PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

EDUARDO TIEPPO

CLASSIFICATION OF HIERARCHICAL DATA STREAMS USING SUMMARIZATION TECHNIQUES

CURITIBA

2023

EDUARDO TIEPPO

CLASSIFICATION OF HIERARCHICAL DATA STREAMS USING SUMMARIZATION TECHNIQUES

Thesis submitted to the Graduate Program in Informatics (PPGIa) of the Pontifícia Universidade Católica do Paraná (PUCPR) in partial fulfillment of the requirements for the degree of Doctor in Informatics.

Major field: Computer Science.

Advisor: Prof. Júlio Cesar Nievola, Ph.D.
Co-advisor: Prof. Jean Paul Barddal, Ph.D.

CURITIBA

2023

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Edilene de Oliveira dos Santos CRB 9 / 1636

T563c
2023

Tieppo, Eduardo
Classification of hierarchical data streams using summarization techniques / Eduardo Tieppo ; Advisor: Júlio Cesar Nievola ; Co-advisor: Jean Paul Barddal. -- 2023
113 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2023.
Bibliografia: f. 105-113

1. Informática. 2. Computação de alto desempenho. 3. Classificação. 4. Hierarquias. 5. Armazenamentos de dados. 6. Sistemas de transmissão de dados. I. Nievola, Júlio Cesar. II. Barddal, Jean Paul. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título

CDD 20. ed. – 004

PUCPR
GRUPO MARISTA

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 24 de maio de 2023.

45-2023

<u>DECLARAÇÃO</u>

Declaro para os devidos fins, que **EDUARDO TIEPPO** defendeu a tese de Doutorado intitulada "**CLASSIFICATION OF HIERARCHICAL DATA STREAMS USING SUMMARIZATION TECHNIQUES**", na área de concentração Ciência da Computação no dia 23 de fevereiro de 2023, no qual foi aprovada.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

[illegible]

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

ACKNOWLEDGMENTS

I am very fortunate to be able to rely on the support of my parents Claudio and Berenice and my sister Paula. Thank you all for showing me how to walk.

Hellen, you make every moment better. Thank you for always believing in me and supporting me. When we were not poring over our research, you were always there for me with the best coffee and popcorn. Thank you, my life. I love you.

This journey also would not have been possible without the encouragement of some close friends. Thank you, Gledson, for talking to me about research autonomy. Thank you, Rodolfo, for introducing me to PPGIa and helping me with various technical problems. Thank you, Gui, for your willingness to help me again and again with statistical delusions. And thank you, João Pedro, for your partnership and support during bad ideas and outbursts.

It would not have been possible to perform this research without the intellectual support and shared experience of my advisors Júlio Nievola and Jean Barddal. I am immensely thankful for your patience and for the time spent with me, even reassuring me. It was an honor. I will use your manners as a reference when guiding my students.

Finally, I would like to thank the Instituto Federal do Paraná (IFPR), the Pontifícia Universidade Católica do Paraná (PUCPR), the Graduate Program in Informatics (PPGIa), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for their structural and financial support during this research.

“*A que parece de limão é de groselha e tem gosto de tamarindo.*
A que parece de groselha é de tamarindo com sabor de limão.
E a que parece de tamarindo é de limão com sabor de groselha.”

Lighthearted metaphor of what real datasets used in Machine
Learning look like – by Chaves – in “Chaves - Nem todos os bons
negócios são negócios da China”, 1977.

ABSTRACT

The classification of hierarchical data streams inherits challenges from its foundation areas, i.e., hierarchical classification and data stream classification. In data stream classification, learning models handle the class structure as flat and without relationships between the classes, losing potentially valuable information within the class taxonomy. Meanwhile, in hierarchical classification, learning models assume finite and stationary data, with models not updating themselves, disregarding time, or incoming data. These assumptions do not reflect real hierarchical data stream problems, where class labels are organized in a structured hierarchy with parent and child nodes, and the underlying distribution of the hierarchical data stream is likely to change over time. This thesis provides an experimental study in the hierarchical data stream classification area by proposing and evaluating learning models suitable for working with potentially unbounded hierarchical data streams. These models use summarization techniques to store or represent data using limited computational resources. Therefore, this work also analyzes how these data summarization strategies affect the learning models regarding prediction correctness and computational performance. This work begins with a systematic literature review comprehending a formal definition of the hierarchical data stream classification field and a description of existing related work. Then, various hierarchical data stream sets are identified, adapted, and arranged. Next, this work presents novel learning models for hierarchical data stream classification based on nearest neighbors, clustering techniques, and gaussian probabilities. Finally, a benchmark for the hierarchical data stream classification field is established, comparing the proposed methods with related work. The results obtained with the proposed methods show that the learning models using summarization techniques, when compared regarding prediction correctness and computational performance, could achieve better rates in one criterion without significant impacts on the other one.

Keywords: Hierarchical Classification; Data Stream Classification; Classification of Hierarchical Data Streams, Data Summarization.

RESUMO

A classificação de fluxos de dados hierárquicos herda simultaneamente os desafios de suas áreas bases, isto é, a classificação hierárquica e a classificação de fluxos de dados. Na classificação de fluxos de dados, os modelos de aprendizado compreendem as classes de um problema como planas e independentes, perdendo toda a informação potencialmente útil contida na taxonomia de classes. Já na classificação hierárquica, modelos de aprendizado assumem dados finitos e estacionários, e não se atualizam ao longo do tempo, independentemente da chegada de novos dados. Essas premissas não são adequadas quando considerados problemas de fluxos de dados hierárquicos, onde as classes são organizadas hierarquicamente através de nós pais e nós filhos, e a distribuição de dados provavelmente mudará ao longo do tempo. Esta tese apresenta um estudo experimental na área de classificação de fluxos de dados hierárquicos, por meio da proposta e avaliação de modelos de aprendizado adequados para trabalhar com fluxos de dados hierárquicos potencialmente ilimitados. Esses modelos usam técnicas de sumarização para armazenar ou representar dados utilizando recursos computacionais limitados; este trabalho também analisa como essas estratégias de sumarização de dados afetam os modelos de aprendizado em relação a taxas de acerto de predição e de desempenho computacional. O trabalho apresenta uma revisão sistemática de literatura sobre o tema, incluindo uma definição formal da área de classificação de fluxos de dados hierárquicos e a descrição de trabalhos relacionados existentes. Em seguida, vários conjuntos de fluxos de dados hierárquicos são identificados, adaptados e organizados. Posteriormente, são introduzidos novos modelos de aprendizado para a classificação de fluxos de dados hierárquicos baseados em técnicas de vizinhos próximos, agrupamento e probabilidades gaussianas. Por fim, é apresentada uma comparação geral entre todos os métodos propostos e trabalhos relacionados a fim de estabelecer um ponto de referência de avaliação para novos modelos de aprendizado na área. Os resultados obtidos mostram que os métodos propostos, quando avaliados com critérios de taxas de acerto de predição e de desempenho computacional, podem alcançar melhores taxas em um dos critérios sem resultar em impactos significativos (perdas) no outro critério.

Palavras-chave: Classificação Hierárquica; Classificação de Fluxos de Dados; Classificação de Fluxos de Dados Hierárquicos; Sumarização de Dados.

LIST OF ALGORITHMS

Algorithm 3.1 - Global kNN-hDS: Adaptive global k-Nearest Neighbors for hierarchical data stream classification..45

Algorithm 3.2 - Local kNC-hDS: Local k-Nearest Centroids for hierarchical data stream classification...50

Algorithm 3.3 - Local Dribble-hDS: Local Dribble for hierarchical data stream classification...51

Algorithm 3.4 - Global kNC-hDS: Global k-Nearest Centroids for hierarchical data stream classification..53

Algorithm 3.5 - Global Dribble-hDS: Global Dribble for hierarchical data stream classification...54

Algorithm 3.6 - GNB-hDS: Gaussian Naive Bayes for hierarchical data stream classification...58

Algorithm 3.7 - GNB-hDS: Gaussian Naive Bayes (adaptive) for hierarchical data stream classification..60

Algorithm 3.8 - Yeo-Johnson Power transformation: traditional application. 62

Algorithm 3.9 - Incremental Yeo-Johnson Power Transformation for data stream classification...65

LIST OF FIGURES

Figure 2.1 - Illustrative example of general approaches of (a) Flat Classification and (b) Hierarchical Classification. 24
Figure 2.2 - Illustrative example of general approaches of (a) Traditional (batch) classification and (b) Data Stream Classification. 27
Figure 2.3 - Illustration of different time window kinds: (a) Landmark window, (b) Sliding window, (c) Fading window, and (d) Tilted-time window. 29
Figure 3.1 - Illustrative comparison between LCPN and GC hierarchical classification approaches. 43
Figure 3.2 - Global approach via label path analysis using label frequency per hierarchy level. 44
Figure 3.3 - General view of Global kNC-hDS and Global Dribble-hDS learning process. 52
Figure 3.4 - Illustration of GNB-hDS method. 56
Figure 3.5 - Illustration of GNB-hDS method using adaptive data stream handling. 59
Figure 3.6 - Overview of the incremental Yeo-Johnson Power Transformation setup in a data stream scenario. 63
Figure 4.1 - kNC-hDS and Dribble-hDS results: critical differences chart for hF rates. 81
Figure 4.2 - kNC-hDS and Dribble-hDS results: critical differences chart for $inst/s$ rates. 81
Figure 4.3 - GNB-hDS results: critical differences chart for hF rates. 84
Figure 4.4 - GNB-hDS results: critical differences chart for $inst/s$ rates. 86
Figure 4.5 - Incremental Yeo-Johson results: critical differences chart for Shapiro-Wilk W Statistic on hierarchical data streams. 88
Figure 4.6 - Incremental Yeo-Johson results: critical differences chart for hF rates. 89
Figure 4.7 - Overall results: critical differences chart for hF rates obtained by methods with averaged best-performing parameters settings. 92
Figure 4.8 - Overall results: critical differences chart for $inst/s$ rates obtained by methods with averaged best-performing parameters settings. 93
Figure 4.9 - Overall results: critical differences chart for hF rates obtained by methods on best trade-off performance. 96
Figure 4.10 - Overall results: critical differences chart for $inst/s$ rates obtained by methods on best trade-off performance. 97

LIST OF TABLES

Table 2.1 - Related work of the hierarchical data stream classification area. 32

Table 2.2 - Adherence of the selected studies to the hierarchical data stream classification properties. 36

Table 3.1 - Overview of the proposed methods and their summarization and prediction strategies. 42

Table 3.2 - Specification of the proposed methods concerning the hierarchical data stream classification properties. 42

Table 4.1 - Hierarchical data stream sets used in the experiments. 68

Table 4.2 - Parameter settings used on methods across experiments. 69

Table 4.3 - GNB-hDS variants specification. 70

Table 4.4 - Global kNN-hDS results: Hierarchical F-Measure (%) obtained by methods considering the averaged best-performing parameter setting across all datasets. 73

Table 4.5 - Global kNN-hDS results: instances per second rates obtained by methods with averaged best-performing parameters settings. 74

Table 4.6 - Global kNN-hDS results: average Hierarchical F-Measure (%) obtained by methods on each variation of n. 75

Table 4.7 - Global kNN-hDS results: Average instances per second rates obtained by methods on each variation of n. 75

Table 4.8 - Global kNN-hDS results: overall average Hierarchical F-Measure (%) and Instances per second rates obtained by methods. 76

Table 4.9 - kNC-hDS and Dribble-hDS results: Hierarchical F-Measure (%) obtained by methods considering the averaged best-performing parameter setting across all datasets. 77

Table 4.10 - Local kNC-hDS and Local Dribble-hDS results: instances per second rates obtained by methods with averaged best-performing parameters settings. 78

Table 4.11 - kNC-hDS and Dribble-hDS results: average Hierarchical F-Measure (%) obtained by methods on each variation of n. 79

Table 4.12 - kNC-hDS and Dribble-hDS results: Average instances per second rates obtained by methods on each variation of n. 80

Table 4.13 - kNC-hDS and Dribble-hDS results: overall average Hierarchical F-Measure (%) and Instances per second rates obtained by methods. 81

Table 4.14 - GNB-hDS results: Hierarchical F-Measure (%) obtained by methods considering the averaged best-performing parameter setting across all datasets. 83

Table 4.15 - GNB-hDS results: instances per second rates obtained by methods with averaged best-performing parameters settings. 85

Table 4.16 - GNB-hDS results: average Hierarchical F-Measure (%) obtained by adaptive methods on each variation of w. 87

Table 4.17 - Incremental Yeo-Johson results: Shapiro-Wilk W Statistic for original, transformed, and incrementally transformed hierarchical data streams. 87

Table 4.18 - Incremental Yeo-Johson results: Hierarchical F-Measure (%) obtained with original, transformed, and incrementally transformed hierarchical data streams. 89

Table 4.19 - Overall results: Hierarchical F-Measure (%) rates obtained by methods with averaged best-performing parameters settings. 91

Table 4.20 - Overall results: instances per second rates obtained by methods with averaged best-performing parameters settings. 92

Table 4.21 - Overall results: Hierarchical F-Measure (%) rates obtained by methods on best trade-off performance. 95

Table 4.22 - Overall results: instances per second rates obtained by methods on best trade-off performance. 96

Table 4.23 - Overall results: MCDM-WPM (values) of methods with averaged best performance. 98

Table 4.24 - Overall results: MCDM-WPM (rankings) of methods with averaged best performance. 98

Table 4.25 - Overall results: MCDM-WPM (values) of methods with best trade-off performance. 99

Table 4.26 - Overall results: MCDM-WPM (rankings) of methods with best trade-off performance. 100

CONTENTS

1 INTRODUCTION 15

1.1 RESEARCH AIM AND OBJECTIVES 18

1.2 HYPOTHESIS 19

1.3 CONTRIBUTIONS 19

1.4 PUBLICATIONS 20

1.5 FINANCIAL SUPPORT 21

1.6 THESIS ROADMAP 22

2 THEORETICAL BACKGROUND 23

2.1 FOUNDATION AREAS 24

2.1.1 Hierarchical Classification 24

2.1.2 Data Stream Classification 27

2.2 PROBLEM STATEMENT 30

2.3 RELATED WORK AND STATE OF THE ART 32

2.3.1 Related Work 32

2.3.2 Considerations concerning the state of the art 34

2.4 A NOTE ON THE RESEARCH GAP 37

3 PROPOSED METHODS 40

3.1 OVERVIEW 40

3.2 GLOBAL K-NEAREST NEIGHBORS FOR HIERARCHICAL DATA STREAMS (GLOBAL KNN-HDS) 42

3.3 K-NEAREST CENTROIDS FOR HIERARCHICAL DATA STREAMS (KNC-HDS) AND DRIBBLE FOR HIERARCHICAL DATA STREAMS (DRIBBLE-HDS) 45

3.4 GAUSSIAN NAIVE BAYES FOR HIERARCHICAL DATA STREAMS (GNB-HDS) 54

4 ANALYSIS 67

4.1 EXPERIMENTAL PROTOCOL 67

4.1.1 Datasets 67

4.1.2 Validation protocol 68

4.2 GLOBAL K-NEAREST NEIGHBORS FOR HIERARCHICAL DATA STREAMS (GLOBAL KNN-HDS) 73

4.3 K-NEAREST CENTROIDS FOR HIERARCHICAL DATA STREAMS (KNC-HDS) AND DRIBBLE FOR HIERARCHICAL DATA STREAMS (DRIBBLE-HDS) 76

4.4 GAUSSIAN NAIVE BAYES FOR HIERARCHICAL DATA STREAMS (GNB-HDS) 83

4.5 BENCHMARKING 90

5 CONCLUSION 102

5.1 FULFILLED OBJECTIVES AND CONTRIBUTIONS 102

5.2 IMPLICATIONS FOR FUTURE WORK 103

REFERENCES 105

1 INTRODUCTION

Pattern recognition is the primary task of Machine Learning. It consists in searching for regularities in the data via algorithms and using these regularities to perform some action, such as clustering similar data or assigning them to different known classes (BISHOP; NASRABADI, 2006). This last example, in which data is related to predetermined classes, is known as classification.

The classification task involves developing, studying, and evaluating methods that receive previously labeled (or classified) input data, build a learning model capable of making predictions or decisions based on that data, and assign the most likely class to the unseen data according to the information seen during the previous training step (KOTSIANTIS, 2007).

Classification is a popular and recurring task in the Machine Learning area as it has been successfully applied to several domains. For instance, in biology, by classifying patients into different clinical groups or by identifying disease groups (ALIZADEH et al., 2000; ROSS et al., 2000; TARCA et al., 2007); in languages, in the organization and classification of documents (SEBASTIANI, 2002); in the classification of images and their contents (CHAPELLE; HAFFNER; VAPNIK, 1999); and even in astronomy, with the automatic cataloging of sky objects (FAYYAD; WEIR; DJORGOVSKI, 1993). More recent applications can be observed from biomedicine with the enhancement of Computer-Aided Diagnosis (YASSIN et al., 2018), to politics, with the detection of fake news and its impact on presidential elections (SHU et al., 2017).

Given the wide variety of problems in which classification techniques can be applied to, these techniques also need to deal with different kinds of data and respond to them accordingly (TSOUMAKAS; KATAKIS, 2007). Classification problems can have two exclusive (non-overlapping) classes (binary classification), several mutually exclusive classes (multi-class classification), many potentially simultaneous classes (multi-labeled classification), and, finally, classes that are layered in a hierarchical structure (hierarchical classification) (SOKOLOVA; LAPALME, 2009).

It is not straightforward to find a starting point or a chronological development of these distinct classification subtasks explicitly on individual studies across the literature. Therefore, it is also complex to define precedence between the subtasks, either chronologically or concerned with their complexities (KOTSIANTIS, 2007).

However, the literature usually presents such subtasks similarly to that presented earlier in this text. Binary classification can be understood as a specific case of multiclass classification (where the number of classes is two), just as the multiclass classification can be understood as a specific case of multilabel classification (in which the number of predicted labels is one). Finally, hierarchical classification problems can be decomposed into other types of classification, ignoring the hierarchy present in the classes (SILLA; FREITAS, 2011).

Despite classification being an established research topic in the literature, research fields in which classification is applied to have also evolved simultaneously, resulting in changes in their processes, such as the way data and information are obtained and stored. Yet, the classification field has mainly focused efforts on batch learning, assuming that all data used to recognize patterns will be available to the model at the same time, on a well-defined training step (GAMA, 2010).

However, this assumption no longer reflects many real-world scenarios where classification is applied based on data streams instead of batch data. Therefore, the classification task needs to adapt itself to handle these ever-changing environments, being able to incorporate new data into its learning processes on the fly (GAMA, 2010).

Furthermore, data stream classification brings new challenges common to all classification subtasks. Data stream continuously provides potentially unbounded data over time, and, thus, learning models need to process data constrained by limited computational resources. Also, due to the intrinsic time component, data streams are expected to be ephemeral and provide non-stationary data. Therefore, learning models need to respond appropriately to newest data (BIFET; KIRKBY, 2009; GAMA, 2010). It is also relevant to state that as data stream classification regards the way the data is provided to the learning model (examples are provided over time rather than in batch), it can be simultaneous to all other kinds of classification problems, and it has been revisiting many aspects related to classification in the last years (GAMA, 2010).

New classification models and algorithms have been proposed or fitted to work with data streams, including new assessment platforms and protocols (AGGARWAL, 2007; BIFET et al., 2010; GAMA; SEBASTIÃO; RODRIGUES, 2013). Thus far, data stream classification research has focused its main efforts on binary, multiclass, and multilabel classification (BIFET et al., 2010; BIFET; KIRKBY, 2009; READ et al., 2012).

Nonetheless, data stream classification seems to have not yet explored the hierarchical classification subtask. On one hand, recently proposed data stream classification methods do not consider any kind of hierarchy in their designs (BAHRI et al., 2021), and literature reviews regarding data stream classification do not even mention the hierarchical subtask (GOMES et al., 2019; KREMPL et al., 2014; RAMÍREZ-GALLEGO et al., 2017). On the other hand, most studies on hierarchical classification do not consider the data stream scenario, handling only batch and stationary data (DUMAIS; CHEN, 2000; FREITAS; CARVALHO, 2007; KOSMOPOULOS et al., 2015; SILLA; FREITAS, 2011).

Novel research at the intersection of these two areas (data stream classification and hierarchical classification) seems likely. In hierarchical classification, advances in obtaining data may require new methods that are able to process data streams, similarly to ongoing research regarding the other classification subtasks. In data stream classification, new approaches to handle existing problems can incorporate naturally existing (but not used yet) hierarchies in the data, such as hierarchical contents of Web pages, hierarchical structures of sensor networks, biological taxonomy of animals, etc. (GAMA, 2010).

Besides, research conducted on hierarchical classification stated that learning models can benefit from incorporating a naturally existent class hierarchy into the classification process, resulting in higher classification prediction rates or more information (FREITAS; CARVALHO, 2007; SILLA; FREITAS, 2011). Yet, recent reviews on data stream classification do not even mention hierarchical classification approaches.

Studies published across different research areas proposed methods improperly associated with the classification of hierarchical data streams. For instance, studies have proposed methods regarding the classification of hierarchical data streams, yet, either use a batch configuration for training using the entire dataset or do not consider any changes in the data distribution (CAO et al., 2018; HUANG et al., 2019; KHOWAJA et al., 2018; PUROHIT et al., 2014; WANG; GONG; GUO, 2009). At bottom, those methods are hierarchical classification methods in which the data source was a stream, but it is assumed to be stationary, and models are not updated.

Furthermore, current state-of-the-art methods introduced for the classification of hierarchical data streams present limitations when applied to real-world problems, as they use complete representations of data and eventually perform redundant steps in their learning models (PARMEZAN; SOUZA; BATISTA, 2018). In other words, methods that are computationally heavy-weighted in terms of processing time and memory consumption may represent infeasible strategies for handling potentially unbounded hierarchical data streams (BIFET; KIRKBY, 2009; GAMA, 2010; PRASAD; AGARWAL, 2016).

This thesis provides an exploratory, experimental study on the hierarchical data stream classification area by proposing and evaluating learning models fitted to work with potentially unbounded hierarchical data streams. These models consider the constraints of hierarchical classification and data stream classification concomitantly. In other words, such learning models handle class hierarchies, are updatable over time, adapt to changes in data behavior, and are computationally light-weighted regarding processing time and memory consumption.

In this sense, the hierarchical data stream models proposed in this thesis use summarization techniques to store or represent data with constrained computational resource usage. Therefore, this thesis also presents an analysis of how these data summarization strategies impact learning models regarding prediction correctness and computational performance.

1.1 RESEARCH AIM AND OBJECTIVES

This project aims at proposing and evaluating learning models with data summarization techniques for classifying hierarchical data streams. This thesis comprehends the following research objectives to fulfill the research aim:

i. To review hierarchical classification literature;
ii. To review data stream classification literature;
iii. To perform a systematic literature review concerning the hierarchical data stream classification;
iv. To provide formalizations for the hierarchical data stream classification problem;
v. To identify and arrange hierarchical data stream sets fitted for the classification task;
vi. To propose learning models that use data summarization techniques for classifying hierarchical data streams;
vii. To evaluate the proposed learning models.

1.2 HYPOTHESIS

This thesis encompasses several minor hypotheses discussed and validated throughout this work in specific sections. Yet, all of them are related to the two central hypotheses defined below.

- H_{01}: There is no significant difference between the hierarchical evaluation metrics obtained by traditional hierarchical data stream classifiers and hierarchical data stream classifiers that use data summarization techniques.
- H_{A1}: There is a significant difference between the hierarchical evaluation metrics obtained by traditional hierarchical data stream classifiers and hierarchical data stream classifiers that use data summarization techniques.
- H_{02}: There is no significant difference between the computational performance of traditional hierarchical data stream classifiers and hierarchical data stream classifiers that use data summarization techniques.
- H_{A2}: There is a significant difference between the computational performance of traditional hierarchical data stream classifiers and hierarchical data stream classifiers that use data summarization techniques.

Note that H_{01} focuses on prediction correctness, while H_{02} regards computational performance. Therefore, the initial expectation of this thesis is to propose learning models that support at least one of the hypotheses. Thus, to propose learning models capable of obtaining better computational performance without significant impacts on prediction correctness or better prediction correctness with similar or improved computational performance.

1.3 CONTRIBUTIONS

This thesis provides learning models based on data summarization techniques tailored to hierarchical data stream classification. In addition to such methods, it also contributes to the formalization and consolidation of the hierarchical data stream classification area.

Specifically, the explicit contributions of this project are as follows:

- The hierarchical data stream classification area is unclear and, until this very point, has been addressed only sparsely in the literature. This thesis introduces a proper formalization, both theoretically and practically (Section 2.2). Furthermore, this thesis maps the area with a systematic literature review, covering the main problems, datasets, classification algorithms, evaluation metrics, and research gaps related to the hierarchical data stream classification task, also discussing the adherence of related works to the field and specifying the state of the art (Sections 2.3 and 2.4).

- Datasets fitted to hierarchical data stream classification are still scarce in the literature. This thesis identifies, adapts, and arranges various datasets and makes them available for further research in the classification of hierarchical data stream area (Section 4.1.1).
- Up to now, studies related to hierarchical classification do not use data streams as input data for their processes; similarly, studies related to data stream classification do not consider possible class hierarchies in the dataset. In this thesis, four new main methods are proposed for the classification of hierarchical data streams: (i) "Global kNN-hDS" – based on nearest neighbors (Section 3.2), (ii) "kNC-hDS" and (iii) "Dribble-hDS" – based on clustering techniques (Section 3.3), and (vi) "GNB-hDS" – based on gaussian probabilities (Section 3.4).
- As a product originating from the experimental improvement of the GNB-hDS method, Section 3.4 also introduces an incremental adaptation of the well-known Yeo-Johnson Power Transformation (YEO; JOHNSON, 2000). This proposed transformation can be used in the hierarchical data stream area as an attached data pre-processing step to reduce the skewness of the data and improve the prediction results.
- The comparison and placement of new learning models in the literature are not straightforward unless the authors follow the same recommendations, experimental assessment procedures, and datasets. In this sense, this thesis provides a benchmark for the hierarchical data stream classification area, comparing all the proposed methods with related work under the same experimental protocol (Section 4.5).

1.4 PUBLICATIONS

The main contributions of this thesis were also reported in the following research manuscripts/articles:

- Eduardo Tieppo, Roger Robson dos Santos, Jean Paul Barddal, and Júlio Cesar Nievola. Hierarchical Classification of Data Streams: A Systematic Literature Review. Artificial Intelligence Review. Pages 1-40. 2021.
- Eduardo Tieppo, Jean Paul Barddal, and Júlio Cesar Nievola. Adaptive Global k-Nearest Neighbors for Hierarchical Classification of Data Streams. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC). Pages 631-636. 2021.
- Eduardo Tieppo, Jean Paul Barddal, and Júlio Cesar Nievola. Classifying Potentially Unbounded Hierarchical Data Streams with Incremental Gaussian Naive Bayes. In: Brazilian Conference on Intelligent Systems (BRACIS). Pages 421-436. 2021.
- Eduardo Tieppo, Jean Paul Barddal, and Júlio Cesar Nievola. Automatic Disease Vector Mosquitoes Identification via Hierarchical Data Stream Classification. In: The 37th ACM/SIGAPP Symposium on Applied Computing (SAC). Pages 1005-1012. 2022.
- Eduardo Tieppo, Jean Paul Barddal, and Júlio Cesar Nievola. Classifying Hierarchical Data Streams using Global Classifiers and Summarization Techniques. In: The 2022 International Joint Conference on Neural Networks (IJCNN). Pages 1-8. 2022.
- Eduardo Tieppo, Jean Paul Barddal, and Júlio Cesar Nievola. Improving Data Stream Classification using Incremental Yeo-Johnson Power Transformation. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC). Pages 3286-3292. 2022.
- (Submitted, under review) Eduardo Tieppo, Jean Paul Barddal, and Júlio Cesar Nievola. Adaptive Learning on Hierarchical Data Streams using Window Weighted Gaussian Probabilities. 2023.

1.5 FINANCIAL SUPPORT

This thesis was financially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), a research agency from the Ministry of Education from Brazil (Ministério da Educação – MEC), within the Programa de

Suporte à Pós-Graduação de Instituições de Ensino Superior (PROSUP), regulated by the ordinance “Portaria CAPES nº 181, de 18 de dezembro de 2012”.

1.6 THESIS ROADMAP

This thesis is organized as follows:

- The current section introduced the rationale for this thesis, comprehending the research aim and objectives to be achieved, the assumptions that underlie the central hypotheses of this project, and the tangible contributions.
- Section 2 provides the theoretical background of this thesis, including both foundations areas (hierarchical classification and data stream classification) and the specification of the hierarchical data stream classification area, with formalizations, state of the art, related works, and research gaps addressed by this thesis.
- Section 3 is dedicated to presenting the learning models proposed in this thesis. Each part presents a method (or a group of similar methods) based on a shared concept. Sections 3.2, 3.3, and 3.4 describes, in that order, the proposed methods based on nearest neighbors, clustering, and gaussian probabilities. The sections include the description of the learning models, their specific definitions, and algorithms. As a topic related to gaussian probabilities, Section 3.4 also presents the proposed data transformation mentioned in the contributions above.
- Section 4 provides the experimental setup and the analysis performed for comparing the proposed learning models with related works and with each other. The first part (Section 4.1) describes the experimental protocol and the hierarchical data stream sets used in the experiments. Following the same organization of Section 3, Sections 4.2, 4.3, and 4.4 show the experimental results obtained by the methods based on nearest neighbors, clustering, and gaussian probabilities. The last section (4.5) shows a multiple comparison experiment between all proposed methods.
- Finally, Section 5 concludes this thesis by summing up the fulfillment of the research objectives, outlining the contributions, and stating implications for further research.

2 THEORETICAL BACKGROUND

Hierarchical data stream classification emerges from the intersection of its foundation areas: hierarchical classification and data stream classification. Although both foundation areas are well-established research topics, their characteristics are usually not addressed concomitantly in the literature.

Moreover, current state-of-the-art techniques are not able to deal with hierarchical data stream classification problems directly. Hierarchical classification techniques cannot handle changing and potentially unbounded data, while data stream classification techniques do not account for hierarchical relationships on classes from data samples (TIEPPO et al., 2021).

Comprehensive reviews on hierarchical classification were presented in (SILLA; FREITAS, 2011), (NAIK; RANGWALA, 2018), and (DEFIYANTI; WINARKO; PRIYANTA, 2019), in which grounding concepts and terminologies of the area were formally defined.

