University of Granada
Doctoral Program in Information and Communication Technologies

Doctoral Dissertation

LEARNING RULES IN DATA STREAM MINING: ALGORITHMS AND APPLICATIONS

Author
Elena Ruiz Sánchez

PhD Advisor
Jorge Casillas Barranquero

Granada, March 2021

Universidad de Granada

Programa de Doctorado en

Tecnologías de la Información y la Comunicación

Tesis Doctoral

LEARNING RULES IN DATA STREAM MINING: ALGORITHMS AND APPLICATIONS

Autora

Elena Ruiz Sánchez

Director

Jorge Casillas Barranquero

Granada, Marzo 2021

Editor: Universidad de Granada. Tesis Doctorales
Autor: Elena Ruiz Sánchez
ISBN: 978-84-1306-877-0
URI: http://hdl.handle.net/10481/68575

This doctoral thesis has been supported by the Spanish National Research Project TIN-2014-57251-P including the associated FPI scholarship BES-2015-073689 holded by the PhD Student.

Agradecimientos

Culmina para mí una etapa de aprendizaje con mayúsculas. Un aprendizaje que no se limita meramente al ámbito de investigación de esta tesis, sino que se ha desarrollado en multitud de sentidos simultáneamente. Durante estos años, he tenido la oportunidad de conocer desde dentro el mundo de la investigación, hacer mis pinitos como docente, colaborar con distintos profesores e investigadores de distintas universidades, disfrutar de estancias en el extranjero, embarcarme en proyectos que parecían pequeños y puntuales pero que a día de hoy siguen creciendo, entre otras cosas. Más allá del desarrollo profesional, ha sido sin duda una etapa de descubrimiento y crecimiento personal. Así, quiero dar las gracias a todas aquellas personas cuyo apoyo ha sido fundamental para que este viaje haya sido posible.

En primer lugar, a mis padres, estoy tentada de decir que por todo. Ambos son un ejemplo de esfuerzo, trabajo duro y constancia. Gracias por todo el cariño y apoyo que siempre me brindáis y por estar siempre ahí. Sabéis que sois un pilar esencial para mí.

A Jose, por el ánimo y apoyo diarios, por la paciencia y la comprensión, especialmente en estos últimos meses. Durante estos años, hemos tenido la oportunidad de apoyarnos mutuamente en nuestras respectivas andaduras predoctorales. Gracias por las risas y las distracciones que tanto ayudan a superar los días más complicados.

Como no podría ser de otra manera, gracias a mi director, Jorge, por el tiempo y la dedicación puesto en el desarrollo de esta tesis. Desde luego, que sin su conocimiento, ideas y supervisión, no hubiera sido posible. Asimismo, gracias al Dr. Leandro Minku por acogerme y permitir realizar una estancia bajo su supervisión en la Universidad de Leicester (Reino Unido).

Gracias a mis amigos, Clara, Natalia, Antonio, Rubén, Isaac y Rafa, por los muchos ratos de conversación y risas durante estos años. Mención especial aquí para Clara, buena amiga y compañera de piso durante varios años (confinamiento pandémico incluido), que siempre me demuestra su apoyo y empatía. También gracias a mis *compañeros de fatigas* del CITIC, Sergio, Jesús, Francisco, Jose, Jorge, entre otros. Por haber hecho mucho más ameno e interesante el camino.

Por último, me gustaría agradecer a mis compañeros de trabajo en PerkinElmer que me han facilitado mucho mi proceso de adaptación, permitiéndome compaginar estos últimos meses mi trabajo en la empresa con el trabajo en esta tesis.

Resumen

Los flujos de datos son secuencias infinitas de registros estructurados que llegan continuamente. La característica clave de estos sistemas es que los datos producidos por estos flujos no se almacenan de forma permanente sino que se procesan "sobre la marcha", es decir, cada dato se recibe, se procesa y finalmente es desechado, pudiendo así tratar con enormes cantidades de datos en tiempo real incluso con capacidades de almacenamiento y cómputo reducidas. Los flujos de datos permiten manejar fuentes de datos que generan continuamente información en orden cronológico y que superan las capacidades habituales de almacenamiento y procesamiento. Las aplicaciones reales de este problema crecen cada día, siendo frecuente su uso en telecomunicaciones, consumo energético, fisiología o redes sociales, entre otros.

