# Detecting Novel Concepts in Data Streams: To Infinity and Beyond

by

Jean-Gabriel Gaudreault

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

**Examining Committee**

The following served on the Examining Committee for this thesis.

External Member: Elaine Ribeiro de Faria
Associate Professor, Faculty of Computing
Federal University of Uberlândia

Carleton Member: Ashraf Matrawy
Professor, School of Information Technology
Carleton University

Internal Member: Burak Kantarci
Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Internal Member: Lionel Briand
Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Supervisor: Paula Branco
Associate Professor, School of Electrical Engineering & Computer Science
University of Ottawa

## Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

## Abstract

As new network attacks develop, new illnesses spread, or new topics surface on social media, many machine learning models become obsolete as they fail to recognize these novel concepts. This requires the development of artificial intelligence models capable of detecting, learning, and adapting to new concepts autonomously. In this context, novelty detection emerges as a critical method for identifying new concepts in data streams while ensuring the accurate classification of known ones. Despite growing research interest, progress in the field is hindered by challenges such as the lack of a comprehensive evaluation framework and the absence of robust algorithms that can adapt to changing data distributions with minimal human intervention. This thesis addresses these fundamental issues through four primary contributions.

First, we present a systematic literature review that establishes an updated taxonomy of the field, analyzing existing works to identify key challenges and research directions. Second, building on this review, we address inconsistencies in current evaluation methods by introducing a comprehensive framework for assessing novelty detection algorithms in multi-class data streams. We empirically demonstrate that key data stream characteristics, which are often overlooked, substantially impact algorithm performance and hinder fair comparisons across studies. Our framework formalizes these characteristics and introduces novel temporal metrics, enabling robust model comparison through a single, evolving performance score.

Third, to address the limitations of existing algorithms, we introduce CASCADE (Clustering-based Adaptive Stream Classification And Detection of Emerging classes). CASCADE integrates binary classifiers and threshold-based filters to mitigate false positives, while leveraging hierarchical clustering on custom meta-features to function effectively without dependence on ground-truth labels or extensive hyperparameter tuning. It achieves state-of-the-art performance, frequently matching or surpassing supervised approaches on diverse real-world and synthetic datasets. Finally, we demonstrate the effectiveness and advantages of novelty detection techniques in the domain of cybersecurity, using numerous benchmark datasets.

Altogether, this thesis provides a comprehensive overview of the current state of the field, establishes a rigorous methodology for performance evaluation, and introduces a high-performing algorithm that addresses critical limitations of existing techniques and matches or even surpasses supervised methods while operating without ground-truth labels or extensive hyperparameter tuning.

**Acknowledgements**

First of all, I would like to extend my heartfelt gratitude to my supervisor, Dr. Paula Branco, for whom the title of supervisor is definitely an understatement. While this journey has not been easy, I thank you for pushing me to go through with it and for your support, guidance, and patience throughout. It has been a pleasure and an honour for me to work under your supervision, and I am truly grateful for it.

I am forever grateful to my wife, Aisha, for whom long nights and stressful deadlines have been all too familiar. Thank you for your unwavering love and support, which helped me persevere throughout this journey. I also want to extend my thanks and gratefulness to my family, my parents, Alain and Chantal, and to the memory of my brother Yann, who always stays by my side. Thank you for believing in me, for your sacrifices, and for your endless love.

I also thank all the professors who helped me with guidance, in particular Dr. João Gama, as well as the thesis committee members. Thank you for your suggestions, discussions, and constructive criticisms throughout the process of this thesis, which helped shape and enhance the work included here.

Lastly, I want to extend my thanks to my friends and colleagues, in particular from the uOttawa-IBM Cyber Range, for sharing wins, failures, and sometimes grievances, as well as discussing ideas and providing me with new research directions.

The work presented in this thesis was supported by the University of Ottawa, the Government of Ontario through the Ontario Graduate Scholarship, the Natural Sciences and Engineering Research Council of Canada, and by computing capacity provided by the Digital Research Alliance of Canada.

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

**AUC** Area Under the Curve xii, 48, 60–62, 68, 69, 79, 83–86, 92, 97

**CNN** Convolutional Neural Network 28, 31, 36

**DDoS** Distributed Denial-of-Service 136, 141, 159

**DNN** Deep Neural Network 31, 131

**DS** Data Stream 1–4, 6–10, 12, 13, 16, 17, 19, 22, 24, 26–31, 35, 36, 38, 40, 41, 43, 45, 47, 49–53, 64, 67, 97–100, 107, 109, 110, 112–114, 119, 120, 125, 126, 128, 130, 150–154, 160–162, 165, 166, 169, 171–173, 175

**IDS** Intrusion Detection System 130–132, 134, 138–141, 146, 150, 152, 175

**IoT** Internet of Things 7, 98

**ML** Machine Learning 1, 2, 6, 14, 17, 24, 27, 30, 31, 54, 55, 60, 69, 77, 87, 130, 131, 149, 150

**ND** Novelty Detection 2–17, 19–24, 27–31, 35, 36, 40, 41, 43, 47–54, 67, 97–107, 109–111, 113, 122, 127–130, 135, 139, 150–154, 157, 159–161, 165, 166, 168, 169, 171–175

**NP** Novelty Pattern 2, 8–11, 21, 22, 49, 101, 103–106, 108, 109, 112, 127, 135, 139, 157, 158, 160, 163, 168, 169

**OSR** Open Set Recognition 21, 130–133, 140, 149

**PR** Precision-Recall xii, 55, 61–63, 68, 73, 79, 83–86, 97

**PRG** Precision-Recall-Gain 63, 79, 83, 86, 89, 92, 97

**ROC** Receiver Operating Characteristic 48, 60, 61, 63, 79, 83

**SLR** Systematic Literature Review 3, 6, 11, 12, 14, 54, 172, 174

**SVM** Support Vector Machine 69

# List of Symbols

AIC Akaike Information Criterion (cf. Equation (5.9)) 105, 108, 124

AMI Adjusted Mutual Information (cf. Equation (5.12)) xiii, xvi, 49, 108, 110, 113, 114, 117–119, 121, 122, 124, 126, 127, 160, 162–164, 166–169, 172

AP Average Precision (cf. Equation (4.7)) 62, 79, 84, 86

CER Combined Error (cf. Equation (5.8)) xvi, 104, 105, 108, 110, 122, 124

CWA Class Weighted Accuracy (cf. Equation (4.3)) 56, 59, 66, 68, 79, 82, 91

$Err$ Total misclassification error percentage (cf. Table 3.9) 48, 102

FE Known class instances that were misclassified (other than FP) 47, 48, 101, 102

FNR False Negative Rate (cf. Table 3.9) 105

FN False Negative 47, 101, 103

FPR False Positive Rate (cf. Table 3.9) 48, 60, 61, 105, 134, 135, 139, 155, 156

FP False Positive 47, 48, 61, 101, 103

$F_\beta$ F-measure (cf. Equation (4.1)) 47, 48, 55, 59, 65, 102

$F_{\mathbf{new}}$ Percentage of known class samples misclassified as novel (cf. Table 3.9) 31, 48, 68, 103, 108, 160

G-Mean Geometric Mean (cf. Equation (4.2)) xiv, 55, 59, 65, 68, 79, 82, 89–92, 97

IR Imbalance Ratio (cf. Equation (4.23)) xi, xiv, xv, 69–73, 75, 77, 79, 84, 85, 88–94

MCC Matthews Correlation Coefficient (cf. Equation (4.5)) xiv, 56, 60, 66, 68, 76, 77, 79, 85, 86, 97

MD Mode Difference (cf. Equation (4.25)) xi, xii, 77–86

$M_\beta$ M-measure (cf. Equation (5.11)) 108, 113, 127, 160, 172

$M_{\mathbf{new}}$ Percentage of novel class samples misclassified as known (cf. Table 3.9) 31, 48, 68, 103, 108, 160

$N_c$ Number of novel samples in the streams 47, 48, 101

$N$ Number of samples 47, 48, 101

TN True Negative 47, 48, 61, 76, 82, 101

TPR True Positive Rate (cf. Table 3.9) 48, 134, 135, 138, 139, 156

TP True Positive 47, 48, 62, 76, 101, 103

TTC Time To Classification (cf. Equation (5.15)) xvi, 109, 110, 125–127, 172

TTD Time To Detection (cf. Equation (5.13)) xii, xiii, 49, 109, 110, 125, 127, 160, 163, 168–170, 172

$Unk$ Unknown Rate (cf. Equation (5.10)) 106

kappa kappa (cf. Equation (4.4)) xiv, 56, 59, 60, 66, 68, 76, 77, 79, 85, 86, 97

# Chapter 1

# Introduction

Most Machine Learning (ML) research focuses on batch learning, where the entire training dataset is expected to be available at once [82]. This particular method of learning assumes that the probability distribution observed during training will remain constant throughout the deployment of the model. In practice, however, many real-world scenarios exhibit dynamic and evolving data distributions, where novel concepts that were not available during training may emerge.

To address such cases, Data Streams (DSs) provide an alternative way of presenting data, where observations arrive continuously, sequentially, and potentially without bound [190], as opposed to a fixed-length static dataset. This approach is more conducive to representing a broad spectrum of real-world applications, such as sensor networks, GPS data, network traffic, among others [82]. However, applying traditional learning methods to DSs is afflicted by numerous challenges, including changes in the data distribution (concept drift), memory limitations, and the emergence of novel classes. As such, the fields of incremental learning and online learning, which develop algorithms to tackle these issues in an online setting, have gained traction in the research sphere [98].

## 1.1 Motivation

In an online scenario, not only can learned classes change, but entirely new classes that were not present at the beginning might appear. This makes it essential to develop algorithms that can detect these novel concepts and adapt to them. This capability is critical in many applications, including intrusion detection, fault monitoring, medical diagnosis, and fraud

detection [71], where a novelty might represent a zero-day attack, a new illness, or a new type of fraud. A model incapable of detecting these novel concepts would misclassify them as known classes, allowing them to go unnoticed, which could have serious implications in these critical domains.

This challenge has led to the development of a sub-field of online learning that specifically focuses on identifying these samples that differ from the known classes: Novelty Detection (ND) in DSs [105]. In contrast to traditional batch ML, ND algorithms operate without assuming that the training data fully defines the problem space. Instead, they actively attempt to identify, categorize, and possibly learn the new classes that appear in the DS. They accomplish this by recognizing collections of similar, unrepresented samples, referred to as Novelty Patterns (NPs) [55]. These clusters can be treated collectively as a single unknown class, in which case the task is a binary classification between known and novel data. Alternatively, in multi-class classification, the model attempts to identify each distinct novel class independently [71]. While a single novel class generally generates multiple NPs, an effective algorithm should ideally minimize this number.

The importance of ND in real-world scenarios has spurred a surge in recent research, with numerous works exploring different techniques to detect unknown classes (e.g. [36, 51, 64, 68, 85, 120, 206]). Comparing these approaches, however, remains difficult. They often make different assumptions about DSs' characteristics, such as data arrival order and true label availability. This issue is exacerbated by a lack of standardized performance metrics for the domain and the use of evaluation methods ill-suited to the ND context.

Moreover, ND algorithms must face other challenges beyond concept drift. Emerging classes may appear sporadically and be mistaken for noise, or they may appear and disappear at irregular intervals, potentially being confused for multiple distinct novel classes. While various propositions have been put forward to mitigate some of these problems, a definitive solution has not yet been identified.

## 1.2 Objectives

Figure 1.1 provides an overview of all the objectives of this thesis and illustrates how each chapter addresses them. In this section, we summarize each objective and explain its significance in the context of this research.

To make relevant contributions to the field, it is necessary to begin with a clear and comprehensive understanding of its current state. Therefore, **the first objective of this thesis is to review, categorize, and provide an updated taxonomy of recent**

**works in the field**. To this end, we conducted a comprehensive Systematic Literature Review (SLR) of over 150 publications in the field, highlighting the architectures of current algorithms, ongoing challenges, and potential directions for future research.

Building on this, and considering the difficulties of comparing algorithms noted in the previous section, **our second objective is to propose an updated and standardized evaluation framework for ND algorithms that accounts for key data characteristics of DSs**. This framework introduces novel metrics for aspects that were previously overlooked, such as the time required to detect novel classes, and enables more reliable comparison between models using standardized metrics. Moreover, by systematically testing how different data characteristics impact performance, this framework provides guidance to future researchers on proper evaluation and reporting practices.

To address limitations of existing techniques in terms of performance and hyperparameter tuning, **our third objective is to propose and evaluate novel algorithmic approaches for ND in DSs**. Specifically, this work investigates the use of meta-classifiers, meta-features, and clustering to construct a new architecture tailored to the challenges of ND.

Finally, **our fourth objective is to demonstrate the practical application of ND algorithms in the context of cybersecurity**. In this domain, ND models can improve the detection of zero-day attacks with minimal human intervention. While prior work exists in this area, it has largely relied on synthetic or outdated datasets which do not reflect the complexity of the modern cyberspace. Our aim is to bridge this gap by evaluating these algorithms on contemporary datasets.

## 1.3 Contributions

Considering the four aforementioned objectives as well as our current publications, this thesis makes the following key contributions:

**Review, categorize, and provide an updated taxonomy of recent works in the field**

- A formalized, standardized definition of ND.
- An analysis of the field's main challenges and a breakdown of existing solutions.
- A novel taxonomy for the classification and comparison of ND algorithms.

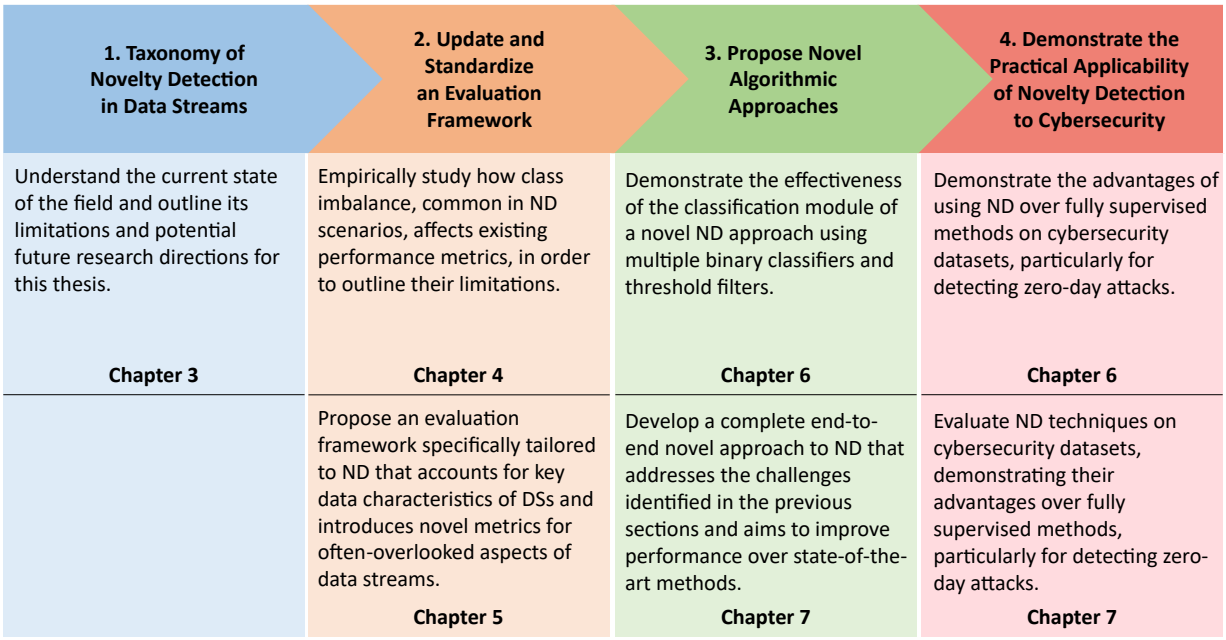



Figure 1.1: Overview of the thesis objectives and sub-objectives

**Propose an updated and standardized evaluation framework for novelty detection algorithms that accounts for key data characteristics of data streams**

- An extensive empirical study of performance metrics for imbalanced scenarios, evaluating their robustness to noise and class imbalance, which are common challenges in ND.
- A novel evaluation framework that enhances model comparability by incorporating diverse DSs' characteristics.
- An empirical demonstration of how various DS characteristics impact the performance of ND algorithms.
- The development and publication of StreamNDR, an open-source Python package providing an implementation of multiple ND algorithms that were not available [89].

**Propose and evaluate novel algorithmic approaches for novelty detection in data streams**

- The design and implementation of CASCADE, a novel unsupervised architecture for ND that does not rely on ground-truth labels.
- The introduction of a novel hierarchical clustering mechanism based on meta-features within the CASCADE framework, enabling the robust detection and characterization of multiple, simultaneous novel classes with minimal hyperparameter tuning.

**Demonstrate the practical application of ND algorithms in the context of cybersecurity**

- A practical application of ND algorithms in a real-world case study of network intrusion detection.
- A demonstration of the advantage of ND algorithms over traditional supervised classifiers across multiple contemporary intrusion detection datasets.

### 1.3.1 Publications

The research presented in this thesis has contributed to several peer-reviewed publications, each corresponding to one or more of the objectives and contributions outlined in the previous section. The following works, listed in chronological order, have been published or are currently under review:

(i) Jean-Gabriel Gaudreault, Paula Branco, and João Gama. An analysis of performance metrics for imbalanced classification. In Carlos Soares and Luis Torgo, editors, *Discovery Science*, pages 67–77, Cham, 2021. Springer International Publishing

(ii) Jean-Gabriel Gaudreault and Paula Branco. Toward streamlining the evaluation of novelty detection in data streams. In Albert Bifet, Ana Carolina Lorena, Rita P. Ribeiro, João Gama, and Pedro H. Abreu, editors, *Discovery Science*, pages 703–717, Cham, 2023. Springer Nature Switzerland

(iii) Jean-Gabriel Gaudreault and Paula Branco. A systematic literature review of novelty detection in data streams: Challenges and opportunities. *ACM Comput. Surv.*, 56(10), May 2024

(iv) Jean-Gabriel Gaudreault and Paula Branco. Empirical analysis of performance assessment for imbalanced classification. *Machine Learning*, 113(8):5533–5575, Aug 2024

(v) Jean-Gabriel Gaudreault and Paula Branco. Detection of novel cyberattacks using multi-binary classifiers. In *2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, pages 1–9, 2024

(vi) Jean-Gabriel Gaudreault and Paula Branco. Prioritizing the essential: A robust evaluation framework for novelty detection. *Machine Learning*, 114(6):143, Apr 2025

(vii) Jean-Gabriel Gaudreault and Paula Branco. Cascade: Clustering-based adaptive stream classification and detection of emerging classes. unpublished, To Be Submitted to SIAM International Conference on Data Mining 2026, N.D

## 1.4 Thesis Organization

The rest of this thesis is structured as follows: Chapter 2 provides the necessary background and defines the main concepts related to ND used throughout this thesis. Chapter 3 presents our SLR of the field, providing essential background information and an analysis of current challenges. Chapter 4 provides an extensive empirical study of performance metrics in imbalanced scenarios and offers recommendations for their use in different contexts. Chapter 5 introduces our proposed framework for evaluating ND algorithms and analyzes the impact of DS characteristics on their performance. Chapter 6 describes preliminary work on a novel ND architecture based on multiple binary classifiers and demonstrates the advantage of ND over traditional ML approaches on intrusion datasets. Chapter 7 outlines the design of our final proposed architecture, CASCADE, a fully unsupervised ND approach, and empirically demonstrates its competitive performance compared to other state-of-the-art methods on both intrusion and general datasets. Finally, Chapter 8 concludes the thesis by summarizing the findings, acknowledging limitations, and identifying future research directions for the field of ND.

# Chapter 2

# Background

This chapter consolidates background information on ND that appeared across our publications, in order to avoid repetition and provide the reader with the key concepts related to ND needed to understand the remainder of this thesis.

## 2.1 Concept Definitions

### 2.1.1 Data Stream

A DS can be described as a continuous, ordered, possibly unbounded set of examples arriving at different timestamps [82, 190]. These samples can be ordered at regular intervals or be more sporadic, and do not necessarily present a dependence between subsequent samples [71]. Numerous examples of real-life data sources naturally produce DSs, such as Internet of Things (IoT) devices, sensor networks, network traffic, shopping transactions, and others, which makes this a prevalent field of study. DS mining contrasts with the traditional batch learning scenario, as it has specific requirements related to its online nature and unboundedness, such as a limited prediction time and restricted memory availability [27].

DSs have been formally defined in several works (e.g. [4, 56, 190]) and can be represented as a stream $S$, which consists of a potentially unbounded sequence of samples, $S = \{x_1, x_2, \ldots, x_M, \ldots\}$. Each sample $x_i$ arrives at an associated timestamp $t_i \in \{t_1, t_2, \ldots, t_M, \ldots\}$ and is represented as a $d$-dimensional feature vector $\bar{x}_i = [x_i^1, x_i^2, \ldots, x_i^d]$, along with a corresponding label $y_i$ that specifies the class to which the sample belongs [56].

### 2.1.2 Novel Class or Concept Evolution

A novel class represents any class or concept, which was not present during the initial training phase of the model, but rather only appeared later, after the deployment, or online phase of the model. It is also sometimes referred to as a concept evolution [147] or as an anomaly, referring to the fact that it usually represents an irregularity, although this is not always the case. To avoid any confusion with the separate problem of anomaly detection [82], which we discuss in Chapter 3, we will not use the term "anomaly".

While the terms "class" and "concept" are often used interchangeably in the context of ND, there is an important distinction between them. Similar to supervised learning, a class refers to the true label assigned to each sample $x_i$, where the label $y_i \in Y$ and $Y$ represents all possible $N$ classes, that is $Y = \{C_1, C_2, C_3, \ldots, C_N\}$. In contrast, a concept refers to the underlying distribution of the data, which may shift over time [81], as defined in the definition of concept drift below. Although this distinction may influence certain applications, both terms are used interchangeably in this thesis, as it does not impact our specific context. When referring to a known class or concept, we say that a class is known if it denotes a class available during the training of the algorithm, such that its label satisfies $y_i \in Y_{\text{known}}$ with $Y_{\text{known}} \subset Y$ [56, 92].

### 2.1.3 Novelty Pattern

While the previously defined novel classes correspond to the true labels of the classes not present in the training phase, NPs are defined as groups of cohesive samples identified by the classification algorithm, hence, an NP does not directly refer to the true labels of the samples, but rather to the different groups of samples that were identified by the algorithm as "new" [55]. Additionally, an NP may indicate other phenomena, such as outliers, noise, or shifts in the distribution of known classes [56]. Consequently, although in the best-case scenario, the algorithm would detect each novel class as only one NP, in practice a single novel class is often detected as multiple NPs.

### 2.1.4 Concept Drift

As typical data sources in DSs generate data from distributions that are not stationary [73, 82], known concepts or classes that were learned by the model originally might shift over time. This behaviour, called concept drift, is one of the numerous challenges

that ND algorithms face. Indeed, once a concept shifts far enough from its initial distribution, it could be considered a novel class by the algorithm while in fact being a known class. The literature identifies four types of drift, namely sudden, gradual, incremental and recurring, which fundamentally all involve a temporal shift in the data distribution [71,142]. Sudden drift occurs when the distribution shifts abruptly at a specific time $t$; gradual drift involves a slower transition between two distributions over time, with both distributions coexisting during a transitional period; incremental drift occurs gradually, with the distribution changing progressively over time; and recurring drift refers to concepts that disappear temporarily and later reappear in the DS [81, 245]. This issue is an important concern in ND, as the algorithm not only needs to identify novel classes appearing in the stream but also needs to distinguish them from changing known classes and update their representation over time [71].

### 2.1.5 Other Concepts

While not integral to ND, the following definitions are often used in relation to the algorithms which perform the ND task and are therefore included to provide more context to the reader when discussing these algorithms.

- ***Offline and Online Phases*** The ND problem can be generally divided into two stages, namely the offline and online phases [71]. These two steps represent, respectively, the inference of an initial model characterizing a set of known classes, and the use of that model in an online fashion to classify incoming samples from a DS as known or not.

- ***Time Window*** A time window represents a subset of data samples within our stream. A common assumption in DSs is that the most recent data coming in the stream is more relevant than older data [83]. As such, many techniques have adopted the usage of sliding windows, where, similar to a queue, only a limited amount of recent data samples will be stored and used for computation and older samples will be discarded in an ongoing fashion, thus avoiding the issue of limited memory in a potentially infinite DS. Different types of sliding window techniques exist. Babcock et al. [20] define two main types, namely sequence-based and timestamp-based windows, where the former determines the window size based on the number of samples, while the latter uses timestamps and a defined time duration.

- ***Buffer*** Before identifying NPs, numerous algorithms implement the idea of a buffer to temporarily store unknown samples that were excluded from the known classes.

This step ensures that enough samples are collected to adequately define each NP and allows for the differentiation of potential novel classes from noise or outliers.

- ***Forgetting Mechanism*** In ND, forgetting mechanisms can be defined as any type of technique used by the algorithm to forget outdated concepts [71]. As new data arrives and data distributions change, previously learned concepts can no longer represent the current DS and need to be discarded.

- ***Micro-Cluster*** As defined in the CluStream framework [4], micro-clusters can be described as an extension of a cluster feature vector [238] with a temporal aspect. They store a statistical summary of a set of data points with five different components, namely the sum of squares of the data values, the sum of the data values, the sum of squares of the timestamps, the sum of the timestamps, and the number of data points. As DSs can potentially be infinite, micro-clusters allow the storage of a representation of the data without needing an infinite amount of memory as new samples arrive.

## 2.2 Problem Statement

ND is the ability of a model to identify samples belonging to novel class(es) within a DS without prior knowledge of them. Depending on the algorithm, these samples may be clustered into distinct novel classes, and the model may then be updated to incorporate a novel class as a known one. The ND task typically involves two phases: an offline phase, during which only a subset of data is available to build an initial decision model, and an online phase, during which new data arrives continuously [56]. The core ND task occurs during the online phase and consists of detecting the emergence of new classes in the stream that were not observed during the offline phase.

The definition of ND can be somewhat contentious within the field, as it has historically been defined as a one-class classification problem in which the goal is only to distinguish known from novel classes [61, 82, 183]. However, in recent years, numerous studies have extended this definition to the multi-class scenario [1, 70, 71, 113, 150], where multiple novel classes can appear in the DS. It is also important to distinguish ND from other terms such as anomaly detection and outlier detection. While these terms are sometimes used interchangeably with ND, these terms often imply unwanted behaviours [71, 82]. This is not necessarily true with novel classes in ND. Furthermore, unlike outlier detection which focuses on the detection of individual samples characterizing abnormal behaviours [116],

ND aims to detect cohesive groups of samples that represent NPs. We survey existing definitions of ND and formalize the problem in the next chapter as part of our SLR.

# Chapter 3

# A Systematic Literature Review of Novelty Detection in Data Streams: Challenges and Opportunities

As mentioned in Chapter 1, one objective of this thesis is to review recent advances in ND, to organize existing approaches within an updated taxonomy, and to identify open research questions. To this end, this SLR surveys over 150 papers published between 2005 and 2022 in major computer science digital libraries. The chapter concludes with an analysis of the field's main challenges and a discussion on future research opportunities.

The contents of this chapter are based on the following paper:

Jean-Gabriel Gaudreault and Paula Branco. A systematic literature review of novelty detection in data streams: Challenges and opportunities. *ACM Comput. Surv.*, 56(10), May 2024

## 3.1 Introduction

Due to increasing interest in research in the field, there have been several studies discussing the application of ND in DSs. While some works have focused on specific solution types, such as ensemble [129], clustering [232], local outlier factor [16], or even specific challenges of ND in DSs, such as concept drift [5] and outlier detection [107], other reviews that are more comprehensive on the overall problem of ND [61, 69, 71, 82, 97, 100, 101, 167, 183] have also been published. However, these more comprehensive works have been either

published a considerable amount of time ago, which in a rapidly evolving field such as this one, implies that they do not include the latest trends and progress, or have been carried out in a non-systematic fashion, meaning that they do not include an extensive list of all proposals that have been published in the field.

In particular, we identified, in order of publication, the works of Faria et al. [71], Din et al. [61], and Salehi et al. [183] which all consist of comprehensive surveys that address ND as a whole and have been published the most recently. However, none of these works have been carried out in a systematic way, and therefore cover only a limited number of works that have been selected by the authors. Moreover, the most recent of these, the survey by Salehi et al., was first submitted in 2021, does not treat of ND specifically but rather the open-world setting as a whole, and adopts a definition of ND that differs from the one presented in this thesis.

Considering that, to the best of our knowledge, no systematic study of ND in DSs has been published, we wanted to provide the research community with an exhaustive analysis of the different proposals that have been suggested in recent years, while providing answers to key research questions on subjects such as challenges, leading researchers, and requirements, as well as an overview of how the field of ND for DSs has evolved throughout the recent years and recommendations on future research directions. Hence, this survey will differ significantly from previous works, by providing an updated, more high-level overview of the field of ND specifically, analysis and formalization of the definition of ND, and study of all recent publications in ND conducted in a systematic fashion.

**Contributions** The main contributions of this chapter can be summarized as follows:

(i) Provide a formalized, standardized definition of ND;

(ii) Provide an analysis of the advancement of ND in DSs in recent years and its main contributors;

(iii) Provide a breakdown of the main challenges faced in this field, as well as typical solutions implemented to mitigate them;

(iv) Propose an updated taxonomy that allows for the classification and comparison of every paper included in this survey;

(v) Provide an in-depth comparative analysis of the architecture of key solutions in the field of ND throughout the years, and their evolution; and

(vi) Provide a comprehensive set of opportunities for future work and a detailed description of challenges that still need to be tackled moving forward.

**Organization** The rest of this chapter will be structured as follows: Section 3.2 will present the methodology of our SLR and the different research questions we aim to answer; Section 3.3 will display our findings as well as answer the different research questions presented; Section 3.4 will discuss recommendations, open challenges, and possible opportunities for future work in the field; and Section 3.5 will conclude the chapter.

## 3.2 Methodology

Our methodology follows the framework for SLRs used by Kitchenham et al. [127], where research questions are first defined to delimit the scope and the questions the survey is ultimately trying to answer. Then, we establish our search process, our inclusion and exclusion criteria, our quality criteria, and finally our data collection process. The following sections describe all these steps in more detail.

### 3.2.1 Research Questions

Based on other surveys in related fields of ML [141, 217], we define our research questions and subsequent sub-questions on the four main steps of the software development life cycle applied to the development of ND algorithms, namely: (i) Background Analysis; (ii) Requirements Definition; (iii) Design; and (iv) Testing. We explain in more detail how these cycles apply to the creation of ND algorithms and how the following research questions relate to them in the sections below.

1. **Background Analysis**
   (a) **RQ1.1:** How is novelty detection in data streams defined?
   (b) **RQ1.2:** How has research activity evolved in the field in recent years?
   (c) **RQ1.3:** Who are the leading researchers in the field?
2. **Requirements Definition**
   (a) **RQ2.1:** What are the challenges and requirements of novelty detection in data streams?

(b) **RQ2.2:** What are the different applications of novelty detection in data streams?

3. **Design**

   (a) **RQ3.1:** How are proposed solutions dealing with the aforementioned challenges?

   (b) **RQ3.2:** What architectures do novelty detection algorithms implement?

   (c) **RQ3.3:** How are the most influential algorithms designed and how have they evolved throughout the years?

4. **Testing**

   (a) **RQ4.1:** In what context have these approaches been tested?

   (b) **RQ4.2:** What performance evaluation methods have been used to evaluate these strategies?

### Background Analysis

When developing new algorithms, whether it is in the field of ND or not, having a good understanding of the underlying concepts as well as already existing solutions is of the utmost importance to develop relevant research. As the definition of novelty and concept evolution can sometimes vary from study to study, establishing a clear definition for the term was paramount (RQ1.1) for our survey. Moreover, we wanted to provide a better understanding of how the field of ND evolved throughout the recent years (RQ1.2), as well as who are the researchers leading this advancement (RQ1.3).

### Requirements Definition

Before moving on to the development of ND strategies, it is important to know what are the requirements and applications of the strategies being developed. In the field of ND, it means having a good grasp of the challenges that may be encountered during development (RQ2.1) and what field the solution is aiming to tackle, for example, intrusion detection, image recognition, or a more general approach (RQ2.2).

#### Design

While previous sections dealt with the general context of where and why novelty algorithms are implemented, the design focuses on how they are implemented, such as the way they deal with the previously mentioned challenges (RQ3.1), the architecture used to build these systems (RQ3.2), and how the more influential algorithms were designed (RQ3.3) to give us a better idea of the general design trends.

#### Testing

Once the software or algorithm development is finished, it is important to test it in real-world scenarios. In the context of ND, it involves the use of different types of data (RQ4.1) to evaluate the algorithm in different contexts, as well as the use of field-specific metrics (RQ4.2) to assess the efficiency of the algorithm in performing the task it was developed to do, in this case, detecting novelties in DSs.

### 3.2.2 Search Criteria

In order to gather most documents relevant to the field of ND in DSs, we devised multiple research terms for the two main concepts of this field, which can be seen in Table 3.1. By combining these terms into the finalized search query displayed below, we used it to query four different databases and search engines relevant to the field at hand, namely Scopus, Web of Science, IEEE Xplore, and ACM Digital Library, while limiting the search to the title, abstract and keywords metadata.

```
(("novel* detection" OR "novel* class" OR "emerging class*" OR "concept
evolution" OR "open set recognition") AND ("data stream*" OR "real-time
analysis" OR "stream class*" OR "online learning" OR "stream mining"))
```

| Concept | Research Terms |
|---|---|
| Novelty Detection | "novel* detection", "novel* class", "concept evolution", "emerging class*", "open set recognition" |
| Data Streams | "data stream*", "real-time analysis", "stream class*", "online learning", "stream mining" |

Table 3.1: Concepts and their associated research terms

### 3.2.3 Inclusion and Exclusion Criteria

We included all relevant papers published between January 1st, 2005, and July 24th, 2022. We chose 2005 as our starting point since we only wanted to include studies that were published somewhat recently and since there are interesting proposals that were published within the field during the years 2005 to 2010 (e.g. [38, 152, 153]), which we wanted to include in our analysis. The selected studies respected the following inclusion criteria:

(i) Papers presenting novel algorithm(s) to perform ND in DSs with details on their architecture and operation.

(ii) Papers consisting of reviews, frameworks without implementation, surveys, or comparative analysis are included in this work in order to analyze challenges and future research directions that were identified in those works. However, these papers were not included in the analysis of our other research questions and they were not assessed with the following quality criteria.

While we did want to gather most papers related to the field of ND, we also established the following exclusion criteria in order to only gather research that was relevant to answer our established research questions:

(i) Papers presenting an algorithm that was designed to only handle concept drift but not able to detect novel classes in the DSs.

(ii) Papers in other languages than English.

### 3.2.4 Quality Assessment

While other SLRs often use the number of citations received as a quality criterion to establish the relevance/quality of a paper, we did not want to do so, as to not exclude more recent or lesser-known works that could be highly relevant to our survey. As such, we devised the following quality criteria and only included work that met all of them to ensure the quality of the selected papers:

(i) Paper has a clear explanation of the proposed algorithm's architecture or the used ML algorithms and is reproducible.

(ii) Paper uses a correct evaluation framework to evaluate the performance of the proposed algorithm, in other words, it has clear results presented on real-world and/or relevant synthetic data and does not exclusively use an improper evaluation metric such as accuracy, which will be discussed in further detail in Section 3.3.10.

### 3.2.5 Data Extraction

Using all of the previously defined criteria, we imported, analyzed, and filtered several papers, which can be seen in Figure 3.1. The data extracted from each paper was motivated by the research questions we previously established and was manually extracted from each paper after a comprehensive reading. The following information was extracted and assessed:

- Year of publication
- Area of publication *(Conference, Journal, ...)*
- Type of paper *(Survey, Proposal, Comparison, ...)*
- Authors' definition of novelty detection
- Main domain of application *(Intrusion Detection, Image Recognition, ...)*
- Type of task *(Unsupervised, Supervised, ...)*
- Number of classifier(s) *(single, ensemble)*
- Type of classifier(s) *(Clustering, Neural Network, ...)*
- Sub-type of classifier(s)
- Number of classifiable classes *(One-class, Multi-class)*
- Name of the proposed algorithm
- Usage of a buffer (Unknown Label) *(Yes, No)*
- Number of detectable novel classes *(One, Multiple)*
- Are the novel classes incorporated in the original classifier(s) *(Yes, No)*

- If the proposed solution addresses concept drift and if so, how
- Access to ground truth *(Never, Partial, All)*
- Dataset(s) used for experiments
- Other algorithms compared to
- Performance metrics used
- Potential future work(s) mentioned
- Limitation(s) of the proposed solution mentioned
- Strength(s) of the proposed solution mentioned

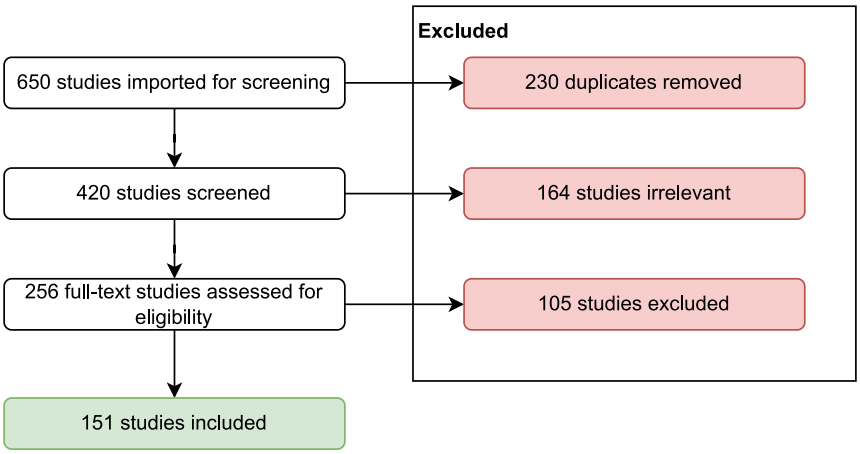



Figure 3.1: Number of extracted studies

## 3.3 Review

This section summarizes our findings for all research questions defined in Section 3.2.1. It begins by giving an overview of the definition and evolution of ND in DSs. Then, it goes over the challenges and applications of those concepts, the design as well as architecture of different proposals, and finally, an outline of the evaluation frameworks typically used in the field.

### 3.3.1 RQ1.1: How is novelty detection in data streams defined?

ND can, broadly speaking, be categorized as a form of detection where the model aims to identify samples belonging to other classes than the ones seen prior in training. However, its interpretation is often contentious between studies, and it is therefore paramount to understand if this definition holds true throughout the different analyzed studies. As such, we extracted from each study how the concept of ND was defined, and provide some insights in this section on its relation to other commonly used terms.

First, we display a co-occurrence map of the concepts within the extracted papers in Figure 3.2 in order to visualize the main concepts related to the field. The figure shows the 60% most relevant words that occur in the abstract or title of at least 15 publications. The relevancy of each word is used to eliminate common terms such as "conclusion" which do not provide useful information for our purpose and is calculated as described in [214]. We can observe in this figure the presence of the majority of terms related to ND that we have previously mentioned, such as "stream", "concept drift", "novel class", and "outlier", as well as other terms that are commonly used in the domain, such as "new class", "emergence", and "clustering". An interesting observation within the graph is the proximity of the terms "challenge", "effectiveness", "concept drift", and "concept evolution", which indicates that these words are often used closely together within the works analyzed. This could demonstrate that many authors primarily associate concept drift and concept evolution as the main challenges of ND, while other challenges might be less likely to be known or mentioned. Moreover, we can also note the presence of multiple terms related to class, such as "classification", "new class", "novel class", and "novel class detection", as well as the absence of terms related to anomaly detection, with the exception of "outlier". This second observation echoes the aforementioned distinction between ND and anomaly detection, as the former is mainly linked to terms related to class, while the latter is not present.

**Anomaly and Outlier Detection** While anomaly detection, outlier detection, and ND are often used interchangeably in the literature, we argue that, while related, they are fundamentally different in their meaning and we should avoid substituting them in order to avoid confusion. Indeed, commonly accepted definitions for these terms define anomaly detection as "the problem of finding patterns in data that do not conform to expected behavior" [41] and outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [116]. While the former definition represents the same overarching goal as ND, the terms anomaly and outliers are often used in the context of undesirable or abnormal behaviours [71, 82], this is especially obvious when thinking about some inherent challenges of anomaly detection such

as obtaining entirely normal training data, when noise is always a factor, a problem that is not applicable to all applications of ND [183]. In contrast, ND refers to the identification of new, emergent concepts or classes in the stream which are unknown to our model. The concepts of "class", "concept", and "pattern" are usually not well defined in the literature, as the definition of when a series of outliers/anomalies become a novel class is often blurry and left to the researchers' interpretation. However, a common consensus [104] is that a single occurrence of a sample that differs from other observations is treated as an outlier, while an agglomeration of those anomalies over a time period can become an NP. This observation is what differentiates the concepts of outlier detection and ND, while the former is interested in any observation which deviates from the normality, the latter is more interested in finding concise groups of these observations [82], which would be of interest for the user. Moreover, another important distinction between ND and anomaly detection is that ND often implies that the model can be updated with the detected concepts, which is not the case for anomaly detection [41, 71].

**One-Class Classification** Historically, ND was mainly defined in a one-class classification framework, where the known class or normal concept is classified as the positive class and the goal is to identify between the known and unknown classes [61, 82]. As such, the terms are sometimes used interchangeably, even in more recent contributions (e.g. [183]). However, in more recent years, ND has been extended to the multi-class scenario by multiple publications (e.g. [1, 70, 113]), where the algorithm is not only able to identify multiple known classes, but multiple novel ones as well. As such, in order to keep ND as a general concept and to encompass all possible implementations, we use one-class classification as a sub-division within ND, which we will define when discussing our proposed taxonomy in section 3.3.7.

**Open Set Recognition** Open Set Recognition (OSR) is a problem that has been well formalized in the literature [185] and typically used in the context of computer vision. It is defined as a task where the model does not have access to all classes at training time, can be given samples from unknown classes at testing time, and needs to be able to classify known classes as well as reject unknown ones from the target domain [143]. While the problem definition of OSR and ND resemble each other and there are a lot of overlap between the two, we argue that the ND problem differs slightly from OSR in its approach to novel classes. Indeed, while ND focuses on the identification of novel concepts and their potential integration into the model, OSR does not assume that the novel classes are even enumerated nor modeled [185].

**Formalization of Novelty Detection** As stated in the previous paragraphs, ND needs to be defined as its own definition, as it has essential differences with other terms. Specifically, we argue that in order to be defined as an ND algorithm, a given model needs to demonstrate at least the following two capabilities: (i) Ability to identify if a previously unseen sample is part of the distribution of known class(es) or not; and (ii) Ability to identify if a group of given points outside the distribution of known classes forms a cohesive group (concept or pattern). This second condition is particularly what distinguishes ND from outlier detection or certain types of anomaly detection, as the model needs to be able to detect NPs within the outliers rather than only the outliers themselves.

Using these requirements, we can formalize an ND algorithm as follows. Let us consider a set of all potential $N$ classes $Y = \{C_1, C_2, C_3, \ldots, C_N\}$, where only a subset of $K$ classes is available at training time, denoted as $Y_{\text{known}} \subset Y$, while the remaining, unknown classes are defined as $Y_{\text{unk}} = Y \setminus Y_{\text{known}}$. Given a sample $x_i$ arriving at time $t_i$, the objective of ND is to determine whether its label $y_i$ belongs to the known class set, i.e., $y_i \in Y_{\text{known}}$. Additionally, given a set of $L$ samples identified as not belonging to $Y_{\text{known}}$, denoted as $X_{\text{unk}} = \{x_{\text{unk1}}, x_{\text{unk2}}, \ldots, x_{\text{unk}L}\}$, i.e., samples for which the model assigns $y_i \notin Y_{\text{known}}$, the model can determine if a subset $X_{\text{nov}} \subseteq X_{\text{unk}}$ referred to as an NP, such that $X_{\text{nov}}$ forms a cohesive cluster in feature space that ideally corresponds to a single class $C_j \in Y_{\text{unk}}$. It is worth noting that "cohesive distribution" has not been formally defined and is open to interpretation.

While this formulation is defined for the multi-class scenario, it can also be applied to the binary case, where an algorithm could only detect between known and unknown with $K = 1$ and $N = 2$.

### 3.3.2 RQ1.2: How has research activity evolved in the field in recent years?

Due to its numerous real-world applications, the topic of ND in DSs has become more prevalent and researched in recent years. To quantify this research trend, we extracted the number of publications by publication year corresponding to our extracted studies which can be seen in Figure 3.3. From this graph, we can clearly see an upward trend in publications in recent years, noting that the amount of publications for the year 2022 is only until the end of July, the date when the databases and search engines were queried. Furthermore, looking at the trend over the years, it is interesting to note that almost every peak is closely preceded by surveys related to classification in DSs which were highly influential according to their number of citations. Indeed, [107], [99], [129], and [101]

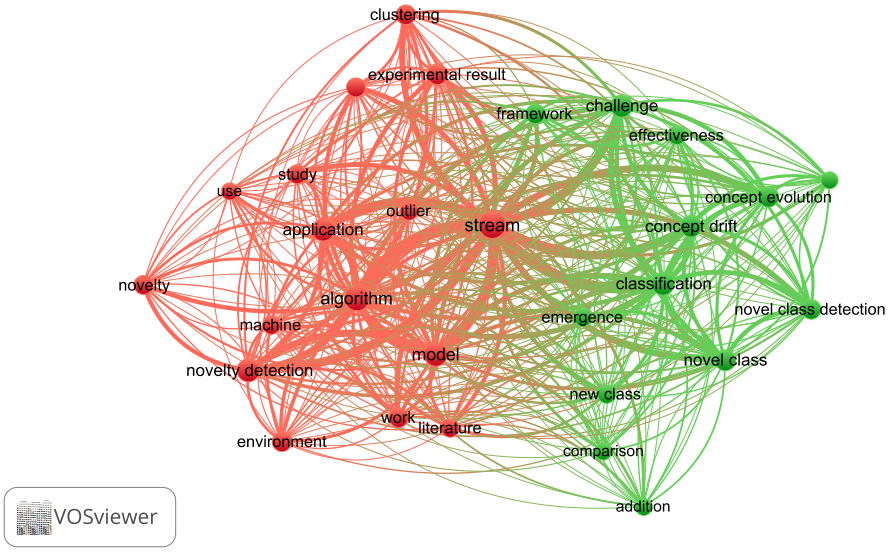



Figure 3.2: Co-occurrence of concepts in title and abstract

were respectively published in 2014, 2017, 2017, and 2019, and all figure within the most cited works between all publications analyzed in this work. On the other hand, the years that saw the most decline, mainly 2014, 2017, and 2021 were either the years where these surveys were published or not closely preceded by their publication. This highlights the importance that this type of survey has on the awareness of the domain and the amount of publications that are released every year.

### 3.3.3 RQ1.3: Who are the leading researchers in the field?

In addition to the yearly publication trends, we also wanted to note and explore who are the leading researchers in the field. As such, we ran a co-authorship analysis, where the relatedness of each author is determined by the number of co-authored publications. We limited our analysis to only authors with more than three publications and extracted the two biggest clusters. The first cluster extracted, which can be seen in Figure 3.4, shows the works that were mainly co-authored by Dr. Khan, which has by far the most co-authored publications in the domain of ND compared to other authors. We do also note, however, in this cluster, the works of Dr. Masud, Dr. Han, Dr. Gao, and Dr. Thuraisingham which also all have a significant number of publications. The second cluster we extracted can be seen in Figure 3.5 and regroup the works co-authored by Dr. Gama, which also has numerous

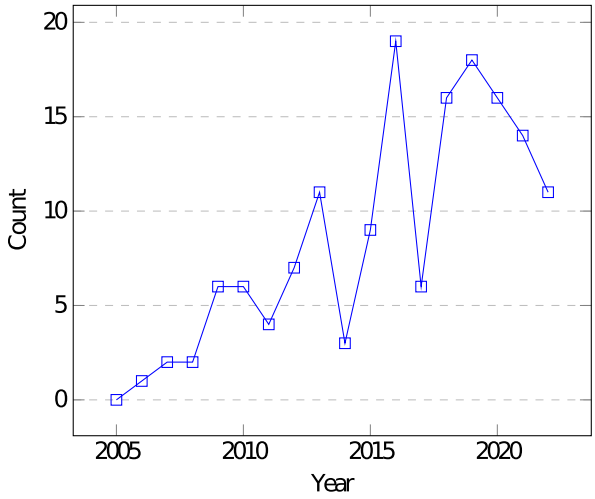



Figure 3.3: Number of publications per year

important publications in the field of ND, including some that laid the foundations of ND in DSs (e.g. [82]). Overall, we notice that ND is still a niche area of research as there are still very few researchers contributing to solutions and we believe that, given the practical importance of ND, it is a field that has the potential to expand a lot and be a promising area of research for new researchers.

### 3.3.4 RQ2.1: What are the challenges and requirements of novelty detection in data streams?

Compared to standard ML approaches, data mining in DSs presents numerous distinct challenges. In this section, we gather and summarize what challenges were encountered by researchers, and we will, in a subsequent section, discuss what was done to overcome these challenges. As expected, most of the challenges mentioned in the papers are related to the intrinsic characteristics of DSs, such as their continuous and evolutionary nature. We clustered all of the challenges extracted from every studied paper within the following categories:

1. **Infinite Nature** The fact that DSs can theoretically be infinite means that the storage of all previous samples and concepts encountered by the model would technically be impossible as it would require infinite storage/memory. As such, it is

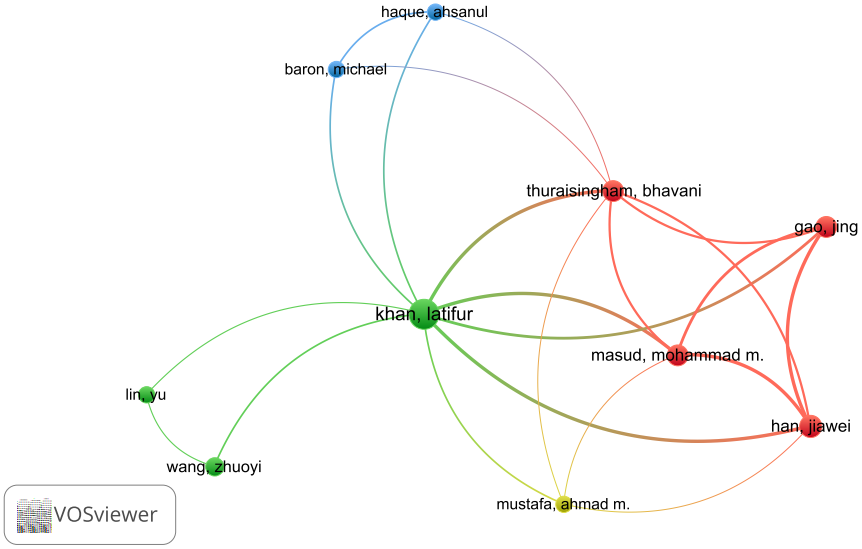



Figure 3.4: First cluster of co-authorship analysis

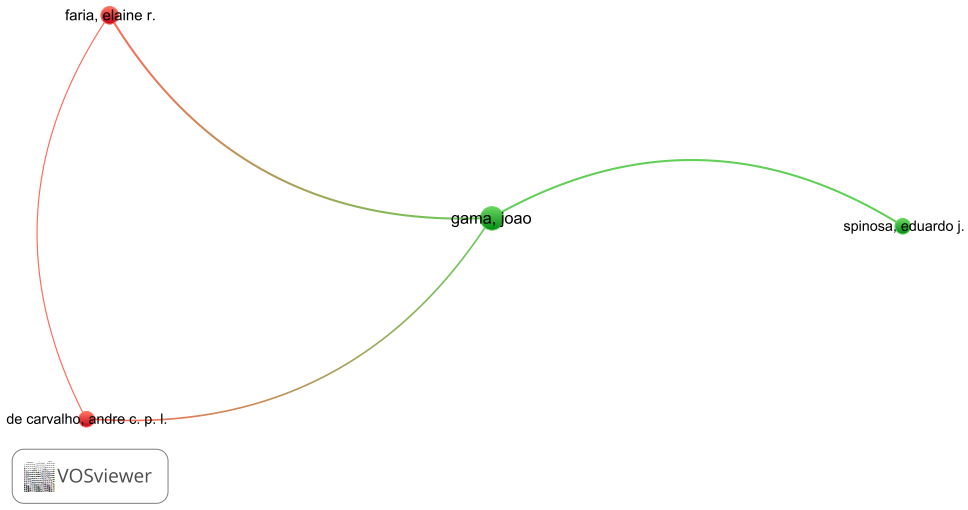



Figure 3.5: Second cluster of co-authorship analysis

paramount that approaches dealing with DSs implement different methods of dealing with a potentially infinite amount of data such as fixed-size chunks or forgetting mechanisms [113].