Similarly, comprehensive reviews of data stream classification were presented in (GOMES et al., 2019), (WANKHADE; DONGRE; JONDHALE, 2020) (BAHRI et al., 2021), showing that a fair amount of effort has been devoted to scenarios where data are made available as a stream and how its challenges can be tackled.

These studies show successful approaches and future challenges in both areas but do not present a perspective comprising both areas together. In other words, despite the research conducted on hierarchical classification and data stream classification areas separately, there is a lack of studies in hierarchical data stream classification that consider the main characteristics of these kinds of problems together, such as hierarchical non-stationary data.

This section introduces and characterizes the hierarchical data stream classification area. First, Section 2.1 reviews both foundation areas of hierarchical data stream classification: hierarchical classification and data stream classification. Section 2.2 formalizes the hierarchical data stream classification problem. Section 2.3 describes the state of the art and related works based on a systematic literature review. Finally, Section 2.4 presents a research gap on which this thesis relies, from the perspective of the limitations of existing state-of-the-art learning models.

2.1 FOUNDATION AREAS

2.1.1 Hierarchical Classification

In hierarchical classification, instances are assigned to a label (class) that is part of a label path, where inner labels in the path represent hierarchical relationships with the outer labels. Classes are arranged in hierarchical structures, where nodes represent the classes and specialized nodes represent specific classes of their general nodes (DEFIYANTI; WINARKO; PRIYANTA, 2019; FREITAS; CARVALHO, 2007; SILLA; FREITAS, 2011).

Figure 2.1 compares a general approach of (a) flat classification and (b) hierarchical classification in an illustrative problem. The class taxonomy can be used to lead to specific decisions about the classes by splitting the context complexity. In flat classification, the decision must be made while considering all the classes of the problem (all the possible song genres). Meanwhile, the hierarchical classification concerns an existent class taxonomy, which can be used to make first smaller and generic decisions about the problem (in the example, to decide first between Rock and R&B genres), and then the specific ones.

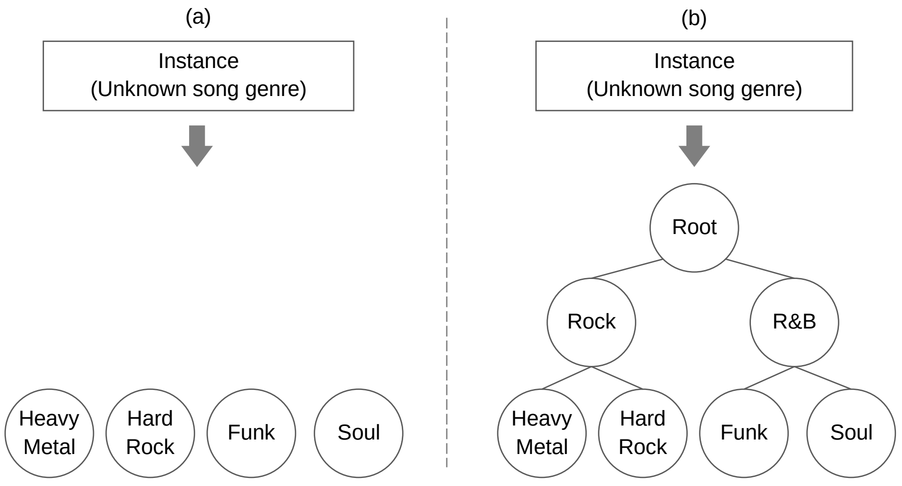


Figure 2.1 - Illustrative example of general approaches of (a) Flat Classification and (b) Hierarchical Classification.

A class taxonomy can be formalized as a regular concept hierarchy (LU, 1997) under a partially ordered set $(Y, \succ)$, where Y represents a finite set containing all target classes of a problem and the relation $\succ$ is defined as a subsumption relation ("is-a" relation) (DEFIYANTI; WINARKO; PRIYANTA, 2019; SILLA; FREITAS, 2011; WU; ZHANG; HONAVAR, 2005).

According to the authors in (SILLA; FREITAS, 2011), a hierarchical classification problem can be categorized in a 3-tuple (Υ, Ψ, Φ), where:

- Υ specifies the data structure used to represent the class taxonomy and may be modeled using a Tree or a Directed Acyclic Graph (DAG) representation, according to how many parent nodes the same node has.
- Ψ defines the label cardinality, where instances of a given problem can have only one single path of labels (SPL) associated with them or multiple paths of labels (MPL); and,
- Φ describes the label depth, where problems support partial depth labeling (PD), or actual classes of the problem are represented only in the leaf nodes with full depth labeling (FD).

Similarly, a hierarchical classification algorithm can be categorized in a 4-tuple $(\Omega, \Delta, \Xi, \Theta)$, where:

- Ω specifies if the algorithm supports Tree or Directed Acyclic Graph (DAG) as data structures;
- Δ indicates if the algorithm can assign to an instance at most one predicted label path (single path prediction - SPP), or it can potentially assign multiple predicted label paths (multiple path prediction - MPP);
- Ξ specifies if the algorithm always assigns leaf node classes as the last class of a predicted label path (mandatory leaf-node prediction - MLNP) or if it can predict label paths where the deeper class is at any hierarchy level (non-mandatory leaf-node prediction - NMLNP);
- Θ describes how the hierarchical classifier handles the class hierarchy in its algorithm, comprising Local classifier per node (LCN), Local classifier per level (LCL), Local classifier per parent node (LCPN), or Global classifier (GC). In the LCN approach, one binary classifier per class handles each class in the hierarchy (except the root node). In the LCPN approach, one multi-class classifier per class (except on the leaf nodes) predicts between its child nodes. In the LCL approach, one multi-class classifier per level predicts between all

nodes at the same level. Finally, in the GC approach, one single multi-class classifier is built to handle all classes using the hierarchy information.

Note that both categorizations are similar but use two different contexts: problems and algorithms. Nevertheless, some algorithm categories may be more suited to deal with specific categories of problems. For instance, a problem with full-depth labeling (FD) should use a Mandatory leaf-node prediction scheme (MLNP) to be able to predict the entire path of labels since predicting only part of it would not represent an actual class of the problem (SILLA; FREITAS, 2011).

Finally, the authors in (KIRITCHENKO et al., 2005) proposed three metrics able to measure the performance of a hierarchical classifier: hierarchical precision (hP), hierarchical recall (hR), and hierarchical F-Measure (hF). These metrics are variations of the traditional classification metrics (Precision, Recall, and F-Measure) but instances are associated with a path of labels and the entire path is evaluated.

The hP metric, depicted in Equation (1), computes the number of labels in a predicted label path ($\widehat{y_i}$) that are also components of the ground-truth label path (y_i) for the i-th instance. On the other hand, hR, depicted in Equation (2), quantifies the number of ground-truth labels comprised by the predicted label path for a given instance.

$$hP = \frac{\sum_i |\hat{y}_i \cap y_i|}{\sum_i |\hat{y}_i|} \tag{1}$$

$$hR = \frac{\sum_i |\hat{y}_i \cap y_i|}{\sum_i |y_i|} \tag{2}$$

Like traditional classification metrics, the hierarchical F-Measure, depicted in Equation (3), is the harmonic mean between hierarchical precision (hP) and hierarchical recall (hR). As in the traditional F-Measure, β weights hP and hR values (CERRI et al., 2015).

$$hF = (1 + \beta^2) \times \frac{hP \times hR}{(\beta^2 \times hP) + hR} \tag{3}$$

2.1.2 Data Stream Classification

Unlike traditional machine learning, where the dataset is static and can be accessed multiple times, data stream algorithms need to consider previously unmapped issues such as limited memory, single-pass data, readiness, and detection and adaptation to changes in the dataset (BIFET; KIRKBY, 2009; GAMA, 2010; GOMES et al., 2019; QUIÑONERO-CANDELA et al., 2008).

Figure 2.2 compares a (a) traditional classification process and a (b) data stream classification process. Dashed arrows represent data flow, dotted lines are optional processes, and solid arrows illustrate a model deployment. In traditional (or batch) classification, data are assumed to be static and completely available to the model at the training step; the dataset is then divided into subsets of training and testing data; the training data are submitted to the learning model that reviews them as many times as necessary, until obtaining a unique satisfactory model. This final model is then applied to the subset of testing data and provides predictions.

In contrast, in data stream classification, data are made available sequentially over time, and even a single instance can be provided to the model. The most common approach for handling streaming data is to process data on an instance basis. In this process, each arriving instance is tested by a current model resulting in a prediction, and, only after that, it is incorporated into the model (being used as training data). Next, the cycle restarts with a new instance from the data stream. Any processed instance must be eventually discarded to maintain the model's ability to process new instances since the data stream is potentially unbounded.

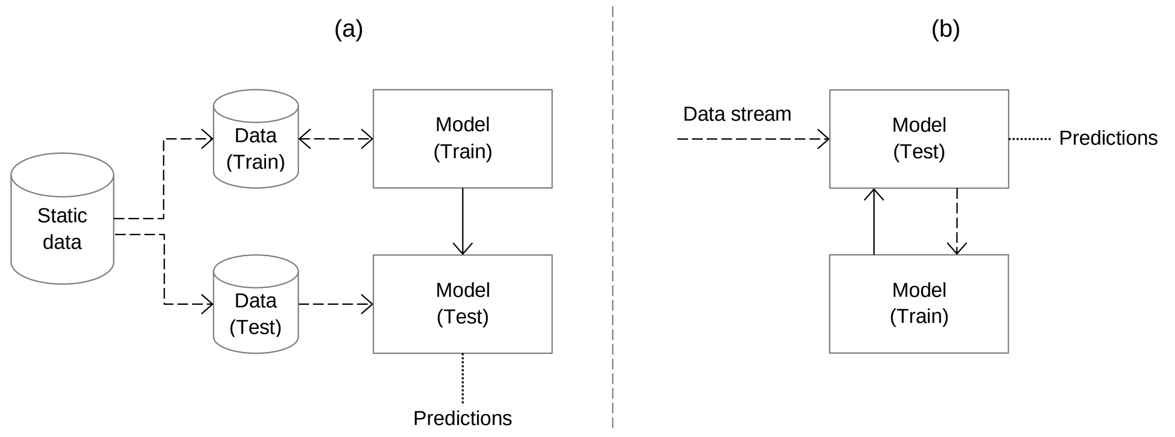


Figure 2.2 - Illustrative example of general approaches of (a) Traditional (batch) classification and (b) Data Stream Classification.

According to the works of (GAMA, 2010), (NGUYEN; WOON; NG, 2015), and (WIDMER; KUBAT, 1996), due to the temporal and unbounded traits of data streams, learning algorithms must meet several constraints:

- Single-pass: each instance in the stream should be examined just once and cannot be reused;
- Readiness: learning and prediction should be made in real-time or near real-time;
- Bounded Memory: the amount of input data is gigantic or potentially infinite; therefore, a summary of the data stream is usually calculated and stored, and approximate results are acceptable; and
- Concept drift detection: data streams are expected to be ephemeral due to the intrinsic time component, and, thus, the underlying data distribution is expected to change, a phenomenon named concept drift (GAMA et al., 2014; TSYMBAL, 2004).

As introduced above, a concept (C) is defined as a set of prior probabilities of the classes and class-conditional probability density function given by $C = \bigcup_{y \in Y}\{(P[y], P[\vec{x}|y])\}$. A concept drift occurs if, between two timestamps t_i and $t_j = t_i + \Delta$ with $\Delta > 1$, $C^{t_i} \neq C^{t_j}$ holds. In addition, concept drifts can be classified according to how the underlying distribution of data changes concerning t. For instance, when $C^{t_i} \neq C^{t_j}$ holds and $\Delta = 1$, an abrupt concept drift occurs; if $\Delta > 1$, the concept drift is gradual (BARDDAL et al., 2016; NGUYEN et al., 2012; TSYMBAL, 2004).

Since data streams are potentially unbounded, it is not possible to process all data at once. Thus, the data must be processed incrementally as data samples are made available. Data processing is performed according to different time window models, all of them sharing the same idea of heeding to specific portions of data (usually the most recent ones). These time windows can be of different types according to the way they process the data.

Figure 2.3 shows different time window types. In the Landmark window (a), there is interest in portions of data between instances called “landmarks”; landmarks can be defined based on time, the number of instances (mini-batches), and memory constraints; older or newer data have the same importance. In the Sliding window (b), there is more interest in the newer data and the time window slides along with time; data outside the window is discarded. In the Fading window (c), the data is weighted and there is more interest in newer data by assigning greater weights according to the data currency; thus, old data becomes less important in the learning process. Finally, in the Tilted-time window (d), there is interest in the most recent data by selecting instances based on elapsed time; the data are represented with different detail levels according to their age; thus, newer data has more instances and details and older data is represented with some smaller-scale pattern.

It is also important to notice that the kinds of time windows are not limited to the ones illustrated in Figure 2.3 and different kinds can even be mixed depending on the characteristics of the data (BARDDAL et al., 2017; NGUYEN; WOON; NG, 2015).

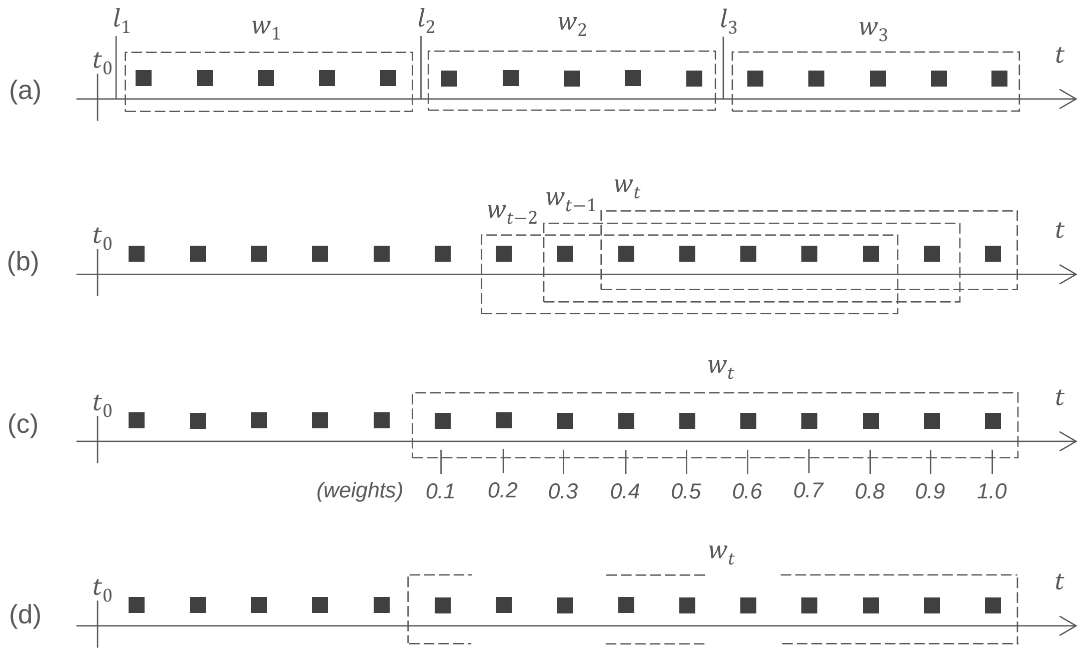


Figure 2.3 - Illustration of different time window kinds: (a) Landmark window, (b) Sliding window, (c) Fading window, and (d) Tilted-time window.

The main challenge in selecting a time window strategy is choosing a well-suited size due to the stability-plasticity dilemma (MERMILLOD; BUGAISKA; BONIN, 2013). Shorter windows may help the model to become more responsive to drifts (plasticity). On the other hand, larger windows may result in more stable models (stability) (BARDDAL et al., 2017; GAMA, 2010).

Traditionally in data mining, algorithms work in batch (offline) mode. With large-scale data, this becomes infeasible due to memory and time limitations. To deal with that, algorithms in data stream classification can work with adaptations of batch mode (mini-batches), be incremental or adaptive (GAMA et al., 2014).

Incremental algorithms regard updating or retraining the models with part of or all data as new instances become available. The Hoeffding Trees and Bayes models are examples of these algorithms (DOMINGOS; HULTEN, 2000; GAMA et al., 2014).

Because of updating or retraining the models with eventually larger datasets, computational resources need to be tracked and evaluated. Nevertheless, as previously mentioned, concept drift also needs to be considered in data stream scenarios. In this case, models also need to adapt themselves to react to these drifts (GAMA et al., 2014).

Adaptive algorithms also regard updating or retraining the models (as incremental ones), but the model includes strategies to forget the information previously learned. The adaptive models can be understood as advanced incremental learning models that are able to adapt to changes in data over time (GAMA et al., 2014).

Finally, concerning evaluation, the authors in (GAMA et al., 2014) proposed the prequential assessing method (or Interleaved Test-Then-Train (BIFET; KIRKBY, 2009)) to evaluate learning algorithms in streaming scenarios. In this process, each instance is used to test the model and the evaluation metrics, e.g., precision, recall, F-Measure, are updated. Next, the instance is used to train/update the model. As metrics are calculated for each instance, results are often summarized using some strategy, such as using maximum or mean values and considering some sampling frequency of the data stream.

2.2 PROBLEM STATEMENT

The hierarchical classification of data streams lies at the intersection of hierarchical classification and data stream classification, two well-established research areas. Consequently, this new area inherits characteristics and challenges from its base areas and differs from traditional classification in two key aspects.

First, concerning hierarchical classification, examples must be assigned to not one independent label (class) but to a label path representing one of many possible label paths composing the class taxonomy. Next, concerning data stream classification, the entire dataset containing examples for a training step is not available; instead, examples are provided to the model sequentially over time.

Hierarchical classification of data stream methods must use data streams as input to their learning processes, not only as a source of data but effectively processing portions of the data over time using the assumption that there is no complete dataset.

Regarding input data, let $hDS = [(\vec{x}^t, \vec{y}^t)]_{t=0}^{\infty}$ be a hierarchical data stream providing instances $(\vec{x}^t, \vec{y}^t)$, each of which arriving at a timestamp t, where $\vec{x}^t$ is a d-dimensional feature set and its values, and $\vec{y}^t$ is the corresponding ground-truth label path (hierarchically structured classes) for the given instance.

As mentioned above, class labels are organized under a regular concept hierarchy under a partially ordered set $(Y, \succ)$, where Y represents a finite set of all concepts and the relation $\succ$ is defined as an asymmetric, anti-reflexive and transitive subsumption (“is-a”) relation (SILLA; FREITAS, 2011).

The hierarchical classification of data streams can be formalized as a mapping function $f^t\colon \vec{x}^t \mapsto \vec{y}^t$, where a hypothesis f^t is continuously updated by features $\vec{x}^t$ to the corresponding labels $\vec{y}^t$ accurately (GAMA, 2010).

Also, concerning the data stream classification foundation area, a hierarchical data stream classifier must consider non-stationary data and, consequently, being able to adapt itself to possible concept drifts.

Considering a set C of prior probabilities of the classes and class-conditional probability density function given by $C = \bigcup_{y \in Y} \{(P[y], P[\vec{x}|y])\}$, a hierarchical data stream classifier must be able to update its mapping function f^t and capture the data dynamics if between two distinct timestamps t_i and $t_j = t_i + \Delta$ a concept drift occurs, i.e., C^{t_i} differs from C^{t_j}.

Finally, methods need to perform their processes using bounded computational resources (time and memory), examining each example only once according to their arrival and processing it in less time than the ratio in which new instances become available. Otherwise, the method will eventually need to drop incoming examples, or it will not be able to adapt quickly enough to handle concept drifts (BARDDAL et al., 2016; BIFET; KIRKBY, 2009).

2.3 RELATED WORK AND STATE OF THE ART

2.3.1 Related Work

Due to the recentness of the hierarchical data stream classification area, a systematic literature review (SLR) was carried out to investigate the main characteristics of the area.

The purpose of the SLR was to summarize and clarify the area guided by five research questions: (i) what kind of problems are handled by hierarchical classification of data streams, (ii) which datasets are frequently used in experiments in the existing studies, (iii) which algorithms, and (iv) evaluation metrics are used in the hierarchical classification of data streams and (v) what are the research gaps in the hierarchical classification of data streams.

The protocol used to perform the SLR, as well as expanded results and discussions, are described in detail in the original paper of the SLR in (TIEPPO et al., 2021). Yet, the main results, findings, and discussions about related works and the state of the art are described below.

Table 2.1 details the related work of the hierarchical data stream classification area resulting from the SLR.

Table 2.1 - Related work of the hierarchical data stream classification area.

Id	Year	Title	Reference
1	2008	Improving the performance of an incremental algorithm driven by error margins	(DEL CAMPO-ÁVILAA et al., 2008)
2	2009	An Adaptive Hierarchical Model Based on Fusion of Ontology and Context	(GU et al., 2009)
3	2009	Hierarchical Classification of Business Information on the Web Using Incremental Learning	(WANG; GONG; GUO, 2009)
4	2009	Problem classification method to enhance the ITIL incident and problem	(SONG; SAILER; SHAIKH, 2009)
5	2010	Hierarchical classification of dynamically varying radar pulse repetition interval modulation patterns	(KAUPPI; MARTIKAINEN; RUOTSALAINEN, 2010)
6	2010	Integrating support vector machine and genetic algorithm to implement dynamic wafer quality prediction system	(CHOU; WU; CHEN, 2010)
7	2010	On-line evolving image classifiers and their application to surface inspection	(LUGHOFER, 2010)
8	2010	Soft Concept Hierarchies to Summarise Data Streams and Highlight Anomalous Changes	(MARTIN; SHEN; MAJIDIAN, 2010)
9	2011	Pitch-density-based features and an SVM binary tree approach for multi-class audio classification in broadcast news	(XIE et al., 2011)

10	2012	A method for classifying packets into network flows based on GHSOM	(SHI et al., 2012)
11	2012	Adaptive object recognition model using incremental feature representation and hierarchical classification	(JEONG; LEE, 2012)
12	2012	Flashes in a star stream: Automated classification of astronomical transient events	(DJORGOVSKI et al., 2012)
13	2012	Hierarchical online problem classification for IT support services	(SONG; SAILER; SHAIKH, 2011)
14	2012	Investigation of broadcast-audio semantic analysis scenarios employing radio-programme-adaptive pattern classification	(KOTSAKIS; KALLIRIS; DIMOULAS, 2012)
15	2014	Classifying XML data of semantic sensor networks	(LA et al., 2014)
16	2014	Dealing with temporal variation in patent categorization	(D'HONDT et al., 2014)
17	2014	Hierarchical multi-label classification of social text streams	(LIANG; REN; DE RIJKE, 2014)
18	2014	Identifying seekers and suppliers in social media communities to support crisis coordination	(PUROHIT et al., 2014)
19	2015	Anatomical-plane-based representation for human-human interactions analysis	(ALAZRAI; MOWAFI; GEORGE LEE, 2015)
20	2015	Automatic identification of oculomotor behavior using pattern recognition techniques	(KORDA et al., 2015)
21	2015	Cost-sensitive learning of hierarchical tree classifiers for large-scale image classification and novel category detection	(FAN et al., 2015)
22	2015	Interactive open-ended learning for 3d object recognition: An approach and experiments	(KASAEI et al., 2015)
23	2015	Trending sentiment-topic detection on twitter	(PENG et al., 2015)
24	2016	Adaptive learning process for the evolution of ontology-described classification model in big data context	(PEIXOTO; CRUZ; SILVA, 2016)
25	2016	Automated species counting using a hierarchical classification approach with Haar cascades and multi-descriptor random forests	(CHAVEZ et al., 2016)
26	2016	Enhancing normal-abnormal classification accuracy in colonoscopy videos via temporal consistency	(PUERTO-SOUZA et al., 2015)
27	2016	Human continuous activity recognition based on energy-efficient schemes considering cloud security technology	(CHEN et al., 2016)
28	2016	On improving performance of surface inspection systems by online active learning and flexible classifier updates	(WEIGL et al., 2016)
29	2016	Phoneme sequence recognition via DTW-based classification	(HAMOONI; MUEEN; NEEL, 2016)
30	2017	A hierarchical approach towards activity recognition	(ANDEREZ et al., 2017)
31	2017	SW-SGD: the sliding window stochastic gradient descent algorithm	(CHAKROUN; HABER; ASHBY, 2017)
32	2018	Adapting Hierarchical Multiclass Classification to changes in the target concept	(SILVA-PALACIOS; FERRI; RAMIREZ-QUINTANA, 2018)
33	2018	Affect recognition from facial movements and body gestures by hierarchical deep spatio-temporal features and fusion strategy	(SUN et al., 2018)
34	2018	Contextual activity based Healthcare Internet of Things, Services, and People (HIoTSP): An architectural framework for healthcare monitoring using wearable sensors	(KHOWAJA et al., 2018)
35	2018	Fine-grained entity type classification with adaptive context	(LIU et al., 2018)
36	2018	GCHAR: An efficient Group-based Context-aware human activity recognition on smartphone	(CAO et al., 2018)
37	2018	Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams	(PESARANGHADER; VIKTOR; PAQUET, 2018)
38	2018	Using deep features for video scene detection and annotation	(PROTASOV et al., 2018)

39	2019	A crowdsource-based sensing system for monitoring fine-grained air quality in urban environments	(HUANG et al., 2018)
40	2019	Diagnosis and Monitoring of Alzheimer's Patients Using Classical and Deep Learning Techniques	(RAZA et al., 2019)
41	2019	Learning skeleton representations for human action recognition	(SAGGESE et al., 2019)
42	2019	Towards Hierarchical Classification of Data Streams	(PARMEZAN; SOUZA; BATISTA, 2018)

As detailed in Table 2.1, the SLR resulted in 42 studies related to the hierarchical data stream classification area. These studies were published in the 2008-2019 period, with an increase in the number of studies published each year.

As aforementioned, the SLR investigated hierarchical data stream classification under five main research questions. The main findings related to them are summarized as follows:

i. The main problems handled by related work include the classification of images, human activities, texts, and audio;
ii. The datasets used in the study experiments are mostly created for the study or comprise synthetic data, revealing that there is not yet a baseline for the creation of testbeds.
iii. The used algorithms are mainly well-known techniques, such as Support Vector Machines, k-Nearest Neighbors or Neural Networks, or adaptations of those.
iv. The used evaluation metrics are mainly well-known techniques, such as accuracy, precision, recall, or F-Measure, with the addition of the Loss metric and other metrics concerned with computational resources.
v. Research gaps in the hierarchical data stream classification context reported by the authors are mainly related to dynamism (ever-changing environment), data complexity (including large-scale data and non-stationary data over time), and computational resources (such as bounded memory and hardware limitations in real-world applications).

2.3.2 Considerations concerning the state of the art

The previously described related works were categorized according to the main properties of hierarchical and data stream classification areas (see Section 2.1) to highlight their adherence to the hierarchical data stream classification.

Table 2.2 details the adherence of the selected studies to the hierarchical data stream classification properties. Columns 2-5 show properties inherited from the hierarchical classification area and columns 6-11 from the data stream classification area. Time window values “S” and “L” stand for sliding and landmark windows; data stream handling strategies “I” and “A” stand for incremental and adaptive approaches. Unfilled cells (-) represent topics not comprehensively addressed by that study.

One can note that there are few cases in the related work in which most of the properties from the hierarchical data stream classification area are fulfilled together. Some works present full hierarchical classifiers but only partially address the data streams aspect when considering data from streams as input while ignoring some constraints brought together by this data (like concept drifts) or only producing theoretical essays on their methods. On the other hand, some data stream classifiers even work with hierarchical data streams but perform their classification process by ignoring the hierarchy and obtaining a flat representation of classes.

As an exception, the study presented in (PARMEZAN; SOUZA; BATISTA, 2018) (Id 42) covered both areas by proposing a method at the intersection of the areas from the beginning and also making available three datasets of hierarchical data stream classification. This method is based on the k-Nearest Neighbors (kNN) technique, represents the data hierarchically, and classifies new data using a top-down strategy within the hierarchy. The proposed algorithm performs a single path (Δ) and non-mandatory leaf-node (Ξ) predictions, represents the hierarchy in a tree (Ω), and uses a local classifier per parent node approach to handle the hierarchy (Θ). In addition, the algorithm uses a sliding window by storing a memory buffer on each class node with the most recent examples of the data stream and discarding older instances when the buffer is full. After a predetermined number of initial instances used for training (burnout window), the method follows the prequential assessing protocol, processing the data stream on an instance basis and discarding each instance after analyzing it.

This method successfully merged both areas (data stream classification and hierarchical classification) being able to classify hierarchical data streams with bounded computational resources and responsive to possible concept drifts. However, this approach still has some limitations in the context of hierarchical data stream classification area since the computational cost for classifying new instances is dependent on the number of instances that the model stores.

Table 2.2 - Adherence of the selected studies to the hierarchical data stream classification properties.