La investigación en minería de flujo de datos ha estado centrada principalmente en clasificación y cambio de concepto (*concept drift*). Además, dentro del ámbito de clasificación se ha prestado interés principalmente a la precisión de los modelos, dejando de lado otros factores, como la legibilidad, que afectan a la utilidad de los métodos en entornos reales. Esta tendencia a priorizar ante todo la precisión de los modelos no es exclusiva de flujo de datos, sino que se enmarca dentro de la que había venido siendo la tendencia general en tareas predictivas de minería de datos. No obstante, en el caso de flujos de datos, las limitaciones de un enfoque únicamente centrado en la precisión predictiva son mayores.

En clasificación en flujos de datos, la distribución de clases subyacente puede variar a lo largo del flujo debido a posibles cambios de concepto. Por lo tanto, las propuestas de clasificación en flujos de datos asumen que todos los datos se continúan recibiendo etiquetados a lo largo del flujo. Si se pone el foco solo en la precisión, estamos continuamente entrenando un modelo que en el mejor de los casos será tan preciso como la entidad que está generando las etiquetas en primer lugar pero seguiremos sin poder prescindir de dicha entidad. Por esta razón, consideramos que el uso de modelos legibles y descriptivos que puedan aportar conocimiento adicional sobre el problema puede resultar más realista. En el caso de existir una variable dependiente clara, la precisión de estos modelos descriptivos sigue siendo importante puesto que evalúa si el modelo está siendo capaz de comprender adecuadamente el problema.

En esta tesis, se propone un algoritmo completamente online basado en el aprendizaje

de reglas para clasificación en flujos de datos, CLAST. El algoritmo aprende dinámicamente una población de reglas que conjuntamente representan la solución al problema. Las reglas son una forma legible de representación del conocimiento que representan relaciones entre variables y, en consecuencia, ofrecen la posibilidad de alcanzar un considerable nivel de detalle de interpretabilidad. Comparada con otros clasificadores de flujos de datos, la propuesta obtiene resultados muy competitivos en términos de precisión predictiva en los experimentos llevados a cabo.

En problemas reales con tasas de llegada muy altas e inmensos volúmenes de datos suele ser difícil encontrar datos que estén completamente etiquetados y estructurados. Por lo tanto, exploramos otros paradigmas de aprendizaje, distintos al supervisado, que permitan evitar la dependencia de la disponibilidad a tiempo de las etiquetas.

En esta línea, se realizan dos propuestas algorítmicas. La primera de ellas es Fuzzy-CSar-AFP; una propuesta de aprendizaje no supervisado para extracción directa de reglas de asociación en flujos de datos (*association stream mining*). Se trata de una propuesta online, que procesa los datos uno a uno en el momento de su llegada, y es capaz de construir y mantener directamente las reglas de asociación, sin necesidad de una etapa previa de identificación de *itemsets* frecuentes.

La última de las propuestas, PAST, consiste en un método semi-supervisado que extiende los dos enfoques anteriores al combinar la capacidad de extraer conocimiento del etiquetado de los datos con la capacidad para aprender de datos no etiquetados. En términos de precisión predictiva, el método presenta un buen rendimiento en los experimentos realizados; mejorando los resultados obtenidos utilizando solo datos etiquetados. Esto significa que el algoritmo es capaz de extraer conocimiento de los datos no etiquetados que le permite mejorar su compresión del problema.

Adicionalmente, se estudia la viabilidad de la extracción de reglas de asociación en flujos de datos en dos aplicaciones reales. La primera de las aplicaciones se basa en datos sobre uso del *smartphone*, mientras que en el segundo casos se explotan flujos de *tweets* de contenido político. En ambos casos, el análisis de las reglas de asociación generadas resulta muy útil para comprender lo que va ocurriendo a lo largo del tiempo, aportándonos un conocimiento que sería muy complicado obtener de otra manera.