2. **Concept Drift** As described in Section 2.1, concept drift is a recurring problem with DSs that occurs when concepts that were previously known by the model change over time. These changes can be slow, often referred to as gradual/incremental drift or abrupt, also called sudden drift [130]. If not dealt with, samples belonging to a known concept could be classified as outliers, or as a new class. As such, known classes and their decision boundaries need to be updated throughout the DS to reflect the changes in distribution, and old, unused concepts need to be forgotten to not run into the aforementioned problem of needing an infinite amount of memory.

3. **Recurring Concepts** Recurring concepts or classes are described as classes that disappear from the DS for a certain period of time, and then reappear [148]. When reappearing, these samples can be mistakenly classified as a novel class as they might have been forgotten or present a concept drift since they were last seen. This challenge is commonly seen in real-world applications displaying cyclical behaviour, such as buying habits or electricity usage, unfortunately, as it is a complex and hard-to-deal-with problem, it is not commonly addressed in the literature.

4. **Novel Concepts** Not only do novel classes need to be discerned from the aforementioned concept drift and recurring concepts, but once they are classified as novel, they also need to be clustered between themselves, as multiple different novel classes could appear during the same time frame within the stream (e.g. two new network attacks occurring during the same time period). Moreover, if we want the model to be able to recognize these classes as they reappear in the DS in the future, the model needs to be updated incrementally and typically in an online fashion to reflect those newly discovered classes.

5. **Outliers and Noise** Outliers and noise commonly appear in any real-world data collected. In DSs, they become even harder to deal with, as they need to be differentiated from the aforementioned concept drift, novel concepts, and recurring concepts, which is a nontrivial task.

6. **Computational Cost** Being an online task, DS algorithms need to be computationally efficient to be able to run and effectively classify incoming samples without delay. Furthermore, if the models are updated incrementally with new classes, they can slowly become more and more computationally expensive which could prevent the model to classify incoming samples as they come through.

7. **Availability of True Labels** While the assumption of having access to the true labels of already known concepts is fair, the availability of the true labels for incoming samples in the DS is often an unrealistic scenario. Indeed, when receiving data in an online fashion, it is unlikely that the model will have access to the ground truth as it would require a human expert to label these samples continuously, a tedious and laborious task, which could be impossible depending on the speed at which new samples are received.

8. **Concept Scarcity** While samples belonging to a concept are often located close to each other, for example within a certain time window, there are cases where these samples can be distributed at various points throughout the DS, where it becomes challenging to recognize them as a novel concept rather than as outliers. This phenomenon can be especially prevalent in domains with a very large amount of samples and involving malicious actors, such as cybersecurity and financial fraud. In these examples, the large number of samples that can change over time, such as network traffic and transactions, respectively, can disperse malicious activity within the DS and therefore avoid detection.

9. **High-Dimensional Data** As will be discussed in the next section, ND algorithms have a myriad of applications in different domains, some of which involve high-dimensional spaces, such as textual [3, 44, 75] and visual [80, 124] data. In addition to the typical challenges encountered by ML models in these high-dimensional spaces, the other challenges applicable to ND that were mentioned prior, for instance, the infinite nature of the streams and computational cost of the algorithms, are amplified by this inherent characteristic of the data in these applications.

10. **Evaluation** As we will further investigate in Section 3.3.7, ND algorithms vary widely in their implementations and usage. This makes the evaluation of these algorithms and their comparison difficult, as they display vastly different properties, such as the ability to detect multiple classes, the availability of the true labels within the DS, and much more, which need to be taken into account to properly evaluate the algorithms' performance. Moreover, ND also presents unique aspects that are not present in batch learning and need to be considered to properly evaluate the algorithm's performance, such as the time to detect a novel class after it first appears and the adaptability of the model to concept drift.

### 3.3.5 RQ2.2: What are the different applications of novelty detection in data streams?

Most of the proposed approaches in the studied papers were presented as general approaches which can be applied to any type of tabular data. We noted, however, the presence of algorithms tailored to specific types of data in order to solve specific challenges related to those fields. As such, we clustered the applications and fields into four groups which are described as follows:

1. **Visual Data** Studies focusing on the fields of image and video recognition [86, 121, 124] are more plentiful than others. Indeed, as with traditional batch learning, dealing with video or images requires specifically made algorithms and processes which can handle high-dimensional data, an example of which would be Convolutional Neural Networks (CNNs). Streaming this type of data in an online fashion, as is the case with DSs, presents further challenges, as we mentioned that the proposed algorithms need to be computationally efficient. Moreover, this type of data commonly requires further and custom preprocessing before being able to be treated by the ND process. Some of the tasks related to this type of data include the detection of novelties in video streams [124], face recognition [80], and gesture classification [10, 11, 131].

2. **Textual Data** Similar to the aforementioned image and video classification, text classification tasks also need to deal with the curse of dimensionality, and heavy preprocessing such as embedding and cleanup needs to be performed prior to the usage of the data for ND. Most of the applications of ND regarding textual data revolved around the subject of topic classification. For example, by trying to find new topics that were not mentioned before in the classification of social media posts [2, 3, 25] or the domain of different product reviews [75].

3. **Signal Data** We noticed that methods dealing with signal data are typically aimed at predictive maintenance in industrial applications. These types of applications are a prime target for ND, as supervised data of normally running machines is usually readily accessible and the detected novelties can be considered as anomalies, therefore not necessitating any manual labeling. Other notable applications for this type of data also include activity recognition [166] and the recognition of unexpected data in robots' environments [165, 171, 246], which can also include visual data.

4. **Tabular Data** This last group regroups all other algorithms which, as we mentioned, do not specify or aim to solve a specific problem. While some propositions do aim

their research and experiments at specific domains, such as intrusion detection [191], these propositions are usually able to handle any type of tabular data, but might not be able to deal with the other types of data very well without significant changes being made to their architecture.

### 3.3.6 RQ3.1: How are proposed solutions dealing with the aforementioned challenges?

Following the list of challenges established in Section 3.3.4, we wanted to explore how they were typically handled and mitigated within the literature. As such, we extracted and summarized in this section, if applicable, the solutions commonly employed to deal with each identified challenge.

1. **Infinite Nature** A very prevalent solution to deal with the infinite nature of DSs is the use of a sliding window, also called chunks, of a predetermined number of samples when performing the online classification and ND, keeping only in memory the samples within that window (e.g.[70,147]). These chunks of data are used to train the model or update it, accordingly, and are then discarded. In addition, another commonly used technique within algorithms using clustering-based models is the concept of micro-clusters which allow condensing the information of each cluster with just a few pieces of information. These concepts were also defined with more details in Section 2.1.

2. **Concept Drift** The solutions to mitigate the impact of concept drift can usually be clustered within two main concepts: forgetting mechanisms and continuous updates. The former usually consists in systems that allow the classes or clusters that have not received new samples for a certain time to be discarded (e.g.[2,21]) or to be moved to a sleep memory (e.g.[56,70]) which can then be discarded when the memory is full. In the latter, the decision model needs to be updated ongoingly with the most recent samples of the known classes. For example, in clustering algorithms, the centroid of the clusters is usually updated with each new sample (e.g.[56]), while other algorithms will use different methods depending on their type, for example, Bouguelia et al. [30] use an extension of the growing neural gas algorithm to update the neurons of their neural network model.

3. **Recurring Concepts** This challenge is one of the lesser-known and handled challenges in ND, as it can also be conflicting with the concept drift problem. Indeed,

the forgetting mechanisms often implemented to deal with concept drift, can forget known concepts that are not showing in the stream for some time and only reappear after a while, which would make the model detect the class as a novelty and start the learning process from scratch. A possible solution for this issue is the use of a long-term memory rather than deleting the concepts, such as the one implemented in MINAS [56], where the concept can then be recovered if it reappears in the stream.

4. **Novel Concepts** To differ between different concepts within detected novelties, some algorithms implemented the idea of the unknown class, also known as a buffer (e.g. [1, 138, 148, 160]), which allows the algorithm to temporarily store the novelties until a certain amount and then separate each class by performing clustering or querying the user for the true labels for example. We will discuss it in more detail in the next section when discussing the architecture of the models.

5. **Outliers and Noise** Similarly to the solution presented in the previous challenge, the usage of a buffer can also mitigate the issue of outliers and noise, as samples will not be considered as a novel class when they are not close enough to multiple other samples. However, the number of samples required for a cluster to be considered as a novel class, as well as the maximum distance between them are all parameters that usually need to be tuned by the user.

6. **Computational Cost** As we will describe in the next section, to deal with the issue of computational cost, most ND algorithms use simpler ML models that are generally quick to train and classify, such as clustering (e.g. [56, 149]) and tree-based (e.g. [72, 181]) models, which are often combined within ensembles to improve their performance. Moreover, updating the models at a lower frequency is also a technique sometimes used to deal with the computational cost.

7. **Availability of True Labels** The usage of unsupervised algorithms for the ND task is a common way of avoiding the scarcity of the true labels in DSs, however, other solutions have also been explored as a middle ground between having full access to the true labels and not having any. An example of this is the use of active learning by Fahy et al. [33], where the algorithm selects a few representative samples which were detected as unknown, to be labeled by the user.

8. **Concept Scarcity** Inherently, if the samples of a novel concept are distributed very sparingly in a DS, it becomes extremely difficult to differentiate them from noise and outliers. A potential solution could be to keep in memory every sample for some time and increase this period if multiple samples are found to be close to one another.

However, to the best of our knowledge, there are no propositions that implement a solution directly to solve this issue at this time.

9. **High-Dimensional Data** Algorithms that are specifically made for domains with high-dimensional data will use suitable classifiers as their underlying models, such as Deep Neural Networks (DNNs) or CNNs (e.g. [86, 121, 166]).

10. **Evaluation** As we will discuss in more detail in Section 3.3.10, a few metrics and frameworks have been devised with the aim of properly evaluating ND algorithms. Notably, the percentage of misclassified novel class instances ($M_{\text{new}}$) and the percentage of misclassified known class instances ($F_{\text{new}}$) have been widely used in the field. However, it is important to note that these metrics only evaluate if a sample is correctly identified as novel or known and do not consider misclassification between novel classes. To address this limitation, some frameworks [55, 90] have recently been proposed to consider the misclassification of novel concepts between themselves as well as other DS properties such as the time to detect a novel class.

### 3.3.7 RQ3.2: What architecture do novelty detection algorithms implement?

To paint a better picture of the types of design used by the proposed algorithms, we propose the taxonomy shown in Figure 3.6, which is partially based on the work of Faria et al. [71] modified with our knowledge gained from performing the systematic review to be able to classify all the works that were analyzed.

As other works have mentioned [61, 71], most ND techniques follow a two-phase process which first includes an offline phase that would generally train a classification model with a traditional static dataset to identify "normal" classes, and then perform a second, online phase, which would be performed on a DS and usually used to identify novel classes and update the original classification model. However, as some of our analyzed algorithms might not include strictly speaking, an offline phase (e.g. [9, 11, 21, 30, 35, 189]), we decided to instead propose a taxonomy based on three other aspects of ND techniques which are universal to all analyzed papers: DSs' characteristics, classification, and ND. The first aspect relates to intrinsic features of the DSs that are assumed by the algorithm, while the classification part refers to the section of the algorithm used to classify known classes. On the other hand, the ND portion refers to how the novelty concepts are handled, for example, some ND techniques will use further ML techniques such as clustering in order to divide the detected novelties into concepts and update the classification model, while

others will only detect the novelties as outliers in the data. The proposed taxonomy can be seen in Figure 3.6, and the following sections will go into more detail about each aspect of the taxonomy and the different existing implementations.

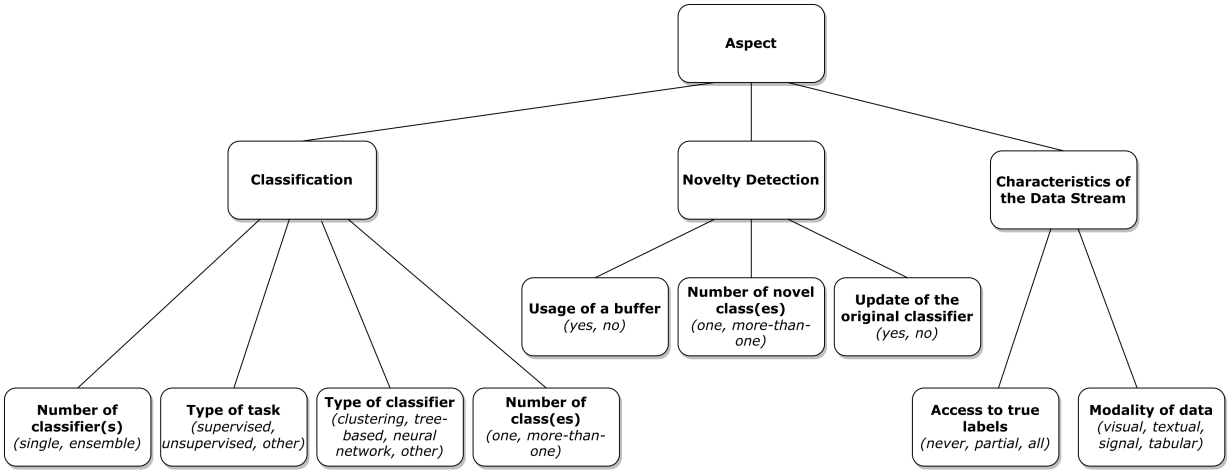



Figure 3.6: Proposed taxonomy

### Classification

As displayed in Tables 3.2 to 3.4, the analyzed papers display a wide range of different approaches for the classification aspect of their proposed algorithm, which we analyze in more detail in this section.

**Number of classifier(s)** For the classification task, the proposed algorithm might be using a **single** classifier. For example, in [30], Bouguelia et al. use a single neural network to classify incoming known classes into their respective labels.

In contrast, it is also common to see proposed algorithms use an **ensemble** rather than a single classifier in order to exploit some of their advantages such as their lower computational cost than other techniques and a good representation of the data space [180]. Some approaches (e.g. [149]) use a fixed-sized ensemble, predetermined by the user, to classify all known classes, while other proposals might use one (e.g. [75]) classifier for each known classes in a one-vs-all fashion. In other cases (e.g. [33]) the known classes are divided even further into multiple clusters and classifiers are trained on each of these clusters.

| Number of classifier(s) | Single | Ensemble | |
|---|---|---|---|
| | [2,3,8–11,17,19,21,22,24,29, 30,34–36,48,50,51,53,60,68, 68, 70, 72, 80, 85, 86, 88, 117, 120, 123, 124, 128, 131, 135, 136, 144, 155, 159, 160, 164–166, 169, 171, 172, 174, 176, 178, 181, 189, 192–194, 200–203, 209, 211–213, 215, 219–225, 227–231, 236, 237, 239, 242,243,246,247] | [1,12,25,33,38–40,47,49,52, 56,64,66,73,75,87,106,108, 112–114, 119, 121, 138, 139, 147–153, 156, 157, 163, 168, 179, 188, 218, 234, 235, 241, 244] | |
| Type of task | Supervised | Unsupervised | Other |
| | [8–12,17,25,33,35,36,47–53, 56,60,64,66,70,72,73,75,80, 85–88,106,108,112–114,119, 121, 123, 128, 131, 136, 138, 139, 147–153, 155–157, 159, 163, 166, 168, 169, 172, 178, 179, 181, 188, 192, 194, 203, 215, 218–224, 227, 229, 234–237,239,241,242,244,247] | [2,3,19,21,22, 24, 30, 34, 38–40,68,68,117, 120, 124, 135, 144, 165, 171, 174, 174, 176, 189, 193, 200–202, 209, 211–213, 225, 228, 230, 231, 243, 246] | [1,18,29,160,164] |

Table 3.2: Number and type of classifier(s) of all reviewed papers using the proposed taxonomy

| Clustering | Tree-based | Neural Network | Other |
|---|---|---|---|
| [1–3, 12, 19, 21, 22, 24, 29, 33, 40, 47, 48, 50, 52, 53, 56, 60, 66, 68, 68, 70, 80, 87, 88, 112–114, 117, 119, 120, 124, 135, 136, 138, 144, 147–151, 153, 155, 163, 169, 176, 189, 192, 200–202, 209, 211–213, 224, 225, 228, 230, 231, 241] | [38, 39, 49, 72, 73, 152, 157, 160, 181, 244] | [9, 17, 25, 30, 35, 51, 85, 86, 106, 108, 121, 128, 165, 166, 171, 178, 179, 194, 221, 223, 227, 229, 239, 242, 246, 247] | [8, 10, 11, 16, 34, 36, 64, 75, 123, 131, 139, 156, 159, 164, 168, 172, 174, 188, 193, 203, 215, 218–220, 222, 234–237, 243] |

Table 3.3: Classifier(s) type of all reviewed papers using the proposed taxonomy

| One | Multi-class | Multi-label |
|---|---|---|
| [34, 38, 39, 123, 124, 144, 155, 156, 164, 165, 171, 174, 193, 194, 200–202, 209, 211, 212, 215, 225, 231, 243, 246] | [1–3, 8–12, 17, 19, 21, 22, 24, 25, 29, 30, 33, 35, 36, 40, 49–53, 56, 60, 64, 66, 68, 68, 70, 72, 73, 75, 80, 85–88, 106, 108, 112–114, 117, 119, 121, 131, 135, 136, 138, 139, 147–153, 157, 159, 160, 163, 166, 168, 169, 172, 176, 178, 179, 181, 188, 189, 192, 203, 213, 218, 219, 221–224, 227–230, 234–237, 239, 241, 242, 244, 247] | [47, 48, 128, 220] |

Table 3.4: Number of known class(es) of all reviewed papers using the proposed taxonomy

**Type of task** Most proposals follow a **supervised** learning approach to train the initial classifier(s) on the original, known classes. For instance, in [85], Gao et al., present SACCOS, a semi-supervised learning algorithm, which trains an artificial neural network in a supervised fashion on the original labeled data. In other words, we classify in this category all algorithms that use the initial true labels to train a supervised classifier.

This is opposed to other classification tasks which we classify as **unsupervised**, where the original data true labels are not used, and unsupervised techniques, typically clustering (k-means, DBSCAN [67], etc.) or anomaly detection (Local Outlier Factor, etc.) are used. For example, in AIS-Clus, a novelty algorithm proposed by Abid et al. [2], they used the DBSCAN algorithm to perform the initial clustering of the data. These approaches have the advantage of not necessitating any prior labeling of the data.

**Type of classifier** One of the most prominent type of algorithm used for the classification task is **clustering**, which includes algorithms such as k-means, k-NN, and DBSCAN. However, rather than using the raw training data directly, which can become problematic due to the potentially infinite nature of DSs, a common strategy is the use of micro-clusters [4], as described in Section 2.1. For example, multiple algorithms (e.g. [70, 88]) utilize the CluStream framework [4] which implements the concept of micro-clusters. This solution consists in condensing the data into five components: the sum of squares of the data values, the sum of the data values, the sum of squares of the timestamps, the sum of the timestamps, and the number of data points. In [70], the authors present MINAS, which uses the CluStream framework to build representative micro-clusters from the training data, which are then treated with another clustering algorithm, in this case, k-means in order to find $k$ clusters for each class.

Another noteworthy area of research, which has slowly garnered increased attention in recent years for its application in ND is the use of fuzzy concepts. Indeed, as fuzzy set theory can allow to make the learning process more flexible [49], it has been demonstrated that its use in ND can allow the model to be more adjustable when dealing with the changes in the DS compared to a non-fuzzy variant in certain cases [50]. Fuzzy concepts have been applied for ND in both more traditional classification models such as decision trees and neural networks [49, 227], as well as clustering approaches [50, 53, 120, 135, 189, 236].

A less commonly used type of classifier is **tree-based** classifiers. An example can be seen in [72], where Farid and Rahman train a decision tree on the initial data and calculate the distribution percentage of the training samples in each tree node, which will then be used to discover novelty classes if the distribution changes over time.

**Neural-network-based** classifiers are also a popular choice, especially in high-dimensional

application domains. For example, in [121], Jameel et al. present an adaptive CNN ensemble classifier able to classify high-dimensional DSs such as images and handle class evolution via dynamic neuron addition and weights updates.

**Number of class(es)** Some algorithms, such as doOCSVM [156] cannot perform the classification of multiple distinct classes on the original training data. Rather, these types of approaches, such as in this example, are more akin to traditional anomaly detection and typically consider the original training data as **one class** which is labeled as normal, or benign.

On the other hand, **multi-class** approaches, which are the more common type, can classify multiple different classes learned from the training data.

A rarer case of proposed methods is those that support **multi-label** classification. In [47], Junior et al. identify the lack of research in this domain and propose MINAS-BR, an extension of the MINAS [56] algorithm that supports multi-label DSs, by applying a binary relevance transformation to the problem.

### Novelty Detection

Tables 3.5 and 3.6 displays the different characteristics for the ND task of the surveyed papers, which include the following aspects.

**Usage of buffer** Some ND algorithms, such as MINAS [56] first classify potential novelty samples as unknowns, by storing them in a temporary **buffer** if the sample lies outside the model's current decision boundary. This method allows the removal of outliers that might not necessarily represent an emerging class. In MINAS, a clustering algorithm is performed on the buffer every time a new sample is added and if a valid cluster is found, it is decided whether it is an extension of a known concept or a new concept depending on its distance with known clusters.

Other types of approaches use **no buffer**, for example, in [35], an ensemble of one-class OC-SVM classifiers will evaluate each sample independently as novel or known, depending on a novelty threshold fixed by the user.

**Update of the original classifier** In AnyNovel [1], once a novel concept has been detected and confirmed as such, a new cluster of the data and its subsequent sub-clusters

| Usage of buffer | Yes | No |
|---|---|---|
| | [1–3, 12, 21, 22, 25, 29, 33, 36, 40, 47–53, 56, 60, 66, 68, 68, 70, 73, 85–88, 106, 112–114, 117, 119, 121, 128, 138, 139, 147–153, 157, 160, 163, 168, 169, 179, 188, 192, 200–203, 209, 212, 213, 221–224, 227, 230, 231, 234–236, 239, 241, 242, 244] | [8–11, 17–19, 24, 30, 34, 35, 38, 39, 64, 72, 75, 80, 108, 120, 123, 124, 131, 135, 136, 144, 155, 156, 159, 164–166, 171, 172, 174, 176, 178, 181, 189, 193, 194, 211, 215, 218–220, 225, 228, 229, 237, 243, 246, 247] |
| Update of the original classifier | Yes | No |
| | [1–3, 12, 21, 22, 24, 25, 29, 30, 33–36, 40, 47–53, 56, 60, 64, 66, 68, 68, 70, 72, 75, 80, 85–88, 106, 112–114, 119–121, 123, 128, 131, 135, 136, 138, 139, 147–153, 157, 159, 160, 163, 165, 168, 169, 171, 172, 176, 179, 188, 189, 192, 200, 202, 203, 211–213, 218–221, 223, 224, 227–231, 234–237, 239, 241, 242, 244, 246] | [8–11, 17–19, 38, 39, 73, 108, 117, 124, 144, 155, 156, 164, 166, 174, 178, 181, 193, 194, 201, 209, 215, 222, 225, 243, 247] |

Table 3.5: Novelty detection aspect of all reviewed papers using the proposed taxonomy

| One | More-than-one |
|---|---|
| [8–11, 17–19, 34, 35, 38, 39, 73, 75, 108, 117, 123, 124, 131, 144, 155, 156, 160, 164–166, 168, 169, 171, 174, 178, 179, 181, 193, 194, 200–202, 209, 211, 212, 215, 224, 225, 231, 243, 244, 246, 247] | [1–3, 12, 21, 22, 24, 25, 29, 30, 33, 36, 40, 47–53, 56, 60, 64, 66, 68, 68, 70, 72, 80, 85–88, 106, 112–114, 119–121, 128, 135, 136, 138, 139, 147–153, 157, 159, 163, 172, 176, 188, 189, 192, 203, 213, 218–223, 227–230, 234–237, 239, 241, 242] |

Table 3.6: Number of novel class(es) of all reviewed papers using the proposed taxonomy

are generated and the baseline model is **updated** with the information of this novel concept so that it can be detected in the future.

On the other hand, in [11], the proposed algorithm by Al-Behadili et al. can detect novel classes, but the original classification model will **not be updated** with the new classes and therefore will be unable to detect recurring novel concepts in the DS.

**Number of novel class(es)** Mr-NoDeS [38] presents an example of an algorithm able to detect only **one class** of novelty, as while it allows for detecting numerous anomaly patterns within the DS, it cannot distinguish between the different classes or clusters of anomalies, the samples are either anomalous or benign.

In contrast, MINAS [56] can separate and cluster different types of novelties together since it performs clustering on the detected outliers detected. It is, therefore, able to detect **more-than-one** novel classes.

### Characteristics of the Data Stream

| Never | Partial | All |
|---|---|---|
| [1–3, 8–11, 17–19, 21, 22, 24, 25, 30, 34, 36, 38–40, 47, 48, 51–53,56,64,66,68,68,70,72, 73,75,80,87,88,106,108,117, 120, 121, 123, 124, 128, 131, 136, 139, 144, 150, 151, 155–157, 160, 164–166, 168, 171, 172, 174, 176, 178, 181, 189, 192–194, 200–202, 209, 212, 213, 215, 219, 224, 225, 227, 228, 230, 231, 237, 239, 243, 246, 247] | [12, 29, 33, 35, 49, 50, 56, 85, 86, 112–114, 119, 135, 138, 147–149, 152, 153, 159, 163, 169, 203, 221–223, 229, 241, 242, 244] | [60, 179, 188, 218, 220, 234–236] |

Table 3.7: True label availability of all reviewed papers using the proposed taxonomy

Finally, Tables 3.7 and 3.8 show how the analyzed papers handle the process of accessing the ground truth labels within the stream of data, which we discuss further in the following section.

| Visual | Textual | Signal | Tabular |
|---|---|---|---|
| [10, 11, 17, 80, 86, 121, 124, 131, 166, 192, 218, 219, 221, 222, 227–229, 239, 242, 244, 247] | [2,3,17,25,75, 86, 221, 222, 225,242] | [34, 35, 135, 144, 165, 166, 171, 172, 176, 178, 179, 193, 194, 215, 224, 243,246] | [1, 8, 9, 12, 18, 19, 21, 22, 24, 29, 30, 33, 36, 38–40, 47–53, 56, 60, 64, 66, 68, 68, 70, 72, 73, 85, 87, 88, 106, 108, 112–114, 117, 119, 120, 123, 128, 136, 138, 139, 147–153, 155–157, 159, 160, 163, 164, 168, 169, 174, 181, 188, 189, 200–203, 209, 211–213, 218–220, 223, 227, 230, 231, 234–237, 239,241] |

Table 3.8: Modality of data of all reviewed papers using the proposed taxonomy

**Access to true labels** Most approaches do not expect the availability of **any** true labels during the DS, as is the case with ActMiner [153], which presents a typical implementation of an unsupervised ND task, where it determines if an incoming sample is detected as a potential novel class or outlier simply if the sample lays outside the decision boundaries previously learned by the ensemble of models and therefore does not need any external feedback.

Others, such as SACCOS [85], require only a **partial** amount of true labels. In this specific instance, the classifier is used to label the samples when it has high confidence. In cases where the confidence is low, a random sample from the cluster will be selected and a human annotator will be asked to provide the ground truth label.

On the contrary, in their Evolving Micro-Clusters algorithm, Din and Shao [60] consider a fully supervised DS context, where **all** true class labels are available after some time. However, it is to be noted that this scenario is less realistic, as it would be unlikely for a human expert to be labeling all samples from an online DS.

**Modality of data** As presented in Section 3.3.5, certain algorithms analyzed were purposefully designed to answer specific modalities of data. For example, in [228], Wang et al. present an ND algorithm aimed at high-dimensional data, and demonstrate its usage on images specifically. Another example of **visual** data, can be seen in VENUS [124], where the authors use video DSs to detect novel events.

In AIS-Clus [2], the authors explicitly target **textual** DSs, as their solution includes specific preprocessing for textual data as well as specific clustering metrics in order to compute the distance between different texts.

A number of other approaches address **signal** data, such as DSs coming from sensor or audio sources. For instance, [172] proposes a framework for ND aimed at fault detection and classification in industrial applications.

Lastly, the large part of algorithms do not specifically aim their solution at a specific type of data modality, but rather work more broadly on **tabular** data as a whole.

It is worth noting that contrary to the other aspects of the taxonomy previously presented, an algorithm can be classified in multiple categories within this characteristic, as there are some algorithms designed to specifically work on multiple modalities of data. For example, ESACOD [222] specifically targets both image and textual data.

### 3.3.8 RQ3.3: How are the most influential algorithms designed and how have they evolved throughout the years?

In order to have more insights into the previous and current trends in ND algorithm design and observe how they have evolved throughout the years, we present in this section, more details on the architecture and design of the most influential algorithms from our range of analyzed algorithms. We classify an algorithm as influential depending on various factors, including the total number of citations, the number of inter-references between all analyzed papers, the number of extensions, and their distinctiveness compared to other propositions. The following algorithms are presented in chronological order of their publication.

#### OLINDDA

Published in 2007 by Spinosa et al., OLINDDA [201] is one of the earliest publications we analyzed able to handle both concept drift and concept evolution within a DS. It presents an architecture that will be emulated by other, more recent algorithms that we will present afterward. Its approach is particular since both the classification and ND tasks are done in a completely unsupervised manner, and the algorithm is based on a one-class approach, as it is not able to distinguish between multiple different normal classes or novel ones. This is in contrast to other more recent algorithms, which will tend to adopt a supervised approach for the classification task and be able to classify multiple classes. In Figure 3.7, we present a top-level overview of the algorithm's design using our previously defined taxonomy.

At first, OLINDDA trains an unsupervised k-means classifier on what will be the "normal" data, this classifier will then be used to determine the decision boundary, by computing the farthest distance between the samples and their corresponding cluster centroids. When performing the classification of new samples from the DS, if a sample is determined to be outside this decision boundary, it will then be classified as unknown and added to a temporary buffer.

In order to perform the ND task, this buffer containing unknown samples is periodically checked for possible new clusters using a k-means algorithm. This technique which can be seen in other algorithms as well, allows the algorithm to distinguish between single outliers/noise and concepts, which would contain multiple samples. Once clusters are detected within this buffer, they can be either considered valid or invalid. This is determined by whether or not the candidate cluster's mean distance between its samples and centroid is higher than the average of those mean distances of the known clusters. If it is higher, the candidate cluster is considered valid as it is far enough away from known concepts and can be either considered as concept drift or as a novel concept.

To distinguish between the two, OLINDDA calculates $d_{max}$, the maximum distance between the global centroid of the model and the clusters' centroid. If the distance between the new cluster and the global centroid of the model is lower than $d_{max}$, it is considered a concept drift, otherwise a novel concept. OLINDDA does not offer to update the model with the novel concepts detected, which means if the same novel concept is detected in the future, it will still be detected as a novel class. However, it does update its model to follow concept drift, when the number of samples in those clusters is higher than a predefined user parameter $k$, the initial model is retrained.

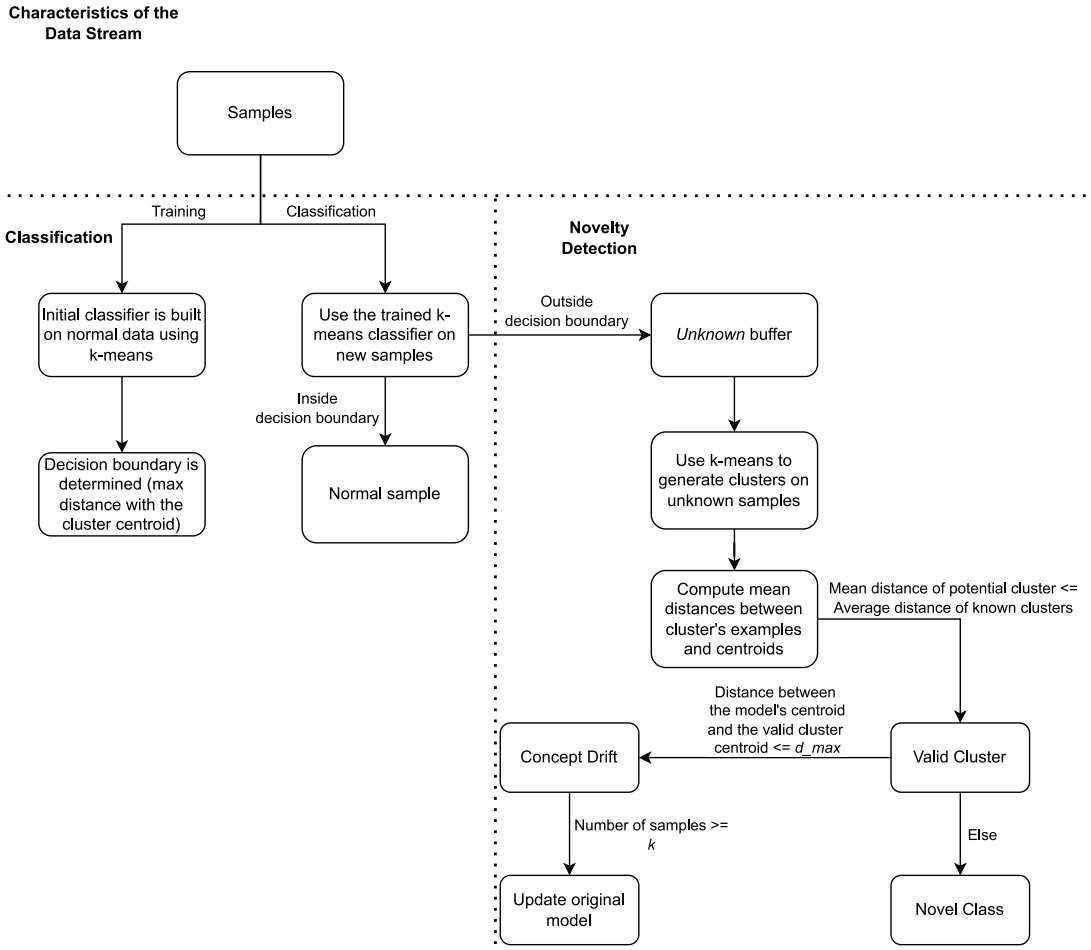



Figure 3.7: Top level overview of the OLINDDA algorithm

### ECSMiner

Being the most cited of all papers we analyzed, ECSMiner [147] is unavoidable when mentioning influential algorithms for ND. Using our previously defined taxonomy, we separate between the general characteristics of the DS as well as the classification and ND tasks and design a top-level overview of how the algorithm operates which can be seen in Figure 3.8. The proposed algorithm first separates the data into chunks of size $S$, these chunks will then be used to either train the classifier or be classified. Contrary to the previously described OLINDDA, in its training phase, ECSMiner uses an ensemble of $M$ base classifiers trained in a supervised fashion on the initial $M$ labeled data chunks, in other words, one classifier is trained on each of the initially labeled chunks. The authors apply their proposed technique with two different base classifiers, namely decision tree and k-NN. When using k-NN, the algorithm only stores a summary of the clusters, similar to the aforementioned micro-clusters technique, to mitigate the storage size and possibly infinite length of DSs. Once the initial ensemble is trained, incoming samples are then stored in a temporary buffer (unlabeled) and once they are labeled by an outside source, these samples are then added to a labeled buffer if they are within the decision boundary of the classifier. This is one of the biggest constraints of ECSMiner, as it assumes that labels will be available after a certain time. Finally, a new classifier is trained periodically, when the number of samples in the labeled buffer reaches the chunk size ($S$), in order to keep the ensemble up to date and adapt to concept drift.

If the classified samples are determined to be outside the decision boundary of the classifier, which the authors classify as F$outlier$, they are stored in an outlier buffer which will be periodically checked for instances that can be removed if they reached a certain age, are now no longer outside the decision boundary after an ensemble update or are now part of an existing class. Simultaneously, but at a lower frequency, once the F$outlier$ buffer reaches a size of $q$ and the ND process was performed at least $q$ time steps away, the ND process is performed to analyze potential novel classes. To evaluate the presence of a novel class, the authors first define $\lambda_{c,q}(x)$, the set of the $q$ nearest neighbors of sample $x$ within class $c$. Then, the mean distance between any point $x$ and its $q$ nearest neighbors can be computed as shown in Equation (3.1). Finally, the presence of a novel class is determined by using the $q$-neighborhood silhouette coefficient ($q$-NSC) which is described in Equation (3.2). This measure estimates the level of cohesion between the samples, which varies from -1 to +1, with a higher level of cohesion being a positive value. When more than $q$ samples have a positive value of $q$-NSC, these samples are marked as novel and a new classifier will be trained with those instances. It is to be noted that ECSMiner allows for concurrent different classes to be detected, as those different classes will be learned

once a new classifier is trained on the novel samples. However, although ECSMiner can perform multi-class classification and incorporate novel classes into its model, which was not possible with OLINDDA, it does not distinguish between concept drift and concept evolution as both will be identified as a novelty.

$$\bar{D}_{c,q}(x) = \frac{1}{q} \sum_{x_i \in \lambda_{c,q}(x)} D(x, x_i) \tag{3.1}$$

$$\text{q-NSC(x)} = \frac{\bar{D}_{c_{min},q}(x) - \bar{D}_{c_{out},q}(x)}{max(\bar{D}_{c_{min},q}(x), \bar{D}_{c_{out},q}(x))} \tag{3.2}$$

where $\bar{D}_{c_{min},q}(x)$ represents the minimum between all $\bar{D}_{c,q}(x)$, with $c$ being a known class, and $\bar{D}_{c_{out},q}(x)$ represents the mean distance between $x$ and the set of $q$ nearest neighbors within the F*outlier* samples.

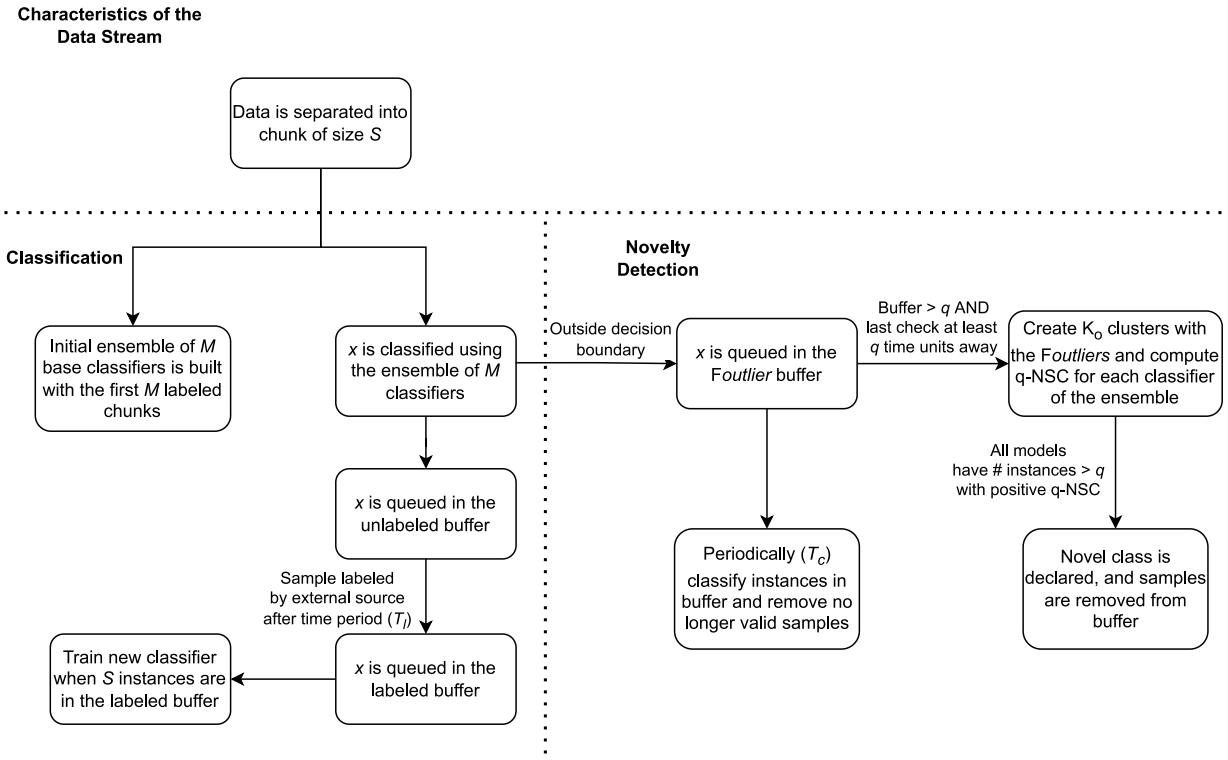



Figure 3.8: Top level overview of the ECSMiner algorithm

#### MINAS

MINAS [56] follows a similar approach as the one previously described in ECSMiner, as it also trains a classifier in a supervised fashion on the labeled data and then uses this classifier to detect if the samples are outside its decision boundary and classify them as outliers which will then be filtered into potential novel concepts. However, it differs in two important ways, as it allows distinguishing between concept drift and novel concepts, and does not require the samples to be labeled by an external source. A general top-level overview of the architecture of the algorithm can be seen in Figure 3.9.

The algorithm first trains on the normal, labeled data, by splitting each defined class into a subset and by training a clustering algorithm on each of these subsets to create $k$ micro-clusters for each class, the decision boundary of each class will be defined by the union of those micro-clusters. Once a new sample from the DS is received, the Euclidean distance between itself and the closest micro-cluster is computed and if this distance is lower than the micro-cluster's radius, it is considered part of its class. Otherwise, the label is considered a potential novelty and added to a buffer like previous approaches. Once a minimum number of samples determined by the user is reached in the buffer, a clustering algorithm is applied to the buffer and the clusters found will be classified as valid or invalid. To do so, the silhouette coefficient is computed and if it is positive and the cluster contains a minimal number of samples, the cluster is considered valid. At this point, ECSMiner would have considered this cluster as a novel concept, oppositely, to distinguish between a novel concept and concept drift, MINAS computes the distance between the cluster's centroid and the closest micro-cluster and if this distance is lower than a threshold, it will be considered as a concept drift of that particular micro-cluster, if not, it is considered a novel concept. Finally, the model is updated with this new information, whether it is a concept drift or a novel concept.

Note that throughout the execution of the algorithm, no external feedback (labeling) has been needed from the user, however, Faria et al. also propose an extension named MINAS-AL, which incorporates external feedback in order to improve the algorithm's performance. It does so by selecting the centroids of the micro-clusters detected in a given interval and asking an external source to label them. Once labeled, if available, these samples are then used to update the micro-clusters.

### 3.3.9 RQ4.1: In what context have these approaches been tested?

The reviewed papers used a mix of real-world and synthetic datasets, with an important number of them using KDDCup99 [204], Forest Covertype [28], and other datasets from

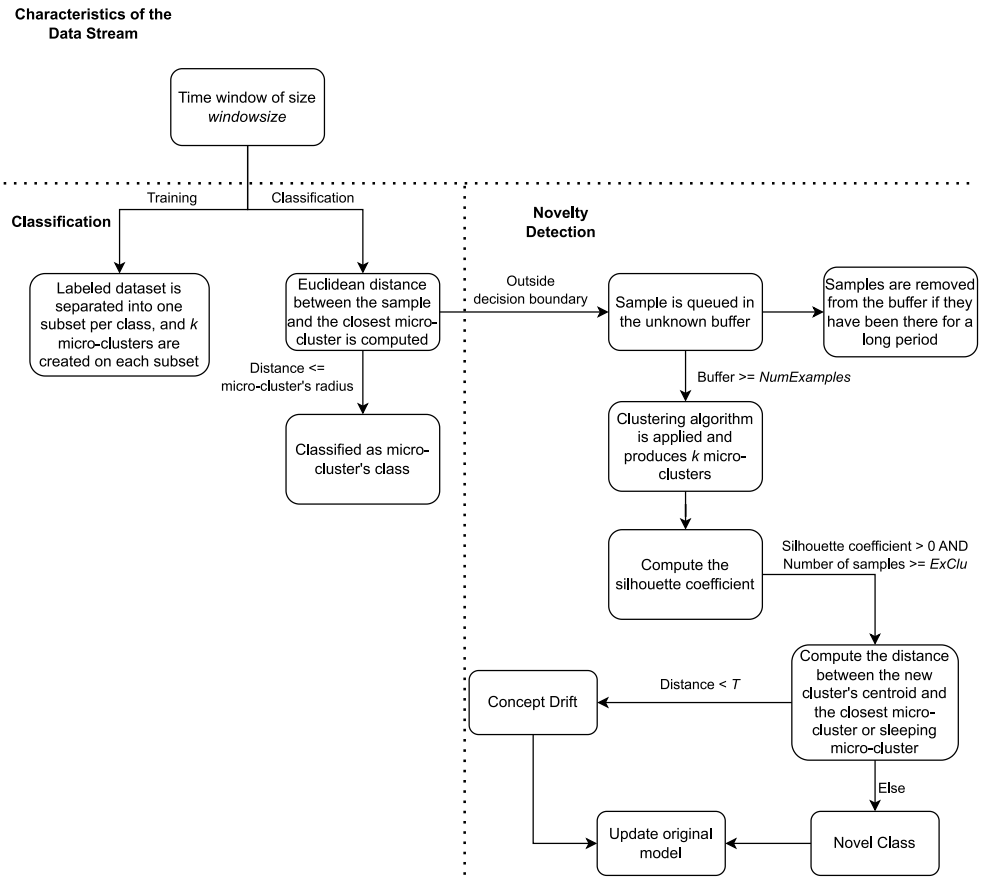



Figure 3.9: Top level overview of the MINAS algorithm

the UCI Machine Learning Repository [63]. A more comprehensive list of the most used datasets can be seen below. A notable trend we noticed while reviewing all of the selected papers was that while most propositions do commonly select some of the same datasets when performing their evaluation, there is no set standard or benchmark in the field of ND, which makes it very difficult to compare between different algorithms. Having a set of datasets that exhibit specific characteristics and have defined partitions with known performance on baseline algorithms could be a very useful future contribution to performance evaluation. Moreover, ND introduces other parameters such as the distribution of the novel classes in the stream, the timeframe of the arrival of samples, and the availability of true labels, which all need to be decided by the researcher. Consequently, as we will highlight in Section 3.4, we believe there is a need for more research and the establishment of an agreed-upon evaluation framework for the domain of ND in DSs.

- **Real-world**: KDDCup99 [204], Forest Covertype [28], MNIST [59], Shuttle [63], Keystroke [126], PAMAP2 [173], Iris [77], Airlines [118], Electricity [115], Poker Hand [37]
- **Synthetic**: Various synthetic DSs, often generated using the MOA framework [26].

#### 3.3.10 RQ4.2: What performance evaluation methods have been used to evaluate these strategies?

In this section, we define the most recurring performance metrics that are used to evaluate the performance of the analyzed algorithms. As novelty classes are often scarce within the DS compared to other, known classes, evaluating the performance of the models for ND suffers much of the same problems as traditional batch learning in imbalanced domains, such as metrics like accuracy usually not offering a good assessment of the performance of the model [96]. As such, researchers often turn towards metrics that are often used when the data distribution is imbalanced, such as Precision, Recall, $F_\beta$-measure, and others that have also been specifically defined for ND. An overview of these metrics can be seen in Table 3.9, where TP, FP, TN, $N$, $N_c$, and FE are defined for ND as follows [71, 147]:

- TP: Number of detected novelties correctly classified;
- FP: Number of known class samples wrongly classified as novelties;
- FN: Number of novelties wrongly classified as known;

- TN: Number of known class samples correctly classified;
- $N$: Total number of samples;
- $N_c$: Number of novel samples in the stream; and
- FE: Existing class instances misclassified (other than FP)

| **Metric** | **Equation** |
|---|---|
| Precision | $\frac{\text{TP}}{\text{TP+FP}}$ |
| Recall | $\frac{\text{TP}}{\text{TP+FN}}$ |
| $F_\beta$-measure | $(1+\beta^2)\cdot\frac{\text{Precision}\cdot\text{Recall}}{(\beta^2\cdot\text{Precision})+\text{Recall}}$ |
| $M_{new}$ | $\frac{\text{FN}\cdot 100}{N_c}$ |
| $F_{new}$ | $\frac{\text{FP}\cdot 100}{N-N_c}$ |
| $Err$ | $\frac{(\text{FP+FN+FE})\cdot 100}{N}$ |
| TPR | $\frac{\text{TP}}{\text{TP+FN}}$ |
| FPR | $\frac{\text{FP}}{\text{TN+FP}}$ |

Table 3.9: Commonly used performance metrics for novelty detection in data streams

Precision, recall and the $F_\beta$-measure are all metrics that were already defined for the conventional batch scenario and that were simply adapted to the ND case by using the novelties as the "Positive" class, as stated above for TP and FP. The Receiver Operating Characteristic (ROC) AUC is another common way of measuring performance in imbalanced domains, which was also adapted for ND, and consists of plotting the TPR against the FPR (cf. Table 3.9) for different decision thresholds and then computing its area under the curve [32]. On the other hand, $M_{\text{new}}$, $F_{\text{new}}$, and $Err$ which are respectively defined as the percentage of novel class instances misclassified as known, the percentage of known class samples misclassified as novelties, and the total misclassification error percentage [147], are all metrics that have been specifically defined for ND.

Although they have previously been used in scenarios where there are multiple classes of novelties, considering that all of the metrics presented only consider a single class of novelties (binary classification), they can not correctly assess the performance of models

in such cases, as simply classifying a new sample as a novelty, does not indicate that the algorithm properly classified the sample in the correct novelty class. Moreover, as ND algorithms are often implemented in an unsupervised fashion, they identify novel classes as NPs rather than their class label, where one novel class can be represented by multiple NPs. Consequently, properly evaluating these algorithms becomes a difficult task, as the NPs' labels do not match with the class labels, and the number of NPs identified also need to be considered to avoid a very high number of independent NPs.

Faria et al. [55] proposed a solution to the aforementioned issue, by adding as many columns as there are NPs in the confusion matrix and matching each with a true class, as well as a column for the samples that are temporarily added to the buffer, and then plotting a metric adapted to multiclass classification accompanied by a metric measuring the rate of unknown samples (samples in the buffer). More recently, we also proposed a framework [90] to streamline the evaluation of ND algorithms, which include the use of the AMI metric to evaluate multi-class scenarios, a novel metric named TTD that evaluates the number of samples of a given novel class needed to detect it as an NP, and an empirical evaluation of the impact of DSs' characteristics on the algorithms' performance. Nonetheless, we believe that there is still room for research in this sector, in order to incorporate other concepts of ND, such as the time to access true labels, the adaptation of the model to concept drift, recurring classes, impact of the density of novel samples, and so on. These aspects will be explored further in Chapter 5.

