replaced with the reactive learner. This method is straightforward, easy to implement, and adaptable at any point in the data stream. However, a trade-off arises when adopting a window-based strategy in determining an appropriate window size. A small window better reflects the latest data distribution, while a large window provides more data for training a new model. To address this challenge, the ADWIN algorithm [39] dynamically adjusts sub window sizes based on the rate of change between two sub-windows, eliminating the need for users to predefine window sizes. Beyond direct model retraining, researchers have explored integrating the drift detection process with the retraining process for specific machine learning algorithms. DELM [40], for instance, extends the traditional ELM algorithm, adapting to concept drift by dynamically adjusting the number of hidden layer nodes in response to increasing classification error rates, indicating a potential concept drift. Similarly, FP-ELM [41] introduces a forgetting parameter to the ELM model to adapt to drift conditions. A parallel version of the ELM-based method [42] has been developed for high-speed classification tasks under concept drift. OS-ELM [43], an online learning ensemble of repressor models, integrates ELM using an ordered aggregation (OA) technique to address the challenge of defining the optimal ensemble size. In the realm of instance-based lazy learners for handling concept drift, the Just-in-Time adaptive classifier [32] follows the ”detect and update model” strategy, extending the traditional CUSUM test [44] to a pdf-free form for drift detection. When a concept drift is identified, old instances (beyond the last T samples) are removed from the case base. Advancements include extending this algorithm to handle recurrent concepts by considering and comparing the current concept to previously stored concepts [30, 32]. NEFCS [25], another

KNN-based adaptive model, employs a competence model-based drift detection algorithm [25] to locate drift instances in the case base and distinguish them from noise instances. The redundancy removal algorithm, Stepwise Redundancy Removal (SRR), is developed to eliminate redundant instances uniformly, ensuring that the reduced case base retains sufficient information for future drift detection.

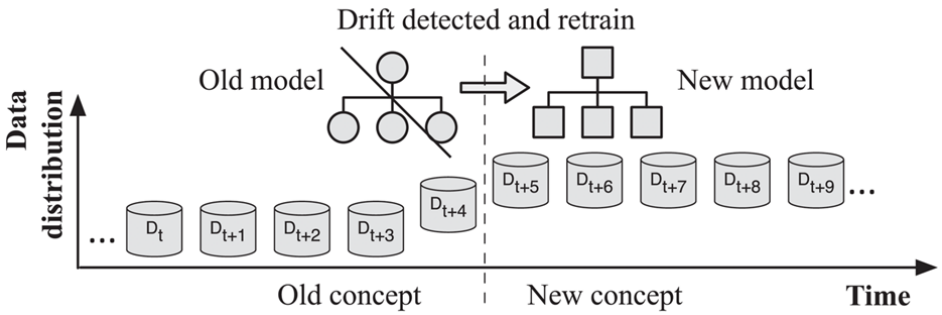



**Figure 2.5:** Approach for retraining a new model.
DOI: 10.1109/TKDE.2018.2876857

B. **Model Ensemble Eetraining**

In recurring concept drift scenarios, preserving and reusing old models can be more efficient than retraining new ones for each recurrence, forming the basis for employing ensemble methods [45]. These methods, a focus in stream data mining research, consist of base classifiers with varied types or parameters. Their outputs, determined by specific voting rules, collectively predict new data. Various adaptive ensemble methods, extending classical ones or introducing adaptive voting rules, address the challenges of concept drift as shown in Fig. 2.6. Classical ensemble methods like Bagging, Boosting, and Random Forests have been adapted for streaming data with concept drift. For instance, online Bagging [46] uses each instance once, simulating batch mode bagging. Leveraging Bagging [47] employs ADWIN drift detection to replace the existing classifier with the worst performance when concept drift is detected. Adaptive boosting [48], monitoring prediction accuracy through a hypothesis test, addresses concept drift, assuming classification errors on non-drifting data follow a Gaussian distribution. The

Adaptive Random Forest (ARF) algorithm [49] extends the random forest tree algorithm, incorporating concept drift detection (e.g., ADWIN) to decide when to replace an obsolete tree. A similar approach is seen in [50], using Beyond classical methods, novel ensemble methods with innovative voting techniques tackle concept drift. Dynamic Weighted Majority (DWM) [51] adapts to drifts through weighted voting rules, managing base classifiers based on individual and global ensemble performance. Learn++NSE [52] addresses frequent classifier additions by weighting them based on prediction error rates. Specific types of concept drift are considered in specialized ensemble methods. Accuracy Update Ensemble (AUE2) [53] equally addresses sudden and gradual drift, using a batch mode weighted voting ensemble method. Optimal Weights Adjustment (OWA) [54] achieves a similar goal with weighted instances and classifiers. Special cases, like class evolution, are considered in [55], while recurring concepts are handled by monitoring concept information [26, 56]. Another method [57], refines the concept pool for recurring concepts.

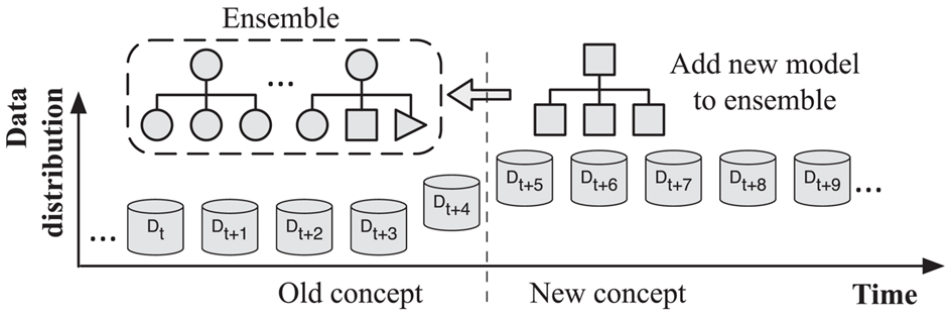



**Figure 2.6:** Ensemble approach for the adaptation phase.
DOI: 10.1109/TKDE.2018.2876857

C. **Model Adjusting**

As Shown in Fig. 2.7, instead of retraining the entire model, an alternative is to construct a model with adaptive learning capabilities, allowing partial updates in response to changing data distributions [58], as in Fig. 2.7. This is efficient when concept drift occurs in localized regions. Many techniques in this category use the decision tree algorithm, leveraging its ability to adapt

to individual sub-regions. VFDT [59] is a foundational contribution for high-speed data streams, employing the Hoeffding bound for node splitting. VFDT processes each instance once, doesn't store instances, and has minimal maintenance costs. CVFDT [60], an extension, addresses concept drift by maintaining a sliding window of the latest data and replacing the original sub-tree with a better-performing alternative. VFDTc [60] enhances VFDT by handling numerical attributes and adapting to concept drift with node-level detection. Later extensions [61, 62] introduce an adaptive leaf strategy in VFDTc, selecting the best classifier from options like majority voting, Naive Bayes, and Weighted Naive Bayes. Recent studies [63, 64] question VFDT's foundation, the Hoeffding bound, for non-independent variables in information gain. An alternative impurity measure is proposed in a new online decision tree model [64], demonstrating its reflection of concept drift and potential use in CVFDT. IADEM-3 [65] addresses Hoeffding bound concerns by computing the sum of independent random variables for drift detection and pruning.

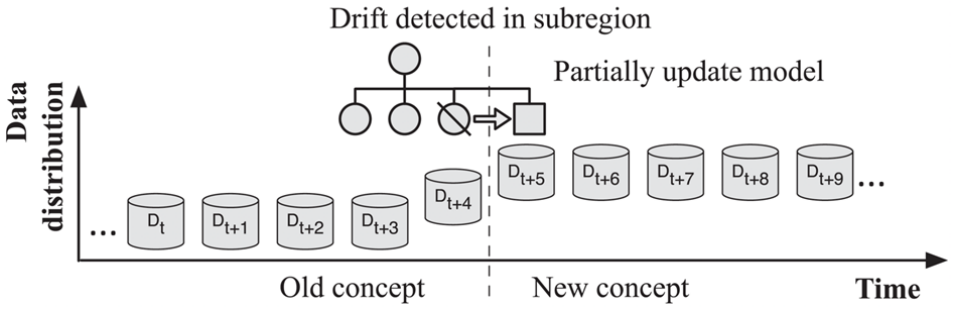



**Figure 2.7:** Partial updating approach for the adaptation phase.
DOI: 10.1109/TKDE.2018.2876857

CHAPTER

# 3

# State-of-the-art

In this chapter, we provide a comprehensive review of recent advancements in stream classification, addressing several critical challenges that arise in dynamic data environments. The rapid evolution of data streams introduces complexities such as concept drift, imbalanced multiclass scenarios, class overlap, classifier ensemble selection, emergence of new classes, and the incorporation of transfer learning. This introduction outlines the structure of the chapter and highlights the significance of each topic in advancing stream classification methodologies. The first section (Setion 3.1) focuses on concept drift, a phenomenon where the statistical properties of the data change over time. We explore various methodologies developed for real-time detection and adaptation to concept drift in streaming scenarios. This includes a review of techniques that enable systems to identify shifts in data patterns promptly, ensuring sustained classification accuracy. By analyzing the strengths of these approaches, we highlight the necessity for robust drift detection mechanisms in evolving data streams. Next, in Section 3.2, we delve into classifier ensemble selection in the context of streaming data. As the characteristics of data streams evolve, selecting the most appropriate ensemble of classifiers becomes crucial. We present algorithms designed to dynamically choose the best-performing classifiers based on the current data distribution. This section emphasizes the importance of adaptive ensemble strategies in enhancing classification performance amidst changing data landscapes. Section 3.3 addresses the challenges posed by imbalanced multiclass scenarios, particularly in the presence of overlapping classes. We discuss specialized oversampling techniques aimed at

countering the effects of class imbalance in streaming data. By providing insights into effective strategies for balancing the representation of minority classes, we lay the groundwork for developing robust frameworks that can maintain performance in imbalanced drifted streams. As new classes emerge in streaming systems, traditional classifiers often struggle to adapt effectively. In Section 3.4, we explore methodologies that specifically focus on the integration of new classes within existing frameworks. This section highlights innovative approaches that enhance the adaptability of stream classification systems, ensuring they remain effective as new data patterns emerge. In Section 3.5, we examine the role of transfer learning in stream classification. This section discusses how transfer learning techniques can leverage knowledge from related tasks to improve classification performance in dynamic environments. We explore current methodologies that facilitate the transfer of information across different domains, particularly in situations where labeled data may be scarce. The insights gained from this discussion will be pivotal for understanding how transfer learning can complement other strategies in our proposed framework. In Section 3.6,To further contextualize our discussion, we conduct a comparative analysis of recent works in the field, focusing on the recent works for each challange. We assess their contributions and identify limitations, illuminating specific gaps in the current research landscape that our work aims to address. Finally, Section 3.7 concludes this chapter by identifying key challenges and gaps, which inform the research direction for the subsequent chapters.

## 3.1 Concept Drift

In machine learning, concept drift refers to the phenomenon where the underlying data distribution changes over time [66–68] leading to a decrease in the accuracy or relevance of previously trained models. Detecting and responding to concept drift promptly is crucial for maintaining the performance of machine learning models. To address this challenge, various concept drift detection methods have been proposed in the literature. One widely used method is the Drift Detection Method (DDM)[33, 47] which employs a statistical test to compare the error rate of a model on consecutive data sets. By identifying significant performance decreases, DDM signals the presence of concept drift. Another approach is the

Early Drift Detection Method (EDDM)[33, 69] an extension of DDM that considers the error rate of a moving window of the latest data compared to the previous window. The ADaptive WINdow (ADWIN)[33, 69] approach uses a sliding window technique to monitor statistical differences between consecutive windows for drift detection. It dynamically adjusts the window size to adapt to changing drift patterns. The Kolmogorov-Smirnov windowing method (KSWIN) [69] calculates the Kolmogorov-Smirnov distance between two sliding windows to detect concept drift. Hoeffding's bounds with moving average test (HDDMA) and its variant HDDMW compute upper and lower bounds for the true mean of the data stream [33, 47] . By comparing these bounds, they can detect changes in the data distribution indicative of concept drift. Lastly, the Page-Hinkley [70] method measures the cumulative sum of errors and detects drift when the sum exceeds a predefined threshold. These concept drift detection methods are crucial in enabling machine learning models to adapt to the evolving nature of data streams. By continuously monitoring and detecting changes in data distributions, these methods facilitate the necessary adjustments and updates to maintain the model's performance and accuracy over time. Incorporating these methods into machine learning frameworks enhances their robustness and enables them to handle concept drift effectively.

## 3.2 Classifier Ensemble Selection

This study focuses on the overproduce-and-select approach for classifier ensemble selection methods [20, 23, 71]. The primary objective of classifier ensemble selection is to identify the optimal subset of classifiers from a larger ensemble, considering various criteria such as performance measures, diversity metrics, meta-learning techniques, and performance estimation approaches. This selection process aims to reduce computational complexity, enhance efficiency, and improve overall ensemble performance, making it highly valuable for real-world applications. By carefully selecting a smaller subset of classifiers, ensemble selection strikes a balance between accuracy and computational resources, adapting to the evolving nature of the data stream. This approach leverages the strengths of different classifiers and adjusts the ensemble composition to handle changing conditions effectively. The goal is to enhance the accuracy, robustness, and overall

performance of classification models in dynamic and challenging scenarios. There are two main approaches to the selection process: static and dynamic selection. Static selection assigns classifiers to specific partitions of the feature space, while dynamic selection chooses a classifier specifically for each unknown data sample based on its local competencies. Dynamic Ensemble Selection (DES) is a widely recognized approach that selects the best classifiers for each test instance, considering their competence within the local region of competence. The Randomized Reference Classifier proposed by Woloszynski and Kurzynski [21] stands out among various approaches. This classifier introduces randomness through beta distribution, enhancing adaptability and robustness. By considering the stochastic nature of class supports, the Randomized Reference Classifier can potentially improve classification performance in concept drift scenarios. However, it is important to note that employing diversity measures during the classifier selection process, as demonstrated by Lysiak [22], may lead to smaller ensembles but does not necessarily enhance classification accuracy. Overall, the overproduce-and-select approach for classifier ensemble selection methods offers a comprehensive framework for addressing the challenges associated with concept drift. By dynamically adapting the ensemble composition and leveraging the competencies of individual classifiers, this approach aims to improve classification performance, efficiency, and adaptability in dynamic and challenging scenarios.

## 3.3 Imbalanced data Streams

In the context of imbalanced data classification, as previously discussed, researchers have categorized three primary methodologies[72]. Our study primarily focuses on the first category, which pertains to addressing imbalanced data streams, particularly through sampling methods, specifically generating synthetic instances to rectify the class imbalance. This process is commonly referred to as oversampling[73] and aims to balance instance quantities across both classes [74–77] . It's important to note that class imbalance can manifest in two scenarios: binary imbalanced classes, featuring skewed distribution between two classes, and multi-class imbalanced situations. Our study is specifically centered on multi-class oversampling techniques. In addressing class imbalances within multi-class

contexts, Multi-Label SMOTE (MLSMOTE) [9] has extended the principles of SMOTE to cater to imbalanced multi-class learning scenarios. MLSMOTE significantly enhances classifier performance by generating synthetic examples for each minority class label. This approach thoughtfully considers neighboring examples in the feature space, ensuring that synthetic examples are appropriately assigned to their respective minority classes. Moreover, a recent advancement known as Multi-Label Synthetic Oversampling based on Local label imbalance (MLSOL) [72] has emerged as a promising technique. MLSOL systematically combats local imbalances within the domain of multi-class classification by employing distinct sampling strategies for each label. Empirical research confirms MLSOL's superiority by showcasing its capability to outperform existing methods in terms of classification accuracy and other evaluation metrics, particularly in effectively addressing local imbalance. Importantly, MLSOL offers a potentially effective approach to enhancing the performance of multi-class classification models, surpassing MLSMOTE in several aspects. Notably, MLSOL constructs synthetic samples exclusively from minority class instances within a restricted neighborhood, resulting in a more compact synthetic dataset compared to MLSMOTE. This attribute bears potential advantages for computational efficiency and helps mitigate concerns related to overfitting.

## 3.4 Streams with Emerging New Classes (SENC)

Existing approaches have been proposed to detect and handle the emergence of new classes in streaming data. Clustering-based methods, such as SACCOS [78], ECSMiner [79], and SAND [80], employ clustering techniques to identify new class emergence. However, these methods require access to true labels for either parts or all instances, limiting their practical applicability. Similarly, SENCMaS [81] uses matrix sketches for detecting emerging new classes but assumes the availability of true label information for all instances. In contrast, tree-based methods like SENCForest [82] and SEEN [83] utilize anomaly detection techniques to identify new classes, often with limited or no label information. However, these methods often suffer from high false positive rates and runtime inefficiencies. Another approach, SENNE [84], focuses on exploiting local information us-

ing the nearest neighbor ensemble for improved detection performance. Nevertheless, the absence of an effective model retirement mechanism in SENNE results in longer runtimes than alternative methods. The k-nearest Neighbor Ensemble-based method (KNNENS) [85] method emerges as a promising solution for the challenges of streaming emerging new class problems. By effectively utilizing a k-nearest neighbor-based hypersphere ensemble and incorporating model updates, the KNNENS approach tackles the issues of new class detection and known class classification within a unified framework. It is worth noting that an explicit limitation of existing methods is their lack of utilization of concept drift techniques for detecting emerging new classes and retraining the classification model. This limitation highlights the need for approaches that can effectively handle concept drift while addressing the emergence of new classes in streaming data.

## 3.5 Transfer Learning

In recent years, transfer learning has received significant attention due to its growing importance in addressing disparities between source and target domain distributions. Bridging the gap in distribution disparities is vital for optimizing performance in transfer learning, resulting in the development of diverse approaches, which can be broadly categorized into instance re-weighting and feature matching [86]. Instance re-weighting methods focus on aligning domain distributions by adjusting the weights of source instances, enabling the reuse of those source instances that closely align with the target domain. Notably, there's a strong emphasis on estimating these instance weights. For example, Huang et al. [87] introduced the Kernel Mean Matching (KMM) technique, which calculates weights by minimizing the mean differences between instances from the source and target domains within a Reproducing Kernel Hilbert Space (RKHS). Sugiyama et al. [88] put forth the Kullback-Leibler Importance Estimation Procedure (KLIEP), utilizing Kullback-Leibler distance as a metric for assessing domain distribution dissimilarity, which includes a model selection step. Building on these instance re-weighting methods, Sun et al. [89] introduced the 2-Stage Weighting Framework for Multisource Domain Adaptation (2SW-MDA) to address challenges in multi-source transfer learning. It simultaneously adjusts the weights of source domains

and their instances to reduce both marginal and conditional distribution disparities, akin to KMM, while leveraging the smoothness assumption for domain weighting. TrAdaBoost [90], a variation of the AdaBoost framework [91], operates by iteratively adjusting the weights of training data. In each iteration, it trains a classifier on a mix of source and target data and uses this classifier to make predictions on the training data. If a source instance is incorrectly predicted, its weight is reduced, diminishing its influence on the classifier. Conversely, the weights of misclassified target instances are increased to amplify their impact. An extension of TrAdaBoost, known as Multisource TrAdaBoost (MsTrAdaBoost) [92], is employed to address multisource transfer learning challenges. MsTrAdaBoost combines each source and target dataset, training a separate classifier for each. Subsequently, it selects the classifier with the least error on the target data to update the instance weights. On a different note, feature matching aims to establish a shared feature representation space between source and target domains, which can be achieved through either symmetric or asymmetric transformations. A typical example of a symmetric transformation is the Transfer Component Analysis (TCA) method by Pan et al. [92], employing Maximum Mean Discrepancy (MMD) [93, 94] [86] to minimize differences in marginal distribution between source and target domains within an RKHS. Expanding on TCA, Joint Distribution Adaptation (JDA) [95] has been introduced to address both marginal and conditional distribution disparities. Recognizing the varying importance of marginal and conditional distribution differences across different problems, Wang et al. [95, 96] introduced the Balanced Distribution Adaptation (BDA) approach, which introduces a balancing factor. Subspace Alignment (SA) [97] focuses on aligning domain distributions in a lower-dimensional subspace, selecting crucial eigenvectors using principal component analysis [97] and learning a linear transformation matrix to minimize differences in eigenvectors between domains. In contrast, the Distribution Alignment between Two Subspaces (SDA-TS) [98] was proposed to align both bases and distributions. Correlation Alignment (CORAL) [99], [87], an asymmetric transformation approach, is designed to align sub-space bases and employs second-order statistics. CORAL uses a learned transformation matrix to project source instances into the target domain. While there is a wide range of feature matching techniques in transfer learning, it is imperative to prevent the negative transfer, which occurs

when transferred knowledge hinders the performance of target tasks. One of the reasons for negative transfer is the inclusion of unrelated or detrimental source samples in the target domain. To mitigate this, transfer joint matching (TJM) [100] introduces sparsity regularization in the feature transformation matrix, aligning features and re-weighting instances simultaneously. Zhong et al. [101] have also developed strategies to mitigate the impact of unrelated source instances and ensure positive transfer. To tackle the challenges posed by partial transfer learning scenarios, where the source domain contains more classes than the target domain, prior works [1, 102, 103] introduce instance-level re-weighting and class-level re-weighting mechanisms. These mechanisms are employed to reduce the influence of outlier classes from the source domains. Additionally, another approach presented in [3] utilizes an adversarial neural network to align domain distributions. In this method, lower weights are assigned to the source samples that are deemed distant from the discriminator. This weighting reflects the perception that such instances have weaker relevance to the target domain. Yang et al. [104] introduce the HE-CDTL approach for Concept Drift Transfer Learning (CDTL). HE-CDTL leverages knowledge from both source domains and historical time steps within the target domain to improve learning performance. Its key advantages include the utilization of the class-wise weighted ensemble for historical knowledge and the implementation of AW-CORAL for knowledge extraction from source domains. The class-wise weighted ensemble empowers individual classes in the current learning process to select historical knowledge independently. AW-CORAL serves to minimize domain disparities between source and target domains while mitigating negative knowledge transfer. Extensive experiments demonstrate that HE-CDTL outperforms baseline methods in addressing transfer learning challenges in the context of concept drift. Melanie addresses the challenge of non-stationary environments by considering an online scenario where data in both source and target domains are generated. This method employs an online ensemble to learn models from each domain, subsequently combining these models using a weighted-sum approach. The models are trained incrementally, with their weights dynamically adjusted to handle concept drift. Generally, Melanie can be adapted to address CDTL by substituting the online learning ensemble with an ensemble designed for chunk-based concept drift.

## 3.6 Comparsion

In this section, we undertake a critical comparison of closely related works addressing the challenges of imbalanced multiclass streams 3.6.1, the emergence of new classes 3.6.2, and the integration of transfer learning 3.6.3 within streaming environments. The increasing complexity of real-world data streams necessitates advanced methodologies that can effectively manage the intricacies of these challenges. By examining various approaches in the literature, we aim to highlight their contributions, strengths, and limitations in dealing with imbalanced data distributions, adapting to new class occurrences, and leveraging transfer learning techniques. This comparative analysis not only sheds light on the current state of research but also underscores the specific gaps and unresolved issues that our work seeks to address, ultimately paving the way for more robust and adaptive solutions in the realm of streaming data classification.

### 3.6.1 Imbalanced Stream

In the context of multi-class classification, addressing class imbalances is crucial. Multi-Label SMOTE (MLSMOTE) extends the principles of SMOTE to generate synthetic examples for each minority class label, thereby enhancing classifier performance by considering neighboring examples in the feature space. Recently, Multi-Label Synthetic Oversampling based on Local label imbalance (MLSOL) has emerged as a more effective technique. MLSOL systematically addresses local imbalances by employing distinct sampling strategies for each label. Empirical research demonstrates that MLSOL outperforms existing methods, including MLSMOTE, in terms of classification accuracy and computational efficiency. By generating synthetic samples exclusively from minority class instances within a restricted neighborhood, MLSOL produces a more compact and efficient synthetic dataset, mitigating concerns related to overfitting. As illustrated in Fig. 3.1, MLSOL is more likely to select $x1$ as a seed instance because it is surrounded by more neighbors of the opposite class for $l3$. MLSMOTE assigns the label vector [0,1,0] to all synthetic instances based on their neighbors. In contrast, MLSOL creates more diverse instances by assigning labels according to their location. Moreover, synthetic instances $c2$ and $c3$ generated by MLSMOTE introduce noise, whereas

MLSOL copies the labels of the nearest instance to the new examples. In summary, MLSMOTE tends to generate new instances biased toward the dominant class in the local area, whereas MLSOL effectively explores and exploits both the feature and label space.