Id	Data Structure ϒ/Ω	Label cardinality Ψ/Δ	Label Depth Φ/Ξ	Hierarchy handling Θ	Single-pass	Readiness	Bounded Memory	Concept drift	Time Window	Data stream handling
1	-	-	-	-	Yes	Yes	No	No	-	I
2	-	-	-	-	-	-	-	-	-	-
3	Tree	MPL/MPP	PD/NMLNP	LCN	No	No	No	No	-	I
4	Tree	SPL/SPP	FD/MLNP	LCN	No	No	No	No	-	I
5	Tree	SPL/SPP	FD/MLNP	LCPN/LCN	No	No	No	No	S	-
6	-	-	-	-	No	Yes	No	Yes	-	I
7	-	-	-	-	Yes	Yes	Yes	Yes	S	I
8	DAG/Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
9	Tree	MPL/MPP	FD/MLNP	LCPN	No	No	No	No	S	-
10	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	S	-
11	-	-	-	-	No	No	No	No	-	I
12	Tree	SPL/SPP	FD /MLNP	LCN	Yes	No	No	No	-	I
13	Tree	SPL/SPP	FD /MLNP	LCN	No	No	No	No	-	I
14	Tree	SPL/SPP	FD/MLNP	LCN	No	No	No	No	-	-
15	Tree	SPL/SPP	FD/MLNP	LCN	No	No	No	No	-	-
16	Tree	-	-	-	No	No	No	Yes	-	-
17	Tree	MPL/MPP	PD/NMLNP	-	No	No	No	Yes	L	I
18	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
19	-	-	-	-	No	No	No	No	S	-
20	Tree	SPL/SPP	FD/MLNP	LCPN	No	No	No	No	S	-
21	Tree	SPL/SPP	PD/NMLNP	LCP	No	No	No	No	S	I
22	-	-	-	-	No	No	No	No	-	I
23	-	-	-	-	No	Yes	No	No	-	-
24	DAG/Tree	MPL/MPP	-	-	No	No	No	Yes	-	A
25	DAG	SPL/SPP	FD/MLNP	LCPN	No	No	No	No	-	-
26	-	-	-	-	No	No	No	No	S	-
27	DAG/Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
28	-	-	-	-	Yes	No	No	No	-	A
29	Tree	SPL/SPP	FD/MLNP	LCPN	No	No	No	No	S	-
30	Tree	SPL/SPP	FD/MLNP	LCPN	No	No	No	No	S	-
31	Tree	MPL/MPP	FD/MLNP	-	No	No	No	No	S	-
32	Tree	MPL/MPP	FD/MLNP	LCN	No	No	No	No	-	I
33	DAG/Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
34	DAG/Tree	SPL/SPP	FD/MLNP	LCPN	No	No	No	No	S	-
35	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	S	-
36	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
37	-	-	-	-	Yes	Yes	Yes	Yes	S	A
38	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	S	-
39	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
40	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
41	Tree	SPL/SPP	FD/MLNP	-	No	No	No	No	-	-
42	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	S	A

Consequently, the available computational resources still limit the method in representing information since larger buffers could compromise its readiness and even make it unfeasible because the processing time will eventually surpass the ratio in which incoming instances become available. Then, the method will need to discard those instances without processing them, and it will not adapt itself swiftly enough to handle concept drifts since information about changes in the data would be lost.

Furthermore, one of the main insights extracted from this review is the difference between the number of candidate studies (primarily retrieved) and the selected studies (the ones listed in Table 2.1 obtained after filtering by inclusion and exclusion criteria).

Direct exclusion criteria (such as the language or gray literature) filtered a considerable number of studies, but here we highlight the difference between the number of candidate studies after exclusion criteria and the number of selected studies. In other words, the lack of adherence between the previously returned studies and the inclusion criteria denotes a specific property of hierarchical data stream classification: it is a roughly unexplored gap that lies between two well-known areas.

In this sense, there is room for methods that fulfill the requirements of the hierarchical data stream classification area and can still work with constant computational resources, avoiding the use of resources linearly scalable by the number of instances provided by the data stream.

2.4 A NOTE ON THE RESEARCH GAP

As discussed in the introduction, the data stream classification area has been revisiting the classification subtasks resulting in new research connections, initially with the binary classification of data streams, and then handling problems hierarchically structured. This evolution seems reasonable considering the improvement in obtaining, collecting, and storing large-scale data.

For example, the authors in (SOUZA et al., 2020) proposed a new version of one of the datasets initially introduced in (PARMEZAN; SOUZA; BATISTA, 2018) (a aforementioned related work), comprehending more instances and with a formal definition of concept drifts within the data stream.

This dataset is composed of instances representing flying insects captured by electronic traps built with optical sensors, obtaining the wing-beat frequency of the insects. The purpose of the dataset is to build classifiers to recognize and capture particular species of interest that may represent disease vectors or cause agricultural damage (SOUZA et al., 2020).

The central dataset generated by the authors contains all instances collected in natural order (randomly) along the data stream. The authors also proposed different filters and arrangements of the dataset based on ambient features, such as the temperature or the luminosity at a given time in which an instance was collected. By that, the authors controlled the occurrence and representation of concept drifts in the dataset. As result, the authors proposed 11 datasets, including the one containing the complete data, plus another ten sets representing balanced and unbalanced versions of five distinct patterns simulating concept drifts.

Although the dataset was introduced by the authors as a flat dataset, the data stream maintains the hierarchical trait as initially proposed in (PARMEZAN; SOUZA; BATISTA, 2018). In other words, each instance component of the data stream represents species of disease vector mosquitoes, which are naturally organized in an entomological taxonomy.

From the perspective of the classification task, this dataset can be handled in a few different ways. Note that, despite being a hierarchical data stream set, learning models from other classification subtasks can be applied if specific data traits are not taken into account.

For instance, a hierarchical classifier can be applied to a hierarchical data stream handling it as a batch dataset, or a flat data stream classifier can be applied disregarding data hierarchy. However, both strategies have drawbacks that strengthen the rationale behind the investigation of the hierarchical data stream classification area proposed in this thesis. These drawbacks are discussed below.

On one hand, regarding hierarchical classification, the main disadvantages of learning models are the premise of finite data available for a well-defined training step and the building of a stationary model. Data streams are potentially unbounded, and data are made available to the model continuously over time. Thus, several strategies used by hierarchical classifiers would fail in a data stream environment, such as storing all data or reprocessing a given instance. In addition, due to the ephemeral trait of the data, the stationary model could result in model degradation over time since it would use old information as a reference that may no longer describe the current data.

Note that the forced use of exclusively hierarchical classifiers in a hierarchical data stream is possible, for example, if considered in hypothetical circumstances in which data can be totally stored in memory. However, it may render a noticeable drawback to the classification results since the learning model would not use any update strategy to respond to possible changes in data distribution that affect predictions.

On the other hand, concerning data stream classification, learning models have the main drawback of not considering intrinsically class hierarchies present in hierarchical data streams, losing information that could be useful to the classification process.

Note that the forced use of flat data stream classifiers in a hierarchical data stream is even feasible by ignoring the hierarchy of classes, predicting a class in a flat and isolated way, and retrieving a hierarchical label path with a further analysis of an attached ontology.

However, all the information on the relationship between the classes is wasted mainly in two aspects. First, the model would not consider any similarity between classes, which may affect the prediction and the classification costs (since the classification costs are different at different spots in the hierarchy). Second, the trade-off between prediction reliability and usefulness is not an option since the classifiers do not have enough information to walk through the hierarchy and make decisions on discriminating generic classes (where there are more examples and thus more confidence but less usefulness) or specifying the prediction to deeper hierarchy classes (where there are fewer examples and thus less confidence but more usefulness) (FREITAS; CARVALHO, 2007).

3 PROPOSED METHODS

This section describes the learning models proposed in this thesis. The first part provides an overview of the methods, summarizing their main characteristics and classification approaches. The remaining part (Sections 3.2, 3.3, and 3.4) describes the proposed methods based on nearest neighbors, clustering, and gaussian probabilities, including their descriptions and algorithms.

3.1 OVERVIEW

As previously reported in the systematic literature review that investigated research related to hierarchical data stream classification (see Section 2.3), the task was totally addressed first in (PARMEZAN; SOUZA; BATISTA, 2018), where the authors proposed a k-Nearest Neighbors (kNN) classifier fitted to work with hierarchically structured data streams.

The method, hereafter referred to as Local kNN-hDS, stores n instances on buffers assigned to each class node in the hierarchy. When the model receives a new instance, it applies a sliding window strategy to forget older data as a first-in/first-out strategy is followed. Additionally, the method uses local classifiers per parent node, building sub-datasets of instances at each step to perform the distance computations required by the kNN method.

Despite being an effective hierarchical data stream classifier, the Local kNN-hDS method has a main drawback regarding hierarchical data stream classification as it stores raw instances on buffers on the learning model and may put in jeopardy time and memory usage constraints required by streaming scenarios.

In this sense, this thesis introduces novel hierarchical data stream methods that use different strategies of data summarization, making them more suitable to the hierarchical data stream classification task.

First, in Section 3.2, it is introduced the Global kNN-hDS method, an adaptive global k-Nearest Neighbors method for hierarchical data stream classification.

The Global kNN-hDS method can be understood as a natural first response to the aforementioned related work Local kNN-hDS, as it uses a global strategy to reduce the dependence of Local kNN-hDS on distance computations.

Likewise, Section 3.3 brings forward methods that use summarization techniques, i.e., incremental centroids (STEINBACH; ERTÖZ; KUMAR, 2004), and cluster feature vectors (CFs) (ZHANG; RAMAKRISHNAN; LIVNY, 1996), as part of a hierarchical data stream classification process, to reduce the dependence on distance computations even more, and also being able to work with potentially unbounded hierarchical data streams with constant memory and time.

Specifically, the section describes two pairs of methods: Local and Global kNC-hDS, and Local and Global Dribble-hDS. Both global methods use the same global strategy used in the Global kNN-hDS method to perform fewer distance computations. Regarding summarization strategies, Local and Global kNC-hDS rely on centroid summarization, while Local and Global Dribble-hDS on cluster feature vectors.

Finally, Section 3.4 introduces the GNB-hDS method, an adaptive Gaussian Naive Bayes that relies on the well-known Gaussian Naive Bayes classification technique (BISHOP; NASRABADI, 2006) and uses statistical descriptors as a summarization strategy instead of storing raw instances.

In addition to the GNB-hDS method, Section 3.4 also describes the incremental Yeo-Johnson Power Transformation, an incremental adaptation of the well-known Yeo-Johnson Power Transformation (YEO; JOHNSON, 2000) that does not require a full view of the data, it is performed instance by instance, and it is adaptive along the data stream. This proposed transformation can be used attached to a classifier as a data pre-processing step to reduce the skewness of the data and improve the prediction results. Therefore, the GNB-hDS method was also experimented with different variations regarding the use of the incremental Yeo-Johnson Power Transformation.

Table 3.1 summarizes all methods proposed in this thesis, briefing their summarization and prediction strategies. Each method is described in detail in the following sections.

Table 3.2 specifies the proposed methods concerning the hierarchical data stream classification properties (using the same categorization previously used in Section 2.3.2, Table 2.2). In the last two columns, “S” stands for the sliding window strategy, and “I” and “A” stand for incremental and adaptive data stream handling approaches.

Observe that the proposed methods fulfill all hierarchical data stream classification constraints.

Also, note that some specifications represent the current method in this thesis but do not necessarily imply a method limitation in that category. For instance, some methods that handle the data stream adaptively can also perform it incrementally only with parameter adjustments. The same idea applies to label cardinality or the window strategy.

Table 3.1 - Overview of the proposed methods and their summarization and prediction strategies.

Proposed Method	Summarization strategy	Prediction strategy
Global kNN-hDS	Buffers of instances	Nearest neighbors and label path analysis
Local kNC-hDS	Centroids	Nearest neighbors
Global kNC-hDS	Centroids	Nearest neighbors and label path analysis
Local Dribble-hDS	Cluster Feature Vectors	Nearest neighbors
Global Dribble-hDS	Cluster Feature Vectors	Nearest neighbors and label path analysis
GNB-hDS	Statistical descriptors	Bayesian probabilities

Table 3.2 - Specification of the proposed methods concerning the hierarchical data stream classification properties.

Proposed Method	Data Structure Υ/Ω	Label cardinality Ψ/Δ	Label Depth Φ/Ξ	Hierarchy handling Θ	Single-pass	Readiness	Bounded Memory	Concept drift	Time Window	Data stream handling
Global kNN-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	S	A
Local kNC-hDS	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	S	A
Global kNC-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	S	A
Local Dribble-hDS	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	S	A
Global Dribble-hDS	Tree	SPL/SPP	FD/MLNP	GC	Yes	Yes	Yes	Yes	S	A
GNB-hDS	Tree	SPL/SPP	FD/MLNP	LCPN	Yes	Yes	Yes	Yes	S	I/A

3.2 GLOBAL K-NEAREST NEIGHBORS FOR HIERARCHICAL DATA STREAMS (GLOBAL KNN-HDS)

The Global k-Nearest Neighbors for Hierarchical Data Streams (Global kNN-hDS) is an adaptive method for the hierarchical data stream classification based on the traditional k-Nearest Neighbors (kNN) technique (AHA; KIBLER; ALBERT, 1991).

The method performs single path predictions (SPP) and mandatory leaf-node predictions (MLNP) using a global classification (GC) approach. Global kNN-hDS processes new instances with less computational effort when compared to local approaches.

Here, it is essential to highlight that a global approach is usually smaller in computational resources than a local approach since it deploys a single model in contrast to multiple models in the local approaches (SILLA; FREITAS, 2011). Figure 3.1 illustrates this difference in two kNN methods using a local classifier approach (LCPN, in this case) and a global classifier (GC) approach, where R stands for the root node of the tree.

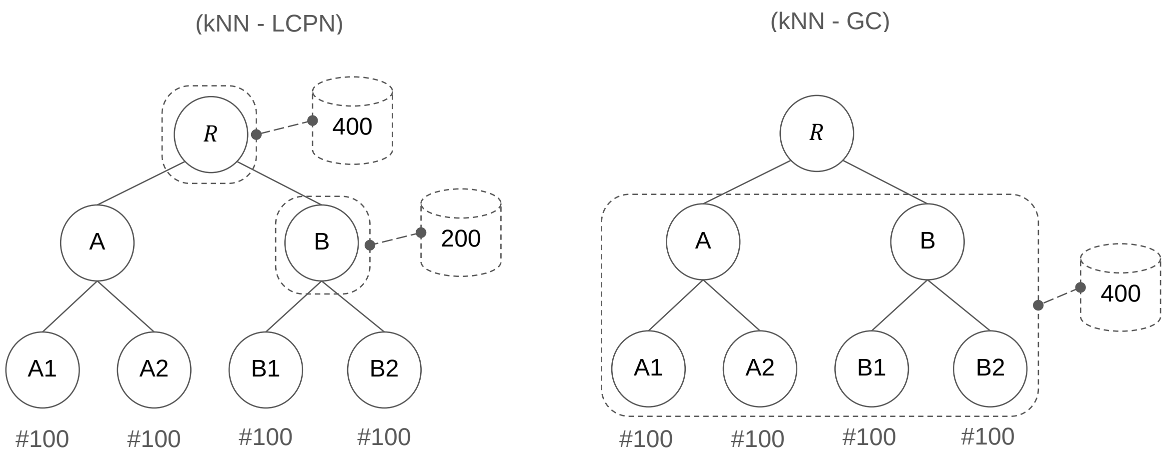


Figure 3.1 - Illustrative comparison between LCPN and GC hierarchical classification approaches.

While kNN-LCPN compares instances at each level in the hierarchy, the kNN-GC performs only one comparison for the whole tree. Note that a kNN using a local approach (Figure 3.1, kNN - LCPN) comprises one classifier at each parent node in the hierarchy to choose between its child nodes using sub-datasets at each step. The kNN-LCPN placed in the root node performs one comparison using 400 instances to predict A or B. After that, another kNN-LCPN placed in the B node (the predicted one in the last step) repeats the process with its child nodes, performing another comparison with another 200 instances.

In contrast, a global approach kNN (Figure 3.1, kNN - GC) performs only one comparison for the whole tree using the 400 instances available. It is important to note that this difference is dependent on the depth of the tree and tends to be even larger in deeper hierarchies.

The global approach is achieved by selecting the kNN of an instance and comparing the labels for each level, picking the label paths with the most frequent label for that level. Next, the process is repeated for the next level of the label path until it reaches a leaf node.

Figure 3.2 illustrates this process with illustrative label paths and using $k = 5$ on the kNN. The most frequent label in each level is chosen, and the other label paths are discarded in the analysis of the next level. Ties are decided using the lowest distance between neighbors. The process is repeated until it reaches a leaf node.

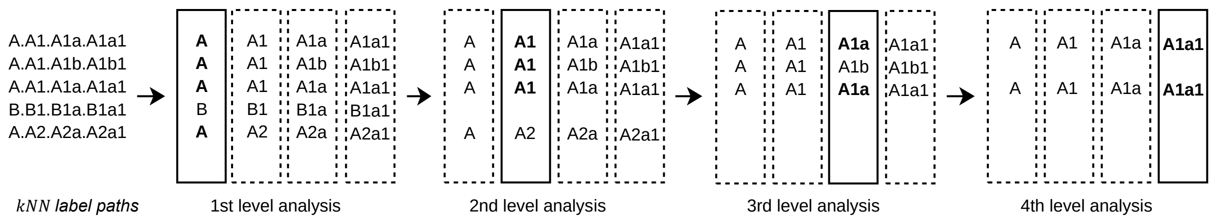


Figure 3.2 - Global approach via label path analysis using label frequency per hierarchy level.

The proposed method is also adaptive since it uses a sliding window as a mechanism to forget older data. The method implements the window using a memory buffer n at each leaf node in the tree. Thus, the method inherently responds to concept drifts.

Algorithm 3.1 shows the pseudocode for the proposed adaptive global k-Nearest Neighbors method for hierarchical data stream classification.

The algorithm receives a hierarchical data stream hDS providing instances $(\vec{x}_t, \vec{y}_t)$ over time t, and the above described k and n parameters, representing the number of nearest neighbors and the upper bound of buffers on each node.

The hierarchy representation step is performed in line 1. The loop started on line 2 simulates the prequential loop.

From lines 3 to 5, the method obtains all the instances temporarily stored in all the descendant nodes of the root node. Next, in lines 6-8, it calculates the Euclidean distance between the new instance and the data stored at the tree (the possible nearest neighbors obtained in the previous step). Thus, the k-nearest neighbors and their labels are obtained by sorting the possible nearest neighbors by the Euclidean distance (line 9).

```
Algorithm
Global kNN-hDS - Adaptive global k-Nearest Neighbors for hierarchical data stream
classification
Input
  $hDS$:  a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time $t$
    $k$:  number of nearest neighbors
    $n$:  maximum number of instances to be stored in each node
Output
  $\hat{\vec{y}}_t$:  a predicted label path for the input instance
--------------------------------------------------------------------------------
 1  $Tree \leftarrow classTaxonomy(hDS)$;
 2  foreach ($\vec{x}_t \in$ hDS) do
 3    foreach ($childNode \in Tree.root.descendants$) do
 4      $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
 5    end
 6    foreach ($target \in targets$) do
 7      $target \leftarrow target \cup \{euclideanDistance(\vec{x}_t, target.data)\}$;
 8    end
 9    $targets \leftarrow (targets)_{1..k}$;
10    $\hat{\vec{y}}_t \leftarrow mostFrequentInLevel(targets)$;
11    $correctNode \leftarrow Tree.\vec{y}_t$;
12    $correctNode.data \leftarrow correctNode.data \cup \{\vec{x}_t\}$;
13    if ($|correctNode.data| > n$) then
14      $correctNode.data \leftarrow correctNode.data \setminus \{(correctNode.data)_1\}$;
15    end
16  end
```

Algorithm 3.1 - Global kNN-hDS: Adaptive global k-Nearest Neighbors for hierarchical data stream classification.

After that (line 10), the model can predict the label path by choosing the label returned by the function $mostFrequentInLevel()$, which implements the strategy previously described in Figure 3.2.

Next, the method completes the prequential step by incorporating the new data into the original dataset using its correct label (lines 11 and 12). In lines 13-15, the model tests whether the number of instances in each node exceeds the stipulated upper bound n. If so, it applies a sliding window strategy by forgetting the oldest instance of that node, performing the proposed adaptive learning, and assuring that the method can work with a constrained and constant memory amount.

3.3 K-NEAREST CENTROIDS FOR HIERARCHICAL DATA STREAMS (KNC-HDS) AND DRIBBLE FOR HIERARCHICAL DATA STREAMS (DRIBBLE-HDS)

In this section, two methods are proposed: k-Nearest Centroids for Hierarchical Data Streams (kNC-hDS) and Dribble for Hierarchical Data Streams (Dribble-hDS), each with local and global variants.

These methods share the same trait of using summarization strategies to represent data using clustering techniques.

Clustering is an unsupervised learning task that aims to find groups of instances in such a way that instances inside a group are similar to each other but distinguishable from those belonging to the other groups. In other words, given a set of instances, a clustering technique divides the instances into groups to minimize the intragroup and maximize the intergroup distance (STEINBACH; ERTÖZ; KUMAR, 2004).

The clustering task helps to understand the data – finding patterns of behavior or similarity between instances – or summarizing them, reducing their size by choosing a sample that is the pattern or the average of the group (STEINBACH; ERTÖZ; KUMAR, 2004).

The instance-based learning process used in a traditional kNN (in the same way as addressed by related work Local kNN-hDS and the proposed Global kNN-hDS) presents a drawback when applied in the data stream context since it depends on the instance comparison, which may not be feasible when considering the entire data stream (NGUYEN; WOON; NG, 2015; STEINBACH; ERTÖZ; KUMAR, 2004).

In this sense, kNC-hDS and Dribble-hDS use clustering techniques in order to improve data representation and allow a faster comparison of a large volume of data. Specifically, both local and global kNC-hDS use incremental centroids (STEINBACH; ERTÖZ; KUMAR, 2004), while both local and global Dribble-hDS use cluster feature vectors (CFs) (ZHANG; RAMAKRISHNAN; LIVNY, 1996) to represent data.

Next, this section describes in detail both Local kNC-hDS and Local Dribble-hDS methods. Afterward, both global variants are specified.

First, both methods organize class labels in a hierarchically structured class taxonomy using a tree data structure, with the paths from the root node to the leaf nodes representing label paths (classes) of instances.

kNC-hDS and Dribble-hDS follow different data representations compared to traditional kNN-based methods.

While in the kNN instance subsets are buffered with their class labels, kNC-hDS and Dribble-hDS use summarization strategies and discard the instances, thus resulting in smaller memory consumption and fewer distance computations.

kNC-hDS summarizes data using centroids, consequently resulting in a smaller number of distance computations when handling larger data volumes. A centroid is defined as a mean of the instances that are clustered together according to some criteria (STEINBACH; ERTÖZ; KUMAR, 2004).

The incremental centroids are built by storing the incremental mean of instance attributes. The incremental mean μ_t with the arrival of t values is computed as depicted in Equation (4), where $\bar{x}_{t-1}$ represents the current average, t is the number of instances observed thus far, and x_t is the arriving value to be incorporated.

$$\mu_t = \frac{\bar{x}_{t-1}(t-1) + x_t}{t} \tag{4}$$

Comparably, the Dribble-hDS method also summarizes instances but it uses Cluster Feature vectors (CFs) (ZHANG; RAMAKRISHNAN; LIVNY, 1996).

CFs enable summarizing data in hyperspherical regions and have been used in both data stream clustering and classification techniques (AGGARWAL et al., 2006).

A CF is a triplet in the $CF = (N, LS, SS)$ format, where N is the number of instances of the cluster summary, and LS and SS are the linear and square sum of the instances, respectively.

Thus, LS and SS are N-dimensional vectors, such that the dimensions match the original features available in instances (ZHANG; RAMAKRISHNAN; LIVNY, 1996).

From these components, the summary centroid (mean, μ) and its radius (r) are computed according to Equations (5) and (6) respectively, where d is the number of features available.

$$\mu(CF_i) = \frac{LS_i}{N_i} \tag{5}$$

$$r(CF_i) = \frac{1}{d}\sum_{i=1}^{d} \sqrt{\frac{N_i(SS_i) - 2(LS_i^2) + N_i(LS_i)}{N_i^2}} \tag{6}$$

In addition to their potential to summarize data, CFs also have an additive property, i.e., two feature vectors CF_i and CF_j can be merged by summing their components according to Equation (7) (ZHANG; RAMAKRISHNAN; LIVNY, 1996):

$$CF_k = CF_i + CF_j = \left(N_i + N_j, LS_i + LS_j, SS_i + SS_j\right) \tag{7}$$

In both kNC-hDS and Dribble-hDS methods, instances are incorporated into the model by composing a set of n incremental centroids (kNC-hDS) or CFs (Dribble-hDS) in the respective class node, where n is a user-defined hyper-parameter.

To retrieve summary descriptions and build sub-datasets for the prediction step, it is performed a top-down traverse in the hierarchy obtaining all centroids/CFs stored at the leaf nodes of the tree using the siblings' policy to consider positive instances from the target node and its descendants (SILLA; FREITAS, 2011).

The retrieval is straightforward in kNC-hDS since the model stores centroids representing a mean instance of the data stream. Meanwhile, in Dribble-hDS, we need an additional step to calculate the mean of the CFs (see Equation (5)). Additionally, to avoid noise incorporation due to the intrinsic characteristic of small CFs, Dribble-hDS implements an outlier control by retrieving only reasonably populated CFs (user-defined, by default, one-third of CFs size).

At any moment, the model can perform a prediction by comparing an unseen instance to the centroids/CFs stored at the nodes of the tree. The class prediction is performed by calculating the Euclidean distance between a new instance and the centroids/mean of CFs represented in the hierarchy nodes, returning the most frequent label between the selected k-nearest neighbors.

Both methods perform single path and mandatory leaf-node prediction, using a local classifier per node (LCPN) approach. Thus, in the prediction step, one multi-class classifier is applied per class to predict between its child nodes. The resulting predicted label is appended to the final label path, representing the full hierarchical label path predicted to a given instance.

After the prediction step, the methods update their summary descriptions. kNC-hDS and Dribble-hDS differ in the way the data is summarized. However, both methods work with a limited size m on centroids/CFs, where m is a user-defined hyper-parameter.

On kNC-hDS, a new (training) instance is incorporated into the stored centroid by incrementing its average (see Equation (4)). If the centroid is already full (i.e., if the number of summarized instances equals m), a new centroid is instantiated to the corresponding class node. As a consequence, if a node reaches the stipulated maximum number n of centroids, a forgetting strategy is performed by applying a sliding window and discarding the oldest centroid.

On Dribble-hDS, when a new (training) instance is received, the method checks if the instance is encompassed by any of the hyperspheres represented in the correct class of the instance (i.e., whether the instance is between the CF mean and its radius or not). If so, it adds the new instance to the respective CF. When the number of summarized instances surpasses m, the method performs a forgetting strategy by subtracting from the CF a statistical description (one mean instance) representing the oldest instance. Otherwise, if the instance is not encompassed by any of the hyperspheres, it starts a new hypersphere (at that moment, yet a single point). If the maximum number (n) of hyperspheres (or points) in a class is reached, the two closest hyperspheres are merged using the additive property of the CFs.

Algorithm 3.2 shows the pseudocode for the proposed Local k-Nearest Centroids for hierarchical data stream classification. Likewise, Algorithm 3.3 shows the pseudocode for proposed Local Dribble for hierarchical data stream classification.

Both algorithms perform the same steps to handle the data stream and also in the prediction step. Thus, both algorithms receive a hierarchical data stream hDS providing instances $(\vec{x}_t, \vec{y}_t)$ over time t and the aforementioned hyper-parameters k, n and m, and output a predicted label path for each incoming instance.

The representation of the class hierarchy is performed on the first line, and the loop started in line 2 applies the described LCPN approach to retrieve the summary descriptions stored at the tree nodes. The prediction step is also performed equally for both methods and it is depicted from lines 11 to 16 on the algorithm. Note that, until this very point, both methods differ only by the data structure retrieved specified in the code as the data from a node. On kNC-hDS, data refers to the stored centroids, while on Dribble-hDS it refers to the means of the CFs.

Both methods differ in summary updating and forgetting strategies, represented in the algorithm from line 19 on. The processes performed in this step on each method separately are described below.

On kNC-hDS (Algorithm 3.2), the method incorporates the new instance into the stored centroid by incrementing its average (line 20) or it creates a new centroid if the newest centroid is already full (line 23). After that, the method tests whether the number of centroids in the ground-truth node exceeds the stipulated maximum number n allowed (line 25). If so, it applies a sliding window strategy by forgetting the oldest centroid (line 26).

Algorithm
Local kNC-hDS – Local k-Nearest Centroids for hierarchical data stream classification

Input

hDS: a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time t
k: number of nearest centroids
n: maximum number of centroids
m: maximum number of instances to be summarized on a centroid

Output

$\widehat{\vec{y}_t}$: a predicted label path for the input instance

```
$Tree \leftarrow classTaxonomy(hDS)$;
foreach ($\vec{x}_t \in$ hDS) do
    $predictedNode \leftarrow Tree.root$;
    while $\neg(predictedNode.isLeaf)$ do
        foreach ($childNode \in predictedNode.children$) do
            $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
            foreach ($descendantNode \in predictedNode.children$) do
                $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
            end
        end
        foreach ($target \in targets$) do
            $target \leftarrow target \cup \{euclideanDistance(\vec{x}_t, target.data)\}$;
        end
        $targets \leftarrow (targets)_{1..k}$;
        $predictedNode \leftarrow argmax(targets)$;
        $\widehat{\vec{y}_t} \leftarrow \widehat{\vec{y}_t} \cup predictedNode.label$;
    end
    $correctNode \leftarrow Tree.\vec{y}_t$;
    if ($correctNode.newestCentroid.count < m$) then
        $correctNode.newestCentroid \leftarrow i\mu$;
    end
    else
        $correctNode.data \leftarrow correctNode.data \cup \{newCentroid(\vec{x}_t)\}$;
    end
    if ($|correctNode.data| > n$) then
        $correctNode.data \leftarrow correctNode.data \setminus \{(correctNode.data)_1\}$
    end
end
```

Algorithm 3.2 - Local kNC-hDS: Local k-Nearest Centroids for hierarchical data stream classification.

On Dribble-hDS (Algorithm 3.3), the method finds the nearest CF (line 19) to: update the CF using its additive property if the new CF is encompassed by the nearest CF (line 21), or create a new CF if it is not encompassed by the nearest CF (line 27). Then, the method checks if the number of instances represented in the CFs stored at the ground-truth node exceeds the stipulated maximum number m allowed (line 22). In such a case, a CF mean is subtracted from the CF to forget a representation of the oldest instance (line 23).

```
Algorithm
Local Dribble-hDS – Local Dribble for hierarchical data stream classification
Input
 $hDS$:  a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time $t$
   $k$:  number of nearest $CF$ means
   $n$:  maximum number of $CFs$
   $m$:  maximum number of instances to be summarized on a $CF$
Output
  $\hat{\vec{y}}_t$:  a predicted label path for the input instance
---------------------------------------------------------------------------------
 1  $Tree \leftarrow classTaxonomy(hDS)$;
 2  foreach ($\vec{x}_t \in$ hDS) do
 3    $predictedNode \leftarrow Tree.root$;
 4    while $\neg(predictedNode.isLeaf)$ do
 5      foreach ($childNode \in predictedNode.children$) do
 6        $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
 7        foreach ($descendantNode \in predictedNode.children$) do
 8          $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
 9        end
10      end
11      foreach ($target \in targets$) do
12        $target \leftarrow target \cup \{euclideanDistance(\vec{x}_t, target.data)\}$;
13      end
14      $targets \leftarrow (targets)_{1..k}$;
15      $predictedNode \leftarrow argmax(targets)$;
16      $\hat{\vec{y}}_t \leftarrow \hat{\vec{y}}_t \cup predictedNode.label$;
17    end
18    $correctNode \leftarrow Tree.\vec{y}_t$;
19    $nearestCF \leftarrow argmin(euclideanDistance(\vec{x}_t, correctNode.data))$;
20    if ($nearestCF.distance < nearestCF.radius$) then
21      $nearestCF \leftarrow nearestCF + newCF(\vec{x}_t)$;
22      if ($nearestCF.count > m$) then
23        $nearestCF \leftarrow nearestCF - nearestCF.mean$;
24      end
25    end
26    else
27      $correctNode.data \leftarrow correctNode.data \cup newCF(\vec{x}_t)$;
28      if ($|correctNode.data| > n$) then
29        $CF_1, CF_2 \leftarrow findClosestCFs(correctNode.data)$;
30        $CF_1 \leftarrow CF_1 + CF_2$;
31      end
32    end
33  end
```

Algorithm 3.3 - Local Dribble-hDS: Local Dribble for hierarchical data stream classification.