## 3.4 Open Challenges and Research Opportunities

Considering all of the information we gathered throughout this study, we summarize in this section what we believe are potential future works and open challenges that necessitate further research in the domain of ND in DSs. While we summarize some of the challenges that were also mentioned in previous works [61, 71, 183], we also identified numerous new points while extracting the data from all studied articles to answer our research questions which led to the following insights. Specifically, we noticed the lack of a constant definition for ND, the absence of information regarding the data setup, and the lack of proper taxonomy, which we explain in more detail below, resulting in some difficulties in analyzing and comparing the papers surveyed. Moreover, we also identified a specific challenge in Section 3.3.6, concept scarcity, which we have not yet seen mentioned in other papers.

### 3.4.1 Definition of Novelty Detection

As mentioned in Section 3.3.1, the term ND has evolved throughout the years and has been often associated or interchangeably used with other terms such as anomaly detection and OSR. However, as discussed, these terms can have different meanings depending on the author (e.g. [183]), which can lead to confusion. While in this study we adopted and formalized a specific definition for ND (cf. Section 3.3.1) which is also commonly used nowadays, the problem of clearly defining ND was evident when filtering and analyzing the different papers throughout this survey. Indeed, many papers might use the term ND to indicate other specific types of anomaly detection which can lead to confusion and difficulties when researching existing works. As such, it outlines the need for the adoption of a clear and precise formulation of ND throughout the field.

### 3.4.2 Experimental Framework

As it was previously mentioned in previous surveys on this topic [61,71], there is still a need for a standardized, experimental framework for the evaluation of ND algorithms. Indeed, while the impact of DSs' characteristics on the performance has been demonstrated [90], there is still no standard methodology specifying things such as the number of samples and known class(es) accessible for training, the order and number of samples available per time frame for the ND phase, and when true labels are available. This makes it extremely hard to compare different ND algorithms between themselves, as each author makes widely different assumptions that will impact the concept drift, concept evolution, and ultimately the performance of the algorithm depending on the data used.

Moreover, although some performance metrics have been defined for the ND case, they have not been used extensively nor do they assess all of the elements of ND algorithms, such as the adaptability of the model to concept drift and the period necessary to update the model with new classes. Besides, as stated in Section 3.3.10, there is even less development in the definition and extension of those metrics for the multi-class scenario, where multiple classes of novelties show up in the DS. Thus, there is a need for more research into the efficacy of the currently used metrics and the definition of novel consistent metrics that correctly assess important components of the ND domain such as the time dimension and concept drift resilience in both the binary and multi-class scenarios.

Lastly, as observed in Section 3.3.9, most algorithms rely on the same standard datasets from the UCI Machine Learning Repository. These datasets are often not true DSs but rather batch learning datasets that have been converted into an online setting, and several

of them are quite dated at the time of writing. We believe that the domain and research community would benefit from the use of newer, more extensive, and updated datasets that represent real DSs, especially in domains such as intrusion detection where the goal is to detect new, not necessarily known, attacks in an online fashion. An analysis of this problem and the challenges associated with it has also been discussed by Souza et al. in their study on benchmarking stream learning algorithms in real-world scenarios [198].

### 3.4.3 Concept Drift and Novel Classes

As we discussed in Section 3.3.6, the challenge of novel concept(s) distributed scarcely throughout the DS, or concept scarcity, is a problem that is very difficult to deal with as differentiating them from noise or outliers is a complex task. As far as we are aware, there are no studies that comprehensively looked at this issue and solutions to solve it, which would be needed.

In addition, while many techniques have been explored for dealing with concept drift as well as recurring concepts, there have not been many that specifically looked at concept drift that occurs very suddenly. This is a specific type of concept drift that occurs in many real-life scenarios and is very difficult to deal with.

### 3.4.4 Domains

Certain fields of study still lack research in the ND domain, more specifically high-dimensional data such as images, text, videos, and multi-label data. These fields have multiple real-world situations and are underrepresented in their amount of publications in ND. Furthermore, since most ND algorithms are based on clustering, they can not necessarily be easily extended to these areas.

### 3.4.5 Taxonomy

Finally, we believe that there is a need in the ND research community for a better representation and the usage of a proper taxonomy in the description of the proposed algorithms. As it is a domain in which the architecture of the proposed frameworks can differ widely from one proposition to another, it is necessary to properly define certain parts, such as the training and online phases, the detection of one or multiple novel classes, and the assumption of the availability of true labels during the ND or not. These specifications

are ambiguous in many research papers, which again, makes it more difficult to compare multiple algorithms. Adopting a taxonomy and reporting these critical components of the solutions published is an important aspect to be followed.

## 3.5 Summary

The work presented in this chapter provides an overview of the domain of ND for DSs and its advancement in recent years, including its challenges, proposals, and main contributors. We examined more than 150 proposals using our proposed updated taxonomy. All works reviewed were analyzed and examined in detail and the most influential solutions were further described in terms of their architecture.

This study revealed several insights on the domain of ND, starting with the definition of ND which is not defined identically throughout different works, and for which we outlined and formalized a definition in Section 3.3.1. Subsequently, we analyzed the number of publications and leading researchers in the field and noted that while the number of works generally followed an increasing trend throughout recent years, the field of ND still has a small number of main authors and is still fairly niche. We analyzed some of the common challenges that ND algorithms face when dealing with the task and their frequent solutions, and extracted some of the same challenges that were identified in other recent surveys [61, 71, 183]. We also identified a novel challenge not previously highlighted, namely concept scarcity, which arises when samples from a novel concept are dispersed scarcely throughout the DS rather than concentrated in clusters.

In Section 3.3.7, we introduced an updated taxonomy that categorizes all analyzed papers based on three aspects of ND, namely the DS characteristics, the classification task, and the ND task. The categorization of the papers within our proposed taxonomy also revealed some insights into different trends in the field of ND. Mainly, we noticed that clustering techniques are still the most prevalent and that while ND first started primarily as one-class classification, newer implementations trend more towards multi-class classification in both the classification of known and unknown classes. Moreover, we also observed that the usage of a buffer is very prevalent in the proposed algorithms and that most of these algorithms were designed with tabular data in mind.

Finally, we identified areas where challenges persist and more research is needed, together with recommendations on how to address them. Specifically, we raised the issues of the lack of a properly defined, commonly used definition for ND, the need for an experimental framework including appropriate evaluation metrics for the ND problem as well

as a framework for the data when testing, and the need for a common taxonomy across different works, all of which would help standardize the field and allow for easier comparison between works. Additionally, we also outlined the lack of application of ND in certain scenarios like high-dimensional data, the need for more recent datasets representing DSs in various domains, and the need for more research for dealing with certain characteristics of DSs such as concept scarcity, concept drift, and recurrent concepts.

Building on these findings, the next chapter begins to address the need for better evaluation in ND. It presents an empirical study of performance metrics in imbalanced contexts, a common scenario in ND, where novel classes often occur far less frequently than known ones. This analysis will establish the foundation needed to understand the limitations of existing metrics and will lay the groundwork for our evaluation framework for ND developed in the subsequent chapter.

# Chapter 4

# Empirical Analysis of Performance Assessment for Imbalanced Classification

Based on the findings of our SLR presented in the previous chapter, which highlighted the need for a comprehensive evaluation framework for ND, this chapter advances our second objective stated in Chapter 1. We experimentally study how the choice among commonly used performance metrics affects the evaluation of binary classifiers, and how data characteristics such as class imbalance and noise influence those metrics. Because ND problems often involve class imbalance, where novel classes are greatly underrepresented relative to known classes, the results in this chapter lay the foundation for our evaluation framework specific to ND, introduced in the next chapter. We also provide practical guidelines for selecting performance metrics under different imbalanced settings.

The contents of this chapter are based on the following paper:

Jean-Gabriel Gaudreault and Paula Branco. Empirical analysis of performance assessment for imbalanced classification. *Machine Learning*, 113(8):5533–5575, Aug 2024

## 4.1 Introduction

Correctly evaluating the performance of any ML task is a difficult undertaking, as there is a vast array of metrics to choose from and the choice of a performance metric over another could lead to different outcomes in the selection of the best model. This task involves

not only having a deep knowledge of the context to choose a measure representative of the desired outcome but also selecting a measure that is appropriate for the intrinsic data characteristics of the dataset. This is a well-known issue in imbalanced classification problems, where one of the classes in the data is vastly underrepresented compared to the other. Multiple application domains suffer from this problem including scenarios such as fraud detection, rare illnesses detection, intrusion detection, and many more. In these domains, there is often a particular interest toward the classification of the minority class [32]. This interest needs to be correctly represented in the metric chosen and, as such, some commonly used metrics prove to not be suitable for correctly evaluating the performance of these models [32]. Some metrics such as the $F_\beta$-measure, the G-Mean, or the PR curve have emerged as staples in the research community when dealing with such imbalanced problems. However, with their use slowly starting to become widespread, it raises the questions of how these metrics compare to one another and if certain of these metrics present some undesirable/desirable properties that could lead to different conclusions, depending on the specific deployment context.

We show a concrete example where the use of one performance metric over another would lead to a different outcome in Table 4.1, where four different classifiers were trained on an imbalanced synthetic dataset, with one of the classes representing only 10% of the total number of samples. For demonstration purposes, each classifier was evaluated using three different performance metrics, namely $F_1$ score ($F_\beta$ with $\beta = 1$), H-measure, and G-Mean, the score of which can be seen in parentheses next to the classifier name within the table. As we can observe, a researcher using the $F_1$ score as their metric of choice would end up choosing a different classifier than a researcher who is using G-Mean. Moreover, while the best classifier signaled by the $F_1$ score and H-measure is the same (Random Forest), researchers using these two metrics would end up with different classifiers in the following rank positions. As we will demonstrate in future experiments, this behaviour becomes even more pronounced at higher levels of imbalance, which can become problematic when selecting a metric for evaluating an imbalanced classification problem. We aim to address this issue throughout this chapter, by providing more insights that can help the researchers and end-users in the performance metrics selection process.

In our previous work [96], we explored the impact of the choice of performance metrics and their implementation on the ranking of ML models in an imbalanced setting and concluded that the impact of choosing a different implementation of a performance metric, namely the PR curve, over another is minimal but that the usage of different types of metrics is paramount to get a proper overview of the performance of a model in an imbalanced setting. However, this preceding work mainly focused on analyzing the different ways of interpolating the PR curve and how it impacts the end result when selecting a classifier

Table 4.1: Rankings of multiple classifiers using different performance metrics

| **Rank** | $F_1$ **score** | **H-measure** | **G-Mean** |
|---|---|---|---|
| 1 | Random Forest (0.89) | Random Forest (0.89) | Decision Tree (0.93) |
| 2 | Decision Tree (0.83) | XGBoost (0.87) | Random Forest (0.89) |
| 3 | XGBoost (0.80) | Support Vector (0.78) | XGBoost (0.86) |
| 4 | Support Vector (0.33) | Decision Tree (0.75) | Support Vector (0.45) |

based on that metric, which means it was limited in scope. As such, in this work, we aim to provide more general recommendations on metric selection in imbalanced domains by answering two main questions: (i) which performance metrics are best suited for specific scenarios, and (ii) how the metrics typically used in imbalanced settings are affected by noise in the dataset. For this purpose, we extend our previous research to encompass six additional performance metrics, namely CWA, kappa, MCC, H-measure, Brier score loss, and log loss. Furthermore, we conducted our experiments on 51 additional datasets and broadened our analyses to include the impact of noise on these metrics, as well as new empirical results on metric correlation. To the best of our knowledge, no prior work has provided a thorough and rigorous study, along with practical recommendations, on which performance metrics to use for a given imbalance scenario in binary classification.

**Contributions** The main contributions of this chapter can be summarized as follows:

(i) assess the impact of the choice of a performance metric on the perceived performance of a model in an imbalance context;

(ii) study the effect of noise combined with class imbalance on the studied performance metrics; and

(iii) provide detailed insights and recommendations on which performance metrics are best to use in different case studies.

**Organization** The rest of this chapter will be structured as follows: Section 4.2 will introduce the performance metrics that will be studied and their characteristics; Section 4.3 will present and discuss the results of the experiments determining the correlation between metrics and the impact of choosing one over another; Section 4.4 will compare the metrics' sensitivity to different types of noise and present the results; and Section 4.5 will present recommendations for which performance metrics to use in specific contexts given the previous results and reiterate the multiple conclusions reached throughout the chapter.

## 4.2 Metrics Overview

In this section, we review and briefly summarize the different metrics that will be used in this study. In order to describe the metrics more easily, we refer the reader to the confusion matrix represented in Table 4.2, which shows the prediction results of a binary classification problem, where the negative class is considered the majority class and the positive class the minority one. In the same table, we also define the recall, specificity, precision, and negative predictive value formulas, which will ultimately be used when defining each metric.

Table 4.2: Confusion matrix of a binary classification problem

| | | **Predicted Values** | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| **Actual Values** | Positive | **True Positive** (TP) | **False Negative** (FN) | **Recall (Sensitivity)** $\frac{TP}{TP+FN}$ |
| | Negative | **False Positive** (FP) | **True Negative** (TN) | **Specificity** $\frac{TN}{TN+FP}$ |
| | | **Precision** $\frac{TP}{TP+FP}$ | **Negative Predictive Value** $\frac{TN}{TN+FN}$ | |

The selected metrics are comprised of well-established metrics commonly seen as appropriate when dealing with binary classification in imbalanced domains, which can be divided into three different types as described by [76]: threshold, ranking, and probabilistic metrics. Each of these groups represents metrics that measure the class prediction errors, the ability to correctly separate the classes predicted, and the uncertainty of the predictions, respectively. A summary of the evaluated metrics' characteristics can be seen in Table 4.3 and will be described in more detail in the coming sections.

### 4.2.1 Threshold Metrics

Threshold metrics are characterized by metrics that quantify the model's prediction errors. These metrics are usually represented by a single scalar value.

Table 4.3: Characteristics of analyzed performance metrics

| **Class** | **Metric** | **Confusion Matrix Information** | **Type** | **Uses probabilities** |
|---|---|---|---|---|
| *Threshold* | | | | |
| | $F_\beta$ | TP, FN, FP | Scalar | No |
| | G-Mean | All | Scalar | No |
| | CWA | All | Scalar | No |
| | kappa score | All | Scalar | No |
| | MCC | All | Scalar | No |
| *Ranking* | | | | |
| | ROC curve | All | Graphical | Yes |
| | ROC AUC | All | Scalar | Yes |
| | H-measure | All | Scalar | Yes |
| | PR curve | TP, FN, FP | Graphical | Yes |
| | PR AUC | TP, FN, FP | Scalar | Yes |
| | PRG curve | TP, FN, FP | Graphical | Yes |
| | PRG AUC | TP, FN, FP | Scalar | Yes |
| *Probabilistic* | | | | |
| | Brier Loss | – | Scalar | Yes |
| | Log Loss | – | Scalar | Yes |

### $F_\beta$ Measure

The $F_\beta$-measure (cf. Equation 4.1) [175] aims to represent the trade-off between precision and recall, with the value of $\beta$ controlling the weight assigned to either metric. If $\beta$ is 1, precision and recall have the same importance, while for values of $\beta$ higher (lower) than 1, higher importance is given to recall (precision). In most cases, researchers give equal importance to both metrics ($\beta = 1$), also referred to as the $F_1$-measure or the harmonic mean of the precision and recall.

While we focused on already well-established metrics in this chapter, it is worth noting that other metrics based on the $F_\beta$-measure have also recently been proposed, such as the F-measure curve [195] which allows to better visualize a classifier performance across different parameters such as the variation in importance between precision and recall or the class skew.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{4.1}$$

### Geometric Mean

G-Mean (cf. Equation 4.2) [133] gives equal importance to both classes and is defined as the square root of the recall of each of the classes.

$$\text{G-Mean} = \sqrt{sensitivity \cdot specificity} \tag{4.2}$$

### Class Weighted Accuracy

Likewise to G-Mean, the CWA metric [45] also takes into account the negative class in its computation. However, it addresses one of the issues of G-Mean: the fact that both classes are weighted equally. The CWA metric, defined in Equation 4.3, deals with this problem by allowing to weight the minority class differently, using a variable $w$ ranging in [0,1].

$$\text{CWA} = w \cdot sensitivity + (1 - w) \cdot specificity \tag{4.3}$$

### Kappa Score

Cohen's kappa, also referred to as the kappa score is a measure originally designed to evaluate the agreement between multiple raters [46] and can be applied to evaluate the

performance of a ML classifier by evaluating the agreement between the evaluated classifier and a random classifier, calculated as per Equation 4.4.

$$\text{kappa} = \frac{2(TP \cdot TN - FN \cdot FP)}{(TP + FP)(FP + TN) + (TP + FN)(FN + TN)} \tag{4.4}$$

#### Matthews Correlation Coefficient

The MCC was first introduced in a biochemistry context [154]. It has since been increasingly used as a metric for performance evaluation in imbalanced domains. Similar to the kappa score, it also provides a value determining the level of agreement between different raters, or in the ML context, between the observations and predictions.

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.5}$$

### 4.2.2 Ranking Metrics

While threshold metrics aim to represent a model's prediction correctness, ranking metrics instead quantify the capability of a ML model to separate predicted classes. These metrics are computed by evaluating scores at different thresholds, using the probabilities produced by the model. Because scores are calculated for many thresholds, these metrics are often plotted as curves to visualize classifier performance across thresholds and to better illustrate trade-offs between measurements. For easier comparison between models, the plotted curve is often summarized into a scalar value by computing the AUC.

#### Receiver Operating Characteristic Curve

The ROC curve [65] is one of the most well-known and widely used ranking metrics. However, its use in imbalanced domains, while widespread, has been criticized by many authors [54, 74, 78, 182], who argue that the metric can be overly optimistic in certain scenarios, particularly in imbalanced domains.

The ROC curve plots recall against the FPR, as defined in Equation 4.6. In order to compute its AUC, one can use approximation techniques such as the trapezoidal method or the equivalent Mann–Whitney $U$-statistic [54, 146], both of which are commonly used.

$$\text{FPR} = \frac{FP}{FP + TN} \tag{4.6}$$

### H-measure

Hand [109] presents an alternative metric to the ROC AUC, named the H-measure, which aims to alleviate some of the weaknesses of the ROC curve. For example, the ROC curve evaluates classifiers using different misclassification cost distributions without considering the context, and interpreting the AUC of intersecting ROC curves can be misleading. The H-measure addresses these issues by acknowledging that different misclassifications incur different costs and by explicitly introducing costs for them. Its estimation relies on the same underlying recall and FPR values used to compute the ROC curve but applies a uniform cost distribution that remains the same regardless of the evaluated classifier, thereby resolving one of the main weaknesses of the ROC AUC. Initially, this distribution was intended to be selected subjectively by researchers depending on the context of their problem and reported alongside a universal distribution. However, a standard distribution for unbalanced problems has since been proposed by the authors, which we adopt in this study [110].

### Precision-Recall Curve

While the ROC curve plots recall against the FPR, the PR curve instead plots precision against recall. Since precision emphasizes the minority class by using true and false positives, in contrast to the FPR which uses FPs and TNs, the PR curve is usually preferred over the ROC curve in imbalanced domains. Unlike the ROC curve, however, computing the AUC of the PR curve is not straightforward. Indeed, as Davis and Goadrich [54] describe, the fact that precision does not vary linearly with recall means that linear interpolation techniques, such as the trapezoidal method, do not provide an accurate representation of the AUC and can yield overly optimistic results in certain scenarios. Consequently, other solutions have been proposed in the literature to address this issue. The following sub-sections present some of these alternatives.

#### Interpolated Precision

Manning et al. [145] propose a simple solution to the aforementioned problem. They estimate the precision at any recall point $r$ as the highest precision value of any recall point

greater than $r$. This approach eliminates abrupt changes in precision between consecutive recall points, enabling the use of linear interpolation for the curve.

**Average Precision**
Defined as the sum of precision of each point weighted by the difference in recall of the previous point [205], the AP can be computed as seen in Equation 4.7.

$$\text{AP} = \sum_{n} (recall_n - recall_{n-1}) precision_n \tag{4.7}$$

where $n$ represents the $n$th threshold computed from the prediction scores.

**Davis' Interpolation**
Davis and Goadrich [54] presented another alternative to computing the AUC of the PR curve, by interpolating values for the TP and False Positive (FP) values between two consecutive points.

Let us consider two successive points, $A$ and $B$, and their corresponding underlying TP and FP values denoted as $TP_A$, $TP_B$, $FP_A$, and $FP_B$. We can calculate the number of negative values equalling to one positive value, which is denoted as the local skew (cf. Equation 4.8).

$$\text{Local Skew} = \frac{FP_B - FP_A}{TP_B - TP_A} \tag{4.8}$$

Afterward, new points are created at regular intervals between $A$ and $B$ such that $TP = TP_A + x$ where $x$ represents all integers between 1 and $TP_B - TP_A$. The values representing the precision and recall of these newly created points using the local skew previously calculated are represented in Equation 4.9.

$$\left(\frac{TP_A + x}{\text{Total Positives}}, \frac{TP_A + x}{TP_A + x + FP_A + \frac{FP_B - FP_A}{TP_B - TP_A} x}\right) \tag{4.9}$$

Including these new points alleviates the overestimation of using linear interpolation on the PR curve when two points are far away, as such, using a linear interpolation such as the previously mentioned trapezoidal method can now be performed to estimate the PR AUC.

#### Precision-Recall-Gain Curve

Flach and Kull [79] highlighted drawbacks of using the PR curve, including incoherent scale assumptions and the absence of desirable properties found in the ROC curve, such as a universal baseline and linear interpolation. To address some of these issues, they proposed a new approach, the Precision-Recall-Gain (PRG) curve, which replots the PR curve in a different coordinate system, as detailed by the equations for precision gain (cf. Equation 4.10) and recall gain (cf. Equation 4.11). This new approach establishes a new baseline represented by $\pi$, corresponding to the always-positive classifier.

$$\text{precG} = \frac{precision - \pi}{(1 - \pi)precision} \tag{4.10}$$

$$\text{recG} = \frac{recall - \pi}{(1 - \pi)recall} \tag{4.11}$$

### 4.2.3 Probabilistic Metrics

Rather than evaluating the correctness of class predictions, probabilistic metrics focus on assessing divergence from the true probabilities. They are therefore particularly useful when we want to evaluate the unreliability of a model's predictions and punish confident probabilities that are wrong. While the suitability of using such metrics in imbalanced problems has been questioned before (e.g. [122]), we nevertheless included a few metrics of this type for comparison and completeness.

#### Brier Score Loss

The Brier score loss, defined as the mean squared error between the predicted and true probabilities of the positive class, is computed as shown in Equation 4.12.

$$\text{Brier Score Loss} = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} (y_i - p_i)^2 \tag{4.12}$$

where $n_{samples}$ represents the number of samples, $y_i$ the real value expected and $p_i$ the predicted probability for a given sample $i$.

#### Log Loss

Log loss, often referred to as cross-entropy loss, aims to represent the average difference between the expected and predicted probabilities.

$$\text{Log Loss} = \frac{-1}{n_{samples}} \sum_{i=1}^{n_{samples}} (y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)) \tag{4.13}$$

where $n_{samples}$ represents the number of samples, $y_i$ the real value expected and $p_i$ the predicted probability for a given sample $i$.

### 4.2.4 Other Scenarios

While this chapter focuses on binary imbalanced classification, we also provide an overview of other scenarios, namely imbalanced multi-class and DS cases, which can often occur in combination with imbalance. These scenarios present their own sets of challenges, which we describe in the following sections and further investigate empirically in the next chapter.

#### Multi-class

Multi-class classification, as its name implies, involves problems with more than two classes to which instances can be assigned. In the imbalance case, multiple classes may be categorized as majority or minority classes, with various levels of data distribution skew among them [207]. Hence, properly evaluating the performance of a classifier can be more challenging, as the user's desired importance of correctly classifying each class is not necessarily equal. Moreover, as we will demonstrate in the following sections, there are multiple ways of extending the aforementioned performance metrics to the multi-class scenario, making the selection of a suitable metric for performance evaluation even more challenging than in the binary case.

#### $F_\beta$ Measure

There are two main ways of extending the $F_\beta$ measure to the multi-class scenario. These extensions are based on averaging the score of each class in slightly different ways, namely micro-average, and macro-average. The former, micro-average, globally computes the total number of TP, FN, and FP over all classes [207] to compute the precision and recall as

shown in Equations 4.14 and 4.15 for $K$ classes, and then compute the resulting $F_\beta$ measure as previously defined in Equation 4.1.

$$\text{Micro-Average Precision} = \frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K} TP_i + \sum_{i=1}^{K} FP_i} \tag{4.14}$$

$$\text{Micro-Average Recall} = \frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{K} TP_i + \sum_{i=1}^{K} FN_i} \tag{4.15}$$

Macro-average, on the other hand, gives equal importance to all classes regardless of their size, as it simply computes the unweighted mean of the $F_\beta$ score of each class in a one-vs-rest fashion, as shown in Equation 4.16. It is worth noting that micro-averaging assigns an equal weight to each sample, while macro-averaging gives an equal weight to all classes, regardless of their size.

$$\text{Macro-Average } F_\beta = \frac{1}{K} \sum_{i=1}^{K} F_{\beta,i} \tag{4.16}$$

Another implementation of the macro-average, often described as the weighted average, uses the weight of each class in order to compute the weighted mean of the $F_\beta$-measure for each class, rather than the unweighted mean.

In imbalanced domains, the preferred method depends on the context, as the macro-average gives minority classes as much importance as the majority ones, which may or may not be desirable depending on the situation.

**G-Mean and CWA**

Similar to the binary case, G-Mean can be defined for multi-class scenarios as the $K$th root of the product of the sensitivities (recalls) of all $K$ classes, as shown in Equation 4.17.

$$\text{G-Mean} = \sqrt[K]{\prod_{i=1}^{K} sensitivity_i} \tag{4.17}$$

It is worth noting that, as in the binary case, G-Mean assigns equal importance to all classes regardless of their data distribution. Thus, as we did for the binary scenario, the

CWA metric can be extended for the multi-class scenario by adding a specific weight $w_i$ to each class' sensitivity, enabling to weigh each class differently, as shown in Equation 4.18.

$$\text{CWA} = \sum_{i=1}^{K} w_i \cdot sensitivity_i \tag{4.18}$$

where $\sum_{i=1}^{K} w_i = 1$.

**Kappa Score and MCC**

As described by Grandini et al. [103], MCC and the kappa score can be expressed in terms of a confusion matrix $C$ for $K$ classes with the following subterms [102]:

- $c = \sum_{k}^{K} C_{kk}$ the total number of elements correctly predicted
- $s = \sum_{i}^{K} \sum_{j}^{K} C_{ij}$ the total number of elements
- $p_k = \sum_{i}^{K} C_{ki}$ the number of time that class $k$ was predicted
- $t_k = \sum_{i}^{K} C_{ik}$ the number of times that class $k$ truly occurred

Using these definitions, MCC and the kappa score for the multi-class case are defined as shown in Equations 4.19 and 4.20, respectively.

$$\text{MCC} = \frac{c \cdot s - \sum_{k}^{K} p_k \cdot t_k}{\sqrt{(s^2 - \sum_{k}^{K} p_k^2) \cdot (s^2 - \sum_{k}^{K} t_k^2)}} \tag{4.19}$$

$$\text{kappa} = \frac{c \cdot s - \sum_{k}^{K} p_k \cdot t_k}{s^2 - \sum_{k}^{K} p_k \cdot t_k} \tag{4.20}$$

An interesting observation and added challenge in the multi-class setting is that MCC, which ranges from $[-1, 1]$ in the binary case, no longer has this constant minimal value. Instead, its minimum now has a dynamic value, falling within the range of $-1$ to 0 depending on the distribution of the true labels [103].

### ROC, H-measure, PR curve, and PRG curve

For ranking metrics, which are typically defined only for binary classification problems, a common accepted generalization [111, 134] for multi-class scenarios is to binarize the output using a one-vs-rest or one-vs-all scheme to compute the metric as if it were a binary problem, and then apply micro- or macro-averaging to combine the resulting values, as previously demonstrated for the $F_\beta$ Measure.

### Brier Loss, Log Loss

Both the Brier loss and log loss can easily be generalized to the multi-class scenario by summing the terms over all classes, as shown in Equations 4.21 and 4.22 for $K$ classes.

$$\text{Brier Score Loss} = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} \sum_{j=1}^{K} (y_{ij} - p_{ij})^2 \tag{4.21}$$

$$\text{Log Loss} = \frac{-1}{n_{samples}} \sum_{i=1}^{n_{samples}} \sum_{j=1}^{K} y_{ij} \cdot \log(p_{ij}) \tag{4.22}$$

### Data Streams

The imbalanced DSs domain refers to scenarios where the data arrives at the decision model in a continuous, dynamic fashion rather than as a one-time static evaluation. This field has recently gained considerable interest, as multiple real-world applications fall under this setting, such as sensor networks, financial data, and network traffic. In addition to the traditional challenges of evaluating an imbalanced problem, such as the under-representation of one or more of the classes, DSs present new challenges, as models typically evolve over time rather than remaining fixed, and the data is continuous and drawn from a non-stationary distribution [84]. A common way to mitigate these issues is to evaluate decision models continuously over a sliding window of a specific number of samples, using the same performance metrics we previously defined for the binary and multi-class cases [6, 84].

Another interesting sub-domain of DSs is the problem of ND. In this task, the objective of the model is to detect incoming samples that belong to concepts different from the already known ones [71]. While regular performance metrics can be used to evaluate these models, correctly identifying novelties is often more important for the end user than accurately labeling the known classes. This has led to the introduction of domain-specific

metrics such as the percentage of novel class samples misclassified as normal ($M_{\text{new}}$) and the proportion of normal samples misclassified as novelties ($F_{\text{new}}$) [71]. There is, however, still a need for further research and the development of performance metrics that properly assess the performance of models in this field, a topic that will be explored in the next chapter.

### 4.2.5 Summary

Considering the different metrics shown in this section, we will compare all five threshold-based metrics ($F_1$-measure, G-Mean, CWA, kappa, and MCC) and the two probabilistic metrics presented (Brier Loss, Log Loss). For the four presented ranking metrics, as one can not easily compare the curves given by most of these metrics, we will compute the AUC where it is applicable. Given the multiple available interpolation methods of PR AUC, we will also compare the three proposed interpolations along the flawed linear interpolation of the curve, resulting in a total of four metrics representing the PR AUC.

## 4.3 Evaluation of the Impact on Rankings

In this section, we first study the impact of using different performance metrics on the interpretation of a model's performance. In other words, we want to analyze how the different performance metrics used in imbalanced domains relate to each other and how the usage of one specific metric could portray a different result when compared to another one. This section is organized into four subsections: (i) a description of the methodology used when performing the experiments; (ii) an exploration of the correlation between the metrics using well-known correlation coefficients; (iii) an analysis of the choice of performance metric on the ranking of a model; and (iv) a summary of the results within this section and main takeaways.

### 4.3.1 Methodology

Based on our previous work [96], we extended our experiments to be performed on a total of 66 binary datasets taken from the KEEL repository [15]. This repository comprises 20 unique datasets that were either originally binary or converted from multi-class to binary form. We must highlight that this is a common repository used by the research community for evaluating solutions for the class imbalance problem. Moreover, the binarized versions

generated from a base dataset have different characteristics, including the number of features, total number of samples, and IR[1]. Still, the fact that these 66 datasets are derived from an initial set of 20 might bring some limitations in terms of the generalizability of our results. These datasets were selected depending on their IR (cf. Equation 4.23), and were split equally into three categories: slightly (IR $< 9$), moderately ($9 \leq$ IR $< 13$), and heavily (IR $\geq 13$) imbalanced. Tables 4.4 to 4.6 summarize the main characteristics of the selected datasets.

$$\text{IR} = \frac{\text{majority class samples}}{\text{minority class samples}} \tag{4.23}$$

The following experiments were conducted using four widely used types of ML models that exhibit different characteristics, namely decision tree, random forest, XGBoost, and Support Vector Machine (SVM). These specific classifiers were chosen because they are generally well understood and widely available in different programming language libraries, which is important for both interpretability and reproducibility purposes, respectively. In addition, we wanted to include a base classifier (decision tree), as well as variants of it which are used with ensemble methods based on both boosting (XGBoost) and bagging (random forest).

These models were evaluated using the 14 performance metrics and AUC interpolation techniques presented in Section 4.2, which represent a broad range of evaluation metrics often used in imbalanced domains, including lesser-used ones included for comparison purposes.

As we will discuss further in the coming sections, two main experiments were performed with the following three goals in mind: (i) to extend our previous work to include new metrics and datasets; (ii) to establish if a correlation exists between the different performance metrics evaluated; and (iii) to analyze the impact of the ranking given by a specific metric compared to another. These experiments will help us garner more information on the strengths and weaknesses of each metric, in order to provide more general recommendations accordingly. The complete results, as well as the code to reproduce these experiments, are freely available at https://github.com/jgaud/PerformanceEvaluationImbalancedClassification.

[1]For instance, the "ecoli1" and the "ecoli-0-1-4-6_vs_5" are both obtained from the base dataset ecoli, but have different characteristics: the "ecoli1" has 7 features, a total of 336 samples, and an IR of 3.36, while the "ecoli-0-1-4-6_vs_5" has 6 features, 280 samples, and an IR of 13. Other examples, such as the different versions of the glass dataset also present a varying total number of samples (ranging from 92 to 214) and IRs (ranging from 1.82 to 22.78).

Table 4.4: Main characteristics of datasets with IR $< 9$, ordered by increasing IR.

| Name | Nr. Features | Nr. cases | IR |
|---|---|---|---|
| glass1 | 9 | 214 | 1.82 |
| wisconsin | 9 | 683 | 1.86 |
| ecoli-0_vs_1 | 7 | 220 | 1.86 |
| pima | 8 | 768 | 1.87 |
| iris0 | 4 | 150 | 2 |
| glass0 | 9 | 214 | 2.06 |
| yeast1 | 8 | 1484 | 2.46 |
| haberman | 3 | 306 | 2.78 |
| vehicle2 | 18 | 846 | 2.88 |
| vehicle1 | 18 | 846 | 2.9 |
| glass-0-1-2-3_vs_4-5-6 | 9 | 214 | 3.2 |
| vehicle0 | 18 | 846 | 3.25 |
| vehicle3 | 18 | 846 | 3.25 |
| ecoli1 | 7 | 336 | 3.36 |
| new-thyroid2 | 5 | 215 | 5.14 |
| new-thyroid1 | 5 | 215 | 5.14 |
| ecoli2 | 7 | 336 | 5.46 |
| segment0 | 19 | 2308 | 6.02 |
| glass6 | 9 | 214 | 6.38 |
| yeast3 | 8 | 1484 | 8.1 |
| ecoli3 | 7 | 336 | 8.6 |
| page-blocks0 | 10 | 5472 | 8.79 |

Table 4.5: Main characteristics of datasets with $9 \leq$ IR $< 13$, ordered by increasing IR.

| Name | Nr. Features | Nr. cases | IR |
|---|---|---|---|
| ecoli-0-3-4_vs_5 | 7 | 200 | 9 |
| yeast-2_vs_4 | 8 | 514 | 9.08 |
| ecoli-0-6-7_vs_3-5 | 7 | 222 | 9.09 |
| ecoli-0-2-3-4_vs_5 | 7 | 202 | 9.1 |
| glass-0-1-5_vs_2 | 9 | 172 | 9.12 |
| yeast-0-3-5-9_vs_7-8 | 8 | 506 | 9.12 |
| yeast-0-2-5-7-9_vs_3-6-8 | 8 | 1004 | 9.14 |
| yeast-0-2-5-6_vs_3-7-8-9 | 8 | 1004 | 9.14 |
| ecoli-0-4-6_vs_5 | 6 | 203 | 9.15 |
| ecoli-0-1_vs_2-3-5 | 7 | 244 | 9.17 |
| ecoli-0-2-6-7_vs_3-5 | 7 | 224 | 9.18 |
| glass-0-4_vs_5 | 9 | 92 | 9.22 |
| ecoli-0-3-4-6_vs_5 | 7 | 205 | 9.25 |
| ecoli-0-3-4-7_vs_5-6 | 7 | 257 | 9.28 |
| yeast-0-5-6-7-9_vs_4 | 8 | 528 | 9.35 |
| vowel0 | 13 | 988 | 9.98 |
| ecoli-0-6-7_vs_5 | 6 | 220 | 10 |
| glass-0-1-6_vs_2 | 9 | 192 | 10.29 |
| led7digit-0-2-4-5-6-7-8-9_vs_1 | 7 | 443 | 10.97 |
| glass-0-1-4-6_vs_2 | 9 | 205 | 11.06 |
| glass2 | 9 | 214 | 11.59 |
| cleveland-0_vs_4 | 13 | 177 | 12.62 |

Table 4.6: Main characteristics of datasets with IR $\geq 13$, ordered by increasing IR.

| Name | Nr. Features | Nr. cases | IR |
|---|---|---|---|
| ecoli-0-1-4-6_vs_5 | 6 | 280 | 13 |
| dermatology-6 | 34 | 358 | 16.9 |
| zoo-3 | 16 | 101 | 19.2 |
| shuttle-6_vs_2-3 | 9 | 230 | 22 |
| glass5 | 9 | 214 | 22.78 |
| winequality-red-4 | 11 | 1599 | 29.17 |
| poker-9_vs_7 | 10 | 244 | 29.5 |
| yeast-1-2-8-9_vs_7 | 8 | 947 | 30.57 |
| abalone-3_vs_11 | 8 | 502 | 32.47 |
| winequality-white-9_vs_4 | 11 | 168 | 32.6 |
| winequality-red-8_vs_6 | 11 | 656 | 35.44 |
| abalone-17_vs_7-8-9-10 | 8 | 2338 | 39.31 |
| winequality-white-3_vs_7 | 11 | 900 | 44 |
| winequality-red-8_vs_6-7 | 11 | 855 | 46.5 |
| abalone-19_vs_10-11-12-13 | 8 | 1622 | 49.69 |
| winequality-white-3-9_vs_5 | 11 | 1482 | 58.28 |
| poker-8-9_vs_6 | 10 | 1485 | 58.4 |
| shuttle-2_vs_5 | 9 | 3316 | 66.67 |
| winequality-red-3_vs_5 | 11 | 691 | 68.1 |
| abalone-20_vs_8-9-10 | 8 | 1916 | 72.69 |
| poker-8-9_vs_5 | 10 | 2075 | 82 |
| poker-8_vs_6 | 10 | 1477 | 85.88 |

### 4.3.2 Correlation Between Metrics

In order to evaluate the correlation of the metrics under different conditions, we used an approach similar to the one used by Ferri et al. [76]. We computed both the Pearson and Spearman correlation matrices on each dataset independently of one another. The former evaluates the linear correlation between two variables [170], while the latter evaluates how well two variables can be represented with a monotonic relation using the ranked values rather than the absolute values [199]. Since each dataset can display widely different characteristics such as a different number of samples, IR, class separation, etc. which in return influence the value of the performance metrics evaluated, it is important to evaluate the correlation matrix of each dataset independently. Subsequently, we compute the average of these correlations over all datasets, and over each imbalance group (slightly, moderately, and highly imbalanced). Since both methods (Pearson and Spearman) led to similar conclusions and for ease of reading, we only included the Spearman rank correlation matrix over all datasets, as shown in Figure 4.1.

Moreover, to display the results of the coefficient matrices in a way where the relations between the different clusters can easily be seen and interpreted, we performed hierarchical clustering using the average method, as described in [161]. This method specifically, was chosen as we found its results interesting and because this choice was also supported by another analysis of metrics which was performed previously [76]. The results of these clusterings are shown in dendrograms corresponding to each of the aforementioned coefficient matrices and can be seen in Figures 4.2, 4.3, 4.4 and 4.5, which correspond to the overall results and each group results starting from the less imbalanced to the most imbalanced datasets.

#### Discussion of Results

A first conclusion that can be drawn from these results is that there generally exist three main clusters of metrics that show up in every imbalance group and which display a high correlation between themselves. Each of these clusters corresponds to a specific group of metrics presented in Section 4.2, that is threshold, ranking, and probabilistic metrics. In effect, this first conclusion is in agreement with our expectations and past conclusions [96]. These different groups of metrics exhibit different properties due to their very nature and thus they evaluate the performance of a model from a different perspective when we compare the three groups.

Looking more closely at the correlations between specific metrics, the different PR curves implementations unsurprisingly display a high correlation between themselves. We also no-

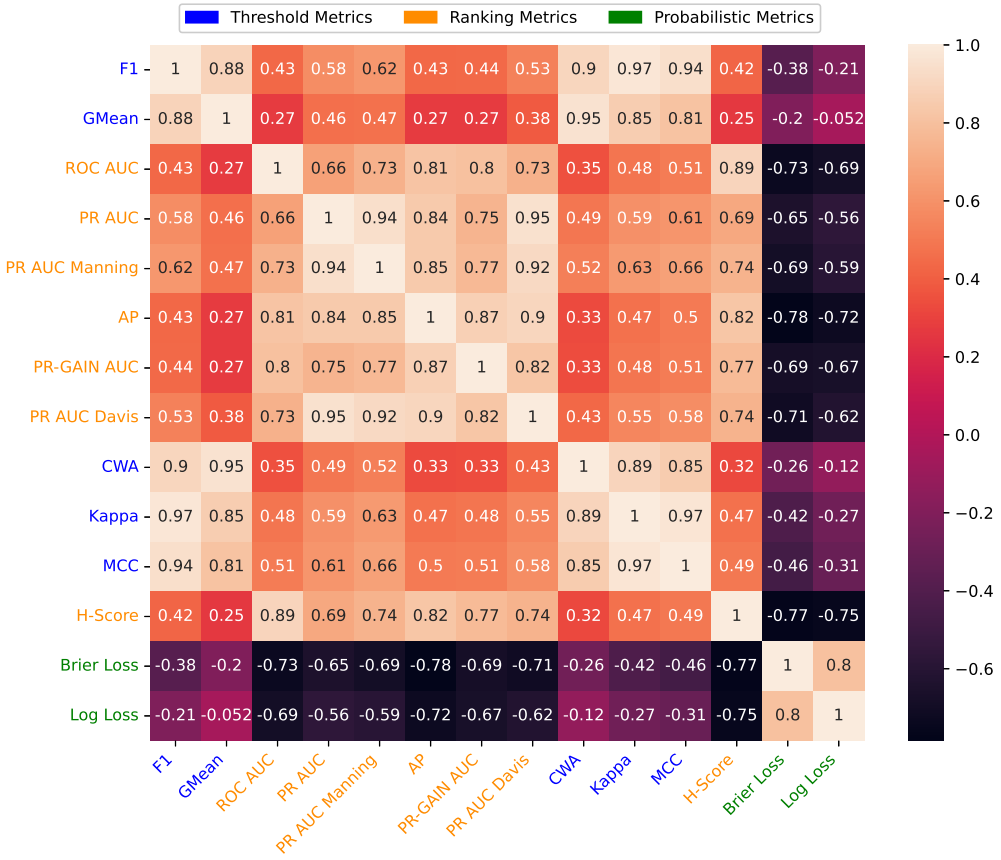



Figure 4.1: Spearman correlation coefficients of all evaluated metrics over all datasets

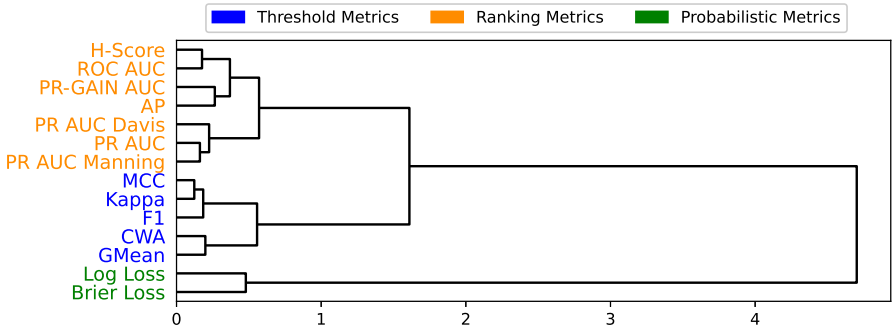



Figure 4.2: Hierarchical clustering of all evaluated metrics over all datasets

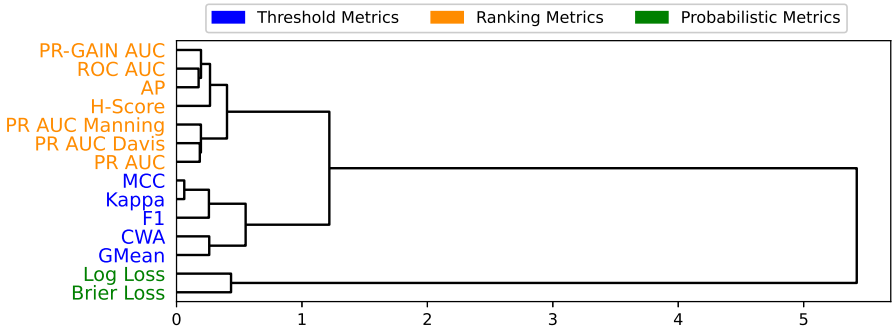



Figure 4.3: Hierarchical clustering of all evaluated metrics over datasets with IR $< 9$

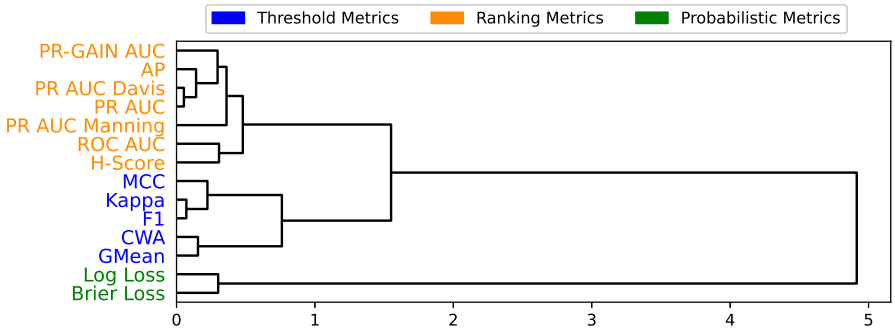



Figure 4.4: Hierarchical clustering of all evaluated metrics over datasets with $9 \leq$ IR $< 13$

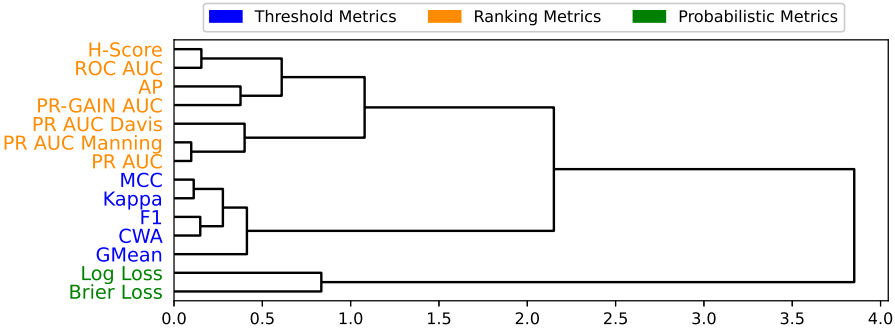



Figure 4.5: Hierarchical clustering of all evaluated metrics over datasets with IR $\geq 13$

tice the existence of a separate group within the threshold metrics when looking at the correlations between MCC, kappa, and $F_1$. While these three metrics are highly correlated, we observe more specifically that the first two metrics present a high level of correlation between themselves across all groups of imbalance, even at the highest ratio. Investigating further, we can look at the pairwise comparison between the two metrics as shown in Figure 4.6 which displays a clear linear relationship between the two metrics, although kappa seems to give a lower score than MCC in some cases. This is a really interesting observation, as both metrics effectively measure correlation or agreement between classifiers. It has been shown [43, 57], however, that while these two metrics can give similar results in most scenarios, kappa can exhibit undesirable behaviour in imbalance scenarios and should therefore be avoided. Indeed, in some cases which occur commonly with imbalanced data, such as when TPs and TNs are zero, the kappa score can provide a higher score for worse classification results and therefore lead to wrong conclusions.

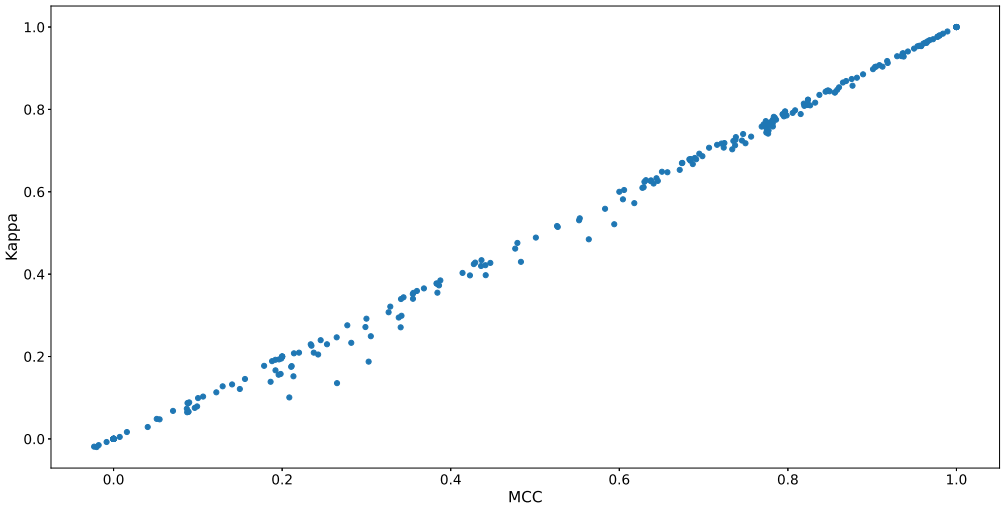



Figure 4.6: Pairwise comparison between MCC and kappa

Inspecting the results from the hierarchical clustering (cf. Figure 4.2), we can observe from the hierarchical clustering that both the threshold and ranking metrics are at a much closer linkage distance than the two evaluated probabilistic metrics (Log Loss and Brier Loss). Indeed, the fact that the latter group of metrics uses the model's probabilities rather than the number of correct/incorrectly predicted labels like the first two groups would likely explain the higher divergence.

Another interesting observation related to the linkage length of the different dendrograms representing each dataset imbalance group (cf. Figure 4.3, 4.4 and 4.5) is that the distance between the different clusters within the threshold and ranking metrics tends to increase with the increase in imbalance, while the previously mentioned distance between probabilistic metrics and the two first groups decreases. This behaviour shows that clusters that are highly correlated, such as MCC and kappa, still are at higher IRs, while other clusters that are less correlated, exhibit a higher variance at higher degrees of imbalance. This reiterates the importance of using multiple metrics, especially in highly imbalanced scenarios, as we can see that these metrics might drift apart and offer different interpretations, a conclusion we will explore further in the following section.

### 4.3.3 Impact of the Choice of Performance Metrics

In this section, we repeat a part of the experiments performed in our previous work with a more extensive set of performance metrics and datasets. This will allow us to observe the validity of our previous results due to the use of more datasets, but also extend our insights on how the newly added metrics behave in comparison to the previous ones. The main objective of these experiments is to evaluate if the use of one performance metric over another provides a significant disparity in the ranking of multiple ML models at different imbalance levels. Therefore, the results will be presented in the form of rankings rather than the absolute values of the metrics. For a given metric, each one of the $k$ evaluated models will be given a rank from 1 to $k$ depending on the score given by the metric. In our experiments we considered 4 ML algorithms (decision tree, random forest, XGBoost, and SVM), thus we will have ranks varying from 1 to 4.