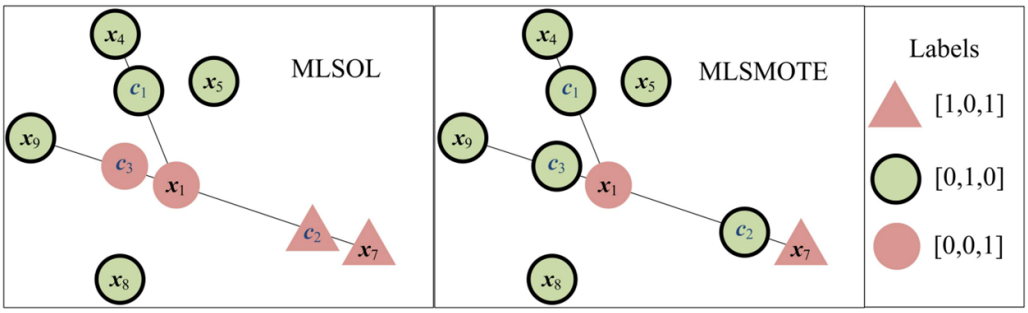



**Figure 3.1:** Comparsion of MLSMOTE and MLSOL generated instances.
DOI: 110.1016/j.patcog.2021.108294

Table 3.1 presents a comparison of two methods, MLSMOTE and MLSOL, which are designed to address the issue of imbalanced data in multi-class classification. MLSMOTE enhances classifier performance by generating synthetic examples for each minority class label, thereby balancing the class distribution. Its primary advantage is the generation of these synthetic examples, which helps mitigate the imbalance. However, it has significant limitations: the synthetic samples generated might be related to the majority class, which can blur the distinction between classes, and the method struggles with overlapping classes, leading to potential misclassification. On the other hand, MLSOL systematically combats local imbalances by employing distinct sampling strategies for each label within a restricted neighborhood. This localized approach allows MLSOL to generate synthetic examples more precisely, addressing local imbalances effectively. Nonetheless, like MLSMOTE, MLSOL faces challenges with overlapping classes, which can result in misclassification in areas where class boundaries are not clear. Despite its advantages in handling local imbalances, the overlapping class issue remains a critical limitation for both methods, affecting their overall effectiveness in classification tasks.

**Table 3.1:** Comparison of the MLSMOTE and MLSOL methods.

| Method | Theory | Advantages | Limitations |
|---|---|---|---|
| **MLSMOTE** | MLSMOTE significantly enhances classifier performance by generating synthetic examples for each minority class label. | Generating synthetic examples for each minority class label. | • Random synthetic samples may be related to the majority class.<br>• Overlapping classes. |
| **MLSOL** | MLSOL systematically combats local imbalances within the domain of multi-class classification by employing distinct sampling strategies for each label. | Generating synthetic examples for each minority class label within a restricted neighborhood. | • Overlapping classes. |

### 3.6.2 Emergence of new classes

Effectively detecting and adapting to new classes in streaming data is crucial for maintaining classification accuracy. Tree-based methods like SENCForest and SEEN utilize anomaly detection but face high false positive rates and runtime inefficiencies. SENNE improves detection performance using a nearest neighbor ensemble but suffers from longer runtimes due to the lack of an effective model retirement mechanism. The k-nearest Neighbor Ensemble-based method (KNNENS) addresses new class detection and known class classification using a k-nearest neighbor-based hypersphere ensemble and dynamic model updates. However, a critical limitation of these methods is their inadequate handling of concept drift, which is essential for detecting new classes and retraining the classification model. These methods represent the best closely related work for our proposal, which aims to build upon them by incorporating robust concept drift techniques for more adaptive and resilient classification systems.

Fig. 3.2 illustrates the SENCForest approach, which divides the space into three regions (normal, outlying, and anomaly) and detects emerging new classes (anomalies) using a calculated threshold path length. Fig. 3.3 depicts the SENNE algorithm, where hyperplanes are drawn in three dimensions ($x1$, $x2$, and $x3$) for each class (Fig. 3.3a). New instances are then classified as emerging or known classes based on the rank of each class (Fig. 3.3b). Fig. 3.4 presents the KNNENS algorithm, which draws hyperplanes for all class samples (Fig. 3.4a), and classifies new instances using a voting mechanism to determine if the instance is an emerging or known class (Fig. 3.4b). These visualizations highlight the operational differences between the SENNE and KNNENS algorithms in handling the classification of emerging and known classes.

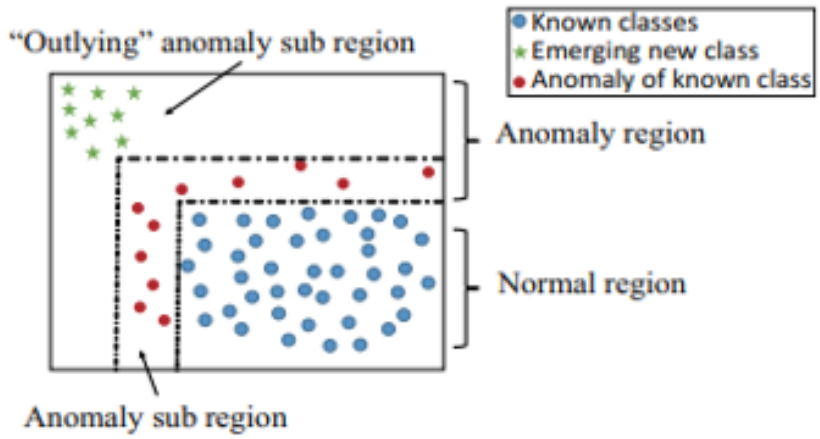



**Figure 3.2:** Overview of SENCForest detection flow.
DOI: 10.1109/TKDE.2017.2691702

Table3.2 compares three methods for emerging class detection: SENCForest, SENNE, and KNNENS. SENCForest employs the anomaly detection method iForest [16] for new class detection and uses a threshold path to identify anomalies, serving both as an unsupervised anomaly detector and a supervised classifier. However, it has a high potential for false positives and depends on a complex path length threshold. SENNE utilizes a nearest neighbor-based hypersphere of one class ensemble to explore local neighborhood information and sort distances, handling both low and high geometric distances between classes. Its limitations include

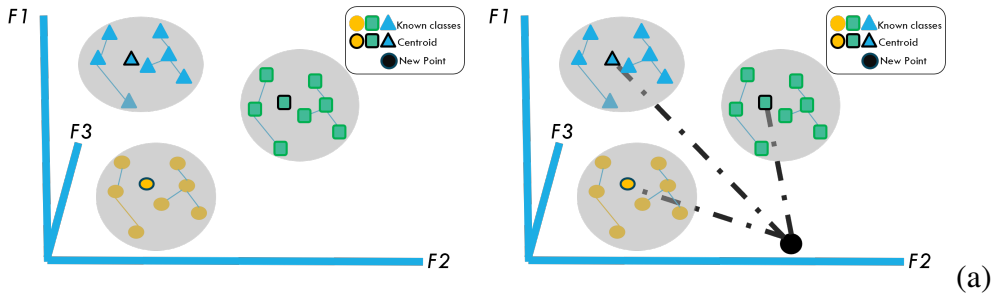



**Figure 3.3:** Overview of the Stream Emerging Nearest Neighbor Ensemble (SENNE).

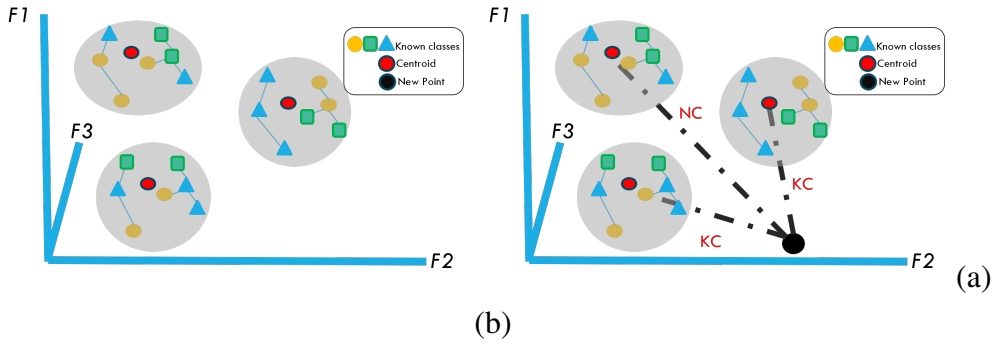



**Figure 3.4:** Overview of the k-nearest Neighbor Ensemble-based (KENNE).

the assumption that the distribution of known classes remains unchanged and it has lengthy update times. KNNENS employs a nearest neighbor-based hypersphere of all class ensembles to explore local neighborhood information, reducing false positives for new classes without needing true labels for model updates. However, like SENNE, it assumes that the distribution of known classes remains unchanged.

### 3.6.3 Transfer Learning

In the realm of transfer learning, three prominent methods—CORAL, Melanie, and HE-CDTL—serve as closely related approaches to our proposed method. Correlation Alignment (CORAL) is an asymmetric transformation approach that aligns sub-space bases using second-order statistics. By employing a learned transformation matrix, CORAL projects source instances into the target domain, thereby min-

**Table 3.2:** Comparison of the SENCForest, SENNE, and KENNE methods.

| Method | Theory | Advantages | Limitations |
|---|---|---|---|
| **SENCForst** | employs anomaly detection method iForest for a new class detection and then applies threshold path to detect the anomalies. | SENCForest serves as both an unsupervised anomaly detector and a supervised classifier. | • Potential for High false positives.<br>• Dependency on path length threshold (more complexity). |
| **SENNE** | nearest neighbor-based hypersphere of one class ensemble to explore local neighborhood information and sort distance to calculate distance. | SENNE is able to handle both the low and high geometric distance between two classes in the feature space. | • Assumes that the distribution of known classes remains unchanged.<br>• Take long time for update. |
| **KENNE** | nearest neighbor-based hypersphere of all class ensemble to explore local neighborhood information. | KNNENS to reduce false positives for the new class. KNNENS does not require true labels to update the model. | • Assumes that the distribution of known classes remains unchanged. |

imizing domain discrepancies and reducing negative knowledge transfer. Melanie addresses the challenge of non-stationary environments through an online ensemble learning approach. It incrementally trains models from both source and target domains, dynamically adjusting their weights to handle concept drift, and combines these models via a weighted-sum approach as shown in Fig. 3.5. As Shown in Fig. 3.6 This method can be extended to Concept Drift Transfer Learning (CDTL) by using an ensemble for chunk-based concept drift. HE-CDTL, designed explicitly for CDTL, leverages knowledge from source domains and historical time steps within the target domain to enhance learning performance. It utilizes a class-wise weighted ensemble for historical knowledge and implements AW-CORAL for ex-

tracting knowledge from source domains. The class-wise weighted ensemble allows individual classes to select historical knowledge independently, while AW-CORAL minimizes domain disparities and mitigates negative knowledge transfer. Extensive experiments have shown HE-CDTL to outperform baseline methods in addressing transfer learning challenges in the context of concept drift. Together, these methods provide a comprehensive framework for effective transfer learning in dynamic and evolving data environments.

Table3.3 compares three methods: CORAL, Melanie, and HE-CDTL. CORAL (Correlation Alignment) utilizes a learned transformation matrix and Singular Value Decomposition (SVD) to project source instances into the target domain, effectively minimizing domain discrepancy and reducing negative knowledge transfer. However, it faces challenges with non-stationary and heterogeneous data. Melanie (Multisource Online Transfer learning for Non-stationary Environments) addresses online learning problems where data in source and target domains are generated from non-stationary environments. Its advantages include considering online problems but it too is limited by the complexities of online learning and data heterogeneity. HE-CDTL (Class-wise Weighted and Domain-wise Ensemble) minimizes domain shift by aligning second-order statistics of source and target distributions, leveraging historical knowledge to reduce disparities between domains. Despite its strengths, it relies on the quality of the source domain and also struggles with heterogeneous data.

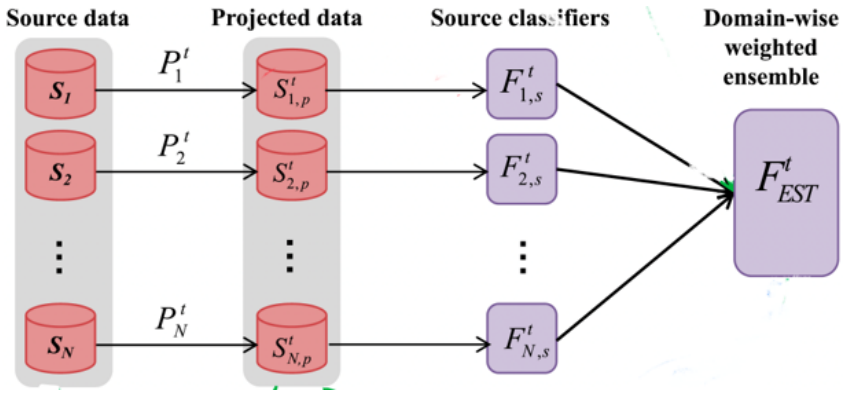



**Figure 3.5:** Overview of CORrelation ALignment (CORAL)

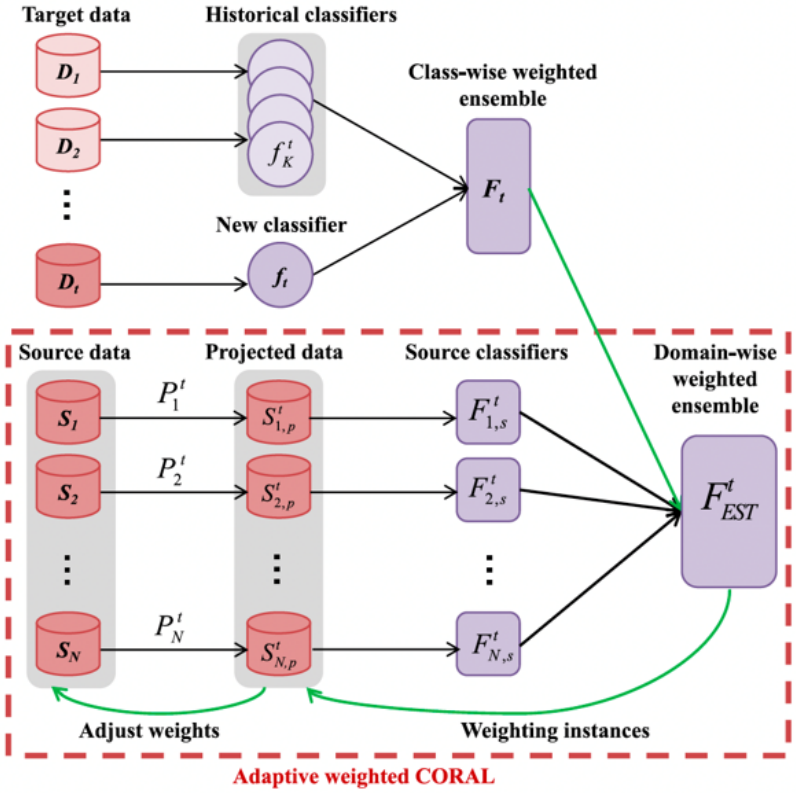



**Figure 3.6:** Overview of Concept Drift Transfer Learning (CDTL).

## 3.7 Challenges

By comparing the literature on ensemble learning for classification tasks, the proposals in this thesis differ from other studies in several ways:

- As evident from our literature review on imbalanced streams, most studies have concentrated on generating synthetic samples while ignoring class overlap. *To address this challenge*, we propose an approach to generate non-overlapping classes in imbalanced streams.

- Oversampling techniques often perform inefficiently in the presence of concept drift. *To tackle this issue*, we introduce a methodology that selects the oversampling technique based on the current and historical distribution of the stream chunks.

**Table 3.3:** Comparison of the CORAL, Malanie, and CDTL methods.

| Method | Theory | Advantages | Limitations |
|---|---|---|---|
| **CORAL** | Correlation Alignment (CORAL) uses a learned transformation matrix and Singular Value Decomposition (SVD) to project the source instances into the target domain. | CORAL can minimize domain discrepancy across source and target domains, meanwhile reducing the negative knowledge transfer. | • Non-stationary environments.<br>• Heterogenous multisource. |
| **Melanie** | Multi-sourcE onLine TrAnsfer learning for Non-statIonary Environments (Melanie). utilize the class-wise weighted . | It considers an online problem in which the data in source and target domains are generated from non-stationary environments. | • Based on the online learning only.<br>• Heterogenous multisource. |
| **MLSOL** | HE-CDTL uses the class-wise weighted and domain wise ensemble for historical knowledge and reduce the disparities between the source and target domains . | HE-CDTL minimizes domain shift by aligning the second-order statistics of source and target distributions. | • Depend on source domain quality.<br>• Heterogenous multisource. |

- Our literature review on non-stationary environments reveals that most works focus on detecting emerging new classes while overlooking distribution changes. *To overcome this challenge*, we propose a combined approach utilizing Dynamic Ensemble Selection (DES) to select the best classifier for each chunk based on stream distribution, k-means clustering, and concept drift to address both emerging new class detection and distribution changes.

- In our literature review on transfer learning, we observed that most studies focus on homogeneous multisource transfer and neglect heterogeneous mul-

tisources in non-stationary environments. *To resolve this issue*, we propose a combined approach integrating Dynamic Ensemble Selection (DES), Concept Drift Transfer Learning (CDTL), eigenvector techniques, and concept drift to address heterogeneous transfer learning in non-stationary environments.

CHAPTER

# 4

# Dynamic Classification Ensembles for Handling Imbalanced Multiclass Drifted Data Streams

In recent years, the explosion of high-speed data streams has presented new challenges for machine learning models. Three critical issues that have emerged are concept drift, class imbalance, and class overlap. Concept drift refers to the phenomenon where the statistical properties of a data generation process change over time [1, 2]. This signifies that the underlying concepts, relationships between variables, or data distribution can change, leading to a fundamental shift in the nature of data. Dealing with concept drift poses a fundamental challenge in machine learning and data mining. This can cause models trained on historical data to become inadequate when applied to new data affected by concept drift, leading to a decline in the model performance [2]. To address this issue, concept drift detectors are used to identify changes in data stream distributions by leveraging information associated with classifier performance or the incoming data items themselves. These signals frequently prompt model updates, retraining, or substitution of an old model with a new one. In addition, class imbalance [2, 4], which is characterized by uneven class distribution, poses a challenge for traditional classifiers [5], particularly in multiclass scenarios where minority class samples are at risk of

misclassification owing to their limited representation [6]. Addressing imbalanced data classification requires specialized techniques to ensure accurate minority class classification without compromising the performance of the majority class [7–9]. This challenge becomes more daunting when minority class instances are scattered in unknown configurations, thereby increasing the likelihood of overfitting during the learning process. To address this issue, three primary methods are employed in the context of imbalanced data classification, which are effective in both binary and multiclass imbalance scenarios [10]. The first approach involves sampling methods that address class imbalance by either reducing the number of majority class instances (undersampling) or generating artificial minority class instances (oversampling) [11, 13]. The second and third groups encompass adaptive algorithms and hybrid methods, respectively. Adaptive algorithms include one-class and cost-sensitive classification [14]. Hybrid methods merge data preprocessing with classification techniques, often utilizing ensemble classifiers to effectively mitigate class imbalance and enhance classifier performance [15, 16, 19]. Class overlap occurs when instances from different classes inhabit the same region in the data space [17, 18]. This overlap complicates the task of distinguishing between representative instances of various classes and posing performance challenges for traditional classifiers. This issue is commonly referred to as class overlap. Researchers have proposed class overlap undersampling techniques to address class imbalance problems [20]. These techniques aim to leverage local similarities among minority instances to identify potentially overlapping majority instances. Although these methods have demonstrated promising results in improving model performance on specific datasets, many of them rely heavily on nearest-neighbor approaches to detect overlapping regions around minority instances. This approach often neglects the global similarity within the overlapping domain, which can lead to local optimal values getting stuck during calculations. Furthermore, determining appropriate parameters for these models poses a significant challenge. If the parameter selection is too extensive, it may lead to the exclusion of valuable instances, while conservative parameters may overlook instances with overlap. This issue can significantly affect classifier performance, particularly when handling streams containing both a minority class and instances with class overlap. Therefore, both class imbalance and class overlap present significant challenges in the realm of data stream analyses.

Consequently, addressing class imbalance is crucial in multiclass learning, leading to research efforts that focus on both concept drift and class imbalance challenges. Researchers have explored techniques such as DES and multiclass oversampling to address these issues. Dynamic classifier ensembles offer the unique ability to adapt their composition based on data characteristics, making them valuable in situations with evolving data conditions [21]. The aim of classifier ensemble selection is to identify the optimal subset of classifiers from a larger ensemble. A prominent approach in classifier ensemble selection is the overproduce-and-select strategy. This selection process is guided by diverse criteria, including individual performance measures, diversity metrics, meta-learning techniques, and performance-estimation approaches. Such optimization is particularly crucial in scenarios in which striking a balance between accuracy and computational resource constraints is paramount. There are two distinct approaches: static and dynamic approaches. Static selection involves assigning classifiers to predefined feature partitions, whereas dynamic selection adaptively selects classifiers based on their competency [22]. Dynamic selection offers two choices: Dynamic Classifier Selection (DCS) and Dynamic Ensemble Selection (DES). DCS algorithms enable the selection of the most appropriate classifier for each data point, based on its local competencies. In contrast, DES focuses on selecting the optimal classifiers for each instance based on their competence within localized regions [23–25]. Competency assessment relies on a Dynamic SELection dataset (DSEL) containing labeled samples. Moreover, innovative techniques, such as the randomized reference classifier, introduce randomness into class supports to enhance adaptability in addressing the challenges related to imbalanced data. The main goal of this study is to formulate a precise classification approach that addresses changing conditions. Specifically, the first proposed approach aims to address scenarios where there is an uneven distribution among several classes, overlapping instances of classes, and instances where the fundamental concept of data evolves. To address these challenges, we employed dynamic classifier ensembles. These ensembles utilize oversampling techniques, implemented either on a global or local scale, as a preprocessing step to address class imbalance. Furthermore, we enhanced multiclass learning techniques to counteract class imbalance through method adaptation.

The remainder of this chapter is organized as follows: In Section 4.1, we present the motivations and the contributions. The first proposed framework are discussed in detail in Section 4.2. The experimental results and the discussion are presented in Sections 6.3.

## 4.1 Motivations and Contributions

1. We introduce a classification approach that dynamically adjusts to multi-class imbalanced data, incorporates mechanisms for detecting concept drift, and optimizes classifier ensemble selection. The objective is to enhance the classification accuracy, specifically for multiclass imbalanced non-stationary streams.
2. Additionally, we propose an adaptive method for the class imbalance issue, considering the data distributions and historical instances of class imbalance. This is particularly relevant in cases where class overlap occurs within the multiclass and drifted datan performance by selecting the most suitable over-sampling method based on the unique characteristics of the data stream.

## 4.2 Proposed Methodology

In this section, we present the primary phases of our study, comprising three distinct stages. Following this introduction, we delve into the synthetic data-generator method, providing a comprehensive breakdown of the four essential steps. Our proposal aims to develop a robust approach designed to overcome the challenging domain of multiclass classification for imbalanced and drifting data streams. In pursuit of this goal, our proposal addresses the four primary challenges inherent in constructing our approach.

- **Multi-class imbalanced streams:** This study focuses on addressing the widespread problem of imbalanced data streams in the context of multiple classes. We aim to address this issue using well-known techniques, notably MLSMOTE and MLSOL.

- **Overlapping class:** We also address a critical challenge involving class overlapping, a factor known to substantially affect model performance [23, 24]. To address this issue, we introduce an adaptive method that generates nonoverlapping synthetic instances, thereby enhancing the overall performance of the model.

- **Drifted streams:** First proposed approach integrates a concept drift detector to identify shifts in the underlying data distribution. This dynamic detection mechanism enables the model to promptly recognize changes and adjust its classifiers, thereby ensuring its effectiveness in handling drifting stream.

- **Classifier performance:** To enhance the classifier performance, our approach employs Dynamic Ensemble Selection (DES). This technique creates a pool of classifiers and dynamically selects the most suitable classifier for each incoming data point, further improving classification accuracy and robustness.

### 4.2.1 Approach Overall Details

First proposed approach is designed with three distinct phases that work together to improve its performance in managing multiclass imbalanced and drifting data streams.

- **DES phase (dynamic ensemble selection phase):** The first phase, known as the dynamic ensemble selection (DES) phase, is responsible for selecting the most appropriate classifier for the incoming data. This ensures that the selected classifier is well-suited for the current data chunk.

- **Drift detector phase:** The second phase of our approach is the drift detector phase, which operates in real-time to continuously monitor the data stream. Its primary function is to identify any signs of concept drift, which indicates shifts in the underlying data distribution over time.

- **Synthetic data generator phase:** The final phase of our approach is the synthetic data generator phase, which is dedicated to generating synthetic data for the minority classes. This step is crucial for addressing class imbalance

by producing additional samples for underrepresented classes, thereby significantly enhancing the model's ability to accurately classify instances from minority classes.

Specifically, as shown in Fig. 4.1, the DES phase retrieves the current data chunk from the stream and applies the DES technique to select the most suitable classifiers for the received chunk. The selected classifiers were then passed to the second phase, where they were employed to predict the class of each instance within the received data chunk. Simultaneously, detectors like ADWIND or DDM are employed to monitor any occurrence of concept drift. If the discrepancy between the class frequency and standard deviation of the current chunk is significant, as described in reference [33], and the imbalance ratio exceeds the average imbalance ratio, the current chunk is forwarded to the third phase, as indicated by the red rectangle in Fig. 1. In the third phase, first proposed method uses a set of equations 4.14.24.3 to identify minority classes. The initial equation computes the frequency of each class within the current chunk, where $i$ represents the current chunk, $c$ denotes the input class, and $y_i$ refers to the predicted class. The second equation determines the optimal frequency for each class based on the size of the chunk ($n$) and the number of classes in the current chunk($C$). Finally, the third equation identifies the classes as minority classes if their frequency deviates significantly from the standard deviation of the current chunk, where $sd_c$ represents the standard deviation of the class instances, and $frq_i$ refers to the standard deviation of the current chunk. As shown in Fig. 4.2, phase These identified minority classes are then fed into the synthetic data generator phase, which increases the minority class samples to balance any imbalanced chunks. This ensures optimal performance for the new classifiers. Algorithm 1 provides a comprehensive outline of the process of the first proposed approach, which is the main contribution of our study and is designed to effectively address multiclass imbalanced and drifting data streams, uses streaming data as input, and systematically executes each step within the approach. The outcome of this process is the classification prediction generated using the first proposed approach.