Finally, the method checks if the number of CFs stored at the ground-truth node exceeds the stipulated maximum number allowed (line 28). If so, it merges the two closest CFs to return to the maximum number n of CFs allowed on that node. To this end, a Euclidean distance calculation is performed between all CFs at the node.

As described at the beginning of this section, two global variants of kNC-hDS and Dribble-hDS methods are proposed next. The Global kNC-hDS and Global Dribble-hDS follow the same strategies to summarize data using incremental centroids and CFs. Also, both methods use a global strategy to handle the hierarchy in their learning process, and thus, process new instances with fewer comparisons when compared to local approaches.

Figure 3.3 illustrates the steps of the learning process shared by both methods. After building the hierarchy representation, each method receives instances $i^t = (\vec{x}_t, \vec{y}_t)$. A test/prediction phase starts by obtaining all data from nodes, comparing them to find nearest neighbors, and applying a global approach via label path analysis. At this point, the model can be requested to predict a class (label path) for $\vec{x}_t$. Next, the instance $\vec{x}_t$ and the corresponding label path $\vec{y}_t$ are used to update the model, and older data is discarded by a sliding window.

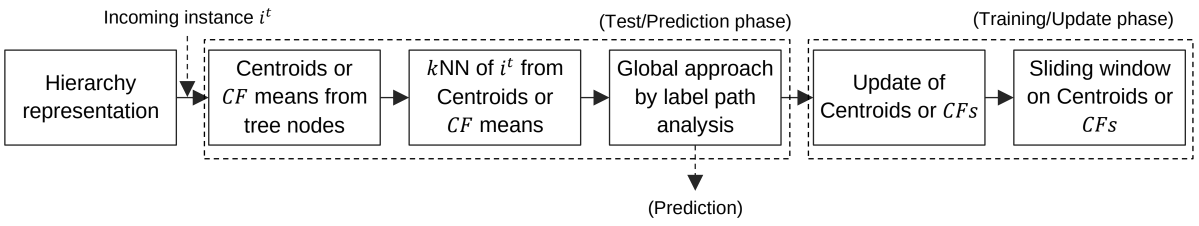

Incoming instance i^t
(Test/Prediction phase)
(Training/Update phase)
Hierarchy representation
Centroids or CF means from tree nodes
kNN of i^t from Centroids or CF means
Global approach by label path analysis
Update of Centroids or CFs
Sliding window on Centroids or CFs
(Prediction)

Figure 3.3 - General view of Global kNC-hDS and Global Dribble-hDS learning process.

Note that both learning models follow the same strategy as their local counterparts, except for the prediction strategy that relies on a global approach instead of the LCPN approach used by the local methods.

The global approach used by both methods is the same one used in the Global kNN-hDS method (see Section 3.2, Figure 3.2). Both methods analyze label paths retrieved from the kNN step, not comparing instances but the frequency of component labels of each level of the label paths. The most frequent label in each level is chosen, and the other label paths are discarded in the analysis of the next level. Ties are decided using the lowest distance between neighbors. The process is repeated until it reaches a leaf node.

The building of datasets for distance computations considers the entire hierarchy of the tree. Therefore, the resulting kNN can represent nodes at any hierarchy level. In contrast, in local approaches, the building of datasets is restricted to portions of the hierarchy (by node, by parent node, or by level) and needs to be performed several times until reaching the deeper levels of the hierarchy.

Algorithm 3.4 and Algorithm 3.5 depict the pseudocodes for the Global kNC-hDS and Global Dribble-hDS methods that encompass the steps described above.

Algorithm
Global kNC-hDS - Global k-Nearest Centroids for hierarchical data stream classification

Input
hDS: a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time t
k: number of nearest centroids
n: maximum number of centroids
m: maximum number of instances to be summarized on a centroid

Output
$\hat{\vec{y}}_t$: a predicted label path for the input instance

```
1   $Tree \leftarrow classTaxonomy(hDS)$;
2   foreach ($\vec{x}_t \in$ hDS) do
3       foreach ($childNode \in Tree.root.descendants$) do
4           $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
5       end
6       foreach ($target \in targets$) do
7           $target \leftarrow target \cup \{euclideanDistance(\vec{x}_t, target.data)\}$;
8       end
9       $targets \leftarrow (targets)_{1..k}$;
10      $\hat{\vec{y}}_t \leftarrow mostFrequentInLevel(targets)$;
11      $correctNode \leftarrow Tree.\vec{y}_t$;
12      if ($correctNode.newestCentroid.count < m$) then
13          $correctNode.newestCentroid \leftarrow i\mu$;
14      end
15      else
16          $correctNode.data \leftarrow correctNode.data \cup \{newCentroid(\vec{x}_t)\}$;
17      end
18      if ($|correctNode.data| > n$) then
19          $correctNode.data \leftarrow correctNode.data \setminus \{(correctNode.data)_1\}$
20      end
21  end
```

Algorithm 3.4 - Global kNC-hDS: Global k-Nearest Centroids for hierarchical data stream classification.

Both algorithms receive a hierarchical data stream hDS providing instances $(\vec{x}_t, \vec{y}_t)$ over time t and the aforementioned k, n and m parameters. The hierarchy representation step (line 1) and the test/prediction phase (lines 2-10) are similar between Global kNC-hDS and Global Dribble-hDS. The global approach by label path analysis is performed by a function that receives a set of label paths, recursively removes infrequent labels at each level of the label hierarchy and returns the most frequent label path of the initial set (line 10).

Next (from line 11 onwards), both methods update themselves and apply the sliding windows differently, equally to their local variants. The Global kNC-hDS method updates the centroids using the incremental mean (Equation (4)) (line 13) and checks the upper boundaries m and n. If m is reached, the method creates a new centroid instead of filling the newest one (lines 12-16). If n is surpassed, the method applies the sliding window strategy by discarding the oldest centroid (lines 18-20).

```
Algorithm
Global Dribble-hDS – Global Dribble for hierarchical data stream classification
Input
 $hDS$:  a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time $t$
   $k$:  number of nearest $CF$ means
   $n$:  maximum number of $CFs$
   $m$:  maximum number of instances to be summarized on a $CF$
Output
  $\hat{\vec{y}}_t$:  a predicted label path for the input instance
--------------------------------------------------------------------------------
 1  $Tree \leftarrow classTaxonomy(hDS)$;
 2  foreach $(\vec{x}_t \in$ hDS$)$ do
 3    foreach $(childNode \in Tree.root.descendants)$ do
 4      $targets \leftarrow targets \cup \{(childNode.label, childNode.data)\}$;
 5    end
 6    foreach $(target \in targets)$ do
 7      $target \leftarrow target \cup \{euclideanDistance(\vec{x}_t, target.data)\}$;
 8    end
 9    $targets \leftarrow (targets)_{1..k}$;
10    $\hat{\vec{y}}_t \leftarrow mostFrequentInLevel(targets)$;
11    $correctNode \leftarrow Tree.\vec{y}_t$;
12    $nearestCF \leftarrow argmin(euclideanDistance(\vec{x}_t, correctNode.data))$;
13    if $(nearestCF.distance < nearestCF.radius)$ then
14      $nearestCF \leftarrow nearestCF + newCF(\vec{x}_t)$;
15      if $(nearestCF.count > m)$ then
16        $nearestCF \leftarrow nearestCF - nearestCF.mean$;
17      end
18    end
19    else
20      $correctNode.data \leftarrow correctNode.data \cup newCF(\vec{x}_t)$;
21      if $(|correctNode.data| > n)$ then
22        $CF_1, CF_2 \leftarrow findClosestCFs(correctNode.data)$;
23        $CF_1 \leftarrow CF_1 + CF_2$;
24      end
25    end
26  end
```

Algorithm 3.5 - Global Dribble-hDS: Global Dribble for hierarchical data stream classification.

On Global Dribble-hDS, the method updates the CFs using the additive property or creating a new CF (lines 14 and 20). In line 14, the incoming instance is encompassed by the nearest CF, while in line 20, it is not. The method also checks the upper boundaries m and n. If m is surpassed, the method subtracts a mean representation of the hypersphere from the CF (lines 15-17). If n is surpassed, the two closest CFs are merged to match the number of CFs to n (lines 21-24).

3.4 GAUSSIAN NAIVE BAYES FOR HIERARCHICAL DATA STREAMS (GNB-HDS)

This section describes the Gaussian Naive Bayes for Hierarchical Data Streams (GNB-hDS), a method for the hierarchical data stream classification based on the well-known Naive Bayes technique (BISHOP; NASRABADI, 2006; FRIEDMAN; GEIGER; GOLDSZMIDT, 1997).

The main idea behind GNB-hDS is the use of dynamic data summaries, specifically the mean, the standard deviation, and the number of data instances, that allow the calculation of probabilities used in the Bayes' Theorem (BISHOP; NASRABADI, 2006; HAN; PEI; KAMBER, 2011).

These dynamic data summaries are attached to nodes of the hierarchy and are updated as new instances are gathered from the data stream.

The method implemented two key adaptations in the traditional Naive Bayes classifier to make it handle hierarchical data streams.

First, regarding the hierarchical data structure, the original algorithm was modified to consider not only one class, but all related classes of a given instance. As the hierarchical data structure represents a subsumption relation, any new instance provided from the data stream also belongs to its ancestors. Thus, the method traverses the hierarchy to update all data summaries of parent nodes recursively until the root node of the hierarchy.

Second, regarding the streaming input data, the algorithm must store incremental or adaptive statistical descriptors instead of the actual instances. Thus, the method computes the mean, the standard deviation, and the count of data instances assigned to each class incrementally or adaptively, discarding the instance after it is analyzed.

The GNB-hDS represents the class taxonomy in a tree structure using local classifiers at each parent node and assigns leaf node classes as the last class of one predicted label path $\vec{y}^t$ (mandatory leaf-node and single path prediction).

Figure 3.4 illustrates the process performed by GNB-hDS when handling the data stream incrementally. Circles represent classes, and dashed squares enclose classifiers. The method represents the class taxonomy in a tree structure, where R stands for the root node of the hierarchy and classes are related to each other.

When receiving an incoming instance for prediction, the method tackles the hierarchy using a local classifier per parent node (LCPN) approach, thus analyzing the current parent node and predicting between its child nodes by using probabilities obtained with the Bayes' Theorem. This process is repeated until a leaf node is reached.

Each node in the tree stores the count of instances (n), a d-dimensional incremental mean ($\bar{x}_n$), and a d-dimensional incremental standard deviation (σ_n) of the class represented (as shown in class 2). After the incoming instance processing, the statistical descriptors ($n, \bar{x}_n, \sigma_n$) are updated incrementally with the instance feature values on all the classes through the hierarchy regarding the ground-truth label path of that instance.

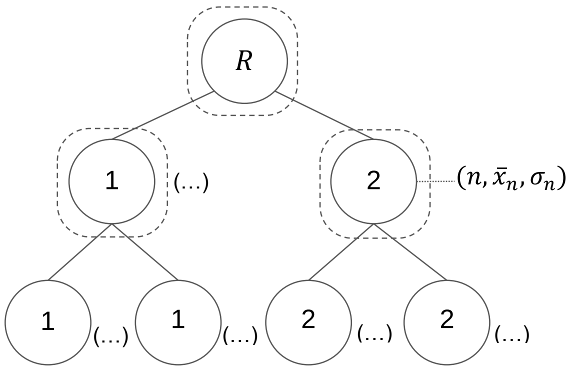


Figure 3.4 - Illustration of GNB-hDS method.

As before introduced, the instances are represented by data summaries comprising three statistical descriptors stored incrementally: (i) the count of class instances, (ii) the d-dimensional mean instance, and the d-dimensional standard deviation of the instances of a given class.

The number of instances assigned to a class C is stored in an attached counter. When an instance is retrieved from the stream, the C-th class counter is incremented alongside the counters of C's ancestors.

The incremental mean ($\bar{x}_n$) and the incremental standard deviation (σ_n) considering each attribute from a d-dimensional x_n instance are obtained, respectively, from Equations (8) and (9), where n stands for the number of instances observed so far assigned to C (CHAN; GOLUB; LEVEQUE, 1983; WEST, 1979).

Also, it is important to reinforce that the incremental mean and the incremental standard deviation are d-dimensional descriptors as the feature set and its values from the d-dimensional x_n instance. Note that Equations (8) and (9) support only continuous feature sets and the current mean and the standard deviation of the previously observed instances assigned to C are represented by $\bar{x}_{n-1}$ and σ_{n-1}.

$$\bar{x}_n = \frac{\left(\bar{x}_{n-1}(n-1)\right) + x_n}{n} \tag{8}$$

$$\sigma_n = \sqrt{\frac{(n-2)\sigma_{n-1}^2(n-1) + (\bar{x}_{n-1} - \bar{x}_n)^2 + (x_n - \bar{x}_n)^2}{n-1}} \tag{9}$$

The prediction of the class to be assigned to an incoming instance provided from the data stream is performed in three steps: (i) computation of the *a priori* probabilities based on the count of class instances, (ii) computation of likelihood probabilities based on the Bayes' Theorem for each attribute of the incoming instance, and (iii) calculation of the maximum value of the *a posteriori* probability from the product of the independent feature probabilities given a class C.

The calculation of the likelihood probability is described in Equation (10), where i represents a feature index and j a class index (BISHOP; NASRABADI, 2006).

$$p\left(x_i \middle| C_j\right) = \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} exp\left\{-\frac{1}{2}\left(\frac{x_i - \bar{x}_{i,j}}{\sigma_{i,j}}\right)^2\right\} \tag{10}$$

To perform the class assignment, the GNB-hDS obtains the class label with the maximum value of the *a posteriori* probability, as described in Equation (11), from the product of the independent feature probabilities given C (BISHOP; NASRABADI, 2006).

$$p\left(C_j \middle| x\right) \propto \left\{\prod_i p\left(x_i \middle| C_j\right)\right\} p\left(C_j\right) \tag{11}$$

Moreover, these three steps are performed from the top of the hierarchy data structure and repeated until a leaf node is reached, resulting in the union of the class assignments made from Equation (11) and representing the final label path assigned to the incoming instance.

Algorithm 3.6 shows the pseudocode for the proposed Gaussian Naive Bayes for Hierarchical Data Streams classification method.

```
Algorithm
GNB-hDS - Gaussian Naive Bayes for Hierarchical Data Streams
Input
 $hDS$:  a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time $t$
Output
   $\hat{\vec{y}}_t$:  a predicted label path for the input instance
-----------------------------------------------------------------------------------------
 1  $Tree \leftarrow classTaxonomy(hDS)$;
 2  foreach $(\vec{x}_t \in hDS)$ do
 3      $predictedNode \leftarrow Tree.root$;
 4      while $\neg(predictedNode.isLeaf)$ do
 5          foreach $(childNode \in predictedNode.children)$ do
 6              $priors \leftarrow priorProbability(childNode.Class)$;
 7          end
 8          $likelihood \leftarrow likelihoodProbability(\vec{x}_t, priors)$;
 9          $posterior \leftarrow posteriorProbability(likelihood, priors)$;
10          $predictedNode \leftarrow argmax(posterior)$;
11          $\hat{\vec{y}}_t \leftarrow \hat{\vec{y}}_t \cup predictedNode.label$;
12      end
13      $UpdateStatisticalDescriptors(\vec{x}_t, \vec{y}_t)$;
14  end
```

Algorithm 3.6 - GNB-hDS: Gaussian Naive Bayes for hierarchical data stream classification.

The algorithm receives a hierarchical data stream hDS providing instances $(\vec{x}_t, \vec{y}_t)$ over time t and, if required, outputs a set of predicted labels (a label path) $\hat{\vec{y}}_t$ for each given instance $(\vec{x}_t, \vec{y}_t)$, where $\vec{x}_t$ represents a d-dimensional feature set and its values, and $\vec{y}_t$ represents the corresponding ground-truth label path of that instance.

The algorithm starts by understanding and representing the class taxonomy from the hierarchical data stream. The first loop (line 2 onwards) receives an incoming instance from the hierarchical data stream. The following loop (lines 4 - 12) handles the hierarchy using the LCPN approach by predicting one of the children labels possible for that parent node.

The *a priori* probabilities are calculated in line 6 using the counts of class instances. The likelihood and posterior probabilities are calculated in lines 8 and 9 by the application of Equations (10) and (11), respectively.

The predicted node for the evaluated parent is obtained in line 10, and the respective single label is appended to a partial label path $\hat{\vec{y}}_t$ (line 11). This process is repeated until a leaf node is reached and the label path $\hat{\vec{y}}_t$ is complete and ready to be output by the algorithm.

Finally, the algorithm updates the statistical descriptors (the count n of class instances, the incremental mean instance $\bar{x}_n$, and the incremental standard deviation σ_n) of all classes contained in $\vec{y}_t$, from the leaf to the root class.

As aforementioned, GNB-hDS can also store adaptive statistical descriptors instead of incremental ones. Figure 3.5 illustrates the process performed by GNB-hDS when handling the data stream adaptively (cf. Figure 3.4).

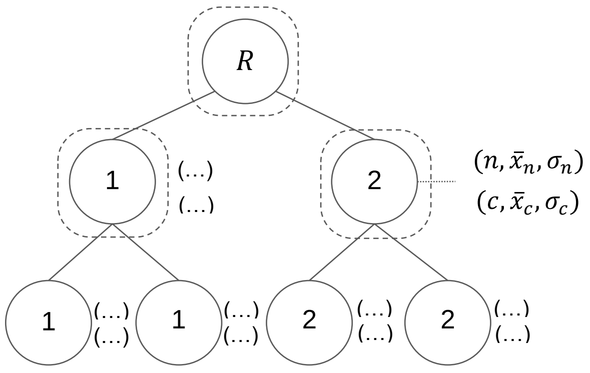


Figure 3.5 - Illustration of GNB-hDS method using adaptive data stream handling.

As in the incremental variant, each node in the tree stores the statistical descriptors $(n, \bar{x}_n \sigma_n)$ of the represented class. However, the adaptive GNB-hDS also stores one additional set $(c, \bar{x}_c, \sigma_c)$ representing the current statistical description of the last c instances.

The prediction of the class to be assigned to an incoming instance provided from the data stream is performed in the same way in both incremental and adaptive variants. However, in the adaptive GNB-hDS, both sets $(n, \bar{x}_n \sigma_n)$ and $(c, \bar{x}_c \sigma_c)$ of statistical descriptors are used to compute the historical and the current likelihood probabilities. Thus, the method obtains the maximum value of the historical *a posteriori* probability weighted by the current *a posteriori* probability.

To update the statistical descriptors adaptively, the method performs a forgetting strategy by subtracting one pseudo-instance (a mean instance) from the current statistical descriptors $(c, \bar{x}_c \sigma_c)$ representing the oldest instance.

Algorithm 3.7 shows the pseudocode for the adaptive variant of the GNB-hDS method.

In addition to the hierarchical data stream hDS, the adaptive GNB-hDS algorithm also receives the w (window) parameter representing the maximum number of instances to be considered on the current statistical descriptors $(c, \bar{x}_c \sigma_c)$.

```
Algorithm
GNB-hDS - Gaussian Naive Bayes (adaptive) for Hierarchical Data Streams
Input
 $hDS$:  a hierarchical data stream providing instances $(\vec{x}_t, \vec{y}_t)$ over time $t$
   $w$:  maximum number of instances on the current statistical descriptors
Output
 $\hat{\vec{y}}_t$:  a predicted label path for the input instance
-------------------------------------------------------------------------------
 1  $Tree \leftarrow classTaxonomy(hDS)$;
 2  foreach $(\vec{x}_t \in hDS)$ do
 3      $predictedNode \leftarrow Tree.root$;
 4      while $\neg(predictedNode.isLeaf)$ do
 5          foreach $(childNode \in predictedNode.children)$ do
 6              $priors_n \leftarrow priorProbability(childNode.Class)$;
 7              $priors_c \leftarrow priorProbability(childNode.Class)$;
 8          end
 9          $likelihood_n \leftarrow likelihoodProbability(\vec{x}_t, priors_n)$;
10          $posterior_n \leftarrow posteriorProbability(likelihood_n, priors_n)$;
11          $likelihood_c \leftarrow likelihoodProbability(\vec{x}_t, priors_c)$;
12          $posterior_c \leftarrow posteriorProbability(likelihood_c, priors_c)$;
13          $weightedPosterior \leftarrow posterior_n \times posterior_c$;
14          $predictedNode \leftarrow argmax(weightedPosterior)$;
15          $\hat{\vec{y}}_t \leftarrow \hat{\vec{y}}_t \cup predictedNode.label$;
16      end
17      $UpdateStatisticalDescriptors(\vec{x}_t, \vec{y}_t)$;
18      $correctNode \leftarrow Tree.\vec{y}_t$;
19      if $(correctNode.data_c.count > w)$ then
20          $UpdateStatisticalDescriptors(correctNode.data_c)$;
21      end
22  end
```

Algorithm 3.7 - GNB-hDS: Gaussian Naive Bayes (adaptive) for hierarchical data stream classification.

Compared to the incremental variant of GNB-hDS, the adaptive GNB-hDS additionally computes the *a priori* (line 7) and the *a posteriori (*lines 11 and 12) probabilities concerning the current statistical descriptors. Also, the weighted *a posteriori* probability is calculated on line 14.

After that, the method tests whether the number of instances represented in the current statistical descriptors $(c, \bar{x}_c \sigma_c)$ exceeds the stipulated maximum number *w* allowed (line 19). If so, it applies a sliding window strategy by forgetting a representation of the oldest instance of $(c, \bar{x}_c \sigma_c)$ (line 20).

As aforementioned, the GNB-hDS method uses the premise of a Gaussian (normal) data distribution to deal with instance representation in the learning model. Thus, the GNB-hDS method may present an additional advantage to the other methods that do not use the same premise since it would be more adapted to classify normally distributed data (PONTES, 2018).

Several studies have used data transformations to make input data more fitted to a given distribution and, consequently, to a learning model that uses this same distribution as a premise. There are examples from Engineering (JIANG; CUKIC; MENZIES, 2008) to Medicine (LIANG et al., 2020), and even general research in machine learning about how data transformation techniques affect the learning models' performance (ZHANG; YANG, 2017). However, applying data transformation in streaming settings is not straightforward. As data streams are potentially unbounded, there is no full dataset to be transformed and supplied to the learning model.

Next in this section, it is proposed an incremental adaptation of the well-known Yeo-Johnson Power Transformation (YEO; JOHNSON, 2000) for streaming settings. Traditionally, the Yeo-Johnson Power Transformation is applied considering a batch setup as it is applied to a fully known dataset, which is transformed into a more gaussian-like distributed dataset.

More specifically, the Yeo-Johnson Power Transformation is proposed here to be applied incrementally, one instance at a time, without prior knowledge about data, and adaptively along the data stream. Thereby, the proposed Incremental Yeo-Johnson Power Transformation can be attached to a data stream learning model that uses a gaussian (normal) data distribution as a premise, such as the well-known Naive Bayes method, in order to improve the prediction performance of this classifier when applied in a data stream scenario (HAN; PEI; KAMBER, 2011).

The traditional Yeo-Johnson Power transformation $\psi(\lambda, x)$ is defined as follows, where x stands for the input data and λ is the power parameter used for the transformation (YEO; JOHNSON, 2000).

$$\psi(\lambda, x) = \begin{cases} \{(x+1)^{\lambda} - 1\}/\,\lambda, & (x \geq 0, \lambda \neq 0) \\ log(x+1), & (x \geq 0, \lambda = 0) \\ -\{(-x+1)^{2-\lambda} - 1\}/(2-\lambda), & (x < 0, \lambda \neq 2) \\ -\,log(-x+1)\,, & (x < 0, \lambda = 2) \end{cases} \qquad (12)$$

The Yeo-Johnson Power transformation is based on the Box-Cox transformation (BOX; COX, 1964; SAKIA, 1992) and was proposed to cope also with negative values. When x is positive, the Yeo-Johnson transformation is the same as the Box-Cox with $(x+1)$. When negative, the Yeo-Johnson transformation is the Box-Cox of $(-x+1)$ with power $2-\lambda$.

Algorithm 3.8 shows the pseudocode for the traditional application of Yeo-Johnson Power transformation (YEO; JOHNSON, 2000).

```
Algorithm
Yeo-Johnson Power transformation - Traditional application
Input
  $D$:  A dataset with instances $\vec{x}$
  $L$:  a set of candidate λ values
Output
  $\widehat{D}$:  a dataset with transformed instances $\hat{\vec{x}}$
-----------------------------------------------------------------
$\hat{\lambda} \leftarrow \underset{\lambda \in L}{argmax} \ell(\lambda)$;
foreach $(\vec{x} \in D)$ do
    $\hat{\vec{x}} \leftarrow \begin{cases} \{(x+1)^{\lambda}-1\}/\lambda, & (x \geq 0, \lambda \neq 0) \\ log(x+1), & (x \geq 0, \lambda = 0) \\ -\{(-x+1)^{2-\lambda}-1\}/(2-\lambda), & (x < 0, \lambda \neq 2) \\ -log(-x+1), & (x < 0, \lambda = 2) \end{cases}$;
    $\widehat{D} \leftarrow \widehat{D} \cup \hat{\vec{x}}$;
end
```

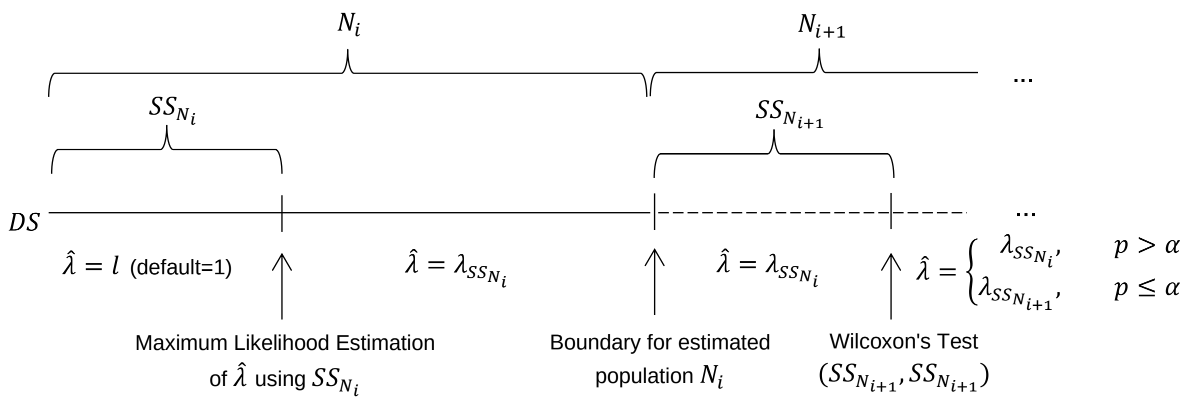
Algorithm 3.8 - Yeo-Johnson Power transformation: traditional application.

Note that the estimated/optimal λ, $\hat{\lambda}$, can be obtained by maximizing the log-likelihood function ℓ of the transformation power parameter λ with Maximum Likelihood Estimation (MLE) using all data D. Also, if $\hat{\lambda}$ is known (or chosen) *a priori*, the arguments of the maxima (line 1) for ℓ can be omitted and the transformation can be done on one single input x at a time (via Equation (12)).

Thus, to extend the traditional Yeo-Johnson Power transformation to handle unbounded data streams, the main concern is related to the estimation of the optimal $\hat{\lambda}$ on the fly and the use of a strategy to ensure that this estimation will remain accurate over time.

In this sense, the proposed Incremental Yeo-Johnson Power transformation tackles these issues by applying sample size determination and hypothesis testing to (i) find the optimal $\hat{\lambda}$ based on a sample set and (ii) check if two sample sets obtained from the data stream at different timestamps significantly differ, and thus require a new $\hat{\lambda}$ estimation.

Figure 3.6 illustrates the incremental Yeo-Johnson Power Transformation structure in a data stream scenario. Let DS be a data stream supplying instances over time, N_i a user-given size for a chunk of DS with $i \in \{1, \ldots, \infty\}$, and SS a sample size estimated based on N_i. In practice, N_i represents the number of instances observed prior to sample size determination (SS_{N_i}). Also, new incoming instances of N_{i+1} trigger new statistical validations resulting in SS_{N_i+1}, and so forth.



N_i
N_{i+1}
...
SS_{N_i}
$SS_{N_{i+1}}$
DS
...
$\hat{\lambda} = l$ (default=1)
$\hat{\lambda} = \lambda_{SS_{N_i}}$
$\hat{\lambda} = \lambda_{SS_{N_i}}$
$\hat{\lambda} = \begin{cases} \lambda_{SS_{N_i}}, & p > \alpha \\ \lambda_{SS_{N_{i+1}}}, & p \leq \alpha \end{cases}$
Maximum Likelihood Estimation of $\hat{\lambda}$ using SS_{N_i}
Boundary for estimated population N_i
Wilcoxon's Test $(SS_{N_{i+1}}, SS_{N_{i+1}})$

Figure 3.6 - Overview of the incremental Yeo-Johnson Power Transformation setup in a data stream scenario.

The process initially assumes $\hat{\lambda} = l$, where $l = 1$ is a user-defined parameter (note that $\lambda = 1$ results in equivalent input and transformed data). The method buffers SS_{N_i} instances and uses them to perform a Maximum Likelihood Estimation of $\hat{\lambda}$.