#### Assessing the Impact of the Performance Metrics

While we can evaluate the performance of classifiers using performance metrics, evaluating the metrics themselves is a difficult task, as there are few ways to objectively measure if a metric is behaving as expected. As such, in order to assess the impact of using a specific performance metric rather than another, we used two benchmark metrics, namely the Krippendorff's Alpha score and the MD, as described in the following sections. These will allow us to measure the agreement between the different metrics and which metric deviates from the rankings given by others. It is worth noting, however, that a higher or lower level agreement between a specific metric compared to others does not necessarily mean that the metric is good or bad, but rather that it gives a similar, or different viewpoint which can be desirable depending on the context.

### Mode Difference

As we defined in [96], MD (cf. Equation 4.25) is a measure used to efficiently observe how the ranking given by an observer, in our case a performance metric, differs from the general consensus of a set of observers. Assuming $n$ performance metrics evaluating $k$ classifiers, the rankings for each metric $m \in \{1, \cdots, n\}$ on each classifier is denoted by $r_{m,1}, r_{m,2}, ..., r_{m,k}$. This allows us to calculate the mode of the rankings for each classifier (cf. Equation 4.24), in other words, the most frequent rank obtained for a classifier, or the average of the most frequent values when there are ties.

$$Mo_j = Avg(Mode(r_{1,j}, r_{2,j}, ..., r_{n,j})) \tag{4.24}$$

$$\text{MD}_{i,j} = \|r_{i,j} - Mo_j\| \tag{4.25}$$

Finally, considering $k$ classifiers, we compute the average of MDs over a performance metric $i$ (cf. Equation 4.26) in order to give an average value of the variability of the rankings of the metric over all the tested classifiers. In other words, the higher this value is, the more a given metric differs from the consensus of the other tested metrics.

$$\text{MD\_avg}_i = Avg(MD_{i,1}, MD_{i,2}, ..., MD_{i,k}) \tag{4.26}$$

The value of $\text{MD\_avg}_i$ for a performance metric $i$ varies between 0 and $k-1$ where the lower bound signifies that the given metric agrees all the time with the consensus of the other metrics and the upper bound means that it disagrees the most.

### Krippendorff's Alpha

As defined by Krippendorff [132], the Krippendorff's alpha is a metric that measures the level of agreement between multiple raters, or in our case, performance metrics. The value ranges from 0 to 1, depending on whether all raters disagree or agree, respectively.

## Results

### Global Results

Our initial results involve computing the score of each one of the 4 classifiers using all 14

evaluated metrics on the 66 selected datasets. As previously mentioned, we then transform these values into rankings and compute the MD for each performance metric. Table 4.7 presents these results for the Wisconsin dataset, the second least imbalanced dataset out of the 66 evaluated, which has been included for demonstration of the calculation of the MD. We do not include all the results for the remaining datasets here due to space, readability, and interpretability constraints[2], but we will observe later that the differences in metrics' rankings are more pronounced on datasets presenting a higher level of imbalance. As such, the results are then combined using an arithmetic average of the MD over each class IR, as defined in Section 4.3.1: slightly (IR $<$ 9), moderately (9 $\leq$ IR $<$ 13), and heavily (IR $\geq$ 13) imbalanced. Table 4.8 presents these aggregated results for all of the performance metrics.

As per our previous work's conclusions, we can confirm the fact that the different interpolations of the same metric and other metrics associated with the PR curve (PR AUC with Linear Interpolation, Interpolated Precision AUC, AP, PRG AUC, and PR AUC Davis) all present similar rankings, as their MD is averaging lower scores. This means that these metrics present a higher level of agreement with the general ranking consensus between all tested metrics. Furthermore, we remark that the G-Mean, the $F_1$ measure, and other newly added metrics, namely the kappa score, MCC, Brier loss, and log loss all present a higher level of disagreement, showing that these metrics provide considerably different ranking results.

Another interesting observation is that the ROC AUC and H-measure metrics, while not being as low as their PR AUC counterparts, both present lower levels of disagreement between each other than other metrics. We believe that this behaviour can be interpreted by two explanations: (i) being both rank-based metrics, they are more likely to give a similar ranking to the PR AUC interpolation methods and thus, obtain a higher level of agreement; and, (ii) as discussed in Section 4.2, the H-measure is actually based on the ROC curve for its computation.

**Threshold Metrics**

Table 4.9 displays the MD averages and Krippendorff's Alpha computed only on the five threshold metrics: $F_1$, G-Mean, CWA, kappa, and MCC. We notice here, as expected, a consensus of all metrics, which leads to an overall lower MD than what we previously saw in the experiment with all metrics. An interesting observation here is that while we

[2]The results on other datasets as well as stratified cross-validated results, that were not included in the manuscript, are available at the following link https://github.com/jgaud/PerformanceEvaluationImbalancedClassification.

Table 4.7: Ranks and MD of each performance metrics on the Wisconsin dataset (best performing model for each metric is underlined).

| Metric | Decision Tree | | Random Forest | | XGBoost | | SVM | | $MD_{avg}$ |
|---|---|---|---|---|---|---|---|---|---|
| | Rank | MD | Rank | MD | Rank | MD | Rank | MD | |
| $F_1$ | 4 | 0 | 2 | 0 | 3 | 1 | <u>1</u> | 2 | 0.75 |
| G-Mean | 4 | 0 | 2 | 0 | 3 | 1 | <u>1</u> | 2 | 0.75 |
| ROC AUC | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |
| PR AUC (Linear Interpolation) | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |
| Interpolated Precision AUC | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |
| AP | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |
| PRG AUC | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |
| PR AUC Davis | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |
| CWA | 4 | 0 | 2 | 0 | 3 | 1 | <u>1</u> | 2 | 0.75 |
| kappa | 4 | 0 | 2 | 0 | 3 | 1 | <u>1</u> | 2 | 0.75 |
| MCC | 4 | 0 | 2 | 0 | 3 | 1 | <u>1</u> | 2 | 0.75 |
| H-measure | 4 | 0 | <u>1</u> | 1 | 3 | 1 | 2 | 1 | 0.75 |
| Brier Loss | 4 | 0 | 2 | 0 | 3 | 1 | <u>1</u> | 2 | 0.75 |
| Log Loss | 4 | 0 | 2 | 0 | <u>1</u> | 1 | 3 | 0 | 0.25 |

Table 4.8: MD of all metrics averaged by imbalance class (best score by imbalance level is underlined; best overall score is in boldface).

| Metric | IR $<$ 9 | 9 $<=$ IR $<$ 13 | 13 $<=$ IR |
|---|---|---|---|
| $F_1$ | 0.4261 | 0.5284 | 0.625 |
| G-Mean | 0.6534 | 0.7556 | 0.7159 |
| ROC AUC | 0.267 | 0.3579 | 0.5227 |
| PR AUC (Linear Interpolation) | 0.2329 | 0.267 | 0.3068 |
| Interpolated Precision AUC | **0.1647** | 0.1988 | 0.25 |
| AP | 0.2556 | 0.2897 | 0.375 |
| PRG AUC | 0.267 | 0.3806 | 0.4431 |
| PR AUC Davis | 0.1875 | 0.2443 | 0.2159 |
| CWA | 0.5397 | 0.6647 | 0.625 |
| kappa | 0.3579 | 0.5284 | 0.5568 |
| MCC | 0.3352 | 0.4829 | 0.534 |
| H-measure | 0.1761 | 0.2556 | 0.534 |
| Brier Loss | 0.3579 | 0.4829 | 0.5795 |
| Log Loss | 0.5284 | 0.4943 | 0.7954 |
| Krippendorff's Alpha | 0.71801 | 0.62049 | 0.50324 |

previously mentioned that the disagreement between different metrics tends to increase in correlation with the increase in the imbalance of the dataset, this behaviour does not seem to be as pronounced when using threshold metrics exclusively. Indeed, we can see with both the MD and Krippendorff's Alpha, that the agreement level stays in fact relatively constant across imbalance levels compared to the previous results and that it is at its lowest in the moderately averaged imbalance setting, mainly caused by G-Mean and CWA. Both of these metrics use specificity, which considers the TNs of the confusion matrix in the computation of their value, while the others do not. The following logical explanation is that giving weight to the majority (negative) class using the true negative value, appears to change the rankings of the classifier in a notable way compared to other metrics that do not. This behaviour could be desirable or not depending on the context of the problem, as we will discuss later on. Moreover, looking more closely at the relationship between one of these metrics, G-Mean, and the $F_1$ score in Figure 4.7, we see that while the score of the two metrics seems to be closely related at higher scores, it does deviate much more at lower ones. Indeed, G-Mean tends to give a much higher score than $F_1$ on classifiers that are performing worse. Again, the fact the G-Mean also gives equal importance to the majority class by using the TNs would explain this behaviour.

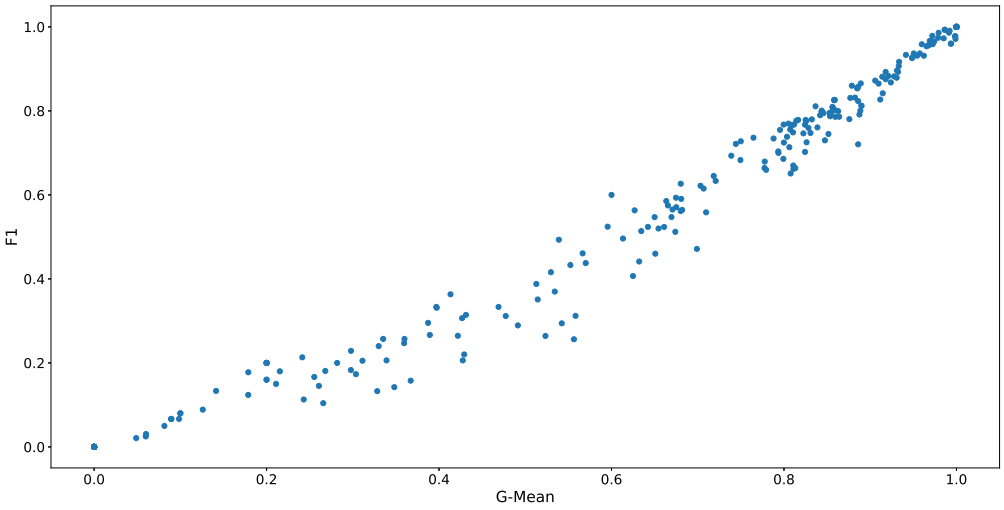



Figure 4.7: Pairwise comparison between G-Mean and $F_1$

Table 4.9: MD of threshold metrics averaged by imbalance class (best score by imbalance level is underlined; best overall score is in boldface).

| Metric | MD Average | | |
|---|---|---|---|
| | IR $<$ 9 | 9 $<=$ IR $<$ 13 | 13 $<=$ IR |
| $F_1$ | <u>0.0227</u> | <u>**0.0113**</u> | 0.1136 |
| G-Mean | 0.2954 | 0.284 | 0.2272 |
| CWA | 0.1818 | 0.2613 | 0.0909 |
| kappa | 0.0454 | 0.0568 | <u>0.0227</u> |
| MCC | 0.0681 | 0.1704 | 0.0681 |
| Krippendorff's Alpha | 0.91623 | 0.87477 | 0.90759 |

**Ranking Metrics**

For ranking metrics, being that we analyzed multiple implementations of the same performance metric (PR AUC) and we expect these implementations to have similar results as per our previous work [96], we chose only one metric to represent all of the PR AUC implementations, namely Davis' interpolation. This implementation was chosen due to it being the most consistent with the consensus of the group in our previous work's experiments, a conclusion which we will also verify in the next section.

Table 4.10 presents the MD Average and Krippendorff's Alpha over these metrics. In this case, in contrast with the threshold metrics previously tested, there is a more noticeable decrease in agreement between the metrics with the increase in imbalance, as seen by the Krippendorff's Alpha. Interestingly, there does not seem to be any metric that stands out of the others with its level of agreement. For example, the ROC AUC does display the most agreement with the common consensus in the lowest imbalance setting, but not in the intermediate one and highest one where PRG AUC is relatively low. It is likely that these ranking metrics evaluate a model quite differently depending on its context and dataset imbalance. For example, this behaviour was previously demonstrated by Hand[109] between the H-measure and ROC AUC and by Davis and Goadrich [54] between the ROC AUC and PR AUC.

**Probabilistic Metrics**

Since we only included two different probabilistic metrics, computing the MD for these would not make sense as they will both get the same value. Nonetheless, we can still compute the Krippendorff's Alpha which measures the general agreement between them

Table 4.10: MD of ranking metrics averaged by imbalance class (best score by imbalance level is underlined; best overall score is in boldface).

| Metric | MD Average | | |
|---|---|---|---|
| | IR < 9 | 9 <= IR < 13 | 13 <= IR |
| ROC AUC | **0.1193** | 0.2727 | 0.2045 |
| PRG AUC | 0.1534 | 0.2045 | 0.1818 |
| PR AUC Davis | 0.2102 | 0.2045 | 0.3863 |
| H-measure | 0.1534 | 0.2045 | 0.2045 |
| Krippendorff's Alpha | 0.86932 | 0.81534 | 0.75710 |

at each imbalance level, the results of which are displayed in Table 4.11. In this case, we observe once again a decrease in agreement as the imbalance between the classes' distribution increases.

Table 4.11: Agreement of probabilistic metrics averaged by imbalance class

| Metric | IR < 9 | 9 <= IR < 13 | 13 <= IR |
|---|---|---|---|
| Krippendorff's Alpha | 0.86477 | 0.85681 | 0.75341 |

**PR AUC Implementations**

In this section, we test the agreement of the different calculations of the PR AUC in order to determine if the use of one implementation over another would lead to different results in the perceived performance of a model compared to another.

The results in Table 4.12 show that across all levels of imbalance, the agreement of all metrics stays relatively high and only drops at higher IRs, mainly due to AP. These results are in line with our previous conclusions [96], as it shows that the different interpolations of the PR AUC do lead to very similar results compared to the difference between entirely different metrics for example. Moreover, PR AUC Davis exhibits the highest level of agreement across all IRs which is why we chose it as our baseline to represent the PR AUC metric in other experiments and why we recommend it as a good implementation to compute the PR AUC as it depicts a good overall representation.

Table 4.12: MD of all PR AUC implementations averaged by imbalance class (best score by imbalance level is underlined; best overall score is in boldface).

| Metric | MD Average | | |
|---|---|---|---|
| | IR < 9 | 9 <= IR < 13 | 13 <= IR |
| PR AUC (Linear Interpolation) | 0.0909 | 0.034 | 0.125 |
| Interpolated Precision AUC | 0.1477 | 0.1477 | <u>0.1022</u> |
| AP | 0.1818 | 0.1022 | 0.3522 |
| PR AUC Davis | <u>0.0795</u> | **<u>0.0113</u>** | <u>0.1022</u> |
| Krippendorff's Alpha | 0.90767 | 0.94602 | 0.86506 |

**Threshold and Ranking Metrics**
Finally, we inspect in Table 4.13 both threshold and ranking metrics together, choosing again PR AUC Davis in order to represent the different PR AUC implementations for the same reasons stated above.

First of all, we can note the much higher agreement of threshold metrics with the consensus compared to ranking metrics. As we saw before, when analyzing each group individually, the agreement of threshold metrics was higher between themselves than ranking ones, especially at higher IRs. Moreover, these results show a significant decrease in the agreement between all metrics as the imbalance increases. In the highest imbalance group, most ranking metrics disagree the most with the consensus while threshold ones agree the most. This observation solidifies the conclusion and recommendation that when dealing with highly imbalanced data, the use of multiple performance metrics, particularly ones that are fundamentally different is strongly recommended, as they exhibit higher variance in their results.

Secondly, we can observe again the cluster formed between $F_1$, kappa, and MCC presented in Section 4.3.2 when using hierarchical clustering. Indeed, these three metrics seem closely related, as we observe a fairly high level of agreement at all imbalance levels.

### 4.3.4 Summary

This section summarizes the multiple conclusions reached in Section 4.3, as we evaluated the impact of using different performance metrics on the ranking of different models.

- The three different types of metrics (threshold, ranking, and probabilistic) exhibit

Table 4.13: MD of threshold and ranking metrics averaged by imbalance class (best score by imbalance level is underlined; best overall score is in boldface).

| Metric | MD Average | | |
|---|---|---|---|
| | IR $<$ 9 | 9 $<=$ IR $<$ 13 | 13 $<=$ IR |
| $F_1$ | 0.1875 | <u>0.1647</u> | 0.1988 |
| G-Mean | 0.4147 | 0.4147 | 0.2897 |
| ROC AUC | 0.4943 | 0.5397 | 0.9147 |
| PRG AUC | 0.4943 | 0.6079 | 0.8238 |
| PR AUC Davis | 0.3806 | 0.6079 | 0.6306 |
| CWA | 0.3011 | 0.3238 | 0.1988 |
| kappa | 0.1193 | <u>0.1647</u> | 0.1193 |
| MCC | <u>0.1079</u> | 0.1875 | **<u>0.0965</u>** |
| H-measure | 0.4034 | 0.5397 | 0.9488 |
| Krippendorff's Alpha | 0.72699 | 0.63443 | 0.53230 |

very different behaviour, and are generally highly correlated with other metrics of their own group.

- Threshold and ranking metrics are much more closely related between themselves than either of them are to probabilistic metrics. However, this correlation does decrease when the imbalance of the data increases, which shows the importance of using multiple types of metrics, especially in high imbalance contexts.

- When looking at threshold metrics specifically, MCC and kappa give extremely similar results, but kappa does exhibit some unwanted properties when dealing with imbalanced domains which means that the use of MCC should be preferred.

- The different implementations of the PR AUC (Linear Interpolation, Interpolated Precision, AP, and Davis) are very close to one another, with AP exhibiting more disagreement than the others, while PR AUC Davis comes out on top in relation to it consistently agreeing with the other implementations, which is why it should be favored when computing the PR AUC.

- PRG displays different properties than the different computations of the PR AUC and should therefore be treated as a different metric.

## 4.4 Evaluation of the Impact of Noise

In this section, we study and analyze how the different performance metrics we selected are affected by different intrinsic characteristics of the data, such as noise and class imbalance. Depending on the data and context of deployment, these characteristics can be present and can potentially affect the ability of certain metrics to accurately give insights into a model's performance. As such, we want to provide future insights and recommendations for researchers and computer scientists to choose the right metric for their application. This section will be divided into two subsections described as follows: (i) presentation of the methodology used throughout these experiments; and (ii) analysis and discussion of the results.

### 4.4.1 Methodology

The problem of evaluating the impact of noise and other dataset properties on performance metrics is complex, as introducing noise and modifying the dataset directly can impact a ML model as well as the performance metric which would render the results unusable for our purpose. For example, a given classifier could perform worse in a scenario where noise is introduced in the data and there would be no way to determine the impact of the noise on the metric itself rather than the classifier. Therefore, we needed to devise a way to isolate the impact on the performance metrics alone. To do so, we adopted an approach similar to the one presented by Ferri et al. [76], which will be presented in the remainder of this section.

First, we start by creating a synthetic dataset containing 1000 samples with the specific characteristics we want to test in a given experiment. Then, a classifier $C_1$ is artificially created on this data, by simply generating probabilities for each $y$ sample, using random values in a uniform distribution between [0, 0.5) or (0.5, 1), depending on the true value of $y$. A total of 10% of these probabilities (100 samples) are then reshuffled, now using a random value within the uniform distribution of [0,1]. In other words, we are artificially creating a classifier that is very accurately predicting the data created, this is done so in order to eliminate the impact that the classifier has and focus solely on the impact on the performance metrics since we are not interested in evaluating the classifier's robustness to noise, but rather the metric's. Finally, we define a second classifier $C_2$, by using the same probabilities as $C_1$, but changing 10% of all probabilities (100 samples) to strictly the opposite of the $y$ true value, essentially creating a strictly worse classifier. This means that at the start of the experimentation, the classifier $C_1$ will always be better performing than the classifier $C_2$.

For each experiment, we perform three different tests, creating the data with an IR of 4.5, 11, and 40, which corresponds to a value near the median of each IR interval previously defined in Section 4.3 (slightly, moderately, and heavily imbalanced). The noise level is changed gradually throughout these tests and we observe which classifier ($C_1$ or $C_2$) achieves a better score for each performance metric. A score of 1 is added if the metric selects the classifier $C_2$ rather than $C_1$, a score of 0 for the opposite, and a score of 0.5 in case both are equal. Each of those tests is performed for a total of 1000 trials, where a new dataset and classifiers are created each time. All trials are then averaged together, which ensures the convergence of the results.

We performed several experiments in order to test the impact of misclassification and probability noises on the performance metrics. As defined in [76], we used two types of noise to be tested, as described below, however, our experiments differ in several ways, as we focus specifically on performance metrics applicable in imbalanced scenarios and consider the IR of the datasets for each experiment.

- **Misclassification Noise**: By applying noise to the true labels by either changing its value to its opposite or randomly, we can evaluate the sensitivity of the metrics to changes in the true labels.
- **Probability Noise**: By applying noise directly to the probabilities of the classifier, we measure the sensitivity of the metric to changes in model probabilities.

### 4.4.2 Results

#### Misclassification Noise

Misclassification noise is defined as noise affecting the data classes directly. In practice, this could be represented by noise in the deployment context of a model or labeling errors for example. Using a random value from a normal distribution, we affect the $y$ true values of our testing scenario we have presented in the previous section, from a range of 0%, where no $y$ values are changed, to 100%, where all the values would be changed randomly. Consequently, we expect that there would no longer be a distinction of performance between the two classifiers, $C_1$ and $C_2$, as the noise reaches 100% since all labels will now be randomly chosen.

Figures 4.8, 4.9, and 4.10 display the results for each imbalance group, slightly (IR = 4.5), moderately (IR = 11), and heavily imbalanced (IR = 40), respectively. While the

x-axis represents the percentage of noise applied to the data as previously mentioned, the y-axis represents the percentage of times classifier $C_2$ was "misclassified" as the better classifier rather than classifier $C_1$ by a given metric, which we will refer to as the **misclassification rate** from now on.

As we expected, all metrics start with a misclassification rate of 0, as there is no noise added to the data yet. As more and more noise is added, we expect the metrics to trend towards a rate of 0.5 since there would be no real distinction between the two classifiers considering all the class labels are getting randomized. In that regard, most metrics do follow that hypothesis, for all the imbalance groups. However, two metrics hold our attention as they do not follow this same pattern. Indeed, $F_1$ score and G-Mean present a distinct behaviour, as both of them not only misclassify the classifiers at a lower noise amount but also trend towards classifying classifier $C_2$ as the best one when the dataset is comprised only of random $y$ values. This observation signifies that both of these metrics do represent a lesser resilience to misclassification noise in general. Moreover, we also note the misclassification rate of PRG, which also represents a slightly higher sensitivity to this type of noise than its counterparts, albeit smaller than $F_1$ score and G-Mean.

Now focusing our attention on class imbalance, we note the heavy effect that it has on the misclassification rate. Undoubtedly, the increase in class imbalance causes most metrics to be less robust to misclassification noise, which make sense since any changes affecting the small number of samples of the minority class would be seen in the overall performance of the model. An interesting observation, however, is that both Brier loss and Log loss seem to be largely unaffected by the class imbalance. This can be explained by the fact that both metrics are using the predicted probabilities of the model rather than the classification prediction errors which would not be as affected by further imbalance in the data.

Finally, we also wanted to note the disparity of our results with the experiments performed by Ferri et al. [76]. Indeed, in their case, all metrics performed much more similarly, reaching an average of 50% misclassification rate at 100% noise. To validate our results, we carried out an experiment similar to our previous one but with all datasets balanced (IR = 1). When performing our experiments on balanced data, we are able to confirm the results obtained in [76] as we observe similar trends which can be seen in Figure 4.11. However, we can observe that in our results, the misclassification rate only starts increasing at around 80% noise which is much later than the results shown in [76]. This is explained by the fact that we made our classifier $C_2$ strictly worse than $C_1$ at the beginning, by changing the predictions to their opposite rather than randomly like it was done in the aforementioned study, which therefore needs a higher percentage of random noise in order for $C_2$ to outperform $C_1$. Compared to the balanced scenario, the imbalance cases showed that all

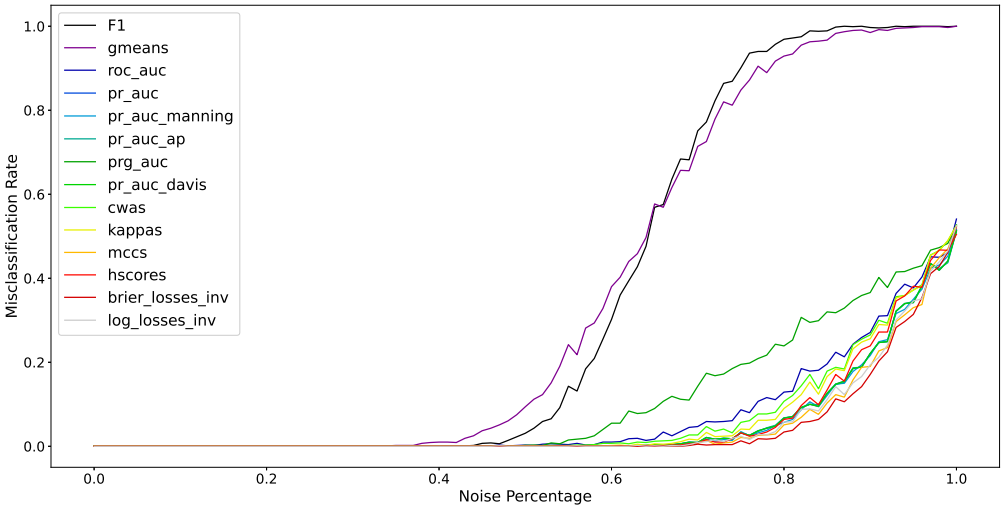



Figure 4.8: Performance metrics robustness to misclassification noise on slightly imbalanced data (IR=4.5)

metrics are actually much more sensitive to misclassification noise when data imbalance is present and that metrics such as $F_1$ and G-Mean should be avoided in these scenarios, as they present a much higher sensitivity even at low imbalance levels. These results show the high impact that class imbalance has on all performance metrics and highlights the importance of performing these experiments specifically on imbalanced data to provide insights for users in this domain.

### Probability Noise

In this section, rather than evaluating noise on the class values, we are evaluating the impact of noise on the probabilities from the classifiers directly. In other words, these experiments test the resilience of the performance metrics to poor prediction probabilities given by the classifier. To test the impact of this type of noise on our performance metrics, we first start by generating a value $\alpha$, from a uniform distribution ranging from $[-\beta,\beta)$ where the value of $\beta$ depends on the amount of noise being added and ranges from 0 to 0.5, depending on the percentage of noise, where a value of 0 represents 0% of noise and 0.5, 100%. The value of $\alpha$ will then be added uniformly to all the probabilities of classifier

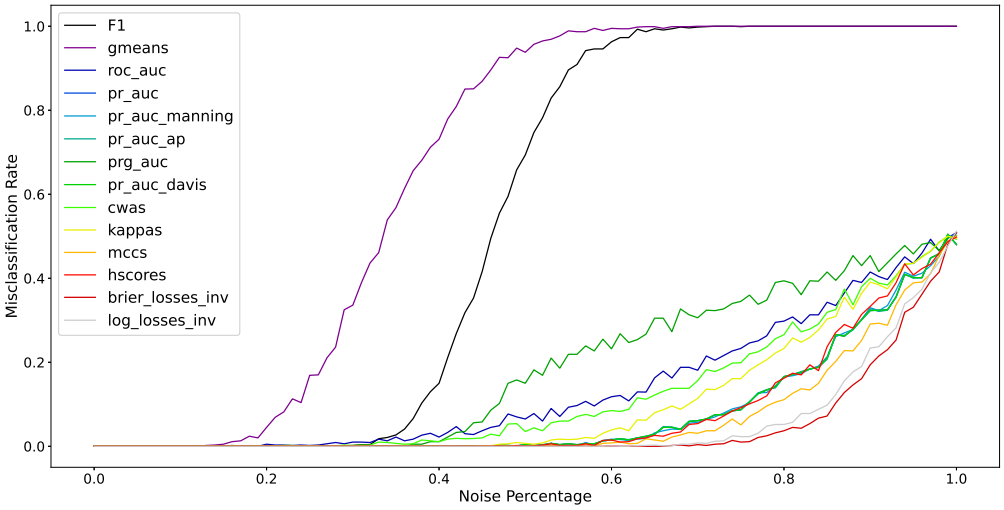



Figure 4.9: Performance metrics robustness to misclassification noise on moderately imbalanced data (IR=11)

$C_1$, while the probabilities of classifier $C_2$ will remain the same. In cases where the value of a given changed probability would reach a value outside of the range [0,1], the value is pinned to the corresponding minimum or maximum instead.

The results of this set of experiments can be observed in Figures 4.12, 4.13, and 4.14 for the slightly, moderately and highly imbalanced data, respectively. In all cases, we observe that there are multiple metrics that seem to be unaffected by the introduction of probability noise. Indeed, we note that all the ranking metrics tested have largely not been influenced by the probability noise, while threshold and probabilistic metrics such as G-Mean, CWA, log loss, and others are significantly affected. This behaviour can be explained by the fact that affecting probabilities in such a way that all probabilities are changed by a common value, does not generally affect their rankings, which is why ranking metrics are largely unaffected. On the other hand, since threshold metrics are based on the classification prediction errors and probabilistic metrics on the probabilities themselves, both types of metrics are heavily affected by the probability noise which ultimately makes the difference, or boundary between both predicted classes harder to distinguish.

Looking at the difference between imbalance groups, we can see similar behaviour to what we observed previously with misclassification noise in Figures 4.8 to 4.10, where

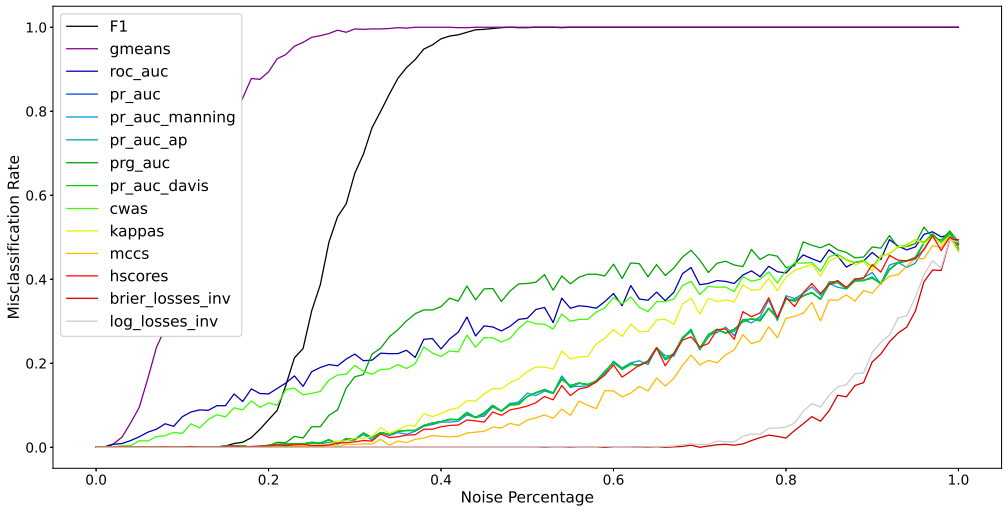



Figure 4.10: Performance metrics robustness to misclassification noise on heavily imbalanced data (IR=40)

most metrics are more sensitive to the noise on higher IRs. Indeed, this effect is also present here, albeit to a lesser degree, as we can see that the metrics that were affected by the noise in lower imbalance scenarios start to be affected at lower levels of noise in the higher imbalance cases which restates one of the numerous difficulties when dealing with imbalanced data.

### 4.4.3 Summary

This section summarizes the multiple conclusions reached in Section 4.4, as we evaluated the impact of misclassification and probability noise on the metrics at different imbalance levels.

- $F_1$ score and G-Mean showed the least resilience to misclassification noise on imbalanced data. While lesser than the two aforementioned metrics, PRG AUC also performed worse with the introduction of misclassification noise than the other evaluated ranking metrics.

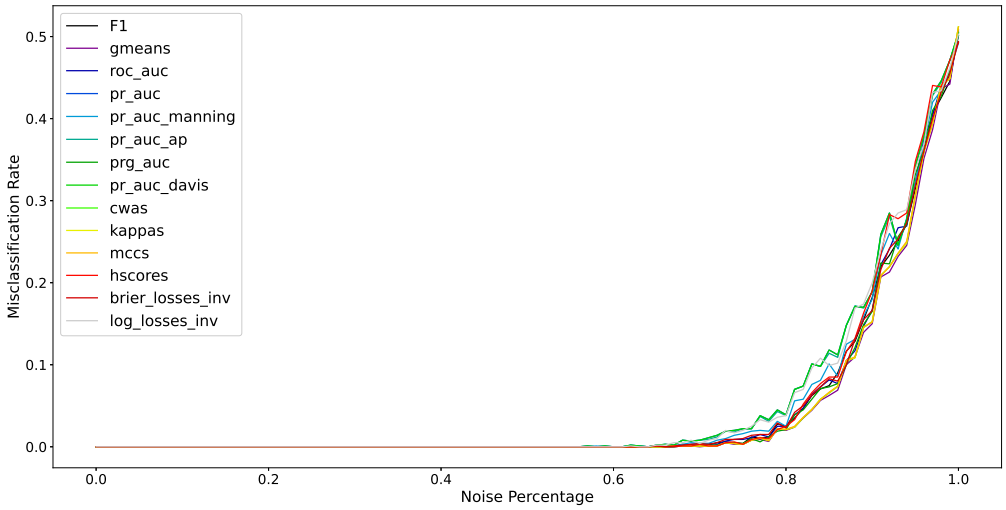



Figure 4.11: Performance metrics robustness to misclassification noise on balanced data (IR=1)

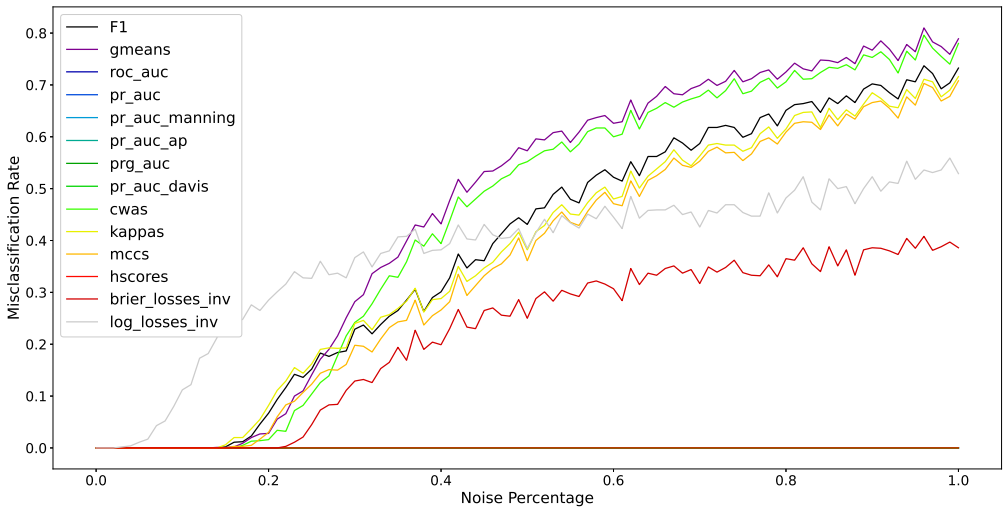



Figure 4.12: Performance metrics robustness to probability noise with IR=4.5

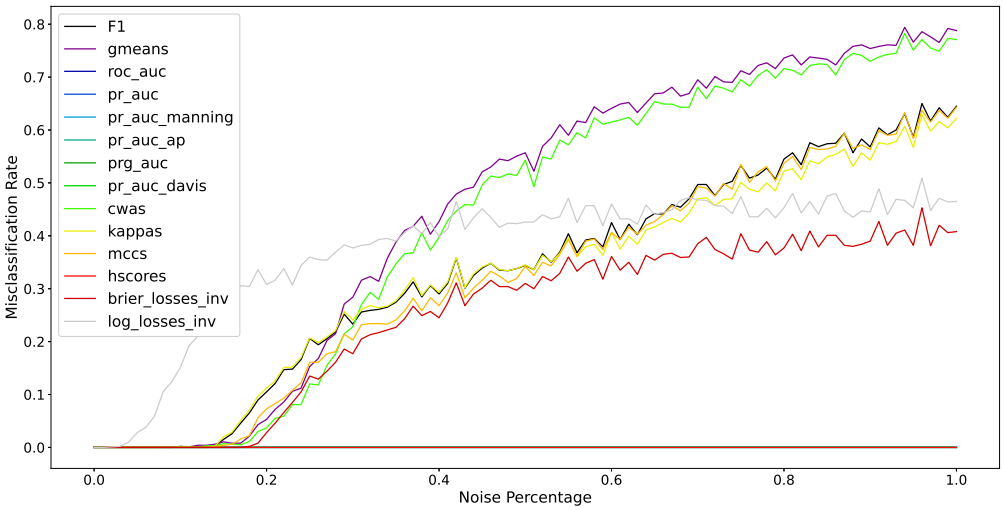



Figure 4.13: Performance metrics robustness to probability noise with IR=11

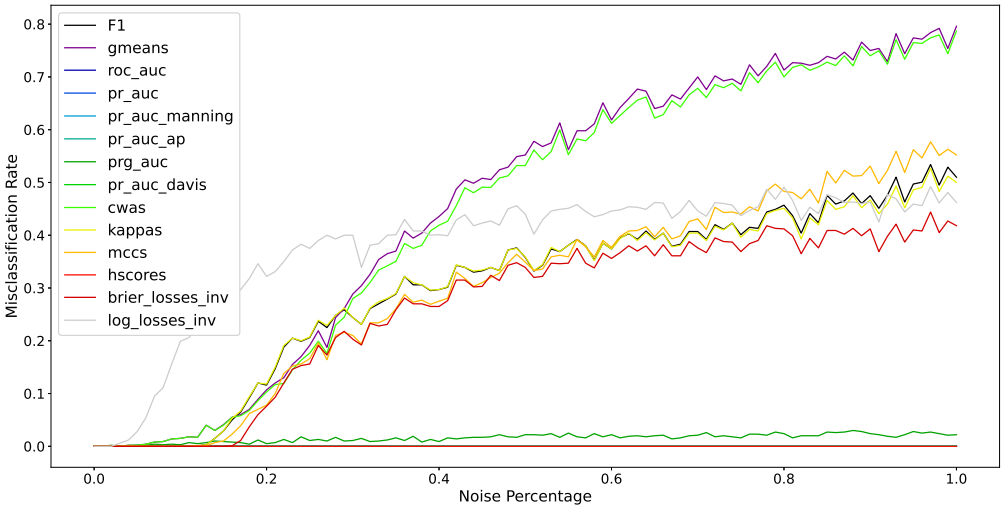



Figure 4.14: Performance metrics robustness to probability noise with IR=40

- The metrics' sensitivity to misclassification noise was demonstrated to be largely driven by class imbalance, as it not only increased with class imbalance but was also much less present on balanced data.
- Ranking metrics proved to be largely unaffected by noise on the model's probabilities (probability noise), while threshold and probabilistic metrics were affected to a much greater extent.
- While class imbalance did not play as much of a role in the sensitivity of the metrics to probability noise compared to misclassification noise, it did increase it slightly on metrics that already showed sensitivity to this type of noise in other scenarios with lesser imbalance.

## 4.5 Summary and Recommendations

Throughout this study, we analyzed multiple performance metrics widely regarded as the given metrics to use when dealing with imbalanced domains. Using the knowledge acquired through our experiments and the conclusions reached, we devised the data flow diagram shown in Figure 4.15 with the goal of helping future researchers, computer scientists, and end-users to choose an appropriate metric depending on the specific context of their applications and use cases. It is important to note that while the following diagram suggests one or a few metrics for each specific scenario, these are merely intended as a starting point and should therefore be treated as such. Moreover, we highly recommend the use of multiple metrics, notably ones that come from different types (threshold, ranking, and probabilistic) to validate the evaluation of a model, since they can present very different behaviour, as we have seen in Section 4.3.

The recommendations given in the diagram follow the conclusions we have previously reached and discussed which can be summarized as follows:

1. **General Conclusions**
   (a) The three different types of metrics (threshold, ranking, and probabilistic) exhibit very different behaviour and their usage should be complementary, especially when dealing with highly imbalanced data.
   (b) Threshold and ranking metrics are much more closely related to each other than either of them are to probabilistic metrics. This indicates that probabilistic metrics are a class apart from the two other ones, which is why we only

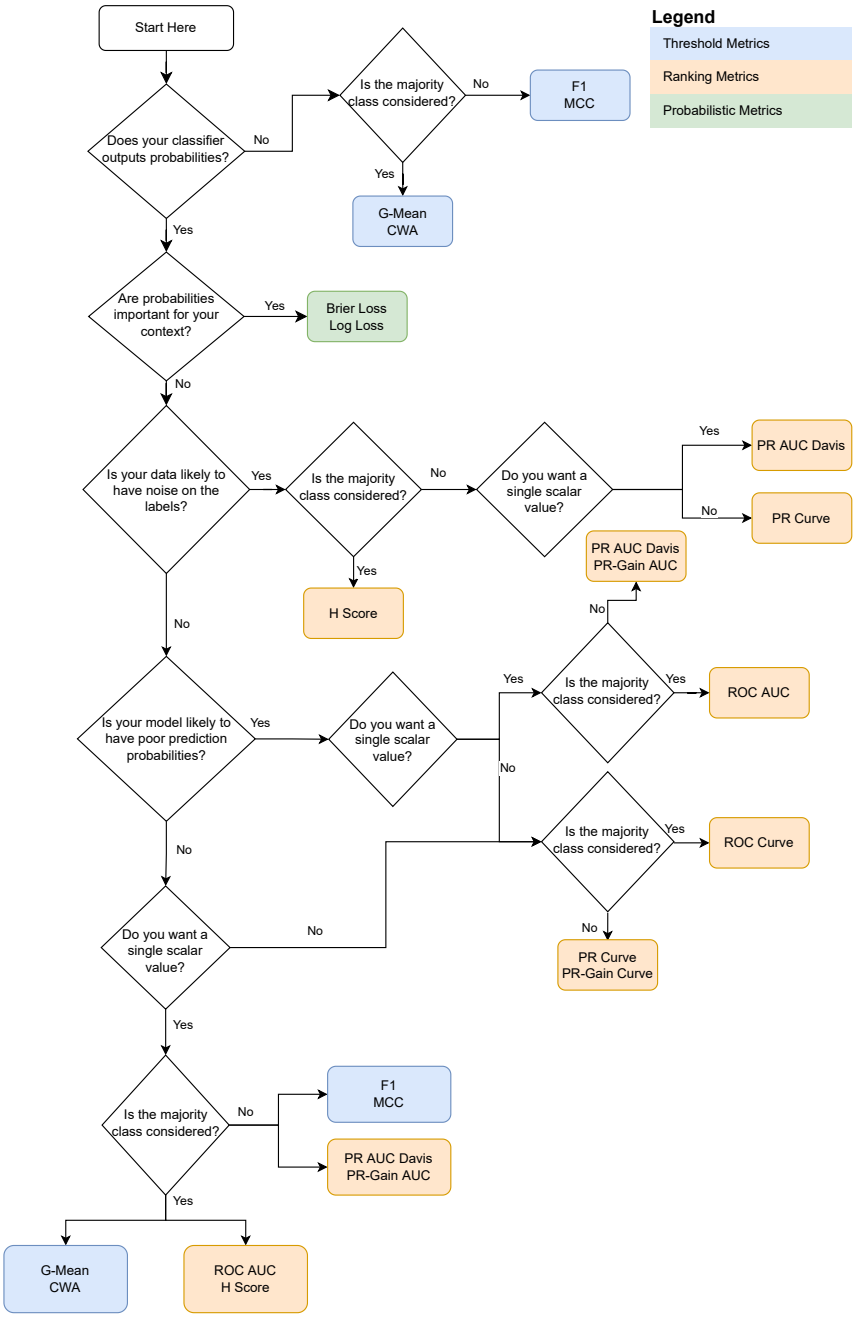



Figure 4.15: Data flow diagram representing recommendations for performance metrics choice in imbalanced domains.

recommended them for specific scenarios where the probabilities of the model are important in the diagram.

(c) In situations where MCC and kappa are applicable, we have recommended the use of MCC over kappa since our results presented that both metrics were mostly similar but, as we have discussed, kappa does display some unwanted properties when it is used in imbalanced domains.

(d) Davis' interpolation of the PR AUC is recommended when a scalar value representing the PR curve is wanted, as we have shown that it was the most consistent with the other interpolations of the PR AUC.

(e) In situations where both the PR curve, PRG curve, or their AUCs are applicable, we recommended the use of both or either, since we have shown that they present different characteristics and can lead to different ranking results.

2. **Conclusions Related to Noise**

(a) $F_1$, G-Mean, and PRG are not recommended when the data is likely to have noise on the labels (misclassification noise) since it was demonstrated that they display a lesser resilience to this type of noise.

(b) We showed that most metrics present a higher sensitivity to misclassification noise on a higher imbalance of the data. We would especially recommend avoiding the aforementioned $F_1$, G-Mean, and PRG in these scenarios. Moreover, we would strongly recommend combining multiple other metrics to ensure a reliable performance evaluation.

(c) When dealing with models likely to have poor prediction probabilities (probability noise), we do not recommend the use of threshold or probabilistic metrics, as they displayed a higher sensitivity to this type of noise at all imbalance levels. We instead recommended ranking metrics, or their AUC when a scalar metric is needed.

The work in this chapter makes an important contribution by experimentally and systematically reviewing performance metrics appropriate for imbalanced domain classification. We studied the impact of choosing one metric over another for model evaluation, as well as the resilience of these different metrics to noise, and provided future researchers with easy-to-follow recommendations for selecting performance metrics in their projects. This serves as a strong first step toward the correct evaluation of models in imbalanced binary classification and, since such problems are also common in ND domains, lays the groundwork for the next chapter, which explores the proper evaluation of ND algorithms in the context of DSs.

# Chapter 5

# Prioritizing the Essential: A Robust Evaluation Framework for Novelty Detection

This chapter addresses the second objective of this thesis mentioned in Chapter 1, namely the proposal of an updated and standardized evaluation framework for ND algorithms that accounts for key data characteristics of DSs. As established in Chapter 4, evaluating performance in imbalanced scenarios is already a significant challenge. In this regard, evaluating ND in DSs is even more difficult, as it presents unique complexities that will be outlined in this chapter. To address these issues, this chapter formalizes a comprehensive evaluation framework and proposes a set of existing and novel metrics to assess the numerous distinct aspects of ND.

The contents of this chapter are based on the following paper:

Jean-Gabriel Gaudreault and Paula Branco. Prioritizing the essential: A robust evaluation framework for novelty detection. *Machine Learning*, 114(6):143, Apr 2025

## 5.1 Introduction

Considering the large quantity of data continually generated by several data sources, such as IoT devices, sensors, network activity, and others, it is common to depict these sources as continuous streams of data, denoted as DSs, rather than static datasets. However, these data sources exhibit distinct unique characteristics, including being unbounded, with

their data distribution evolving over time, and novel classes or concepts may emerge or disappear within the stream [152,190]. As such, classification algorithms deployed in those scenarios must be versed in handling these novelties and adapting to the shifting data distributions [82].

ND algorithms, which aim to identify and learn previously unknown classes within a DS, emerge as a solution to address these challenges. However, as the field garners increasing attention within the research community, establishing reliable and accurate methods for evaluating and comparing the performance of these algorithms becomes crucial. The evaluation of ND algorithms currently suffers from some limitations, particularly the use of binary classification metrics for multi-class data, leading to misleading performance assessments, and the proposal of non-comparable metrics across datasets, hindering generalizability and performance comparisons [55,61,92]. Additionally, current evaluation practices often omit certain performance factors, such as the time required to detect or classify a novel class, a critical metric in real-world applications, and often fail to report on key data characteristics that significantly impact algorithm performance. As such, this study proposes a comprehensive evaluation framework for ND algorithms which aims to address the identified inconsistencies and provide an accessible evaluation structure to facilitate the comparison and a fair assessment of these algorithms.

In previous work [90], we presented the main drawbacks of existing ND evaluation methodologies and laid the groundwork for an evaluation framework mitigating these issues. Specifically, we showed how some data characteristics could potentially impact the performance of ND algorithms and suggested the use of novel metrics to better evaluate both the classification and temporal performance of these algorithms. However, this work was limited in scope, as we wanted to first confirm our hypotheses that DSs' characteristics can influence the performance evaluation of ND algorithms in a significant way and that the metrics proposed were coherent with other evaluation techniques. Hence, this work broadens the scope of our research in two significant aspects by (i) formalizing an evaluation framework and the evaluation problem in a way that is easy to follow for practitioners and (ii) extending the previous framework with the inclusion of other characteristics and a new metric that provides additional insights. To this end, this chapter summarizes and extends our previous research, by including a novel data characteristic, the concept sparsity, as well as a novel temporal metric allowing us to follow the time required to classify each class over time, an aspect of DSs formerly not considered at all by other evaluation techniques. Moreover, we extended previous experiments with 22 additional datasets and three supplementary algorithms for a total of 24 datasets and four algorithms. This extension enabled a more comprehensive analysis of the impact of data characteristics on multiple different data types, as well as a broader comparison of the proposed metrics compared to

previously suggested ones.

**Contributions** The main contributions of this chapter can be summarized as follows:

(i) formalize a comprehensive and accessible evaluation framework that enables fair assessment, ensures repeatability, and facilitates comparison between algorithms;

(ii) introduce and empirically show the impact of different data characteristics of DSs, as well as demonstrate how they can be reported when building a DS;

(iii) propose novel metrics that incorporate the temporal aspect of DSs within the evaluation, an important aspect that is not considered in former evaluation frameworks.

**Organization** The rest of this chapter will be structured as follows: Section 5.2 reviews current methods used for the evaluation of ND; Section 5.3 formalizes the proposed framework and breaks it down into three key aspects; Section 5.4 empirically presents the usage of the proposed framework and compares it to other existing metrics; and Section 5.5 summarizes the findings and provides concluding remarks.

## 5.2 Evaluation of Novelty Detection

The evaluation of ND algorithms has historically been quite inconsistent in the literature, with very few recent studies assessing and proposing evaluation frameworks to properly evaluate these algorithms [55,90]. Furthermore, the evaluation of these algorithms will vary depending on whether the algorithm considers the ND task as: (i) only one novel class; (ii) multiple novel classes, appearing one at a time; or (iii) multiple novel classes which can appear at the same time [55]. However, despite considering the ND task as one of the two latter categories, some algorithms (e.g., [13, 113, 130]) use metrics that would only be appropriate for the former scenario, as they do not consider the misclassification between novel classes. In the following sections, we will delve into details on typical evaluation methods for ND that were defined for both the binary and multi-class scenarios.

### 5.2.1 Binary Scenario

In the context of a binary scenario, where the goal is only to distinguish novel classes from known ones, several metrics have been defined in the literature. To review these metrics, we

first recall the following notation, introduced in Chapter 3, which will be used throughout this chapter:

- TP: Number of detected novelties correctly classified;
- FP: Number of known class samples wrongly classified as novelties;
- FN: Number of novelties wrongly classified as known;
- TN: Number of known class samples correctly classified;
- $N$: Total number of samples;
- $N_c$: Number of novel samples in the stream; and
- FE: Existing class instances misclassified (other than FP)

Using these definitions, we can devise a confusion matrix (cf. Table 5.1) for the binary ND problem where each new NP detected will add a new column, resulting in a rectangular confusion matrix, as it was first proposed by Faria et al. [55]. In this table, $C_i$ represents the class $i$, $\text{NP}_i$ the $i$th NP discovered by the model, and $k$ is the number of known classes, which is lower than the total number of classes $m$ ($k < m$).