### 4.2.2 Synthetic Data Generator

Fig. 4.2 presents a comprehensive overview of the synthetic data generator phase, which is an essential component responsible for generating synthetic samples by considering both data distribution and historical chunk behaviors. This phase has several advantages and can perform a wide range of tasks.

- **Similar chunk analysis:** Initially, the phase analyzes the current chunk distribution and identifies a similar chunk from historical data. This analysis forms the basis for generating synthetic samples that align with prevailing distribution patterns.

- **Oversampling method selection:** This phase utilizes the knowledge of the oversampling technique applied to the identified similar chunk. Consequently, it employs an alternative oversampling technique, using MLSMOTE and MLSOL, to create the most effective synthetic data. This step is designed not only to optimize the current classification but also to preemptively address potential drifts in similar future chunks.

- **Class overlap validation:** This step involves generating synthetic samples for the minority classes to effectively address the issue of minority classes and consequently enhance the classifier performance. The process utilizes the K-Nearest Neighbor (KNN) algorithm, as applied in [25], to identify overlaps between newly generated data instances and existing instances. If overlaps are detected, the first proposed approach iteratively removes these samples because their presence can potentially diminish the overall classifier performance [23, 24]. Consequently, the first proposed approach generates alternative samples to address this challenge and preserve the primary objectives of the synthetic data generator step.

- **Continuous refinement:** This process iterates until it successfully generates high-quality synthetic data that aligns with the data distribution, minimizes overlap, and mitigates potential concept drift. The generated data were subsequently utilized in the training phase to improve classifier accuracy.

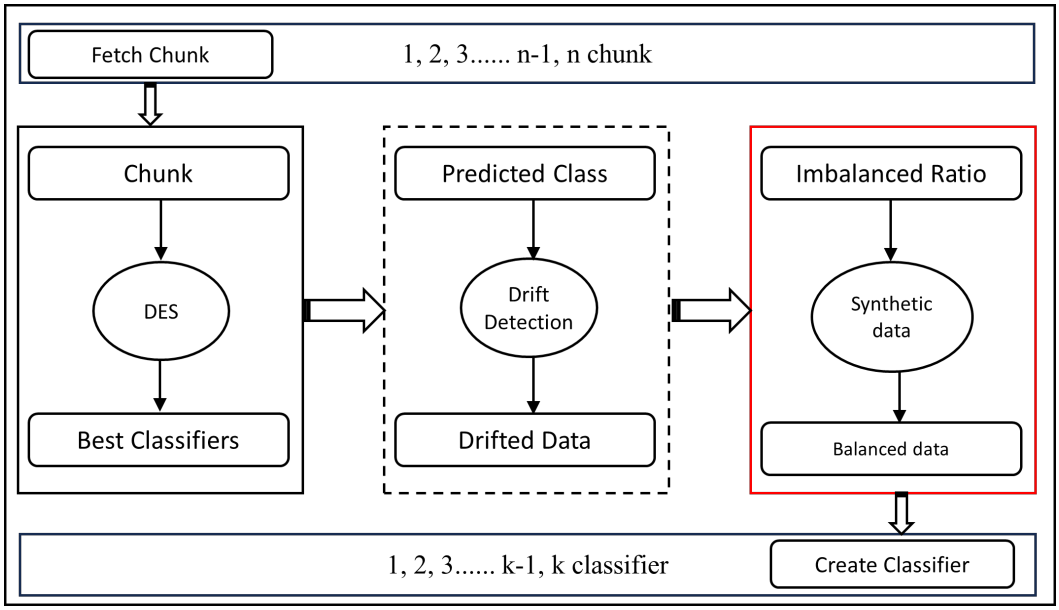



**Figure 4.1:** First Proposed Approach (PA1) for imbalanced multi-class drifted data streams.

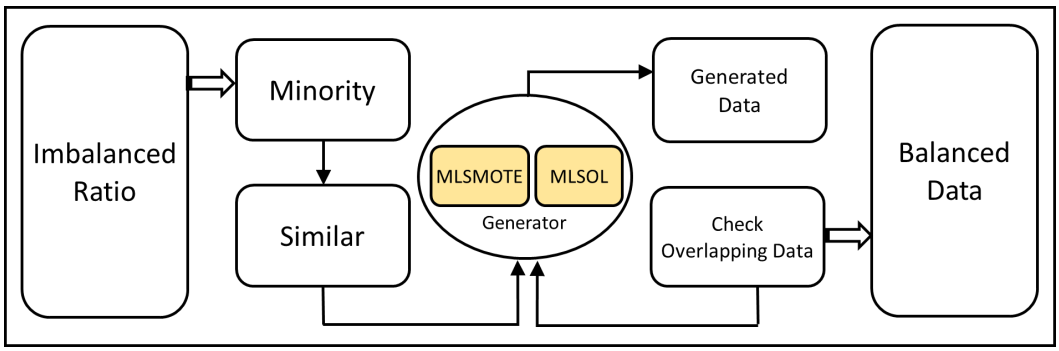



**Figure 4.2:** Flow of the synthetic data generator.

$$frq_c = \sum_{i=1}^{\text{chunk size}} \begin{cases} 1, & \text{if } y_i = c \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, 2, 3, \ldots \text{chunk size} \tag{4.1}$$

$$\text{best } freq_n = \frac{|n|}{|C|} \tag{4.2}$$

$$\text{classes typechunk} = \sum c = 1^{C} \begin{cases} \text{Minority}, & \text{if } diff(sd_c - frq_i) > \text{best } freq_{\text{chunk}} \\ \text{Majority}, & \text{otherwise} \end{cases} \tag{4.3}$$

**Algorithm 1:** First proposed approach algorithm for imbalanced multi-class drifted data streams.

**Input:** data stream, maximum classifiers pool size $\kappa$
**Output:** Prediction $P$

$\psi, \Psi, \Omega, \mu \leftarrow \emptyset$;
$\omega \leftarrow 0$;
**for** *stream have chunk* **do**
  **if** $a$ *is the First chunk* **then**
    $k \leftarrow$ `trainingNewClassifier`$(a)$;
    $P \leftarrow$ `getPrediction`$(a, k)$;
  **else**
    $k \leftarrow$ `DES`$(a, \Psi)$;
    $P \leftarrow$ `getPrediction`$(a, k)$;
    $\psi \leftarrow$ `conceptDriftDetector`$(P)$;
    **if** $\psi > 0$ **then**
      $\Omega \leftarrow$ get classes frequency according to Eq.1;
      $\omega \leftarrow$ best frequency according to Eq.2;
      $\mu \leftarrow$ get minority classes according to Eq.3;
      $b \leftarrow$ utilize $a$ and $\mu$ to get the synthetic data according to Algorithm 2;
      trainingData $\leftarrow a + b$;
      $k \leftarrow$ `trainingNewClassifier`(trainingData);
      $\Psi \leftarrow \Psi + k$;
      **if** $\Psi > \kappa$ **then**
        `removeWorstClassifier`$(\Omega)$;
    $P \leftarrow$ `getPrediction`$(a, k)$;
**return** $P$

In Algorithm 2, also known as the Synthetic Data Generator, we input three essential elements: the minority class samples, the current data chunk, and the desired size for generating synthetic data. The primary objective of this algorithm is to produce synthetic data samples. To achieve this, we employ the KNN algo-

rithm to identify any overlapping instances within the current chunk (Line 3). This algorithm utilizes two specific techniques, MLSMOTE [33] and MLSOL [105]. MLSMOTE was chosen for its introduction of randomness during instance generation, reducing its reliance on the local characteristics and distribution of the minority class. This randomization diminishes the likelihood of producing overlapping instances, particularly in cases where minority class instances are situated in overlapping regions. In contrast, MLSOL considers the local behavior of minority classes, resulting in synthetic points that closely resemble the minority class. This approach significantly improves the accuracy of the classifier (lines 4-10). Additionally, in this algorithm, specifically from lines 11 to 17, these lines are dedicated to generating synthetic instances that ensure non-overlap with existing classes. This procedure depends on the selected oversampling method and utilizes the KNN algorithm to guarantee that the generated instances do not overlap with the existing ones.

**Algorithm 2:** Synthetic data generator.

**Input:** Minority classes $\mu$, current chunk $a$, sample size $\eta$, historical chunks $h$
**Output:** Generated data $b$
$b \leftarrow \emptyset$;
$f \leftarrow$ MLSMSOTE;
$knn \leftarrow$ kNearestNeighbor$(a)$;
$chunk \leftarrow$ similarChunk$(a, h)$;
$f \leftarrow$ similarChunkOverSamplingMethod$(chunk)$;
**if** $f =$ *MLSMSOTE* **then**
  $f \leftarrow$ MLSOL;
**else**
  $f \leftarrow$ MLSMSOTE;
**while** $|b| < \eta$ **do**
  $p \leftarrow$ generateSyntheticPoint$(\mu, f)$;
  $similarPointsClass \leftarrow$ KNN.getKneighbor$(b)$;
  **if** $similarPointsClass = \mu$ **then**
    $b \leftarrow b \cup \{p\}$;
**return** $b$;

## 4.3 Experimental Results

The objective of the experiments conducted in this study was to evaluate the efficacy of multiclass oversampling techniques in enhancing the performance of the first proposed method in imbalanced drifted multiclass classification streams. Our primary goal was to develop a novel approach that combines Dynamic Ensemble Selection (DES) to improve classification accuracy and robustness in such streams. These experiments yielded valuable insights that could further refine the performance of the first proposed approach and its ability to effectively handle imbalanced data streams. These findings provide a better understanding of the capabilities of the first proposed approach and offer insights into an optimal strategy for tackling minority class issues and concept drift in imbalanced data streams. This study contributes to the advancement of stream mining techniques for generating more accurate and robust classification models in dynamic data stream environments. By addressing the challenges posed by minority classes and concept drift, this study offers valuable insights for improving the performance of the first proposed approach and enhancing the overall efficiency of stream mining. The two main questions to be answered are:

- $Q_1$: What is the impact of reduced and consistent data on the performance of ensemble learning?
- $Q_2$: Is it possible with the search capability of swarm intelligence to enhance the combination of classifiers?

### 4.3.1 Experimental setup

The evaluation of the first proposed approach incorporated the utilization of various metrics such as recall, precision, specificity, f1 score, balanced accuracy score (BAC), and geometric mean score (G-mean) [34]. The experimental protocol utilized for evaluation was the test-then-train approach [35], where the classification classifier was trained on a specific data chunk and subsequently evaluated on the subsequent one. The chunk size was standardized for all utilized data streams to 2,000 instances. We employed four classification classifiers as base estimators: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Gaussian Naive Bayes

(GNB), and Hoeffding Tree (HT), as implemented in scikit-learn [36]. A pool of classifiers was constructed with a maximum size of L = 8, where the DES selected the best classifier for each chunk. If the pool surpassed the set threshold (L), the classifier with the lowest performance was eliminated. The experiments were conducted using Python programming language, and the source code was publicly available on GitHub[1] . We conducted a comparison between multiclass oversampling techniques (MLSMOTE and MLSOL) and first proposed approach to demonstrate the effectiveness of our contribution. Additionally, we conducted these experiments using two different concept drift detectors, ADWIN [28] and DDM [27], to demonstrate the adaptability and robustness of first proposed approach across varying drift detectors.

### 4.3.2 Data Streams

In this study, the first approach was assessed using various datasets including benchmark datasets, a real application stream dataset, and synthetic data streams. The Stream-learn Python library was used to conduct the evaluations [31]. Table 4.1 illustrates the benchmark dataset employed in this study, which consists of the Covertype dataset containing 40 features, seven classes, and 581,010 instances. For real application stream evaluation, the Sensor stream dataset was used, which consisted of five features, 58 classes, and 392,600 instances. This represents a real-world application scenario and provides valuable insights into the performance of the first proposed approach in practical settings. Synthetic datasets were generated using Scikit-learn Python library to evaluate the performance of the first proposed approach. The synthetic dataset was designed to simulate data streams and comprised 10 features and four classes divided into 200 chunks of 2,000 instances each. The performance of the first approach was systematically evaluated using these datasets and a stream-learn library. These evaluations provided insights into the effectiveness of the first approach in handling different types of data streams, including benchmark datasets, real application streams, and synthetic data streams.

[1] `https://github.com/Amadkour/dynamic__classification_ensembles_for_handling_imbalanced_multi-class_drifted_data_streams.git`

**Table 4.1:** Characteristics of the datasets used in the experiments.

| Dataset | Number of Features | Number of Classes | Number of Instances |
|---|---|---|---|
| Covertype dataset[1] | 40 | 7 | 581,010 |
| Sensor Stream dataset[2] | 5 | 58 | 392,600 |
| Synthetic stream | 8 | 3 | 200,000 |

### 4.3.3 Analysis of Experimental Results

The performance of the first proposed approach was comprehensively assessed on multiple data streams, considering two distinct concept drift detectors, ADWIN and DDM. To ensure thorough evaluation, six key performance metrics—F1 score, recall, precision, G-mean, specificity, and balanced accuracy—were carefully presented using two visualization diagrams: radar and line. A radar diagram was strategically utilized to provide an overview that effectively depicted the performance of each algorithm across the six metrics. The mean value of each metric was calculated to present the overall performance of each method (MLSMOTE, MLSOL, PA1). It is important to note that the PA1 is represented by red lines.

#### 4.3.3.1 Results on the Benchmark Stream

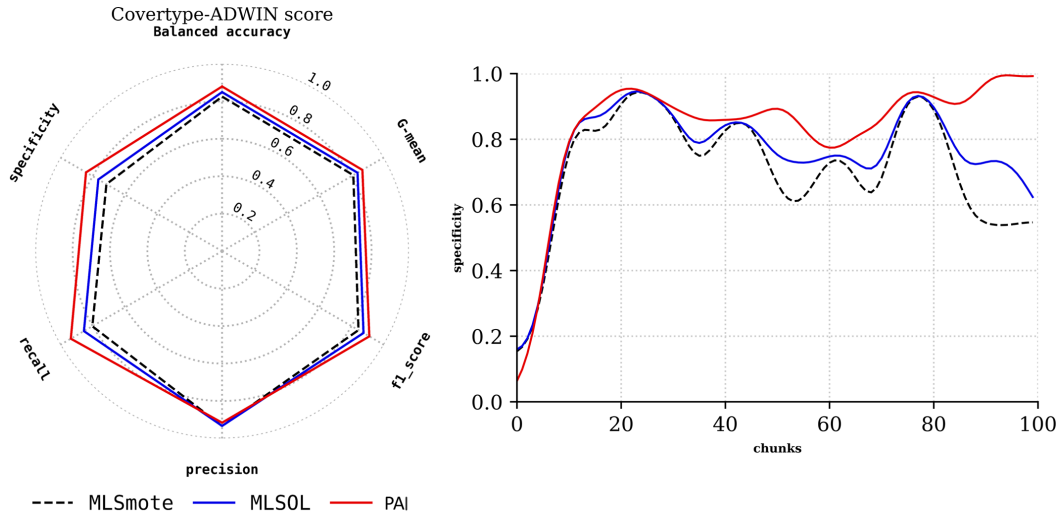



**Figure 4.3:** Performance of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Covertype Benchmark dataset using the ADWIN concept drift detector.

The results of the mentioned methods (MLSMOTE, MLSOL, PA1) applied to the Covertype dataset are presented in Fig. 4.3 , utilizing ADWIN as the drift detector. The radar diagram illustrates the comparative performance of three models—MLSMOTE, MLSOL, and PA1—across six evaluation metrics: precision, recall, specificity, F1 score, G-mean, and balanced accuracy. Each axis represents a metric, normalized between 0 and 1, allowing for direct comparison. The PA1 model exhibits superior performance across most metrics, particularly in precision, F1 score, and G-mean, as indicated by the red line. The MLSOL model, represented by the blue line, shows a balanced performance but underperforms PA1 in specificity and G-mean. In contrast, the MLSMOTE model, depicted by the dashed black line, achieves relatively lower scores, particularly in recall and balanced accuracy. This diagram highlights the trade-offs among the models, offering a clear depiction of their strengths and weaknesses across multiple dimensions, which is essential for informed decision-making in machine learning applications.. The line diagram in Fig. 4.3 shows the classification accuracy across 100 data chunks using the specificity metric for the methods in each chunk. Notably, all methods exhibit suboptimal accuracy during the first 20 chunks. However, a noticeable improvement was observed beyond this initial phase. The key factor driving this improvement was the expansion of the classifier pool, which now encompasses a growing number of classifiers. This expansion enables the Dynamic Ensemble Selection (DES) technique to become more proficient in selecting the most suitable classifier for each incoming chunk. Consequently, accuracy experienced a significant boost in later chunks, reflecting the adaptability and effectiveness of the ensemble approach. From chunk 20 to the last chunk, PA1 achieved the highest accuracy, whereas MLSMOTE recorded the lowest accuracy. PA1's superior performance of the PA1 is credited to its use of historical chunks to generate optimal nonoverlapping samples, thereby effectively training the pool classifiers. In Fig. 4.4 , the same dataset is employed with the DDM functioning as the drift detector. The radar plot illustrates results comparable to those of the prior experiment, whereas the line diagram underscores PA1's supremacy in most chunks. However, it also reveals that all approaches demonstrate diminished performance, in contrast to Fig. 4.3 (ADWIN), suggesting that ADWIN surpasses DDM when utilized in the Covertype data stream.

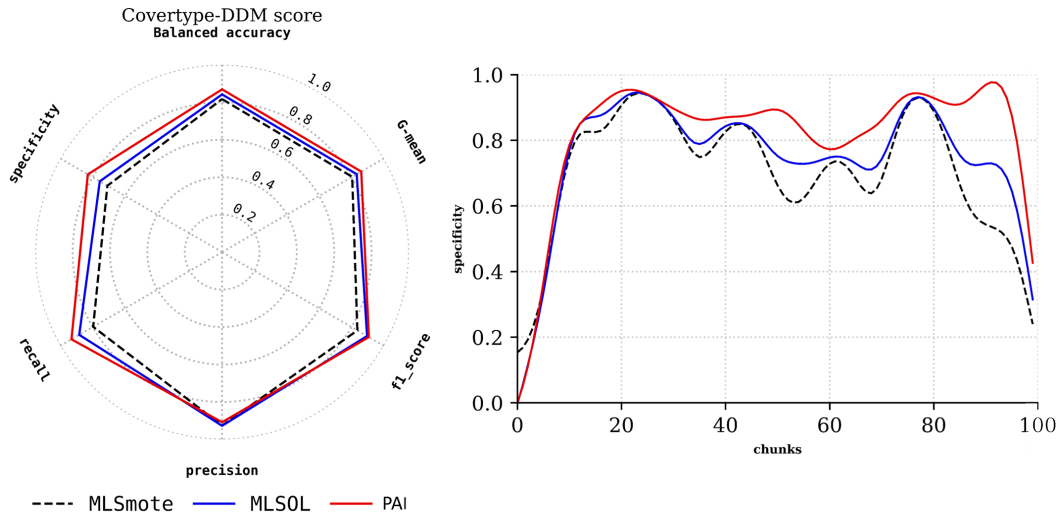



**Figure 4.4:** Performance of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Covertype Benchmark stream using the DDM concept drift detector.

#### 4.3.3.2 **Results on the Real Application Stream**

Fig. 4.5 illustrates the outcomes of employing the three methods on the Sensor data stream using ADWIN as the drift detector. The radar graph depicts metric values ranging from 0.6 to 0.8, which indicate the pronounced drift and imbalance of the Sensor stream. In terms of the precision and recall metrics, the three methods exhibited almost identical values, whereas PA1 stood out in the other metrics. In contrast, MLSMOTE and MLSOL displayed similar values. By examining the line graph in Fig. 4.5 , it is evident that during the initial 30 chunks, PA1's performance of PA1 might be suboptimal in some instances because of the limited number of classifiers in the pool. However, after the first 30 chunks, the performance significantly improved with the addition of more classifiers. In the same graph, PA1 consistently achieved the highest performance across all chunks, whereas MLSMOTE exhibited lower performance in certain chunks, and MLSOL exhibited lower performance in the other chunks. In Fig. 4.6, using the same dataset with the DDM as the drift detector, the radar plot metrics indicate lower results compared to the previous experiment. Nonetheless, the line graph underscores PA1's dominance of PA1 in most chunks, although all methods exhibit nearly identical performance

across the entire set of chunks. Overall, the performance of all the methods in Fig. 4.6 is inferior to that in Fig. 4.5 (ADWIN), which highlights ADWIN's superiority of ADWIN over DDM when applied to the Sensor data stream.

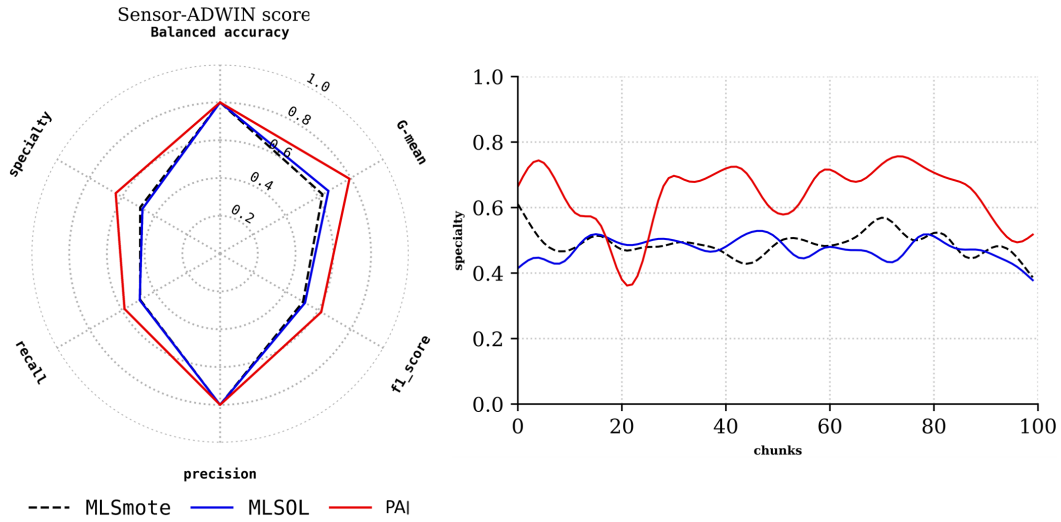



**Figure 4.5:** Performance of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Sensor real application stream using the ADWIN concept drift detector.

#### 4.3.3.3 **Results on the Synthetic Stream**

The results of applying the same methods on the synthetic data stream using ADWIN as the drift detector are presented in Fig. 4.7. The radar diagram indicates metric values ranging from 0.6 to 0.8, suggesting that the synthetic stream is prone to frequent drifts. While MLSOL exhibited the highest precision, MLSOTE exhibited the lowest values. Conversely, PA1 performed well in other metrics, and MLSMOTE demonstrated the least favorable values. Upon examining the line diagram in Fig. 4.7, it becomes evident that during the initial ten chunks, the PA1's performance might be suboptimal because of the limited number of classifiers in the pool. However, after the first ten chunks, the performance improved significantly with the inclusion of more classifiers. In the same diagram, PA1 consistently achieves the highest performance across all chunks, whereas MLSOL maintains satisfactory performance, and MLSMOTE exhibits lower performance throughout.

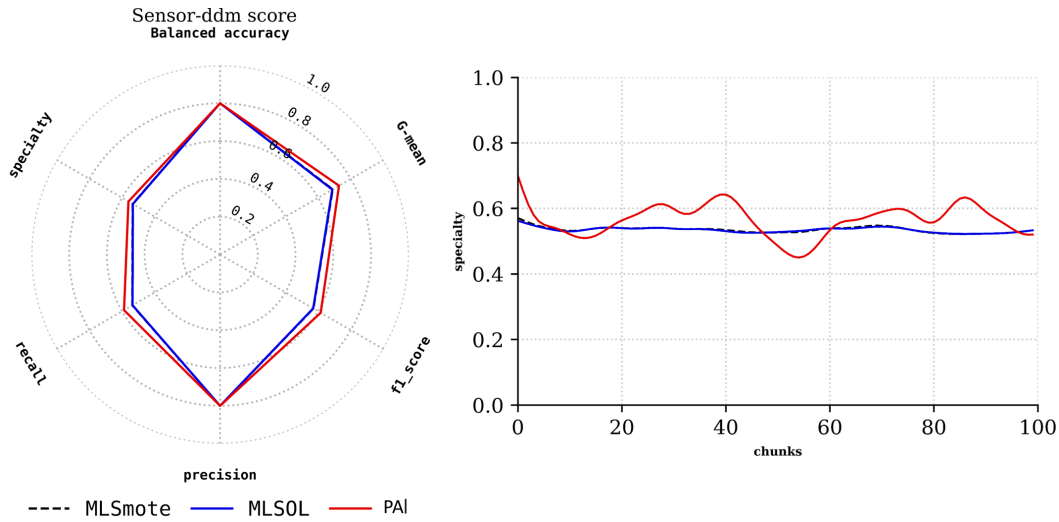



**Figure 4.6:** Performance of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Sensor real application stream using the DDM concept drift detector.