Abstract

Data streams are infinite sequences of structured records that arrive continuously. The key feature of these systems is that the data produced by these streams are not stored permanently but are processed "on the fly", i.e., each sample is received, processed and finally cast aside, thus being able to deal with huge amounts of data in real time even with reduced storage and computational capacities. Data streams make it possible to handle data sources that continuously generate information in chronological order and that exceed the usual storage and processing capacities. The real applications of this problem are growing every day, with frequent use in telecommunications, energy consumption, physiology or social networks, among others.

Research in data stream mining has been mainly focused on classification and concept drift. Furthermore, within the field of classification, the main focus has been on model accuracy, leaving aside other factors, such as readability, that affect the usefulness of the methods in real-world environments. This tendency to prioritize model accuracy above all is not exclusive to data streams, but rather is part of what had been the general trend in predictive data mining tasks. However, in the case of data streams, the limitations of an approach focused solely on predictive accuracy are more significant.

In data stream classification, the underlying class distribution may vary along the stream due to possible concept changes. Therefore, proposals for classification in data streams assume that all data continues to be received labeled throughout the stream. If the emphasis is placed on accuracy alone, we are continuously training a model that at best will be as accurate as the entity that is generating the labels in the first place but we will still not avoid needing that entity. For this reason, we believe that the use of legible and descriptive models that can provide additional insight into the problem may be more realistic. In the case where there is a clear dependent variable, the accuracy of these descriptive models is still important as it assesses whether the model is being able to adequately understand the problem.

In this thesis, a fully online algorithm based on learning rules for classification in data streams, CLAST, is proposed. The algorithm dynamically learns a population of rules that together represent the solution to the problem. Rules are a legible knowledge representation form that represent relationships between variables and, consequently, offer the possibility of reaching a considerable level of interpretability detail. Compared to other data stream

classifiers, the proposal obtains very competitive results in the experiments carried out.

In real-world problems with very high arrival rates and immense volumes of data is often difficult to find data that are completely labeled and structured. Therefore, we explore other learning paradigms, besides supervised learning, that allow us to avoid dependence on timely available labels.

In this line, two algorithmic proposals are made. The first one is Fuzzy-CSar-AFP; an unsupervised learning proposal for direct extraction of association rules in data streams (association stream mining). It is an online proposal, which processes the data one by one at the time of arrival, and is able to directly build and maintain association rules, without the need for a previous stage of frequent itemset identification.

The last of the proposals, PAST, consists of a semi-supervised method that extends the two previous approaches by combining the ability to extract knowledge from the data labeling with the ability to learn from unlabeled data. In terms of predictive ability, the method presents a good performance in the experiments conducted; improving the results obtained using only labeled data. This means that the algorithm is able to extract knowledge from unlabeled data that allows it to improve its understanding of the problem.

Moreover, the viability of association rule extraction in data streams is studied in two real applications. The first application is based on smartphone usage data, while the second one exploits streams of tweets with political content. In both cases, the analysis of the generated association rules is very useful to understand what is happening over time, providing knowledge that would otherwise be very difficult to obtain.

Table of Contents

Chapter I

Introduction

I.1 Motivation

Today human activity is subject to a high degree of computerization where almost everything is recorded (or susceptible to be recorded) and sometimes stored and processed. Normally this information is stored exclusively for later consultation, although on many occasions it is also used to obtain models (data mining) that simplify the complex reality that exists in this information, generally with the aim of predicting behavior or trends. However, on other occasions the interest lies in monitoring the system to prevent situations, understand dynamics, support decision making, etc. The process of obtaining models, either for prediction or monitoring, is often difficult or impossible to perform because the amount of data to be analyzed is too large to be stored before processing. In addition, it is increasingly common to find data sources that continuously generate information in chronological order and that exceed the usual storage and processing capacities.

To address this problem, data streams can be handled, which are infinite sequences of structured records that arrive continuously (i.e., they are not transient data but flows of information that last over time) (Gama, 2010). The key feature of these systems is that the data produced by these streams are not stored permanently but are processed "on the fly", i.e., each data is analyzed, processed and finally forgotten, thus being able to deal with huge amounts of data in real time even with reduced storage and computation capacities. The real applications of this problem grow every day, being frequent the use in telecommunications, vehicle traffic, energy consumption, commerce, finance, physiology, robotics or social networks, among others.