It is important to note that while there can be multiple known classes or multiple novel classes, as shown in the confusion matrix, we still refer to this problem as binary, as there are no distinctions between multiple novel classes, i.e. all accurate or inaccurate labeling of novel classes are grouped within one class (FP, FN, or TP). As we will further discuss in the following sections, employing binary evaluation metrics when there are multiple novel classes may not be optimal, as they will not compute the misclassification between novel classes and therefore not accurately capture the performance of the employed algorithms.

With the above definitions of TP, FP, FN, and TN, most metrics commonly used in binary classification for batch learning could, in theory, be applied to the binary ND problem. However, since ND problems often exhibit class imbalance, with typically far more samples from known classes than from novelties, some metrics may not be suitable for evaluation, as discussed in the previous chapter [92]. In the following sections, we describe the metrics that have historically been used to evaluate the binary ND problem.

Table 5.1: Confusion matrix of a binary novelty detection problem

| | | | **Predicted Values** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Known Classes | | | | Novelty Patterns | | | | |
| | | | $C_0$ | $C_1$ | ... | $C_k$ | $\text{NP}_1$ | $\text{NP}_2$ | ... | $\text{NP}_j$ | $Unk$ |
| **Actual Values** | Known Classes | $C_0$ | TN | FE | FE | FE | FP | | | | |
| | | $C_1$ | FE | TN | FE | FE | | | | | |
| | | ... | ... | | | | | | | | |
| | | $C_k$ | FE | FE | FE | TN | | | | | |
| | Novel Classes | $C_{k+1}$ | FN | | | | TP | | | | |
| | | ... | | | | | | | | | |
| | | $C_m$ | | | | | | | | | |

### Accuracy and Error Rate

Accuracy (cf. Equation (5.1)) and its complement the error rate (*Err*) (cf. Equation (5.2)), are widely used in batch learning scenarios and have also been applied to the ND case. An interesting property of the error rate is that it considers the misclassification between known classes (FE) which might be desirable depending on the context. However, as mentioned previously, since both metrics do not consider the misclassification between the novel classes, they are classified as binary metrics.

Moreover, it is important to emphasize that accuracy and error rate have been shown to be inadequate in imbalanced scenarios, as the majority class disproportionately influences these measures compared to the minority class [32]. As mentioned in the previous section, ND often exhibits such scenarios, where the known classes contain significantly more samples than the novel classes. Their use should therefore be considered carefully depending on the context.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N} \qquad (5.1) \qquad\qquad Err = \frac{(\text{FP} + \text{FN} + \text{FE})}{N} \qquad (5.2)$$

### $F_\beta$ Measure

The $F_\beta$ measure is a well-established metric that we examined in the previous chapter in the context of imbalanced domains. It has also been used quite extensively in the ND domain (e.g. [3, 29, 47, 48, 160]). However, since the positive class corresponds to the novel

classes in the stream, this metric exclusively evaluates whether or not the algorithm can correctly identify novel samples. This behaviour may be adequate in certain domains where detecting the novelties is typically solely the aim of the algorithm, such as intrusion detection in cybersecurity or fraud detection, but might not be suitable in domains where classifying correctly the known classes is also relevant.

#### $M_{new}$ and $F_{new}$ Measures

$M_{\text{new}}$ (cf. Equation (5.3)) and $F_{\text{new}}$ (cf. Equation (5.4)) are metrics that have been defined and used extensively for the binary ND problem [117, 200]. They, respectively, represent the percentage of novel class samples misclassified as a known class and the percentage of known class samples misclassified as a novel class [152].

While these two metrics specifically evaluate the ND task, as they compute the misclassification of novel classes into known ones as well as the opposite, they also present several limitations: (i) as seen in Table 5.1, they will consider samples labeled as unknown as correctly predicted (TP) for unknown classes and as incorrect (FP) for known classes [55]; (ii) they necessitate the tracking of two individual metrics; and (iii) they do not compute the misclassification between multiple known classes.

$$M_{new} = \frac{\text{FN}}{N_c} \cdot 100 \qquad (5.3)$$

$$F_{new} = \frac{\text{FP}}{N - N_c} \cdot 100 \qquad (5.4)$$

### 5.2.2 Multi-class Scenario

Considering the limitations of the binary scenario discussed in the previous section, it is possible to consider the ND problem as a multi-class scenario, by considering the error of each class independently from each other. Doing so results in the confusion matrix seen in Table 5.2, where $\text{TP}_A$ represents the number of TP in class $A$ (i.e. the number of samples from class $A$ correctly classified as such) and $E_{A,B}$ represents the number of samples misclassified in class $B$ that were actually from class $A$, as defined by Tharwat [210]. Using this notation, we can easily compute the FN and FP for any known class $A$, as shown respectively in Equations 5.5 and 5.6, for $m$ total classes and $J$ NPs.

It should be pointed out that the bottom right section of this confusion matrix, which represents novel classes detected as NPs, is not filled ($\text{TP}_A$ and $E_{A,B}$). This is because the NPs detected by the model cannot directly be linked to the true class labels of the novel samples, as they might have been discovered in an unsupervised way, and there might be

multiple NPs representing a single novel class. Faria et al. [55] propose to take the most frequent true label in each NP, to associate each of them with a true label, which would allow the confusion matrix to be completed with the missing terms. The metrics presented in this section were all proposed in the same framework [55], which combined with our previous work [90], is to the best of our knowledge, the only framework addressing the proper evaluation of multi-class in ND. Hence, the rest of this section assumes the use of this technique to associate NPs with true class labels. However, our proposed framework, presented in Section 5.3, does not require this assumption.

$$\mathrm{FN}_A = \sum_{i \neq A}^{k} E_{A,i} + \sum_{j=1}^{J} E_{A,\mathrm{NP}_j} \tag{5.5}$$

$$\mathrm{FP}_A = \sum_{i \neq A}^{m} E_{i,A} \tag{5.6}$$

Table 5.2: Confusion matrix of a multi-class novelty detection problem

| | | | **Predicted Values** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Known Classes | | | | Novelty Patterns | | | | |
| | | | $C_0$ | $C_1$ | ... | $C_k$ | $\mathrm{NP}_1$ | $\mathrm{NP}_2$ | ... | $\mathrm{NP}_j$ | $Unk$ |
| **Actual Values** | Known Classes | $C_0$ | $\mathrm{TP}_0$ | $E_{0,1}$ | ... | $E_{0,k}$ | $E_{0,\mathrm{NP}_1}$ | $E_{0,\mathrm{NP}_2}$ | ... | $E_{0,\mathrm{NP}_j}$ | $Unk_0$ |
| | | $C_1$ | $E_{1,0}$ | $\mathrm{TP}_1$ | ... | $E_{1,k}$ | $E_{1,\mathrm{NP}_1}$ | $E_{1,\mathrm{NP}_2}$ | ... | $E_{1,\mathrm{NP}_j}$ | $Unk_1$ |
| | | ... | ... | | | | ... | | | | |
| | | $C_k$ | $E_{k,0}$ | $E_{k,1}$ | ... | $\mathrm{TP}_k$ | $E_{k,\mathrm{NP}_1}$ | $E_{k,\mathrm{NP}_2}$ | ... | $E_{k,\mathrm{NP}_j}$ | $Unk_k$ |
| | Novel Classes | $C_{k+1}$ | $E_{k+1,0}$ | $E_{k+1,1}$ | ... | $E_{k+1,k}$ | ? | ? | ... | ? | $Unk_{k+1}$ |
| | | ... | ... | | | | ... | | | | |
| | | $C_m$ | $E_{m,0}$ | $E_{m,1}$ | ... | $E_{m,k}$ | ? | ? | ... | ? | $Unk_m$ |

### Combined Error

The Combined Error (CER) (cf. Equation (5.8)), is a commonly used measure for traditional multi-class classification in batch learning. It combines and can be defined as the

weighted average of the FPR and FNR (cf. Equation (5.7)). It is worth noting that, as defined in [55], it does not take into consideration the samples predicted as unknown ($Unk_A$) in its computation.

$$\mathrm{FPR}_i = \frac{\mathrm{FP}_i}{\mathrm{FP}_i + \mathrm{TN}_i} \quad \mathrm{FNR}_i = \frac{\mathrm{FN}_i}{\mathrm{FN}_i + \mathrm{TP}_i} \tag{5.7}$$

$$\mathrm{CER} = \frac{1}{2} \sum_{i=1}^{m} \frac{\#ExC_i}{\#Ex} (\mathrm{FPR}_i + \mathrm{FNR}_i) \tag{5.8}$$

where $\#ExC_i$ represents the number of samples of class $C_i$, $\#Ex$ the total number of samples, and $m$ the total number of classes.

### Akaike Information Criterion

An important characteristic to take into account when evaluating multi-class scenarios is the complexity of the model. Indeed, when using the technique previously mentioned to assign each NP to a class label, a model could assign each identified novel sample to its own NP and therefore have no misclassification error between novel classes, as long as it can discern them from known samples. However, having a multitude of small, independent clusters is undesirable, hence, Faria et al. [55] propose an adaptation of the Akaike Information Criterion (AIC) (cf. Equation (5.9)) to compute the complexity of the model, where a higher value implies a more complex model.

$$\mathrm{AIC} = -2 \ln(1 - \mathrm{CER}) + \frac{2p}{\ln(N)} \tag{5.9}$$

where $p$ is the number of classes detected by the ND algorithm (NPs and known classes) and $N$ is the total number of samples without those labeled as unknown.

### Unknown Rate

As a solution to the problem of having samples labeled as unknown being computed as correctly classified novelties, as previously mentioned in the binary scenario, the authors propose to exclude them from the computation of the other metrics (CER and AIC) and

instead compute the Unknown Rate ($Unk$) (cf. Equation (5.10)) which represents the rate of samples classified as unknown, or the percentage of samples in the buffer prior to the creation of a NP. This metric is useful in specific scenarios where granularity is important, such as when diagnosing which specific classes are being properly detected, and therefore we do not change it in this study.

$$Unk = \frac{1}{m} \sum_{i=1}^{m} \frac{\#Unk_i}{\#ExC_i} \tag{5.10}$$

where $\#Unk_i$ represents the number of samples of class $i$ classified as unknown, $\#ExC_i$ the total number of samples of class $i$, and $m$ the total number of classes.

### 5.2.3 Summary

In this section, we presented the metrics that have been historically used to evaluate ND algorithms in both binary and multi-class scenarios. We demonstrated a few problems with some of these metrics in correctly assessing the performance depending on the context, as well as a lack of a common framework used when evaluating these algorithms. In the next section, we will propose a comprehensive framework to mitigate these issues and allow for a better comparison between algorithms.

## 5.3 Evaluation Framework

In this section, we present an extension of the evaluation framework we first proposed [90]. We propose this framework with the aim of solving some of the limitations currently present in the evaluation of ND in the literature and to unify the assessment of these algorithms.

### 5.3.1 Data Stream Characteristics

First of all, we advocate in our proposed framework for the reporting of the data characteristics used when evaluating a model in ND scenarios. Just like the data used to train a model or the ratio between the train and test set in batch learning, these types of settings can greatly influence the performance of the algorithm tested, as we will empirically demonstrate in a later section. This problem is exacerbated in the context of ND, as other

parameters such as which classes are used as the known/unknown classes and the order of the samples come into play. Moreover, as most datasets used in the field are not inherently DSs but rather converted from batch datasets, the way these datasets are converted relies, more often than not, on the researchers performing the evaluation, which can vary vastly and lead to over-optimistic performance or produce biased evaluation without proper reporting of these characteristics. Considering this issue and based on our experimental analysis, we have curated a list of four main data characteristics that should be reported on in our framework.

- **Time Between Novel Classes ($t$)** This characteristic corresponds to the time in between the appearance of each new class in the DS. A time of 0 ($t = 0$) coincides with no time between the arrival of the classes, hence the samples of multiple novel classes can appear one after the other.
- **Ratio of Offline Samples ($r$)** Similar to the train/test ratio in batch learning, this characteristic represents how much of the original dataset is used for the initial training of the algorithm in the offline phase. It ranges from 0 to 1, representing a percentage of the total number of known class samples available.
- **Ratio of Known Classes ($n$)** This aspect depicts how many classes within the dataset are presented to the algorithm during its offline phase, it ranges from 0 to 1, representing a percentage of the total number of classes. It is important to note that not only the ratio needs to be reported for this characteristic, but also which specific classes were considered, as some classes might be easier to detect when considered known and harder when considered novel, or vice versa.
- **Concept Sparsity ($s$)** Lastly, this characteristic refers to the temporal density of samples from the same class in the DS, in other words, whether instances of a given class arrive in contiguous blocks or are sparsely interleaved with other classes. We use $s$ to control this effect. When $s = 1$, all classes are selected with equal probability, resulting in a random selection. A value of $s > 1$ means that the class that appeared most recently has $s$ times greater weight than any other class, hence increasing the probability that the last class will reappear consecutively, while a value of $s < 1$ reduces the likelihood that the same class appears again in succession. As most ND algorithms implement the idea of a buffer and will remove samples that remain in it for too long, an algorithm may more easily identify a novel class when samples from that class arrive all at once, in a concentrated manner. Conversely, a novel class that is very sparsely distributed throughout the stream may be more difficult to detect.

### 5.3.2 Classification Evaluation

In this section, we review the metrics we first presented in our previous work [90] and that we propose to use for the evaluation of the classification performance of algorithms in our proposed framework. We argue that these metrics offer numerous advantages compared to others that we have reviewed in Section 5.2, as we will discuss in the coming sections.

**Binary Scenario** For the binary scenario, where the aim is to exclusively detect whether a sample belongs to a known class or a novel class, we suggest the use of $M_\beta$ (cf. Equation (5.11)), which refers to the harmonic mean of the complement of both $M_{\text{new}}$ and $F_{\text{new}}$. We believe it presents several advantages compared to other metrics, as it: (i) evaluates both misclassifications in novel classes and known classes, compared to $F_\beta$, for example, which focuses primarily on novel samples; (ii) can be adjusted with the value of $\beta$ to give more weight to either correctly classifying the novel classes ($\beta < 1$), or the known classes ($\beta > 1$) depending on the context; (iii) necessitates to only track one metric which allows for easier comparison between models; and (iv) can be used in imbalanced domains, as $M_{\text{new}}$ is computed using only the total number of novel samples in the stream rather than all samples.

$$M_\beta = (1+\beta^2) \cdot \frac{(1 - M_{new}/100) \cdot (1 - F_{new}/100)}{(\beta^2 \cdot (1 - M_{new}/100)) + (1 - F_{new}/100)} \tag{5.11}$$

**Multi-Class Scenario** For the multi-class scenario, where there is a need to evaluate the inter-misclassification between novel classes and known classes, we suggest the use of the AMI. Commonly used in the unsupervised domain, the use of this metric allows us to avoid the problem of mapping the NPs to a true label which we discussed in Section 5.2.2. Moreover, it presents numerous advantages compared to the use of CER and AIC which we discussed in previous work [90]. Specifically, it: (i) is adjusted for chance and will provide a value close to 0 for a random clustering and (ii) is bounded (between 0 and 1) which provides a baseline and allows for comparison between different datasets whereas AIC is not; (iii) combines both the evaluation of the classification and model's complexity in one metric rather than two; and (iv) is adapted to use in imbalanced scenarios [177].

$$\text{AMI}(U, V) = \frac{MI - E[MI]}{mean(H(U), H(V)) - E[MI]} \tag{5.12}$$

where $U$ and $V$ are two clusters (partitions), $MI$ is the mutual information of those partitions, $E[MI]$ is the expected mutual information, and $H(x)$ is the entropy of $x$.

### 5.3.3 Temporal Aspect

The last facet of our proposed evaluation framework is the evaluation of the performance related to the temporal aspect of the DS. Indeed, as DSs are inherently temporal and numerous ND algorithms will not detect novel classes when they first appear in the stream as they need to gather more data before declaring an NP, the time (or number of samples) needed for the model to identify the novel class is an important aspect to evaluate as in certain contexts, such as cybersecurity or fraud, detecting an NP as soon as possible is paramount. Hence, we introduce two new metrics in our framework, namely the Time To Detection (TTD) (cf. Equation (5.13)) and the Time To Classification (TTC) (cf. Equation (5.15)). The former computes the number of samples of a specific novel class that is necessary for each novel class to be detected as an NP (other than known or unknown), while the latter is an ongoing average, which computes the average number of samples it takes to classify a class as other than unknown. An illustrative example showing how these two metrics are computed on a DS is provided in Figure 5.1.

$$\text{TTD}_i = \sum_{j=\text{TFA}_i}^{\text{TFD}_i} [c_j = i] \tag{5.13}$$

where $\text{TFA}_i$ is the time of first appearance of the novel class $i$, $\text{TFD}_i$ the time when it was first detected as an NP, and $c_j$ is the true class label of the $j$th sample of the stream.

$$t_{n,i} = \sum_{j=\text{TLA}_i}^{\text{TLD}_i} [p_j = \text{Unk}] \tag{5.14}$$

$$\text{TTC}_i = \text{mean}(\{t_{0,i}, ..., t_{n,i}\}) \tag{5.15}$$

where $t_{n,i}$ represents the number of samples needed for class $i$ to be classified the $n$th time it was detected, $\text{TLA}_i$ is the time of last appearance of the class since $t_{n-1,i}$, $\text{TLD}_i$ the last time it was classified as other than unknown, and $p_j$ the predicted class label on the $j$th sample of the stream.

The new proposed metrics based on the time needed to classify an NP are a novel and essential way to evaluate the performance of the algorithms that had not been considered in previous research.

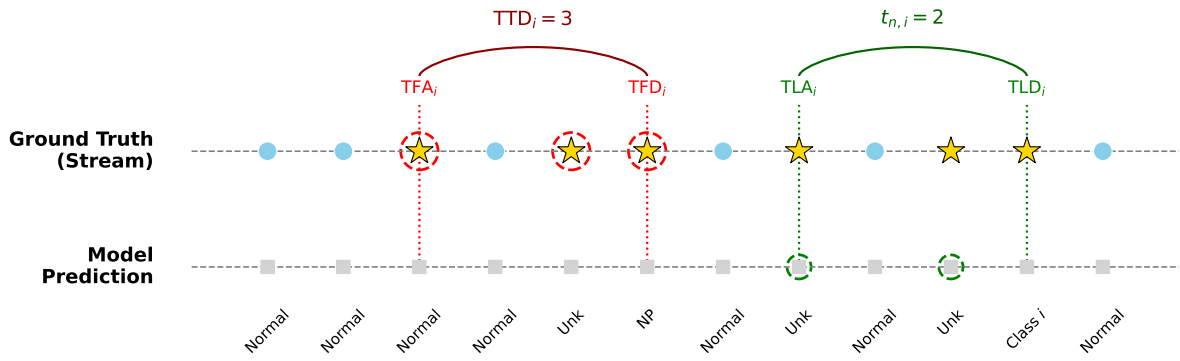



Figure 5.1: Visual representation of the proposed temporal evaluation metrics

## 5.4 Experiments

This section reviews the empirical evaluation of the proposed evaluation framework through the testing of different ND algorithms on multiple datasets, both binary and multi-class. Our objectives with this empirical study are mainly to: (i) show the impact that different DSs characteristics have on the performance and the necessity of reporting them correctly; (ii) compare the classification performance obtained using the metrics proposed in our framework against the previously used metrics; and (iii) demonstrate the utility of the novel metrics we proposed concerning the temporal aspect of DSs. Each of these objectives will be analyzed in its corresponding section below.

The first objective is explored in Section 5.4.2, where we demonstrate how much of an impact each data characteristic has on the overall performance of each algorithm, in both the binary and multi-class settings. The second objective is covered in Section 5.4.2, in which we examine the correlation between our proposed metric, AMI, and previously used metrics like CER, demonstrating that CER often overestimates ND algorithm performance. Finally, Section 5.4.2 addresses our third objective by evaluating our proposed temporal metrics (TTD and TTC) and demonstrates their effectiveness in detecting an algorithm's ability to quickly identify novel classes in real-time scenarios.

### 5.4.1 Methodology

Table 5.3: Overview of the binary datasets characteristics used in the experiments

| Name | No. of features | No. of samples | Type |
|---|---|---|---|
| Adult | 104 | 45222 | Real |
| Credit Card | 30 | 284807 | Real |
| Electricity | 14 | 45312 | Real |
| NOAA | 8 | 18159 | Real |
| Random RBF | 10 | 100000 | Synthetic |
| Random Tree | 12 | 100000 | Synthetic |

Table 5.4: Overview of the multi-class datasets characteristics used in the experiments

| Name | No. of Classes | No. of features | No. of samples | Type |
|---|---|---|---|---|
| Connect4 | 3 | 126 | 67557 | Real |
| Gas Sensor | 6 | 128 | 13910 | Real |
| Nursery | 5 | 27 | 12960 | Real |
| Powersupply | 24 | 2 | 29928 | Real |
| Rialto | 10 | 27 | 82250 | Real |
| Room Occupancy | 4 | 16 | 10129 | Real |
| SensIT | 3 | 100 | 98528 | Real |
| Shuttle | 7 | 8 | 58000 | Real |
| Random RBF | 4 | 10 | 100000 | Synthetic |
| Random Tree | 4 | 12 | 100000 | Synthetic |

The experiments in this section were conducted on a total of six binary datasets and ten multi-class datasets using a mix of real-world and synthetic datasets, as shown in Tables 5.3 and 5.4 for binary and multi-class, respectively. Additionally, all real-world multi-class datasets were binarized using a one-vs-rest method, where the most occurring class was labeled as one class against all other labels, resulting in a total of 24 datasets. The majority of these datasets were taken from the USP DS Repository [198], when available. Four well-known ND algorithms were evaluated, namely MINAS [56], ECHO [114], ECSMiner [147], and ECSMinerWF, an adaptation of ECSMiner without external feedback [56].

The three types of algorithms selected for our testing perform ND in a similar manner. During the offline phase, they first establish an ensemble of models representing the known

classes using a portion of the data. In the online phase, they classify incoming samples, identify outliers in the DS using this ensemble, and temporarily store these outliers in a buffer, where they may eventually be clustered into NPs. The main differences between the three approaches lie in how they handle concept drift and use true labels. ECSMiner trains a new classifier when a certain number of instances are labeled. In contrast, MINAS computes the distance between clusters of outliers and known clusters, updating the known cluster if this distance is below a certain threshold. ECHO detects concept drift by monitoring the classifiers' confidence scores and updates the classifiers when a decrease in confidence is observed. Regarding access to true labels, both ECHO and ECSMiner require labeled instances of the DS after a delay. However, ECHO only needs a limited number of instances based on the classifiers confidence scores, while ECSMiner requires all data instances to be labeled. On the other hand, MINAS operates in a completely unsupervised manner, without the need for any labeled data [56, 92, 114, 147].

Given the number of datasets tested and their varying characteristics, establishing a consistent methodology for evaluating each data characteristic, as presented in Section 5.3, was essential. For instance, a time between novel classes ($t$) of 10000 samples may be suitable for a dataset containing 500000 samples, but inadequate for one with only 20000 samples. Therefore, we developed the following methodology to determine each data attribute value in our experiments, which takes into account the specific characteristics of each dataset:

- **Time Between Novel Classes ($t$)** The time between novel classes was calculated by multiplying a factor ($f$) by the number of known class samples over the total number of classes, including unknown classes. (cf. Equation (5.16)).
- **Ratio of Offline Samples ($r$)** The ratio of offline samples is used as is. It is computed on the total of known class samples.
- **Ratio of Known Classes ($n$)** To designate known classes, we ranked classes in descending order based on the number of samples, initially using those with the highest counts as known classes. This characteristic was only tested on multi-class datasets as it does not apply for binary datasets.
- **Concept Sparsity ($s$)** Concept sparsity was applied directly, as it is unaffected by the dataset's specific characteristics.

$$t = f * \frac{\#\text{ known class samples}}{\text{total }\#\text{ of classes}} \tag{5.16}$$

Using this methodology, we tested several variations of these characteristics on the selected datasets. Specifically, we selected a time between novel classes computed using the aforementioned factor $f \in \{0, 0.1, 0.2\}$, a ratio of offline samples $r \in \{0.2, 0.5, 0.8\}$, a ratio of known classes $n \in \{0.2, 0.5, 0.6\}$, and a concept sparsity $s \in \{0.1, 1, 100, 1000\}$. These values were chosen to simulate scenarios across a wide range of conditions. We aimed to model cases where either a small or large amount of data is available offline, and the inverse during the online phase. Additionally, we intended to obtain distinct numbers of known classes when converting the known class ratio into an absolute value based on the number of classes available for each dataset. However, it is important to note that, while uncommon, this translation of the known class ratio into an integer can occasionally result in the same number of known classes for different scenarios, as seen in the Nursery dataset (cf. Figure 5.6(a)) where $n = 0.5$ and $n = 0.6$ yield the same outcome. To determine the impact of each data characteristic independently of one another, we performed our experiments, similarly to an ablation study, by starting with a baseline and modifying only one data characteristic at a time. This baseline, selected based on commonly used parameters in the field, consisted of no wait time between novel classes ($f = 0$), 20% of the known class samples selected for the training ($r = 0.2$), 20% of the classes selected as known ($n = 0.2$), and no particular sparsity ($s = 1$). We evaluated each data characteristic against this baseline across every dataset and algorithm combination, resulting in a total of over 1300 experiments.

The hyperparameters of each tested algorithm were tuned on each dataset using the baseline. A grid search was employed, selecting the parameter combination that yielded the highest AMI over either 5% of the total dataset samples, or 10000 samples, whichever was greater.

### 5.4.2 Results and Discussion

#### Impact of Data Characteristics

In this section, we analyze and show the impact on the performance of the different DSs characteristics we included in our evaluation framework in order to justify the necessity of reporting them when evaluating ND algorithms. Using the baseline established in Section 5.4.1, we computed the mean, median, and standard deviation of both the $M_1$ ($M_\beta$ with $\beta$=1) and AMI (for multi-class datasets) for each individual dataset. We then determined the difference in these values when varying each data characteristic individually

compared to the baseline. Next, we averaged these results across binary and multi-class datasets independently, as presented in Tables 5.5 and 5.6, respectively. This approach allows us to provide an accessible summary of each data characteristic's effect across all experiments, which is challenging given the large number of experiments conducted (over 1300).

Additionally, for each data characteristic, we present plots illustrating the performance of the worst and best algorithm-dataset combination relative to the baseline, using the average $M_1$ for binary datasets and AMI for multi-class. Specifically, the best-case scenario highlights the algorithm with the highest average increase in performance compared to the baseline, while the worst-case scenario shows the algorithm with the largest decrease. The baseline is consistently represented by a blue dashed line in each plot. To make this work easier to read, only these representative plots are shown. However, all other plots, along with the source code and raw data for every experiment, are available in an online repository[1].

It is also important to note that each data characteristic impacts each algorithm and dataset differently, and as such, the goal of this section is not necessarily to reach a broad consensus on whether changing a characteristic in a certain way leads to a positive or negative performance gain, but rather to display how impactful they can be in DSs and why researchers should report them to ensure repeatability and fair performance evaluation.

**Time Between Novel Classes** Table 5.5 presents the impact of adding various interval of time between the appearance of new classes in the DS on binary datasets. Compared to the baseline ($f = 0$), introducing a small wait time ratio ($f = 0.1$) leads to an overall increase in performance, based on the average and median, for most algorithms with the exception of ECSMiner. This exception could be attributed to ECSMiner being a supervised algorithm and therefore not benefiting from the appearance of novel classes after a delay since it can adapt more effectively when they appear gradually as it can leverage the availability of their true labels. Increasing the wait time ratio further ($f = 0.2$) results in even more pronounced performance improvements for the majority of algorithms, while ECSMiner continues to show no significant change. Another interesting observation is the increase in standard deviation across all algorithms when incorporating these delays between the appearance of novel classes. This suggests that although adding time intervals can enhance performance, the results can vary depending on the dataset and algorithm. These results echo the importance of reporting on data characteristics such as this one to ensure the replicability of the results and prevent potential misinterpretations.

[1] https://github.com/jgaud/PerformanceEvaluationNDDataStreams

Table 5.5: Difference of performance on different characteristics compared to a baseline (*tested* − *baseline*) over binary datasets (positive in boldface).

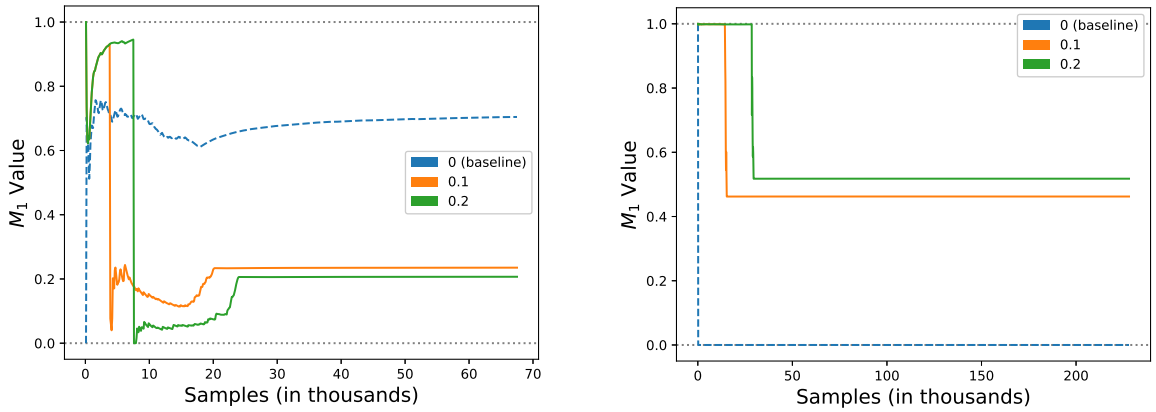
| | | | $M_1$ **Change compared to baseline** | | |
|---|---|---|---|---|---|
| **Characteristic** | **Value** | **Algorithm** | **Mean** | **Median** | **Std** |
| Time between novel classes | 0.1 | MINAS | **0.091** | **0.074** | **0.045** |
| | | ECHO | **0.103** | **0.074** | **0.072** |
| | | ECSMiner | **0.005** | -0.004 | **0.033** |
| | | ECSMinerWF | **0.105** | **0.067** | **0.118** |
| | 0.2 | MINAS | **0.124** | **0.102** | **0.054** |
| | | ECHO | **0.103** | **0.084** | **0.059** |
| | | ECSMiner | **0.011** | -0.007 | **0.057** |
| | | ECSMinerWF | **0.137** | **0.077** | **0.155** |
| Ratio of offline samples | 0.5 | MINAS | **0.030** | **0.034** | -0.013 |
| | | ECHO | -0.058 | -0.085 | -0.003 |
| | | ECSMiner | -0.110 | -0.081 | **0.045** |
| | | ECSMinerWF | -0.088 | -0.093 | **0.001** |
| | 0.8 | MINAS | **0.071** | **0.073** | 0 |
| | | ECHO | **0.038** | **0.041** | -0.020 |
| | | ECSMiner | -0.232 | -0.296 | **0.018** |
| | | ECSMinerWF | -0.095 | -0.102 | **0.004** |
| Concept sparsity | 0.1 | MINAS | -0.006 | -0.007 | **0.002** |
| | | ECHO | -0.011 | -0.006 | -0.015 |
| | | ECSMiner | **0.022** | **0.019** | **0.008** |
| | | ECSMinerWF | 0 | 0 | 0 |
| | 100 | MINAS | -0.008 | -0.005 | -0.008 |
| | | ECHO | **0.043** | **0.033** | **0.031** |
| | | ECSMiner | -0.011 | -0.005 | -0.008 |
| | | ECSMinerWF | **0.004** | **0.003** | **0.002** |
| | 1000 | MINAS | **0.025** | **0.012** | **0.045** |
| | | ECHO | **0.072** | **0.065** | **0.060** |
| | | ECSMiner | -0.026 | -0.018 | **0.057** |
| | | ECSMinerWF | **0.068** | **0.049** | **0.065** |

Table 5.6: Difference of performance on different characteristics compared to a baseline (*tested* − *baseline*) over multi-class datasets (positive in boldface).

| | | | AMI **Change compared to baseline** | | | $M_1$ **Change compared to baseline** | | |
|---|---|---|---|---|---|---|---|---|
| **Charac.** | **Value** | **Algo.** | **Mean** | **Mdn** | **Std** | **Mean** | **Mdn** | **Std** |
| Time between novel classes | 0.1 | MINAS | -0.001 | -0.001 | **0.008** | **0.061** | **0.035** | **0.047** |
| | | ECHO | **0.006** | **0.001** | **0.025** | **0.072** | **0.049** | **0.049** |
| | | ECSM | -0.005 | -0.007 | 0 | -0.027 | -0.033 | 0 |
| | | ECSMWF | -0.001 | **0.002** | **0.006** | **0.038** | **0.023** | **0.056** |
| | 0.2 | MINAS | -0.005 | -0.006 | **0.013** | **0.110** | **0.081** | **0.066** |
| | | ECHO | **0.020** | **0.013** | **0.021** | **0.099** | **0.077** | **0.048** |
| | | ECSM | -0.014 | -0.016 | **0.003** | -0.026 | -0.033 | **0.007** |
| | | ECSMWF | -0.002 | **0.001** | **0.007** | **0.069** | **0.041** | **0.087** |
| Ratio of offline samples | 0.5 | MINAS | -0.005 | -0.004 | **0.002** | -0.012 | -0.009 | **0.007** |
| | | ECHO | **0.011** | **0.009** | **0.007** | **0.051** | **0.071** | **0.020** |
| | | ECSM | -0.036 | -0.046 | 0 | -0.064 | -0.028 | **0.064** |
| | | ECSMWF | **0.015** | **0.017** | **0.005** | -0.057 | -0.057 | 0 |
| | 0.8 | MINAS | **0.018** | **0.013** | **0.009** | **0.012** | **0.014** | **0.001** |
| | | ECHO | **0.068** | **0.078** | **0.025** | **0.156** | **0.209** | **0.040** |
| | | ECSM | -0.070 | -0.074 | **0.007** | -0.118 | -0.071 | **0.095** |
| | | ECSMWF | **0.012** | **0.012** | **0.013** | -0.068 | -0.064 | **0.006** |
| Ratio known classes | 0.5 | MINAS | 0 | -0.002 | **0.002** | **0.032** | **0.021** | -0.009 |
| | | ECHO | **0.158** | **0.166** | -0.003 | **0.018** | **0.044** | -0.004 |
| | | ECSM | -0.006 | -0.010 | -0.007 | -0.066 | -0.049 | **0.020** |
| | | ECSMWF | **0.082** | **0.086** | **0.001** | **0.074** | **0.069** | **0.020** |
| | 0.6 | MINAS | **0.014** | **0.010** | **0.002** | **0.107** | **0.095** | -0.004 |
| | | ECHO | **0.205** | **0.215** | **0.004** | **0.016** | **0.039** | -0.012 |
| | | ECSM | **0.011** | **0.005** | -0.013 | -0.098 | -0.069 | **0.008** |
| | | ECSMWF | **0.095** | **0.096** | **0.006** | **0.064** | **0.066** | **0.020** |
| Concept sparsity | 0.1 | MINAS | -0.014 | -0.018 | **0.002** | -0.004 | -0.004 | -0.006 |
| | | ECHO | **0.003** | **0.004** | -0.003 | **0.001** | **0.002** | -0.009 |
| | | ECSM | -0.007 | -0.011 | 0 | -0.037 | -0.049 | **0.003** |
| | | ECSMWF | 0 | **0.001** | 0 | **0.025** | **0.026** | **0.005** |
| | 100 | MINAS | **0.005** | **0.007** | **0.006** | -0.018 | -0.027 | -0.003 |
| | | ECHO | **0.002** | -0.001 | -0.001 | -0.024 | -0.023 | -0.011 |
| | | ECSM | -0.046 | -0.048 | -0.003 | -0.086 | -0.092 | -0.015 |
| | | ECSMWF | -0.002 | **0.003** | -0.001 | 0 | **0.003** | -0.003 |
| | 1000 | MINAS | -0.003 | -0.007 | **0.009** | -0.075 | -0.102 | **0.029** |
| | | ECHO | **0.010** | **0.010** | **0.006** | -0.042 | -0.051 | **0.018** |
| | | ECSM | -0.093 | -0.109 | **0.001** | -0.169 | -0.178 | **0.014** |
| | | ECSMWF | -0.007 | -0.001 | **0.002** | **0.018** | **0.008** | **0.045** |

In the context of multi-class datasets presented in Table 5.6, a similar outcome can be seen. The algorithms' ability of distinguish between known and novel classes ($M_1$) is increased with the introduction of a delay between novel classes, with the exception of ECSMiner. However, when evaluating AMI, this change in data characteristic does not result in any significant difference. This indicates that while increasing the time interval between novel classes generally allows the algorithms to better distinguish between known and novel classes ($M_1$), it does not appear to significantly affect their ability to distinguish among the novel classes themselves (AMI).

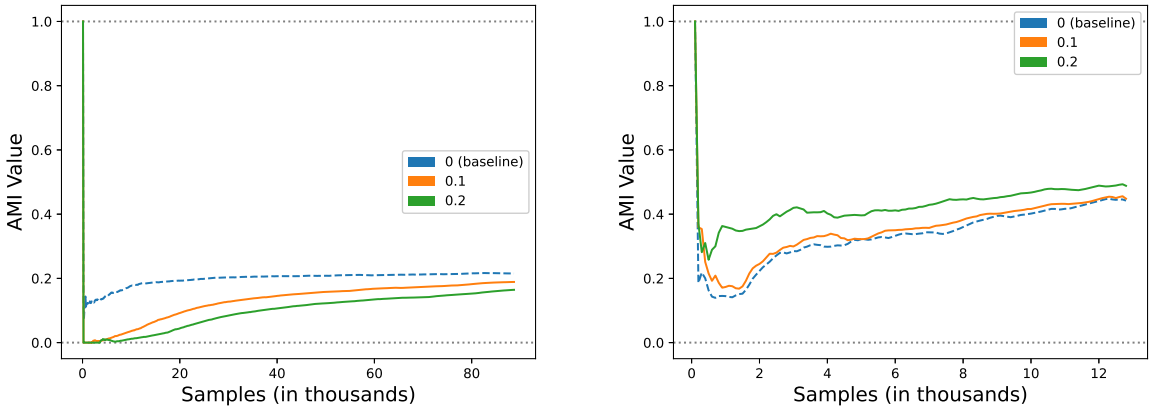
Figures 5.2 and 5.3 illustrate the worst- and best-case scenarios on binary and multi-class datasets, respectively. In the binary scenario's worst case (cf. Figure 5.2(a)), the addition of a wait time prior to the appearance of a novel class leads the algorithm (ECSMiner) to drop significantly in performance as soon as the novel class first appears. This aligns with our previous analysis, indicating that ECSMiner, being supervised, likely benefits from a gradual novel class appearance rather than a more dense appearance after a delay. On the other hand, in the best case (cf. Figure 5.2(b)), ECSMinerWF initially fails to detect the novel class entirely without the addition of a wait time prior to its appearance. In the worst and best-case scenarios of the multi-class data in Figure 5.3, the AMI results show less pronounced differences, but confirm our prior conclusions. The added interval between novel classes does not lead to a global improvement or decline in the algorithm's ability to distinguish among novel classes, although it can still be significant, depending on the specific dataset and algorithm.




Worst case
(a) ECSMiner on Rialto (binarized)

Best case
(b) ECSMinerWF on Credit Card

Figure 5.2: Effect of the time between the appearance of novel classes in binary scenarios




Worst case
(a) MINAS on SensIT

Best case
(b) ECHO on Gas Sensor

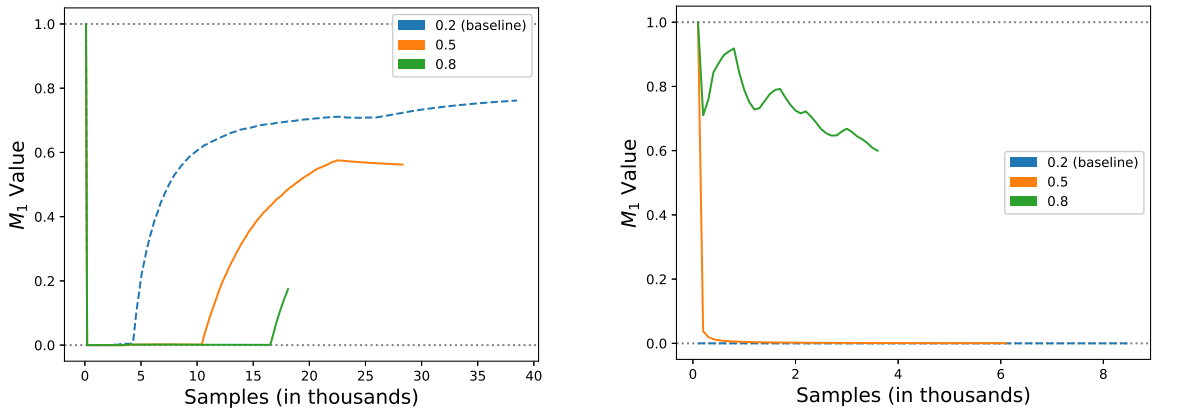
Figure 5.3: Effect of the time between the appearance of novel classes in multi-class scenarios

**Ratio of Offline Samples** The ratio of offline samples does not show as much consensus as the time between novel classes. As observed in Table 5.5 for binary datasets, while the MINAS algorithm appears to benefit from an increase in offline samples, other algorithms, particularly ECSMiner and ECSMinerWF, exhibit a notable decrease in performance. ECHO also demonstrated varying results, where an intermediate offline sample ratio ($r = 0.5$) led to reduced performance, but a higher ratio ($r = 0.8$) improved it.

In multi-class scenarios (cf. Table 5.6), the results are similarly mixed. In this case, ECHO demonstrates the most improvement in both $M_1$ and AMI when additional offline samples are included, while other algorithms either show a performance decline, similar to the binary scenario, or no significant change from the baseline, as seen with MINAS. It is worth noting that increasing the samples used in the offline phase reduces those in the online phase, potentially limiting the algorithm's ability to gradually learn new classes. Additionally, a higher density of known samples in the offline phase increases the density of novel class samples in the online phase, potentially complicating the algorithm's ability to distinguish between classes and contributing to the different behaviours observed.

This behaviour is evident in the worst-case scenarios shown in Figures 5.4(a) and 5.5(a), where the algorithm not only requires more time to recognize novel classes but also has a shorter period for adaptation due to the shorter online phase. In particular, Figure 5.4(a) illustrates that $M_1$ performance, with an 80% offline ratio ($r = 0.8$), begins to improve

around 16,000 samples but is not able to potentially continue increasing before the stream ends. Conversely, in the best-case scenarios (Figures 5.4(b) and 5.5(b)), this problem is not seen as the performance of the baseline is either very low or does not increase as much. These figures demonstrate why comparing different methods of creating the DS is flawed and can result lead to evaluations. Indeed, these graphs show that the offline curve varies in length depending on the ratio of offline samples used. A lower ratio leaves more samples for the online phase, which provides more time for the algorithm's performance to either increase or decrease over the course of the DS.

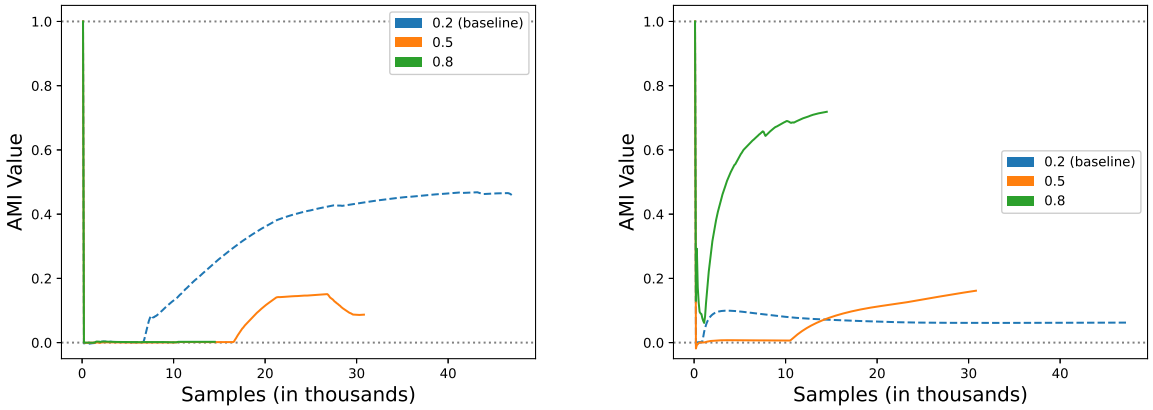



Worst case
(a) ECSMiner on Adult

Best case
(b) ECHO on Room Occupancy (binarized)

Figure 5.4: Effect of the ratio of offline samples in binary scenarios

**Ratio of Known Classes** As noted in Section 5.4.1, the ratio of known classes was exclusively tested on multi-class datasets as it is not applicable to binary datasets. Examining the results in Table 5.6 reveals that most algorithms, with the exception of ECSMiner, benefited from an increased number of known classes. Specifically, each of these algorithms demonstrated an improvement in averaged $M_1$ score indicating an improvement in the algorithms' ability to detect between novel and known classes. Given that most of these algorithms operate in an unsupervised fashion during the online phase, the results align with expectations that incorporating more classes in the supervised, offline phase would enhance performance. Conversely, similar to previous results, as ECSMiner performs in a supervised fashion in the online phase, it does not seem to benefit, but rather is disadvantaged by having a higher ratio of known classes. Furthermore, looking at the AMI




Worst case
(a) ECSMiner on Shuttle

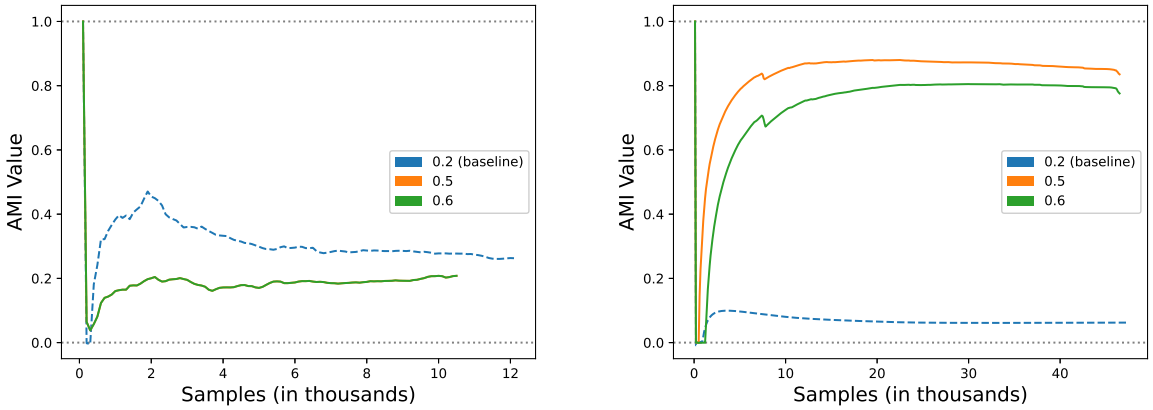
Best case
(b) ECHO on Shuttle

Figure 5.5: Effect of the ratio of offline samples in multi-class scenarios

metric, two algorithms, namely ECHO and ECSMinerWF saw notable improvements in their ability to effectively separate and classify the different classes. ECHO, in particular, exhibited a higher average increase (20.5% vs. 15.8%) with the highest ratio of known classes (60%) compared to the lower percentage (50%). These results once again highlight the importance of reporting on these data characteristics, as some algorithms may benefit significantly, improving by over 20%, while others may not.

The worst and best-case scenario graphs for this change are shown in Figure 5.6. Looking at the worst-case result (cf. Figure 5.6(a)), it shows an example of how adding more known classes, in the case of ECSMiner, resulted in a lower performance. Specifically on this particular dataset, the 50% and 60% ratios of known classes resulted in the same number of known classes (3 classes) due to the low number of classes in the dataset (5 classes) which explain the identical results. In contrast, the best-case result (cf. Figure 5.6(b)) shows that ECHO leveraged the additional known classes, achieving over a 70% increase for the intermediate ratio of known classes ($n = 0.5$) and a smaller yet notable improvement at the highest ratio ($n = 0.6$) on this particular dataset.

**Concept Sparsity** In this work, we introduce the concept sparsity as a novel data characteristic, which, to our knowledge, has not been explored in prior research. As mentioned before, concept sparsity refers to the probability that the same class appears in a row in the DS. The baseline used of 1 ($s = 1$) implies that all classes have the same chance of




Worst case
(a) ECSMiner on Nursery

Best case
(b) ECHO on Shuttle

Figure 5.6: Effect of the ratio of known classes in multi-class scenarios

appearing at any point, while a value of less than 1 decreases the probability that the same class appears twice or more in a row, and a value greater than 1 increases this probability. Analyzing the aggregated results for both binary and multi-class datasets (Tables 5.5 and 5.6) the lowest tested sparsity level ($s = 0.1$) showed minimal impact on algorithm performance. ECSMiner, the algorithm with the largest change, displayed a 2.2% average M1 increase on binary datasets, but a 3.7% decrease on multi-class datasets. Although a slight decrease in performance was anticipated due to fewer samples of the same class appearing consecutively, this limited impact is likely due to the low sparsity value (10%) and the buffer in these algorithms, which allows them to retain novel samples over time and mitigate this effect.

This trend of having different results between binary and multi-class datasets is also continued when looking at the sparsity levels higher than 1. Specifically, on binary datasets (cf. Table 5.5), the intermediate level ($s = 100$) resulted in a mitigated change in performance with a small bump in performance for ECHO, while the highest ($s = 1000$) showed an increase in average $M_1$ across the board, except for ECSMiner. On the other hand, multi-class datasets (Table 5.6) displayed the opposite, showing a significant decrease in $M_1$ score, except for ECSMinerWF which seems to be somewhat unaffected. On AMI performance, most algorithms other than ECSMiner, which show a significant decrease, are largely unaffected by the change in sparsity.

These findings suggest that higher sparsity levels yield small performance gains on

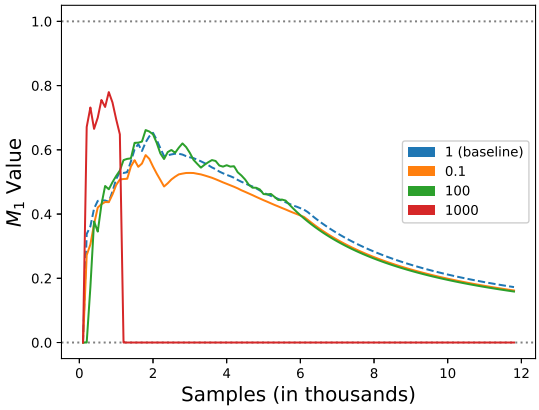
binary datasets for most algorithms but significantly lower performance on multi-class datasets. However, it is essential to note that, based on our results, the concept sparsity's effect varies substantially by dataset and algorithm, even more so than other data characteristics. This variability is illustrated by the worst- and best-case scenarios on binary datasets in Figures 5.7(a) and 5.7(b). In the worst-case (cf. Figure 5.7(a)), ECHO's performance on this particular dataset remained close to baseline under lower sparsity levels (s = 0.1 and s = 100), but dropped to zero around a thousand samples with the highest sparsity level (s = 1000). Here, the stream contained only novel samples initially, and once known class samples were introduced, the algorithm failed to detect any as known. In the best-case scenario (cf. Figure 5.7(b)), the graph shows a similar drop in ECHO's performance around a thousand samples at the highest sparsity level (s = 1000). However, in this case, the algorithm recovers as soon as it learns the class that was introduced and maintains a much higher score than the baseline. These graphs emphasize the variability in performance that the same algorithm can exhibit, depending on the dataset. In the multi-class case, the worst-case (cf. Figure 5.8(a)) displays ECSMiner, which as we discussed earlier, seems to perform worse in all cases where the sparsity was changed compared to the baseline, while the best-case (cf. Figure 5.8(b)) shows that MINAS benefitted, in this particular case, from an increase in the sparsity.