In Fig. 4.8, using the same synthetic data stream but with the DDM as the drift detector, the radar plot metrics produce similar results to the previous experiment. However, the line diagram emphasizes the same outcomes as those in the previous experiment. Nevertheless, MLSOL and MLSMOTE achieved lower values than the previous experiment. These findings confirmed ADWIN's superiority of ADWIN over DDM when applied to synthetic data streams. These results highlight the versatility and robustness of the algorithm, demonstrating its effectiveness in handling concept drift across diverse datasets, including the challenging synthetic dataset, Covertype dataset, and Sensor dataset, regardless of whether ADWIN or DDM is used as the concept drift detector.

### 4.3.4 **Analysis of Class Overlap Factor between Proposed Approach (PA1), MLSMOTE, and MLSOL Techniques.**

In our experimental study, we aimed to identify the critical factors that influence the selection of an optimal approach for minority classes in imbalanced drifted streams. These factors include various aspects, such as dataset characteristics, the

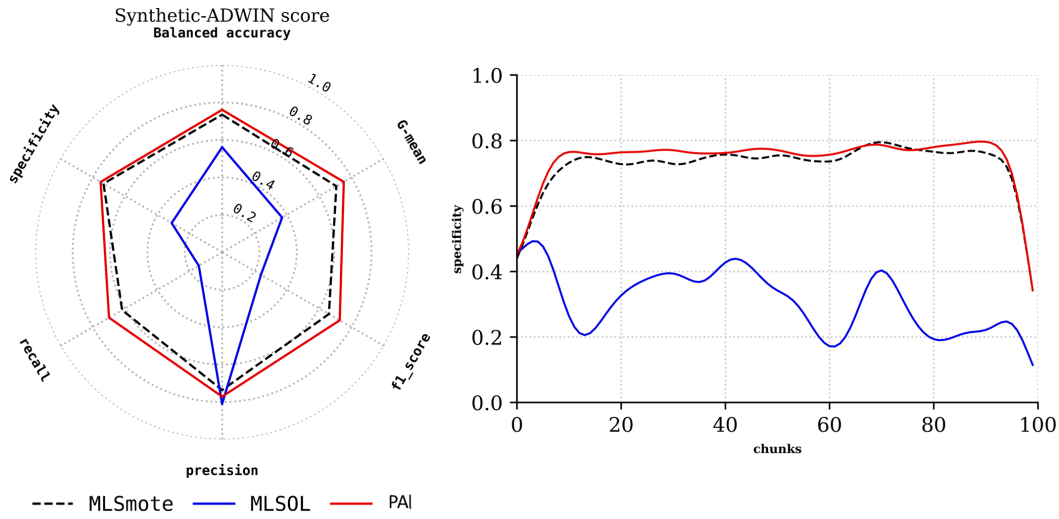



**Figure 4.7:** Performance of the firstProposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Synthetic stream using the ADWIN concept drift detector.

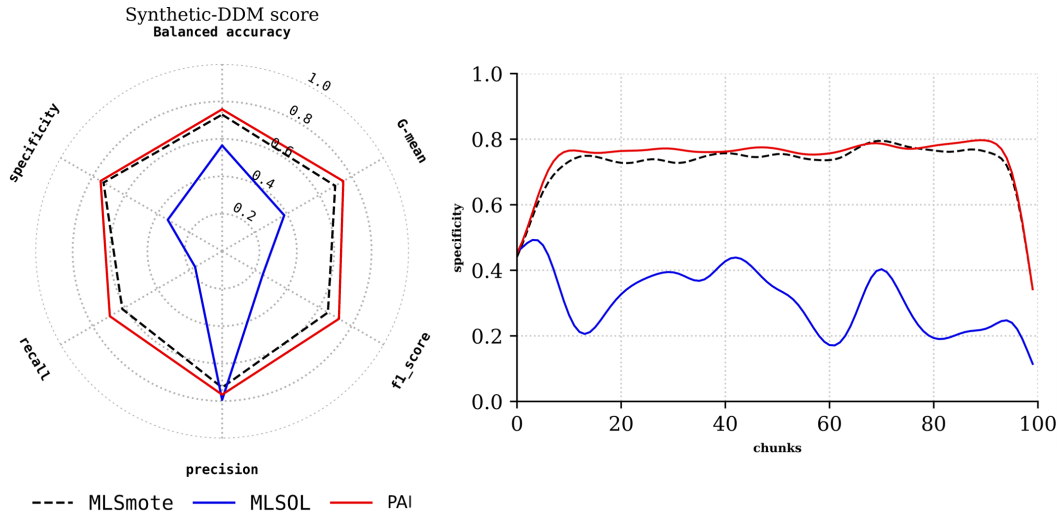



**Figure 4.8:** Performance of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Synthetic stream using the DDM concept drift detector.

choice of concept drift detectors, and the effectiveness of the algorithm in handling class overlap. To compare the overlapping class behavior of first proposed approach (PA1) with MLSMOTE and MLSOL, we systematically designed experiments organized into groups of ten chunks. The results were visualized in bar diagrams, contrasting PA1 with other methods (MLSMOTE and MLSOL). Fig-

ures 4.94.104.11 present ten groups, each with three bars representing MLSOTE, MLSOL, and PA1, respectively. Each bar visually represents overlapped samples for ten chunks of several data streams, considering distinct concept drift detectors, specifically ADWIN and DDM. Fig. 4.9 shows two diagrams for the ADWIN and DDM detectors. In the ADWIN diagram, the third bar group (chunks 20-30) lacks overlapping samples because this chunk range does not have drifts, and no synthetic samples are generated for the training step. The sixth and tenth groups have the highest overlapped samples because these groups experience many drifts, leading to the generation of several samples and, consequently, the data indicates that MLSMOTE displays the highest number of overlapping samples across all groups, while PA1 exhibits very few, except for the last group (90-100), which has a small number of overlapping samples. However, the DDM diagram has more overlapping samples than the ADWIN diagram because the DDM detector detects fewer drifts than ADWIN. Specifically, the last value on the Y-axis of the ADWIN diagram is 3,000 samples, while the last value on the Y-axis of the DDM diagram is 2,000 samples. Fig. 4.10 and Fig. 4.11 display the overlapping samples of the three methods on the Sensor and synthetic data streams, respectively. In Fig. 4.10, the ADWIN diagram demonstrates fewer overlapped samples than in Fig. 4.10, indicating a lower number of drifts in the Sensor stream. The overall diagram indicates that first proposed approach achieves fewer overlapped samples than MLSOL and MLSMOTE, with MLSMOTE having the highest number of overlapped samples. In the DDM diagram of Fig. 10, the number of overlapped samples is lower than in Fig. 4.9, with first proposed approach consistently achieving the lowest number of overlapped samples across most group bars. In Fig. 4.11, the ADWIN and DDM diagrams exhibit the greatest number of overlapped samples compared to the earlier figures. This suggests that the synthetic stream underwent frequent drifts and was more susceptible to noise. Consequently, the last value on the Y-axis in both the ADWIN and DDM diagrams was recorded for 7,000 samples. First proposed approach consistently achieves the lowest number of overlapped samples, whereas MLSMOTE consistently attains the highest number of overlapped samples across most group bars in both the ADWIN and DDM diagrams. In conclusion, our thorough examination of overlapping class instances across MLSMOTE, MLSOL, and

first approach consistently reveals minimal overlapped samples compared to other methods, regardless of whether DDM or ADWIN is employed as drift detectors

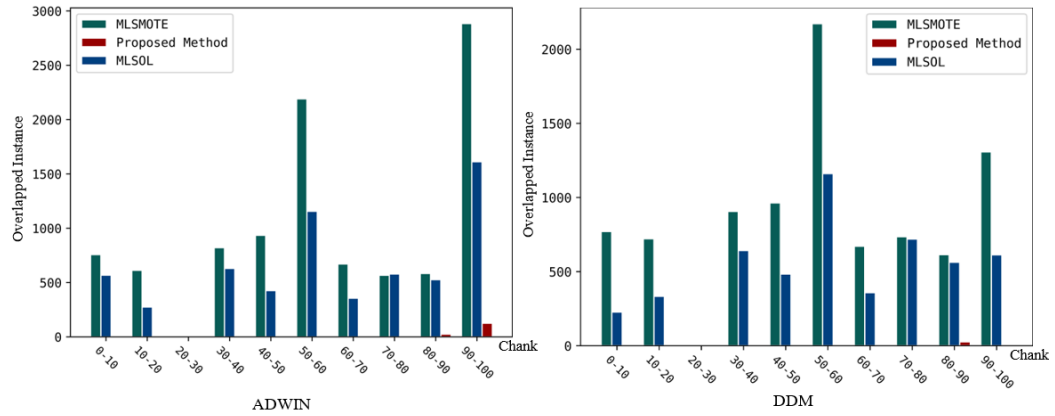



**Figure 4.9:** Overlapping generated points of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Covertype benchmark stream.

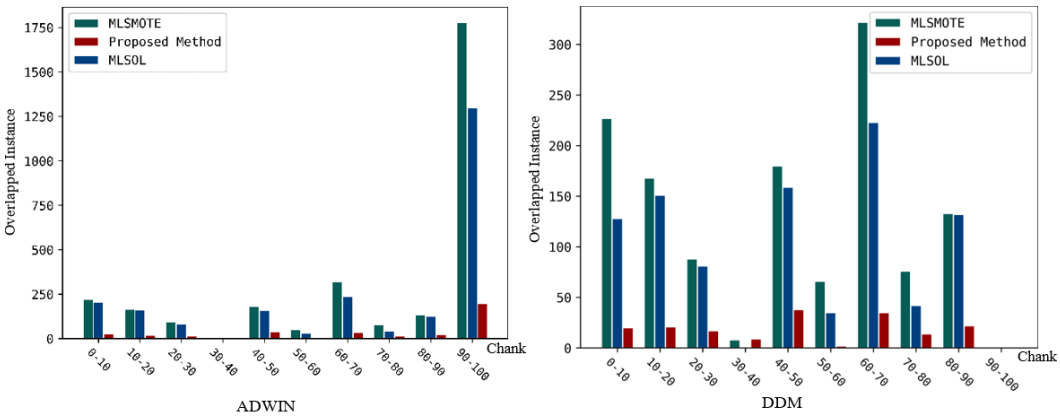



**Figure 4.10:** Overlapping generated points of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the Sensor real application stream.

### 4.3.5 Analyzing Runtime Factor Between the First Proposed Approach, MLSMOTE, and MLSOL Techniques

The results of our experiments indicate that the choice of the best algorithm for multiclass imbalanced streams depends on various factors, including dataset

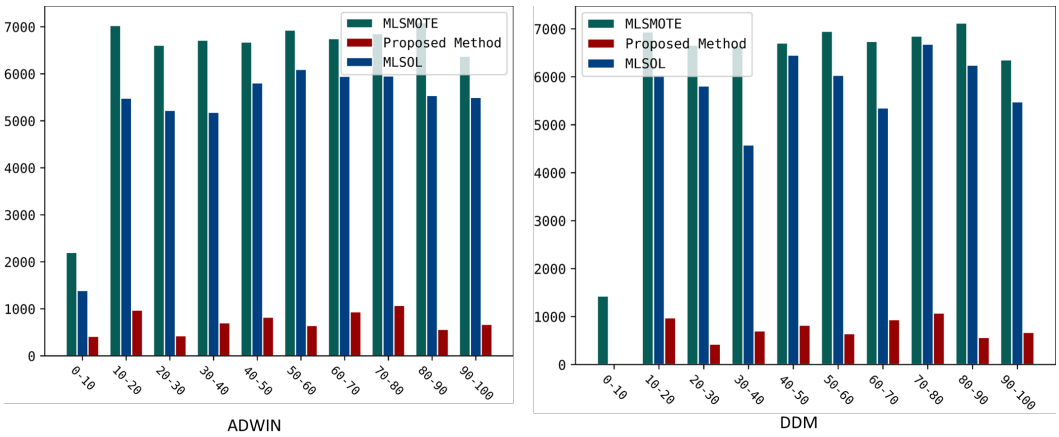



**Figure 4.11:** Overlapping generated instances of the first Proposed Approach (PA1), MLSMOTE, and MLSOL techniques on the synthetic stream.

characteristics, the concept drift detector used, the presence of an overlapping class problem, and the algorithm's runtime demands. To investigate the runtimes of MLSMOTE, MLSOL, and first approach, we conducted experiments, as shown in Table 4.2. Our findings reveal that first proposed approach is highly efficient, regardless of whether ADWIN or DDM is used as the concept drift detector. Specifically, when ADWIN was used, the first proposed approach algorithm took 8344 s to train and predict the Covertype stream for ensemble classifiers, whereas it took 8019 s when using the DDM detector. In contrast, MLSMOTE and MLSOL require more time for the same task. Notably, first approach maintains efficiency, even when using the DDM concept detector. Additionally, first approach demonstrates shorter processing times in the Sensor stream and synthetic data compared with other methods. Consistently, the DDM detector achieved less time across all experiments owing to its lower detection of drifts compared to the ADWIN detector. This is because there are fewer instances that trigger training for pool classifiers when using the DDM. The bold highlighting in Table 4.2 further emphasizes the efficiency of first approach with both the ADWIN and DDM detectors across all dataset streams. Because our proposal generates fewer overlapped samples, leading to an overall decrease in the running time.

**Table 4.2:** Runtime of MLSMOTE, MLSOL, and the first Proposed Approach (PA1).

| Stream | Concept Drift Detector | MLSMSOTE | MLSOL | PA1 |
|---|---|---|---|---|
| Benchmark | ADWIN | 9559 | 8655 | **8344** |
| | DDM | 8388 | 8031 | **8019** |
| Real Application | ADWIN | 1291 | 1310 | **1102** |
| | DDM | 585 | 607 | **521** |
| Synthetic | ADWIN | 12870 | 4866 | **4834** |
| | DDM | 12397 | 4958 | **4687** |

### 4.3.6 Analyzing Non-parametric Tests between the First Proposed Approach, MLSMOTE, and MLSOL Techniques

We conducted a thorough series of statistical analyses encompassing 12 comparisons across three diverse datasets, three methods, and two drift detectors. These analyses were rigorously evaluated using a non-parametric test, specifically the Kruskal-Wallis test. The results of this test were striking, revealing substantial variations in the G-mean measurements across most experiments. Importantly, these differences were not due to random chance [37]. Upon closer examination of the assessments for the three methods, PA1, MLSMOTE, and MLSOTE, as detailed in Table 4.3, we found compelling evidence supporting the acceptance of the null hypothesis (H0). This implies that the expected and observed data exhibited statistically significant disparities. H0 is rejected when significant differences are not observed and H0 is accepted if the P-value falls below the critical value. Underlining the significance of our analysis, the Kruskal-Wallis test was conducted with a 95% confidence level, and the P-value was rounded to the first three digits after the decimal point. Nevertheless, it is important to note that a similarity in the performances of the methods emerged in the third and fourth experiments, resulting in the rejection of H0. This similarity can be attributed to the fact that the DDM detected fewer drifts in these experiments.

**Table 4.3:** Kruskal-Wallis test results for MLSMOTE, MLSOL, and PA1.

| Dataset | Drift detector | Comparison | P-value | Critical value | H0 |
|---|---|---|---|---|---|
| Covertype stream | ADWIN | PA1 - MLSSMOTE | 0.001 | 0.05 | Accept |
| | | PA1 - MLSOL | 0.014 | 0.05 | Accept |
| | DDM | PA1 - MLSSMOTE | 0.361 | 0.05 | Rejected |
| | | PA1 - MLSOL | 0.401 | 0.05 | Rejected |
| Sensor stream | ADWIN | PA1 - MLSSMOTE | 0.001 | 0.05 | Accept |
| | | PA1 - MLSOL | 0.001 | 0.05 | Accept |
| | DDM | PA1 - MLSSMOTE | 0.001 | 0.05 | Accept |
| | | PA1 - MLSOL | 0.001 | 0.05 | Accept |
| Synthetic stream | ADWIN | PA1 - MLSSMOTE | 0.001 | 0.05 | Accept |
| | | PA1 - MLSOL | 0.001 | 0.05 | Accept |
| | DDM | PA1 - MLSSMOTE | 0.001 | 0.05 | Accept |
| | | PA1 - MLSOL | 0.001 | 0.05 | Accept |

CHAPTER

# 5

# Addressing Emerging New Classes in Incremental Streams via Concept Drift Techniques

In various real applications, data streams have introduced new challenges to learning algorithms. Data streams are continuous with high volumes of data arriving rapidly and dynamically. This data deluge poses unprecedented challenges for learning algorithms because they must adapt to the dynamic nature of the data environment [1–3]. Among the various research areas in machine learning, sequential learning under data Streams with Emerging New Classes (SENC) has garnered considerable attention because of its practical relevance and unique challenges [106], [81], [83]. SENC refers to a scenario in which new classes that were not present during the initial training of a learning model emerged in the data stream. This poses a significant challenge for traditional learning approaches that are typically designed to handle fixed or predefined class distributions. The ability to effectively recognize and adapt to these novel classes in real-time is crucial for maintaining accurate and up-to-date models. Furthermore, the inherent limitations of data streams, such as limited memory and storage constraints, impose additional complexities in the learning process. Learning algorithms must operate efficiently within these resource constraints to ensure real-time processing and

to avoid overwhelming computational overhead. To address the challenges presented by data streams containing emerging new classes, dynamic ensemble selection (DES) [20, 23, 71]. Dynamic ensemble selection involves utilizing multiple classifiers in machine learning to make collective predictions or classifications of data. Dynamic ensemble selection is distinguished by its ability to dynamically adapt the ensemble based on the characteristics of the data. Instead of relying on a fixed ensemble of classifiers, dynamic ensemble selection continuously assesses the performance of individual classifiers and selects the subset that demonstrates the highest competence for the current data. This adaptability enables the ensemble to enhance its performance over time by incorporating the most suitable classifiers for prevailing data conditions. Furthermore, if a classifier becomes ineffective owing to concept drift or the emergence of new classes, it can be excluded from the ensemble to prevent it from negatively affecting the overall performance. Adaptive Windowing (ADWIN) is another widely employed method to address concept drift. Concept drift refers to the phenomenon in which the statistical properties of data change over time, leading to a decline in the learning algorithm performance [33, 67, 69]. ADWIN continuously monitors incoming data and detects changes or drifts in data distribution. It achieves this by maintaining sliding windows of variable sizes and monitoring statistical measures such as the mean or variance within these windows. Upon detecting a significant change or drift, the ADWIN triggers an update in the ensemble or classifier configuration. This update may involve retraining the classifiers with new data or incorporating new classifiers that are better suited to the updated data distribution. By adapting the ensemble to changing data conditions, ADWIN ensures that the classifier system remains accurate and up-to-date even in the presence of concept drift or the emergence of new classes. The primary objective of utilizing these approaches, such as dynamic ensemble selection and ADWIN, is to establish a flexible and effective classification system that is capable of handling data streams containing emerging new classes. By dynamically adjusting the ensemble based on the data characteristics and detecting and adapting to concept drift, these approaches enable the system to maintain accurate predictions over time. Adaptability and responsiveness are crucial for addressing the unique challenges posed by the dynamic nature of data streams and the emergence of new classes within them.

The subsequent sections of this paper adhere to a well-structured organization. Section 5.2 introduces the second proposed approach, providing intricate explanations of its constituents, including dynamic classifier ensembles, concept drift handling, and emergency class identification, and the adaptive proposed method of the emerging pool size. Section 5.3 outlines the experimental setup and presents the results, providing details regarding the employed datasets, evaluation metrics, and procedures. Finally, in Section **??**, we offer concluding remarks, Highlighting the main discoveries, examining their significance, and suggesting possible directions for future investigations.

## 5.1 Motivations and Contributions

This research proposes efficient algorithms for classifying data streams in real-time scenarios, focusing on addressing the issues caused by emerging new classes in data distributions. We aim to develop novel algorithms that can effectively handle the emergence of new classes issue in dynamic data streams, achieve high accuracy of classification, and reduce the complexity of computations. The key contributions of this study are outlined as follows:

1. Our first contribution involves utilizing the ADWIN, DES, and K-means techniques in combination with the ensemble stratified bagging technique to detect and adapt to emerging new classes. This approach allows us to dynamically update the classification model to accommodate an evolving data environment.
2. The second contribution is the introduction of an adaptive method to adapt the emerging pool size depend on the stream distribution.

## 5.2 Proposed Methodology

In this section, we outline the key stages of our research, which consist of three distinct steps designed to address the challenges of handling emerging new classes in changeable data streams. Second proposed approach aims to develop a robust framework capable of tackling these complex challenges through a comprehensive, three-step methodology.

1. **Emerging new classes:** Our study addresses the widespread issue of emerging new classes in drifted streams using well-known techniques, including concept drift detection and K-means clustering.

2. **Drifted streams:** To address changes in the data distribution, our approach integrates a concept drift detector that dynamically identifies shifts, allowing the model to promptly adapt its classifiers to maintain effectiveness.

3. **Classifier performance:** Classifier Performance: We utilize Dynamic Ensemble Selection (DES) to boost classifier performance. This method involves constructing a pool of classifiers and dynamically choosing the most appropriate one for each new data point, thereby enhancing both accuracy and robustness.

### 5.2.1 Second Proposed Approach Flow

Second proposed approach is composed of three main phases that work together to manage emerging new classes and drifting data streams effectively:

1. **Dynamic ensemble selection:** The initial phase, DES, identifies the most suitable classifier for each incoming data chunk to ensure optimal performance.

2. **Drift detector phase:** In this phase, the system continuously observes the data stream in real time to detect concept drift, indicating changes in the underlying data distribution.

3. **Emerging new class identifier phase:** The final phase identifies unknown classes in drifted streams, creating new classifiers for emerging classes to improve the model's proficiency in accurately classifying new instances.

As depicted in Fig. 5.1, DES extracts the current data chunk and utilizes the DES technique to identify the most effective classifiers for that chunk (black box of Fig. 5.1). These classifiers are used in the second phase to predict class labels, while drift detectors like ADWIN or DDM monitor for concept drift. If a drift is detected (highlighted by the red rectangle), the chunk is forwarded to the third phase, where

new classes are identified (see Fig. 5.2). The overall approach is implemented in Algorithm 3, which takes a data stream and a DES Pool threshold as inputs. Key components include training the initial ensemble (Lines 4-6), using DES to select the best classifier for the current chunk (Line 8), detecting drifted chunks (Line 10), creating new classifiers for emerging classes (Line 12), and removing the worst classifier if the DES Pool threshold is exceeded (Lines 14-16).

### 5.2.2 Emerging New Classes Phase Details

In this section we present the Emerging new classes phase details. As shown in Fig.2, The emerging phase contains two primary steps:

1. **Emerging new classes identifier:** The emerging class detection step plays a crucial role in the second proposed approach (the black rectangle of Fig. 5.2) by utilize the K-means technique to clustering the drifted chunk into a set of clusters. And identifying new emerging classes by evaluating the distances between instances and their nearest class centroid. This process involves comparing the distance between an instance and the nearest class centroid with the maximum distance between class centroids ($d_c$), as described in Eq.5.1 and Eq.5.2, where ED represents the Euclidean Distance method and $c_i$, $c_j$ denotes the centroids of the current classes. If the calculated distance ($d_x$) exceeds this threshold ($d_c$), this indicates the presence of a new emerging class, denoted by EC in Eq. 5.3; otherwise, it is $P_DES$ (prediction class of the new instance).

2. **Adaptive the emerging classes pool size:** This critical step adapts the pool size based on the emergence rate of new classes and drift distribution, using statistical measures like standard deviation, first derivative, and average historical drift updates.

The Emerging New Classes phase is implemented in Algorithm 4, which utilizes drifted instances and current changes as inputs. Key steps include applying K-means (Line 1), calculating cosine similarity between centroids (Line 2), determining the maximum centroid distance (Line 3), setting the adaptive pool threshold (Lines 4-7), using KNN to find the nearest neighbor (Line 9), storing instances in

the emerging pool if the distance exceeds the threshold (Lines 11-12), and creating a new classifier if the pool size surpasses the adaptive limit (Lines 14-17). A simulated scenario is illustrated in Figures 5.3 and 5.4. In Fig. 5.3 (a), the drifted chunk is divided into three clusters using the K-means algorithm. Fig. 5.3 (b) shows the calculation of the maximum distance between cluster centroids, which serves as a threshold for identifying new classes. The procedure for classifying instances is as follows: when a drifted instance is detected, it is considered a new class if the distance between the instance and its nearest centroid exceeds the maximum inter-centroid distance Fig. 5.4 (a). If the distance is within the threshold, the instance is classified as a known class Fig. 5.4 (b).