Data streams pose new challenges for machine learning and data mining, since traditional methods have been designed for static datasets and are not capable of efficiently analyzing rapidly growing amounts of data and taking into account features such as:

- Limited computer resources such as memory and time, as well as tight needs to make predictions in a reasonable time.

- The phenomenon called concept drift (Gama et al., 2014), i.e., changes in the distribution of data that occur in the sequence over time. This could dramatically deteriorate the performance of the model used.
- Data can arrive at such a high rate in some applications that the labeling of all elements may be delayed or even impossible.

We can distinguish two ways to characterize this field. On the one hand, we can attend to the models (such as sampling, load shedding, sketching, synopsis, or aggregation) and computational techniques (approximate algorithms, sliding window or algorithm output granularity) used to manage data streams (Gama, 2010).

On the other hand, depending on the learning problem to be solved (Gama, 2010; Sayed-Mouchaweh and Lughofer, 2012), it is intended to address it through automatic learning algorithms, either by adapting them to the characteristics of this type of data or by developing new approaches capable of better managing this information (Lughofer, 2011). The main problems addressed in this case can be divided into three:

- classification (supervised learning; the existence of data streams labeled with a class is assumed (Orriols-Puig et al., 2008b) and algorithms are studied to not only maximize the success rate but also to react and adapt to changing situations—concept drift (Scholz and Klinkenberg, 2007; Orriols-Puig and Casillas, 2011)—),
- clustering (Guha et al., 2000; Silva et al., 2013) (unsupervised learning; it is a problem widely studied in data mining literature, however, it is more difficult to adapt the clustering to data flows due to the limitations of being able to analyze each data only once) and
- frequent patterns (unsupervised learning; sets of frequent elements are maintained incrementally as data stream is received).

Of these problems, classification (and concept drift) is the most studied in the specialized literature in recent years. Thus, we can find a considerable number of proposals based on different types of models, such as decision trees (Domingos and Hulten, 2000; Hulten et al., 2001; Bifet and Gavaldà, 2009a), k-Nearest Neighbors (Zhang et al., 2011; Losing et al., 2016) or Support Vector Machines (Bordes et al., 2005; Rai et al., 2009). Nevertheless, ensembles have become the most widely studied classifiers in the data stream research field in the last few years (Krawczyk et al., 2017).

However, the main research focus has been on prediction accuracy, disregarding other factors, such as readability, that affect the real usefulness of the methods. It is worth mentioning that this phenomenon is not exclusive to data streams but has been a general trend within the machine learning models proposed for all types of predictive tasks. Indeed, research areas, such as eXplainable Artificial Intelligence (XAI) (Gunning, 2017), that try to revert this trend are gaining increasing attention. Nevertheless, in a

data stream context, the usefulness of generating highly accurate black box models (or with very limited interpretability) may be especially questionable.

One of the characteristics of data streams is that the underlying class distribution is likely to undergo changes over time. Therefore, data stream classification methods expect all data to arrive labeled over the entire duration of the stream. The trained models aim to be at best as accurate as the entity that is providing the labels in the first place despite that such entity will continue to be needed. For this reason, we believe that the use of legible and descriptive models (e.g., rule-based models) that can provide additional insight into the problem may be more realistic.

Moreover, having structured and class-labeled data at all times is difficult in many real data stream environments. This results in a shortage of real problems that can be used for experimentation. Although it should be noted that this limitation is circumscribed to supervised learning for classification, in the case of regression, much less studied in the data stream mining literature, it is common that the dependent variable to be predicted is the result of measuring a natural phenomenon (e.g., energy consumption) where the real-time approach is realistic. This is also applicable to classification problems that really come from simplifications of regression problems; for example, energy consumption could be labeled as high and low, and this is done in one of the most commonly used data sets in the field, the airlines problem (Bifet et al., 2010a) of flight delay, where the dependent variable is really continuous —time of delay—and from there the class is extracted according to whether or not a threshold is exceeded.