In conclusion, our findings illustrate that concept sparsity can significantly impact algorithm performance, whether positively or negatively, depending on the dataset and algorithm used. To our knowledge, no prior work has examined this data characteristic; however, as our study demonstrates, concept sparsity can meaningfully influence ND results and should therefore be reported when evaluating such algorithms.

### Comparison of the Proposed Framework

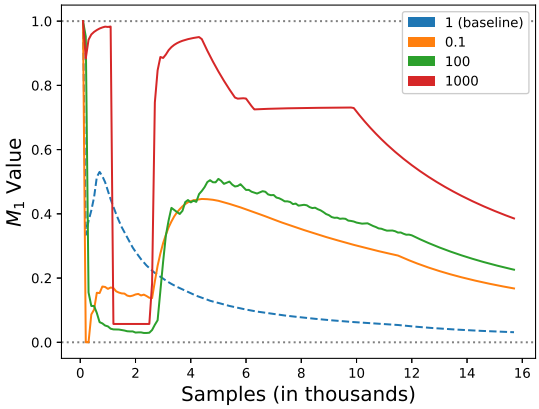
In this section, we compare the classification performance of the metric we proposed for the evaluation of ND algorithms (AMI) compared to the previously used metric (CER) in order to demonstrate some of its advantages. Specifically, we explore the correlation between these metrics and highlight how evaluating multi-class ND algorithms using AMI accounts for model complexity without requiring additional metrics, contrary to CER, one of the few other metrics proposed for such scenarios, which does not.

We computed the mean of both metrics on every experiment performed in this study and plotted a pairwise comparison shown in Figure 5.9. It is worth noting that we used the inverted CER for a better comparison between the two metrics, as normally, a lower value of CER represents a better score, while a lower value of AMI is worse. Moreover, while CER is bounded between 0 and 1, AMI is only bounded to 1 and can output a negative value, with




Worst case

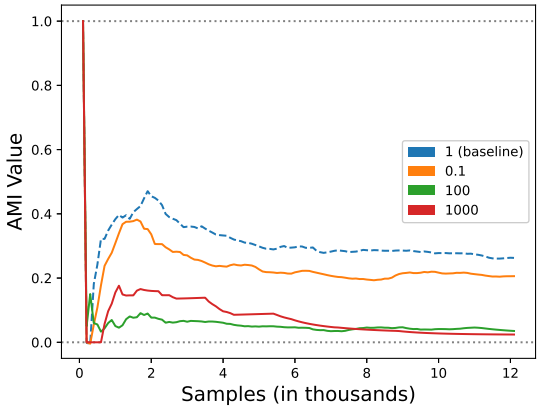
(a) ECHO on Gas Sensor (binarized)




Best case

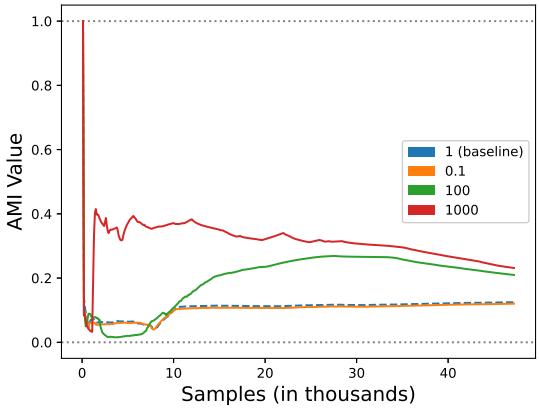
(b) ECHO on NOAA

Figure 5.7: Effect of the concept sparsity in binary scenarios




Worst case

(a) ECSMiner on Nursery




Best case

(b) MINAS on Shuttle

Figure 5.8: Effect of the concept sparsity in multi-class scenarios

0 representing a random classifier. With this into consideration, Figure 5.9 shows that AMI did not display values much lower than 0, which is to be expected as ideally, the algorithms perform better than a random classifier, while the values of inverted CER tended to be much higher than AMI, even for scenarios that performed poorly when looking at the latter. This can be explained by the fact that CER does not take the complexity of the model (number of clusters) into consideration, which is why the authors originally proposed to combine it with another value (AIC) to evaluate it, as mentioned in Section 5.2.2. Otherwise, as shown in the plot, CER tends to overestimate the performance of the algorithms. Specifically, we observe that many results with an AMI near 0, indicative of a random or uniform label assignment, achieved an inverted CER ranging from approximately 0.4 up to a perfect score of 1.0. Conversely, the absence of points in the lower-right quadrant below the identity line indicates that no cases exhibited a high AMI value with a very low CER, demonstrating that the algorithm consistently avoided falsely suggesting strong performance when it did not accurately classify the samples. In these cases, the use of AMI allows us to take into account the complexity of the model, and uniform label assignment, using only one metric and to compare between datasets as it offers a baseline, which AIC does not.

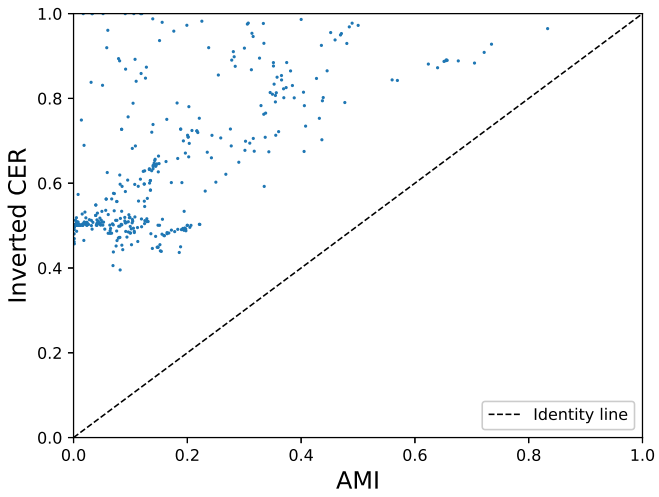



Figure 5.9: Correlation between AMI and Inverted CER on multi-class datasets

### Temporal Metrics

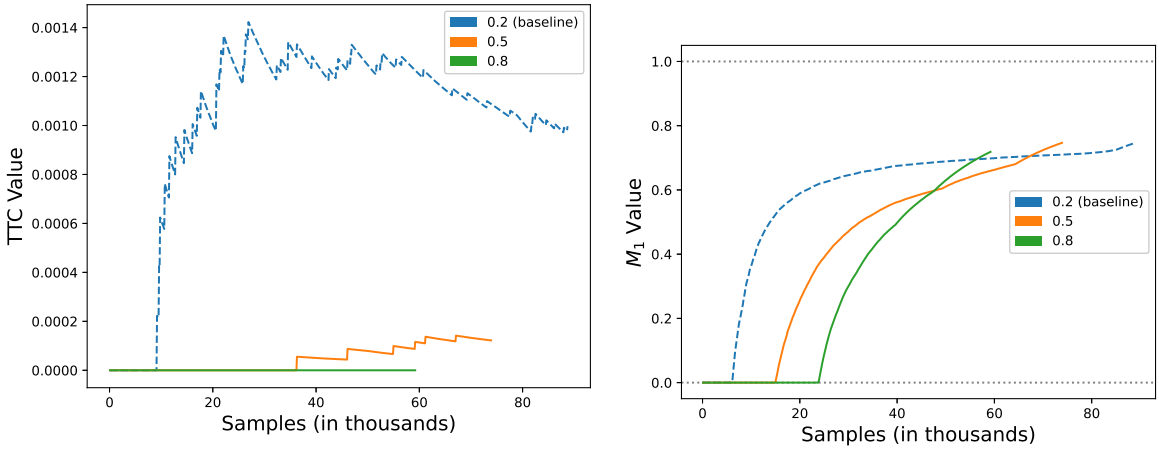
In this section, we address our third objective, which is to demonstrate the utility of the proposed novel metrics in relation to the temporal aspects of DSs (TTD and TTC). To this end, as in Section 5.4.2, we present the data characteristics in a worst-case scenario for both TTD and TTC, representing the maximum increase compared to the baseline established in Section 5.4.1 and excluding cases where the algorithm did not find the novel class. This approach highlights the relevance of temporal aspects in DSs and underscores the importance of evaluating them when assessing these scenarios. The ratio of known classes is not considered here, as varying unknown classes across experiments make comparisons between changes and the baseline impractical.

Starting with the TTD metric, the worst-case scenario occurred when the ratio of offline samples increased on the SensIT (binarized) dataset using ECSMiner. Table 5.7 shows that the TTD of the dataset's only unknown class (Class 0) rose significantly with each increase of the offline sample ratio. This indicates that the algorithm took a lot longer in order to detect this specific novel class. TTD allows us to identify classes problematic for the algorithm and assess its adaptability in quickly identifying novel classes, a very important capability in scenarios such as intrusion detection. In addition, we also examined TTC and $M_1$ in this scenario, demonstrating the distinct insights each metric provides, as shown in Figure 5.10. $M_1$ was selected due to the dataset's binary nature. Figure 5.10(a) shows that, while TTC values are generally low, they increase slightly with a higher ratio of offline samples, reflecting that while the time needed to detect the novel class (TTD) increases, the classification time post-detection (TTC) does not significantly decrease at higher offline sample ratios. Lastly, $M_1$ in Figure 5.10(b) illustrates a performance decline as the offline sample ratio rises, reinforcing each metric's unique perspective. These findings underscore the value of incorporating multiple metrics to provide a comprehensive evaluation of temporal aspects in DSs.

Table 5.7: Impact of Offline Sample Ratio on the ECSMiner Algorithm's TTD Performance for the SensIT (binarized) Dataset

| **Ratio Offline** | **TTD Class 0** |
|---|---|
| 0.2 (Baseline) | 2989 |
| 0.5 | 7379 |
| 0.8 | 13898 |

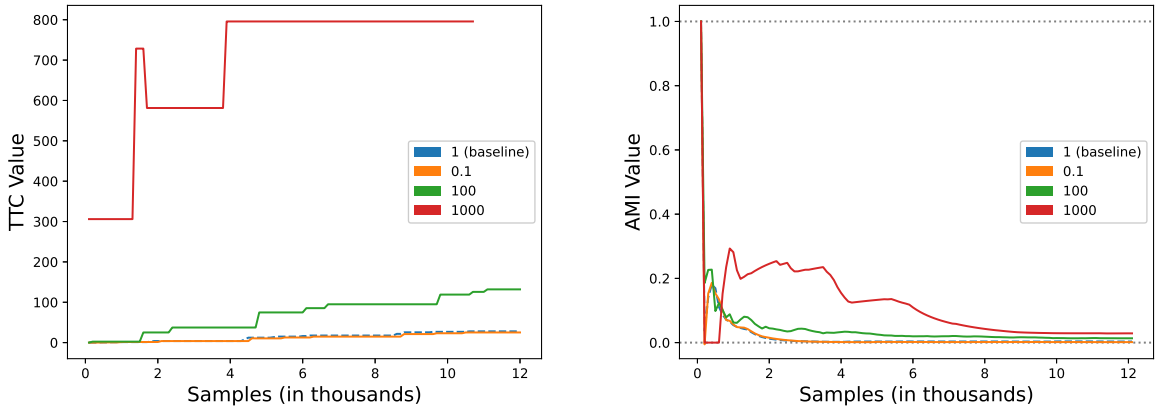
In the second scenario, where the most significant increase in TTC compared to the




(a) TTC of class 0 (b) $M_1$

Figure 5.10: Effect of the ratio of offline samples for the ECSMiner algorithm on the SensIT (binarized) dataset

baseline was observed, it occurred in class 0 (a known class) of the Nursery dataset with the MINAS algorithm when increasing the concept sparsity. Figure 5.11(a) shows a drastic rise in TTC after increasing the sparsity to 100, and further to 1000. The graph demonstrates that, while TTC values increase gradually over the course of the DS in all cases, as new classes are introduced, higher sparsity exacerbates the algorithm's difficulty in classifying the known class. This behaviour may be due to higher sparsity levels, making it more likely for identical classes to appear consecutively, which could cause the algorithm to forget other known classes, making them harder to classify. As before, we also provide an additional performance metric to evaluate the algorithm's overall performance, in this case AMI, given that the dataset is multi-class, as shown in Figure 5.11(b). Here, we observe that the results with the highest performance are actually the ones with the highest sparsity ($s = 100$ and $s = 1000$), demonstrating once again the different conclusions that each metric can give, where while the higher sparsity increases the algorithm's overall performance in detecting and classifying novel classes on this dataset, it also at the same time increases the time needed to classify the known class.




(a) TTC of class 0 (b) AMI

Figure 5.11: Effect of the concept sparsity for the MINAS algorithm on the Nursery dataset

## 5.5 Summary

In this chapter, we present a formalization of an evaluation framework for ND which aims to resolve some of the issues that currently afflict the field, namely the common use of solely binary metrics on multi-class problems that do not provide a fair evaluation of the models, as well as the lack of a common set of evaluation standards, which makes repeatability and comparisons difficult. First, our proposed framework outlines four main data characteristics that should be set and reported when evaluating ND algorithms, specifically the time between novel classes, the ratio of offline samples, the ratio of known classes, and the concept sparsity. We also empirically demonstrated, on multiple different datasets and algorithms, the impact, both positive and negative, that each of these characteristics can potentially have on the end result, which can be quite significant. Secondly, our evaluation structure suggests the use of two metrics, $M_\beta$, and AMI, for the binary and multi-class scenarios, respectively, which allow us to easily track the model's classification performance through only one metric which is more easily comparable. We also demonstrated that the use of the proposed metric, specifically for multi-class, allows us to also take into account the complexity of the model, and avoid an over-optimistic evaluation when the model outputs too many small, independent NPs compared to currently used metrics in the field. Lastly, we propose two novel metrics to evaluate the temporal aspect of ND (TTD and TTC), an aspect currently not evaluated in most works that is an important feature to consider in many applications. We also empirically showcase how these metrics can be

used, their relevance when evaluating such scenarios, and the unique insights they offer compared to traditional performance metrics.

We hope that this work can serve as a future reference for properly evaluating ND algorithms in DSs scenarios, as well as entice future work in this field, which could include the exploration of the impact of concept drift or the addition of other data characteristics. We also use this work as a baseline for evaluating our proposed algorithm, which will be discussed in Chapter 7.

# Chapter 6

# Detection of Novel Cyberattacks Using Multi-Binary Classifiers

This chapter focuses on our third and fourth objectives introduced in Chapter 1 and presents an early precursor implementation of a novel approach for novelty detection using multiple binary classifiers and threshold filters. The goal of this approach is to enable online updates without taking the model offline while limiting the number of false detections. This early work represents a first step toward our broader objective of implementing a complete architecture for novelty detection, and here we focus only on the classification component to ensure its validity before integrating it into the final system.

The contents of this chapter are based on the following paper:

Jean-Gabriel Gaudreault and Paula Branco. Detection of novel cyberattacks using multi-binary classifiers. In *2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, pages 1–9, 2024

While the initial publication of this work demonstrated results solely on network intrusion datasets, we also performed experiments on a wider range of datasets from different domains to verify the robustness of the approach before extending it into a fully fledged ND algorithm. We therefore expand upon the original work in this chapter to include these additional results, providing a broader perspective in this thesis.

## 6.1 Introduction

As computer systems have become an essential part of our daily lives, so have cyberattacks from malicious actors. Given the vast quantity of data being transferred, system administrators have long relied on automated tools to prevent these attacks from reaching their targets. Prime examples of such tools are Intrusion Detection Systems (IDSs), which have become an essential security layer of any modern network. While the usage of signature-based IDSs, which detect threats by representing malicious activities as signatures, is not new, there has been, throughout the years, a notable shift towards more advanced IDSs based on ML techniques due to their generally increased performance [137].

By leveraging large amounts of data, ML techniques can extract patterns that are often hard or impossible to observe manually by human experts. This allows for the development of IDSs that require minimal domain expertise while maintaining a high detection rate and few false alarms. ML strategies can be further classified as supervised or unsupervised, based on whether they utilize labeled data for training or not, respectively. Although acquiring labeled data is typically a significant hurdle, the efficacy of unsupervised techniques in identifying attacks is generally lower than supervised ones, which has historically made the latter a preferable option [137].

When using ML, the problem of intrusion detection can be modeled as a DS and approached in two main ways: as a binary classification, where the system only indicates whether a given sample is an attack or not, or as a multi-class classification, where it also identifies the specific type of attack. Although the latter approach provides more fine-grained information on the nature of the detected attack, it has a significant drawback, as it usually cannot identify new, previously unseen attacks that were not part of its training data. This limitation is particularly problematic in cybersecurity, as the system is unable to identify new attacks properly and quickly becomes obsolete as these attacks emerge. This issue relates to a common classification problem in ML known as OSR, where there is an assumption that not all of the classes are available at training time and the model needs to be robust enough to detect these unknown classes while also classifying known ones [7]. More specifically, cybersecurity problems can be framed as ND tasks, where newly detected attacks are treated as novel classes and can later be integrated into the decision model once they are confirmed by a cybersecurity specialist, an aspect that we explore in Chapter 7.

As discussed in the following section, most approaches proposed in the literature present several limitations, such as solely identifying binary labels or failing to address novel attacks that may emerge post-training. In this chapter, we introduce a novel approach for intrusion

detection in multi-class OSR scenarios, which can also be applied more generally to any other datasets with emerging novel classes.

**Contributions** The main contributions of this chapter can be summarized as follows:

(i) use of multiple binary supervised classifiers leverages the higher performance of supervised classification while enabling the detection of novel attacks;

(ii) propose the use of threshold filters based on a weighted Youden Index to automatically find a threshold in order to increase the detection of novel attacks;

(iii) utilization of shallow, simple ML models ensures quick response time without the need for extensive training data compared to other methods such as using DNNs;

(iv) the modularity of the proposed model allows for the integration of newly identified attacks into the system without taking it offline;

(v) test our approach on contemporary intrusion detection datasets, demonstrating its efficacy in handling recent attack vectors.

**Organization** The rest of this chapter will be structured as follows: Section 6.2 reviews other approaches that have been proposed in the field of OSR for intrusion detection; Section 6.3 goes into details on the architecture of our proposed solution; Section 6.4 presents our evaluation methodology; Section 6.5 presents and discusses the experimental results; and Section 6.6 summarizes this chapter.

## 6.2 Related Works

While there is a significant amount of literature proposing various ML techniques for IDSs, research addressing the problem as an OSR issue and therefore being able to not only classify into multiple known attacks but also detect novel ones, is somewhat limited.

Several studies have explored the application of OSR to IDSs in the context of deep learning. In particular, Xu et al. [226] propose the central clustering method, which involves identifying the boundary of each known class cluster and can be adapted to existing network architectures. By using negative samples as the unknown class during training, they can establish a threshold for known classes and identify potential novel attacks using

a fuzzy distance measure. In contrast, Zhang et al. [240] propose an open-set classification network based on a convolutional network that detects potential unknown attacks and classifies known ones by learning an optimal feature representation for each class, assuming they are sufficiently distinct. They further introduce a semantic embedding clustering method to separate specific unknown classes within the identified unknown samples. Finally, incremental learning can be performed on the newly identified classes to update the classifier without the need for retraining. Similarly, Soltani et al. [196] present a deep learning-based framework consisting of three phases. The first phase performs the OSR task by classifying known attacks and detecting novel ones using different OSR deep learners. The second phase clusters the unknown samples using unsupervised methods, and the third phase retrains the model once the labels have been confirmed by a human expert.

Other approaches beyond deep learning have also been explored, particularly in the context of multi-stage IDSs. These approaches typically involve employing a combination of multiple detection methods in stages to enhance detection and classification performance. Al-Yaseen et al. [14] propose a multi-level IDS model using hybrid SVM and ELM classifiers to analyze network traffic data, where one classifier at each level classifies the packet into a category. Samples not classified into known categories are considered unknown. Bovenzi et al. [31] propose a two-stage approach that first utilizes a multi-modal deep autoencoder, which then feeds into soft-output classifiers in the second stage. These classifiers use a threshold on the prediction confidence to detect potential novel attacks. To optimize this threshold, the classifier in the first stage is trained on data by sequentially removing each known class. Verkerken et al. [216] employ a similar multi-stage approach composed of three stages. In the first stage, an anomaly detector outputs an anomaly score, which determines if the sample is benign or if it should proceed to the next stage. In the second stage, the classifier determines if the sample belongs to a known class; if not, it is sent to the third stage, where the anomaly score is used with another threshold to ascertain whether it is an unknown attack or a benign sample.

Other novel techniques have also been explored to identify unknown attacks while classifying known ones. For example, Souza et al. [197] use an adaptation of a single-class energy-based flow classifier, which employs the concept of flow energy, a measure of the degree to which a flow does not conform to a given probability distribution. The energy of a flow can be compared to the energy of known classes to determine if the sample is benign or not. They adapt this technique to a multi-class scenario, allowing them to classify samples into different types of attacks and identify novel samples.

While all the works presented in this section employ techniques for implementing IDSs in the context of OSR, without assuming that classes shown during training are constant, our approach differs significantly. Our method utilizes multiple binary classifiers in conjunction

with threshold filters, offering several distinct advantages. Specifically, through the use of simple learners, our approach requires minimal training data and ensures rapid response times for classifying incoming samples. Furthermore, the modularity of our approach facilitates the incorporation of new classes without necessitating re-training or offline model updates. We also validate the efficacy of our technique across various contemporary datasets, demonstrating its capability to detect recent attacks included in these datasets.

## 6.3 Proposed Approach

In this section, we present the overall architecture of our proposed approach. Traditional classification methods based on supervised learning typically assume that all classes encountered during deployment are available during training. To address the OSR problem and detect novel classes that may emerge during deployment while using supervised classification, we devised an architecture that adapts this classification to the OSR problem. A high-level view of our proposed approach is depicted in Figure 6.1.

Our architecture consists of three main components: the binary classifiers, the threshold filters, and a voting classifier. The main advantage of this approach is its ability to leverage the high predictive power of supervised learners while also detecting novel classes and keeping a low false alarm rate due to the help of the threshold filters, which screen out benign cases. Moreover, although this aspect is something we do not explored in this chapter, the modular and straightforward nature of the proposed architecture allows for the addition of new binary classifiers for novel classes in an online setting, enabling quick training and deployment of models without the need for extensive hyperparameter tuning. In the following sections, we provide detailed explanations of each component of the proposed approach.

### 6.3.1 Binary Classifiers

The first component and phase of our proposed approach involves using binary classifiers, each trained to distinguish one class from all others in the training data. Therefore, with $n$ known classes in the training data, the total set will contain $n$ classifiers. Although any supervised algorithm that outputs probabilities can be employed to create these classifiers, we recommend and have used ensemble algorithms in our testing. Ensemble algorithms have shown satisfactory performance across various data types, particularly imbalanced data, which is common in intrusion detection scenarios. Additionally, these algorithms

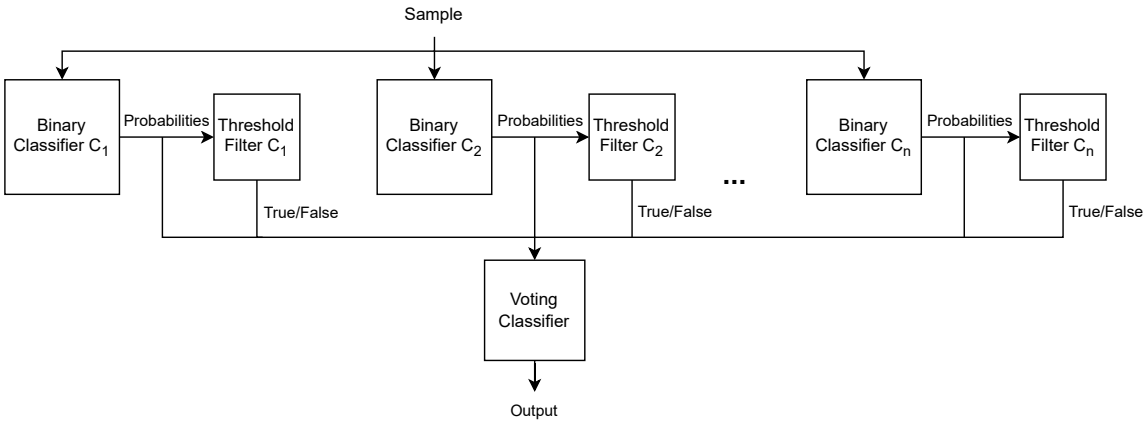



Figure 6.1: High-level architecture of the proposed approach

typically require less data compared to other types and can output predictions quickly, a critical feature for the effective deployment of IDSs [62].

During deployment, when a sample arrives, each binary classifier independently classifies it. The probabilities outputted by each classifier are subsequently analyzed by the corresponding threshold filter for each classifier. Additionally, new binary classifiers can be seamlessly incorporated into the existing set during deployment, without disrupting the overall architecture's operation. This flexibility allows for potential extensions of the framework, such as the integration of new classes, which is explored in the subsequent chapter, or the updating of existing classifiers to address concept drift.

### 6.3.2 Threshold Filters

As previously mentioned, each binary classifier is paired with a corresponding threshold filter serving as a second certainty check before confirming whether a sample belongs to the classifier's class. This threshold is applied directly to the probability output and is determined after training using the Youden Index [233], a common summary statistic of the receiver operating characteristic curve. We selected the Youden Index to compute the threshold, as it effectively identifies the optimal threshold value that maximizes the TPR and minimizes the FPR [187], corresponding to the probability of detection and probability of false alarm, respectively. It has been used effectively in other domains, particularly in disease detection where it is used to determine the optimal cut-point to differentiate between a healthy and diseased individual [186]. While the standard Youden Index gives equal

weight to both TPR and FPR, our implementation places greater emphasis on minimizing false alarms (FPR) of the known classes, while conversely increasing the detection of novel attacks. Hence, we propose the use of a weighted Youden Index, defined in Equation (6.1), where the weight ($w$) can be adjusted based on the specific data characteristics.

$$J_w = TPR + (w * (1 - FPR)) \tag{6.1}$$

### 6.3.3 Voting Classifier

Lastly, the probabilities from all binary classifiers and the binary outputs from the threshold filters are forwarded to the voting classifier for final classification. In cases where all threshold filters yield negative outputs (indicating none of the classifiers recognized the sample as belonging to their class), the sample is considered a novel class. Although these samples are not further processed in this work, as we consider only one novel class, they could be labeled as unknown and subsequently handled by an algorithm capable of clustering them into new classes of attacks, also referred to as NPs, in order to perform a complete ND task. This extension is explored in Chapter 7.

On the other hand, if one or multiple classifiers consider it as their class, a straightforward voting scheme is employed where the classifier with the highest probability determines the final classification. This simple architecture facilitates a quick training and prediction time for our classifier.

## 6.4 Evaluation Methodology

This section provides detailed information on the empirical setting employed for all experiments and results outlined in Section 6.5. Initially, we introduce the datasets and preprocessing procedure applied. Then, we detail the algorithms and corresponding hyperparameters utilized both in our methodology and as a baseline for comparison. Finally, we outline the metrics used to evaluate the performance of the models. The source code for all experiments and more details on the implementation are also freely available online[1].

[1]https://github.com/jgaud/CASCON2024

### 6.4.1 Datasets

#### Intrusion Detection

We employed five state-of-the-art intrusion detection datasets to evaluate the algorithms: NSLKDD [208], NDSec1 [23], CIC IDS 2017 - Improved [140], NF-UNSW-NB15-v2 [184], and NF-CSE-CIC-IDS2018-v2 [184]. These datasets were selected due to their prevalence as multi-class benchmark datasets in the field of intrusion detection, as well as including a wide range of attacks, including more recent ones. To prepare the data for model training, we applied several preprocessing steps. Specifically, for the datasets with numerous distinct labels and relatively few samples per label (e.g. NDSec1, CIC IDS 2017, NF-CSE-CIC-IDS2018-v2), we categorized the labels into broader categories. For instance, various types of Distributed Denial-of-Service (DDoS) attacks were grouped under a single label, and a similar approach was applied to other attack types, aiming to reduce the number of classes to ten or fewer.

#### Other Datasets

Additionally, to confirm the generalization of our technique to other scenarios, we included six real-world datasets from the USP DS Repository [198] and two synthetically generated datasets. For these datasets, since classes with very few samples cannot be grouped with others as was done for the cybersecurity datasets, we removed any class containing fewer than five samples. Furthermore, as these datasets do not explicitly distinguish between "benign" and "malicious" classes, we considered the most frequent class as benign, and treated the second most frequent and the least frequent classes as novel attacks to be detected for the purposes of our evaluation.

#### Preprocessing

Table 6.1 shows an overview of the characteristics of each dataset after preprocessing, while Table 6.2 details the class names for all intrusion detection datasets. Samples containing undefined values were removed, and an 80%-20% split was applied for training and testing, respectively. To address class imbalance, SMOTE [42] was applied to balance all classes in the training data, except for the NF-CSE-CIC-IDS2018-v2 dataset, where size and technical constraints prevented its use. All features were then standardized by removing their mean and scaling to unit variance.

As an initial point of comparison, we evaluated the scenario where all classes are available in the training data. Then, to evaluate our approach's ability to identify novel attacks, we also tested the scenario where one class (other than benign) was randomly removed from the training data but remained present in the testing data.

Table 6.1: Overview of the main characteristics of the datasets

| **Dataset** | **No. of Classes** | **No. of Features** | **No. of Samples** |
|---|---|---|---|
| NSLKDD | 5 | 124 | 148517 |
| NDSec1 | 10 | 65 | 1384626 |
| CIC IDS 2017 | 10 | 87 | 2097858 |
| NF-UNSW-NB15-v2 | 10 | 42 | 2390275 |
| NF-CSE-CIC-IDS2018-v2 | 9 | 42 | 18893708 |
| Gas Sensor | 6 | 128 | 13910 |
| INSECTS-Abrupt (Balanced) | 6 | 33 | 52848 |
| Nursery | 4 | 27 | 12958 |
| Rialto | 10 | 27 | 82250 |
| Room Occupancy | 4 | 16 | 10129 |
| Shuttle | 7 | 8 | 58000 |
| Random RBF (Synt.) | 4 | 10 | 100000 |
| Random Tree (Synt.) | 4 | 12 | 100000 |

### 6.4.2 Algorithms

As previously mentioned, ensemble methods were chosen to implement the binary classifiers in our architecture due to their strong predictive performance and ability to be trained quickly. During testing, boosting methods, notably LightGBM which surpasses other algorithms in terms of computational speed and memory usage [125], performed well on the type of data we are using compared to other algorithms. Consequently, we opted to use it for all subsequent testing phases to implement our binary classifiers, as well as for our baseline.

Table 6.2: Overview of class names for the intrusion detection datasets

| **Dataset** | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ | $C_9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| NSL-KDD | Normal | Probe | DoS | U2R | R2L | - | - | - | - | - |
| NDSec1 | Botnet | Brute force | DoS | Exploit | Malware | Misc. | Normal | Probe | Spoofing | Web Attack |
| CIC IDS 2017 | Normal | Botnet | DDoS | DoS | FTP-Patator | Heartbleed | Infilt. | Portscan | SSH-Patator | Web Attack |
| NF-UNSW-NB15-v2 | Analysis | Backdoor | Normal | DoS | Exploits | Fuzzers | Generic | Recon. | Shellcode | Worms |
| NF-CSE-CIC-IDS2018-v2 | Normal | Bot | DDoS | DoS | FTP-Bruteforce | Infilt. | SSH-Bruteforce | Web Attack | LOIC | - |

To establish a benchmark for our proposed approach against a baseline, we also trained LightGBM as a traditional multi-class classifier and evaluated its performance. We used the default hyperparameters provided by the library for both the multi-class classifier and the binary classifiers in our approach. The goal was not to achieve optimal performance on specific datasets but to demonstrate the advantages of our approach over traditional classification methods and its efficacy in detecting novel attacks. These hyperparameters include using a traditional gradient boosting decision tree algorithm with no maximum depth, 100 base estimators, and a maximum of 31 leaves per base estimator. Further details on the implementation can be found in the previously mentioned online repository.

For our approach, we tested two configurations: first, without any filtering applied to the output of the binary classifiers, and second, using a weighted Youden Index of 100 (w = 100). This weighting heavily prioritizes achieving a low false alarm rate on known classes, thereby potentially enhancing the detection of novel attacks, as discussed in Section 6.3. This value was selected after testing different values, as it demonstrated a balanced performance between detection sensitivity and false alarm rates.

### 6.4.3 Metrics

To assess the performance of the evaluated algorithms, we primarily relied on the $F_1$ score, which is considered a suitable metric for imbalanced scenarios as demonstrated in Chapter 4 [92], and aggregates both precision and recall (TPR) into a single metric, giving equal weight to both. Additionally, we used precision and recall independently, as these metrics are highly informative in the domain of IDSs. Precision measures the percentage

of detected samples that are relevant, while recall represents the percentage of relevant samples that have been detected. Since this chapter presents preliminary work and does not yet apply our approach to the full ND task, meaning that NPs are not formed, the evaluation methodology introduced in Chapter 5 is not used here. That methodology is instead applied in the following chapter, where this work is extended to address the complete ND problem.

It is also important to highlight that when dealing with multiple classes, we employed macro-averaging to compute the $F_1$ score, which is essentially the unweighted average of the $F_1$ of all classes. This approach ensures that each class receives equal weighting in the evaluation process, reflecting the importance of all classes in our analysis.

Additionally, we also report on two other metrics commonly used in IDSs scenarios, namely the detection rate and the false alarm rate, which correspond to the TPR and the FPR, respectively. These metrics can provide further insight into the algorithm's performance and how accurately it can detect attacks while also minimizing false positives.

## 6.5 Results

This section presents and discusses the results of the experimental evaluation of the two configurations of our proposed algorithm compared to the aforementioned baseline. We begin with an overview of the overall multi-class classification performance of both algorithms. Next, we present results on the attack detection and false alarm rates achieved by the algorithms. Finally, we provide an analysis of the runtime performance of our algorithm and finish by discussing the ability of our algorithm to successfully detect novel classes and its performance in doing so. All experiments were conducted using the datasets introduced in the previous section.

### 6.5.1 Multi-Class Classification

An overview of the global classification results (Macro $F_1$) on the network intrusion datasets can be seen in Figure 6.2. First, looking at the results where no unknown class is introduced, we observe that both implementations of our proposed algorithm perform either on par with or better than the baseline algorithm, with the exception of one dataset (NF-UNSW) where it performs marginally worse. In other cases, notably on the CIC IDS and NF-CSE datasets, our algorithm significantly outperforms the baseline in both configurations which highlights the efficiency of our method compared to using a traditional technique even

without novel classes. Additionally, while the configuration utilizing the threshold tends to perform slightly worse overall compared to the one without when no unknown class is present, the difference is minimal and offers other benefits that will be discussed in the following sections.

On other datasets, as shown in Figure 6.3, both implementations of our algorithm generally achieve results comparable to the baseline. The main exceptions are the Rialto and INSECTS datasets, where performance degrades more noticeably in the threshold-based configuration. This reduction may be due to the threshold imposing stricter constraints on classification in scenarios with greater class overlap compared to cybersecurity attacks. In addition, the INSECTS dataset introduces concept drift within the classes, which may explain the poorer performance of the threshold-based approach, as it might not recognize drifted samples as belonging to their corresponding class.

When an unknown class is introduced by omitting it from the training data but including it in the testing data, we observe that the performance of the baseline classifier drastically decreases in almost all cases. This contrasts with the scenario where all classes are available, in which the baseline performs similarly or better than our approaches. This outcome is expected, as the baseline classifier lacks the capability to detect novel classes, leading to consistent misclassification of these samples and consequently a decrease in the overall classification performance. This highlights the advantage of addressing the IDS problem as an OSR, as our approach showed a much smaller performance decrease when an unknown class was added. Exceptions to this pattern occur with the INSECTS dataset under the threshold-based configuration, likely due to concept drift as previously discussed, and with the NF-CSE dataset when $C_5$ is introduced as an unknown class. The latter anomaly suggests that our approach struggled to correctly identify this particular novel class, an issue we will investigate further in the following sections.

Comparing the two configurations of our approach on scenarios with an unknown class, we can see that the configuration using the threshold filters performs better or equally well in most cases (21 out of 29) compared to the configuration without them. This aligns with our expectations, as the thresholds were selected using a weighted Youden Index of 100, as discussed in Section 6.4.2, which prioritizes achieving a low false alarm rate for known classes. Since the output indicating whether a sample belongs to a known class is used to detect novel classes, limiting the false positives detected by each known class classifier enhances the detection of novel classes, at the cost of a potentially lower classification performance for the known classes. This behaviour is exhibited in the majority of cases, with a few exceptions where the configuration without threshold filters performs better. These exceptions may be due to the cost of lower classification performance being too high compared to the increase in novel class detection or the model's inability to adequately

detect the novel class.

### 6.5.2 Attack Detection and False Alarms

While the previous analysis on multi-class classification performance is valuable for determining if the models can accurately classify attacks and benign traffic into fine-grained categories, such as DDoS, web attacks, SSH, and others, the primary focus in IDS scenarios is often to first only identify whether a sample is an attack or not (binary output). In order to assess the performance of our proposed approach in this regard, Table 6.3 includes the detection rate (percentage of attacks that were detected) and the false alarm rate (percentage of non-attacks misclassified as attacks) for all three scenarios (baseline, our approach without thresholds, and our approach with thresholds) across each intrusion detection dataset, with either no hidden class or one hidden at training time. Since this analysis is specifically relevant to intrusion detection, we do not include it for the other datasets.

Focusing first on the detection rate, we observe that our proposed architecture achieves better results in almost all cases except one, where $C_6$ is unknown in the CIC IDS dataset. Additionally, the configuration using threshold filters outperforms the one without by a small margin in most cases. These results align with our conclusions from the previous section and highlight that our proposed approach, particularly the configuration using thresholds, is better at detecting attacks compared to the baseline, even in scenarios where no unknown class is introduced. While the increase in detection compared to the baseline is small in some cases, it is noteworthy that some scenarios exhibit a considerable increase in detection such as $C_6$ on NF-UNSW (+16.5%), $C_2$ on NSLKDD (+5.62%), $C_4$ on NF-CSE (+5.24%) and no unknown class on NF-CSE (+4.7%).

Although achieving a high detection rate is important, an IDS detecting every sample as an attack would not be effective. Therefore, it is also necessary to examine the false alarm rate to ensure that the algorithm does not mistakenly identify too many benign samples as attacks (false positives). Looking at Table 6.3, we observe that the best-performing results are perfectly divided between the datasets. Our approach outperforms the baseline in all tests conducted on the CIC IDS and NF-CSE datasets, whereas the baseline outperforms our approach on the other three datasets. While our approach shows a slightly higher false alarm rate compared to the baseline on those datasets, the difference is minimal, between 1 and 2 percent higher at most. This demonstrates that our approach achieves significantly higher detection of attacks (detection rate) while maintaining a reasonably low rate of false detections (false alarm rate).

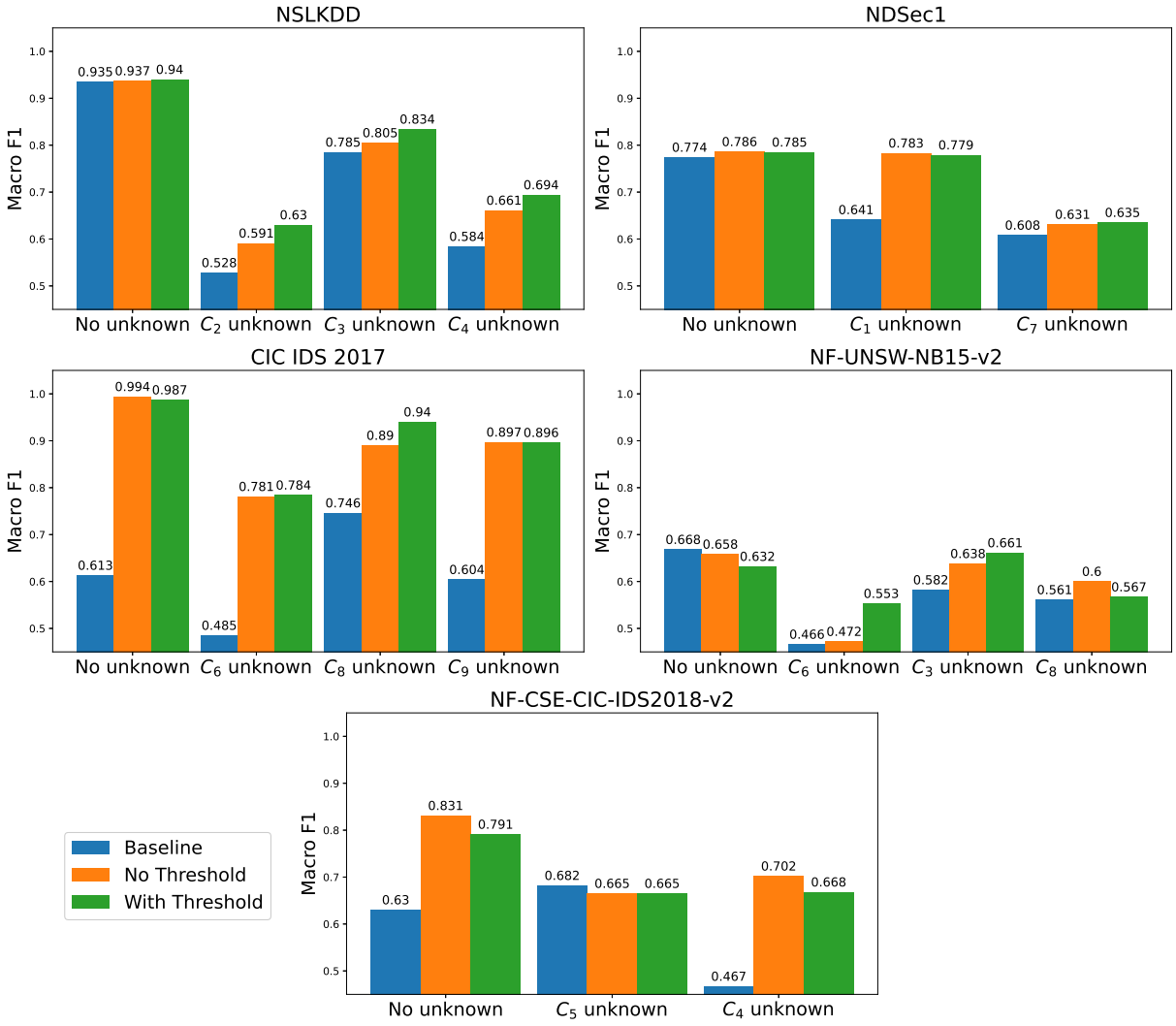



Figure 6.2: Macro $F_1$ results of the baseline and proposed approach on network intrusion datasets

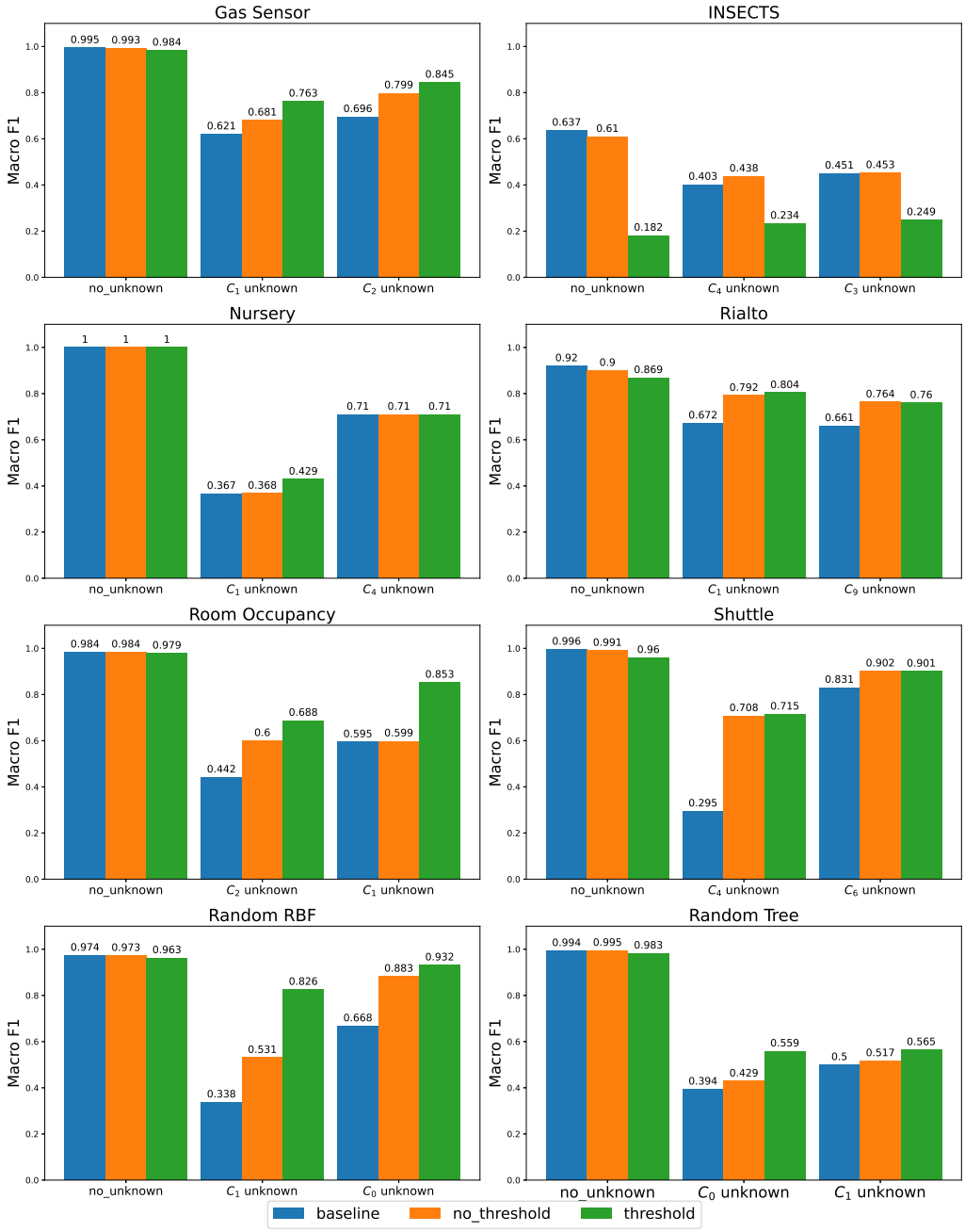



Figure 6.3: Macro $F_1$ results of the baseline and proposed approach on other datasets

Table 6.3: Detection and false alarm rate results on network intrusion datasets (best in boldface, WT = With Threshold, NT = No Threshold)

| **Dataset** | **Unknown Class** | **Detection Rate** | | | **False Alarm Rate** | | |
|---|---|---|---|---|---|---|---|
| | | ***Baseline*** | ***NT*** | ***WT*** | ***Baseline*** | ***NT*** | ***WT*** |
| NSLKDD | None | 0.9992 | **0.9998** | **0.9998** | **0.0007** | 0.0021 | 0.0024 |
| | $C_2$ | 0.5352 | 0.5885 | **0.5914** | **0.0006** | 0.0012 | 0.0010 |
| | $C_3$ | 0.9985 | **0.9989** | **0.9989** | **0.0008** | 0.0018 | 0.0016 |
| | $C_4$ | 0.8198 | 0.8621 | **0.8645** | **0.0002** | 0.0006 | 0.0006 |
| NDSec1 | None | **1** | **1** | **1** | **0.0176** | 0.0296 | 0.0291 |
| | $C_1$ | 0.9962 | 0.9962 | **0.9969** | **0.0137** | 0.0203 | 0.0233 |
| | $C_7$ | 0.9975 | 0.9996 | **0.9999** | **0.0182** | 0.0288 | 0.0318 |
| CIC IDS 2017 | None | 0.9953 | **1** | **1** | 0.0437 | **0.0016** | 0.0025 |
| | $C_6$ | **0.8499** | 0.8112 | 0.8113 | 0.0329 | **0.0004** | **0.0004** |
| | $C_8$ | 0.9680 | 0.9709 | **0.9913** | 0.0077 | **0.0014** | 0.0030 |
| | $C_9$ | 0.9954 | 0.9983 | **0.9984** | 0.0329 | **0.0011** | 0.0023 |
| NF-UNSW-NB15-v2 | None | 0.9943 | 0.9960 | **0.9965** | **0.0038** | 0.0040 | 0.0044 |
| | $C_6$ | 0.8300 | 0.9946 | **0.9950** | **0.0037** | 0.0039 | 0.0043 |
| | $C_3$ | 0.9884 | 0.9919 | **0.9937** | **0.0037** | 0.0039 | 0.0043 |
| | $C_8$ | 0.9944 | 0.9950 | **0.9963** | **0.0038** | 0.0041 | 0.0047 |
| NF-CSE-CIC-IDS2018-v2 | None | 0.9525 | 0.9658 | **0.9995** | 0.0021 | **0.0005** | 0.5341 |
| | $C_5$ | 0.7609 | **0.7855** | 0.7834 | 0.0024 | 0.0095 | **0** |
| | $C_4$ | 0.9470 | 0.9674 | **0.9994** | 0.0339 | **0.0017** | 0.5228 |

### 6.5.3 Detection of Novel Classes

Finally, we conducted a detailed analysis of our approach's capability to detect novel classes and the impact of adding the threshold filters. Table 6.4 lists the precision (denoted as P) and recall (denoted as R) of both configurations in scenarios where an unknown class was excluded from the training data. Precision, in this context, indicates the percentage of samples identified as unknown that are truly unknown, while recall reflects the percentage of unknown samples that were successfully identified out of the total number of unknown samples in the dataset. We do not report on the performance of the baseline since it cannot detect novel classes.

Beginning with the configuration without thresholds, the results generally show relatively high precision and low recall. This indicates that the samples identified as novel by the algorithm are mostly correct (high precision), but the algorithm missed detecting many of these samples (low recall). In our architecture, samples incorrectly classified as known classes contribute to lowering the recall for novel attacks or classes. This motivated the introduction of thresholds on the known classes, which tighten the requirements for assigning samples to known categories and, in turn, allow more samples to be correctly identified as novel.

This hypothesis is confirmed by the results of the second configuration with thresholds, shown in Table 6.4, where the addition of thresholds led to an increase or no decrease in recall across all tested cases. Furthermore, this improvement in recall did not result in a significant decrease in precision; in most cases, precision remained within a few percentage points of the original result, and in some cases, such as CIC IDS $C_8$, Nursery $C_1$, and Room Occupancy $C_1$, it even increased substantially. Out of the 29 combinations of datasets and classes tested, only four showed a more noticeable decrease in precision (greater than 5%). Overall, these results indicate that this configuration improved the detection of novel classes by effectively enhancing recall through the inclusion of thresholds. This conclusion is further supported by the average results across all tested datasets, which show an overall increase of more than 6 percentage points in precision and 19 percentage points in recall.

Some notable cases include those where the model was unable to detect the novel classes (very low recall), specifically classes $C_6$ and $C_9$ on the CIC dataset, classes $C_5$ and $C_4$ on the NF-CSE dataset, and class $C_4$ on the Nursery dataset. For the CIC and Nursery datasets, this behaviour is likely caused by the mixed class distribution and substantial class overlap, which may have led the model to frequently misclassify these samples as belonging to known classes. In the case of the NF-CSE dataset, the inability to detect these novel classes is likely due to the strong class imbalance and the fact that we could not apply SMOTE during preprocessing, unlike for the other datasets, because of technical

constraints.