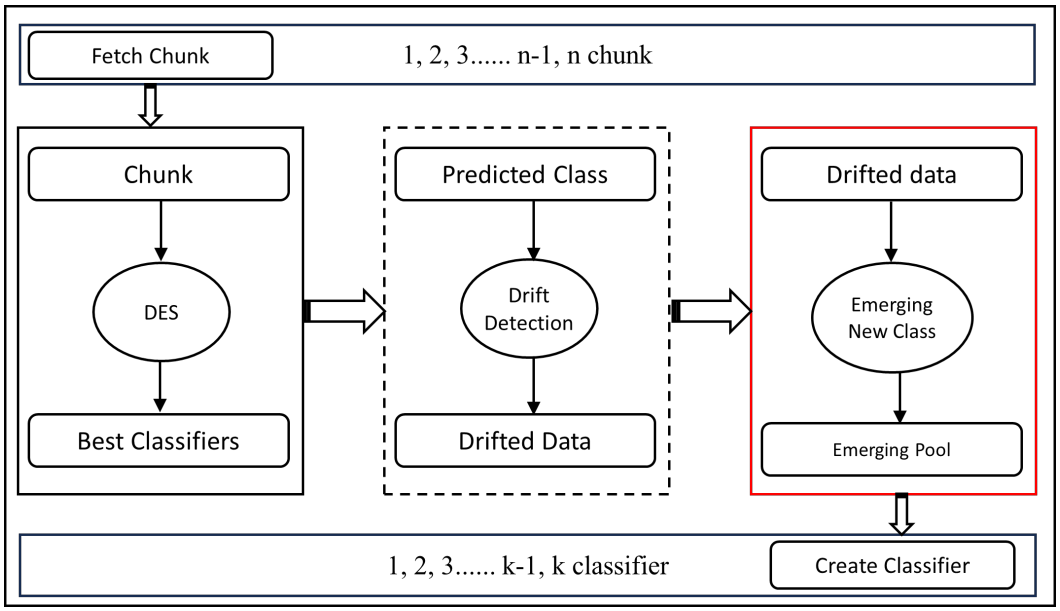



**Figure 5.1:** Flow of the second proposed approach for emerging drifted data.

$$d_c = \arg\max_{d} \sum_{i=1}^{i} \sum_{j=i+1}^{i} ED(c_i, c_j) \tag{5.1}$$

$$d_x = \arg\min_{i} \sum_{i=1}^{i} ED(x, c_i) \tag{5.2}$$

$$p = \begin{cases} EC & \text{if } d_x > d_c \\ P_{DES} & \text{otherwise} \end{cases} \tag{5.3}$$

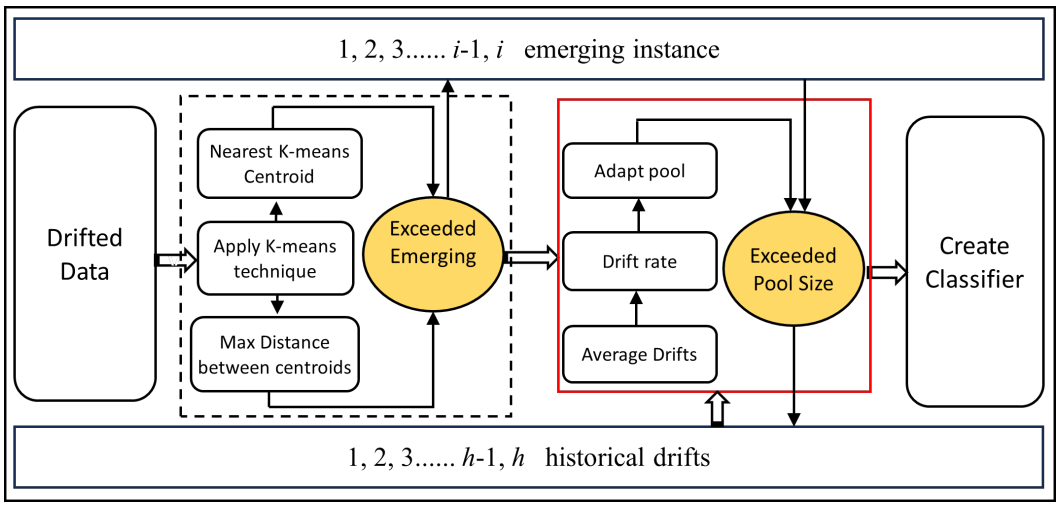



**Figure 5.2:** Flow of the Emerging Phase.

## 5.3 Experimental Results

The experiments conducted in this study were designed to assess the effect of concept drift on the performance of the second proposed approach in detecting emerging new classes. The primary goal was to identify the machine learning algorithm in conjunction with DES, Dynamic Ensemble Selection technique, which improves the performance of classification when faced with the emergence of new classes. By conducting these experiments, valuable insights were gained, leading to potential enhancements in the effectiveness of the second proposed approach in managing imbalanced data streams and its overall performance. These experiments provide valuable insights into the capabilities of the framework and shed light on the optimal approach for handling emerging new classes in the presence of concept drift. The results provide valuable guidance for selecting the most suitable classification machine learning algorithm and DES configuration aims to enhance both classification accuracy and robustness. The experiments detailed in this study offer crucial insights into improving the second proposed approach's performance and tackling challenges related to emerging new classes and concept drifts in incremental drifted streams. These insights are instrumental in advancing stream mining techniques and developing more precise and resilient classification models for dynamic and evolving data-stream environments. The two main questions to be

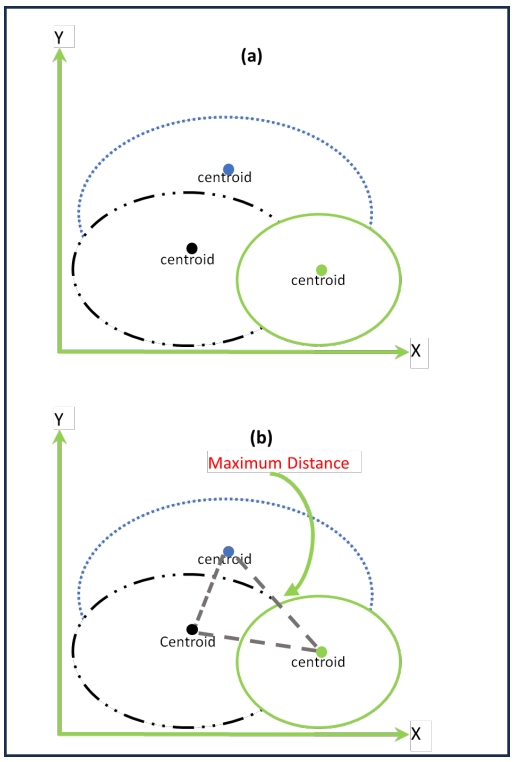



**Figure 5.3:** Example steps for a dataset with two features (x, y) and three classes (Blue, Black, Green): (a) Use the K-means algorithm to cluster the current chunk. (b) Determine the maximum distance between class centroids.

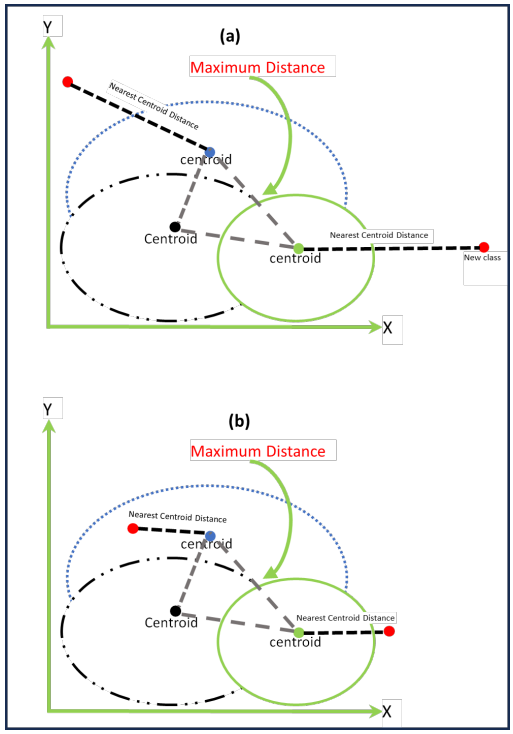



**Figure 5.4:** Example steps for new points (red points) with two features (x, y): (a) Identify new incoming instances (represented by red points) as potential new classes. (b) Determine whether the new instance is part of a nearby existing class or belongs to an emerging new class.

answered are:

- $Q_1$.How does the emergence of new classes in data streams affect the stability and performance of ML models?
- $Q_2$. how to employee concept drift to solve the emerging new classes problem?

### 5.3.1 Experimental Setup

The evaluation of the second proposed approach involves a comparison with SENCForest [82], SENNE [1], and KENNES [85]. The evaluation utilizes several metrics, including recall, precision, F1 score [107], BAC [108], and G-mean [109]. The procedure for experimentation followed a test-then-train technique [110], where the classifier is first trained on a specific chunk and then evaluated on the subsequent chunk. Each chunk was standardized to 2000 instances. Four different classification models served as base estimators: Gaussian Naive Bayes (GNB) technique, K-Nearest Neighbors (KNN), the Support Vector Classifier (SVC), and the Hoeffding Tree (HT), all features are implemented using scikit-learn [111]. We establish an ensemble classifier pool with a set limit of L = 8, wherein each ensemble consists of N = 4 base models. While these constraints remained fixed across all our experiments, the threshold for the pool classifier in each approach was maintained at eight. Consequently, if the threshold is surpassed, the least-performing classifier is systematically eliminated. ADWIN [69] and DDM [33] are employed as concept drift detectors as implemented in the River library [1] , to identify concept drifts, as they are considered more accurate techniques for use in incremental data streams [33, 66, 67, 69]. This configuration was applied consistently across all approaches to ensure fair engagement. The experiments were conducted using Python, and the source code is available publicly on GitHub [2].

### 5.3.2 Data Streams

In this section, the performance of the second Proposed Approach (PA2), was evaluated using several datasets, including synthetic data streams, a real-world application stream, and benchmark datasets. To conduct the evaluations, the stream-learn and River python libraries [111] were used. As detailed in Table 5.1, the Covertype dataset stream is used as the benchmark dataset in the study. This dataset comprises 52 features, 7 classes, and a total of 581,010 instances. It serves as a

[1] https://riverml.xyz/0.21.0/introduction/basic-concepts/

[2] https://github.com/Amadkour/transfer_learning_with_concept_drift.git

standard benchmark dataset widely used in stream mining research. For the evaluation of real application streams, the Sensor stream dataset was utilized. This dataset includes 5 features, 58 classes, and 392,600 instances in total, reflecting a real-world application scenario. Synthetic dataset stream was generated using the scikit-learn Python package [111]. This synthetic stream was generated to mimic data streams and assess the performance of the second proposed method. It included 10 features and four classes, divided into 200 chunks, each with a size of 2,000. The effectiveness of the second proposed method was rigorously assessed using these datasets along with a stream-learn library. This thorough evaluation offered important insights into how well the approach manages various types of data streams, encompassing benchmark datasets, synthetic data streams, and real-world application streams.

**Table 5.1:** The datasets utilized in the experiments exhibited diverse characteristics.

| Data stream | Features | Classes | Instances | Chunk Size |
|---|---|---|---|---|
| Sensor stream [1] | 5 | 58 | 392600 | 2000 |
| Covertype stream[2] | 52 | 7 | 581010 | 2000 |
| Synthetic stream | 10 | 4 | 200000 | 2000 |

### 5.3.3 Compared Approaches

In the following section, we present a comparative analysis between our proposed GNB methodology and three established benchmark techniques, detailed as follows:

1. **SENCForest [82]:** The SENCForest method employs anomaly detection techniques to identify new classes and is founded on entirely random trees. It constructs isolation trees for both classification and detection by subsampling from each class, utilizing 20 random trees and a subsample size of 20. Although it can operate with incomplete or no label information, it frequently suffers from elevated false positive rates and suboptimal runtime efficiency in practical applications.

2. **SENNE [83]:** Utilizing a hypersphere ensemble mechanism, SENNE calculates scores to identify both new and known classes. It operates with a buffer capacity of 100 and sets a new class-score threshold of 0.5.

3. **KENNES [85]:** The KNNENS method addresses the dual challenge of detecting new classes and classifying known classes within a unified framework. It was configured with a buffer size of 100 and a new class score threshold of 0.5.

### 5.3.4 Examination of Experimental Findings

This section offers an in-depth analysis of the second approach's performance across various data streams. To ensure a thorough evaluation, five performance metrics— recall, F1 score, precision, recall, G-mean, and BAG—are displayed through two types of visual diagrams: radar and line charts. The radar chart effectively summarizes the performance of each algorithm by highlighting their performance across the six key metrics. Additionally, to evaluate the overall performance of each method—including PA2, SENCForest, SENNE, and KENNES—the average value for each metric was computed. Additionally, a line diagram using the G-mean metric was used to compare these methods for each chunk across 200 chunks. A radar diagram provided a detailed and nuanced evaluation of the second approach's performance across various experimental scenarios.

#### 5.3.4.1 Results On The Benchmark Dataset

This experiment aimed to evaluate the performance of the second approach on the Covertype data stream using different buffer sizes with ADWIN as the concept drift detector. The findings are visually represented using scatter plots in Fig. 5.5, which contains four subplots for emerging buffer sizes of 5, 10, 20, and adaptive sizes, respectively. In Fig. 5.5(b), the G-mean accuracy of SENCForest, SENNE, KENNES, and PA2 is shown across 200 chunks, specifically focusing on a buffer size of 5 instances for emerging new classes. At first, all methods exhibit less-than-ideal accuracy in the initial 20 chunks. However, a marked improvement is observed after this period, as the classifier pool grows. This growth bolsters the

DES, dynamic ensemble selection, technique, allowing it to choose the most suitable classifiers for subsequent chunks. Consequently, there are significant gains in accuracy from chunk 20 onward, PA2 consistently achieved the highest accuracy, while KENNES and SENNE recorded the lowest. Additional experiments with buffer sizes of 10 and 20, shown in Fig. 5.5(b) and Fig. 5.5(c), revealed slightly lower accuracy compared to the buffer size of 5, likely due to fewer updates when using larger buffer sizes. Finally, the adaptive buffer size experiment in Fig. 5.5(d) highlights the PA2 algorithm's superior performance across all chunks, with the adaptive emerging pool size delivering the best results.

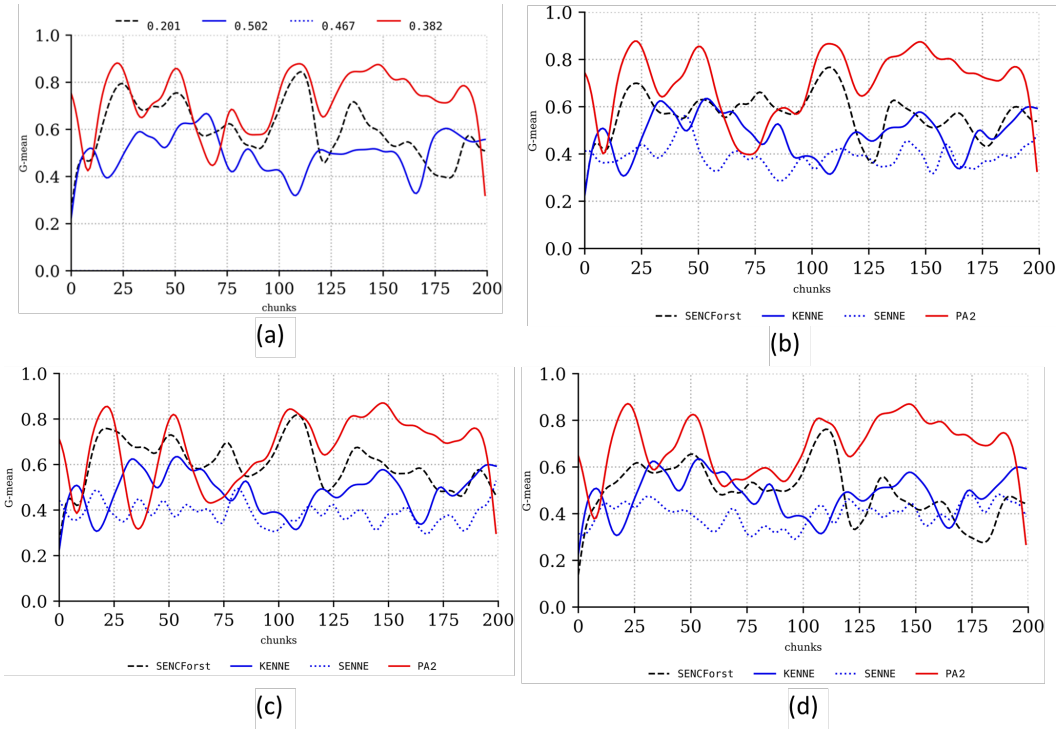



**Figure 5.5:** Covertype stream for various emerging buffer sizes: (a) buffer size of 5, (b) buffer size of 10, (c) buffer size of 20, and (d) adaptive buffer size.

#### 5.3.4.2 **Results On Real Application Stream**

This experiment focused on assessing the accuracy of SENCForest, SENNE, KENNES, and PA2 across 200 chunks of a sensor data stream, utilizing varying buffer sizes similar to the previous experiment. The results, shown in Fig. 5.6,

were visualized using scatter plots. Fig. 5.6 (a) displays the classification accuracy for each algorithm with a buffer size of 5. The PA2 algorithm achieved the highest performance overall, except for chunks 43 to 48, where some instances were incorrectly assumed as outliers rather than emergent. In contrast, SENCForest and SENNE had the lowest accuracy. Additional experiments with buffer sizes of 10 and 20 (Fig. 5.6(b) and Fig. 5.6(c)) demonstrated reduced accuracy compared to the buffer size of 5, attributed to fewer updates with larger buffer sizes. The adaptive buffer size experiment in Fig. 5.6(d) reaffirmed the effectiveness of the PA2 algorithm, which outperformed others across all chunks, with the adaptive emerging pool size providing optimal results.

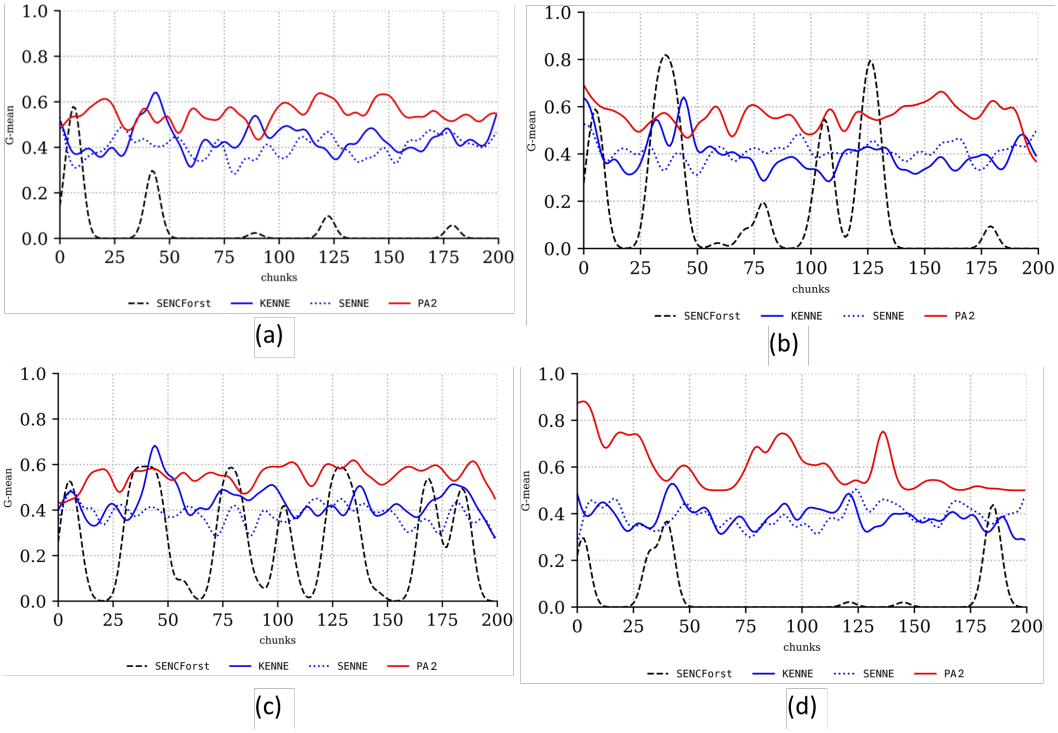



**Figure 5.6:** Sensor stream for various emerging buffer sizes: (a) buffer size of 5, (b) buffer size of 10, (c) buffer size of 20, and (d) adaptive buffer size.

#### 5.3.4.3 **Results On Synthetic Stream**

This experiment aimed to assess the performance of the previously tested methods on a synthetic data stream. Results, depicted in Fig. 5.7, were analyzed using

radar and scatter plots. Fig. 5.7(a) shows the performance of various algorithms using a buffer size of 5, with the radar plot highlighting PA2's superior performance. Scatter plots illustrate that PA2 consistently delivered the highest accuracy when applied to the synthetic stream with buffer sizes of 5, 10, and 20, as shown in Fig. 5.7(a), Fig. 5.7(b), and Fig. 5.7(c), respectively. Fig. 5.7(d) demonstrates that the adaptive pool size yielded the best accuracy on the synthetic stream, underscoring the adaptability and efficiency of the PA2 algorithm.

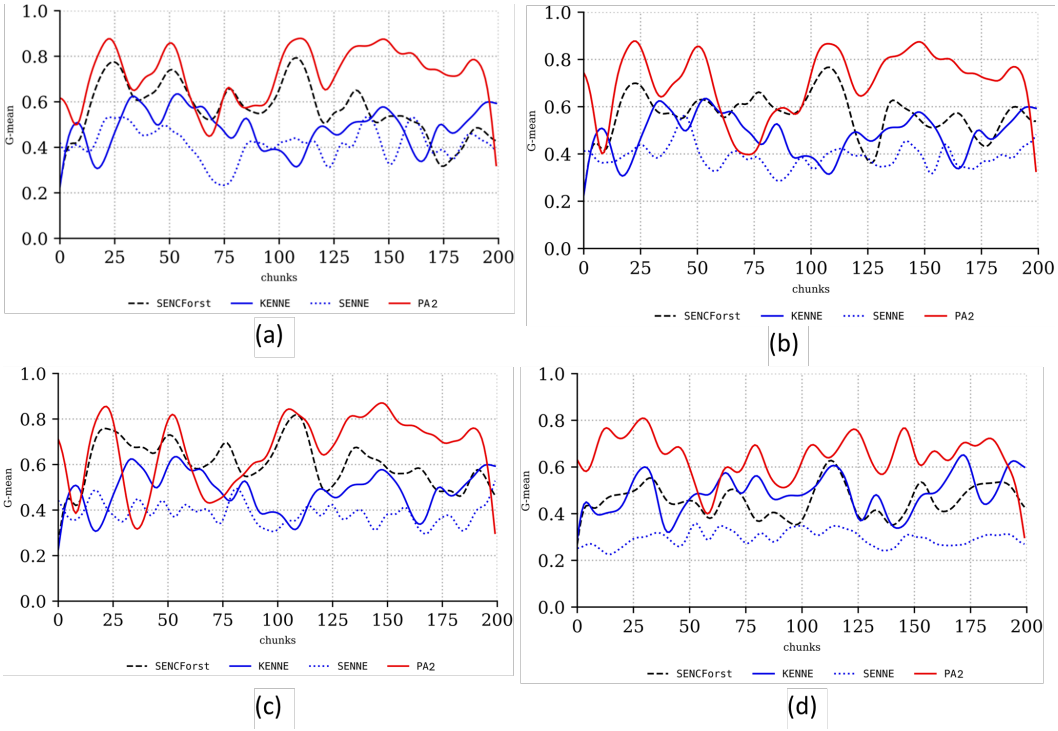



**Figure 5.7:** Synthetic stream for various emerging buffer sizes: (a) buffer size of 5, (b) buffer size of 10, (c) buffer size of 20, and (d) adaptive buffer size.

### 5.3.5 Runtime Analysis Of The Best Algorithms

Our experimental results revealed that selecting the optimal algorithm for detecting emerging new classes is influenced by various factors, including dataset characteristics, buffer length, and the algorithm's runtime requirements. We conducted a series of experiments to evaluate the runtime performance of different algorithms and observed that the PA2 algorithm demonstrated outstanding efficiency.

Specifically, with the adaptive pool size approach, the PA2 algorithm trained bagging classifiers 150 times within 2131 seconds when using the Covertype stream, outperforming others despite SENCForest achieving the lowest runtime but with fewer updates compared to PA2 as shown in Table 5.2. On the Sensor stream, PA2 delivered the best runtime for 66 updates, while SENCForest, KENNES, and SENNE required more time to complete the same or fewer iterations. Similarly, PA2 exhibited superior runtime performance on the Synthetic stream. These results underscore the PA2 algorithm's efficiency, making it an ideal choice for time-constrained scenarios. The superior runtime performance of PA2 is further highlighted in bold in Table 5.2.

### 5.3.6 Comparison between GNB, KNN, and HT, SVC

In this section, we compare various algorithms (KNN, SVC, GNB, and HT) as base classifiers for the bagging technique. The comparison, illustrated in Fig.5.8, focuses on the Covertype dataset stream, where GNB (blue line) and HT (red line) consistently achieve the highest scores across most chunks. Both classifiers also excel in radar plots across key performance metrics, including accuracy, precision, recall, and F1-score. Based on these results, GNB and HT emerge as the most suitable base classifiers for the bagging technique. We further compared GNB and HT in terms of runtime and update frequency, as shown in Table 5.3. The results indicate that GNB has the best runtime performance, while HT demonstrates superior accuracy, likely due to its higher number of updates compared to GNB, as reflected in Table 5.3 and Fig. 5.8.