Furthermore, learning as the data comes one by one can entail an efficiency gain before data sets with immense instances. While its usefulness is unquestionable in the era of big data, this is not strictly speaking data stream mining, but incremental learning (online approach to a problem that is actually off-line). In those classification cases where the predictive model does not have much relevance, generating descriptive models with good interpretability that allow monitoring the system can be more useful.

In contrast, unsupervised or semi-supervised learning, although less studied in the literature, can be more directly applicable to real data stream problems. In unsupervised learning, we work directly with the information that flows structured in variables, without the need to have a system that previously labels all the training data.

Within unsupervised learning, incremental clustering has experienced a significant development (Guha et al., 2000; Silva et al., 2013). Despite this, the knowledge that is extracted (segmentation) is often insufficient to support decision making on real problems. The aggregations are difficult to understand, they are not self-explanatory, but require a calm "manual" analysis of the segmentation generated in order to draw useful conclusions on the problem under study. This is complicated in dynamic problems that can fluctuate rapidly.

Frequent pattern and association rule discovery is an ideal way to address many data stream problems where the aim is to monitor (not predict) the system using meaningful, readable and simple independent models. The models can be highly readable themselves,

as they are expressed with a syntax easily understandable by human.

A real case that well reflects the type of problems to be addressed would be that of detecting potential attacks on websites and computer networks (Corral et al., 2011). In this scenario, there is a set of characteristics that indicate suspicious acts on the network (e.g., strange characters in login interfaces, port access, unusual traffic flows, etc.) by malicious users trying to identify system vulnerabilities in order to take possession of it. Thus, instead of dealing with the problem in a traditional way based on supervised learning with labeled examples, the system is continuously monitored, dynamically obtaining models that explain the system situation and help humans to detect possible threats, with the advantage of adapting to previously unknown network attack techniques.

In these problems, the approaches proposed so far have been based on providing solutions such as heavy hitters (finding elements that exceed a frequency threshold, i.e., it would be the most basic case with a single item) (Karp et al., 2003; Cormode et al., 2008) or obtaining sets of frequent items by means of sliding windows (Yang et al., 2007; Tan et al., 2010; Wang and Chen, 2011; Patnaik et al., 2013). Anyway, the main limitation of the current state of the art in pattern learning from data streams lies in the difficulty to extract rules that define causal relationships, which would be much more powerful and useful. This is due to the impossibility of applying the traditional two-phase approach used in static data, where first frequent objects are obtained and then rules are extracted by analyzing the reliability of the causal relationship as a function of the data set, something impossible when the data arrive in flow fashion. However, there are a few proposals based on maintaining association rules and updating them as data is received (Fan et al., 2009; Chen et al., 2010; Tan et al., 2010), but they are very inefficient and impractical in problems with relatively high data influx.

On the other hand, semi-supervised learning extends both unsupervised and supervised learning by incorporating additional information characteristic of the other learning paradigm. It allows to take advantage of both unlabeled data, which would be discarded by a supervised approach, and labels, with which unsupervised methods are not prepared to deal. Hence, semi-supervised learning has been proven useful to address problems where there is a target variable of interest and only a small amount of labels are available compared to a greater proportion of unlabeled data. Data labeling can be costly due to different reasons, for example, due to the combination of a "manual" labeling process and a large amount of data to be labeled.

As it has been already mentioned, the characteristics of data streams often make it difficult to maintain a high percentage of labeled data. For instance, consider a center for bank fraud detection which daily reviews credit card transactions to identify which are fraudulent. To build an automatic fraud-detection predictive model is key that banking experts provide a certain number of fraudulent transactions as training (labeled) examples. The labeling ability of the experts is limited and probably no bank is willing to pay experts for manually labeling every single transaction, provided that thousands of transactions may be received on a daily basis. Thus, interest in semi-supervised learning approaches for

data streams has increased over the past few years (Masud et al., 2012; Haque et al., 2016; Feng et al., 2016; Wang and Li, 2018). The different alternatives mentioned for addressing data stream mining problems are summarized in Figure I.1.