The sample size determination follows Equation (13), where N_i stands for the population estimated size, z for the standard normal distribution, p and q for the complementary proportions for the population, and d for the error component of interval estimate. The sample size determination considers population proportions and also a finite population correction (DANIEL; CROSS, 2018).

$$SS_{N_i} = \frac{N_i z^2 pq}{d^2(N_i - 1) + z^2 pq} \tag{13}$$

The Maximum Likelihood Estimation of $\hat{\lambda}$ is performed via Brent's algorithm (BRENT, 2013) by minimizing the negative log-likelihood function of the transformation power parameter λ for the input data buffered (SS_{N_i}), resulting in an optimal $\lambda_{SS_{N_i}}$, which is used for the ongoing data transformation.

Once the data stream DS has provided N_i instances, the statistical warranty boundary for the estimated population N_i is reached. In other words, the sample set buffered in SS_{N_i} may not be representative anymore for DS. Thus, the process is restarted by buffering the newest SS_{N_i+1} instances to compare both sample sets SS_{N_i} and SS_{N_i+1}.

When SS_{N_i+1} is reached, a Wilcoxon's hypothesis test (WILCOXON, 1992) is performed to verify significant differences between both sample sets SS_{N_i} and SS_{N_i+1}. The Wilcoxon's test is applied with a user-defined significance level α (default $=\ 0.05$), according to the protocol provided in (DEMŠAR, 2006).

Thus, the new value of $\hat{\lambda}$ is assigned as follows (Equation (14)) according to the $p\text{-}value$ returned from Wilcoxon's test. If both sample sets are significantly different, the optimal $\hat{\lambda}$ is recomputed for the newest input data buffered ($SS_{N_{i+1}}$); otherwise, the transformation continues with the $\lambda_{SS_{N_i}}$ computed on the previously MLE.

$$\hat{\lambda} = \begin{cases} \lambda_{SS_{N_i}}, & p\text{-}value > \alpha \\ \lambda_{SS_{N_{i+1}}}, & p\text{-}value \leq \alpha \end{cases} \tag{14}$$

Algorithm 3.9 shows the pseudocode for the proposed Incremental Yeo-Johnson Power Transformation fitted for data stream classification.

Note that, oppositely to the traditional application of the Yeo-Johnson Power Transformation (cf. Algorithm 3.8), the algorithm receives a data stream (DS) providing instances $\vec{x}_t$ over time instead of a complete dataset. In addition, the algorithm receives the parameters needed to perform the sample size determination (see Equation (13)) and an initial *lambda* set $\vec{l}$ (default $= 1$).

The loop started on line 3, and the transformations performed on line 33 are equivalent to the ones on the traditional algorithm. The difference between both algorithms lies in how the $\hat{\lambda}$ estimation is performed. In the traditional algorithm, the estimation considers the entire dataset (cf. Algorithm 3.8, line 1), whereas, in the incremental strategy, this process is made along the data stream (lines 4-32).

First, the algorithm determines the sample size (line 1) and assigns the initial *lambda* to $\hat{\lambda}$. Note that $\hat{\lambda}$ is d-dimensional as the $\vec{x}_t$, with possibly distinct values for each d.

Inside the loop, the algorithm buffers $|SS_{N_i}|$ instances (line 5) until the computed sample size and uses them to perform the first $\hat{\lambda}$ estimation (line 9). Note that, if provided, MLE considers a set L of candidate λ values. However, this is not critical when using Brent's algorithm, and the final estimation may not be a member of L (BRENT, 2013).

Algorithm

Incremental Yeo-Johnson Power Transformation for Data Stream Classification

Input

DS: a data stream providing instances $\vec{x}_t$ over time $t \in \mathbb{N}$
L: a set of candidate λ
N: population estimated size
d: error component of interval estimate
p: proportion for population ($q = 1 - p$)
q: $1 - p$
α: significance level
$\vec{l}$: Initial *lambda*

Output

$\widehat{DS}$: an incrementally transformed data stream

```
$SS \leftarrow (Nz^2pq)/(d^2(N-1) + z^2pq)$;
$\hat{\lambda} \leftarrow \vec{l}$;
foreach ($\vec{x}_t \in DS$) do
  if ($|SS_{N_i}| < SS$) then
    $SS_{N_i} \leftarrow SS_{N_i} \cup \vec{x}_t$;
  end
  else
    if $\neg(estimated)$ then
      $\hat{\lambda} \leftarrow \underset{\lambda \in L}{argmax} \ell(SS_{N_i})$;
      $Model \leftarrow \emptyset$;
      $estimated \leftarrow \top$;
    end
    else
      if ($t \bmod N = 0$) then
        $tested \leftarrow \perp$;
      end
      if ($|SS_{N_{i+1}}| < SS$) then
        $SS_{N_{i+1}} \leftarrow SS_{N_{i+1}} \cup \vec{x}_t$;
      end
      else
        if $\neg(tested)$ then
          $p\text{-}value \leftarrow \text{Wilcoxon}(SS_{N_i}, SS_{N_{i+1}})$;
          if ($p\text{-}value <= \alpha$) then
            $\hat{\lambda} \leftarrow \underset{\lambda \in L}{argmax} \ell(SS_{N_{i+1}})$;
            $Model \leftarrow \emptyset$;
            $SS_{N_i} \leftarrow SS_{N_{i+1}}$;
          end
          $tested \leftarrow \top$;
        end
      end
    end
  end
  $\hat{\vec{x}}_t \leftarrow \begin{cases} \{(x+1)^{\lambda} - 1\}/\lambda, & (x \geq 0, \lambda \neq 0) \\ log(x+1), & (x \geq 0, \lambda = 0) \\ -\{(-x+1)^{2-\lambda} - 1\}/(2-\lambda), & (x < 0, \lambda \neq 2) \\ -log(-x+1), & (x < 0, \lambda = 2) \end{cases}$;
  $\widehat{DS} \leftarrow \widehat{DS} \cup \hat{\vec{x}}_t$;
end
```

Algorithm 3.9 - Incremental Yeo-Johnson Power Transformation for data stream classification.

When a new $\hat{\lambda}$ estimation is performed, it is necessary to clear (or at least update) any current learning model (line 10) since any new instance $\vec{x}_t$ will be transformed into $\hat{\vec{x}}_t$ using a different power parameter λ.

The algorithm controls when the theoretical population size is reached (line 14) and starts to populate a new sample set with $|SS_{N_{i+1}}|$ instances (line 18). When $SS_{N_{i+1}}$ is full, the algorithm tests both initial SS_{N_i} and current $SS_{N_{i+1}}$ sample sets with the Wilcoxon's Test (line 22). If both sets are significantly different, the $\hat{\lambda}$ is re-estimated using $SS_{N_{i+1}}$ (line 24), and the current sample set is defined as the reference sample set.

Observe that a learning model can be appended by replacing line 34 and works with every single $\hat{\vec{x}}_t$ over time without any inner changes to the model since all the steps described in the proposed Incremental Yeo-Johnson Power Transformation are performed before the instance $\vec{x}_t$ is supplied to the classifier.

Finally, two key aspects related to the application of the described Incremental Yeo-Johnson Power Transformation in the data stream classification scenario are noteworthy. First, every $\hat{\lambda}$ estimation requires a new learning model that, at least, is more concerned with the newest data since subsets of transformed data with different λ present distinct distributions and may mislead the classifier. Also, note that this aspect justifies the use of Wilcoxon's test instead of promptly recomputing the optimal *lambda* with the newest buffer to try to avoid a change in $\hat{\lambda}$ and keep more representative data.

Second, depending on the error component of interval estimate (d), few instances can theoretically represent unbounded data. For example, assuming a confidence level of 95% and a margin of error of 2%, the sample size is bounded to 2,401 samples regardless of N size. Naturally, this is not the case in a data stream classification problem. Thus, recomputing λ can act as a response to concept drifts (TSYMBAL, 2004) on the underlying distribution of the data stream. In other words, the DS size (N) can represent a user-defined boundary to renew the statistical warranty about the sample set with a trade-off between responsiveness to concept drifts and maintaining historical data.

4 ANALYSIS

This section describes the experimental setup for comparing the proposed learning methods with previously identified related work and with each other to assess the proposals of this thesis concerning existing methods. Next, the experimental protocol is described, including the hierarchical data stream sets used in the experiments and the validation protocol (evaluation metrics and statistical validation).

Sections 4.2, 4.3, and 4.4 show the results obtained by the proposed methods. Note that the sections are related to each other in both method and analysis sections (e.g., 3.2 describes Global kNN-hDS, while 4.2 depicts its analysis).

Finally, the last section (4.5) shows a multiple comparison experiment between all proposed methods following a distinct protocol attempting to benchmark the hierarchical data stream classification area.

4.1 EXPERIMENTAL PROTOCOL

4.1.1 Datasets

The systematic literature review in Section 2.3 listed the datasets used in studies in the hierarchical data stream classification area. However, as previously discussed, most studies represented false positives for the target area when considering all its constraints. As an outcome, the datasets used by the studies returned in the review do not necessarily represent hierarchical data stream sets.

Consequently, to obtain datasets able to be used as input data into a hierarchical data stream classification, the datasets were filtered to keep only the ones with complete adherence to the hierarchical data stream classification area, resulting in three hierarchical data streams proposed in (PARMEZAN; SOUZA; BATISTA, 2018).

In addition, as previously introduced in Section 2.4, the authors in (SOUZA et al., 2020) introduced a new version of one of the datasets used in (PARMEZAN; SOUZA; BATISTA, 2018), with more instances and a formal definition of the concept drifts within the data stream. This dataset was presented by the authors as a flat dataset, but it maintained the intrinsic hierarchical structure as initially proposed in (PARMEZAN; SOUZA; BATISTA, 2018) since the instances still represent species of disease vector mosquitoes, which are naturally organized in an entomological taxonomy.

Table 4.1 depicts the resulting 14 hierarchical data stream sets used in the experiments, comprehending the three data sets proposed in (PARMEZAN; SOUZA; BATISTA, 2018) plus the 11 hierarchically labeled datasets resulting from the hierarchy incorporation on the Insects datasets proposed in (SOUZA et al., 2020).

Note that, among the datasets proposed in (SOUZA et al., 2020), the “Insects-o-o-c” represents the main dataset, while other ones represent datasets built with different sampling strategies to simulate natural effects in insects' behavior and induce distinct concept drifts on data. Also, in the same datasets, the words “Abrupt”, “Balanced”, “Gradual”, “Imbalanced”, “Incremental” and “Reoccurring” are reduced to their initials to increase readability.

Table 4.1 - Hierarchical data stream sets used in the experiments.

Dataset	Instances	Features	Classes	Labels per level	Reference
Entomology	21,722	33	14	4,6,9,14	(PARMEZAN; SOUZA; BATISTA, 2018)
Ichthyology	22,444	15	15	2,6,10,15	(PARMEZAN; SOUZA; BATISTA, 2018)
Insects-a-b	52,848	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-a-i	355,275	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-a-r-b	79,986	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-a-r-i	452,044	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-b	57,018	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-g-b	24,15	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-g-i	143,323	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-i	452,044	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-r-b	79,986	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-i-r-i	452,044	33	6	1,1,2,6	(SOUZA et al., 2020)
Insects-o-o-c	905,145	33	24	4,10,14,24	(SOUZA et al., 2020)
Instruments	9,419	30	31	5,10,31	(PARMEZAN; SOUZA; BATISTA, 2018)

4.1.2 Validation protocol

During the experiments, the predictive correctness of the classifiers was measured using the hierarchical F-Measure (hF) (KIRITCHENKO et al., 2005) following a prequential evaluation method (interleaved test-then-train) (BIFET; KIRKBY, 2009; GAMA; SEBASTIÃO; RODRIGUES, 2013).

As previously described, the hierarchical F-Measure is the harmonic mean of its hierarchical precision (hP) and hierarchical recall (hR) components (Equations (1), (2), and (3)). The hF rates were computed and incrementally averaged for all incoming instances from the hierarchical data stream.

Regarding performance assessment, the time performance of all methods was measured by calculating the number of instances that the method can process and classify per second ($inst/s$).

The methods proposed in this thesis were compared to the Local kNN-hDS method, previously described as a related work (Section 2.3.1) and understood as a state-of-the-art method in the hierarchical data stream classification area since it is the only retrieved study on the SLR that fulfills the area constraints.

Table 4.2 summarizes the parameters used in all methods across all experiments. Note that all the methods were experimented with identical parameters when applicable. The comparison between Local and Global kNN-hDS used identical n and k parameters. Then, the comparison between the Local kNN-hDS and kNC-hDS and Dribble-hDS also used n and k, in addition to m used on both proposed methods to perform the summarization strategy. Differently, the adaptive variant of GNB-hDS uses a w parameter to perform a forgetting strategy not related to the other methods.

Table 4.2 - Parameter settings used on methods across experiments.

Method	Parameter	Description	Experimented values
Local kNN-hDS	n	Maximum number of instances to be stored in a node	{1, 5, 10, 15, 20}
	k	Nearest neighbors	{1, 3, 5}
Global kNN-hDS	n	Maximum number of instances to be stored in a node	{1, 5, 10, 15, 20}
	k	Nearest neighbors	{1, 3, 5}
kNC-hDS (Local and Global)	n	Number of centroids	{1, 5, 10, 15, 20}
	m	Maximum number of instances summarized in a centroid	{5, 10, 30}
	k	Nearest centroids	{1, 3, 5}
Dribble-hDS (Local and Global)	n	Number of CFs	{1, 5, 10, 15, 20}
	m	Maximum number of instances summarized in a CF	{5, 10, 30}
	k	Nearest CF means	{1, 3, 5}
GNB-hDS (when adaptive)	w	Maximum number of instances on the current statistical descriptors	{10,50,100,500,1000,5000}

Besides, the GNB-hDS method was experimented over six method variations resulting from the incremental and adaptive variants of the method, plus the use of the incremental Yeo-Johnson Power transformation. These six variations are specified in Table 4.3.

Table 4.3 - GNB-hDS variants specification.

Base method	**Data stream handling**	**Window strategy**	**Incremental Yeo-Johnson**	**Method name**
GNB-hDS	Incremental	-	No	GNB-hDS
			Yes	GNB-hDS-iYJ
	Adaptive	Cw (Only the statistical descriptors of the current window are considered)	No	GNB-hDS-Cw
			Yes	GNB-hDS-Cw-iYJ
		Hw (The historical statistical descriptors are weighted by the statistical descriptors of the current window)	No	GNB-hDS-Hw
			Yes	GNB-hDS-Hw-iYJ

Furthermore, the proposed incremental Yeo-Johnson Power Transformation was evaluated by measuring data normality and its impact on prediction performance. Data normality was measured using the Shapiro-Wilk test of normality (SHAPIRO; WILK, 1965). The test was performed on all data streams considering the original data, the transformed data with access to the complete data stream (known DS), and the data incrementally transformed with the proposed incremental Yeo-Johnson Power Transformation.

The prediction performance of the incremental primary version of GNB-hDS was measured when applied with and without the incremental Yeo-Johnson Power Transformation attached, using the same datasets previously specified in Table 4.1. It is noteworthy that the Incremental Yeo-Johnson Power Transformation can be attached to any data stream classifier without modifications in the main process since it represents an additional on-the-fly data preprocessing step.

The GNB-hDS and GNB-hDS-iYJ classifiers were applied to scenarios with the original data stream, the transformed data with access to the complete data stream (known DS), and the incrementally transformed data.

All the experiments related to the incremental Yeo-Johnson Power Transformation were performed using $N = 1.0 \times 10^7$, $d = (0.95, 0.02)$, $p = 0.5$, $\alpha = 0.05$ and $\vec{l} = 1$ (see Algorithm 3.9). Note that the population estimated size N was set up with the first power of ten that reaches the upper bound (2,401) of sample size determination (see Equation (13)) and simulates a population estimated size always bigger than the known DS. In addition, the pair of values provided in d represent, respectively, the confidence level and the margin of error components of d.

Finally, the results obtained by all methods were compared using significance tests of pairwise or multiple comparisons considering a 95% confidence level according to the protocol provided in (DEMŠAR, 2006). The sample sets used as a basis for the statistical tests comprehend ordinal, non-parametric data, representing the evaluation metrics obtained by the classifiers in different datasets, assuming the null hypothesis that there is no significant difference between the results of the classifiers. More specifically, the Wilcoxon's hypothesis test (WILCOXON, 1992) was used to perform pairwise comparisons, and the Friedman's hypothesis test (FRIEDMAN, 1937) to make multiple comparisons in non-parametric data assuming a null hypothesis that there is no significant difference between the results of all methods in terms of predictive and performance rates. In the multiple comparison scenario, and in the case of the null hypothesis being rejected, the Nemenyi *post hoc* test (NEMENYI, 1963) was applied to identify significant differences between two specific classifiers.

Finally, in Section 4.5, a separate analysis was conducted to compare all proposed methods together. In addition to the above-mentioned statistical validation tests, this analysis compared the methods using the previously described hF and $inst/s$ rates understood as multiple attributes in a problem of Multi-Criteria Decision Making.

Multi-Criteria Decision Making (MCDM) is a problem-solving technique fitted to deal with multiple conflicting objectives. It was first introduced in (ZIONTS, 1979) and used to develop multiple criteria metrics for the evaluation of data mining algorithms in (NAKHAEIZADEH; SCHNABL, 1997).

The MCDM analyzes distinct alternatives representing possible solutions to a problem. These alternatives are assumed to be finite and eventually ranked. The problem is associated with multiple decision criteria representing the dimensions from which the alternatives can be evaluated. Also, the criteria are usually conflicting and depicted via incommensurable units (TRIANTAPHYLLOU, 2000).

Formally, an MCDM problem can be expressed as a finite set $A = a_i,\ i = \{1, 2, 3, \dots, n\}$ of alternatives and a finite set $G = g_j, j = \{1, 2, 3, \dots, m\}$ of goals or criteria, and each goal has an associated weight W (or desirability). The solution determines the optimal alternative a that maximizes the degree of desirability concerning all goals g_j (TRIANTAPHYLLOU, 2000).

The analysis performed in Section 4.5 understands the proposed methods as the alternatives in the MCDM problem. The goals, or multiple decision criteria, are instantiated by hF and $inst/s$ rates.

Specifically, the MCDM was applied in a weighted product model (WPM) where each alternative (classifier) is compared against other alternatives by multiplying ratios for each weighted criterion (hF and $inst/s$ rates). When used to compare or rank multiple alternatives, a variant approach of the WPM that uses only products without ratios can be used. Equation (15) shows the MCDM-WPM calculation, where n is the number of criteria, a_{ij} is the metric value obtained by the $i\text{-}th$ classifier concerning the $j\text{-}th$ criterion, and w_j is the weight of $j\text{-}th$ criterion (TRIANTAPHYLLOU, 2000).

$$WPM(A_i) = \prod_{j=1}^{n} \left(a_{ij}\right)^{w_j} \tag{15}$$

The hF and $inst/s$ rates were normalized using the linear scale transformation (Max-Min method), considering both maximum and minimum performance ratings of hF and $inst/s$. The normalized value r_{ij} obtained by the linear scale transformation is calculated via Equation (16), where a_{ij} is the metric value obtained by the $i\text{-}th$ classifier concerning the $j\text{-}th$ criterion, and a_j^{max} and a_j^{min} are the maximum and the minimum performance rate obtained by the classifiers concerning the $j\text{-}th$ criterion (ÇELEN, 2014).

$$r_{ij} = \frac{a_{ij} - a_j^{min}}{a_j^{max} - a_j^{min}} \tag{16}$$

Besides, a constant equal to 1.00×10^{-6} was added to r_{ij} to avoid zero multiplication (which would result in ignoring the $j\text{-}th$ criterion) when $a_{ij} = a_j^{min}$.

Finally, the weights of the hF and $inst/s$ rates were defined following percentages $w \in \left\{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\right\}$, with hF and $inst/s$ assuming complementary values on five distinct scenarios representing different importance given to prediction correctness and speed performance.

4.2 GLOBAL K-NEAREST NEIGHBORS FOR HIERARCHICAL DATA STREAMS (GLOBAL KNN-HDS)

Table 4.4 shows the hierarchical F-Measure obtained by both Local kNN-hDS and Global kNN-hDS classifiers in the hierarchical data stream sets (greater values highlighted in bold). These results represent the best hF rates obtained by methods considering the averaged best-performing parameter configuration across all datasets. In addition, Table 4.4 also depicts, in the last row, the average ranking for the methods per dataset.

Table 4.4 - Global kNN-hDS results: Hierarchical F-Measure (%) obtained by methods considering the averaged best-performing parameter setting across all datasets.

	Local kNN-hDS	**Global kNN-hDS**
Datasets	$n = 20$ $k = 1$	$n = 20$ $k = 1$
Entomology	**51.51**	51.48
Ichthyology	**40.55**	40.54
Insects-a-b	80.95	80.95
Insects-a-i	79.14	79.14
Insects-i-a-r-b	79.49	**79.52**
Insects-i-a-r-i	78.52	**78.53**
Insects-i-b	79.78	79.78
Insects-i-g-b	83.29	**83.41**
Insects-i-g-i	78.94	**78.95**
Insects-i-i	78.63	**78.64**
Insects-i-r-b	80.14	**80.18**
Insects-i-r-i	78.60	**78.61**
Insects-o-o-c	55.24	**55.28**
Instruments	**65.42**	65.06
Avg. hF	**72.16**	72.15
Avg. Ranking	1.68	**1.32**

Both methods obtained their best results with more comprehensive data representations (n = 20) and with minimal nearest neighbors (k = 1).

The Global kNN-hDS method obtained better hF rates in 8 out of the 14 datasets. However, hF values are similar across local and global approaches, such that the average difference between them is 0.01%, favoring the local proposal. The best average ranking of 1.32 against 1.68 of the local method also reflects the better hF rates achieved by Global kNN-hDS. Yet, despite the improvements, a Wilcoxon's signed-rank test showed that there is no statistical difference between hF rates obtained by the methods (p-$value$ = 0.1949).

In terms of processing time, Table 4.5 compares the average number of instances per second processed by both methods in each dataset with the same best-performing parameters settings (greater values highlighted in bold).

Table 4.5 - Global kNN-hDS results: instances per second rates obtained by methods with averaged best-performing parameters settings.

	Local kNN-hDS	**Global kNN-hDS**
Datasets	$n = 20$ $k = 1$	$n = 20$ $k = 1$
Entomology	127	**209**
Ichthyology	157	**223**
Insects-a-b	151	**383**
Insects-a-i	153	**384**
Insects-i-a-r-b	153	**386**
Insects-i-a-r-i	153	**374**
Insects-i-b	148	**363**
Insects-i-g-b	158	**385**
Insects-i-g-i	154	**385**
Insects-i-i	152	**377**
Insects-i-r-b	153	**388**
Insects-i-r-i	153	**386**
Insects-o-o-c	75	**135**
Instruments	79	**110**
Avg. $inst/s$	140.43	**320.50**
Avg. Ranking	2.00	**1.00**

The Global kNN-hDS method was able to process more instances per second across all datasets, with an average rate of 320.50 instances against 140.43 of the local approach. Note that this value varies according to the number of features in the dataset (it needs more computational effort to calculate distances between two instances) and the number of classes (more comparisons due to the number of instances in memory buffers of each class node).

On average, the global method was able to process 2.28 times more instances than the local approach. The Instruments, Ichthyology, Entomology, and Insects-o-o-c datasets (in that order) resulted in the smallest differences in the rate of instances per second obtained by both methods, equal to 1.56 times on average. In contrast, in all other insect datasets, the global approach was able to process 2.49 times more instances than the local approach.

The rates of instances processed per second were submitted to a one-tailed Wilcoxon's test to verify whether the rates achieved by the global approach are greater than the rates of the local method. The test indicated a statistical difference between instances per second rates obtained by both methods ($p\text{-}value = 4.88 \times 10^{-4}$) and confirmed that Global kNN-hDS is significantly faster when compared to the local approach.

In addition to the general analysis, the behavior of both methods over variations of the n parameter was evaluated. Table 4.6 and Table 4.7 depict, respectively, the average Hierarchical F-Measure (%) and the average instances per second rates obtained by both methods on each variation of n.

Table 4.6 - Global kNN-hDS results: average Hierarchical F-Measure (%) obtained by methods on each variation of n.

n	Local kNN-hDS	Global kNN-hDS
1	62.82	**62.87**
5	67.51	**67.62**
10	69.90	**69.94**
15	71.17	**71.20**
20	72.08	**72.09**
Avg. hF	68.70	**68.74**
Avg. Ranking	2.00	**1.00**

Table 4.7 - Global kNN-hDS results: Average instances per second rates obtained by methods on each variation of n.

n	Local kNN-hDS	Global kNN-hDS
1	459.07	**541.04**
5	316.64	**481.11**
10	213.57	**405.80**
15	169.21	**350.19**
20	140.43	**310.19**
Avg. $inst/s$	259.78	**417.67**
Avg. Ranking	2.00	**1.00**

Global kNN-hDS showed similar results compared to the local approach when analyzing the hF averages in each variation of n. Still, the Global kNN-hDS method obtained a clear first ranking on the overall comparison.

Regarding instances per second rates, Global kNN-hDS stands out from Local kNN-hDS with higher rates on all n variations, processing around 417 instances per second. On average, Global kNN-hDS can process about 18% more instances with minimal data representation (n = 1), and more than twice as many instances with bigger data representations ($n \in \{15, 20\}$).

Also, note that Global kNN-hDS presents a small decrease in the instances per second rates with higher values in k (see Table 4.5, local average with n = 20 and k = 1) since the global approach via label path analysis is performed on more k label paths.

Finally, to validate the overall better results obtained by Global kNN-hDS in the experiments, a Wilcoxon's test was applied using as sample sets the results obtained by Global and Local kNN-hDS methods with all parameters considering the hF and $inst/s$ rates. Table 4.8 summarizes these sample sets by averaging all hF and $inst/s$ rates obtained by methods and shows the overall average ranking of methods.

Table 4.8 - Global kNN-hDS results: overall average Hierarchical F-Measure (%) and Instances per second rates obtained by methods.

		Local kNN-hDS	**Global kNN-hDS**
hF (%)	**Avg. hF**	69.60	**69.65**
	Avg. Ranking	1.81	**1.19**
$inst/s$	**Avg. $inst/s$**	259.79	**398.69**
	Avg. Ranking	2.00	**1.00**

Even with similar hF rates, a Wilcoxon's test (one-tailed) identified a statistical difference ($p\text{-}value = 8.46 \times 10^{-14}$) between both methods favoring the Global kNN-hDS method.

Regarding $inst/s$ rates, as expected, a Wilcoxon's test (one-tailed) also showed a statistical difference ($p\text{-}value = 1.77 \times 10^{-13}$) between both methods favoring the Global kNN-hDS method.

These results demonstrate that the Global kNN-hDS method can use the global approach to obtain a more effective strategy to classify hierarchical data streams, statistically outperforming the Local kNN-hDS method in specific prediction correctness analysis and in all processing speed comparisons.

4.3 K-NEAREST CENTROIDS FOR HIERARCHICAL DATA STREAMS (KNC-HDS) AND DRIBBLE FOR HIERARCHICAL DATA STREAMS (DRIBBLE-HDS)

The results obtained by the kNC-hDS and Dribble-hDS methods are presented together in this section comprehending local and global variants of the methods.

Table 4.9 shows the hierarchical F-Measure obtained by Local kNC-hDS, Local Dribble-hDS, Global kNC-hDS, and Global Dribble-hDS (also comparing them to the related work Local kNN-hDS) in the hierarchical data stream sets (greater values highlighted in bold). These results represent the hF rates obtained by methods considering the averaged best-performing parameter configuration across the different parameters experimented for each configuration. Note that the Local kNN-hDS results are the same presented in the previous section, shown here to facilitate the direct comparison.

Table 4.9 - kNC-hDS and Dribble-hDS results: Hierarchical F-Measure (%) obtained by methods considering the averaged best-performing parameter setting across all datasets.

	Local kNN-hDS	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**
Datasets	$n = 20$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$
Entomology	51.51	57.38	53.61	**57.41**	53.71
Ichthyology	40.55	41.52	37.00	**41.72**	37.11
Insects-a-b	80.95	**84.37**	83.33	**84.37**	83.33
Insects-a-i	79.14	**82.62**	82.55	82.60	82.55
Insects-i-a-r-b	79.49	**84.30**	83.42	84.28	83.42
Insects-i-a-r-i	78.52	**82.64**	82.11	82.62	82.11
Insects-i-b	79.78	**84.05**	83.91	84.03	83.91
Insects-i-g-b	83.29	**88.02**	86.66	87.99	86.66
Insects-i-g-i	78.94	82.91	**83.11**	82.93	**83.11**
Insects-i-i	78.63	82.58	**83.08**	82.57	**83.08**
Insects-i-r-b	80.14	**84.50**	83.48	**84.50**	83.48
Insects-i-r-i	78.60	82.63	**82.70**	82.62	**82.70**
Insects-o-o-c	55.24	**65.66**	59.50	65.56	59.50
Instruments	**65.42**	55.04	56.53	55.59	56.68
Avg. hF	72.16	75.59	74.36	**75.63**	74.38
Avg. Ranking	4.57	**2.00**	3.18	2.29	2.96

Overall, Local kNC-hDS and Local Dribble-hDS outperform Local kNN-hDS across 13 out of 14 datasets. Local kNC-hDS showed an average hF rate of 75.59% and Local Dribble-hDS of 74.36%. The difference from Local kNC-hDS to Local kNN-hDS was 3.43%, and from Local Dribble-hDS to the kNN method of 2.20%.

Besides, Local kNC-hDS and Local Dribble-hDS showed similar hF rates in all experiments with small differences favoring Local kNC-hDS (1.23% on average).

Concerning the higher hF rates obtained, Local kNC-hDS achieved greater rates in 10 out of the 14 datasets, and Local Dribble-hDS in 3 out of the 4 remaining hierarchical data streams.

Regarding global variants, Global kNC-hDS and Global Dribble-hDS methods obtained better results than Local kNN-hDS also in 13 of the 14 datasets, with average

hF rates of 75.63% and 74.38%, respectively, against values close to 72% of the Local kNN-hDS method.

Also, note that all local and global kNC-hDS and Dribble-hDS methods obtained similar rates across all datasets, with a maximum difference of 1.27% between Global kNC-hDS and Local Dribble-hDS.

Furthermore, Table 4.9 shows the average ranking for the hF performances of the methods. One can observe that Local kNC-hDS presents the best results with an average ranking of 2.00. Global kNC-hDS achieved a ranking close to second place with 2.29. Both local and global Dribble-hDS achieved rankings close to the third place, and Local kNN-hDS obtained the last place in 12 out of the 14 datasets, resulting in the lower rank of 4.57.