### 6.5.4 Runtime Performance

As mentioned in Section 6.1, the use of multiple shallow classifiers in our approach not only enables good classification performance but also provides faster training and response times compared to more complex methods. In this section, we demonstrate this advantage by comparing the training and testing times of our approach relative to the baseline. Tables 6.5 and 6.6 present the average training and testing times (in seconds) for the baseline and both configurations of our approach across each dataset. For prediction time, non-cybersecurity datasets were excluded, as their times were too small to provide meaningful insights. Additionally, the tables include the ratio of the baseline[2], indicating how many times longer our approach took compared to the baseline, shown in parentheses.

Examining the training times, we observe that while our approach takes longer, which is expected due to the additional complexity of using multiple classifiers rather than a single one, the difference remains relatively small, with the highest ratio being 1.85 times longer for the NSLKDD dataset. Furthermore, when comparing the two configurations, we observe that incorporating thresholds affects the training time, necessitating additional computations to determine the threshold using the Youden Index. This results in an average ratio of 1.38 times longer than the baseline for the threshold-based configuration, compared to 1.19 times without thresholds. However, given that training is not frequently required and that the maximum training time across all datasets, some with millions of samples, did not exceed 4 minutes, we conclude that our approach remains relatively fast and scalable for training on large datasets.

Looking at the prediction time, which is an important consideration when deploying an IDS in a real-world scenario, our approach is closer to the baseline than during training. The average ratio of our approach is only 1.15 times slower than the baseline without thresholds and 1.14 times slower with thresholds. Since these values are nearly identical, we conclude that the inclusion of thresholds does not significantly impact prediction time, with the difference falling within the margin of error. Prediction time correlates directly with the number of samples to be classified, with larger datasets, such as NF-CSE, containing over 18 million samples, requiring the most time. The prediction times of our algorithm, when compared to the baseline, differ by only a few seconds, demonstrating that our approach is able to maintain comparable speed while providing enhanced classification performance.

[2]Ratio of baseline is defined as follows: Ratio of baseline $= \frac{\text{alternative time}}{\text{baseline time}}$

Table 6.4: Results on unknown classes (best in boldface; P: precision; R: recall)

| Dataset | Unknown Class | No Threshold | | Threshold | |
|---|---|---|---|---|---|
| | | ***P*** | ***R*** | ***P*** | ***R*** |
| NSLKDD | $C_2$ | **0.9902** | 0.0786 | 0.9866 | **0.0943** |
| | $C_3$ | 0.1842 | 0.0648 | **0.3393** | **0.1759** |
| | $C_4$ | **0.9839** | 0.2124 | 0.9811 | **0.3189** |
| NDSec1 | $C_1$ | **0.9930** | 0.6749 | 0.9898 | **0.7370** |
| | $C_7$ | **0.9779** | 0.4429 | 0.9661 | **0.4881** |
| CIC IDS 2017 | $C_6$ | 0.7544 | 0.0038 | **0.8831** | **0.0130** |
| | $C_8$ | 0.0072 | 0.0007 | **0.4720** | **0.7018** |
| | $C_9$ | 0.0062 | **0.0096** | **0.0096** | **0.0096** |
| NF-UNSW-NB15-v2 | $C_6$ | 0.4260 | 0.1608 | **0.6078** | **0.6140** |
| | $C_3$ | **0.4915** | 0.4634 | 0.4293 | **0.6653** |
| | $C_8$ | **0.2328** | 0.9222 | 0.1570 | **0.9790** |
| NF-CSE-CIC-IDS2018-v2 | $C_5$ | 0.0203 | 0.0033 | **0.0235** | **0.0039** |
| | $C_4$ | **0** | **0** | **0** | **0** |
| Gas Sensor | $C_1$ | **0.9320** | 0.0900 | 0.9310 | **0.3640** |
| | $C_2$ | **0.9680** | 0.2400 | 0.9260 | **0.4340** |
| INSECTS | $C_4$ | 0.4740 | 0.1710 | **0.5120** | **0.9410** |
| | $C_3$ | 0.2290 | 0.0470 | **0.3260** | **0.3640** |
| Nursery | $C_1$ | 0 | 0 | **0.9970** | **0.0910** |
| | $C_4$ | **0** | **0** | **0** | **0** |
| Rialto | $C_1$ | **0.6990** | 0.5520 | 0.6750 | **0.7420** |
| | $C_9$ | **0.6460** | 0.4750 | 0.6010 | **0.5990** |
| Room Occupancy | $C_2$ | **0.9960** | 0.3060 | 0.9780 | **0.4840** |
| | $C_1$ | 0.5000 | 0.0020 | **0.9760** | **0.6250** |
| Shuttle | $C_4$ | **0.9990** | 0.5580 | **0.9990** | **0.5920** |
| | $C_6$ | **0.5000** | **0.3000** | 0.4290 | **0.3000** |
| Random RBF | $C_1$ | **0.9910** | 0.3590 | 0.9810 | **0.8380** |
| | $C_0$ | **0.9510** | 0.5710 | 0.8120 | **0.8940** |
| Random Tree | $C_0$ | **0.9840** | 0.0510 | 0.9580 | **0.2280** |
| | $C_1$ | **0.9630** | 0.0310 | 0.9250 | **0.1140** |
| **Average** | | 0.5827 | 0.2342 | **0.6507** | **0.4280** |

Table 6.5: Average training time

| **Dataset** | **Training Time in Seconds (Ratio of Baseline)** | | |
|---|---|---|---|
| | ***Baseline*** | ***No Threshold*** | ***Threshold*** |
| NSLKDD | 2.03 | 3.33 (1.64) | 3.75 (1.85) |
| NDSec1 | 64.64 | 79.11 (1.22) | 101.94 (1.58) |
| CIC IDS 2017 | 112.10 | 136.04 (1.21) | 165.32 (1.47) |
| NF-UNSW-NB15-v2 | 137.62 | 178.09 (1.29) | 224.11 (1.63) |
| NF-CSE-CIC-IDS2018-v2 | 98.44 | 109.36 (1.11) | 138.81 (1.41) |
| Gas Sensor | 0.65 | 0.83 (1.28) | 0.86 (1.33) |
| INSECTS | 0.47 | 0.60 (1.28) | 0.66 (1.40) |
| Nursery | 0.16 | 0.14 (0.88) | 0.15 (0.94) |
| Rialto | 1.39 | 1.68 (1.21) | 1.88 (1.35) |
| Room Occupancy | 0.23 | 0.24 (1.04) | 0.25 (1.09) |
| Shuttle | 0.93 | 0.99 (1.06) | 1.27 (1.37) |
| Random RBF | 0.39 | 0.44 (1.13) | 0.50 (1.28) |
| Random Tree | 0.46 | 0.52 (1.13) | 0.60 (1.30) |
| **Average Ratio of Baseline** | | **1.19** | **1.38** |

Table 6.6: Average prediction time

| **Dataset** | **Prediction Time in Seconds (Ratio of Baseline)** | | |
|---|---|---|---|
| | ***Baseline*** | ***No Threshold*** | ***Threshold*** |
| NSLKDD | 0.24 | 0.32 (1.29) | 0.32 (1.31) |
| NDSec1 | 5.48 | 6.12 (1.12) | 6.23 (1.14) |
| CIC IDS 2017 | 8.83 | 9.98 (1.13) | 9.89 (1.12) |
| NF-UNSW-NB15-v2 | 9.67 | 10.98 (1.13) | 10.74 (1.11) |
| NF-CSE-CIC-IDS2018-v2 | 87.07 | 94.90 (1.09) | 88.41 (1.02) |
| **Average Ratio of Baseline** | | **1.15** | **1.14** |

### 6.5.5 Summary

In summary, the results presented in this section demonstrated that for multi-class classification (cf. Figures 6.2 and 6.3), our approach performed similarly or better than the baseline when no unknown classes were introduced, indicated by a higher Macro $F_1$ score in most cases. With the introduction of a novel class, our approach almost always outperformed the baseline significantly due to its ability to detect the novel class. This highlights the importance of addressing the OSR problem in IDS scenarios, as new attacks may emerge after the model has been trained, which traditional ML models fail to identify.

Analyzing the attack detection rate in Table 6.3, our approach generally outperformed the baseline in all cases but one, showing a higher detection rate with substantial differences in some instances. Additionally, the false alarm rate of our approach remained close to the baseline, sometimes outperforming it and in other cases being only slightly higher. These results demonstrate the efficiency of our approach in correctly detecting attacks in IDS scenarios while maintaining a low false positive rate.

Finally, comparing the two proposed configurations, the results indicated that the approach using thresholds generally outperformed the other in detecting the novel class, as demonstrated by a higher recall in Table 6.4. Furthermore, this approach maintained high precision for samples detected as novel. Although this configuration showed a slight decrease in overall classification performance (Macro $F_1$) when no unknown class was introduced (cf. Figures 6.2 and 6.3), this decrease was not significant. Given the advantageous results when an unknown class is present, we conclude that this approach should be preferred. It is also worth noting that the value of the Youden Index can be adjusted according to the specific data to optimize detection results and minimize the false alarm rate, which was not done for our experiments. In real-world deployments, even a small number of false alarms can often limit an algorithm's adoption, as reviewing them can require significant human resources. We plan to address this limitation in future work by exploring further filtering of these detected samples.

While the results presented in this section demonstrate the effectiveness of our approach in achieving strong performance for detecting novel attacks in the OSR scenario, it is important to note that further hyperparameter tuning is necessary for both the binary classifiers and the Youden Index to optimize performance on specific datasets. The proposed approach can also be extended to support the detection of multiple novel classes rather than a single one, a direction that will be explored in Chapter 7. In addition, although few alternatives in this domain have exactly the same constraints as our approach, a comprehensive comparison with other state-of-the-art methods could serve as valuable alternative baselines, which we plan to address in future work.

## 6.6 Summary

This chapter introduces a novel supervised ML approach for IDSs and other related scenarios that effectively addresses the challenge of detecting novel classes within a supervised learning framework. The proposed architecture employs multiple binary classifiers, each specialized in detecting a specific class, in conjunction with a filtering mechanism. This design maintains a high attack detection rate while keeping false positives low, even in the presence of novel classes. It can also be extended to learn new classes on the fly while remaining fully online. We tested and evaluated the performance of our approach against a baseline in both multi-class classification and novel attack detection scenarios using five contemporary intrusion detection datasets and eight additional datasets from other domains.

The empirical results demonstrate that our approach not only matches or surpasses traditional multi-class classification methods in overall performance when no novel classes are present but also significantly improves detection capabilities when confronted with novel attacks. Furthermore, we compared two different configurations of our architecture. By incorporating threshold filters based on a weighted Youden Index, we were able to enhance the detection of novel attacks while maintaining a relatively simple and efficient design.

This work forms the foundation for the classification module of our proposed ND algorithm, which is outlined in the next chapter. We will extend this proposal to operate on DSs in a fully online context, enabling the differentiation of multiple novel classes, the filtering of novel samples that may represent noise, and a comprehensive comparison of our approach with other state-of-the-art methods in ND.

# Chapter 7

# CASCADE: Clustering-based Adaptive Stream Classification and Detection of Emerging Classes

This chapter focuses on our third and fourth objectives mentioned in Chapter 1 and brings together the different components developed throughout this thesis to address key challenges in ND algorithms identified in Chapter 3. It builds on the evaluation methodology proposed in Chapter 5 and uses the algorithm outlined in Chapter 6. Building on the multi-binary classifier approach introduced and validated in the previous chapter, we propose a more comprehensive ND algorithm named CASCADE (**C**lustering-based **A**daptive **S**tream **C**lassification **A**nd **D**etection of **E**merging classes). Specifically, we extend our previous approach by adding the capability to filter novel samples and detect multiple novel classes through hierarchical clustering and specifically defined meta-features in a completely unsupervised manner. Furthermore, CASCADE incorporates the ability to learn new classes by dynamically adding new classifiers in a fully online DS setting. We evaluate our approach against other state-of-the-art ND techniques across 11 datasets, including multiple cybersecurity benchmarks.

The contents of this chapter are based on the following paper:

Jean-Gabriel Gaudreault and Paula Branco. Cascade: Clustering-based adaptive stream classification and detection of emerging classes. unpublished, To Be Submitted to SIAM International Conference on Data Mining 2026, N.D

## 7.1 Introduction

The continuous generation of large-scale data in domains such as network security, Internet of Things (IoT) devices, sensor networks, and finance has created a need for DS mining, where knowledge is extracted from continuously evolving DSs. Within this domain, ND, the process of identifying previously unknown patterns or classes as data arrives, has emerged as an essential strategy for maintaining the security and efficiency of these systems, particularly where traditional batch learning methods are insufficient. One such key application is IDSs, where new attacks can emerge quickly and the large volume of network packets makes manual labeling impractical. These scenarios make automating the discovery of novel classes a priority.

Despite its relevance, existing ND approaches face key limitations. Many rely on ground-truth labels in the online phase, which are often unavailable or costly to obtain in real-time due to the sheer volume and velocity of data. Current unsupervised methods can struggle to detect multiple novel classes simultaneously, are sensitive to hyperparameter tuning, and may fail to handle the concept drift inherent in evolving DSs. These limitations hinder their deployment in real-world, largely unlabeled DSs.

To address these challenges, we propose CASCADE, an ND algorithm designed for DSs that operates online in a completely unsupervised fashion. CASCADE features a classification module that leverages binary classifiers with threshold filters to separate known and unknown samples, and an ND module that applies hierarchical clustering over carefully selected meta-features to isolate emergent classes. This design enables dynamic adaptation without manual intervention or labeled feedback.

**Contributions** The main contributions of this chapter can be summarized as follows:

(i) propose a novel unsupervised architecture for ND in DSs that operates online without any ground-truth labels;

(ii) introduce a hierarchical clustering-based mechanism leveraging meta-features to detect and characterize novel classes;

(iii) perform an extensive empirical evaluation across 11 datasets, including cybersecurity benchmarks, demonstrating that CASCADE achieves state-of-the-art performance among unsupervised methods, often matching or exceeding supervised techniques, while requiring minimal hyperparameter tuning.

**Organization** The rest of this chapter will be structured as follows: Section 7.2 goes over existing methodologies for ND; Section 7.3 details the proposed CASCADE architecture; Section 7.4 describes our experimental setup, presents and discusses the results; and Section 7.5 concludes the chapter and outlines directions for future work.

## 7.2 Related Work

Various algorithms have been proposed for the task of ND in DSs, differing significantly in how they classify known classes and detect novel ones.

The most straightforward approach treats ND as a one-class problem, similar to anomaly detection, where only a single known class is considered and the algorithm can detect at most one novel class. OLINDDA [201] is a typical example of this strategy. It trains a k-means classifier on offline data of the known class, and the distance between incoming samples and their cluster centroids is used to determine the decision boundary.

Building on this idea, algorithms such as ECSMiner [147] and CLAM [13] extend detection to multiple known classes by constructing a class-based ensemble using the offline data. However, these approaches suffer from two major limitations. First, they rely on the assumption that ground-truth labels will eventually become available, a condition that is often unrealistic in common DS applications, such as sensor monitoring or network intrusion detection, where a large quantity of samples needs to be classified quickly. Second, they can only indicate the presence of novel classes but cannot distinguish multiple novel classes if they appear simultaneously before one has been confirmed. ECHO [114] partially alleviates the labeling requirement by requesting labels only for samples with low confidence scores, but it still shares the same fundamental limitations.

MINAS, proposed by Faria et al. [56], addresses many of these shortcomings by offering an unsupervised method capable of classifying multiple known classes and detecting multiple novel classes concurrently in the stream. It achieves this by training a clustering algorithm for each offline class subset to create micro-clusters, which collectively form the class decision boundary. While MINAS mitigates several limitations of prior methods, it is highly sensitive to its numerous hyperparameters due to its reliance on unsupervised clustering during the offline phase.

Our proposed method, CASCADE, builds upon these advances. It retains the advantages of state-of-the-art methods like MINAS, such as multi-class recognition, multiple novel class detection, and unsupervised streaming operation, while reducing hyperparam-

eter sensitivity and improving overall performance by incorporating supervised algorithms in the offline phase.

## 7.3 CASCADE

To address the challenge of identifying novel classes in evolving DS, which are often affected by concept drift, we propose a new architecture tailored to the task of ND. Our approach is designed to overcome common limitations in the field, including the inability to detect more than one novel class at a time, reliance on ground-truth labels, and sensitivity to hyperparameter tuning.

### 7.3.1 Top-Level Architecture

Our proposed architecture consists of two primary modules: a classification module and an ND module, with a high-level overview shown in Figure 7.1. The classification module, trained on offline data, either assigns incoming samples to known classes or forwards unknown samples to the buffer of the ND module. Once the buffer reaches a threshold, it is analyzed to detect potential novel classes. When a novel class is identified, the model is dynamically updated to integrate the new class, enabling the classifier to recognize future instances of that class in the DS.

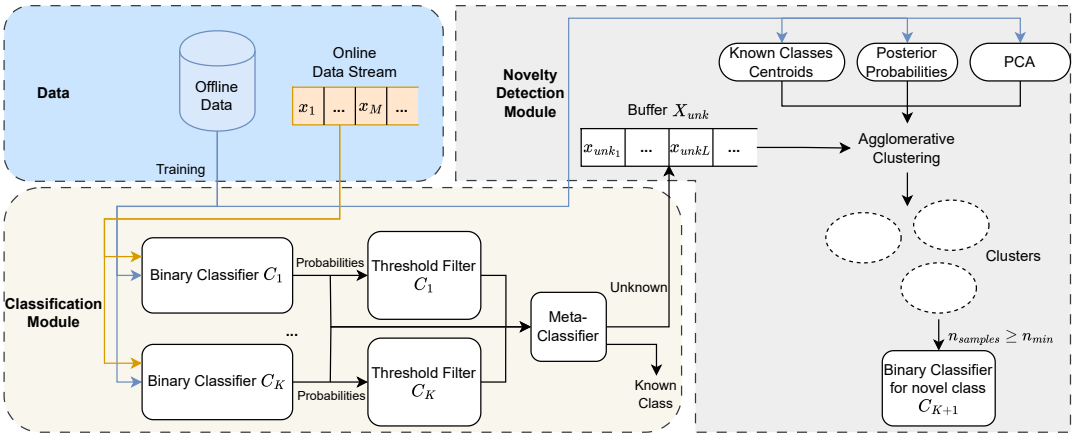



Figure 7.1: Top-level view of our proposed architecture

### 7.3.2 Classification Module

The classification module consists of three components: binary classifiers, threshold filters, and a meta-decision layer. This design allows us to leverage the typically high accuracy of supervised learners trained on labeled data, while maintaining a low FPR through the use of threshold filters. Additionally, the modular nature of the architecture enables the dynamic integration of new classifiers in an online fashion as novel classes are detected. Although this modularity also allows for updating existing classifiers with newly observed samples to adapt to concept drift, we do not employ this option in this work due to the performance degradation observed in most cases.

#### Binary Classifiers

The first component of the classification module employs a one-vs-rest strategy: it contains $K$ independent binary classifiers, each trained to recognize a single class while rejecting all others. For a subset of the data $\{(x_i, y_i)\}_{i=1}^{M}$ available in the offline phase, the optimization problem for the classifier with parameters $\theta_k$ associated with class $k$ can be defined as:

$$\theta_k^\star \;=\; \arg\min_{\theta_k} \frac{1}{M} \left[ \sum_{i=1}^{M} \ell\big(y_{ik},\; s_k(x_i)\big) \;+\; \lambda\, \Omega(\theta_k) \right] \tag{7.1}$$

where $y_{ik} = 1$ if $y_i = k$ and 0 otherwise, $s_k(x_i)$ is the decision score produced by classifier $k$, $\ell$ represents a point-wise binary loss function, $\Omega(\theta_k)$ is a regularization term and $\lambda > 0$ controls the strength of the regularization. While any learning algorithm that outputs calibrated posterior probabilities can be used within this framework, we found that ensemble methods to be particularly suitable due to their strong predictive performance and rapid training. Among these, gradient boosting algorithms, and LightGBM in particular, performed well in our experiments while also offering high computational speed and low memory usage [125]. Consequently, we use LightGBM in our evaluations.

It is also worth noting that although the low memory usage of LightGBM, combined with the relatively high memory capacity of modern computing systems, makes running out of memory unlikely in most practical scenarios, each classifier has a counter recording the number of samples processed since it last classified a sample as belonging to its class. In constrained environments, or in scenarios involving a theoretically unbounded number of novel classes, this counter could be used to remove the oldest or least recently used classifiers in order to remain within memory constraints.

### Threshold Filters

To reduce the number of false positives, each classifier is paired with a corresponding threshold filter that adds a layer of certainty before confirming a sample as belonging to the classifier's target class. The threshold is applied directly to the classifier's probability output and can be determined using one of two tunable methods, depending on the problem setting: (i) by computing the average of the probability outputs for that classifier over the training set; or (ii) by using the Youden Index [233] to dynamically identify the threshold that optimizes the trade-off between TPR and FPR.

Although the standard Youden Index assigns equal weight to TPR and FPR, our implementation of threshold filters is designed to ensure that each binary classifier only declares a sample as belonging to its class when it is highly confident. To prioritize reducing false detections (FPR), we introduce a weighted Youden Index with a tunable parameter ($w$) that adjusts the relative importance of minimizing FPR versus maximizing TPR, defined as: $J_w = TPR + (w(1 - FPR))$.

To formalize our threshold strategy, let $\sigma$ denote a mapping function that converts a classifier's raw score $s_k$ into a probability $p_k \in [0, 1]$ such that $\hat{p}_k = \sigma(s_k(x; \theta_k^\star))$. To determine if a sample should be accepted by classifier $k$, we then compare its output probability $\hat{p}_k(x)$ against a class-specific threshold $\tau_k$. As mentioned previously, we evaluated two different strategies for the selection of the threshold, defined as:

$$\tau_k = \frac{1}{M} \sum_{i=1}^{M} \hat{p}_k(x_i) \text{ (mean)} \tag{7.2}$$

$$\tau_k = \arg\max_{\tau \in \mathcal{T}_k} J_w^k(\tau) \text{ (weighted Youden Index)} \tag{7.3}$$

In the first case, $\tau_k$ is simply set to the mean probability assigned by classifier $k$ to its training samples, serving as a basic baseline. In the second case, $\tau_k$ is obtained by maximizing the Youden Index $J_w^k(\tau)$ over the set of candidate thresholds $\mathcal{T}_k = \{\hat{p}_k(x_i)\}_{i=1}^M$, which corresponds to the classifier's output on the training set.

Once the threshold $\tau_k$ is determined, the final decision function for a classifier $k$ is defined as:

$$f_k(x) = \mathbf{1}\{\hat{p}_k(x) \geq \tau_k\} \tag{7.4}$$

#### Meta-Decision Layer

Finally, the classification module includes a meta-decision layer that integrates the outputs from all binary classifiers to produce a single class label. Specifically, if at least one binary classifier identifies the input sample as belonging to its class, the meta-decision layer determines the most appropriate known class. This meta-layer allows the system to capture relationships between the binary classifiers' probabilities, increasing classification accuracy for samples identified as known. To achieve this, we employ a meta-classifier trained on a rich feature space, which includes both the posterior probabilities provided by the binary classifiers and the features of the original sample. If none of the binary classifiers identify the sample as belonging to their respective classes, the meta-decision layer classifies the sample as unknown.

Formally, let $\hat{p}(x) = [\hat{p}_1(x), \ldots, \hat{p}_K(x)]^\top$ be the vector of probability outputs from all $K$ classifiers. The meta-classifier receives as input the concatenation of the raw input $x$ as well as the probability outputs $\hat{p}(x)$, forming a combined feature vector $z(x) = \text{concat}(x, \hat{p}(x))$. This allows the model to learn higher-order decision boundaries that capture interactions between the input features and classifier outputs. This method is particularly useful in cases where confidence alone is insufficient to resolve class ambiguity or when classifier outputs interact in complex, nonlinear ways. Formally, the predicted class $\hat{y}_{meta}$ is defined as:

$$\hat{y}_{meta} = \begin{cases} g(z(x); \phi^\star) & \text{if } C^+(x) \neq \emptyset \\ -1 \text{ (unknown)} & \text{otherwise} \end{cases} \tag{7.5}$$

where $g(\cdot; \phi)$ is a multi-class model trained on the augmented dataset $\{(z(x_i), y_i)\}_{i=1}^{M}$, $\phi^\star$ denotes the optimal parameters learned during training, and $C^+(x)$ represents the set of classifiers that consider $x$ to be inside their decision boundary.

### 7.3.3 Novelty Detection Module

All samples classified as unknown by the classification module are forwarded to the ND module, where they are temporarily stored in a buffer ($X_{unk}$). Once the buffer accumulates a user-defined number of samples, i.e. when $L > L_{\min}$, the ND is triggered. The buffer is then analyzed using the procedure described below to identify potential clusters of samples forming NPs, with the goal of detecting emerging novel classes. To ensure timely processing, reduce noise, and manage memory efficiently, samples that remain in the buffer beyond a user-defined time window are discarded.

To facilitate the detection of NPs in the unknown samples, we define three meta-features designed to capture complementary and intrinsic aspects of the unknown samples in relation to the known classes. These meta-features were derived through empirical experimentation, during which we identified representations that enable clear discriminative boundaries between classes. For each unknown sample $x_j \in X_{unk}$, the following meta-features are extracted and defined as:

- **Posterior probabilities**: the estimated probabilities that the sample belongs to one of the known classes, given by $\hat{p}(x_j) = [P(y = 1|x_j), \ldots, P(y = K|x_j)]$
- **Distances to class centroids:** the Euclidean distances between the sample and the centroid of each known class, defined as $d(x_j) = [d_{j1}, \ldots, d_{jK}]$ with $d_{jk} = \|x_j - \mu_k\|_2$
- **Principal component representation:** the projection of the sample onto a lower-dimensional subspace using a PCA function pre-fitted on the offline training data, denoted by $v_j = f_{PCA}(x_j)$

Let $\Psi$ denote the composite feature set obtained by concatenating the meta-feature vectors for all unknown samples. Specifically, this includes the posterior probabilities $\hat{P} = \{\hat{p}(x_j)\}_{j=1}^{L}$, the distances to known class centroids $D = \{d(x_j)\}_{j=1}^{L}$, and the PCA projections $V = \{v_j\}_{j=1}^{L}$. Agglomerative hierarchical clustering [162] is then applied to this combined feature representation to group similar samples based on their relationships to known classes. This clustering method was selected due to its ease of tuning and small hyperparameter space compared to many commonly used alternatives [158], which aligns with our objective of limiting the sensitivity of our proposed algorithm to hyperparameters.

To isolate potential novel patterns, we retain only those clusters ($\mathcal{C}l$) that contain at least a predefined minimum number of samples, denoted $n_{\min}$. The resulting set of candidate clusters considered as NPs is defined as:

$$X_{nov} = \{\, \mathcal{C}l \in \text{AggloClust}(\Psi) \mid |\mathcal{C}l| \geq n_{\min}\}. \tag{7.6}$$

Each cluster in $X_{nov}$ serves as the basis for constructing a new class. For every such cluster, a new binary classifier is trained using the corresponding cluster's samples as the positive class and samples from the known classes as the negative class. This classifier is then integrated into the classification module, expanding the system's capacity to recognize and classify instances of the newly discovered class in future data.

## 7.4 Experimental Evaluation

### 7.4.1 Datasets

A total of 11 datasets were used in our evaluation, including 2 synthetic and 9 real-world datasets. Of these, three state-of-the-art intrusion detection datasets: NDSec1 [23], NF-UNSW-NB15-v2 [184], and NSLKDD [208] were specifically selected to evaluate the performance of our algorithm in this critical application domain. Intrusion detection represents a particularly suitable use case for ND, as detection systems must continuously adapt to emerging, previously unseen threats.

The remaining real-world datasets are taken from the USP DS Repository [198], while the two synthetic datasets were generated using the MOA framework [26]. An overview of the key characteristics of all datasets is provided in Table 7.1.

For the NDSec1 dataset, which contains a large number of distinct labels with relatively few samples per label, an additional preprocessing step was introduced to reduce this granularity. For example, the various types of DDoSs attacks were merged into a single label. This process was applied to all attack categories to reduce the number of classes to fewer than 10.

Table 7.1: Overview of the main characteristics of the datasets

| Dataset | Classes | Features | Samples |
|---|---|---|---|
| Gas Sensor | 6 | 128 | 13910 |
| INSECTS-Abrupt (Balanced) | 6 | 33 | 52848 |
| Nursery | 5 | 27 | 12960 |
| Rialto | 10 | 27 | 82250 |
| Room Occupancy | 4 | 16 | 10129 |
| Shuttle | 7 | 8 | 58000 |
| Random RBF (Synt.) | 4 | 10 | 100000 |
| Random Tree (Synt.) | 4 | 12 | 100000 |
| NDSec1 | 10 | 65 | 1384626 |
| NF-UNSW-NB15-v2 | 10 | 42 | 2390275 |
| NSLKDD | 5 | 124 | 148517 |

### 7.4.2 Baseline Methods

We tested our approach against four state-of-the-art approaches in ND, namely ECHO [114], MINAS [56], ECSMiner [147], and ECSMinerWF, an adaptation of ECSMiner without external feedback [56]. It is important to note that among these algorithms, ECSMiner and ECHO are supervised methods that require access to ground-truth labels during processing. ECSMiner assumes full label availability, while ECHO is designed to operate in a semi-supervised fashion, with partial labeling [93].

As previously discussed, access to ground-truth labels in a DS is often unrealistic due to the high arrival rate of samples and the limited availability of timely annotations in many scenarios. Regardless, we included these supervised methods in our comparison as a point of reference and to demonstrate the robustness of our unsupervised approach, even when other methods have access to ground truth labels in the DS.

### 7.4.3 Evaluation

We selected three primary metrics to evaluate the performance of the algorithms in our experiments, following the approach described in Chapter 5 [94]. The first is $M_1$ (i.e., $M_\beta$ with $\beta = 1$), which measures the algorithm's ability to distinguish between known and novel classes; $M_1$ is the harmonic mean of the inverse of both the misclassification rate of novel samples as known ($M_{\text{new}}$) and the misclassification of known samples as novel ($F_{\text{new}}$), giving an equal weight to both when $\beta = 1$. However, $M_1$ only evaluates the model's capacity for binary separation and does not account for misclassifications within the known or novel classes.

To address this limitation, we also employ AMI, which assesses the quality of clustering by measuring the agreement between predicted clusters and ground-truth class labels. AMI adjusts for chance alignment and accounts for the complexity of the clustering structure, producing values close to 0 for random clustering, which makes it particularly suitable for evaluating ND performance in multi-class settings.

In addition, to assess the statistical significance of performance differences in both $M_1$ and AMI across all algorithms and datasets, we conducted Friedman tests followed by Nemenyi post-hoc tests when appropriate [58]. Two evaluation scenarios were considered: one comparing CASCADE exclusively against other unsupervised methods, and another comparing it against all methods, including supervised approaches.

Finally, we also compute TTD, which measures the number of samples required for each novel class to be detected as an NP after its first occurrence in the DS [94]. This

metric provides insight into whether the algorithm can correctly detect and isolate a novel class into its own cluster, as well as how quickly it achieves this detection. In scenarios such as intrusion detection, rapid identification of new attack types is critical, making it highly relevant.

### 7.4.4 Methodology

Using the methodology proposed in Chapter 5 [94] to transform our datasets into DSs suitable for ND, we first define a consistent set of baseline parameters applied across all experiments. Specifically, the time between novel class appearances is set to a factor of 0, we use a 30% ratio of offline samples, a 55% ratio of known classes, and a concept sparsity value of 1, indicating that all classes have an equal probability of appearing throughout the stream. These settings were chosen for their prevalence in prior work and for their ability to create challenging yet realistic ND evaluation scenarios.

To generate a DS from each dataset, we begin by identifying the known classes as the top 55% of classes with the largest number of samples. The dataset is then partitioned into offline and online subsets, with the offline subset containing only samples from the selected known classes.

For each dataset-algorithm pair, we perform hyperparameter tuning using only the offline subset to avoid data leakage. The number of known classes used during tuning is computed using the same 55% ratio, applied to the classes present in the offline subset.

In datasets with fewer than 200,000 samples, we exhaustively evaluate all possible combinations of known and unknown classes according to the computed number of known classes. For larger datasets, due to computational limits, we evaluate a single combination by selecting the known classes with the highest number of samples. Each combination is further split into sub-offline and sub-online sets for tuning.

Hyperparameter tuning is conducted via grid search, as outlined in Algorithm 7.1, over a predefined set of configurations selected based on recommendations from the original papers and preliminary empirical tests. Notably, CASCADE requires only limited tuning: four configurations (two values for two hyperparameters) are tested, compared to 24, 9, 9, and 72 configurations for ECHO, ECSMiner, ECSMinerWF, and MINAS, respectively. All tested configurations, final hyperparameters and source code for reproducibility are available online[1]. The optimal configuration is selected based on the average performance across all combinations, using as the evaluation metric the harmonic mean of the average $M_1$ and

[1][https://anonymous.4open.science/r/CASCADE-387F/]

AMI scores. This metric considers equally the algorithm's ability to distinguish known from novel classes ($M_1$) and to minimize inter-class misclassification within both subsets (AMI). This configuration is then applied to the online subset to assess the algorithm's final performance on the full DS.

**Algorithm 7.1** Hyperparameter tuning procedure

**Input**: Offline dataset $D_{\text{off}}$, known class ratio $n$, offline sample ratio $r$, sample threshold $T$
**Parameter**: Hyperparameter grid $\Theta$
**Output**: Best hyperparameters $\theta^*$
$C_{\text{avail}} \leftarrow$ all unique classes in $D_{\text{off}}$
$K_{\text{avail}} \leftarrow |C_{\text{avail}}|$
$K_{\text{known}} \leftarrow \lceil K_{\text{avail}} \times n \rceil$
**if** $|D_{\text{off}}| < T$ **then**
    $\mathcal{S} \leftarrow \left\{ (C_{\text{known}},\, C_{\text{avail}} \setminus C_{\text{known}}) \;\middle|\; C_{\text{known}} \subseteq C_{\text{avail}},\ |C_{\text{known}}| = K_{\text{known}} \right\}$
**else**
    $C_{\text{known}} \leftarrow$ top $K_{\text{known}}$ classes in $D_{\text{off}}$ by sample count
    $\mathcal{S} \leftarrow \{(C_{\text{known}},\, C_{\text{avail}} \setminus C_{\text{known}})\}$
**end if**
bestScore $\leftarrow -\infty$
**for all** $(C_{\text{off}}, C_{\text{on}}) \in \mathcal{S}$ **do**
    Split $D_{\text{off}}$ into sub-offline $D'_{\text{off}}$ and sub-online $D'_{\text{on}}$ using ratio $r$
    **for all** $\theta \in \Theta$ **do**
        score $\leftarrow$ Evaluate$(D'_{\text{off}}, D'_{\text{on}}, \theta)$
        **if** score $>$ bestScore **then**
            bestScore $\leftarrow$ score
            $\theta^* \leftarrow \theta$
        **end if**
    **end for**
**end for**
**return** $\theta^*$

### 7.4.5 Results

In this section, we present and analyze the results of our proposed algorithm alongside the baseline methods previously described, evaluated across all datasets. Specifically, we

assess each algorithm's ability to: (i) distinguish between known and novel classes ($M_1$); (ii) accurately classify multiple classes, as captured by AMI; and (iii) detect novel classes as NPs, including the time required for their detection (TTD).

Table 7.2: Comparison of $M_1$ across all algorithms. The best overall result per dataset is in **bold**, and the best in each category (Supervised vs. Unsupervised) is underlined. Higher values are better.

| Dataset | Supervised | | Unsupervised | | |
|---|---|---|---|---|---|
| | ECHO | ECSM | ECSMWF | MINAS | CASCADE |
| Random Tree | **0.69** | 0.47 | 0.02 | 0.10 | 0.63 |
| Random RBF | **0.94** | 0.30 | 0.02 | 0.60 | 0.74 |
| Nursery | 0.62 | 0.51 | **0.74** | 0.60 | 0.00 |
| Gas Sensor | 0.38 | 0.47 | 0.02 | 0.52 | **0.59** |
| INSECTS-Abrupt | **0.63** | 0.36 | 0.04 | 0.30 | 0.52 |
| Rialto | **0.41** | 0.32 | 0.18 | 0.23 | 0.19 |
| Room Occupancy | 0.51 | 0.10 | 0.02 | 0.18 | **0.64** |
| Shuttle | 0.00 | 0.30 | 0.00 | 0.37 | **0.67** |
| NSLKDD | **0.85** | 0.03 | 0.00 | 0.34 | 0.29 |
| NDSec1 | 0.00 | 0.52 | 0.00 | 0.19 | **0.56** |
| NF-UNSW-NB15-v2 | **0.78** | 0.00 | 0.00 | 0.09 | 0.12 |
| Average | **0.53** | 0.31 | 0.09 | 0.32 | 0.45 |

### Performance on Novel Class Detection

We begin by evaluating the ability of each algorithm to differentiate novel from known classes, focusing on the misclassification of novel samples as known and vice versa. As presented in Table 7.2, our proposed approach, CASCADE, consistently ranks among the top-performing unsupervised methods, achieving the best performance in 8 out of 11 datasets. On average, CASCADE surpasses other unsupervised baselines, achieving a mean $M_1$ score 0.13 higher than the next best unsupervised algorithm. Specifically, MINAS achieves an average $M_1$ score of 0.32 across all datasets while CASCADE achieves 0.45.

Furthermore, CASCADE delivers the highest overall $M_1$ score in four datasets (Gas Sensor, Room Occupancy, Shuttle, and NDSec1), outperforming even supervised methods despite not having access to ground-truth labels during the online phase. The Friedman

Table 7.3: Comparison of AMI across all algorithms. The best overall result per dataset is in **bold**, and the best in each category (Supervised vs. Unsupervised) is underlined. Higher values are better.

| Dataset | Supervised | | Unsupervised | | |
|---|---|---|---|---|---|
| | ECHO | ECSM | ECSMWF | MINAS | CASCADE |
| Random Tree | 0.22 | 0.20 | 0.16 | 0.15 | **0.49** |
| Random RBF | **0.81** | 0.69 | 0.67 | 0.68 | 0.76 |
| Nursery | 0.40 | 0.33 | 0.36 | 0.38 | **0.59** |
| Gas Sensor | 0.35 | 0.35 | 0.30 | 0.33 | **0.44** |
| INSECTS-Abrupt | 0.34 | 0.35 | 0.35 | 0.27 | **0.37** |
| Rialto | 0.09 | 0.13 | 0.12 | **0.17** | 0.05 |
| Room Occupancy | 0.31 | 0.18 | 0.15 | 0.38 | **0.59** |
| Shuttle | **0.78** | 0.50 | 0.42 | 0.31 | 0.38 |
| NSLKDD | 0.59 | 0.61 | **0.63** | 0.35 | 0.07 |
| NDSec1 | 0.01 | 0.05 | 0.01 | **0.05** | 0.02 |
| NF-UNSW-NB15-v2 | **0.47** | 0.01 | 0.01 | 0.38 | 0.20 |
| Average | **0.40** | 0.31 | 0.29 | 0.31 | 0.36 |

Table 7.4: Average rankings from Friedman test across all datasets (lower is better). Best results per column are in **bold**.

| | Unsupervised Only | | All Methods | |
|---|---|---|---|---|
| **Algorithm** | $M_1$ **Rank** | **AMI Rank** | $M_1$ **Rank** | **AMI Rank** |
| **CASCADE** | **1.36** | **1.64** | 2.18 | **2.45** |
| MINAS | 1.82 | 2.00 | 2.91 | 3.27 |
| ECSMinerWF | 2.82 | 2.36 | 4.50 | 3.82 |
| ECSMiner | - | - | 3.32 | 2.91 |
| ECHO | - | - | **2.09** | 2.55 |
| $p$-value | 0.002 | 0.234 | 0.002 | 0.237 |

test results reported in Table 7.4 confirm that statistically significant differences exist among algorithms for the $M_1$ metric in both the unsupervised-only comparison and the comparison including all methods, with $p < 0.05$ in both cases. Among unsupervised approaches, CASCADE achieves the best average rank of 1.36. When considering all methods, CASCADE also attains a competitive average rank of 2.18, close to ECHO, which achieves the best rank of 2.09.

Since the Friedman tests indicate statistically significant differences, we further conducted Nemenyi post-hoc pairwise comparisons. The corresponding critical difference diagrams for the unsupervised-only and all-methods scenarios are shown in Figures 7.2 and 7.3, respectively. In these diagrams, horizontal red bars connect algorithms whose performance differences are not considered statistically significant under the conservative Nemenyi criterion, with $p \geq 0.05$. In the unsupervised comparison shown in Figure 7.2, CASCADE and MINAS both perform significantly better than ECSMinerWF, while remaining statistically comparable to each other, despite CASCADE achieving a higher average rank.

Similarly, in the comparison including both supervised and unsupervised methods, the Nemenyi test in Figure 7.3 shows that ECHO, CASCADE, MINAS, and ECSMiner form a group of statistically comparable algorithms. However, CASCADE and ECHO form a subgroup with substantially higher ranks and are not clustered with ECSMinerWF, with which MINAS and ECSMiner are grouped. These results highlight CASCADE competitiveness against supervised approaches in distinguishing between novel and known classes although it does not have access to true labels during the stream.

A closer examination of individual datasets highlights cases where CASCADE significantly outperforms other algorithms. For example, on the Shuttle dataset, it achieves an $M_1$ score of 0.67, 0.3 higher than the second-best result. Similarly, our approach shows robust performance on the Gas Sensor and Room Occupancy datasets. However, it underperforms on the Nursery dataset, where it fails to detect any novel classes, resulting in an $M_1$ score of 0. Interestingly, ECSMinerWF, which performs poorly on most datasets, achieves the best score on this particular dataset. This anomaly may suggest that Nursery exhibits characteristics such as class overlap or imbalanced distributions that hinder the performance of some ND algorithms while favoring others.

CASCADE also performs strongly on cybersecurity datasets, where it achieved the best overall score on NDSec1, the best unsupervised result on NF-UNSW, and an average score on NSLKDD showing that it is able to detect novel network attacks in a DS. Furthermore, on INSECTS-Abrupt, CASCADE achieves a score of 0.52, demonstrating its ability to maintain reliable performance in the presence of concept drift.

Overall, these results demonstrate that CASCADE effectively distinguishes between

novel and known classes, often outperforming or closely matching state-of-the-art ND techniques. Apart from one notable exception (Nursery), CASCADE delivers consistently strong performance across diverse datasets. In addition, it substantially narrows the performance gap between unsupervised and supervised methods while requiring minimal hyperparameter tuning. These characteristics make CASCADE particularly well-suited for real-world DS applications, where access to ground-truth labels is limited and adaptability is essential.

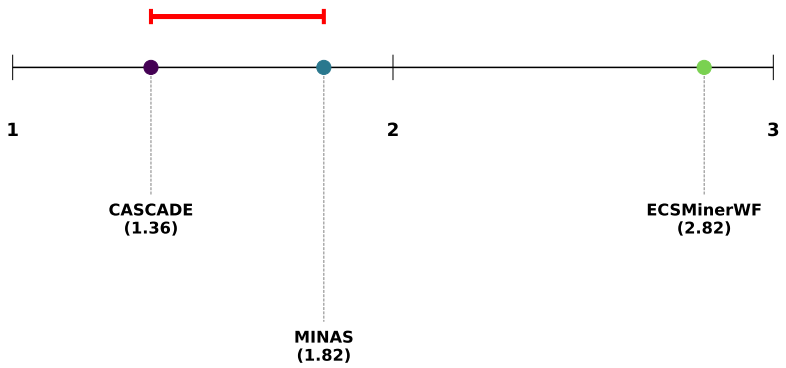



Figure 7.2: Critical difference diagram comparing the rankings of unsupervised methods based on the $M_1$ metric.

### Multi-Class Misclassification

We now look at the results of the AMI metric in Table 7.3, which evaluates inter-class misclassification across both known and novel classes. In scenarios where it is not only important to distinguish between known and novel classes but also to correctly identify the specific class within these categories, algorithms with access to ground-truth labels have a clear advantage. Access to ground-truth labels enables the construction of more precise decision boundaries for both emerging and known classes. In contrast, fully unsupervised algorithms may suffer from gradual performance degradation due to the absence of feedback from labeled data.

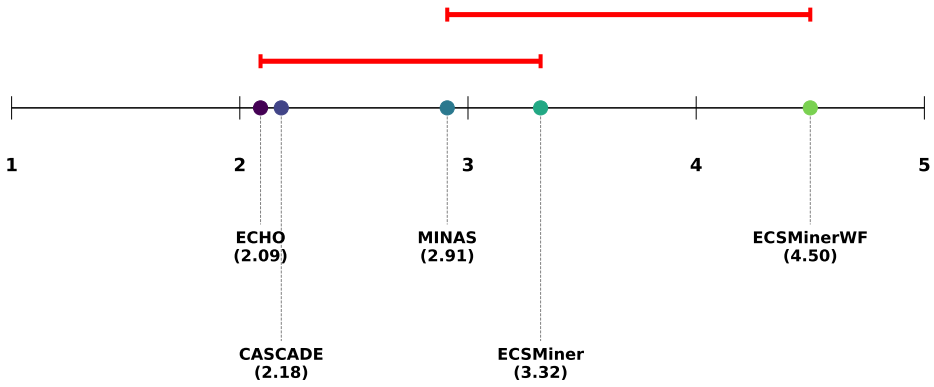



Figure 7.3: Critical difference diagram comparing the rankings of all evaluated methods based on the $M_1$ metric.

Despite this limitation, CASCADE demonstrates highly competitive performance, achieving the highest AMI score on five datasets (Random Tree, Nursery, Gas Sensor, INSECTS-Abrupt, and Room Occupancy), outperforming both supervised and unsupervised baselines. It also ranks as the best-performing unsupervised method in six datasets and achieves the highest average AMI among unsupervised techniques, with a mean score of 0.36, surpassing MINAS, the next best unsupervised method, which averages 0.31. While the supervised algorithm ECHO still achieves the highest overall AMI (0.40), CASCADE significantly narrows the performance gap, trailing by only 0.04 points, despite operating without access to ground-truth labels.

Examining the Nursery dataset, where CASCADE previously underperformed in terms of $M_1$, we observe an interesting contrast. While CASCADE struggled to correctly identify novel classes, as previously discussed, it achieved the highest AMI score among all algorithms, indicating strong performance in classifying known classes. Conversely, in the cybersecurity datasets, particularly NDSec1, NF-UNSW, and NSLKDD, CASCADE performs well in detecting novel attacks (as reflected by $M_1$) but struggles with accurate class assignments within these attacks. It achieves only average AMI scores for NDSec1 and NF-UNSW, and a notably low score for NSLKDD. This suggests a potential limitation of the algorithm in scenarios involving significant overlap between fine-grained classes.

The Friedman test results for the AMI metric (cf. Table 7.4) indicate that CASCADE achieves the best average rank in both the unsupervised comparison (1.64) and global comparison (2.45). However, the corresponding $p$-values exceed 0.05, indicating that the variability across datasets is too high to claim statistical significance. Consequently, post-hoc tests were not conducted. This outcome reflects the mixed performance observed on certain datasets, particularly some cybersecurity benchmarks and the Rialto dataset. Nonetheless, CASCADE still demonstrates strong overall performance, consistently achieving top ranks in the Friedman analysis and delivering excellent results on datasets with other challenging data characteristics, such as achieving the best result on INSECTS, which includes concept drift.

### Time To Detection

Finally, Table 7.5 presents the number of samples required to detect each class as an NP, as measured by the TTD. Empty cells indicate that the corresponding class was not clustered as a novel class by the algorithm. It is important to clarify that this does not imply the algorithm failed to recognize the samples as unknown; rather, it did not group them into a distinct novel class. ECSMinerWF is excluded from the table, as it does not support distinguishing between different novel classes in its output.

Examining trends in the speed of recognizing each class as an NP, MINAS stands out, detecting 16 classes while maintaining a low overall mean TTD of 260, likely due to its frequent ND cycle. It achieves the fastest detection times on the Gas Sensor, NF-UNSW, Room Occupancy, and Shuttle datasets. CASCADE, while detecting slightly fewer classes (13), demonstrates strong detection speed and outperforms the supervised methods on several datasets, including Gas Sensor, NSLKDD, Rialto, and Room Occupancy. Notably, CASCADE achieves the fastest overall mean TTD of 131 on the classes it successfully detects. In contrast, both supervised methods exhibit substantially slower detection, with mean TTD values around 400, indicating delayed recognition of novel classes.

Detecting an NP from a small number of samples can be advantageous in certain scenarios, especially when novel class instances are sparse and risk being discarded as noise before forming a novel class. Early detection in these cases helps preserve rare but critical events. However, this often introduces a tradeoff with classification accuracy, as highlighted in the earlier discussion of the $M_1$ and AMI metrics. Detecting a class too early can lead to unstable cluster formation, poor boundary definitions, and increased inter-class confusion, ultimately reducing classification performance.

Examining the classes that were not clustered as NPs, we observe that none of the

classes from NDSec1, as well as classes 6 and 7 from Shuttle, and class 2 from Nursery, were detected. This may be attributed to their very small size or significant overlap with known classes, which can hinder the ability of ND algorithms to form distinct clusters. Furthermore, some classes are identified by only a single algorithm. For instance, Shuttle class 2 is detected exclusively by MINAS, while NSLKDD class 3 is detected solely by CASCADE. Similarly, Nursery class 4 and Random Tree class 1 are only detected by supervised algorithms and not by any unsupervised method.

These cases highlight the variability in sensitivity across algorithms and suggest that certain classes may require more nuanced detection strategies to be reliably identified as NPs, particularly when class boundaries are ambiguous or class frequency is low.

## 7.5 Summary

This chapter introduces CASCADE, an unsupervised algorithm for ND in DSs, designed to overcome limitations of existing methods, including reliance on ground-truth labels in the DS and sensitivity to hyperparameter tuning. CASCADE integrates an adaptive classification module based on binary classifiers to handle known classes and an ND module that leverages hierarchical clustering over a unique set of meta-features to identify novel classes in real time.

Our empirical evaluation across 11 datasets demonstrates that CASCADE consistently matches or outperforms state-of-the-art unsupervised methods, achieving, among unsupervised approaches, the highest average $M_1$ for distinguishing known from novel classes and the highest classification accuracy, as reflected by AMI. Furthermore, CASCADE significantly narrows the performance gap with supervised algorithms, and in some cases surpasses them, while operating without labeled data and requiring minimal hyperparameter tuning.

Despite its strong performance, CASCADE exhibits some limitations, particularly in the trade-off between detection speed (TTD) and classification accuracy (AMI) in datasets with high class overlap, such as the cybersecurity benchmarks. Future work will explore dynamic adaptation strategies, including automatically adjusting clustering thresholds and buffer sizes, to better balance early detection with high classification accuracy. We also plan to explore adaptation mechanisms for handling different types of concept drift. Finally, we intend to extend the experiments with CASCADE to additional real-world ND applications to further validate its robustness and scalability.