I.2 Objectives

The objective of this PhD dissertation is to address some of the gaps identified in the literature on data stream mining. Research efforts in data stream mining have been highly focused on classification and, within this field, priority has been given to designing and developing algorithms that are as accurate as possible over proposing legible models. This presents certain limitations on real utility and applicability, as discussed in the previous section. In this thesis, we aim to research on descriptive models that are able to dynamically adapt to the data and to provide interesting knowledge about what is happening in the system. In this sense, we will study the use of rule learning models in data stream classification, but we also consider important to explore other learning paradigms, such as unsupervised and semi-supervised learning, which we believe can better fit the real conditions of data streams. Thus, the main objective of the thesis can be divided into the following research lines or subobjectives:

1. **Interpretability in data stream classification models**. The development of a fully online classifier for data streams, which employs a descriptive knowledge representation based on rules, is proposed. The algorithm should be able to dynamically evolve a set of rules as new labeled training data are received. The use of fuzzy sets to deal with continuous variables will be explored, along with the idea of using an evolutionary algorithm for rule discovery. Furthermore, we will try to minimize the number of configuration parameters of the proposal to facilitate its use. In this sense, the use of a Hoeffding bound (Hulten et al., 2001; Hoeffding, 1994) will be studied.

2. **Association stream mining**. We will work on the improvement of the Fuzzy-CSar algorithm (Sancho-Asensio et al., 2016) with the design of an advanced version oriented to increase the diversity of the obtained set of rules, facilitate its use and improve its capacity of adaptation to the particularities of each problem. Furthermore, in this context, issues such as the difficulty of a fair assessment in association rules mining and the importance of result interpretability will be addressed. Using evaluation measures based only on the quantity of quality rules is in many cases insufficient. There may be a high percentage of rules that are very similar to each other, which in practice is not very useful for helping the expert decision making. New measures and representations of the quality of the results, taking into account the heterogeneity of the set of rules, will be proposed. In addition, an adequate graphic representation of the results can facilitate their understanding. Different types of visualizations will be studied to help the analysis of the results of association rule mining, in general, and of association stream mining, in particular.