Finally, note that Local kNN-hDS and both kNC-hDS methods obtain their averaged best performances with the biggest data representation (n = 20), while both Dribble-hDS methods obtained their best performances with a smaller representation (n = 5), taking advantage of the data summarization strategy using CFs to represent data.

Next, Table 4.10 reports the number of instances classified per second by the methods during the experiments (greater values highlighted in bold).

Table 4.10 - Local kNC-hDS and Local Dribble-hDS results: instances per second rates obtained by methods with averaged best-performing parameters settings.

	Local kNN-hDS	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**
Datasets	$n = 20$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$
Entomology	127	134	327	200	**382**
Ichthyology	157	186	286	291	**380**
Insects-a-b	151	152	452	353	**543**
Insects-a-i	153	151	467	354	**542**
Insects-i-a-r-b	153	152	456	354	**541**
Insects-i-a-r-i	153	150	468	351	**542**
Insects-i-b	148	152	456	345	**541**
Insects-i-g-b	158	165	459	372	**542**
Insects-i-g-i	154	155	466	359	**543**
Insects-i-i	152	151	467	354	**545**
Insects-i-r-b	153	152	458	356	**543**
Insects-i-r-i	153	151	469	353	**540**
Insects-o-o-c	75	78	247	127	**275**
Instruments	79	121	221	164	**256**
Avg. $inst/s$	140.43	146.54	407.00	309.40	**479.60**
Avg. Ranking	4.57	4.43	2.07	2.93	**1.00**

Global Dribble-hDS obtained an absolute first place in the number of instances classified per second. Similarly, Local Dribble-hDS obtained a consistent second place in 13 out of 14 datasets, resulting in a ranking of 2.07.

This result was expected since Dribble-hDS methods obtained their best performance with a considerably smaller data representation (n = 5) when compared to other methods (n = 20).

Global kNC-hDS obtained third place in 13 out of 14 datasets taking advantage of its global approach when compared to Local kNN-hDS and Local kNC-hDS.

Using the same data representation, Local kNN-hDS and Local kNC-hDS obtained similar instances per second rates. Local kNN-hDS placed in the last place in 8 datasets against 6 of Local kNC-hDS.

Moreover, the behavior of the methods over variations of n was evaluated. Table 4.11 compares the average hF (%) rates obtained by methods on each variation of the n parameter.

Table 4.11 - kNC-hDS and Dribble-hDS results: average Hierarchical F-Measure (%) obtained by methods on each variation of n.

n	Local kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS
1	62.82	68.77	72.51	68.77	**72.52**
5	67.51	73.50	68.74	**73.52**	69.88
10	69.90	74.63	68.56	**74.65**	69.74
15	71.17	75.12	68.39	**75.14**	69.49
20	72.08	75.40	68.27	**75.44**	69.28
Avg. hF	68.70	73.48	69.29	**73.50**	70.18
Avg. Ranking	3.80	2.30	4.20	**1.50**	3.20

Global kNC-hDS and Global Dribble-hDS showed the best average results in all variations of n, with Global Dribble-hDS obtaining the best result with $n = 1$ and Global kNC-hDS with $n \in \{5, 10, 15, 20\}$, resulting in the best average ranking (1.50) for the Global kNC-hDS method.

The Local kNN-hDS method obtained its better hF rates with higher values of n in all datasets. Similarly, Local and Global kNC-hDS methods also perform better with more stored data (higher n). Meanwhile, both Dribble-hDS methods obtained their best mean results with smaller values in n (i.e., $n \in \{1,5\}$).

As previously stated, Dribble-hDS benefits from smaller numbers of CFs (n), thus achieving competitive hF rates even using less stored data.

The rationale behind the decrease of hF rates with the increase of n on Dribble-hDS is related to noise incorporation, as several CFs are potentially created to represent a few instances, and thus, are not as representative as those that incorporate most of the data. On the other hand, the performance of Dribble-hDS with small data representations is noticeable, as it with n = 1 manages to obtain similar or better hF rates than the other methods with n = 20.

Likewise, Table 4.12 compares the average $inst/s$ rates obtained by methods on each variation of the n parameter.

Table 4.12 - kNC-hDS and Dribble-hDS results: Average instances per second rates obtained by methods on each variation of n.

n	Local kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS
1	459.07	457.41	448.70	**551.76**	527.77
5	316.64	313.71	379.28	**467.20**	466.17
10	213.57	226.06	297.10	**398.77**	379.49
15	169.21	184.41	237.77	**351.00**	297.00
20	140.43	149.84	190.06	**310.36**	236.85
Avg. $inst/s$	259.79	266.29	310.58	**415.82**	381.46
Avg. Ranking	4.40	4.20	3.40	**1.00**	2.00

One can observe that the number of instances that methods can classify per second decreases according to higher n values. On average, when increasing $n \in \{1, 5, 10, 15, 20\}$, the number of instances classified drops by roughly 20% in each step. This behavior is expected since larger n values induce larger numbers of distance computations between query instances and stored data.

Furthermore, Global kNC-hDS and Global Dribble-hDS stand out from the other methods with the first and second rankings in all n variations, processing, respectively, around 415 and 381 instances per second. On average, both methods can process at least 70% more instances than the Local kNN-hDS method.

Besides, Local Dribble-hDS achieved a consistent third place since the increase of n affects the method less due to the use of an outlier control (see Section 3.3) and consequently fewer distance computations performed than Local kNN-hDS or Local kNC-hDS.

Finally, to look for differences in the results obtained by the methods in the experiments, a Friedman statistical test was applied using as sample sets the results obtained by methods with all parameters considering the hF and $inst/s$ rates.

Table 4.13 summarizes these sample sets by averaging all hF and $inst/s$ rates obtained by methods and shows the overall average ranking of methods used in the Friedman test.

Table 4.13 - kNC-hDS and Dribble-hDS results: overall average Hierarchical F-Measure (%) and Instances per second rates obtained by methods.

		Local kNN-hDS	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**
hF (%)	**Avg. hF**	69.60	74.39	73.77	**74.42**	73.80
	Avg. Ranking	4.79	2.21	3.08	**2.20**	2.72
$inst/s$	**Avg. $inst/s$**	259.79	272.54	336.59	**418.81**	395.49
	Avg. Ranking	4.67	3.93	3.30	**1.41**	1.69

First, the Friedman test showed a significant difference between the methods in both hF ($p\text{-}value = 8.00 \times 10^{-28}$) and $inst/s$ rates ($p\text{-} value = 3.27 \times 10^{-47}$). After that, a *post hoc* Nemenyi test was applied to perform pairwise comparisons.

Figure 4.1 and Figure 4.2 show the resulting two critical difference charts for the hF and $inst/s$ rates obtained by Local kNN-hDS, Local kNC-hDS, Local Dribble-hDS, Global kNC-hDS, and Global Dribble-hDS methods.

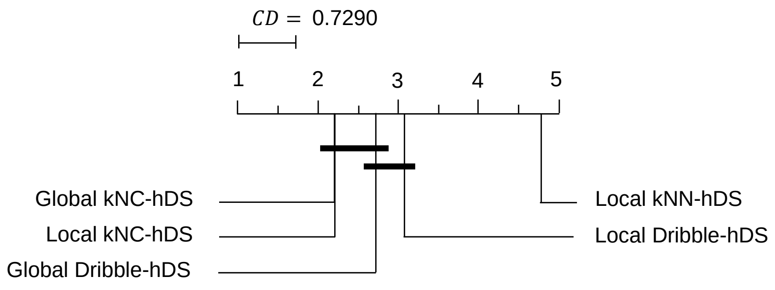


Figure 4.1 - kNC-hDS and Dribble-hDS results: critical differences chart for hF rates.

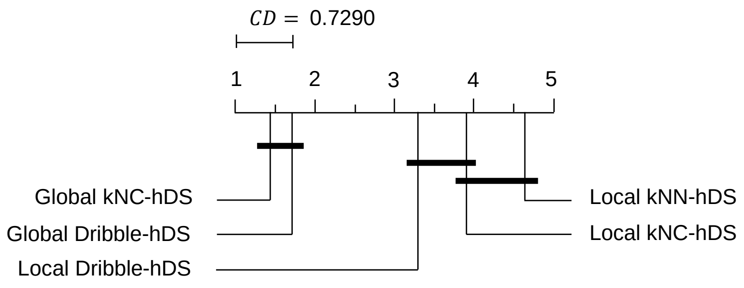


Figure 4.2 - kNC-hDS and Dribble-hDS results: critical differences chart for $inst/s$ rates.

Regarding hF rates, all proposed methods differ significantly from the Local kNN-hDS. Moreover, both kNC-hDS methods and Global Dribble-hDS do not differ significantly, with average rankings of 2.20, 2.21, and 2.72. Similarly, Global Dribble-hDS do not differ significantly from Local Dribble-hDS.

In contrast, Local kNN-hDS (in last place) obtained an average ranking of 4.79, surpassing more than twice the critical difference from the fourth place, the Local Dribble-hDS method.

These results suggest that kNC-hDS can obtain better hF rates than the other methods, and Dribble-hDS can also obtain better hF rates than Local kNN-hDS. Furthermore, when considering scenarios where data changes are less severe, and the entire data stream contains relevant information, the Local kNN-hDS method requires a memory buffer big enough to consider all concepts together in the sampled instances. However, this strategy may become infeasible due to computational resource constraints of specific scenarios. The same cannot be said for both Local and Global kNC-hDS and Dribble-hDS, which summarize the entire data stream via centroids and CFs, thus enabling a representation of the data using all training instances and not putting computational performance in jeopardy.

Regarding speed comparison, the *post hoc* Nemenyi test identified a clear difference between both global methods and the other ones, proving that Global kNC-hDS and Global Dribble-hDS methods can use the global approach to obtain a more effective strategy to classify hierarchical data streams, statistically outperforming local methods in processing time.

Furthermore, Local Dribble-hDS is significantly faster than Local kNN-hDS even using a local approach. Here, it is important to highlight that both kNC-hDS and Dribble-hDS methods use additional steps in their learning processes to apply data summarization strategies, specifically the creation of centroids in kNC-hDS and the creation and merging of CFs in Dribble-hDS. At first glance, this might indicate that both methods could be slower compared to the kNN method. However, the summarization property itself reverses this difference by summarizing data inside a centroid or a CF. Note that even using equal values in n, the kNC-hDS and Dribble-hDS methods manage to perform fewer distance computations within their data representations due to the summarization obtained by the parameter m while the centroid or the CF is not full.

Combining the analysis of prediction quality and computational resources, one can observe that kNC-hDS and Dribble-hDS methods have the advantage of summarizing information with different granularity levels depending on the problem, making them more versatile than the traditional hierarchical kNN method. Also, considering the better averaged best-performing results obtained by kNC-hDS in terms of hF and by Dribble-hDS regarding processing time, the best setup depends on specific data distribution characteristics and available resources.

4.4 GAUSSIAN NAIVE BAYES FOR HIERARCHICAL DATA STREAMS (GNB-HDS)

As previously introduced in Section 4.1.2, the GNB-hDS method was experimented over six distinct variations: GNB-hDS, GNB-hDS-iYJ, GNB-hDS-Cw, GNB-hDS-Cw-iYJ, GNB-hDS-Hw, GNB-hDS-Hw-iYJ (see Table 4.3).

Table 4.14 shows the hierarchical F-Measure obtained by all GNB-hDS variants plus the related work Local kNN-hDS (greater values highlighted in bold). These results represent the hF rates obtained by methods considering the averaged best-performing results across the different parameters experimented for each dataset.

Table 4.14 - GNB-hDS results: Hierarchical F-Measure (%) obtained by methods considering the averaged best-performing parameter setting across all datasets.

	Local kNN-hDS	**GNB-hDS**	**GNB-hDS -iYJ**	**GNB-hDS -Cw**	**GNB-hDS -Cw-iYJ**	**GNB-hDS -Hw**	**GNB-hDS -Hw-iYJ**
Datasets	$n = 20$ $k = 1$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Entomology	51.51	48.63	**52.82**	50.08	50.02	50.41	51.59
Ichthyology	40.55	46.82	**49.72**	46.64	47.15	47.39	48.80
Insects-a-b	80.95	81.11	81.96	86.49	**86.97**	86.10	86.81
Insects-a-i	79.14	80.88	84.03	85.46	86.17	85.53	**86.90**
Insects-i-a-r-b	79.49	81.42	84.07	85.84	85.94	86.21	**86.59**
Insects-i-a-r-i	78.52	81.57	83.70	84.93	85.51	85.00	**86.22**
Insects-i-b	79.78	80.55	82.40	83.83	84.05	83.65	**84.44**
Insects-i-g-b	83.29	81.53	81.50	**86.16**	86.14	85.64	86.03
Insects-i-g-i	78.94	80.40	83.38	**86.93**	85.42	85.23	86.00
Insects-i-i	78.63	80.90	83.18	84.97	85.50	84.93	**86.14**
Insects-i-r-b	80.14	78.57	80.21	85.79	86.08	85.64	**86.18**
Insects-i-r-i	78.60	81.61	83.72	84.88	85.51	84.99	**86.25**
Insects-o-o-c	55.24	64.14	**69.38**	59.33	62.40	61.47	66.54
Instruments	**65.42**	48.31	49.48	40.06	36.91	45.93	42.36
Avg. hF	72.16	72.60	74.97	75.10	75.27	75.58	**76.49**
Avg. Ranking	6.07	5.64	4.07	3.79	3.00	3.64	**1.79**

Overall, all GNB-hDS methods outperformed Local kNN-hDS. The primary incremental version (GNB-hDS) obtained a slight advantage of 0.45% over Local kNN-hDS. Its variant using the incremental Yeo-Johnson (GNB-hDS-iYJ) achieved a larger difference of 2.81%.

The adaptive versions of GNB-hDS using current windows (GNB-hDS-Cw and GNB-hDS-Cw-iYJ) obtained 75.10 and 75.27 respectively, and outperformed Local kNN-hDS and both GNB-hDS incremental versions. Likewise, the adaptive versions of GNB-hDS using historical windows (GNB-hDS-Hw and GNB-hDS-Hw-iYJ) achieved the highest hF rates, with GNB-hDS-Hw-iYJ obtaining the best hF rate in half of the datasets and the best average ranking of 1.79.

Also, note that the incremental Yeo-Johnson improved the hF rate obtained by the methods in all cases. The gains regarding GNB-hDS, GNB-hDS-Cw, and GNB-hDS-Hw were, respectively, 2.37, 0.17, and 0.91 when using the data transformation.

To validate the better results obtained by GNB-hDS when compared to the Local kNN-hDS, a Friedman test was applied using as sample sets the hF rates obtained by the methods.

The Friedman test showed a significant difference between the methods regarding the hF rates ($p\text{-}value = 6.50 \times 10^{-7}$) and a *post hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 4.3 shows the resulting critical difference chart for the hF rates obtained by all GNB-hDS methods.

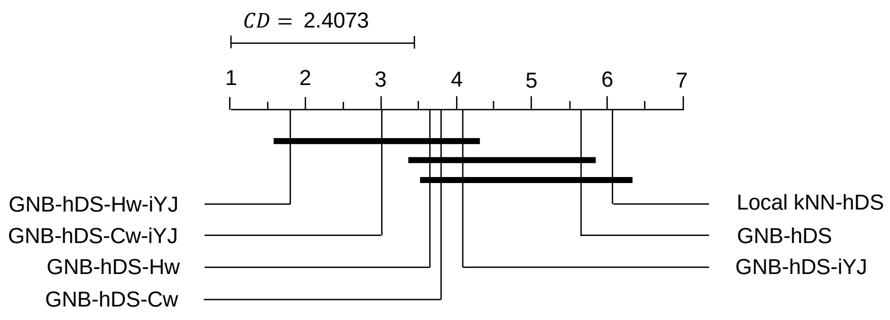


Figure 4.3 - GNB-hDS results: critical differences chart for hF rates.

Both adaptive GNB-hDS methods with the incremental Yeo-Johnson differ significantly from the Local kNN-hDS. Furthermore, both adaptive GNB-hDS methods without the incremental Yeo-Johnson (GNB-hDS-Cw and GNB-hDS-Hw) do not differ significantly from any other method, making possible improvements in processing time without losses in hF rates, as described below.

Table 4.15 compares the $inst/s$ rates obtained by all methods in each dataset with the same best-performing parameters settings (greater values highlighted in bold).

Table 4.15 - GNB-hDS results: instances per second rates obtained by methods with averaged best-performing parameters settings.

	Local kNN-hDS	**GNB-hDS**	**GNB-hDS -iYJ**	**GNB-hDS -Cw**	**GNB-hDS -Cw-iYJ**	**GNB-hDS -Hw**	**GNB-hDS -Hw-iYJ**
Datasets	$n = 20$ $k = 1$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Entomology	127	380	395	**397**	370	335	287
Ichthyology	157	395	**426**	398	394	366	338
Insects-a-b	151	**490**	472	469	437	374	325
Insects-a-i	153	**495**	473	472	444	378	325
Insects-i-a-r-b	153	**496**	467	475	441	374	320
Insects-i-a-r-i	153	**492**	465	476	442	376	323
Insects-i-b	148	501	**503**	500	454	402	334
Insects-i-g-b	158	**483**	469	466	441	371	323
Insects-i-g-i	154	**496**	470	475	444	377	326
Insects-i-i	152	**497**	479	473	445	385	323
Insects-i-r-b	153	**491**	459	474	439	363	324
Insects-i-r-i	153	**494**	451	476	443	377	323
Insects-o-o-c	75	**283**	277	280	271	237	212
Instruments	79	263	**284**	257	263	259	217
Avg. $inst/s$	140.43	**446.80**	434.95	434.77	409.01	355.35	307.20
Avg. Ranking	7.00	**1.50**	2.21	2.50	3.86	4.93	6.00

Similar to the hF rates, all GNB-hDS methods outperformed Local kNN-hDS in $inst/s$ rates. The primary incremental version (GNB-hDS) obtained the highest average $inst/s$ rate, processing more than three times the number of instances processed by Local kNN-hDS. Note that the other GNB-hDS variants obtained decreasing rates in the same order as shown in the table columns.

Note that the primary incremental version of GNB-hDS performs fewer steps than its variants, resulting in a higher $inst/s$. In this sense, the incremental Yeo-Johson and the forgetting strategy used on the adaptive versions of GNB-hDS constitute additional steps to the learning process and impact the computational performance.

Thus, one can understand that GNB-hDS-iYJ and GNB-hDS-Cw perform one additional step than GNB-hDS, GNB-hDS-Cw-iYJ and GNB-hDS-Hw perform two additional steps, and GNB-hDS-Hw-iYJ performs three additional steps since the application of the historical window results in the storage and processing of two distinct sets of statistical descriptors.

Besides, note that these additional steps represent opposite results in the hF and $inst/s$ rates. The highest hF rates were obtained by the methods that perform more additional steps, and, consequently, have lower $inst/s$ rates. Likewise, the GNB-hDS method, which obtained the lowest average hF rate, also obtained the highest number of instances processed per second.

As with hF rates, a Friedman test was applied using as sample sets the $inst/s$ rates obtained by the methods.

The Friedman test showed a significant difference between the methods regarding the $inst/s$ rates ($p\text{-}value = 1.27 \times 10^{-14}$) and a *post hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 4.4 shows the resulting critical difference chart for the $inst/s$ rates obtained by all GNB-hDS methods.

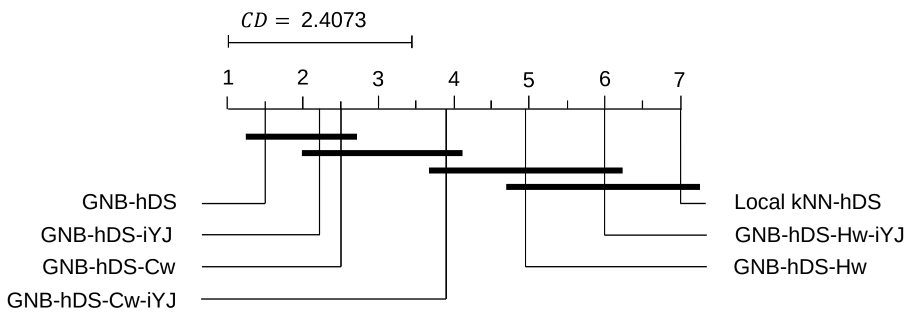


Figure 4.4 - GNB-hDS results: critical differences chart for $inst/s$ rates.

The incremental GNB-hDS, the GNB-hDS-iYJ, and both adaptive GNB-hDS methods using current windows (GNB-hDS-Cw and GNB-hDS-Cw-iYJ) differ significantly from the Local kNN-hDS. Also, note that Local kNN-hDS obtained the absolute last place, and GNB-hDS surpassed the local kNN by more than two critical distances. It is noteworthy that GNB-hDS-Cw-iYJ differs significantly from the Local kNN-hDS in both hF and $inst/s$ analyses.

Two additional analyses were also performed regarding the window parameter w and the effectiveness of the incremental Yeo-Johnson Power Transformation.

First, the behavior of the methods over variations of w was evaluated. Table 4.16 compares the average hF (%) rates obtained by methods on each variation of the w parameter. Observe that this analysis considers only the adaptive variants of GNB-hDS.

Table 4.16 - GNB-hDS results: average Hierarchical F-Measure (%) obtained by adaptive methods on each variation of w.

w	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
10	72.55	72.83	73.76	**74.52**
50	74.85	75.27	75.51	**76.52**
100	75.10	75.20	75.58	**76.49**
500	73.94	74.77	74.89	**76.01**
1000	73.92	74.84	74.66	**75.87**
5000	74.20	74.64	74.01	**75.28**
Avg. hF	74.09	74.59	74.73	**75.78**
Avg. Ranking	3.83	2.67	2.50	**1.00**

GNB-hDS-Hw-iYJ showed the best average results in all variations of w, resulting in a clear first ranking. Note that the average hF rates follow the same order that the one obtained in the best-performing results previously described (Table 4.14). Also, the best results of all methods occur with $w \in \{50,100\}$, suggesting that these values obtain a reasonable trade-off between historical and current data.

Regarding the effectiveness of the incremental Yeo-Johnson Power Transformation, the proposed data transformation was assessed regarding data normality and prediction performance, following the protocol previously described in Section 4.1.2.

Table 4.17 depicts the Shapiro-Wilk W Statistic for original, transformed (known DS), and incrementally transformed data streams (greater values highlighted in bold).

Table 4.17 - Incremental Yeo-Johson results: Shapiro-Wilk W Statistic for original, transformed, and incrementally transformed hierarchical data streams.

Datasets	Original	Transformed (known DS)	Incrementally transformed
Entomology	0.7489	**0.9517**	0.9513
Ichthyology	0.9028	**0.9839**	0.9773
Insects-a-b	0.7236	**0.9240**	0.9204
Insects-a-i	0.7248	**0.9272**	0.9229
Insects-i-a-r-b	0.7268	**0.9273**	0.9250
Insects-i-a-r-i	0.7234	0.9269	**0.9311**
Insects-i-b	0.7239	0.9239	**0.9254**
Insects-i-g-b	0.7280	**0.9273**	0.9148
Insects-i-g-i	0.7227	**0.9288**	0.9166
Insects-i-i	0.7234	0.9269	**0.9305**
Insects-i-r-b	0.7250	**0.9252**	0.9249
Insects-i-r-i	0.7234	0.9269	**0.9307**
Insects-o-o-c	0.7416	**0.9468**	0.9462
Instruments	0.9689	**0.9868**	0.9865
Avg. W	0.7577	**0.9381**	0.9360
Avg. Ranking	3.00	**1.29**	1.71

The average W Statistic obtained with the original data is 0.7577. In contrast, the averages obtained with both transformations (known DS and incremental) surpass 0.93. The W statistic is similar across all hierarchical data stream sets, with Yeo-Johnson being applied with the data known *a priori* and incrementally. The average W Statistic of traditional and incremental transformations differ by 0.0021.

Figure 4.5 shows the critical difference chart for the Shapiro-Wilk W Statistic obtained with original data and both transformations on the hierarchical data streams.

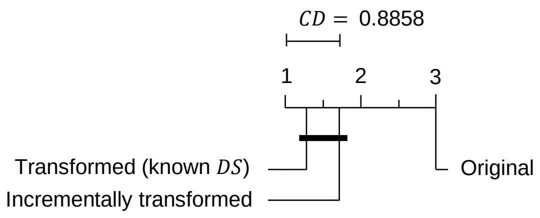


Figure 4.5 - Incremental Yeo-Johson results: critical differences chart for Shapiro-Wilk W Statistic on hierarchical data streams.

The test showed a significant difference between the Shapiro-Wilk W Statistic obtained with both transformations and the original data ($p\text{-}value = 1.45 \times 10^{-5}$). Also, the test confirmed no difference between both transformations. Thus, the Incremental Yeo-Johnson could achieve the same improvements in the data normality without accessing the complete data stream.

Regarding prediction performance, Table 4.18 depicts the Hierarchical F-Measure (hF) obtained by GNB-hDS with original, transformed, and incrementally transformed (GNB-hDS-iYJ) hierarchical data streams (greater values highlighted in bold).

As expected, both transformations improve the prediction performance of the GNB-hDS. The average hF obtained using the original data is 72.60%, and it is improved by more than 2% with transformed (known DS) and incrementally transformed data stream sets. The values obtained with both transformations are similar across all data streams, with an average difference of 0.02% favoring the incremental one.

It is noteworthy that the GNB-hDS-iYJ was even able to obtain better results using the incremental transformation than the traditional (known DS) transformation in 9 out of the 14 data streams, probably due to its ability to obtain a better λ estimation.

Table 4.18 - Incremental Yeo-Johson results: Hierarchical F-Measure (%) obtained with original, transformed, and incrementally transformed hierarchical data streams.

Datasets	Original	Transformed (known DS)	Incrementally transformed
Entomology	48.64	**53.87**	52.82
Ichthyology	46.82	**50.27**	49.72
Insects-a-b	81.11	81.90	**81.96**
Insects-a-i	80.88	**84.05**	84.03
Insects-i-a-r-b	81.42	83.48	**84.07**
Insects-i-a-r-i	81.57	83.40	**83.70**
Insects-i-b	80.55	82.28	**82.40**
Insects-i-g-b	**81.53**	81.42	81.50
Insects-i-g-i	80.40	83.16	**83.38**
Insects-i-i	80.90	83.05	**83.18**
Insects-i-r-b	78.57	79.58	**80.21**
Insects-i-r-i	81.61	83.45	**83.72**
Insects-o-o-c	64.14	69.46	69.38
Instruments	48.31	49.93	49.48
Avg. hF	72.60	74.95	**74.97**
Avg. Ranking	2.86	1.71	**1.43**

The hF rates obtained with all data transformations were submitted to a Friedman test to check if GNB-hDS and GNB-hDS-iYJ could achieve the same prediction results using both transformations.

Figure 4.6 shows the critical difference chart obtained after Friedman and Nemenyi tests for the hF rates obtained.

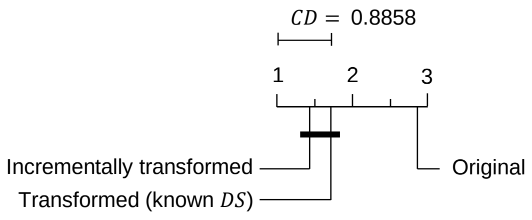


Figure 4.6 - Incremental Yeo-Johson results: critical differences chart for hF rates.

Friedman and Nemenyi's tests identified a difference between the performance of the classifiers with and without data transformations but not between the traditional (GNB-hDS) and the incremental transformations (GNB-hDS-iYJ) ($p\text{-}value = 3.35 \times 10^{-4}$).

These results corroborate the claims that the proposed Incremental Yeo-Johnson Power Transformation can be applied to a data stream classification learning model as an attached data processing step without the need for a full view of the input data and can still improve the prediction performance of the classifier by reducing the skewness of the data.

In conclusion, considering predictive performance and processing speed rates, GNB-hDS can obtain computational performance improvements without significant threats to the predictive performance by using statistical summaries of data combined with the class hierarchy information.

Moreover, as the GNB-hDS method uses the premise of a gaussian data distribution to deal with instance representation in the learning model, it presents an additional advantage to the Local kNN-hDS method (or to any other method that does not use the premise of data normality) since it is more adapted to classify normally distributed data.

This advantage can be even more noticeable when we consider the data stream context, where data are potentially unbounded and statistical descriptors, such as mean and standard deviation, are more likely to obtain better representations of the population.

4.5 BENCHMARKING

This section presents a separate analysis to compare all proposed methods attempting to benchmark the hierarchical data stream classification area.

The proposed methods were compared against the related work Local kNN-hDS and among each other. The comparisons were based on the averaged best performance of all methods (like the analysis carried out in the previous sections) and on a trade-off performance, which considered the best trade-off between hF and $inst/s$ rates obtained by all methods separately resulting from an MCDM-WPM analysis (as described in Section 4.1.2).

Table 4.19 depicts the overall hierarchical F-Measure obtained by all proposed methods (also comparing them to the related work Local kNN-hDS) in the hierarchical data stream sets (greater values highlighted in bold).

As aforementioned, these results represent the hF rates obtained by methods considering the averaged best-performing parameter settings.

Table 4.19 - Overall results: Hierarchical F-Measure (%) rates obtained by methods with averaged best-performing parameters settings.