Table 7.5: Mean TTD. Empty cell indicates no detection. Best result per class/dataset pair is in **bold**. Lower values are better.

| Dataset | Class | Algorithm | | | |
|---|---|---|---|---|---|
| | | **ECHO** | **ECSM** | **MINAS** | **CASCADE** |
| Random Tree | 1 | 364 | **255** | | |
| Random RBF | 0 | **97** | 280 | 115 | 178 |
| Nursery | 2 | | | | |
| | 4 | 175 | **82** | | |
| Gas Sensor | 2 | 536 | 316 | **6** | 223 |
| | 5 | 542 | 327 | **9** | 254 |
| INSECTS-Abrupt | 0 | **89** | 445 | 1789 | |
| | 3 | **84** | 1066 | 1666 | |
| Rialto | 6 | 1304 | 1014 | 36 | **30** |
| | 7 | 1178 | 454 | **13** | 29 |
| | 8 | 1067 | 431 | **15** | 22 |
| | 9 | 1016 | 321 | 109 | **29** |
| Room Occupancy | 1 | 216 | 383 | **13** | 184 |
| Shuttle | 2 | | | **14** | |
| | 6 | | | | |
| | 7 | | | | |
| NSLKDD | 3 | | | | **50** |
| | 4 | 172 | 642 | 335 | **54** |
| NDSec1 | 3 | | | | |
| | 4 | | | | |
| | 8 | | | | |
| | 9 | | | | |
| NF-UNSW-NB15-v2 | 0 | 59 | | **8** | 233 |
| | 1 | 46 | | **20** | 210 |
| | 8 | 49 | | **13** | 211 |
| | 9 | 60 | | **4** | |
| **Mean TTD** | | 415 | 463 | 260 | **131** |
| **Detected Classes** | | 17 | 13 | 16 | 13 |

# Chapter 8

# Conclusion and Future Work

ND in DSs remains a domain of active interest across multiple fields, as self-learning algorithms that operate online and can automatically detect and adapt to new classes without requiring the model to be taken offline and retrained are applicable to a wide range of contexts. In domains such as intrusion detection or fraud detection, the emergence of new threats is often the norm rather than the exception, making the accurate identification and clustering of these novel classes paramount. However, building such algorithms remains a significant challenge, as dynamic DSs present numerous complexities that must be addressed. Furthermore, ensuring that these algorithms are properly evaluated and free from limitations that could hinder their adoption in real-world deployments is equally essential. This thesis advances the field of ND in DSs through several key contributions designed to address these challenges, which are outlined in the following paragraphs.

## 8.1 Contributions

This thesis addresses fundamental challenges in the field of ND and presents several contributions structured around four main objectives: (i) conducting a systematic literature review and proposing an updated taxonomy of the field; (ii) developing a standardized evaluation framework for ND that accounts for key data characteristics of DSs; (iii) designing a novel unsupervised ND algorithm tailored to the specific challenges inherent to this domain; and (iv) applying ND to the context of cybersecurity and intrusion detection.

### 8.1.1 Systematic Literature Review and Updated Taxonomy

In Chapter 3, we presented a comprehensive synthesis of ND in DSs. Our SLR provided an overview of the state of the field by analyzing more than 150 papers to identify research trends and common architectural patterns used in ND algorithms. Our contributions include the proposal of an updated taxonomy to classify existing algorithms based on their assumptions regarding DSs, their classification strategies, and their novelty handling mechanisms, along with a formalized, standardized definition of ND. Finally, we reviewed and synthesized the key challenges still present in the field and highlighted critical gaps, such as the lack of a consistent evaluation framework.

### 8.1.2 Framework for Evaluating Novelty Detection Algorithms

To address the lack of consistent evaluation in ND identified in our SLR, we began in Chapter 4 by empirically exploring commonly used performance metrics in imbalanced scenarios, which represent a common challenge in ND, in order to gain a better understanding of their behaviour and limitations. Through extensive experimentation, we evaluated the robustness of these metrics to noise and class imbalance and provided recommendations for selecting appropriate performance metrics across different imbalanced contexts.

In Chapter 5, we then shifted our focus specifically to ND. We formalized a reporting protocol for four key DS characteristics: time between novel classes, ratio of offline samples, ratio of known classes, and concept sparsity. We empirically demonstrated that these characteristics can significantly influence algorithm performance and should therefore be reported to ensure comparability across studies. Furthermore, we reviewed existing ND metrics and proposed a suite of metrics that simplify the evaluation around two main aspects: the binary detection of known versus novel classes ($M_\beta$) and the clustering or multi-class classification capability of algorithms (AMI), providing a more comprehensive view of an algorithm's performance. Finally, we introduced two new metrics, TTD and TTC, to capture temporal performance aspects previously overlooked in the domain.

### 8.1.3 CASCADE: High-Performing Unsupervised Novelty Detection

To demonstrate the applicability of ND to the cybersecurity domain and to validate our core idea of using multiple binary classifiers combined with a filtering mechanism and a voting classifier, we evaluated the performance of our proposed approach against classical

multi-class classification algorithms in Chapter 6. Using several modern intrusion detection datasets, we confirmed that our approach enhanced the detection of novel attacks while overall matching or surpassing traditional classification techniques.

Building on this approach as the foundation for our fully fledged ND algorithm presented in Chapter 7, we introduced CASCADE, a novel algorithm designed to overcome key limitations of existing methods, such as their reliance on ground-truth labels and sensitivity to hyperparameter tuning. CASCADE combines our classification module from Chapter 6 with hierarchical clustering and automatic classifier updates. The proposed method demonstrated strong performance compared to other state-of-the-art ND approaches, achieving the highest overall binary and multi-class classification accuracy for novel classes, even surpassing supervised methods.

In addition to the previously mentioned contributions, this thesis also led to the development and publication of an open-source Python package that implements various state-of-the-art techniques which were previously unavailable as open-source tools or existed only in other programming languages, making comparison difficult [89].

## 8.2 Limitations and Challenges

While this thesis provides significant advancements in the field of ND, several overarching limitations and encountered challenges should be acknowledged, as they may serve as the basis for future work.

First, although we used multiple modern intrusion detection datasets to demonstrate the applicability of ND in this domain, the availability of large, annotated, and up-to-date cybersecurity datasets beyond intrusion detection remains limited. This lack of diversity constrains the ability to fully evaluate the generalization of ND approaches. In addition, the absence of publicly available datasets representing true DSs in this field remains a major limitation. Most evaluations currently rely on transforming batch learning datasets into DSs, which does not necessarily capture the same temporal dynamics and dependencies inherent to this domain.

Another limitation lies in the scope of our evaluation of CASCADE compared to other state-of-the-art methods. While we selected a comprehensive set of datasets, including those exhibiting concept drift, and assessed multiple aspects of algorithmic performance, such as the ability to detect novel classes, distinguish between them, and identify them in a timely manner, we did not evaluate additional factors such as the handling of concept

drift, the trade-off between speed and accuracy, hyperparameter impact, and computational requirements. These aspects represent promising directions for future research.

Finally, the lack of accessible source code for many ND algorithms, combined with differences in their underlying assumptions, as discussed in Chapter 3, hinders direct comparison between methods. We aim to gradually address this issue by implementing and releasing an increasing number of these algorithms through our package. However, this effort is ongoing and resource-intensive, which represented a significant challenge during the completion of this work and also explains why only four alternative techniques were included for comparison.

## 8.3 Future Work

Based on the limitations outlined in the previous section as well as other research directions outlined in our SLR that could not be addressed in this thesis, we highlight the following areas as potential future work.

### 8.3.1 Enhancing the Evaluation

As mentioned in the limitations section, broadening the evaluation of ND methods through a systematic analysis of the trade-off between speed and accuracy, computational requirements, and hyperparameter dependence would be a promising research direction, as these factors play a crucial role and can directly hinder the deployment of ND algorithms in real-world scenarios.

Moreover, a thorough analysis of the ability of these algorithms to handle concept drift also represents an important research avenue. While we did not include this aspect in our evaluation framework due to the difficulty of measuring it and the lack of real-world datasets with controlled concept drift, an extension of our framework could incorporate drift analysis as a new data characteristic, along with a metric specifically designed to measure an algorithm's ability to adapt to such drift. Additionally, different types of drift, such as gradual and sudden changes, could be further explored and integrated into the evaluation process.

### 8.3.2 Improving CASCADE

CASCADE demonstrated strong performance across the tested datasets but could be further improved through several enhancements. First, while CASCADE offers the significant advantage of being fully unsupervised during the online phase, which removes the need for unreasonable labeling, its performance could likely be improved by incorporating optional active learning feedback. This would be particularly beneficial in scenarios where it is realistic for some labels to become available after a certain period, allowing the model to refine its predictions over time.

Another promising direction for improvement is the integration of adaptive thresholds and automatic calibration in a dynamic fashion. Currently, the thresholds set by the filters are static, but dynamically adjusting these thresholds throughout the DS could potentially yield higher performance without increasing computational costs. Finally, exploring different adaptation mechanisms for the binary classifiers, along with a thorough evaluation of the impact of various types of concept drift, also represents an important direction for future research and for the potential application of CASCADE in domains such as continual learning.

### 8.3.3 Broadening the Application of Novelty Detection

Finally, while ND is a promising field of research, it remains relatively small compared to other areas of artificial intelligence, and its applications are still largely confined to specific domains. In this thesis, as in much of the existing research, we focused on the application of ND to tabular data and its specific use in intrusion detection. However, other potential applications, such as image and text analysis, fraud detection, and medical domains, are promising and would benefit from further research.

As mentioned in the limitations section, the availability of purpose-built ND datasets and, more broadly, datasets that capture challenges specific to DSs, such as memory limitations and concept drift, remains a significant obstacle. The creation of such datasets would be crucial for advancing the field and enabling the real-world deployment and testing of ND methods, particularly in applications such as IDSs.

# References


[1] Zahraa S. Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. Anynovel: detection of novel concepts in evolving data streams. *Evolving Systems*, 7(2):73–93, Jun 2016.

[2] A. Abid, S. Jamoussi, and A.B. Hamadou. AIS-Clus: A Bio-Inspired Method for Textual Data Stream Clustering. *Vietnam. J. Comput. Sci.*, 6(2):223–256, 2019.

[3] Amal Abid and Salma Jamoussi. Novelty detection in data stream clustering using the artificial immune system. In *European Mediterranean & Middle Eastern Conference on Information Systems*, volume 2016, 2016.

[4] Charu C. Aggarwal, Philip S. Yu, Jiawei Han, and Jianyong Wang. - a framework for clustering evolving data streams. In Johann-Christoph Freytag, Peter Lockemann, Serge Abiteboul, Michael Carey, Patricia Selinger, and Andreas Heuer, editors, *Proceedings 2003 VLDB Conference*, pages 81–92. Morgan Kaufmann, San Francisco, 2003.

[5] Supriya Agrahari and Anil Kumar Singh. Concept drift detection in data stream mining : A literature review. *Journal of King Saud University - Computer and Information Sciences*, 34(10, Part B):9523–9540, 2022.

[6] Gabriel Aguiar, Bartosz Krawczyk, and Alberto Cano. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework, 2022.

[7] Rasheed Ahmad, Izzat Alsmadi, Wasim Alhamdani, and Lo'ai Tawalbeh. Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, 56(10):10733–10811, Oct 2023.

[8] H. Al-Behadili, A. Grumpe, L. Migdadi, and C. Wöhler. Incremental parzen window classifier for a multi-class system. *Int. J. Simul. Syst. Sci. Technol.*, 17(34):6.1–6.11, 2016.

[9] Husam Al-Behadili, Arne Grumpe, Christian Dopp, and Christian Wöhler. Extreme learning machine based novelty detection for incremental semi-supervised learning. In *2015 Third International Conference on Image Information Processing (ICIIP)*, pages 230–235, 2015.

[10] Husam Al-Behadili, Arne Grumpe, Lubaba Migdadi, and Christian Wöhler. Semi-supervised learning using incremental support vector machine and extreme value theory in gesture data. In *2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim)*, pages 184–189, 2016.

[11] Husam Al-Behadili, Arne Grumpe, and Christian Wöhler. Semi-supervised learning using incremental polynomial classifier and extreme value theory. In *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 332–337, 2015.

[12] Tahseen Al-Khateeb, Mohammad M. Masud, Khaled M. Al-Naami, Sadi Evren Seker, Ahmad M. Mustafa, Latifur Khan, Zouheir Trabelsi, Charu Aggarwal, and Jiawei Han. Recurring and Novel Class Detection Using Class-Based Ensemble for Evolving Data Stream. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 28(10):2752–2764, 2016.

[13] Tahseen Al-Khateeb, Mohammad M. Masud, Latifur Khan, Charu Aggarwal, Jiawei Han, and Bhavani Thuraisingham. Stream classification with recurring and novel class detection using class-based ensemble. In *2012 IEEE 12th International Conference on Data Mining*, pages 31–40, 2012.

[14] Wathiq Laftah Al-Yaseen, Zulaiha Ali Othman, and Mohd Zakree Ahmad Nazri. Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Systems with Applications*, 67:296–303, 2017.

[15] Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

[16] Omar Alghushairy, Raed Alsini, Terence Soule, and Xiaogang Ma. A review of local outlier factor algorithms for outlier detection in big data streams. *Big Data and Cognitive Computing*, 5(1), 2021.

[17] Marta Amorim, Frederico D. Bortoloti, Patrick M. Ciarelli, Evandro O. T. Salles, and Daniel C. Cavalieri. Novelty Detection in Social Media by Fusing Text and Image Into a Single Structure. *IEEE ACCESS*, 7:132786–132802, 2019.

[18] Annalisa Appice, Michelangelo Ceci, Corrado Loglisci, Costantina Caruso, Fabio Fumarola, Michele Todaro, Donato Malerba, et al. A relational approach to novelty detection in data streams. In *SEBD*, pages 89–100, 2009.

[19] Ira Assent, Philipp Kranen, Corinna Baldauf, and Thomas Seidl. Detecting outliers on arbitrary data streams using anytime approaches. In *Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques*, StreamKDD '10, page 10–15, New York, NY, USA, 2010. Association for Computing Machinery.

[20] Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a moving window over streaming data. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, page 633–634, USA, 2002. Society for Industrial and Applied Mathematics.

[21] Nathalie A. Barbosa, Louise Travé-Massuyès, and Victor H. Grisales. A novel algorithm for dynamic clustering: Properties and performance. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 565–570, 2016.

[22] Jean Paul Barddal, Heitor Murilo Gomes, and Fabrício Enembreck. Sncstream: A social network-based data stream clustering algorithm. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, page 935–940, New York, NY, USA, 2015. Association for Computing Machinery.

[23] Frank Beer, Tim Hofer, David Karimi, and Ulrich Bühler. A new attack composition for network security. In *10. DFN-Forum Kommunikationstechnologien*, pages 11–20. Gesellschaft für Informatik e.V., Bonn, 2017.

[24] Clauber Gomes Bezerra, Bruno Sielly Jales Costa, Luiz Affonso Guedes, and Plamen Parvanov Angelov. An evolving approach to data streams clustering based on typicality and eccentricity data analytics. *INFORMATION SCIENCES*, 518:13–28, 2020.

[25] Xin Bi, Chao Zhang, Xiangguo Zhao, Donghang Li, Yongjiao Sun, and Yuliang Ma. CODES: Efficient Incremental Semi-Supervised Classification Over Drifting and Evolving Social Streams. *IEEE ACCESS*, 8:14024–14035, 2020.

[26] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010.

[27] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. Moa: Massive online analysis, a framework for stream classification and clustering. In Tom Diethe, Nello Cristianini, and John Shawe-Taylor, editors, *Proceedings of the First Workshop on Applications of Pattern Analysis*, volume 11 of *Proceedings of Machine Learning Research*, pages 44–50, Cumberland Lodge, Windsor, UK, 01–03 Sep 2010. PMLR.

[28] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[29] Mohamed-Rafik Bouguelia, Yolande Belaid, and Abdel Belaid. Efficient active novel class detection for data stream classification. In *2014 22nd International Conference on Pattern Recognition*, pages 2826–2831, 2014.

[30] Mohamed-Rafik Bouguelia, Slawomir Nowaczyk, and Amir H. Payberah. An adaptive algorithm for anomaly and novelty detection in evolving data streams. *DATA MINING AND KNOWLEDGE DISCOVERY*, 32(6):1597–1633, 2018.

[31] Giampaolo Bovenzi, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. A hierarchical hybrid intrusion detection approach in iot scenarios. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–7, 2020.

[32] Paula Branco, Luís Torgo, and Rita P. Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Comput. Surv.*, 49(2), aug 2016.

[33] C. Fahy, S. Yang, and M. Gongora. Classification in Dynamic Data Streams with a Scarcity of Labels. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[34] C. Puerto-Santana, C. Bielza, J. Diaz-Rozo, G. Ramirez-Gargallo, F. Mantovani, G. Virumbrales, J. Labarta, and P. Larrañaga. Asymmetric HMMs for online ball-bearing health assessments. *IEEE Internet of Things Journal*, pages 1–1, 2022.

[35] Jesus A. Carino, Miguel Delgado-Prieto, Jose Antonio Iglesias, Araceli Sanchis, Daniel Zurita, Marta Millan, Juan Antonio Ortega Redondo, and Rene Romero-Troncoso. Fault Detection and Identification Methodology Under an Incremental Learning Framework Applied to Industrial Machinery. *IEEE ACCESS*, 6:49755–49766, 2018.

[36] Ander Carreño, Iñaki Inza, and Jose A. Lozano. Sndprob: A probabilistic approach for streaming novelty detection. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):6335–6348, 2023.

[37] Robert Cattral and Franz Oppacher. Poker Hand. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C5KW38.

[38] Michelangelo Ceci, Annalisa Appice, Corrado Loglisci, Costantina Caruso, Fabio Fumarola, and Donato Malerba. Novelty detection from evolving complex data streams with time windows. In Jan Rauch, Zbigniew W. Raś, Petr Berka, and Tapio Elomaa, editors, *Foundations of Intelligent Systems*, pages 563–572, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[39] Michelangelo Ceci, Annalisa Appice, Corrado Loglisci, Costantina Caruso, Fabio Fumarola, Carmine Valente, and Donato Malerba. Relational frequent patterns mining for novelty detection from data streams. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, pages 427–439, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[40] M.B. Chandak. Role of big-data in classification and novel class detection in data streams. *J. Big Data*, 3(1), 2016.

[41] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[42] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[43] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. *IEEE Access*, 9:78368–78381, 2021.

[44] Clément Christophe, Julien Velcin, Jairo Cugliari, Philippe Suignard, and Manel Boumghar. How to detect novelty in textual data streams? A comparative study of existing methods. *CoRR*, abs/1909.05099, 2019.

[45] Gilles Cohen, Mélanie Hilario, Hugo Sax, Stéphane Hugonnet, and Antoine Geissbuhler. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine*, 37(1):7–18, 2006.

[46] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46, 1960.

[47] Joel D. Costa Júnior, Elaine R. Faria, Jonathan A. Silva, João Gama, and Ricardo Cerri. Novelty detection for multi-label stream classification. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 144–149, 2019.

[48] Joel D. Costa Júnior, Elaine R. Faria, Jonathan A. Silva, João Gama, and Ricardo Cerri. Pruned sets for multi-label stream classification without true labels. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[49] A.L. Cristiani, T.P. da Silva, and H. de Arruda Camargo. *A Fuzzy Approach for Classification and Novelty Detection in Data Streams Under Intermediate Latency*, volume 12320 LNAI. Springer Science and Business Media Deutschland GmbH, 2020. Journal Abbreviation: 9th Brazilian Conference on Intelligent Systems, BRACIS 2020 Publication Title: 9th Brazilian Conference on Intelligent Systems, BRACIS 2020.

[50] André Luis Cristiani and Heloisa de Arruda Camargo. A fuzzy multi-class novelty detector for data streams under intermediate latency. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2021.

[51] D. -W. Zhou, Y. Yang, and D. -C. Zhan. Learning to Classify With Incremental New Class. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2429–2443, 2022.

[52] Tiago Pinho da Silva and Heloisa de Arruda Camargo. Possibilistic approach for novelty detection in data streams. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2020.

[53] Tiago Pinho da Silva, Leonardo Schick, Priscilla de Abreu Lopes, and Heloisa de Arruda Camargo. A fuzzy multiclass novelty detector for data streams. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, 2018.

[54] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 233–240, New York, NY, USA, 2006. Association for Computing Machinery.

[55] Elaine Ribeiro de Faria, Isabel Ribeiro Goncalves, Joao Gama, and Andre Carlos Ponce de Leon Ferreira Carvalho. Evaluation of multiclass novelty detection algorithms for data streams. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 27(11):2961–2973, 2015.

[56] Elaine Ribeiro de Faria, André Carlos Ponce de Leon Ferreira Carvalho, and João Gama. Minas: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, 30(3):640–680, May 2016.

[57] Rosario Delgado and Xavier-Andoni Tibau. Why cohen's kappa should be avoided as performance measure in classification. *PLOS ONE*, 14(9):1–26, 09 2019.

[58] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[59] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[60] Salah Ud Din and Junming Shao. Exploiting evolving micro-clusters for data stream classification with emerging class detection. *INFORMATION SCIENCES*, 507:404–420, 2020.

[61] S.U. Din, J. Shao, J. Kumar, C.B. Mawuli, S.M.H. Mahmud, W. Zhang, and Q. Yang. Data stream classification with novel class detection: a review, comparison and challenges. *Knowl. Inf. Systems. Syst.*, 63(9):2231–2276, 2021.

[62] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, Apr 2020.

[63] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[64] F. Dufrenois. Incremental and compressible kernel null discriminant analysis. *Pattern Recogn.*, 127, 2022.

[65] James P Egan and James Pendleton Egan. *Signal detection theory and ROC-analysis*. Academic press, 1975.

[66] Sarah M. Erfani, Sutharshan Rajasegarar, and Christopher Leckie. An efficient approach to detecting concept-evolution in network data streams. In *2011 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pages 1–7, 2011.

[67] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[68] C. Fahy and S. Yang. Finding and Tracking Multi-Density Clusters in Online Dynamic Data Streams. *IEEE Trans. Big Data*, 8(1):178–192, 2022.

[69] Conor Fahy, Shengxiang Yang, and Mario Gongora. Scarcity of Labels in Non-Stationary Data Streams: A Survey. *ACM Comput. Surv.*, 55(2), 2022.

[70] Elaine R. Faria, João Gama, and André C. P. L. F. Carvalho. Novelty detection algorithm for data streams multi-class problems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, page 795–800, New York, NY, USA, 2013. Association for Computing Machinery.

[71] Elaine R. Faria, Isabel J. C. R. Gonçalves, André C. P. L. F. de Carvalho, and João Gama. Novelty detection in data streams. *Artificial Intelligence Review*, 45(2):235–269, Feb 2016.

[72] Dewan Md. Farid and Chowdhury Mofizur Rahman. Novel class detection in concept-drifting data stream mining employing decision tree. In *2012 7th International Conference on Electrical and Computer Engineering*, pages 630–633, 2012.

[73] Dewan Md. Farid, Li Zhang, Alamgir Hossain, Chowdhury Mofizur Rahman, Rebecca Strachan, Graham Sexton, and Keshav Dahal. An adaptive ensemble classifier for mining concept drifting data streams. *EXPERT SYSTEMS WITH APPLICATIONS*, 40(15):5895–5906, 2013.

[74] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[75] Geli Fei, Shuai Wang, and Bing Liu. Learning cumulatively to become more knowledgeable. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1565–1574, New York, NY, USA, 2016. Association for Computing Machinery.

[76] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[77] R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.

[78] Peter Flach. Performance evaluation in machine learning: The good, the bad, the ugly, and the way forward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9808–9814, Jul. 2019.

[79] Peter Flach and Meelis Kull. Precision-recall-gain curves: Pr analysis done right. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[80] Floris Gaisser, Maja Rudinac, Pieter P. Jonker, and David Tax. Online face recognition and learning for cognitive robots. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–9, 2013.

[81] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), March 2014.

[82] Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.

[83] João Gama. A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, Apr 2012.

[84] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, Mar 2013.

[85] Yang Gao, Swarup Chandra, Yifan Li, Latifur Khan, and Thuraisingham Bhavani. Saccos: A semi-supervised framework for emerging class detection and concept drift adaption over data streams. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1416–1426, 2022.

[86] Yang Gao, Yi-Fan Li, Bo Dong, Yu Lin, and Latifur Khan. Sim: Open-world multitask stream classifier with integral similarity metrics. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 751–760, 2019.

[87] K.D. Garcia, E.R. de Faria, C.R. de Sá, J. Mendes-Moreira, C.C. Aggarwal, A.C.P.L.F. de Carvalho, and J.N. Kok. *Ensemble Clustering for Novelty Detection in Data Streams*, volume 11828 LNAI. Springer, 2019. Journal Abbreviation: 22nd International Conference on Discovery Science, DS 2019 Publication Title: 22nd International Conference on Discovery Science, DS 2019.

[88] K.D. Garcia, M. Poel, J.N. Kok, and A.C.P.L.F. de Carvalho. *Online Clustering for Novelty Detection and Concept Drift in Data Streams*, volume 11805 LNAI. Springer Verlag, 2019. Journal Abbreviation: 19th EPIA Conference on Artificial Intelligence, EPIA 2019 Publication Title: 19th EPIA Conference on Artificial Intelligence, EPIA 2019.

[89] Jean-Gabriel Gaudreault. Streamndr: Novelty detection for data streams in python.

[90] Jean-Gabriel Gaudreault and Paula Branco. Toward streamlining the evaluation of novelty detection in data streams. In Albert Bifet, Ana Carolina Lorena, Rita P. Ribeiro, João Gama, and Pedro H. Abreu, editors, *Discovery Science*, pages 703–717, Cham, 2023. Springer Nature Switzerland.

[91] Jean-Gabriel Gaudreault and Paula Branco. Detection of novel cyberattacks using multi-binary classifiers. In *2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, pages 1–9, 2024.

[92] Jean-Gabriel Gaudreault and Paula Branco. Empirical analysis of performance assessment for imbalanced classification. *Machine Learning*, 113(8):5533–5575, Aug 2024.

[93] Jean-Gabriel Gaudreault and Paula Branco. A systematic literature review of novelty detection in data streams: Challenges and opportunities. *ACM Comput. Surv.*, 56(10), May 2024.

[94] Jean-Gabriel Gaudreault and Paula Branco. Prioritizing the essential: A robust evaluation framework for novelty detection. *Machine Learning*, 114(6):143, Apr 2025.

[95] Jean-Gabriel Gaudreault and Paula Branco. Cascade: Clustering-based adaptive stream classification and detection of emerging classes. unpublished, To Be Submitted to SIAM International Conference on Data Mining 2026, N.D.

[96] Jean-Gabriel Gaudreault, Paula Branco, and João Gama. An analysis of performance metrics for imbalanced classification. In Carlos Soares and Luis Torgo, editors, *Discovery Science*, pages 67–77, Cham, 2021. Springer International Publishing.

[97] Chuanxing Geng, Sheng-Jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *CoRR*, abs/1811.08581, 2018.

[98] Alexander Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*, 2016.

[99] Heitor Murilo Gomes, Jean Paul Barddal, Fabricio Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM COMPUTING SURVEYS*, 50(2), 2017.

[100] Heitor Murilo Gomes, Maciej Grzenda, Rodrigo Mello, Jesse Read, Minh Huong Le Nguyen, and Albert Bifet. A Survey on Semi-Supervised Learning for Delayed Partially Labelled Data Streams. *ACM Comput. Surv.*, 2022.

[101] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: State of the art, challenges, and opportunities. *SIGKDD Explor. Newsl.*, 21(2):6–22, 2019.

[102] J. Gorodkin. Comparing two k-category assignments by a k-category correlation coefficient. *Computational Biology and Chemistry*, 28(5):367–374, 2004.

[103] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview, 2020.

[104] Christian Gruhl, Abdul Hannan, Zhixin Huang, Chandana Nivarthi, and Stephan Vogt. The problem with real-world novelty detection - issues in multivariate probabilistic models. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 204–209, 2021.

[105] Christian Gruhl, Bernhard Sick, and Sven Tomforde. Novelty detection in continuously changing environments. *Future Generation Computer Systems*, 114:138–154, 2021.

[106] Xiaojie Guo, Amir Alipour-Fanid, Lingfei Wu, Hemant Purohit, Xiang Chen, Kai Zeng, and Liang Zhao. Multi-stage deep classifier cascades for open world recognition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 179–188, New York, NY, USA, 2019. Association for Computing Machinery.

[107] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 26(9):2250–2267, 2014.

[108] Fatma Hamdi and Younès Bennani. Learning random subspace novelty detection filters. In *The 2011 International Joint Conference on Neural Networks*, pages 2273–2280, 2011.

[109] David J Hand. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123, 2009.

[110] David J Hand and Christoforos Anagnostopoulos. A better beta for the h measure of classification performance. *Pattern Recognition Letters*, 40:41–46, 2014.

[111] David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, Nov 2001.

[112] Ahsanul Haque, Latifur Khan, and Michael Baron. Semi supervised adaptive framework for classifying evolving data stream. In Tru Cao, Ee-Peng Lim, Zhi-Hua Zhou, Tu-Bao Ho, David Cheung, and Hiroshi Motoda, editors, *Advances in Knowledge Discovery and Data Mining*, pages 383–394, Cham, 2015. Springer International Publishing.

[113] Ahsanul Haque, Latifur Khan, and Michael Baron. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1652–1658. AAAI Press, 2016.

[114] Ahsanul Haque, Latifur Khan, Michael Baron, Bhavani Thuraisingham, and Charu Aggarwal. Efficient handling of concept drift and concept evolution over stream data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 481–492, 2016.

[115] Michael Harries, New South Wales, et al. Splice-2 comparative evaluation: Electricity pricing. *Technical Report*, 1999.

[116] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[117] Morteza Zi Hayat and Mahmoud Reza Hashemi. A dct based approach for detecting novelty and concept drift in data streams. In *2010 International Conference of Soft Computing and Pattern Recognition*, pages 373–378, 2010.

[118] Elena Ikonomovska, João Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 23(1):128–168, Jul 2011.

[119] M.R. Islam. *Recurring and novel class detection in concept-drifting data streams using class-based ensemble*, volume 8444 LNAI. Springer Verlag, Tainan, 2014. Journal Abbreviation: 18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2014 Publication Title: 18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2014.

[120] J. W. Sangma, Y. Rani, V. Pal, N. Kumar, and R. Kushwaha. FHC-NDS: Fuzzy Hierarchical Clustering of Multiple Nominal Data Streams. *IEEE Transactions on Fuzzy Systems*, pages 1–12, 2022.

[121] Syed Muslim Jameel, Manzoor Ahmed Hashmani, Mobashar Rehman, and Arif Budiman. An Adaptive Deep Learning Framework for Dynamic Image Classification in the Internet of Things Environment. *SENSORS*, 20(20), 2020.

[122] Nathalie Japkowicz. *Assessment Metrics for Imbalanced Learning*, chapter 8, pages 187–206. John Wiley & Sons, Ltd, 2013.

[123] Olga Jodelka, Christos Anagnostopoulos, and Kostas Kolomvatsos. Adaptive novelty detection over contextual data streams at the edge using one-class classification. In *2021 12th International Conference on Information and Communication Systems (ICICS)*, pages 213–219, 2021.

[124] J.M. Kang, M.A. Ahmad, A. Teredesai, and R. Gaborski. Cognitively motivated novelty detection in video data streams. In *Multimedia Data Min. and Knowl. Discov.*, pages 209–233. Springer London, 2007. Journal Abbreviation: Multimedia Data Min. and Knowl. Discov.

[125] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc.

[126] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134, 2009.

[127] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, 51(1):7–15, 2009. Special Section - Most Cited Articles in 2002 and Regular Research Papers.

[128] Yanika Kongsorot, Punyaphol Horata, and Pakarat Musikawan. An Incremental Kernel Extreme Learning Machine for Multi-Label Learning With Emerging New Labels. *IEEE ACCESS*, 8:46055–46070, 2020.

[129] Bartosz Krawczyk, Leandro L. Minku, Joao Gama, Jerzy Stefanowski, and Michal Wozniak. Ensemble learning for data stream analysis: A survey. *INFORMATION FUSION*, 37:132–156, 2017.

[130] Bartosz Krawczyk and Michal Wozniak. One-class classifiers with incremental learning and forgetting for data streams with concept drift. *SOFT COMPUTING*, 19(12, SI):3387–3400, 2015.

[131] Bartosz Krawczyk and Michał Woźniak. Reacting to different types of concept drift with adaptive and incremental one-class classifiers. In *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, pages 30–35, 2015.

[132] Klaus Krippendorff. Computing krippendorff's alpha-reliability, Jan 2011.

[133] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[134] Thomas C.W. Landgrebe and Robert P.W. Duin. Approximating the multiclass roc by pairwise analysis. *Pattern Recognition Letters*, 28(13):1747–1758, 2007.

[135] Andre Lemos, Walmir Caminhas, and Fernando Gomide. Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *INFORMATION SCIENCES*, 220:64–85, 2013.

[136] Xiangjun Li, Yong Zhou, Ziyan Jin, Peng Yu, and Shun Zhou. A Classification and Novel Class Detection Algorithm for Concept Drift Data Stream Based on the Cohesiveness and Separation Index of Mahalanobis Distance. *JOURNAL OF ELECTRICAL AND COMPUTER ENGINEERING*, 2020, 2020.

[137] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20), 2019.

[138] Jing Liu, Guo-Sheng Xu, Da Xiao, Li-Ze Gu, and Xin-Xin Niu. A Semi-supervised Ensemble Approach for Mining Data Streams. *JOURNAL OF COMPUTERS*, 8(11, SI):2873–2879, 2013.

[139] K.-H. Liu, W.-P. Zhan, Y.-F. Liang, Y.-N. Zhang, H.-Z. Guo, J.-F. Yao, Q.-Q. Wu, and Q.-Q. Hong. The design of error-correcting output codes algorithm for the open-set recognition. *Appl Intell*, 52(7):7843–7869, 2022.

[140] Lisa Liu, Gints Engelen, Timothy Lynar, Daryl Essam, and Wouter Joosen. Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 254–262. IEEE, 2022.

[141] Sin Kit Lo, Qinghua Lu, Chen Wang, Hye-Young Paik, and Liming Zhu. A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Comput. Surv.*, 54(5), may 2021.

[142] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019.

[143] Atefeh Mahdavi and Marco Carvalho. A survey on open set recognition. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, December 2021.

[144] Sepehr Maleki and Chris Bingham. Robust hierarchical clustering for novelty identification in sensor networks: With applications to industrial systems. *APPLIED SOFT COMPUTING*, 85, 2019.

[145] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[146] S. J. Mason and N. E. Graham. Areas beneath the relative operating characteristics (roc) and relative operating levels (rol) curves: Statistical significance and interpretation. *Quarterly Journal of the Royal Meteorological Society*, 128(584):2145–2166, 2002.

[147] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M. Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859–874, 2011.

[148] Mohammad M. Masud, Tahseen M. Al-Khateeb, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. Detecting recurring and novel

classes in concept-drifting data streams. In *2011 IEEE 11th International Conference on Data Mining*, pages 1176–1181, 2011.

[149] Mohammad M. Masud, Qing Chen, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Classification and novel class detection of data streams in a dynamic feature space. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 337–352, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[150] Mohammad M. Masud, Qing Chen, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. Addressing concept-evolution in concept-drifting data streams. In *2010 IEEE International Conference on Data Mining*, pages 929–934, 2010.

[151] Mohammad M. Masud, Qing Chen, Latifur Khan, Charu C. Aggarwal, Jing Gao, Jiawei Han, Ashok Srivastava, and Nikunj C. Oza. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 25(7):1484–1497, 2013.

[152] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Integrating novel class detection with classification for concept-drifting data streams. In Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 79–94, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[153] Mohammad M. Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. Classification and novel class detection in data streams with active mining. In Mohammed J. Zaki, Jeffrey Xu Yu, B. Ravindran, and Vikram Pudi, editors, *Advances in Knowledge Discovery and Data Mining*, pages 311–324, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[154] B.W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975.

[155] Shon Mendelson and Boaz Lerner. Online cluster drift detection for novelty detection in data streams. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 171–178, 2020.

[156] Xuedan Miao, Ying Liu, Haiquan Zhao, and Chunguang Li. Distributed Online One-Class Support Vector Machine for Anomaly Detection Over Networks. *IEEE TRANSACTIONS ON CYBERNETICS*, 49(4):1475–1488, 2019.

[157] Y. Miao, L. Qiu, H. Chen, J. Zhang, and Y. Wen. *Novel class detection within classification for data streams*, volume 7952 LNCS. Springer Verlag, Dalian, 2013. Journal Abbreviation: 10th International Symposium on Neural Networks, ISNN 2013 Publication Title: 10th International Symposium on Neural Networks, ISNN 2013.

[158] Siddhartha Mishra, Nicholas Monath, Michael Boratko, Ariel Kobren, and Andrew McCallum. An evaluative measure of clustering methods incorporating hyperparameter sensitivity. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7):7788–7796, Jun. 2022.

[159] Saad Mohamad, Moamar Sayed-Mouchaweh, and Abdelhamid Bouchachia. Active learning for classifying data streams with unknown number of classes. *NEURAL NETWORKS*, 98:1–15, 2018.

[160] Xin Mu, Kai Ming Ting, and Zhi-Hua Zhou. Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1605–1618, 2017.

[161] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[162] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[163] Ahmad M. Mustafa, Gbadebo Ayoade, Khaled Al-Naami, Latifur Khan, Kevin W. Hamlen, Bhavani Thuraisingham, and Frederico Araujo. Unsupervised deep embedding for novel class detection over data stream. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1830–1839, 2017.

[164] Gyoung S. Na, Donghyun Kim, and Hwanjo Yu. Dilof: Effective and memory efficient local outlier detection in data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1993–2002, New York, NY, USA, 2018. Association for Computing Machinery.

[165] Emre Ozbilge. Experiments in online expectation-based novelty-detection using 3D shape and colour perceptions for mobile robot inspection. *ROBOTICS AND AUTONOMOUS SYSTEMS*, 117:68–79, 2019.

[166] Jinxing Pan, Xiaoshan Yang, Yi Huang, and Changsheng Xu. Few-shot egocentric multimodal activity recognition. In *ACM Multimedia Asia*, MMAsia '21, New York, NY, USA, 2022. Association for Computing Machinery.

[167] Cheong Hee Park. Outlier and anomaly pattern detection on data streams. *JOURNAL OF SUPERCOMPUTING*, 75(9):6118–6128, 2019.

[168] Brandon Parker, Ahmad M. Mustafa, and Latifur Khan. Novel class detection and feature via a tiered ensemble approach for stream mining. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, pages 1171–1178, 2012.

[169] Brandon S. Parker and Latifur Khan. Detecting and tracking concept class drift and emergence in non-stationary fast data streams. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2908–2913. AAAI Press, 2015.

[170] Karl Pearson. Vii. mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, pages 253–318, 1896.

[171] Lenka Pitonakova and Seth Bullock. The robustness-fidelity trade-off in Grow When Required neural networks performing continuous novelty detection. *NEURAL NETWORKS*, 122:183–195, 2020.

[172] Roozbeh Razavi-Far, Ehsan Hallaji, Mehrdad Saif, and Gregory Ditzler. A Novelty Detector and Extreme Verification Latency Model for Nonstationary Environments. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 66(1):561–570, 2019.

[173] Attila Reiss. PAMAP2 Physical Activity Monitoring. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5NW2H.

[174] Huorong Ren, Zhixing Ye, and Zhiwu Li. Anomaly detection based on a dynamic Markov model. *INFORMATION SCIENCES*, 411:52–65, 2017.

[175] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, USA, 2nd edition, 1979.

[176] Nathalie Barbosa Roa, Louise Trave-Massuyes, and Victor H. Grisales-Palacio. DyClee: Dynamic clustering for tracking evolving environments. *PATTERN RECOGNITION*, 94:162–186, 2019.

[177] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. Adjusting for chance clustering comparison measures. *JMLR*, 17(1):4635–4666, 2016.

[178] Stefano Rovetta, Zied Mnasri, and Francesco Masulli. Detection of hazardous road events from audio streams: An ensemble outlier detection approach. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–6, 2020.

[179] Andrzej Rusiecki. Robust neural network for novelty detection on data streams. In Leszek Rutkowski, Marcin Korytkowski, Rafał Scherer, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artificial Intelligence and Soft Computing*, pages 178–186, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[180] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[181] Deepita Saha, Mozzammel Haque, Akash Sarkar, Famina Alam, Dewan Md Farid, Chowdhury Mofizur Rahman, and Swakkhar Shatabda. Ieee wiecon-ece 2018 novel class detection in concept drifting data streams using decision tree leaves. In *2018 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, pages 87–90, 2018.

[182] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21, 03 2015.

[183] Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *CoRR*, abs/2110.14051, 2021.

[184] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, 27(1):357–370, Feb 2022.

[185] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

[186] Enrique F. Schisterman and Neil Perkins. Confidence intervals for the youden index and corresponding optimal cut-point. *Communications in Statistics - Simulation and Computation*, 36(3):549–563, 2007.

[187] Enrique F Schisterman, Neil J Perkins, Aiyi Liu, and Howard Bondell. Optimal cut-point and its corresponding youden index to discriminate individuals using pooled blood samples. *Epidemiology*, 16(1):73–81, 2005.

[188] Sajjad Kamali Siahroudi, Poorya Zare Moodi, and Hamid Beigy. Detection of evolving concepts in non-stationary data streams: A multiple kernel learning approach. *EXPERT SYSTEMS WITH APPLICATIONS*, 91:187–197, 2018.

[189] Bruno Sielly Jales Costa, Clauber Gomes Bezerra, Luiz Affonso Guedes, and Plamen Parvanov Angelov. Unsupervised classification of data streams based on typicality and eccentricity data analytics. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 58–63, 2016.

[190] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Comput. Surv.*, 46(1), July 2013.

[191] Dimitrios Sisiaridis, Fabrizio Carcillo, and Olivier Markowitch. A framework for threat detection in communication systems. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, PCI '16, New York, NY, USA, 2016. Association for Computing Machinery.

[192] James Seale Smith, Seth Baer, Zsolt Kira, and Constantine Dovrolis. Unsupervised continual learning and self-taught associative memory hierarchies. *CoRR*, abs/1904.02021, 2019.

[193] Mark Smith, Steven Reece, Stephen Roberts, Ioannis Psorakis, and Iead Rezek. Maritime abnormality detection using Gaussian processes. *KNOWLEDGE AND INFORMATION SYSTEMS*, 38(3):717–741, 2014.

[194] M.H. Soleimani-Babakamali, R. Sepasdar, K. Nasrollahzadeh, and R. Sarlo. A system reliability approach to real-time unsupervised structural health monitoring without prior information. *Mech Syst Signal Process*, 171, 2022.

[195] Roghayeh Soleymani, Eric Granger, and Giorgio Fumera. F-measure curves: A tool to visualize classifier performance under imbalance. *Pattern Recognition*, 100:107146, 2020.

[196] Mahdi Soltani, Behzad Ousat, Mahdi Jafari Siavoshani, and Amir Hossein Jahangir. An adaptable deep learning-based intrusion detection system to zero-day attacks. *Journal of Information Security and Applications*, 76:103516, 2023.

[197] Manuela M. C. Souza, Camila Pontes, Joao Gondim, Luis P. F. Garcia, Luiz DaSilva, and Marcelo A. Marotta. A novel open set energy-based flow classifier for network intrusion detection, 2022.

[198] Vinicius M. A. Souza, Denis M. dos Reis, André G. Maletzke, and Gustavo E. A. P. A. Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858, Nov 2020.

[199] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 100(3/4):441–471, 1987.

[200] Eduardo J. Spinosa, Andre Ponce de Leon F. de Carvalhoa, and Joao Gama. Novelty detection with application to data streams. *INTELLIGENT DATA ANALYSIS*, 13(3):405–422, 2009.

[201] Eduardo J. Spinosa, André Ponce de Leon F. de Carvalho, and João Gama. Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, page 448–452, New York, NY, USA, 2007. Association for Computing Machinery.

[202] Eduardo J. Spinosa, André Ponce de Leon F. de Carvalho, and João Gama. Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, page 976–980, New York, NY, USA, 2008. Association for Computing Machinery.

[203] Noppayut Sriwatanasakdi, Masayuki Numao, and Ken-ichi Fukui. Concept drift detection for graph-structured classifiers under scarcity of true labels. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 461–468, 2017.

[204] Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. KDD Cup 1999 Data. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C51C7N.

[205] Wanhua Su, Yan Yuan, and Mu Zhu. A relationship between the average precision and the area under the roc curve. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, page 349–352, 2015.

[206] J. Tanha, N. Samadi, Y. Abdi, and N. Razzaghi-Asl. CPSSDS: Conformal prediction for semi-supervised classification on data streams. *Inf Sci*, 584:212–234, 2022.

[207] Jafar Tanha, Yousef Abdi, Negin Samadi, Nazila Razzaghi, and Mohammad Asadpour. Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data*, 7(1):70, Sep 2020.

[208] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, 2009.

[209] Yogita Thakran and Durga Toshniwal. Unsupervised outlier detection in streaming data using weighted clustering. In *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 947–952, 2012.

[210] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 17(1):168–192, Jan 2021.

[211] Rosane M. M. Vallim, Jose A. Andrade Filho, Rodrigo F. de Mello, and Andre C. P. L. F. de Carvalho. Online behavior change detection in computer games. *EXPERT SYSTEMS WITH APPLICATIONS*, 40(16):6258–6265, 2013.

[212] Rosane M. M. Vallim, Jose A. Andrade Filho, Rodrigo F. de Mello, Andre C. P. L. F. de Carvalho, and Joao Gama. Unsupervised density-based behavior change detection in data streams. *INTELLIGENT DATA ANALYSIS*, 18(2):181–201, 2014.

[213] Rosane M.M. Vallim, José A. Andrade Filho, André C.P.L.F. de Carvalho, and João Gama. A density-based clustering approach for behavior change detection in data streams. In *2012 Brazilian Symposium on Neural Networks*, pages 37–42, 2012.

[214] Nees Jan van Eck and Ludo Waltman. Text mining and visualization using vosviewer, 2011.

[215] Anna Vergeles, Alexander Khaya, Dmytro Prokopenko, and Nataliia Manakova. Unsupervised real-time stream-based novelty detection technique an approach in a corporate cloud. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pages 166–170, 2018.

[216] Miel Verkerken, Laurens D'hooge, Didik Sudyana, Ying-Dar Lin, Tim Wauters, Bruno Volckaert, and Filip De Turck. A novel multi-stage approach for hierarchical intrusion detection. *IEEE Transactions on Network and Service Management*, 20(3):3915–3929, 2023.

[217] Zhiyuan Wan, Xin Xia, David Lo, and Gail C. Murphy. How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, 47(9):1857–1871, 2021.

[218] X. Wang and Y. Xing. An online support vector machine for the open-ended environment. *Expert Sys Appl*, 120:72–86, 2019.

[219] Yi Wang, Nan Xue, Xin Fan, Jiebo Luo, Risheng Liu, Bin Chen, Haojie Li, and Zhongxuan Luo. Fast factorization-free kernel learning for unlabeled chunk data streams. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2833–2839. AAAI Press, 2018.

[220] Yigong Wang, Zhuoyi Wang, Yu Lin, Latifur Khan, and Dingcheng Li. Cifdm: Continual and interactive feature distillation for multi-label stream learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2121–2125, New York, NY, USA, 2021. Association for Computing Machinery.

[221] Zhuoyi Wang, Zelun Kong, Swarup Changra, Hemeng Tao, and Latifur Khan. Robust high dimensional stream classification with novel class detection. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1418–1429, 2019.

[222] Zhuoyi Wang, Hemeng Tao, Zelun Kong, Swarup Chandra, and Latifur Khan. Metric learning based framework for streaming classification with concept evolution. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

[223] Zhuoyi Wang, Yigong Wang, Yu Lin, Evan Delord, and Khan Latifur. Few-sample and adversarial representation learning for continual stream mining. In *Proceedings of The Web Conference 2020*, WWW '20, page 718–728, New York, NY, USA, 2020. Association for Computing Machinery.

[224] Christian Weiss and Andreas Zell. Novelty detection and online learning for vibration-based terrain classification. *Intelligent Autonomous Systems 10, IAS 2008*, 01 2008.

[225] Dominik Wurzer and Yumeng Qin. Parameterizing kterm hashing. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 945–948, New York, NY, USA, 2018. Association for Computing Machinery.

[226] Shuyuan Xu, Linsen Li, Hangjun Yang, and Junhua Tang. Kcc method: Unknown intrusion detection based on open set recognition. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1343–1347, 2021.

[227] Yiyang Xu, Yan Li, and YunJie Li. Fuzzy artmap network and clustering for streaming classification under emerging new classes. In *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, pages 1–5, 2019.

[228] Y. Wang, Y. Ding, X. He, X. Fan, C. Lin, F. Li, T. Wang, Z. Luo, and J. Luo. Novelty Detection and Online Learning for Chunk Data Streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2400–2412, 2021.

[229] Shipeng Yan, Jiale Zhou, Jiangwei Xie, Songyang Zhang, and Xuming He. An em framework for online incremental learning of semantic segmentation. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 3052–3060, New York, NY, USA, 2021. Association for Computing Machinery.

[230] Jia Ming Yeoh, Fabio Caraffini, Elmina Homapour, Valentino Santucci, and Alfredo Milani. A Clustering System for Dynamic Data Streams Based on Metaheuristic Optimisation. *MATHEMATICS*, 7(12), 2019.

[231] Yogita and Durga Toshniwal. A framework for outlier detection in evolving data streams by weighting attributes in clustering. *Procedia Technology*, 6:214–222, 2012. 2nd International Conference on Communication, Computing & Security [ICCCS-2012].

[232] Yogita and Durga Toshniwal. Clustering techniques for streaming data-a survey. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 951–956, 2013.

[233] William J Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

[234] Poorya ZareMoodi, Hamid Beigy, and Sajjad Kamali Siahroudi. Novel class detection in data streams using local patterns and neighborhood graph. *NEUROCOMPUTING*, 158:234–245, 2015.

[235] Poorya ZareMoodi, Sajjad Kamali Siahroudi, and Hamid Beigy. A support vector based approach for classification beyond the learned label space in data streams. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, page 910–915, New York, NY, USA, 2016. Association for Computing Machinery.

[236] Poorya ZareMoodi, Sajjad Kamali Siahroudi, and Hamid Beigy. Concept-evolution detection in non-stationary data streams: a fuzzy clustering approach. *KNOWLEDGE AND INFORMATION SYSTEMS*, 60(3):1329–1352, 2019.

[237] Dan Zhang, Yan Liu, and Luo Si. Serendipitous learning: Learning beyond the predefined label space. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 1343–1351, New York, NY, USA, 2011. Association for Computing Machinery.

[238] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: A new data clustering algorithm and its applications. *Data mining and knowledge discovery*, 1(2):141–182, 1997.

[239] Z. Zhang, Y. Li, C. Jin, and M. Gao. Adaptive matrix sketching and clustering for semisupervised incremental learning. *IEEE Signal Process Lett*, 25(7):1069–1073, 2018.

[240] Zhao Zhang, Yong Zhang, Da Guo, and Mei Song. A scalable network intrusion detection system towards detecting, discovering, and learning unknown attacks. *International Journal of Machine Learning and Cybernetics*, 12(6):1649–1665, Jun 2021.

[241] Xiulin Zheng, Peipei Li, Xuegang Hu, and Kui Yu. Semi-supervised classification on data streams with recurring concept drift and concept evolution. *KNOWLEDGE-BASED SYSTEMS*, 215, 2021.

[242] Da-Wei Zhou, Yang Yang, and De-Chuan Zhan. Detecting sequentially novel classes with stable generalization ability. In Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty, editors, *Advances in Knowledge Discovery and Data Mining*, pages 371–382, Cham, 2021. Springer International Publishing.

[243] Yuxun Zhou, Reza Arghandeh, and Costas J. Spanos. Online learning of contextual hidden markov models for temporal-spatial data analysis. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6335–6341, 2016.

[244] Yong-Nan Zhu and Yu-Feng Li. Semi-supervised streaming learning with emerging new labels. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):7015–7022, Apr. 2020.

[245] Indre Zliobaite. Learning under concept drift: an overview. *CoRR*, abs/1010.4784, 2010.

[246] E. Özbilge. On-line expectation-based novelty detection for mobile robots. *Rob Autom Syst*, 81:33–47, 2016.

[247] R. Šajina, N. Tanković, and D. Etinger. Novel class detection in non-stationary streaming environment with a discriminative classifier. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1109–1113, 2020.