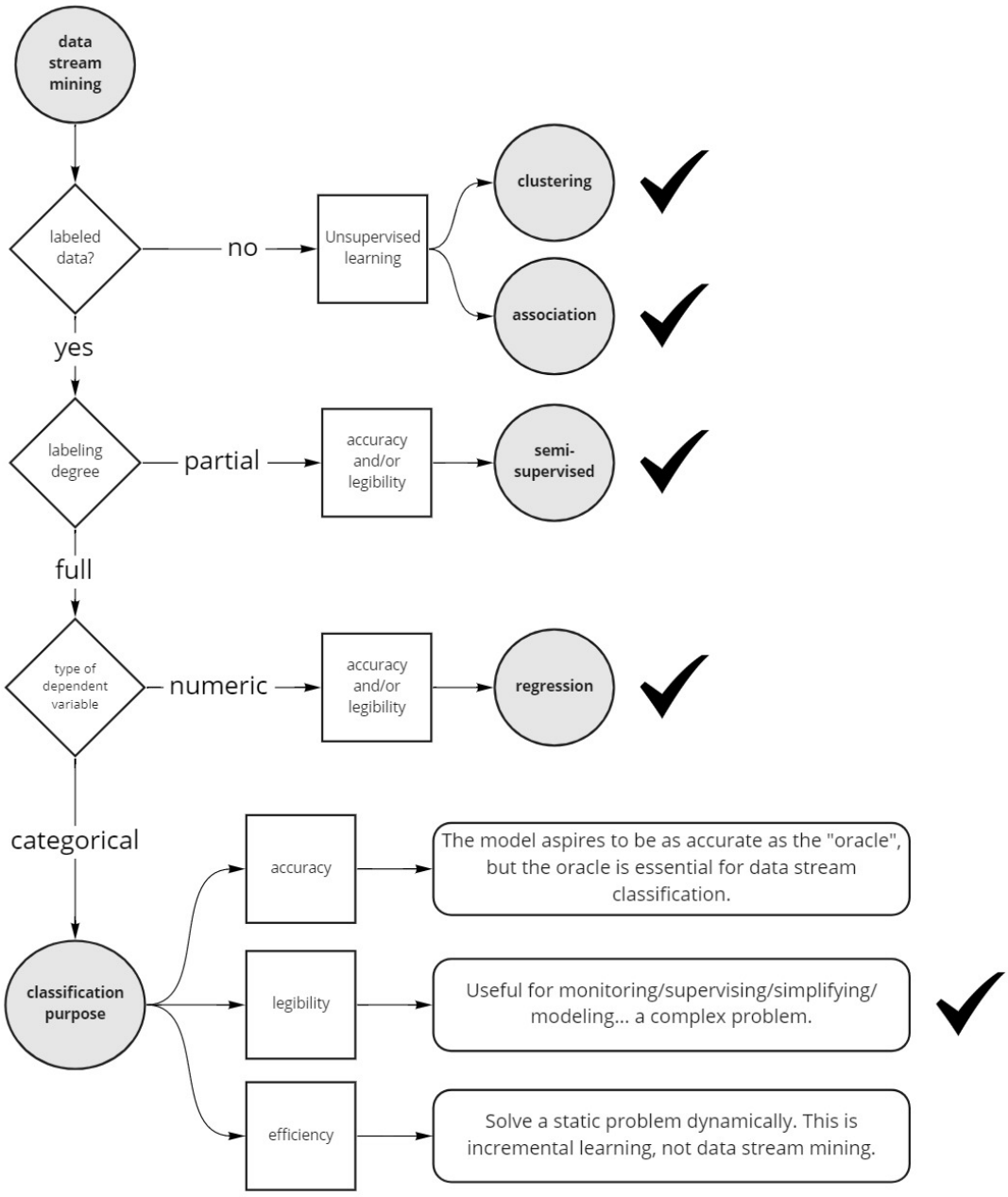


Figure I.1: Usefulness of the different approaches in data stream mining.

3. **Semi-supervised classification in data streams**. The characteristics of data streams often make difficult to get high percentages of labeled training data. The objective is to develop a semi-supervised method for data streams able to learn from both labeled and unlabeled data to generate an accurate and descriptive model. When the amount of available labels is low, the method should be able to improve its performance thanks to the use of unlabeled instances.

4. **Application of the developed algorithms in real-world problems**. The aim is to address real applications, with original data, in which large volumes of data are received as a chronologically ordered sequence with high arrival rate, and use descriptive online models to monitor what is happening in the system.

I.3 Structure

The remaining of this PhD dissertation is organized as follows. Chapter II reviews the background of the field describing the main research studies in the data stream mining problems addressed. The following three chapters are devoted to presenting the algorithmic proposals and their results. Thus, Chapter III describes a supervised learning proposal for classification problems in data streams. Chapter IV is focused on an association stream mining proposal aimed at maintaining a dynamic and interpretable model capable of explaining at any time what is happening. Chapter V introduces a semi-supervised learning approach. This proposal is an adaptation of the supervised proposal presented in Chapter III to environments where there is a dearth of labels. Furthermore, Chapter VI provides examples of original real applications of some of the proposed techniques. Finally, Chapter VII summarizes the main conclusions reached and discusses possible lines of future work.

Chapter II

Related work

II.1 Classification in data streams

Classification is one of the most widespread data mining techniques, and the most important case of supervised learning. Hundreds of different classifiers have been developed that belong to a wide number of paradigms, such as divide-and-conquer methods, rule learners, lazy learners, kernel methods, graphical models, etc. Decision trees are one of the most popular classification techniques since they are interpretable models that can be visualized graphically. Nonetheless traditional machine learning methods have been designed for static data sets and are not capable of efficiently analyzing rapidly growing amounts of data. Thus, classical decision tree learning algorithms like CART (Breiman et al., 1984), ID3 (Quinlan, 1986) or C4.5 (Quinlan, 2014) assume that all training examples can be stored simultaneously in the main memory and are therefore very limited in the number of examples they can learn from. And, in particular, they are not applicable to data streams, where potentially the number of examples can be infinite.