	Local kNN-hDS	Global kNN-hDS	Local kNC-hDS	Local Dribble-hDS	Global kNC-hDS	Global Dribble-hDS	GNB-hDS	GNB-hDS-iYJ	GNB-hDS-Cw	GNB-hDS-Cw-iYJ	GNB-hDS-Hw	GNB-hDS-Hw-iYJ
Datasets	$n = 20$ $k = 1$	$n = 20$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Entomology	51.51	51.48	57.38	53.61	**57.41**	53.71	48.63	52.82	50.08	50.02	50.41	51.59
Ichthyology	40.55	40.54	41.52	37.00	41.72	37.11	46.82	**49.72**	46.64	47.15	47.39	48.80
Insects-a-b	80.95	80.95	84.37	83.33	84.37	83.33	81.11	81.96	86.49	**86.97**	86.10	86.81
Insects-a-i	79.14	79.14	82.62	82.55	82.60	82.55	80.88	84.03	85.46	86.17	85.53	**86.90**
Insects-i-a-r-b	79.49	79.52	84.30	83.42	84.28	83.42	81.42	84.07	85.84	85.94	86.21	**86.59**
Insects-i-a-r-i	78.52	78.53	82.64	82.11	82.62	82.11	81.57	83.70	84.93	85.51	85.00	**86.22**
Insects-i-b	79.78	79.78	84.05	83.91	84.03	83.91	80.55	82.40	83.83	84.05	83.65	**84.44**
Insects-i-g-b	83.29	83.41	**88.02**	86.66	87.99	86.66	81.53	81.50	86.16	86.14	85.64	86.03
Insects-i-g-i	78.94	78.95	82.91	83.11	82.93	83.11	80.40	83.38	**86.93**	85.42	85.23	86.00
Insects-i-i	78.63	78.64	82.58	83.08	82.57	83.08	80.90	83.18	84.97	85.50	84.93	**86.14**
Insects-i-r-b	80.14	80.18	84.50	83.48	84.50	83.48	78.57	80.21	85.79	86.08	85.64	**86.18**
Insects-i-r-i	78.60	78.61	82.63	82.70	82.62	82.70	81.61	83.72	84.88	85.51	84.99	**86.25**
Insects-o-o-c	55.24	55.28	65.66	59.50	65.56	59.50	64.14	**69.38**	59.33	62.40	61.47	66.54
Instruments	**65.42**	65.06	55.04	56.53	55.59	56.68	48.31	49.48	40.06	36.91	45.93	42.36
Avg. hF	72.16	72.15	75.59	74.36	75.63	74.38	72.60	74.97	75.10	75.27	75.58	**76.49**
Avg. Ranking	10.32	9.96	5.39	6.96	5.71	6.75	9.50	6.07	5.36	4.18	5.07	**2.71**

The Local kNN-hDS and Global kNN-hDS methods obtained the lowest average hF rates and the lowest average rankings among all methods (10.32 and 9.96, respectively). Next, the GNB-hDS method obtained a ranking of 9.50. Most of the other methods obtained similar rankings between the fourth and seventh ranking. As an exception and highlight, the GNB-hDS-Hw-iYJ method achieved the best average hF rate (76.49%) and the best average ranking (2.71), reaching the best overall results in 7 out of 14 hierarchical data streams.

The overall hF results were submitted to a Friedman test, which identified a significant difference between the methods ($p\text{-}value = 7.15 \times 10^{-10}$). Thus, a *post hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 4.7 shows the resulting critical difference chart for the hF rates obtained by all methods.

The four adaptive variants of GNB-hDS plus both local and global kNC methods are significantly different from the Local kNN-hDS method. The Global kNN-hDS shows a similar set of differences, except that it does not significantly differ from the Global kNC-hDS. Also, the primary incremental GNB-hDS significantly differs from its adaptive counterparts that use the incremental Yeo-Johnson Power Transformation.

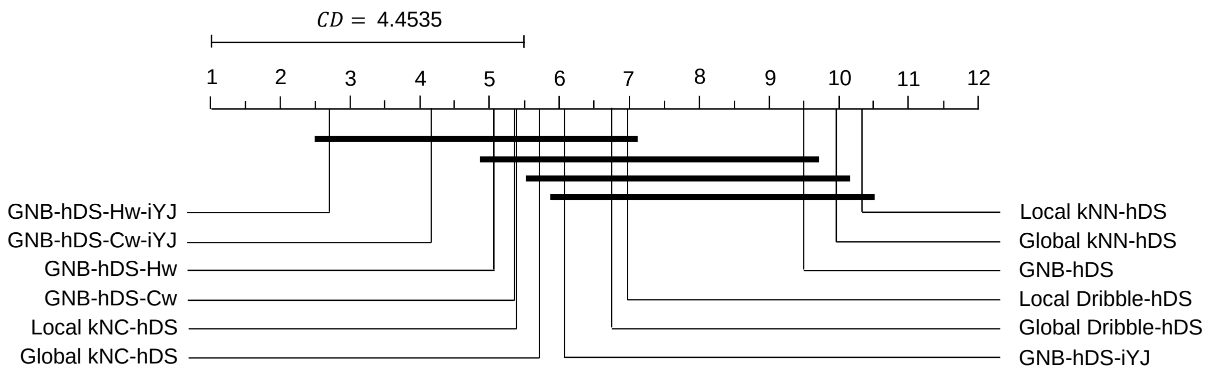


Figure 4.7 - Overall results: critical differences chart for hF rates obtained by methods with averaged best-performing parameters settings.

Complementarily to the overall results regarding hF rates, Table 4.20 depicts the overall $inst/s$ rates obtained by all proposed methods. Likewise, these results represent the $inst/s$ rates obtained by methods when using the averaged best-performing parameter settings.

Table 4.20 - Overall results: instances per second rates obtained by methods with averaged best-performing parameters settings.

	Local kNN-hDS	**Global kNN-hDS**	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**	**GNB-hDS**	**GNB-hDS-iYJ**	**GNB-hDS-Cw**	**GNB-hDS-Cw-iYJ**	**GNB-hDS-Hw**	**GNB-hDS-Hw-iYJ**
Datasets	$n = 20$ $k = 1$	$n = 20$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$	$n = 20$ $m = 10$ $k = 3$	$n = 5$ $m = 30$ $k = 1$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Entomology	127	209	134	327	200	382	380	395	**397**	370	335	287
Ichthyology	157	223	186	286	291	380	395	**426**	398	394	366	338
Insects-a-b	151	383	152	452	353	**543**	490	472	469	437	374	325
Insects-a-i	153	384	151	467	354	**542**	495	473	472	444	378	325
Insects-i-a-r-b	153	386	152	456	354	**541**	496	467	475	441	374	320
Insects-i-a-r-i	153	374	150	468	351	**542**	492	465	476	442	376	323
Insects-i-b	148	363	152	456	345	**541**	501	503	500	454	402	334
Insects-i-g-b	158	385	165	459	372	**542**	483	469	466	441	371	323
Insects-i-g-i	154	385	155	466	359	**543**	496	470	475	444	377	326
Insects-i-i	152	377	151	467	354	**545**	497	479	473	445	385	323
Insects-i-r-b	153	388	152	458	356	**543**	491	459	474	439	363	324
Insects-i-r-i	153	386	151	469	353	**540**	494	451	476	443	377	323
Insects-o-o-c	75	135	78	247	127	275	**283**	277	280	271	237	212
Instruments	79	110	121	221	164	256	263	**284**	257	263	259	217
Avg. $inst/s$	140.43	320.50	146.54	407.00	309.40	**479.60**	446.80	434.95	434.77	409.01	355.35	307.20
Avg. Ranking	11.57	8.00	11.36	5.50	9.00	**2.00**	2.29	3.07	3.21	5.43	7.21	9.36

Both Local kNN and kNC methods obtained the lowest average $inst/s$ rates and the lowest average rankings among all methods (11.57 and 11.36, respectively). Next, the GNB-hDS-Hw-iYJ method obtained a ranking of 9.36. On the other ranking side, GNB-hDS-iYJ and GNB-hDS-Cw obtained similar rankings of 3.21 and 3.07. Also, GNB-hDS and Global Dribble-hDS achieved similar rankings (2.29 and 2.00, respectively), with Global Dribble-hDS in first place in 10 out of 14 hierarchical data streams.

The overall $inst/s$ results were submitted to a Friedman test, which identified a significant difference between the methods ($p\text{-}value = 2.15 \times 10^{-24}$). Thus, a *post hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 4.8 shows the resulting critical difference chart for the $inst/s$ rates obtained by all methods.

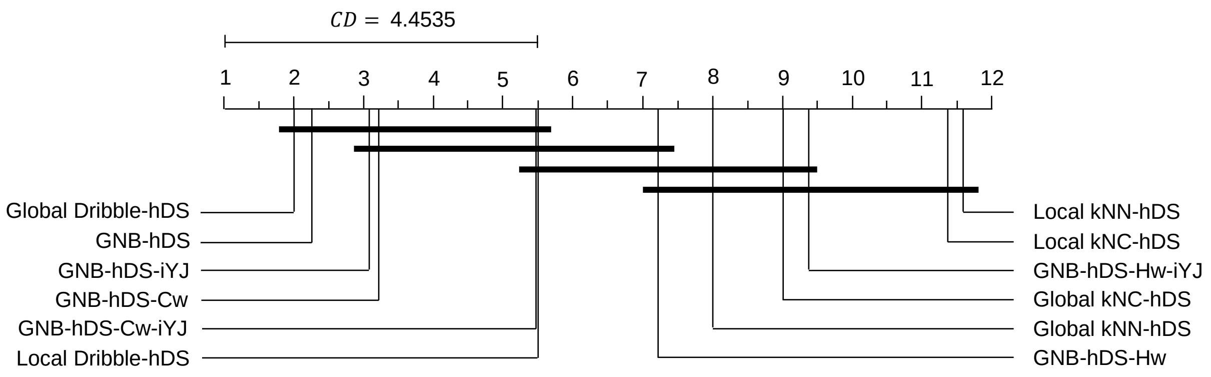


Figure 4.8 - Overall results: critical differences chart for $inst/s$ rates obtained by methods with averaged best-performing parameters settings.

Both Dribble methods plus four variants of GNB-hDS (incremental ones and both adaptive with current window (Cw)) are significantly different from the Local kNN-hDS method. Global Dribble-hDS, GNB-hDS, GNB-hDS-iYJ, and GNB-hDS-Cw are also significantly different from both kNN, both KNC, and GNB-hDS-Hw-iYJ methods.

Considering both hF and $inst/s$ rates together, one can observe that some methods obtained slightly opposite rankings in both analyses. For instance, GNB-hDS-Hw-iYJ achieved first place (ranking of 2.71) regarding hF rates, but the third to last place on $inst/s$ (ranking of 9.36). The same can be said about both kNC methods, with competitive hF rates, but slower $inst/s$ rates.

Note that, as already discussed, both Dribble methods take advantage of using a smaller data representation (n = 5) to obtain their averaged best performance. This ensured the first place for Global Dribble-hDS among the $inst/s$ performances and yet a competitive hF rate. A broader analysis in this sense regarding the averaged best performances is shown later in this section, with the application of the Multi-Criteria Decision Making (MCDM) technique.

Also in that context, the MCDM was applied to all performances of the classifiers in an isolated view to retrieve the best trade-off performances of all methods considering both hF and $inst/s$ rates together (see Section 4.1.2).

As previously described, the hF and $inst/s$ rates obtained by a method with each parameter setup were compared among each other concerning five different assignments of importance (weights, w) following the complementary percentage set $w \in \left\{\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\right\}$. In other words, each result set obtained with a specific parameter setup was evaluated with the MCDM-WPM analysis using complementary weighted hF and $inst/s$ rates (w_{hF} and $w_{inst/s}$) resulting in a WPM performance.

After that, the WPM performances were ranked among each other to obtain the best result set for each pair $(w_{hF}, w_{inst/s})$. Lastly, all sets of WPM rankings were averaged, and the best overall average was understood as the best trade-off performance of that method.

Table 4.21 depicts the hierarchical F-Measure obtained by all proposed methods in the hierarchical data stream sets considering the best trade-off performance of each method retrieved by the MCDM-WPM analysis (greater values per dataset highlighted in bold). Note that the table also depicts the parameter setup used by each method to achieve its best trade-off performance.

All kNN, kNC, and Dribble methods obtained their best trade-off performances with n = 5. This is probably because this parameter setup is the fastest possible with a reasonable representation of data. Also, note that the averaged best performances and the best trade-off performances of all GNB-hDS methods are the same since the w parameter does not affect the processing speed.

All GNB-hDS variants (except for the incremental GNB-hDS) obtained the best average hF rates and rankings regarding the trade-off performances. Next, kNC and Dribble methods obtained similar average rates. The incremental GNB-hDS obtained the third to last place, and both kNN obtained the lowest average rate among all methods, with the Local kNN-hDS method in the last position.

Table 4.21 - Overall results: Hierarchical F-Measure (%) rates obtained by methods on best trade-off performance.

	Local kNN-hDS	**Global kNN-hDS**	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**	**GNB-hDS**	**GNB-hDS-iYJ**	**GNB-hDS-Cw**	**GNB-hDS-Cw-iYJ**	**GNB-hDS-Hw**	**GNB-hDS-Hw-iYJ**
Datasets	$n=5$ $k=1$	$n=5$ $k=1$	$n=5$ $m=30$ $k=3$	$n=5$ $m=30$ $k=1$	$n=5$ $m=30$ $k=3$	$n=5$ $m=30$ $k=1$			$w=100$	$w=50$	$w=100$	$w=100$
Entomology	46.79	46.79	55.68	53.61	**55.71**	53.71	48.63	52.82	50.08	50.02	50.41	51.59
Ichthyology	36.66	36.66	39.44	37.00	39.53	37.11	46.82	**49.72**	46.64	47.15	47.39	48.80
Insects-a-b	77.63	77.64	84.05	83.33	84.07	83.33	81.11	81.96	86.49	**86.97**	86.10	86.81
Insects-a-i	75.32	75.32	82.29	82.55	82.30	82.55	80.88	84.03	85.46	86.17	85.53	**86.90**
Insects-i-a-r-b	76.89	76.96	84.00	83.42	83.96	83.42	81.42	84.07	85.84	85.94	86.21	**86.59**
Insects-i-a-r-i	74.98	75.00	82.31	82.11	82.29	82.11	81.57	83.70	84.93	85.51	85.00	**86.22**
Insects-i-b	76.43	76.42	83.62	83.91	83.62	83.91	80.55	82.40	83.83	84.05	83.65	**84.44**
Insects-i-g-b	79.97	80.12	87.61	86.66	**87.62**	86.66	81.53	81.50	86.16	86.14	85.64	86.03
Insects-i-g-i	75.62	75.65	82.22	83.11	82.20	83.11	80.40	83.38	**86.93**	85.42	85.23	86.00
Insects-i-i	75.01	75.03	82.40	83.08	82.40	83.08	80.90	83.18	84.97	85.50	84.93	**86.14**
Insects-i-r-b	77.33	77.42	84.33	83.48	84.29	83.48	78.57	80.21	85.79	86.08	85.64	**86.18**
Insects-i-r-i	75.05	75.06	82.31	82.70	82.31	82.70	81.61	83.72	84.88	85.51	84.99	**86.25**
Insects-o-o-c	49.99	50.21	65.00	59.50	64.90	59.50	64.14	**69.38**	59.33	62.40	61.47	66.54
Instruments	50.60	50.48	50.29	56.53	50.68	**56.68**	48.31	49.48	40.06	36.91	45.93	42.36
Avg. hF	67.73	67.77	74.68	74.36	74.71	74.38	72.60	74.97	75.10	75.27	75.58	**76.49**
Avg. Ranking	11.21	10.79	6.14	6.43	5.86	6.14	9.07	5.79	5.07	4.00	4.79	**2.71**

Both sides of the ranking (first and last positions) on both averaged best performances and best trade-off performances are the same. This occurs since the methods with the lowest trade-off performance rates were not able to obtain enough increases in hF rates even with larger data representations. Also, the primary version of GNB-hDS does not differ significantly from both kNN regarding hF rates, but it does differ regarding $inst/s$ rates (as discussed in its specific section).

Following the same protocol of the analysis performed with the averaged best performances, the overall hF results of the trade-off performances were submitted to a Friedman test. As expected, it identified a significant difference between the methods ($p\text{-}value = 1.49 \times 10^{-12}$). Thus, a *post hoc* Nemenyi test was applied to perform pairwise comparisons. Figure 4.9 shows the resulting critical difference chart for the hF rates obtained by all methods in the trade-off performances.

All the proposed methods obtained significantly higher rates than the related work Local kNN-hDS, except for methods Global kNN-hDS and GNB-hDS. Also, the incremental GNB-hDS significantly differs from its adaptive counterparts that use the incremental Yeo-Johnson Power Transformation.

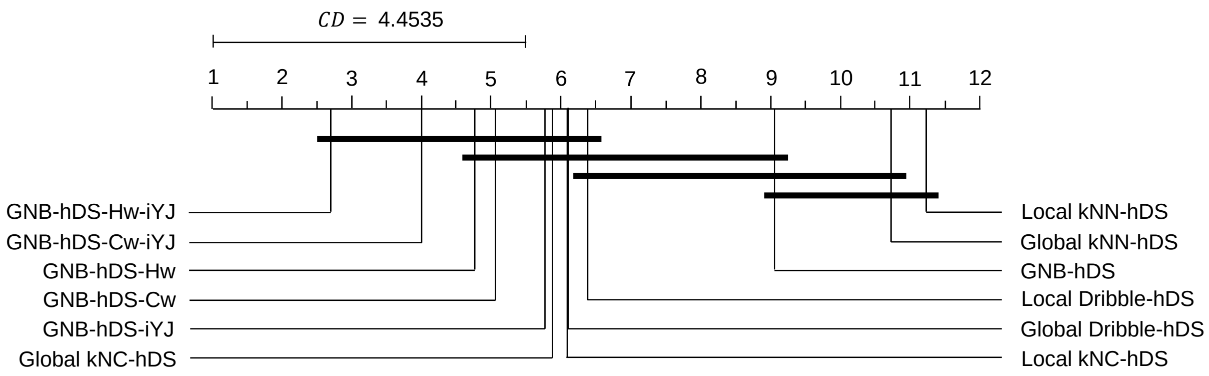


Figure 4.9 - Overall results: critical differences chart for hF rates obtained by methods on best trade-off performance.

Regarding speed comparison, Table 4.22 depicts the overall $inst/s$ rates obtained by all proposed methods on their best trade-off performances.

Table 4.22 - Overall results: instances per second rates obtained by methods on best trade-off performance.

	Local kNN-hDS	**Global kNN-hDS**	**Local kNC-hDS**	**Local Dribble-hDS**	**Global kNC-hDS**	**Global Dribble-hDS**	**GNB-hDS**	**GNB-hDS-iYJ**	**GNB-hDS-Cw**	**GNB-hDS-Cw-iYJ**	**GNB-hDS-Hw**	**GNB-hDS-Hw-iYJ**
Datasets	$n = 5$ $k = 1$	$n = 5$ $k = 1$	$n = 5$ $m = 30$ $k = 3$	$n = 5$ $m = 30$ $k = 1$	$n = 5$ $m = 30$ $k = 3$	$n = 5$ $m = 30$ $k = 1$			$w = 100$	$w = 50$	$w = 100$	$w = 100$
Entomology	283	370	265	327	335	382	380	395	**397**	370	335	287
Ichthyology	294	363	309	286	384	380	395	**426**	398	394	366	338
Insects-a-b	354	**568**	355	452	534	543	490	472	469	437	374	325
Insects-a-i	357	**570**	360	467	534	542	495	473	472	444	378	325
Insects-i-a-r-b	340	**583**	360	456	535	541	496	467	475	441	374	320
Insects-i-a-r-i	345	**582**	359	468	535	542	492	465	476	442	376	323
Insects-i-b	365	**584**	297	456	527	541	501	503	500	454	402	334
Insects-i-g-b	354	**579**	366	459	540	542	483	469	466	441	371	323
Insects-i-g-i	348	**581**	364	466	536	543	496	470	475	444	377	326
Insects-i-i	358	448	340	467	536	**544**	497	479	473	445	385	323
Insects-i-r-b	344	**576**	360	458	537	543	491	459	474	439	363	324
Insects-i-r-i	343	**581**	359	469	536	540	494	451	476	443	377	323
Insects-o-o-c	170	196	179	247	231	275	**283**	277	280	271	237	212
Instruments	178	215	187	221	230	256	263	**284**	257	263	259	217
Avg. $inst/s$	316.64	**485.54**	318.60	407.00	466.46	479.60	446.80	434.95	434.77	409.01	355.35	307.20
Avg. Ranking	11.00	3.50	10.50	7.29	4.07	**2.71**	3.64	4.43	4.57	6.93	8.21	11.14

Global kNN-hDS obtained the best average $inst/s$ rate (485.54), while Global Dribble-hDS obtained the best average ranking. Global kNN-hDS and GNB-hDS achieved close rankings, as well as Global kNC-hDS, GNB-hDS-iYJ and GNB-hDS-

Cw. The slower performances resulted from both local kNN and kNC methods and from the GNB-hDS-Hw-iYJ.

The $inst/s$ rates of the trade-off performances were also submitted to a Friedman test. As with the hF rates, the Friedman test identified a significant difference between the methods ($p\text{-}value = 1.40 \times 10^{-19}$). Figure 4.10 shows the resulting critical difference chart for the $inst/s$ rates obtained by all methods in the trade-off performances after the application of a *post hoc* Nemenyi test.

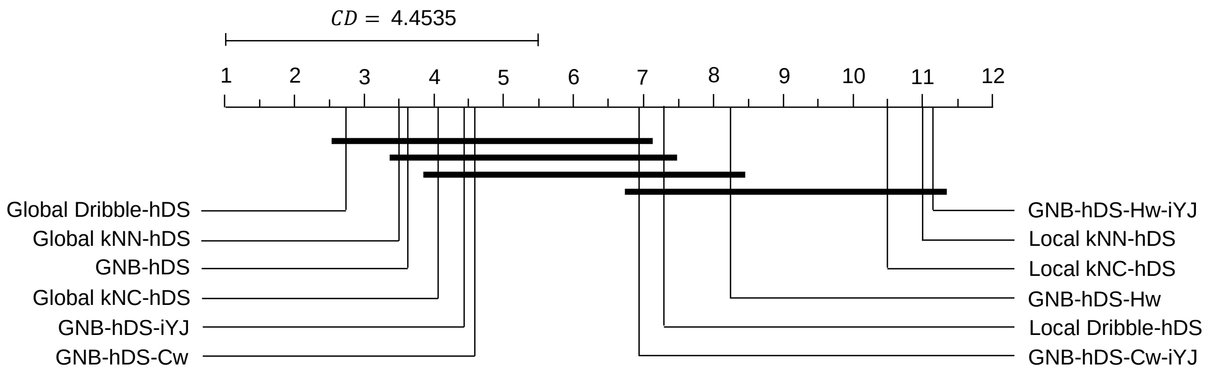


Figure 4.10 - Overall results: critical differences chart for $inst/s$ rates obtained by methods on best trade-off performance.

All global methods and both incremental GNB-hDS and GNB-hDS-Cw obtained significantly faster $inst/s$ rates than the related work Local kNN-hDS (and than Local kNC-hDS and GNB-hDS-Hw-iYJ). In the first and second places, Global Dribble-hDS and Global kNN-hDS significantly differ from Local kNN-hDS and Local kNC-hDS, and from both adaptive GNB-hDS variants with historical windows.

Attempting to portray an overview of both hF and $inst/s$ rates together, the MCDM-WPM analysis was also applied to the overall results of all methods together, considering both averaged best performance and trade-off performance.

First, regarding the averaged best performance, Table 4.23 details the WPM performances for each variation of the hF criterion weight (w_{hF}). Note that, as previously described, the $w_{inst/s}$ is the complementary percentage of w_{hF}. The last column shows the average WPM performance concerning all w variations. Also, note that zero with decimal places is a rounding from a constant equal to 1.00×10^{-6} and represents the minimum value obtained with the criterion normalization.

Table 4.23 - Overall results: MCDM-WPM (values) of methods with averaged best performance.

Method	w_{hF}					**Avg.**
	1/6	**2/6**	**3/6**	**4/6**	**5/6**	WPM
Local kNN-hDS	0.0000	0.0000	0.0000	0.0002	0.0006	0.0002
Global kNN-hDS	0.0590	0.0066	0.0007	0.0001	0.0000	0.0133
Local kNC-hDS	0.0338	0.0636	0.1194	0.2244	0.4216	0.1726
Local Dribble-hDS	0.7310	0.6799	0.6323	0.5881	0.5470	0.6357
Global kNC-hDS	0.5393	0.5838	0.6319	0.6841	0.7405	0.6359
Global Dribble-hDS	**0.8952**	**0.8014**	0.7174	0.6422	0.5749	0.7262
GNB-hDS	0.6308	0.4406	0.3077	0.2149	0.1501	0.3488
GNB-hDS-iYJ	0.8273	0.7883	0.7510	0.7156	0.6818	0.7528
GNB-hDS-Cw	0.8332	0.8000	**0.7681**	0.7375	0.7081	**0.7694**
GNB-hDS-Cw-iYJ	0.7792	0.7668	0.7546	0.7425	0.7307	0.7548
GNB-hDS-Hw	0.6575	0.6822	0.7078	0.7344	0.7619	0.7088
GNB-hDS-Hw-iYJ	0.5535	0.6230	0.7012	**0.7893**	**0.8884**	0.7111

Additionally, Table 4.24 shows the rankings corresponding to the values shown in Table 4.23 regarding each w variation. The last column shows the final overall average ranking of all methods when considering their averaged best performance.

Table 4.24 - Overall results: MCDM-WPM (rankings) of methods with averaged best performance.

Method	w_{hF}					**Avg.**
	1/6	**2/6**	**3/6**	**4/6**	**5/6**	**Ranking**
Local kNN-hDS	12.00	12.00	12.00	11.00	11.00	11.60
Global kNN-hDS	10.00	11.00	11.00	12.00	12.00	11.20
Local kNC-hDS	11.00	10.00	10.00	9.00	9.00	9.80
Local Dribble-hDS	5.00	6.00	7.00	8.00	8.00	6.80
Global kNC-hDS	9.00	8.00	8.00	6.00	3.00	6.80
Global Dribble-hDS	**1.00**	**1.00**	4.00	7.00	7.00	4.00
GNB-hDS	7.00	9.00	9.00	10.00	10.00	9.00
GNB-hDS-iYJ	3.00	3.00	3.00	5.00	6.00	4.00
GNB-hDS-Cw	2.00	2.00	**1.00**	3.00	5.00	**2.60**
GNB-hDS-Cw-iYJ	4.00	4.00	2.00	2.00	4.00	3.20
GNB-hDS-Hw	6.00	5.00	5.00	4.00	2.00	4.40
GNB-hDS-Hw-iYJ	8.00	7.00	6.00	**1.00**	**1.00**	4.60

The Local kNN-hDS method obtained the last combined ranking (11.60), followed by the proposed Global kNN-hDS method. Regardless of the weights given to hF and $inst/s$ rates, both methods did not perform well in any scenario as they have the lowest individual rates. On the other hand, the GNB-hDS-Cw method obtained the first combined ranking (2.60), obtaining the second place with smaller weights in the hF rate, the first place with equal weights in both rates, and the third and fifth place with greater weights on the hF rate.

The GNB-hDS-Cw-iYJ method obtained the second-best combined ranking (3.20), with a worse performance than the GNB-hDS-Cw method, mainly in the speed comparisons.

The Global Dribble-hDS and GNB-hDS-iYJ methods share the third-best combined ranking (4.00). The GNB-hDS-iYJ method proved to be competitive in all variations of w, and even obtained better rankings than the Global Dribble-hDS method with higher weights in w_{hF}. Furthermore, it is noteworthy that Global Dribble-hDS achieved the first place in the ranking when the weights were favoring the $inst/s$ rate.

Moreover, note that the GNB-hDS-Hw-iYJ method obtained only the sixth combined ranking (4.60), even though it obtained the first place in the ranking with weights favoring the hF rate, meaning that the method, despite having the best prediction correctness, could not obtain competitive processing times when compared to the other methods.

The same general MCDM-WPM analysis was applied to the overall results of the methods considering their best trade-off performance. Table 4.25 details the WPM performances for each variation of the hF criterion weight (w_{hF}). The last column shows the average WPM performance concerning all w variations.

Table 4.25 - Overall results: MCDM-WPM (values) of methods with best trade-off performance.

Method	w_{hF}					**Avg.**
	1/6	**2/6**	**3/6**	**4/6**	**5/6**	WPM
Local kNN-hDS	0.0086	0.0014	0.0002	0.0000	0.0000	0.0021
Global kNN-hDS	0.3984	0.1587	0.0632	0.0252	0.0100	0.1311
Local kNC-hDS	0.0973	0.1480	0.2252	0.3427	0.5215	0.2670
Local Dribble-hDS	0.5885	0.6188	0.6506	0.6841	0.7194	0.6523
Global kNC-hDS	0.8761	0.8596	0.8433	**0.8273**	0.8116	0.8436
Global Dribble-hDS	**0.9286**	**0.8919**	**0.8568**	0.8230	0.7905	**0.8581**
GNB-hDS	0.7394	0.6985	0.6598	0.6233	0.5887	0.6619
GNB-hDS-iYJ	0.7336	0.7512	0.7693	0.7878	0.8068	0.7698
GNB-hDS-Cw	0.7349	0.7551	0.7757	0.7970	**0.8188**	0.7763
GNB-hDS-Cw-iYJ	0.6113	0.6546	0.7010	0.7506	0.8037	0.7042
GNB-hDS-Hw	0.3298	0.4028	0.4919	0.6008	0.7338	0.5118
GNB-hDS-Hw-iYJ	0.0000	0.0001	0.0010	0.0100	0.1000	0.0222

As well, Table 4.26 shows the rankings corresponding to the values shown in Table 4.25 regarding each w variation. The last column shows the final overall average ranking of all methods when considering their best trade-off performances.

Table 4.26 - Overall results: MCDM-WPM (rankings) of methods with best trade-off performance.

Method	w_{hF}					**Avg. Ranking**
	1/6	**2/6**	**3/6**	**4/6**	**5/6**	
Local kNN-hDS	11.00	11.00	12.00	12.00	12.00	11.60
Global kNN-hDS	8.00	9.00	10.00	10.00	11.00	9.60
Local kNC-hDS	10.00	10.00	9.00	9.00	9.00	9.40
Local Dribble-hDS	7.00	7.00	7.00	6.00	7.00	6.80
Global kNC-hDS	2.00	2.00	2.00	**1.00**	2.00	**1.80**
Global Dribble-hDS	**1.00**	**1.00**	**1.00**	2.00	5.00	2.00
GNB-hDS	3.00	5.00	6.00	7.00	8.00	5.80
GNB-hDS-iYJ	5.00	4.00	4.00	4.00	3.00	4.00
GNB-hDS-Cw	4.00	3.00	3.00	3.00	**1.00**	2.80
GNB-hDS-Cw-iYJ	6.00	6.00	5.00	5.00	4.00	5.20
GNB-hDS-Hw	9.00	8.00	8.00	8.00	6.00	7.80
GNB-hDS-Hw-iYJ	12.00	12.00	11.00	11.00	10.00	11.20

As in the analysis concerning the best average performance of the methods, the Local kNN-hDS method also obtained the last place in the combined ranking considering the best trade-off performance. Even with smaller data representations (n = 5) and competitive $inst/s$ rates, the method could not maintain the hF rates obtained when using more data.