### 5.3.7 Comparison between ADWIN and DDM

In this section, we compare two concept drift detection methods: the Drift Detection Method (DDM) [33] and the Adaptive Window (ADWIN) [33, 69]. These methods are widely recognized for their excellent performance in handling incremental drifted streams [33, 66, 67, 69]. As shown in Fig. 5.9, using the Synthetic stream with GNB as the base classifier, ADWIN consistently outperforms DDM in both radar and line plots across all performance metrics, including accuracy. Furthermore, when comparing their runtime and update frequency, as presented in

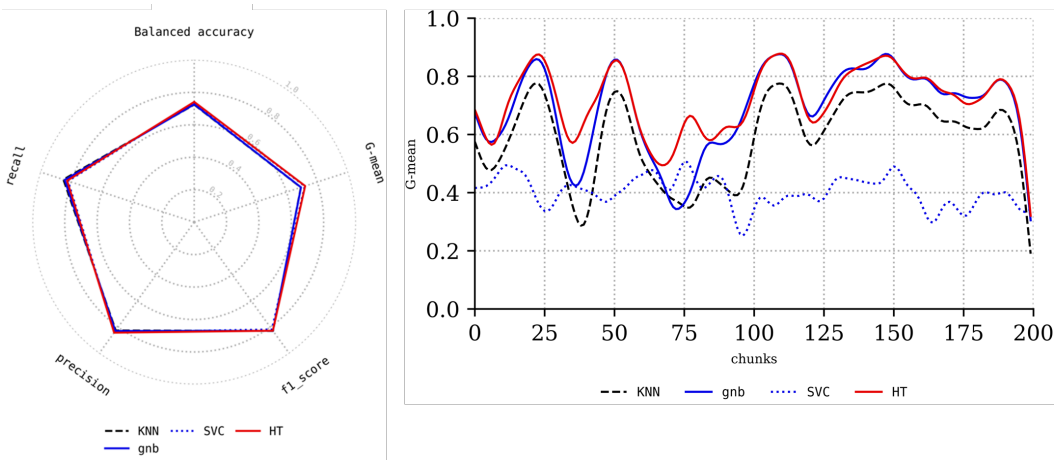



**Figure 5.8:** Covertype stream for concept drift detectors (ADWIN, DDM) evaluated using several metrics: recall, precision, balanced accuracy, G-mean, and F1 score.

Table 5.4, ADWIN proves to be the superior detector for drifted streams. It identifies the highest number of drifts (139) in the least amount of time (272 seconds), demonstrating its efficiency and effectiveness.

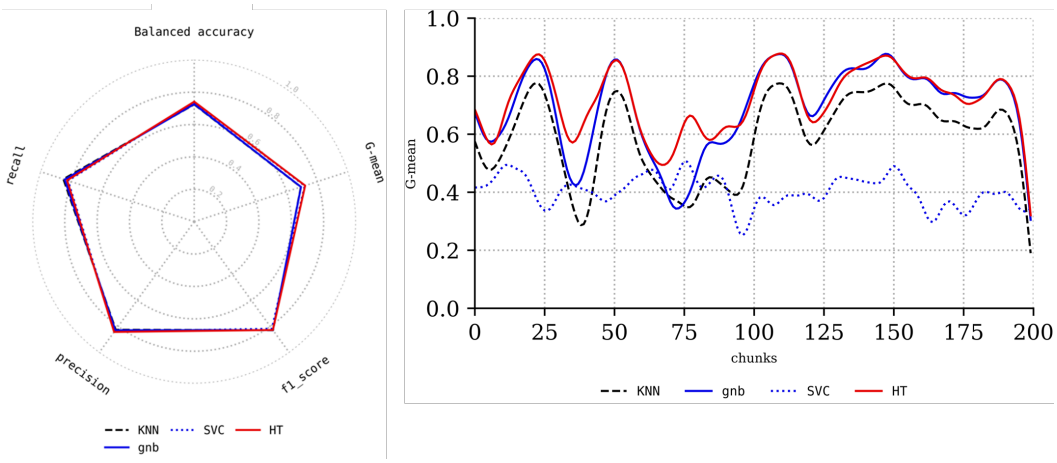



**Figure 5.9:** Synthetic stream for concept drift detectors (ADWIN, DDM) evaluated using several metrics: recall, precision, balanced accuracy, mean, and F1 score.

**Algorithm 3:** Second proposed approach algorithm for emerging drifted data streams.

**Input:** data stream, classifier threshold $\kappa$
**Data:** current chunk $a$, classifier $k$, classifiers pool $\Psi$, drifted pool $\psi$
**Output:** prediction $p$
$\psi \leftarrow \phi$
$\Psi \leftarrow \phi$
**for** *stream has chunk* **do**
  **if** a *is the first chunk* **then**
    $k \leftarrow$ trainingNewClassifier$(a)$
    $P \leftarrow$ getPrediction$(a, k)$
  **else**
    $k \leftarrow$ DES$(a, \Psi)$
    $P \leftarrow$ getPrediction$(a, k)$
    $\psi \leftarrow$ conceptDriftDetector$(P)$
    **if** $\psi > 0$ **then**
      $k \leftarrow$
       utilize $a$ and $\psi$ to get new classes according to Algorithm 4
      $\Psi \leftarrow \Psi + k$
      **if** $|\Psi| > \kappa$ **then**
        removeWorstClassifier$(a)$
      **end**
      $p \leftarrow$ getPrediction$(a, k)$
    **end**
  **end**
**end**
**return** p

**Algorithm 4:** Flow of the Emerging Phase.

**Input:** current chunk $a$, drifted pool $\psi$, historical drifts $h$

**Data:** current chunk $a$, classifier $k$, Stream Emerging New Classes SENC, SENC Pool $\Psi$, K-means centroids $\lambda$, centroid distance $\Phi$, maximum centroid distance $dc$, chunk distance $i$, Nearest neighbor $n$, historical drifts $h$, average historical drifts $\mu$

**Output:** classifier $k$

$\lambda \leftarrow$ apply Kmeans$(a)$
$\Phi \leftarrow$ apply cosine similarity$(\lambda)$
$dc \leftarrow \max(\Phi)$
$\mu \leftarrow \frac{\sum h}{|h|}$
$\sigma \leftarrow$ std$(\psi)$
$\Delta \leftarrow$ first derivative$(h)$
threshold $\leftarrow \mu + \Delta \times \sigma$
**for** $\psi$ *has instance* **do**
    $n \leftarrow$ apply KNN$(\lambda, i)$
    $d \leftarrow$ apply cosine similarity$(i, n)$
    **if** d $\geq$ dc **then**
        $\Psi \leftarrow \Psi + i$
    **end**
    **if** $|\Psi| \geq$ *threshold* **then**
        $k \leftarrow$ trainingNewClassifier$(a)$
        $h \leftarrow h + \psi$
        $\Psi \leftarrow \phi$
    **end**
**end**
**return** k

**Table 5.2:** Running time of SENCForest, SENNE, KENNE, and the second Proposed Approach (PA2).

| Stream | Algorithm | Training Times | Time (seconds) |
|---|---|---|---|
| Covertype | SENCForest | 142 | **1997** |
| | SENNE | 5 | 2167 |
| | KENNES | 3 | 1742 |
| | PA2 | 150 | 2131 |
| Sensor | SENCForest | 23 | 1244 |
| | SENNE | 17 | 3825 |
| | KENNES | 152 | 2109 |
| | PA2 | 66 | **1075** |
| Synthetic | SENCForest | 12 | 107 |
| | SENNE | 26 | 224 |
| | KENNES | 149 | 202 |
| | PA2 | 47 | **91** |

**Table 5.3:** Running time of KNN, SVC, GNB, and HT as base classifiers.

| Algorithm | KNN | SVC | GNB | HT |
|---|---|---|---|---|
| Training Times | 140 | 150 | **139** | 148 |
| Time (seconds) | 586 | 878 | **291** | 1169 |

**Table 5.4:** Running time of ADWIN and DDM as drift detectors.

| Algorithm | ADWIN | DDM |
|---|---|---|
| Training Times | 139 | 55 |
| Time (seconds) | 272 | 545 |

CHAPTER

# 6

# Dynamic Classification Ensembles for Handling Imbalanced Multiclass Drifted Data Streams

Transfer learning plays a pivotal role in addressing the intricate challenges posed by dynamic data streams and inherent concept drifts. Transfer learning aims to enhance a model's learning performance within a target domain by leveraging the knowledge gleaned from the source domains [4, 112]. This research domain focuses extensively on addressing target tasks that grapple with limited or unlabeled data, with a dual emphasis on amplifying positive knowledge transfer and alleviating negative knowledge transfer [112]. To facilitate the transfer of valuable knowledge, researchers have developed diverse techniques, including methodologies aimed at reducing the domain gap, such as instance re-weighting [98, 113, 114] and feature matching [92, 98]. Conversely, strategies for mitigating negative knowledge transfer often involve down weighting irrelevant data sources [112]. Although a substantial body of transfer learning research has focused on static environments, where data in each domain are assumed to conform to the same distribution, real-world scenarios, including financial data analysis, energy demand prediction, and climate data analysis, often involve dynamic environments. Within these dynamic contexts, the concept drift problem [50, 102] emerges as data distri-

butions evolve over time. Dynamic environment transfer learning requires continuous adaptation to concept drift and adaptive utilization of valuable knowledge from source domains within each distinct environment. These new challenges in transfer learning, particularly within dynamic environments, represent a profound departure from traditional transfer learning paradigms that do not inherently address concept drift and the dynamic nature of evolving data. To overcome these challenges, Dynamic Ensemble Selection (DES) [20, 23, 71], ADWIN, and Streaming Ensemble Algorithm (SEA) have become prominent in the scientific literature [33, 67, 69] Dynamic ensemble selection employs multiple classifiers within the realm of machine learning to make collective predictions or classifications, and leverage data characteristics. Its key feature is its dynamic adaptability, as it continuously evaluates the performance of individual classifiers and selects a subset that demonstrates the highest competence within the prevailing data conditions. This adaptability empowers the ensemble to progressively enhance its performance by incorporating best-fitting classifiers for the existing data context. Moreover, ineffective classifiers can be excluded from the ensemble in scenarios marked by concept drift, thereby protecting the overall performance from detrimental effects. Adaptive Windowing (ADWIN) is another widely used method designed to address the challenges of concept drift [67]. Concept drift refers to the phenomenon in which the statistical properties of data change over time, leading to a decline in the learning algorithm performance. ADWIN monitors incoming data by maintaining sliding windows of variable sizes and evaluating statistical attributes, such as the mean or variance within these windows. Upon detecting substantial shifts or drifts, the ADWIN initiates adjustments to the ensemble or classifier configuration. These adjustments may involve retraining classifiers with fresh data, or incorporating new classifiers that are better suited to the updated data distribution. Additionally, the Streaming Ensemble Algorithm (SEA) is an integral component for addressing these dynamic challenges [33, 67, 69]. SEA is specifically designed to manage data streams and inherent concept drifts. This is achieved by adapting the ensemble of classifiers in real time as new data arrives. SEA's adaptability of SEA ensures that the ensemble remains robust and accurate even in the face of shifting data conditions and the emergence of new classes. This study proposes efficient algorithms for

classifying data streams in real-time scenarios, focusing on addressing the challenges posed by heterogeneous transfer learning in data distributions. The goal is to develop a novel algorithm (Heterogeneous Transfer Learning) that can effectively handle non-stationary data streams, achieve high classification accuracy, and minimize computational complexity. The subsequent sections of this paper adhere to a well-structured organization. Section 6.1 presents the motivations and contributions. Section 5.2 introduces the third proposed approach, providing intricate explanations of its constituents, including dynamic classifier ensembles, concept drift handling, and heterogeneous multisource transformation. Section 6.3 outlines the experimental setup and presents the results, providing details regarding the employed datasets, evaluation metrics, and procedures.

## 6.1 Motivations and Contributions

This paper presents a novel contribution to real-time streaming scenarios by addressing the challenges of heterogeneous multisource streams, with key contributions that can be summarized as follows:

1. The primary innovation involves incorporating a concept drift detection method in conjunction with an ensemble classifier, enabling real-time adaptation and refinement of the third proposed approach in response to transfer learning in non-stationary environments. This methodology ensures continuous evolution of the classification model in accordance with the changing data landscape.
2. The second significant advancement is the introduction of a precise weighting method to assess the significance of each local classifier within the ultimate classifier.
3. The third significant contribution is the development of an innovative approach that employs the eigenvector technique to facilitate the transfer of knowledge from heterogeneous source domains to the target domain.

## 6.2 Third Proposed Methodology

In this section, we introduce the Heterogeneous Transfer Learning (HTL) algorithm tailored for non-stationary environments. HTL adeptly assimilates knowledge from both heterogeneous and homogeneous sources, operating within the framework of data streams subject to concept drift. Leveraging an online learning inductive parameter transfer strategy, HTL achieves seamless knowledge transfer. We delineate the core components and workflow of the HTL algorithm. To address the challenges intrinsic to our approach, we focus on three key aspects:

- **Heterogeneous sources:** Traditional transfer learning methods often assume homogeneity in the data distribution across source and target domains. However, in real-world scenarios, data sources may vary significantly in terms of feature space. HTL addresses this challenge by allowing knowledge transfer from sources with different dimensionalities. This means that the algorithm can effectively learn from diverse dimentionality sources, enabling a more comprehensive knowledge transfer process.

- **Drifted streams:** In dynamic environments, data distributions may change over time, leading to concept drift. This phenomenon poses a significant challenge for machine learning algorithms, as models trained on historical data may become obsolete as the underlying data distribution shifts. HTL incorporates a concept drift detector that continuously monitors the incoming data stream. Upon detecting a shift in the data distribution, the algorithm adapts its classifiers accordingly to ensure continued effectiveness in handling drifting streams. This dynamic adjustment mechanism allows HTL to maintain high performance even in the presence of concept drift.

- **Classifier performance:** Classifier performance is crucial for the overall effectiveness of the transfer learning process. HTL employs Dynamic Ensemble Selection (DES) to enhance classifier performance. DES creates a diverse ensemble of classifiers, each trained on different subsets of the data. When presented with a new data point, DES dynamically selects the most suitable classifiers from the ensemble based on its performance on similar chunk. This adaptive selection process ensures that the most appropriate classifier

is chosen for each data point, leading to improved classification accuracy and robustness. By leveraging DES, HTL maximizes the utility of available classifiers, resulting in superior performance in non-stationary environments with heterogeneous data sources.

### 6.2.1 HTL Overall Details

The aim of this section is to harness knowledge from diverse dimensional multi-source domain streams. Following a similar framework to CDTL [1], this approach employs a class-wise and domain-weighted strategy. However, HTL enhances the weight function of CDTL in a class-wise manner, as demonstrated in Equation 1. This equation factors in both correct and incorrect predictions for each classifier class, with K representing the number of classifiers in the current chunk, C denoting the number of classes in the current chunk, and i indicating the current chunk. The components of HTL operate synergistically, as depicted in Fig. 6.1, encompassing three phases:

- **Dynamic ensemble selection phase (DES Phase):**: The primary objective DES phase is to identify the optimal classifier for incoming data. This is crucial for ensuring that the selected classifier effectively aligns with the unique characteristics of the current data segment.

- **Drift detector phase:** After the DES phase, our method advances to the drift detector phase, where it operates in real-time to continually monitor the data stream. This phase employs ADWIN and DDM techniques, which play a pivotal role in swiftly identifying any signs of concept drift. These techniques are designed to detect changes in the underlying data distribution over time, thus enabling the algorithm to adapt to evolving data patterns.

- **Feature scaling** The concluding phase of our approach is featuring scaling, operating in real-time to harmonize the diverse dimensionalities of the multisource streams with the target dimensionality. Leveraging the eigen vector technique, this phase facilitates the transformation of data into a unified dimensionality, essential for creating new classifiers tailored to the target and domain streams. By ensuring compatibility with the learning framework,

this process enhances the algorithm's effectiveness in handling diverse dimensionality streams.

### 6.2.2 Classifier Creation Details

Fig. 6.2 provides a comprehensive overview of the classifier creation phase, a crucial component responsible for generating new classifiers based on the current chunk of the target domain, previous classifiers, and source domain classifiers. This phase offers several advantages and perform three tasks.

- **Source projection:** This step involves projecting the source domain classifiers onto the current chunk of the target domain. The projection likely employs the source weight function, as described in CDTL [1], to assign a weight to each classifier based on its relevance to the current chunk of data. This weighting mechanism ensures that classifiers from the source domains contribute appropriately to the creation of the new classifier for the target domain.

- **Class-wise weights:** This block computes class-specific weights for each classifier using Equations 6.1 and 6.2 . The process involves analyzing the current chunk of the target stream (denoted as $D$) and considering both correct and incorrect predictions made by each classifier where $prediction_i$ refers to the predicted class of the current instance and $c$ to the income class. These weights are crucial for determining the significance of individual classifiers across different classes. The equations facilitate the calculation of weights that reflect the classifiers' performance on specific classes, enabling the creation of a well-balanced ensemble classifier.

- **Classifier combination:** In this phase, the class-wise weights, along with the current chunk of data and the projected source domain classifiers, are combined to derive the final classifier. The combination process follows the details outlined in Eq. 6.3 , where historical classifiers (denoted as $H$), projected data classifiers (denoted as $P$), and the classifier of the current chunk (denoted as $K$) are integrated. This integration ensures that the resulting classifier incorporates contributions from both historical and projected data

sources, leveraging the strengths of each to enhance predictive performance. The resulting classifier is then utilized to make predictions based on the data chunk, effectively leveraging the insights gleaned from both historical and current data sources.

### 6.2.3 HTL Detailed Algorithm

As illustrated in Algorithm 5, the HTL algorithm involves several stages, beginning with training a classifier for the target stream and proceeding through various steps related to heterogeneous multisource preprocessing, classifier weighting, concept drift detection, and classifier management to ensure the accuracy and adaptability of the final classifier. In this section, detailed explanations of each individual step within the HTL algorithm are provided.

- **Converting heterogeneous multisource to homogeneous multisource:** In the initial step, the algorithm unifies the various data sources that might have different characteristics (heterogeneous multisource). This is achieved using eigenvectors and the feature count of the current chunk (line 7).

- **Training a new classifier for the first target chunk:** In line 8, the new classifier is trained specifically for the first chunk of the target stream. This classifier was used to predict the initial chunk of the target stream.

- **Calculate heterogeneous multisource weights:** The algorithm computes the weights for each heterogeneous data source. These weights are determined based on the characteristics of the data from each source and the target classifier. This weighting process helps prioritize more relevant data sources (line 9).

- **Calculating class-wise weights:** In line 27, the algorithm calculates class-wise weights using Equations 1 and 2. These weights were essential for determining the contribution of each class to the final classifier.

- **Converting homogeneous multisource to projected data source:** Line 11 involves using the weights calculated in the third step to transform the homo-

geneous multisource representation to a projected data source. This transformation likely uses the source weight function of CDTL [1].

- **Prediction of the current chunk:** In line 16, the algorithm determines the output class using Equation 3.
- **Monitoring for concept drift:** The algorithm continuously monitors the accuracy of its predictions to detect any concept drift, a situation in which the underlying data distribution changes, potentially leading to a degradation in model performance. Line 19 has been used for this purpose.
- **Updating the projected multisource classifiers:**Lines 29–32 are dedicated to updating the classifiers associated with the projected multisource. It is necessary to adapt to changes in the data or to maintain the accuracy and relevance of the classifiers over time.
- **Managing the pool of classifiers:** If the pool of classifiers exceeds a predefined maximum size threshold, lines 23 and 24 indicate that a new classifier is trained, and the worst-performing classifier is removed. This process helps to maintain a manageable and effective set of classifiers.

The heterogeneous Transfer Learning (HTL) algorithm represents a significant advancement in addressing the complexities of non-stationary environments prone to concept drift. By integrating heterogeneous and homogeneous sources within dynamic data streams, HTL demonstrates remarkable adaptability and effectiveness. Through its meticulously designed workflow, HTL can efficiently handle the challenges posed by evolving data landscapes. From the initial reception of environmental data to the nuanced processing of new data chunks and vigilant monitoring of concept drift, HTL embodies a comprehensive approach to knowledge transfer and adaptation. Owing to its ability to compute class-specific weights and leverage vital classifiers, HTL offers a robust solution for navigating non-stationary environments with confidence and precision.

$$classWeightk, c, \mathcal{D} = \sum_{i \in \mathcal{D}} \frac{|prediction_i = c|}{|\mathcal{D}|} \times \frac{|prediction_i \neq c|}{|\mathcal{D}|} \tag{6.1}$$

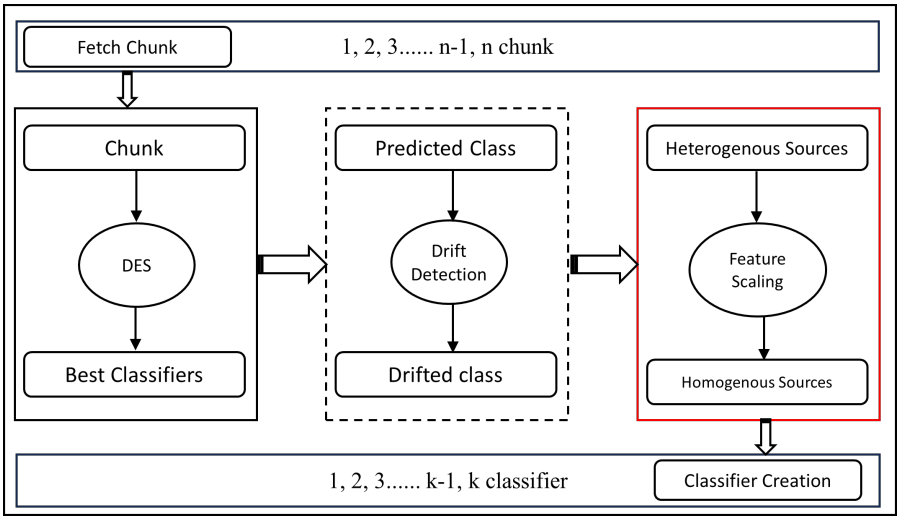



**Figure 6.1:** Heterogeneous Transfer Learning (HTL) Approch Flow.

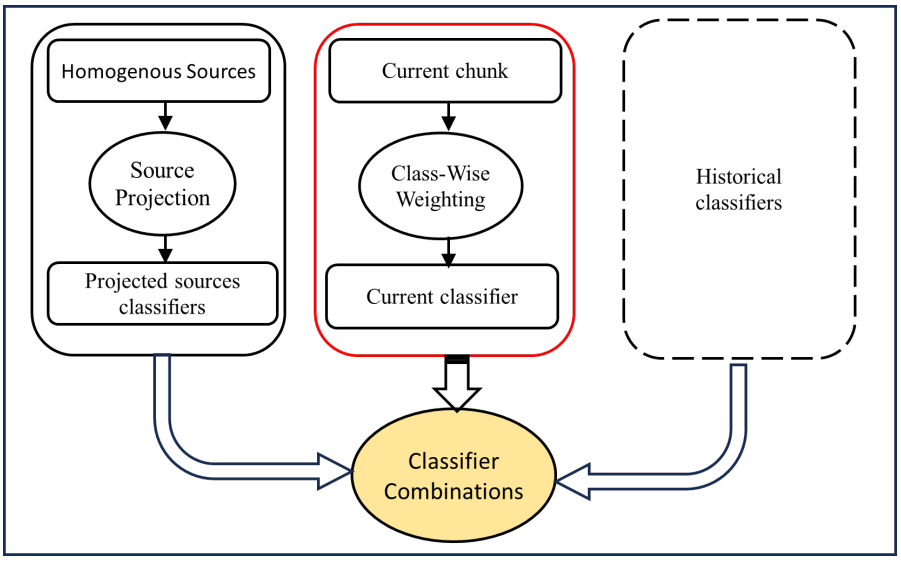



**Figure 6.2:** Approach for Classifier Creation.

$$classifierWeight = \sum_{k \in K} \sum_{c \in C} classWeight_{k,c,\mathcal{D}} \quad where \mathcal{D} = 1, 2, 3 \ldots N \quad (6.2)$$

$$Prediction_{p,H,k,chunk} = \sum_{p \in P} prediction^{p}_{chunk} + \sum_{h \in H} prediction^{h}_{chunk} + prediction^{k}_{chunk} \quad (6.3)$$

**Algorithm 5:** Flow of the Heterogeneous Transfer Learning (HTL).

**Input:** Target domain stream *stream*, heterogeneous multisource domain $\Psi$, pool of classifiers, threshold $\ell$

**Data:** Current chunk *a*, classifiers *k*, source classifiers $\psi$, projected source domain *S*, target domain weights $\omega$, source domain weights $\lambda$

**Output:** Prediction

```
for stream have chunk do
    if a is the First chunk then
        Ψ ← convertSourcesToTargetDim(Ψ, a)
        k ← trainingNewClassifier(a)
        λ ← SourcesDomainWeights(Ψ, k)
        ω ← classWiseWeights(K, a)
        S ← projectedSourceDomain(Ψ, λ)
        for source in S do
            newClassifier ← trainingNewClassifier(source)
            ψ ← ψ ∪ newClassifier
        prediction ← getPrediction(a, k, ψ, λ, ω)
    else
        prediction ← getPrediction(a, k, ψ)
        driftResult ← conceptDriftDetector(prediction)
        if driftResult have drift then
            newClassifier ← trainingNewClassifier(a)
            k ← k ∪ newClassifier
            if size(K) ≥ ℓ then
                removeWorstClasssifier(k)
            λ ← SourcesDomainWeights(Ψ, k)
            ω ← classWiseWeights(K, a)
            S ← projectedSourceDomain(Ψ, λ)
            for source in S do
                newClassifier ← trainingNewClassifier(source)
                ψ ← ψ ∪ newClassifier
            prediction ← getPrediction(a, k, ψ, λ, ω)
return prediction
```

## 6.3 Experimental Results

In this section, the proposed HTL, CORAL [92], and CDTL [1], and Melanie [2] algorithms are compared. Three subsections are introduced: the first section focuses on the experimental setup; second, a discussion on data streams to apply our experiments on heterogeneous and homogeneous source domains, including a comparison to evaluate the runtime factor between all methods; and finally, a comparison is made between online learning and chunk-based concept drift. The two main questions to be answered are:

- $Q_1$. How does the Heterogeneous sources in data streams affect the stability and performance of Transfer Learning Technique?
- $Q_2$. What approaches or techniques can be employed in heterogeneous transfer learning to facilitate knowledge transfer across diverse sources and domains?

### 6.3.1 Experimental setup

The evaluation of Heterogeneous Taxonomy Learning (HTL) involves a comparison with CORAL [92], CDTL [1], and Melanie [2] which employs multiple metrics such as precision, recall, F1 score [107], BAC [108], and G-mean [109]. The experimental protocol employed for evaluation followed the test-then-train approach [110], where the classification model is trained on a specific data chunk and subsequently evaluated on the next chunk. The chunk size was standardized to 2000 instances. Four different classification models were used as base estimators: k-Neighbors (KNN) algorithm, Support Vector Machine (abbreviated as SVM), Gaussian Naive Bayes abbreviated as (GNB), and Hoeffding Tree (abbreviated as HT), as implemented in scikit-learn [36]. We establish an ensemble classifier pool with a set limit of L = 8, wherein each ensemble consists of N = 4 base models. While these constraints remained fixed across all our experiments, the threshold for the pool classifier in each approach was maintained at eight. Consequently, if the threshold is surpassed, the least-performing classifier is systematically eliminated. This configuration was applied consistently across all approaches to ensure

fair engagement. The experiments were carried out using the Python programming language, with the source code publicly accessible on GitHub[1] . When dealing with heterogeneous multisource streams, it is often impractical or not applicable to truly heterogeneous experiences. Consequently, the eigenvector technique was utilized in all related approaches. This decision stems from the fact that the algorithms in related work primarily focus on homogeneous source domains and do not address the challenges posed by heterogeneous multisource data. Therefore, heterogeneous experiments were conducted.

#### 6.3.1.1 **Data Streams**

In this study, the performance of the third proposed approach was evaluated using various datasets, including synthetic data streams, a real application stream, and dataset benchmark datasets. To conduct the evaluations, the stream-learn Python library [67, 111] was used. As detailed in Table 1, the benchmark dataset employed in the study is the Covertype dataset. This dataset comprises 52 features, 7 classes, and a total of 581,010 instances. It serves as a standard benchmark dataset widely used in stream mining research. For the evaluation of real application streams, the Sensor Stream dataset was utilized. This dataset includes 5 features, 58 classes, and a total of 392,600 instances, representing a real-world application scenario. Synthetic datasets were generated using the scikit-learn Python package. This synthetic dataset was created to simulate data streams and evaluate the performance of the framework. The synthetic datasets consisted of 10 features and four classes and were divided into 200 chunks. Each chunk had a size of 2,000. The performance of the third proposed approach was systematically evaluated using these datasets and a stream-learn library. These evaluations provided valuable insights into the effectiveness of the framework in handling different types of data streams, including benchmark datasets, real application streams, and synthetic data streams.

[1] `https://github.com/Amadkour/transfer_learning_with_concept_drift.git`

**Table 6.1:** Characteristics of the datasets utilized in the experiments.

| Dataset | Number of Features | Number of Classes | Number of Instances |
|---|---|---|---|
| Covertype dataset[1] | 40 | 7 | 581,010 |
| Sensor Stream dataset[2] | 5 | 58 | 392,600 |
| Synthetic stream-52 | 52 | 5 | 200,000 |
| Synthetic stream-8 | 8 | 4 | 200,000 |

#### 6.3.1.2 Compared Approaches

In the subsequent section, we introduce the established benchmark techniques as outlined below:

- **AW- CORAL [92]:** The AW- CORAL (Adaptive Weighted CORrelation ALignment) method is a simple domain adaptation technique designed to align the distributions of both source and target features without supervision. This is accomplished by aligning second-order statistics, focusing specifically on covariance, to match the distributions between the domains.

- **HE-CDTL [92]:** The HE-CDTL approach tackles Concept Drift Transfer Learning (CDTL) by integrating knowledge from source domains and historical time steps in the target domain to improve learning performance. HE-CDTL features class-wise weighted ensemble for independent selection of historical knowledge by each class, and AW-CORAL to mitigate domain discrepancy and reduce negative knowledge transfer.

- **Melanie [2]:** Multisource Online Transfer Learning for Non-stationary Environments, known as Melanie, represents the inaugural method capable of transferring knowledge across various data streaming sources in non-stationary environments. Melanie constructs numerous sub-classifiers to grasp diverse facets from distinct source and target concepts dynamically. It identifies sub-classifiers that align closely with the prevailing target concept, assembling them into an ensemble for predicting instances originating from the target concept.

While Melanie extends online learning through chunk-based learning akin to CDTL, it exhibits two limitations:

- utilizing a global weight for each classifier, disregarding performance variation across different locations of a data chunk
- combining learned classifiers from these domains directly with the target classifier, which may impede effective knowledge transfer due to domain discrepancies. Hence, we compare the third proposed approach (HTL) with AW- CORAL [92] and HE-CDTL yang2021concept.

### 6.3.2 Analysis of Experimental Results

This section presents a comprehensive evaluation of the performance of the proposed approach across multiple data streams. To guarantee a thorough assessment, five performance metrics—F1 score, precision, recall, G-mean, and BAG—were presented using two visualization diagrams: radar and line. A radar chart was cleverly utilized to provide a comprehensive summary, precisely demonstrating the performance of each algorithm across the six key metrics. By calculating the average value of each metric, the overall performance of each approach (HTL, CORAL, and CDTL) was determined. On the other hand, the line diagram utilizes the G-mean metric to compare these methods for each chunk across 100 chunks. A line diagram was used in the first five experiments, whereas a combination of radar and line diagrams was employed in the last two experiments. This approach ensured a detailed and nuanced evaluation of the performance of the third proposed approach in diverse experimental scenarios.

#### 6.3.2.1 Results on Target Domains Dataset without Source Domain

Fig. 6.3 showcases the results of applying the aforementioned methods to the Covertype dataset, where no stream is utilized as a source domain. The line diagram specifically highlights the classification accuracy measured by the G-mean metric across 100 data chunks for each method. Notably, all methods display suboptimal accuracy during the initial 10 chunks. However, a significant improvement is evident in subsequent phases. This improvement can be attributed to the expansion of the classifier pool, encompassing an increasing number of classifiers. This expansion allows the DES technique to become more adept at selecting the most

appropriate classifier for each incoming chunk. Consequently, accuracy experiences a notable upsurge in later chunks, underscoring the adaptability and efficacy of the ensemble approach. From chunk 20 to the final chunk, HTL consistently achieves the highest accuracy across most chunks. In contrast, CORAL shows lower performance in certain chunks, while CDTL displays lower performance in others. The superior performance of HTL can be credited to its utilization of the proposed weighting method, which assigns a weight to each historical classifier. This approach contributes to the effective training of the pool classifiers, thereby enhancing the overall performance of the approach.

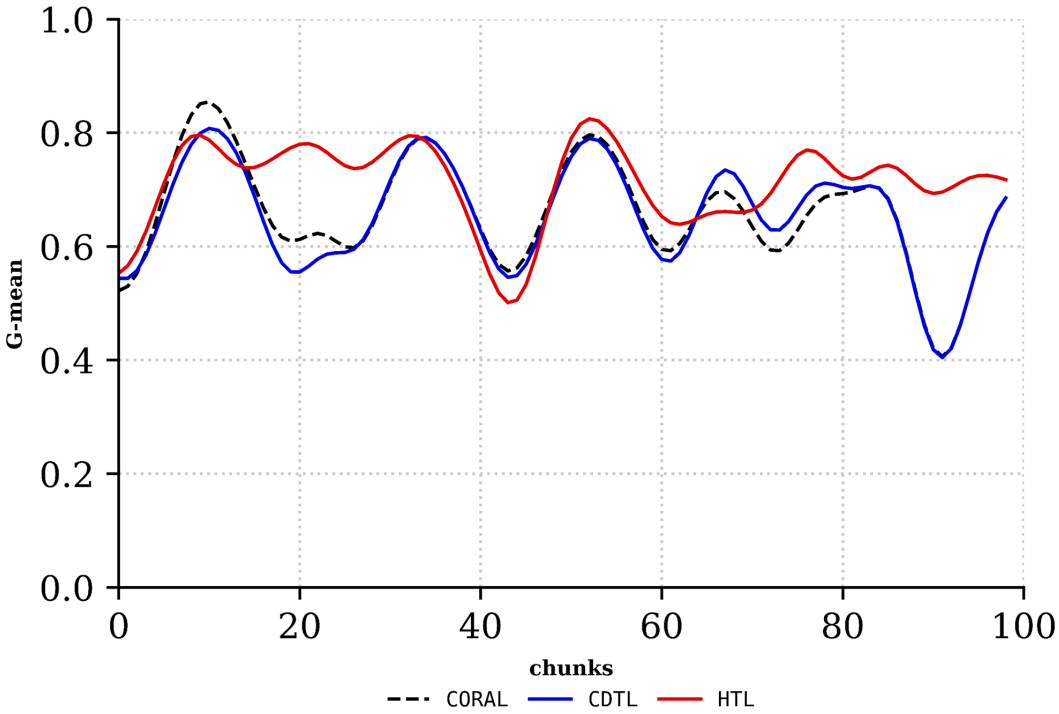



**Figure 6.3:** Results of the Covertype Stream as the Target Domain in the Absence of the Source Domain.

#### 6.3.2.2 Results on the Homogeneous Source Domains Dataset

Fig. 6.4 depicts the results of applying the compared methods to the Covertype data stream as the target domain and Synthetic-52 as a homogenous source domain, which has the same dimensionality as the Covertype stream (52 features and

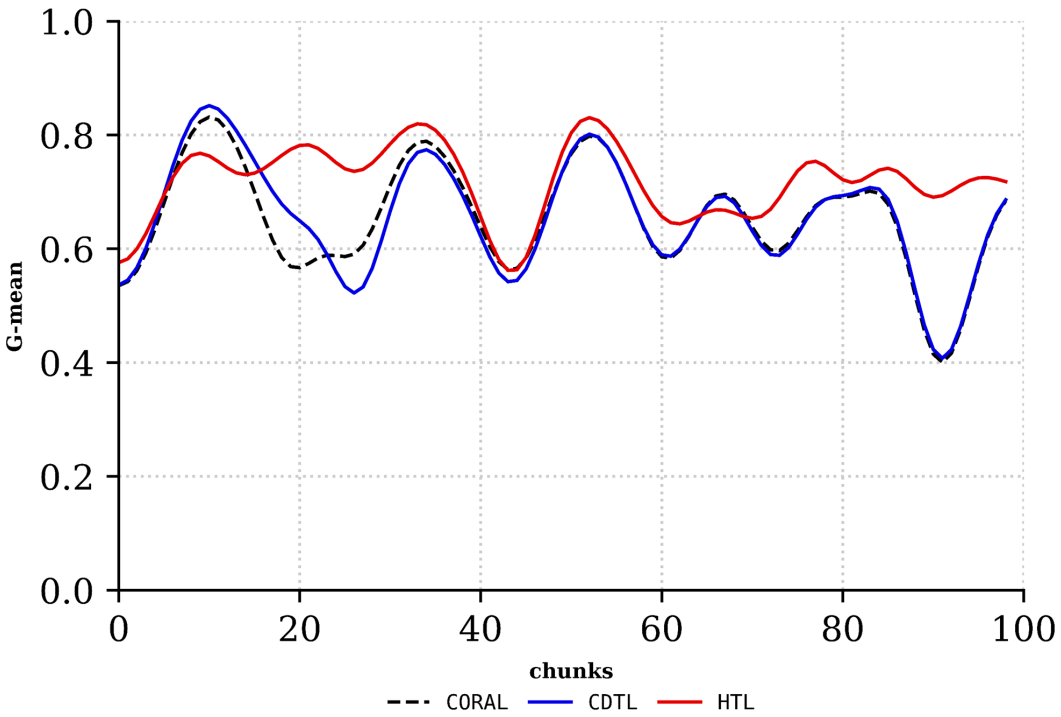



**Figure 6.4:** Results of the Covertype Stream as the Target Domain with Homogeneous Source Domains.

seven classes). Upon examining the line diagram, it becomes evident that HTL's performance in the first eight chunks may be suboptimal due to the insufficient number of classifiers available. However, following the initial eight chunks, the addition of more classifiers results in an improvement in HTL's performance. In contrast, CDTL maintains satisfactory performance throughout the chunks, while MLSMOTE consistently displays lower performance. It is important to note that HTL consistently achieves the highest performance across all chunks in the diagram. Notably, the CDTL and HTL methods in Fig. 6.4 demonstrated superiority over the same methods in Fig. 6.3. This is attributed to the experiment shown in Fig. 6.4, which incorporates Synthetic-52 as a source domain, thereby enhancing the performance of the methods. In contrast to the experiment in Fig. 6.3, which lacks a source domain.

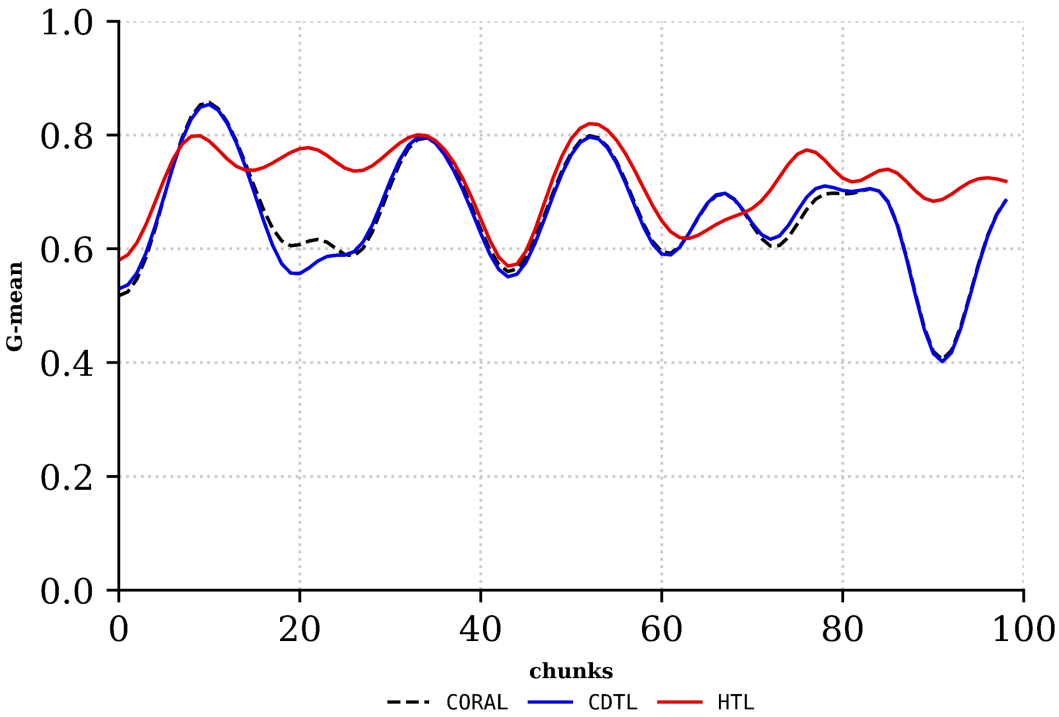



**Figure 6.5:** Results of the Covertype Stream as the Target Domain with One Heterogeneous Source Domain.

#### 6.3.2.3 Results on One Heterogeneous Source Domain

In this experiment, the Synthetic-8 stream was utilized as a heterogeneous source domain. Considering the nature of the HTL algorithm, which can operate with heterogeneous sources, specific adjustments were implemented to enable CORAL and CDTL, which typically do not operate with such sources, to function in a heterogeneous domain environment. The eigenvector technique was applied to CORAL and CDTL to facilitate compatibility with the heterogeneous sources. Fig. 6.5 shows the results of applying the compared methods to the Covertype data stream as the target domain, with Synthetic-8 acting as the heterogeneous source domain. Synthetic-8 has a different dimensionality (eight features and four classes) compared with the Covertype stream. An analysis of the line diagram reveals that the performance of the compared approaches may vary across different chunks depending on the source domain and the positive knowledge it contributes. HTL consistently outperforms, achieving the highest performance across almost all chunks,

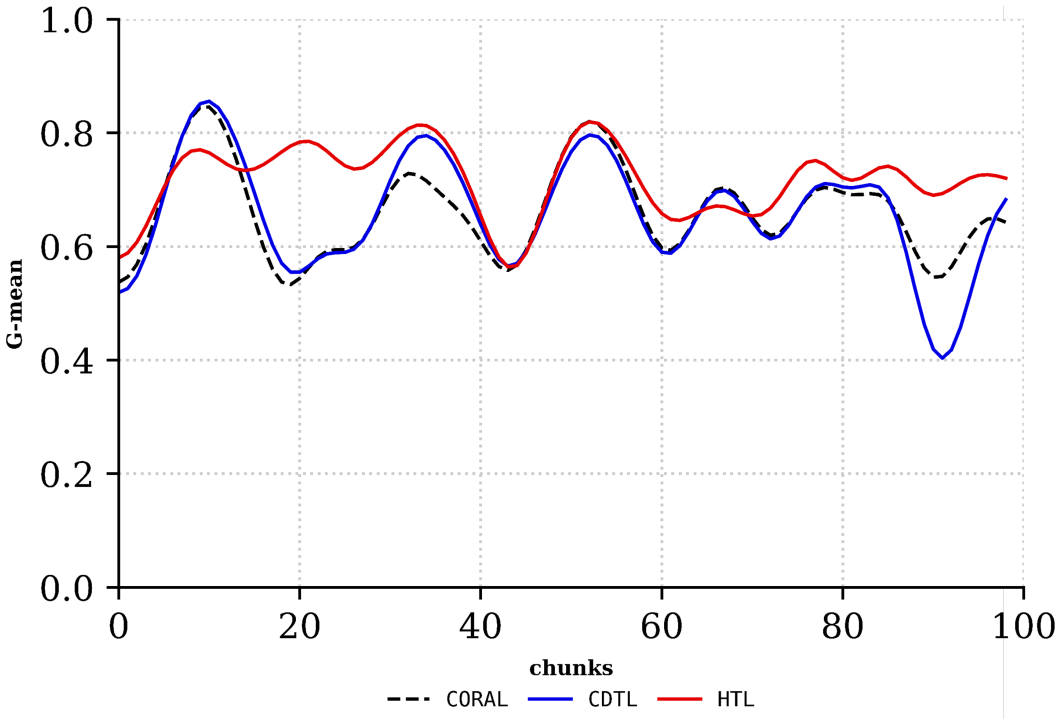



**Figure 6.6:** Results of the Covertype Stream as the Target Domain with Multiple Heterogeneous Source Domains.

whereas CDTL and CORAL show lower performance.

#### 6.3.2.4 Results on All Heterogeneous Source Domains Stream and Covertype Stream as the Target Domain

Fig. 6.6 presents the results of applying the compared methods to the Covertype data stream as the target domain, with the Synthetic-8, Synthetic-52, and Sensor streams serving as heterogeneous source domains. From the analysis, the performance in this figure exceeds that in Fig. 6.6. This improvement is likely due to the larger number of source domains in this experiment, which leads to an incremental increase in positive knowledge with each added source domain, consequently improving method performance. In Fig. 6.7, the same experimental setup was employed, but the target domain was switched from the cover type to the sensor stream. Additionally, the Covertype stream was included as one of the source domains in this configuration. This modification was intended to evaluate

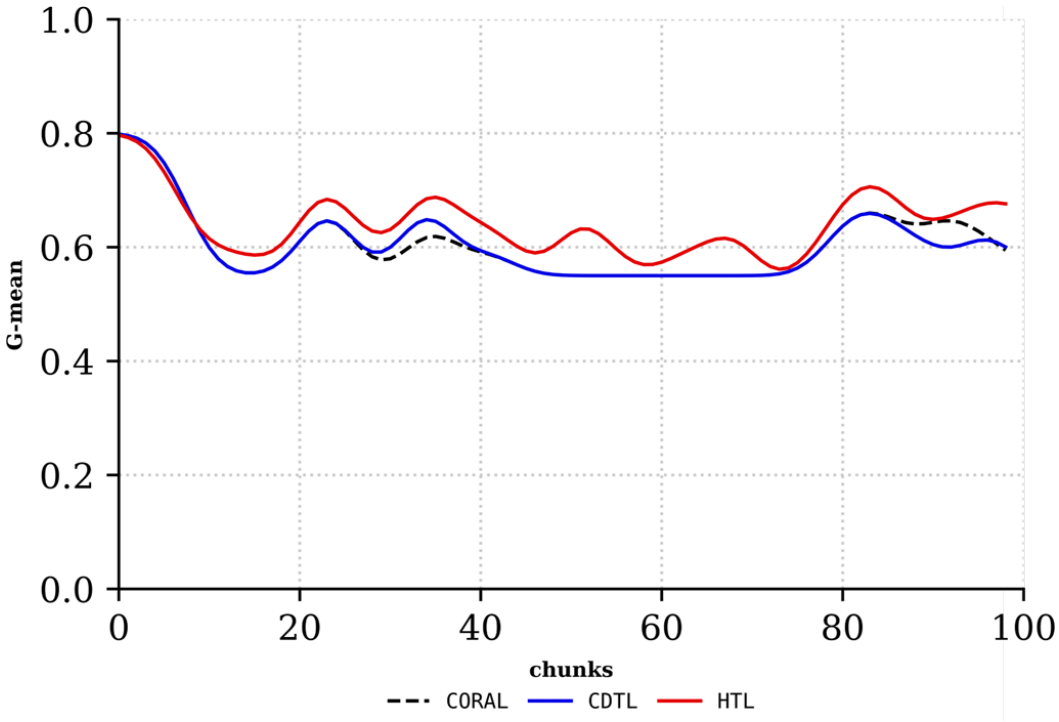



**Figure 6.7:** Results of the Sensor Stream as the Target Domain with Multiple Heterogeneous Source Domains.

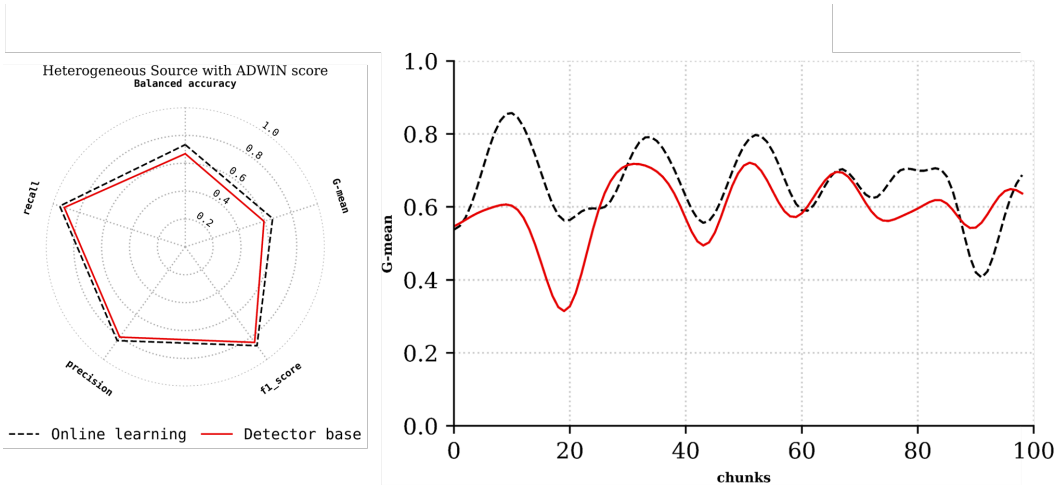



**Figure 6.8:** Online learning and detector chunk-based result in Covertype stream via ADWIN detector

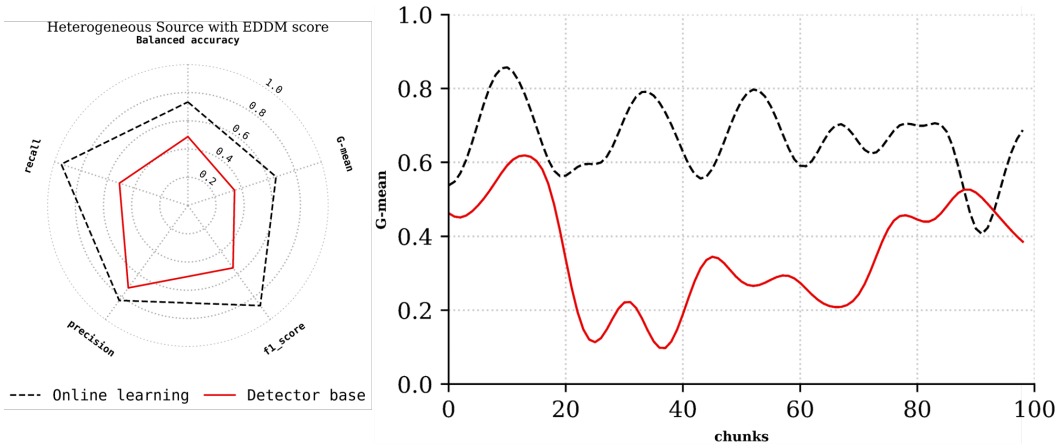



**Figure 6.9:** Results of Online Learning and Chunk-Based Detection in the Covertype Stream with the ADWIN Detector.

the performance of the HTL across the various target streams. Fig. 6.7 shows that the performance of all methods might be lower than that of the Covertype stream, mainly because of the increased noise in the sensor stream and numerous drifts. However, HTL consistently outperforms, achieving the highest performance across all chunks, whereas CORAL and CDTL display nearly identical values in most chunks.