Next, the GNB-hDS-Hw-iYJ method obtained only the second-to-last ranking since it obtained the lowest overall $inst/s$ rate. Even with the best overall hF rate, the method did not achieve a good position in the combined ranking, as the gains in hF rates were not enough to offset the low $inst/s$ rates in the combined MCDM-WPM analysis.

On the other side of the combined ranking, Global kNC-hDS and Global Dribble-hDS methods obtained the first and second places, respectively (1.80 and 2.00). The Global kNC-hDS method achieved the first place with $w_{hF} = 4/6$, and the second place in the other variations of w. The Global Dribble-hDS method achieved the first place in the ranking when the weights were favoring the $inst/s$ rate and with equal weights in both rates.

Furthermore, when the weights were strongly favoring the hF rate, the GNB-hDS-Cw method obtained the first place in the ranking, resulting in a third place in the combined ranking (2.80).

Overall, regarding averaged best performance analysis, all GNB-hDS variants, plus the Global Dribble-hDS method, presented competitive results with each other, with the Global Dribble-hDS method obtaining higher processing speed rates, GNB-hDS-Hw-iYJ obtaining better prediction correctness rates, and the other methods in between, with GNB-hDS-Cw obtaining the best equally weighted performance.

Regarding best trade-off performance analysis, Global kNC-hDS and Global Dribble-hDS methods stand out, with Global Dribble-hDS obtaining higher processing speed rates and Global kNC-hDS obtaining better prediction correctness, together with the GNB-hDS-Cw method.

Also, it is important to point out that the analysis using the best trade-off performances presents less bias in the comparison of the methods, since the analysis using the best average performances intrinsically gives greater relevance to the hF rate in the selection of the parameter configuration to be considered in the MCDM analysis, even before weights are assigned to the hF and $inst/s$ criteria.

Finally, two key aspects related to the application of the described MCDM-WPM analysis are noteworthy. First, it should be understood only as a guide for interpreting the results of the methods when compared together. The application of the MCDM-WPM method with different criteria and different weight ranges can change the rank of the methods. In this thesis, the MCDM-WPM analysis was performed separately on each method (in order to remove the initial bias of the averaged best performances) and then in the methods together, normalizing the rates obtained by the methods. However, other MCDM protocols are possible and could result in the same rankings or not, and generate other interpretations.

Second, merging evaluation metrics is not straightforward nor trivial, and the resulting overall average ranking obtained in the MCDM-WPM analysis should not be understood as a single measure that can be used instead of individual ones. The best learning model may depend on several external traits not measured by the hF and $inst/s$ metrics computed in this thesis. It may also require specific solutions to specific problems which are not comprehended by this study, such as unusual data distributions, higher responsiveness to concept drifts, and different constraints regarding computational resources.

5 CONCLUSION

This section concludes this thesis by summarizing the fulfilled objectives and listing the resulting contributions. In addition, implications for upcoming works and experiments are provided to guide further research on the hierarchical data stream classification area.

5.1 FULFILLED OBJECTIVES AND CONTRIBUTIONS

In this thesis, learning models for classifying hierarchical data streams using data summarization techniques have been proposed and evaluated.

To fulfill the research objectives of this thesis and build a theoretical foundation, formal concepts of classification, hierarchical classification, and classification of data stream areas were presented (Section 2). In addition, this thesis presented a systematic literature review (Section 2.3) focused on the hierarchical data stream classification, corroborating the initial claims of lack of studies in the area and non-adherence of the few existent studies in the literature to the characteristics and constraints imposed by both foundation areas together.

Regarding the research aim of this thesis, four new main methods were proposed for the classification of hierarchical data streams: (i) “Global kNN-hDS” – based on nearest neighbors (Sections 3.2 and 4.2), (ii) “kNC-hDS” and (iii) “Dribble-hDS” – based on clustering techniques (Sections 3.3 and 4.3), and (vi) “GNB-hDS” – based on gaussian probabilities. Additionally, an incremental adaptation of the Yeo-Johnson Power Transformation was proposed to be used attached to a hierarchical data stream learning model as a data pre-processing step to reduce the skewness of the data and improve the prediction results (Sections 3.4 and 4.4).

This thesis also tried to provide a testbed for benchmarking the hierarchical data stream classification area. To this end, all hierarchical data stream sets available in the literature were identified, adapted, and arranged. Also, the proposed methods were experimented using different parameter settings and statistical validation and compared under the same experimental protocol (Sections 4.1 and 4.5).

Regarding the comparisons with the state-of-the-art method and the investigation of the hypothesis of this thesis, the Global kNN-hDS method is statistically faster than the Local kNN-hDS, while achieving similar prediction rates. These methods, based on nearest neighbors, presented high computational costs due to the distance computations used in their prediction strategies and performed on potentially complete data representation. Therefore, usually, there is a trade-off between prediction correctness and computational performance related to the possibility of bigger data representation.

However, as a response to this limitation (or characteristic) and corroborating the claims of the hypotheses, the methods based on clustering techniques and gaussian probabilities propose improvements in the representation of data using summarization techniques, allowing better computational performance without significant impacts on prediction correctness. Thus, the kNC-hDS, Dribble-hDS, and GNB-hDS methods showed the greatest gains in this relationship.

Local versions of kNC-hDS and Dribble-hDS obtained similar or better computational performance and higher prediction correctness, while global versions obtained simultaneously better prediction correctness and computational performance. Likewise, GNB-hDS could also obtain better computational performance without significant impacts on prediction correctness.

Therefore, all the proposed methods validated the initial expectation concerning the hypotheses of the thesis. Regarding prediction correctness and computational performance, all proposed learning models using summarization techniques could achieve better rates in one criterion without significant impacts on the other.

5.2 IMPLICATIONS FOR FUTURE WORK

Lastly, it is noteworthy that even though this thesis addressed the hierarchical data stream classification area and introduced novel learning models fitted to classify hierarchical data streams, the resulting contributions should be understood as an initial step in the research area. In this sense, some implications and proposals for future work in the area are briefly described below:

- The datasets used on the experimental protocol adopted in this thesis contain different features, instances, and domains, thus allowing a reasonable assessment of the behavior of the proposed methods in different scenarios. However, there is room for work that aims to create hierarchical data stream sets, detailing the underlying distribution of data, eventual concept drifts, and metrics concerning imbalanced data. Although the emergence of hierarchical data streams seems natural (given the evolution in data generation and storage mechanisms), creating labeled datasets fitted for the classification task is a quite expensive challenge, especially considering the difficulty of labeling large-scale data.

- The experimental protocol adopted in this thesis comprehended limited parameter settings and did not experiment variations of each parameter exhaustively. For instance, kNN and kNC methods seem to obtain better prediction correctness with bigger data representations while putting in jeopardy the computational performance. Thus, it is worth describing the relationship between these variables (data representation and prediction correctness) and eventually using this information to support the design of new methods or summarization techniques.
- Among the datasets used in the experimental protocol, the ones proposed in (SOUZA et al., 2020) were introduced with well-defined concept drifts along the stream. In future work, different time window strategies and the application of existing drift detectors (BARROS et al., 2017; BIFET; GAVALDA, 2007; FRÍAS-BLANCO et al., 2015) could be tested to increase the responsiveness to changes in the data distribution. For instance, pilot experiments with the adaptive variants of GNB-hDS showed that the size of the window significantly affects the number of instances used by the model to adapt itself to concept drifts compared to other variants.
- Finally, the next natural step in this research is the design of a learning model based on a different learning paradigm or method, such as decision trees. State-of-the-art learning models of the foundation areas – such as CLUS-HMC (VENS et al., 2008) on Hierarchical Classification, and Adaptive Random Forest (GOMES et al., 2017) on Data Stream Classification – must be accounted for, used as a concept idea for the designing of a novel decision tree-based method fitted to the hierarchical data stream classification task, and benchmarked against the methods proposed in this thesis to understand its behavior regarding prediction correctness and computational performance on this new classification task.

REFERENCES

AGGARWAL, Charu C. et al. A framework for on-demand classification of evolving data streams. **IEEE Transactions on Knowledge and Data Engineering**, v. 18, n. 5, p. 577-589, 2006.

AGGARWAL, Charu C. (Ed.). **Data streams: models and algorithms**. New York: Springer, 2007.

AHA, David W.; KIBLER, Dennis; ALBERT, Marc K. Instance-based learning algorithms. **Machine learning**, v. 6, n. 1, p. 37-66, 1991.

ALAZRAI, Rami; MOWAFI, Yaser; LEE, CS George. Anatomical-plane-based representation for human–human interactions analysis. **Pattern Recognition**, v. 48, n. 8, p. 2346-2363, 2015.

ALIZADEH, Ash A. et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. **Nature**, v. 403, n. 6769, p. 503-511, 2000.

ANDEREZ, Dario Ortega et al. A hierarchical approach towards activity recognition. In: **Proceedings of the 10th International Conference on Pervasive Technologies Related to Assistive Environments**. 2017. p. 269-274.

BABCOCK, Brian et al. Models and issues in data stream systems. In: **Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems**. 2002. p. 1-16.

BARDDAL, Jean Paul et al. On dynamic feature weighting for feature drifting data streams. In: **Joint european conference on machine learning and knowledge discovery in databases**. Springer, Cham, 2016. p. 129-144.

BARDDAL, Jean Paul et al. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. **Journal of Systems and Software**, v. 127, p. 278-294, 2017.

BARROS, Roberto SM et al. RDDM: Reactive drift detection method. **Expert Systems with Applications**, v. 90, p. 344-355, 2017.

BAHRI, Maroua et al. Data stream analysis: Foundations, major tasks and tools. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 11, n. 3, p. e1405, 2021.

BIFET, Albert et al. Moa: Massive online analysis, a framework for stream classification and clustering. In: **Proceedings of the first workshop on applications of pattern analysis**. PMLR, 2010. p. 44-50.

BIFET, Albert; GAVALDA, Ricard. Learning from time-changing data with adaptive windowing. In: **Proceedings of the 2007 SIAM international conference on data mining**. Society for Industrial and Applied Mathematics, 2007. p. 443-448.

BIFET, Albert; KIRKBY, Richard. Data stream mining a practical approach. 2009.

BISHOP, Christopher M.; NASRABADI, Nasser M. **Pattern recognition and machine learning**. New York: springer, 2006.

BOX, George EP; COX, David R. An analysis of transformations. **Journal of the Royal Statistical Society: Series B (Methodological)**, v. 26, n. 2, p. 211-243, 1964.

BRENT, Richard P. **Algorithms for minimization without derivatives**. Courier Corporation, 2013.

CAO, Liang et al. GCHAR: An efficient Group-based Context-Aware human activity recognition on smartphone. **Journal of Parallel and Distributed Computing**, v. 118, p. 67-80, 2018.

ÇELEN, Aydın. Comparative analysis of normalization procedures in TOPSIS method: with an application to Turkish deposit banking market. **Informatica**, v. 25, n. 2, p. 185-208, 2014.

CERRI, Ricardo et al. An extensive evaluation of decision tree-based hierarchical multilabel classification methods and performance measures. **Computational Intelligence**, v. 31, n. 1, p. 1-46, 2015.

CHAKROUN, Imen; HABER, Tom; ASHBY, Thomas J. SW-SGD: the sliding window stochastic gradient descent algorithm. **Procedia Computer Science**, v. 108, p. 2318-2322, 2017.

CHAN, Tony F.; GOLUB, Gene H.; LEVEQUE, Randall J. Algorithms for computing the sample variance: Analysis and recommendations. **The American Statistician**, v. 37, n. 3, p. 242-247, 1983.

CHAPELLE, Olivier; HAFFNER, Patrick; VAPNIK, Vladimir N. Support vector machines for histogram-based image classification. **IEEE transactions on Neural Networks**, v. 10, n. 5, p. 1055-1064, 1999.

CHAVEZ, Arturo Gomez et al. Automated species counting using a hierarchical classification approach with haar cascades and multi-descriptor random forests. In: **OCEANS 2016-Shanghai**. IEEE, 2016. p. 1-6.

CHEN, Zhide et al. Human continuous activity recognition based on energy-efficient schemes considering cloud security technology. **Security and Communication Networks**, v. 9, n. 16, p. 3585-3601, 2016.

CHOU, Pao-Hua; WU, Menq-Jiun; CHEN, Kuang-Ku. Integrating support vector machine and genetic algorithm to implement dynamic wafer quality prediction system. **Expert Systems with Applications**, v. 37, n. 6, p. 4413-4424, 2010.

D’HONDT, Eva et al. Dealing with temporal variation in patent categorization. **Information retrieval**, v. 17, n. 5, p. 520-544, 2014.

DANIEL, Wayne W.; CROSS, Chad L. **Biostatistics: a foundation for analysis in the health sciences**. Wiley, 2018.

DEFIYANTI, Sofi; WINARKO, Edi; PRIYANTA, Sigit. A survey of hierarchical classification algorithms with big-bang approach. In: **2019 5th International Conference on Science and Technology (ICST)**. IEEE, 2019. p. 1-6.

DEL CAMPO-AVILA, José et al. Improving the performance of an incremental algorithm driven by error margins. **Intelligent Data Analysis**, v. 12, n. 3, p. 305-318, 2008.

DEMŠAR, Janez. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine learning research**, v. 7, p. 1-30, 2006.

DJORGOVSKI, S. George et al. Flashes in a star stream: Automated classification of astronomical transient events. In: **2012 IEEE 8th International Conference on E-Science**. IEEE, 2012. p. 1-8.

DOMINGOS, Pedro; HULTEN, Geoff. Mining high-speed data streams. In: **Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining**. 2000. p. 71-80.

DUMAIS, Susan; CHEN, Hao. Hierarchical classification of web content. In: **Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval**. 2000. p. 256-263.

FAN, Jianping et al. Cost-sensitive learning of hierarchical tree classifiers for large-scale image classification and novel category detection. **Pattern Recognition**, v. 48, n. 5, p. 1673-1687, 2015.

FAYYAD, Usama M.; WEIR, Nicholas; DJORGOVSKI, S. Skicat: A machine learning system for automated cataloging of large scale sky surveys. In: **Machine Learning: Proceedings of the Tenth International Conference**. 1993. p. 112-119.

FREITAS, Alex; CARVALHO, André. A tutorial on hierarchical classification with applications in bioinformatics. **Research and trends in data mining technologies and applications**, p. 175-208, 2007.

FRÍAS-BLANCO, Isvani et al. Online and non-parametric drift detection methods based on Hoeffding's bounds. **IEEE Transactions on Knowledge and Data Engineering**, v. 27, n. 3, p. 810-823, 2014.

FRIEDMAN, Milton. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the american statistical association**, v. 32, n. 200, p. 675-701, 1937.

FRIEDMAN, Nir; GEIGER, Dan; GOLDSZMIDT, Moises. Bayesian network classifiers. **Machine learning**, v. 29, n. 2, p. 131-163, 1997.

GABER, Mohamed Medhat; ZASLAVSKY, Arkady; KRISHNASWAMY, Shonali. Mining data streams: a review. **ACM Sigmod Record**, v. 34, n. 2, p. 18-26, 2005.

GAMA, João et al. A survey on concept drift adaptation. **ACM computing surveys (CSUR)**, v. 46, n. 4, p. 1-37, 2014.

GAMA, João. **Knowledge discovery from data streams**. CRC Press, 2010.

GAMA, João; SEBASTIÃO, Raquel; RODRIGUES, Pedro Pereira. On evaluating stream learning algorithms. **Machine learning**, v. 90, n. 3, p. 317-346, 2013.

GOMES, Heitor Murilo. et al. Adaptive random forests for evolving data stream classification. **Machine Learning**, v. 106, n. 9, p. 1469-1495, 2017.

GOMES, Heitor Murilo et al. Machine learning for streaming data: state of the art, challenges, and opportunities. **ACM SIGKDD Explorations Newsletter**, v. 21, n. 2, p. 6-22, 2019.

GU, Ping. et al. An adaptive hierarchical model based on fusion of ontology and context. **Transactions of Beijing Institute of Technology**, v. 29, n. 10, p. 885-889, 2009.

HAMOONI, Hossein; MUEEN, Abdullah; NEEL, Amy. Phoneme sequence recognition via DTW-based classification. **Knowledge and Information Systems**, v. 48, n. 2, p. 253-275, 2016.

HAN, Jiawei; PEI, Jian; KAMBER, Micheline. **Data mining: concepts and techniques**. Elsevier, 2011.

HUANG, Jingchang et al. A crowdsource-based sensing system for monitoring fine-grained air quality in urban environments. **IEEE Internet of Things Journal**, v. 6, n. 2, p. 3240-3247, 2018.

HUANG, Kun-Yi et al. Speech emotion recognition using deep neural network considering verbal and nonverbal speech sounds. In: **ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. IEEE, 2019. p. 5866-5870.

JEONG, Sungmoon; LEE, Minho. Adaptive object recognition model using incremental feature representation and hierarchical classification. **Neural Networks**, v. 25, p. 130-140, 2012.

JIANG, Yue; CUKIC, Bojan; MENZIES, Tim. Can data transformation help in the detection of fault-prone modules?. In: **Proceedings of the 2008 workshop on Defects in large software systems**. 2008. p. 16-20.

KASAEI, S. Hamidreza et al. Interactive open-ended learning for 3d object recognition: An approach and experiments. **Journal of Intelligent & Robotic Systems**, v. 80, n. 3, p. 537-553, 2015.

KAUPPI, Jukka-Pekka; MARTIKAINEN, Kalle; RUOTSALAINEN, Ulla. Hierarchical classification of dynamically varying radar pulse repetition interval modulation patterns. **Neural Networks**, v. 23, n. 10, p. 1226-1237, 2010.

KHOWAJA, Sunder Ali et al. Contextual activity based Healthcare Internet of Things, Services, and People (HIoTSP): An architectural framework for healthcare monitoring using wearable sensors. **Computer Networks**, v. 145, p. 190-206, 2018.

KIRITCHENKO, Svetlana et al. Functional annotation of genes using hierarchical text categorization. In: **Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics**. 2005.

KORDA, Alexandra I. et al. Automatic identification of oculomotor behavior using pattern recognition techniques. **Computers in Biology and Medicine**, v. 60, p. 151-162, 2015.

KOSMOPOULOS, Aris et al. Evaluation measures for hierarchical classification: a unified view and novel approaches. **Data Mining and Knowledge Discovery**, v. 29, n. 3, p. 820-865, 2015.

KOTSAKIS, Rigas; KALLIRIS, George; DIMOULAS, Charalampos. Investigation of broadcast-audio semantic analysis scenarios employing radio-programme-adaptive pattern classification. **Speech Communication**, v. 54, n. 6, p. 743-762, 2012.

KOTSIANTIS, Sotiris B. et al. Supervised machine learning: A review of classification techniques. **Emerging artificial intelligence applications in computer engineering**, v. 160, n. 1, p. 3-24, 2007.

KREMPL, Georg et al. Open challenges for data stream mining research. **ACM SIGKDD explorations newsletter**, v. 16, n. 1, p. 1-10, 2014.

LA, Lei et al. Classifying XML data of semantic sensor networks. **Arabian Journal for Science and Engineering**, v. 39, n. 5, p. 3733-3745, 2014.

LIANG, Shangsong; REN, Zhaochun; DE RIJKE, Maarten. Fusion helps diversification. In: **Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval**. 2014. p. 303-312.

LIANG, Yongbo et al. Impact of data transformation: an ecg heartbeat classification approach. **Frontiers in digital health**, v. 2, p. 610956, 2020.

LIU, Jin et al. Fine-grained entity type classification with adaptive context. **Soft Computing**, v. 22, n. 13, p. 4307-4318, 2018.

LU, Yichun. **Concept hierarchy in data mining: Specification, generation and implementation**. 1997. Theses (School of Computing Science)/Simon Fraser University.

LUGHOFER, Edwin. On-line evolving image classifiers and their application to surface inspection. **Image and Vision Computing**, v. 28, n. 7, p. 1065-1079, 2010.

MARTIN, Trevor; SHEN, Yun; MAJIDIAN, Andrei. Soft concept hierarchies to summarise data streams and highlight anomalous changes. In: **International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems**. Springer, Berlin, Heidelberg, 2010. p. 44-54.

MERMILLOD, Martial; BUGAISKA, Aurélia; BONIN, Patrick. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. **Frontiers in psychology**, v. 4, p. 504, 2013.

NAIK, Azad; RANGWALA, Huzefa. **Large scale hierarchical classification: state of the art**. Springer International Publishing, 2018.

NAKHAEIZADEH, Gholamreza; SCHNABL, Alexander. Development of Multi-Criteria Metrics for Evaluation of Data Mining Algorithms. In: **KDD**. 1997. p. 37-42.

NEMENYI, Peter Bjorn. **Distribution-free multiple comparisons**. Princeton University, 1963.

NGUYEN, Hai-Long et al. Heterogeneous ensemble for feature drifts in data streams. In: **Pacific-Asia conference on knowledge discovery and data mining**. Springer, Berlin, Heidelberg, 2012. p. 1-12.

NGUYEN, Hai-Long; WOON, Yew-Kwong; NG, Wee-Keong. A survey on data stream clustering and classification. **Knowledge and information systems**, v. 45, n. 3, p. 535-569, 2015.

PARMEZAN, Antonio Rafael Sabino; SOUZA, Vinicius; BATISTA, Gustavo E.A.P.A. Towards hierarchical classification of data streams. In: **Iberoamerican Congress on Pattern Recognition**. Springer, Cham, 2018. p. 314-322.

PEIXOTO, Rafael; CRUZ, Christophe; SILVA, Nuno. Adaptive learning process for the evolution of ontology-described classification model in big data context. In: **2016 SAI Computing Conference (SAI)**. IEEE, 2016. p. 532-540.

PENG, Baolin et al. Trending sentiment-topic detection on twitter. In: **International conference on intelligent text processing and computational linguistics**. Springer, Cham, 2015. p. 66-77.

PESARANGHADER, Ali; VIKTOR, Herna; PAQUET, Eric. Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams. **Machine Learning**, v. 107, n. 11, p. 1711-1743, 2018.

PONTES, Edel Alexandre Silva. A brief historical overview of the Gaussian curve: from Abraham de Moivre to Johann Carl Friedrich Gauss. **International Journal of Engineering and Science Invention (IJESI)**, p. 28-34, 2018.

PRASAD, Bakshi Rohit; AGARWAL, Sonali. Stream data mining: platforms, algorithms, performance evaluators and research trends. **International journal of database theory and application**, v. 9, n. 9, p. 201-218, 2016.

PROTASOV, Stanislav et al. Using deep features for video scene detection and annotation. **Signal, Image and Video Processing**, v. 12, n. 5, p. 991-999, 2018.

PUERTO-SOUZA, Gustavo A. et al. Enhancing normal-abnormal classification accuracy in colonoscopy videos via temporal consistency. In: **Computer-Assisted and Robotic Endoscopy**. Springer, Cham, 2015. p. 129-139.

PUROHIT, Hemant et al. Identifying seekers and suppliers in social media communities to support crisis coordination. **Computer Supported Cooperative Work (CSCW)**, v. 23, n. 4, p. 513-545, 2014.

QUIÑONERO-CANDELA, Joaquin et al. (Ed.). **Dataset shift in machine learning**. Mit Press, 2008.

RAMÍREZ-GALLEGO, Sergio et al. A survey on data preprocessing for data stream mining: Current status and future directions. **Neurocomputing**, v. 239, p. 39-57, 2017.

RAZA, Mohsin et al. Diagnosis and monitoring of Alzheimer's patients using classical and deep learning techniques. **Expert Systems with Applications**, v. 136, p. 353-364, 2019.

READ, Jesse et al. Scalable and efficient multi-label classification for evolving data streams. **Machine Learning**, v. 88, n. 1, p. 243-272, 2012.

ROSS, Douglas T. et al. Systematic variation in gene expression patterns in human cancer cell lines. **Nature genetics**, v. 24, n. 3, p. 227-235, 2000.

SAGGESE, Alessia et al. Learning skeleton representations for human action recognition. **Pattern Recognition Letters**, v. 118, p. 23-31, 2019.

SAKIA, Remi M. The Box-Cox transformation technique: a review. **Journal of the Royal Statistical Society: Series D (The Statistician)**, v. 41, n. 2, p. 169-178, 1992.

SEBASTIANI, Fabrizio. Machine learning in automated text categorization. **ACM computing surveys (CSUR)**, v. 34, n. 1, p. 1-47, 2002.

SHAPIRO, Samuel Sanford; WILK, Martin B. An analysis of variance test for normality (complete samples). **Biometrika**, v. 52, n. 3/4, p. 591-611, 1965.

SHI, Hongbo et al. A method for classifying packets into network flows based on GHSOM. **Mobile Networks and Applications**, v. 17, n. 6, p. 730-739, 2012.

SHU, Kai et al. Fake news detection on social media: A data mining perspective. **ACM SIGKDD explorations newsletter**, v. 19, n. 1, p. 22-36, 2017.

SILLA, Carlos N.; FREITAS, Alex A. A survey of hierarchical classification across different application domains. **Data Mining and Knowledge Discovery**, v. 22, n. 1, p. 31-72, 2011.

SILVA-PALACIOS, Daniel; FERRI, Cesar; RAMIREZ-QUINTANA, M. Jose. Adapting hierarchical multiclass classification to changes in the target concept. In: **Conference of the Spanish Association for Artificial Intelligence**. Springer, Cham, 2018. p. 118-127.

SOKOLOVA, Marina; LAPALME, Guy. A systematic analysis of performance measures for classification tasks. **Information processing & management**, v. 45, n. 4, p. 427-437, 2009.

SONG, Yang; SAILER, Anca; SHAIKH, Hidayatullah. Problem classification method to enhance the ITIL incident and problem. In: **2009 IFIP/IEEE International Symposium on Integrated Network Management**. IEEE, 2009. p. 295-298.

SONG, Yang; SAILER, Anca; SHAIKH, Hidayatullah. Hierarchical online problem classification for IT support services. **IEEE Transactions on Services Computing**, v. 5, n. 3, p. 345-357, 2011.

SOUZA, Vinicius et al. Challenges in benchmarking stream learning algorithms with real-world data. **Data Mining and Knowledge Discovery**, v. 34, n. 6, p. 1805-1858, 2020.

STEINBACH, Michael; ERTÖZ, Levent; KUMAR, Vipin. The challenges of clustering high dimensional data. In: **New directions in statistical physics**. Springer, Berlin, Heidelberg, 2004. p. 273-309.

SUN, Bo et al. Affect recognition from facial movements and body gestures by hierarchical deep spatio-temporal features and fusion strategy. **Neural Networks**, v. 105, p. 36-51, 2018.

TARCA, Adi L. et al. Machine learning and its applications to biology. **PLoS computational biology**, v. 3, n. 6, p. e116, 2007.

TIEPPO, Eduardo et al. Hierarchical classification of data streams: a systematic literature review. **Artificial Intelligence Review**, p. 1-40, 2021.

TRIANTAPHYLLOU, Evangelos. Multi-criteria decision making methods. In: **Multi-criteria decision making methods: A comparative study**. Springer, Boston, MA, 2000. p. 5-21.

TSOUMAKAS, Grigorios; KATAKIS, Ioannis. Multi-label classification: An overview. **International Journal of Data Warehousing and Mining (IJDWM)**, v. 3, n. 3, p. 1-13, 2007.

TSYMBAL, Alexey. The problem of concept drift: definitions and related work. **Computer Science Department, Trinity College Dublin**, v. 106, n. 2, p. 58, 2004.

VENS, Celine et al. Decision trees for hierarchical multi-label classification. **Machine learning**, v. 73, n. 2, p. 185-214, 2008.

WANG, Yi; GONG, Zhiguo; GUO, Jingzhi. Hierarchical classification of business information on the web using incremental learning. In: **2009 IEEE International Conference on e-Business Engineering**. IEEE, 2009. p. 303-309.

WANKHADE, Kapil K.; DONGRE, Snehlata S.; JONDHALE, Kalpana C. Data stream classification: a review. **Iran Journal of Computer Science**, v. 3, n. 4, p. 239-260, 2020.

WEIGL, Eva et al. On improving performance of surface inspection systems by online active learning and flexible classifier updates. **Machine Vision and Applications**, v. 27, n. 1, p. 103-127, 2016.

WEST, D. H. D. Updating mean and variance estimates: An improved method. **Communications of the ACM**, v. 22, n. 9, p. 532-535, 1979.

WIDMER, Gerhard; KUBAT, Miroslav. Learning in the presence of concept drift and hidden contexts. **Machine learning**, v. 23, n. 1, p. 69-101, 1996.

WILCOXON, Frank. Individual comparisons by ranking methods. In: **Breakthroughs in statistics**. Springer, New York, NY, 1992. p. 196-202.

WU, Feihong; ZHANG, Jun; HONAVAR, Vasant. Learning classifiers using hierarchically structured class taxonomies. In: **International symposium on abstraction, reformulation, and approximation**. Springer, Berlin, Heidelberg, 2005. p. 313-320.

XIE, Lei et al. Pitch-density-based features and an SVM binary tree approach for multi-class audio classification in broadcast news. **Multimedia systems**, v. 17, n. 2, p. 101-112, 2011.

YASSIN, Nisreen IR et al. Machine learning techniques for breast cancer computer aided diagnosis using different image modalities: A systematic review. **Computer methods and programs in biomedicine**, v. 156, p. 25-45, 2018.

YEO, In-Kwon; JOHNSON, Richard A. A new family of power transformations to improve normality or symmetry. **Biometrika**, v. 87, n. 4, p. 954-959, 2000.

ZHANG, Min-Ling; ZHOU, Zhi-Hua. ML-KNN: A lazy learning approach to multi-label learning. **Pattern recognition**, v. 40, n. 7, p. 2038-2048, 2007.

ZHANG, Tian; RAMAKRISHNAN, Raghu; LIVNY, Miron. BIRCH: an efficient data clustering method for very large databases. **ACM sigmod record**, v. 25, n. 2, p. 103-114, 1996.

ZHANG, Tonglin; YANG, Baijian. Box-cox transformation in big data. **Technometrics**, v. 59, n. 2, p. 189-201, 2017.

ZIONTS, Stanley. MCDM - If not a roman numeral, then what?. **Interfaces**, v. 9, n. 4, p. 94-101, 1979.