#### 6.3.2.5 **Results on One Heterogeneous Source Domain**

This section presents the overall performances of the compared methods (CORAL, CDTL, and HTL) across five key performance metrics: F1 score, precision, recall, G-mean, and BAG. Table 2 showcases the performance of each method across different experiments for the five metrics. The table illustrates the average value of each metric, which represents the overall performance of each method for 100 chunks. The emphasis on the bold highlighting in the table underscores the superiority of the HTL algorithm in various source domain configurations. In the first experiment (without the source domain), CORAL showed the best performance in terms of precision, while HTL demonstrated the best performance in the other metrics. Furthermore, in most experiments, HTL outperformed the other methods across most metrics, except for the recall metric in the third experiment, in which CDTL achieved the best performance. Notably, each subsequent experiment

demonstrated improved performance compared with the previous experiment. This trend arose from the incremental nature of the source domain in each experiment. The incremental addition of the source domain enhances the positive knowledge of each experiment, leading to an increased performance of the compared methods in most experiments.

**Table 6.2:** Comprehensive Performance of CORAL, CDTL, and HTL Across All Previous Experiments.

| **Experiment** | **Metric** | **CORAL** | **CDTL** | **HTL** |
|---|---|---|---|---|
| Without source domain | BAC | 0.723 | 0.723 | **0.725** |
| | G-mean | 0.652 | 0.654 | **0.717** |
| | F1-score | 0.875 | 0.874 | **0.890** |
| | Precision | 0.827 | 0.823 | **0.851** |
| | Recall | **0.950** | 0.944 | 0.939 |
| Homogenous source domain | BAC | 0.731 | 0.730 | **0.755** |
| | G-mean | 0.658 | 0.653 | **0.717** |
| | F1-score | 0.876 | 0.875 | **0.858** |
| | Precision | 0.830 | 0.829 | **0.851** |
| | Recall | 0.950 | 0.950 | **0.953** |
| Single heterogeneous source domain | BAC | 0.732 | 0.732 | **0.757** |
| | G-mean | 0.661 | 0.657 | **0.717** |
| | F1-score | 0.875 | 0.875 | **0.859** |
| | Precision | 0.835 | 0.835 | **0.851** |
| | Recall | 0.947 | 0.950 | **0.953** |
| Multisource heterogeneous domain (Covertype stream) | BAC | 0.732 | 0.732 | **0.757** |
| | G-mean | 0.661 | 0.661 | **0.717** |
| | F1-score | 0.875 | 0.875 | **0.859** |
| | Precision | 0.835 | 0.835 | **0.851** |
| | Recall | 0.947 | 0.950 | **0.953** |
| Multisource heterogeneous domain (Sensor stream) | BAC | 0.998 | 0.998 | **0.998** |
| | G-mean | 0.971 | 0.971 | **0.992** |
| | F1-score | 0.997 | 0.997 | **0.997** |
| | Precision | 0.998 | 0.998 | **0.998** |
| | Recall | 0.998 | 0.998 | **0.998** |

### 6.3.3 Analysis Runtime between CORAL, CDTL, and HTL Techniques

Our experimental results highlight that the selection of the optimal algorithm for handling heterogeneous transfer learning is influenced by various factors. These factors include the dataset characteristics and runtime requirements of the algorithm. In the experiments, the focus was on analyzing the runtimes of different algorithms, and the results were insightful. Notably, the HTL algorithm was exceptionally efficient. For example, when considering the training of the first experience (homogeneous source domain), the HTL algorithm completed 100 iterations within 304 seconds. In contrast, the CORAL and CDTL algorithms require more time to achieve the same number of training iterations. Furthermore, the HTL algorithm consistently demonstrated remarkable efficiency, particularly when the source domain settings were absent, single heterogeneous, or multisource heterogeneous. These findings highlight the superior runtime performance of the HTL algorithm, positioning it as an attractive choice for scenarios with stringent time constraints. The bold highlighting in Table 3 further accentuates the efficiency of the HTL algorithm across various source domain settings, including homogeneous, absent, single heterogeneous, and multi-source heterogeneous settings, making it a compelling option for applications where time efficiency is of paramount importance.

**Table 6.3:** Runtimes (in seconds) for CORAL, CDTL, and HTL.

| Experience | Target domain | Source Domain | CORAL | CDTL | HTL |
|---|---|---|---|---|---|
| without source domain | Covertype | None | 312 | 312 | **304** |
| Homogenous source domain | Covertype | Synthetic-52 | 6165 | 614 | **606** |
| Single heterogeneous source domain | Covertype | Synthetic-8 | 6060 | 4703 | **4475** |
| Multi-source heterogeneous domain | Covertype | Sensor, Synthetic-52, and Synthetic-8 | 23011 | 14043 | **13824** |
| Multi-source heterogeneous domain | Sensor | Covertype, Synthetic-52, and Synthetic-8 | 14524 | 3052 | **3005** |

### 6.3.4 Analysis accuracy and runtime of HTL for online learning and chunk-based concept drift

In this experiment, we investigated the performance durations of various learning flow techniques, specifically online learning and chunk-based concept drift,

aiming to provide a thorough comparison between them. As shown in Table 6.4, the EDDM drift detector demonstrated significant time savings by successfully detecting 40 drifted chunks out of 100 chunks, resulting in 40 learning times. Conversely, the ADWIN drift detector achieved time savings superior to the online technique, having detected 63 drifted chunks and requiring 63 learning times. On the other hand, online learning exhibited the least efficient time savings, as it necessitated 100 times for 100 chunks. Moreover, in Figures 6.8,6.9, each figure includes two diagrams (radar and line diagram). As seen in the radar and line diagrams of Fig. 6.8, used to compare online learning and the ADWIN detector, online learning emerges as the most effective technique for achieving optimal performance, albeit with an increase in time consumption. Similarly, Fig. 6.9 is employed to compare online learning with the EDDM detector, demonstrating that online learning is the most effective method for optimal performance. Consequently, online learning stands out as the most effective approach. Hence, by examining the radar diagrams of both figures, it becomes evident that ADWIN is the most similar alternative to online learning. As a result, ADWIN proves to be a favorable compromise, striking a balance between online learning and EDDM detectors in terms of accuracy. In contrast, EDDM exhibits the lowest accuracy. Therefore, in the context of chunk-based concept drift, particularly when utilizing the ADWIN detector, this approach is ideal for environments that demand strong performance while minimizing time expenditure.

**Table 6.4:** Runtimes (in seconds) for the Online Learning, ADWIN, and EDMM drift detectors.

| **Technique** | **Runtime** | **Learning Times** |
|---|---|---|
| Online Learning | 3102 | 100 |
| ADWIN detector | 2257 | 63 |
| EDMM detector | 1948 | 40 |

CHAPTER

# 7

# Conclusions and Future Work

In this chapter, we presented a comprehensive methodology for incremental learning in data streams, particularly addressing the complexities of imbalanced data, including minority and overlapping classes 7.1, Emerging new classes in the incremental drifted streams 7.2 and the heterogeneous transfer learning in the drifted streams 7.2.

## 7.1 Imalanced multiclass stream

This study integrates a novel oversampling technique, concept drift detection, and Dynamic Ensemble Selection (DES) to identify the most suitable ensemble classifier. Extensive experimentation across various datasets, including benchmarks, real-world streams, and synthetic data, demonstrated the effectiveness of our methodology. A key feature of our approach is the rapid detection of concept drifts, which allows the system to adapt quickly by training new base classifiers, ensuring continued relevance and accuracy in real-time scenarios. We addressed the minority class problem through advanced oversampling, while the KNN algorithm was employed to prevent the creation of overlapping class instances. The DES technique played a crucial role in selecting the most effective classifiers, optimizing overall performance.

The results of our evaluation, based on multiple performance metrics, show that our methodology is particularly effective in handling multiclass imbalanced

stream challenges. It excels in maintaining high accuracy as class distributions evolve. Beyond accuracy, the approach is marked by its adaptability, efficiency, and scalability. The capability for dynamic model updates driven by incoming data allows for continuous adaptation to changing data distributions, ensuring the model's reliability and relevance in real-time applications.

However, our methodology does have certain limitations. The time required to generate non-overlapping synthetic instances is significant, and the approach's success heavily depends on the accuracy of MLSMOTE and MLSOL techniques. Future research should focus on developing more advanced oversampling techniques that avoid overlapping synthetic instances. Additionally, exploring meta-learning methods to better assess imbalanced multiclass ratios and minority class identification could further improve the approach.

## 7.2 Emerging new classes

This section, a robust framework for incremental learning in data streams with emerging new classes. By integrating the ADWIN algorithm for concept drift detection, K-means clustering, and Gaussian Naive Bayes (GNB) as the base classifier within an ensemble stratified bagging framework, our method effectively tackles the challenges posed by such dynamic data streams. The ADWIN algorithm is particularly important, enabling the prompt detection of concept drifts and new classes, which is critical for training new classifiers and maintaining accuracy over time.

The use of ensemble stratified bagging combined with DES significantly enhances predictive performance and robustness. Our evaluations confirm that the framework is highly effective in classifying data streams with evolving distributions, especially when GNB serves as the base classifier. In addition to its high accuracy, the framework's strengths lie in its adaptability, efficiency, and scalability. The dynamic updates to the model, based on incoming data, ensure continuous adaptation to shifting data patterns, supporting real-time reliability and relevance.

Looking ahead, future work could explore several avenues for improvement. These include the development of advanced concept drift detection algorithms, the

exploration of alternative ensemble strategies, and the incorporation of deep learning techniques. Additionally, expanding the framework to address a broader range of real-world applications and refining the selection and prediction methods for classifiers will be crucial for further enhancing the methodology's effectiveness.

## 7.3 Heterogeneous transfer learning

This study introduces a comprehensive Approach for incremental learning in data streams, particularly focusing on heterogeneous transfer learning. The framework effectively addresses the challenges associated with this context by integrating concept drift detection through the ADWIN algorithm, employing eigenvectors, and utilizing ensemble classifiers. The efficacy of this approach was validated through extensive experiments on a variety of datasets, including benchmark datasets, real-world application streams, and synthetic data.

A key strength of this approach is the pivotal role played by the ADWIN algorithm, which enables the timely detection of concept drifts and changes in data patterns. This allows the framework, referred to as HTL, to adapt by training new base classifiers, ensuring their continued relevance and accuracy in real-time scenarios. The use of ensemble classifiers further enhances predictive performance and robustness by aggregating predictions from multiple base classifiers. Dynamic Ensemble Selection (DES) is utilized to select the most suitable base classifiers, optimizing overall performance.

Our thorough evaluation using various performance measures confirms the approach's effectiveness in addressing the challenges of heterogeneous multisource domains. The framework excels in accurately classifying data streams with evolving class distributions, particularly when leveraging ADWIN as the drift detector, eigenvectors for knowledge transfer across heterogeneous domains, and DES for classifier selection.

Beyond its strong accuracy and performance, the framework offers significant advantages in terms of adaptability, efficiency, and scalability. The ability to update the model dynamically, based on incoming data instances, ensures continuous adaptation to changing data distributions, maintaining the framework's reliability and relevance in real-time scenarios.

Looking ahead, future research could focus on further enhancing this approach. Potential improvements include developing advanced concept drift detection methods, refining classifier selection by focusing on positive target and multisource classifiers, and incorporating deep learning techniques to predict future drifts and optimize classifier performance. These advancements could further bolster the framework's capability to manage complex, evolving data streams in diverse real-world applications.

# Bibliography


[1] C. Yang, Y.-m. Cheung, J. Ding, and K. C. Tan, "Concept drift-tolerant transfer learning in dynamic environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3857–3871, 2021.

[2] B. Dong, Y. Gao, S. Chandra, and L. Khan, "Multistream classification with relative density ratio estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3478–3485.

[3] J. Shan, H. Zhang, W. Liu, and Q. Liu, "Online active learning ensemble framework for drifted data streams," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 486–498, 2018.

[4] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[5] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[6] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[7] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International journal of pattern recognition and artificial intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[8] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Addressing imbalance in multilabel classification: Measures and random resampling algorithms," *Neurocomputing*, vol. 163, pp. 3–16, 2015.

[9] ——, "Mlsmote: Approaching imbalanced multilabel learning through synthetic instance generation," *Knowledge-Based Systems*, vol. 89, pp. 385–397, 2015.

[10] Z. Daniels and D. Metaxas, "Addressing imbalance in multi-label classification using structured hellinger forests," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[11] B. Liu and G. Tsoumakas, "Making classifier chains resilient to class imbalance," in *Asian Conference on Machine Learning*. PMLR, 2018, pp. 280–295.

[12] N. Japkowicz, C. Myers, M. Gluck *et al.*, "A novelty detection approach to classification," in *IJCAI*, vol. 1. Citeseer, 1995, pp. 518–523.

[13] V. López, A. Fernández, J. G. Moreno-Torres, and F. Herrera, "Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. open problems on intrinsic data characteristics," *Expert Systems with Applications*, vol. 39, no. 7, pp. 6585–6608, 2012.

[14] M.-L. Zhang, Y.-K. Li, H. Yang, and X.-Y. Liu, "Towards class-imbalance aware multi-label learning," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, pp. 4459–4471, 2020.

[15] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*. Springer, 2003, pp. 107–119.

[16] S. Wang, H. Chen, and X. Yao, "Negative correlation learning for classification ensembles," in *The 2010 international joint conference on neural networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[17] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[18] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195–216, 2018.

[19] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, 2012.

[20] L. I. Kuncheva, "Clustering-and-selection model for classifier combination," in *KES'2000. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies. Proceedings (Cat. No. 00TH8516)*, vol. 1. IEEE, 2000, pp. 185–188.

[21] T. Woloszynski and M. Kurzynski, "A probabilistic model of classifier competence for dynamic ensemble selection," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2656–2668, 2011.

[22] R. Lysiak, M. Kurzynski, and T. Woloszynski, "Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers," *Neurocomputing*, vol. 126, pp. 29–35, 2014.

[23] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti, "Meta-des. oracle: Meta-learning and feature selection for dynamic ensemble selection," *Information fusion*, vol. 38, pp. 84–103, 2017.

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

[25] N. Lu, J. Lu, G. Zhang, and R. L. De Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence*, vol. 230, pp. 108–133, 2016.

[26] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[27] V. Losing, B. Hammer, and H. Wersing, "Knn classifier with self adjusting memory for heterogeneous concept drift," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 291–300.

[28] A. Storkey, "When training and test sets are different: characterizing learning transfer," 2008.

[29] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[30] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. d. Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–31, 2013.

[31] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 2, no. 5-6, pp. 311–327, 2009.

[32] C. Alippi and M. Roveri, "Just-in-time adaptive classifiers—part i: Detecting nonstationary changes," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145–1153, 2008.

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

[34] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 2, pp. 324–334, 2016.

[35] T. D. S. K. S. Venkatasubramanian and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams."

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

[37] M. Yamada, A. Kimura, F. Naya, and H. Sawada, "Change-point detection with feature selection in high-dimensional time-series data," in *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[38] S. H. Bach and M. A. Maloof, "Paired learners for concept drift," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 23–32.

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

[40] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, pp. 433–449, 2017.

[41] D. Liu, Y. Wu, and H. Jiang, "Fp-elm: An online sequential learning algorithm for dealing with concept drift," *Neurocomputing*, vol. 207, pp. 322–334, 2016.

[42] D. Han, C. Giraud-Carrier, and S. Li, "Efficient mining of high-speed uncertain data streams," *Applied Intelligence*, vol. 43, pp. 773–785, 2015.

[43] S. G. Soares and R. Araújo, "An adaptive ensemble of on-line extreme learning machines with variable forgetting factor for dynamic system prediction," *Neurocomputing*, vol. 171, pp. 693–707, 2016.

[44] B. F. Manly and D. Mackenzie, "A cumulative sum type of method for environmental monitoring," *Environmetrics: The official journal of the International Environmetrics Society*, vol. 11, no. 2, pp. 151–166, 2000.

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

[46] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 359–364.

[47] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavalda, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 139–148.

[48] F. Chu and C. Zaniolo, "Fast and light boosting for adaptive mining of data streams," in *Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004. Proceedings 8*. Springer, 2004, pp. 282–292.

[49] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, pp. 1469–1495, 2017.

[50] P. Li, X. Wu, X. Hu, and H. Wang, "Learning concept-drifting data streams with random ensemble decision trees," *Neurocomputing*, vol. 166, pp. 68–83, 2015.

[51] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[52] R. Elwell and R. Polikar, "Incremental learning of concept drift in non-stationary environments," *IEEE transactions on neural networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[53] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 81–94, 2013.

[54] P. Zhang, X. Zhu, and Y. Shi, "Categorizing and mining concept drifting data streams," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 812–820.

[55] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1532–1545, 2016.

[56] J. B. Gomes, M. M. Gaber, P. A. Sousa, and E. Menasalvas, "Mining recurring concepts in a dynamic feature space," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 95–110, 2013.

[57] Z. Ahmadi and S. Kramer, "Modeling recurring concepts in data streams: a graph-based framework," *Knowledge and Information Systems*, vol. 55, pp. 15–44, 2018.

[58] M. Pratama, J. Lu, and G. Zhang, "Evolving type-2 fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 574–589, 2015.

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

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

[61] H. Yang and S. Fong, "Incrementally optimized decision tree for noisy big data," in *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2012, pp. 36–44.

[62] ——, "Countering the concept-drift problems in big data by an incrementally optimized stream mining model," *Journal of Systems and Software*, vol. 102, pp. 158–166, 2015.

[63] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision trees for mining data streams based on the mcdiarmid's bound," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1272–1279, 2012.

[64] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "A new method for data stream mining based on the misclassification error," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 5, pp. 1048–1059, 2014.

[65] I. Frias-Blanco, J. del Campo-Avila, G. Ramos-Jimenez, A. C. Carvalho, A. Ortiz-Díaz, and R. Morales-Bueno, "Online adaptive decision trees based on concentration inequalities," *Knowledge-Based Systems*, vol. 104, pp. 179–194, 2016.

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

[67] A. H. Madkour, A. Elsayed, and H. Abdel-Kader, "Historical isolated forest for detecting and adaptation concept drifts in nonstationary data streaming," *IJCI. International Journal of Computers and Information*, vol. 10, no. 2, pp. 16–27, 2023.

[68] C. H. Tan, V. C. Lee, and M. Salehi, "Information resources estimation for accurate distribution-based concept drift detection," *Information Processing & Management*, vol. 59, no. 3, p. 102911, 2022.

[69] J. N. Adams, S. J. van Zelst, T. Rose, and W. M. van der Aalst, "Explainable concept drift in process mining," *Information Systems*, vol. 114, p. 102177, 2023.

[70] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[71] K. Jackowski, B. Krawczyk, and M. Woźniak, "Improved adaptive splitting and selection: the hybrid training method of a classifier based on a feature space partitioning," *International journal of neural systems*, vol. 24, no. 03, p. 1430007, 2014.

[72] T. Yin, C. Liu, F. Ding, Z. Feng, B. Yuan, and N. Zhang, "Graph-based stock correlation and prediction for high-frequency trading systems," *Pattern Recognition*, vol. 122, p. 108209, 2022.

[73] J. Ren, Y. Wang, Y.-m. Cheung, X.-Z. Gao, and X. Guo, "Grouping-based oversampling in kernel space for imbalanced data classification," *Pattern Recognition*, vol. 133, p. 108992, 2023.

[74] V. C. Nitesh, "Smote: synthetic minority over-sampling technique," *J Artif Intell Res*, vol. 16, no. 1, p. 321, 2002.

[75] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[76] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in knowledge discovery and data mining: 13th Pacific-Asia conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 proceedings 13*. Springer, 2009, pp. 475–482.

[77] T. Maciejewski and J. Stefanowski, "Local neighbourhood extension of smote for mining imbalanced data," in *2011 IEEE symposium on computational intelligence and data mining (CIDM)*. IEEE, 2011, pp. 104–111.

[78] Y. Gao, S. Chandra, Y. Li, L. Khan, and T. Bhavani, "Saccos: A semi-supervised framework for emerging class detection and concept drift adaption over data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 3, pp. 1416–1426, 2020.

[79] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Transactions on knowledge and data engineering*, vol. 23, no. 6, pp. 859–874, 2010.

[80] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[81] X. Mu, F. Zhu, J. Du, E.-P. Lim, and Z.-H. Zhou, "Streaming classification with emerging new class by class matrix sketching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[82] X. Mu, K. M. Ting, and Z.-H. Zhou, "Classification under streaming emerging new classes: A solution using completely-random trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1605–1618, 2017.

[83] Y.-N. Zhu and Y.-F. Li, "Semi-supervised streaming learning with emerging new labels," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 7015–7022.

[84] X.-Q. Cai, P. Zhao, K.-M. Ting, X. Mu, and Y. Jiang, "Nearest neighbor ensembles: An effective method for difficult problems in streaming classification with emerging new classes," in *2019 IEEE international conference on data mining (ICDM)*. IEEE, 2019, pp. 970–975.

[85] J. Zhang, T. Wang, W. W. Ng, and W. Pedrycz, "Knnens: A k-nearest neighbor ensemble-based method for incremental learning under data stream with emerging new classes," *IEEE transactions on neural networks and learning systems*, vol. 34, no. 11, pp. 9520–9527, 2022.

[86] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2200–2207.

[87] ——, "Transfer joint matching for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1410–1417.

[88] Q. Sun, R. Chattopadhyay, S. Panchanathan, and J. Ye, "A two-stage weighting framework for multi-source domain adaptation," *Advances in neural information processing systems*, vol. 24, 2011.

[89] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 193–200.

[90] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *icml*, vol. 96. Citeseer, 1996, pp. 148–156.

[91] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 1855–1862.

[92] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[93] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He, "Deep subdomain adaptation network for image classification," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 4, pp. 1713–1722, 2020.

[94] M. Long, Y. Cao, Z. Cao, J. Wang, and M. I. Jordan, "Transferable representation learning with deep adaptation networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 12, pp. 3071–3085, 2018.

[95] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang, and Q. Yang, "Transfer learning with dynamic distribution adaptation," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 1, pp. 1–25, 2020.

[96] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2960–2967.

[97] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[98] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE transactions on neural networks*, vol. 22, no. 2, pp. 199–210, 2010.

[99] M. M. Rahman, C. Fookes, M. Baktashmotlagh, and S. Sridharan, "Correlation-aware adversarial domain adaptation and generalization," *Pattern Recognition*, vol. 100, p. 107124, 2020.

[100] E. Zhong, W. Fan, J. Peng, K. Zhang, J. Ren, D. Turaga, and O. Verscheure, "Cross domain distribution adaptation via kernel mapping," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1027–1036.

[101] Z. Cao, M. Long, J. Wang, and M. I. Jordan, "Partial transfer learning with selective adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2724–2732.

[102] Z. Cao, K. You, M. Long, J. Wang, and Q. Yang, "Learning to transfer examples for partial domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2985–2994.

[103] L. Li, Z. Wan, and H. He, "Dual alignment for partial domain adaptation," *IEEE transactions on cybernetics*, vol. 51, no. 7, pp. 3404–3416, 2020.

[104] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *arXiv preprint arXiv:2010.16061*, 2020.

[105] A. Liu, Y. Song, G. Zhang, and J. Lu, "Regional concept drift detection and density synchronized drift adaptation," in *IJCAI International Joint Conference on Artificial Intelligence*, 2017.

[106] Q. Da, Y. Yu, and Z.-H. Zhou, "Learning with augmented class by exploiting unlabeled data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, no. 1, 2014.

[107] Y. Sasaki *et al.*, "The truth of the f-measure. teach tutor mater, 1 (5), 1–5," 2007.

[108] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 3121–3124.

[109] M. Kubat, S. Matwin *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97, no. 1. Citeseer, 1997, p. 179.

[110] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[111] P. Ksieniewicz and P. Zyblewski, "Stream-learn—open-source python library for difficult data stream batch analysis," *Neurocomputing*, vol. 478, pp. 11–21, 2022.

[112] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell, "Characterizing and avoiding negative transfer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 293–11 302.

[113] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 114.

[114] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh, "Sample selection bias correction theory," in *International conference on algorithmic learning theory*. Springer, 2008, pp. 38–53.