The main problem of building a decision tree in data stream setting, where not all the data can be stored, is the necessity of reusing training examples to calculate the best splitting attributes. Domingos and Hulten (2000) addressed this problem with the development of the Hoeffding Tree, an incremental decision tree algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. The key of the Hoeffding Tree is the use of Hoeffding bound to assess the sample size needed to estimate a variable with sufficient precision (for example, the information gain). This guarantees that the classifier is independent from the sample size. Thus, without keeping the full data stream in memory, the algorithm is able to ensure its output is asymptotically nearly identical to that of a non-incremental learner using infinitely many examples (as shown by the authors in (Domingos and Hulten, 2000)). The Hoeffding Tree algorithm maintains in each node the statistics needed for splitting attributes. A majority class strategy is used to classify the examples at the leaves. VFDT (Very Fast Decision Tree) is the implementation of the Hoeffding Tree algorithm, with

some practical improvements added, described in (Domingos and Hulten, 2000). Several methods extending VFDT have been proposed over the years.

A key feature of the data stream model is that the streams evolve over time, and the algorithms must adapt to the changes. Change management strategies can be broadly grouped into three categories, or a combination of them. They can use adaptive estimators for relevant statistics, and then an algorithm that keeps a model in sync with those statistics. They can create models that are adapted or reconstructed when a change detector indicates that a change has occurred. They can be ensemble methods, which maintain dynamic populations of models. The first strategy relies on the fact that many model builders monitor a set of statistics from the stream and then combine them into a model. This strategy works by having a dynamic estimator for each relevant statistic in a way that reflects its current value, and letting the model builder feed on those estimators. The simplest estimator algorithm for the expected value is the linear estimator, which simply returns the average of the data items contained in the *Memory*. The *Memory* can be, for instance, a sliding window that stores the most recent W items received.

In the second strategy, one or more change detection algorithms run in parallel with the main model-building algorithm. When they detect a significant change in the stream, a revision algorithm is activated. A particular case of this strategy is when the change is detected not by observing the incoming stream but observing the performance of the model, for example, watching for decreases in the accuracy of a predictor. In the third strategy, the responsibility for detecting and reacting to change lies mainly with the ensemble manager, although the individual models may have this capability as well.

Two of the most popular change detectors are the Drift Detection Method (DDM) (Gama et al., 2004) and ADaptive sliding WINdow (ADWIN) (Bifet, 2010; Bifet and Gavaldà, 2007). DDM is applicable in the context of predictive models. It monitors the number of errors produced by a model learned on the previous stream items. Typically, the error of the model should decrease or remain stable as the amount of data used increases, provided that the learning method controls overfitting and that the data and class distribution is stationary. Therefore, if, instead, DDM observes the prediction error increasing, it interprets this as evidence that change has happened. This approach is generic and simple to use, but it may be sometimes too slow in responding to changes. Moreover, for slow change, the number of instances kept in memory may get large. ADWIN is a change detector and estimation algorithm based on exponential histograms (Datar et al., 2002). It resolves the trade-off between reacting quickly to changes and having few false alarms by checking change at many scales simultaneously. There is no need for the user to guess at which frequency the change will occur or how large the deviation should be to trigger an alarm. The use of exponential histograms allows this to be done more efficiently in time and memory than by brute force. On the other hand, it is computationally more expensive (in time and memory) than simpler methods, so it should be used when the scale of change is unknown and this may be problematic.

In theory, Hoeffding trees are able to adapt to some extent to concept drift. Leaves

that would no longer grow in a stationary context may begin to grow again if evidence is detected that such growth would improve accuracy. However, this is often too slow a process in practice. Concept-adapting Very Fast Decision Tree algorithm (CVFDT) was presented in Hulten et al. (2001) as an extension of VFDT to deal with continuously-changing data streams. The algorithm maintains a model that is consistent with the examples included in a sliding window. An alternative subtree is grown whenever an old one becomes questionable, and the old subtree is replaced by the new one when the new gets to be more accurate. The model learnt by CVFDT is similar in accuracy to the one that would be learnt if VFDT were reapplied to a moving window of examples every time a new example arrives, but with O(1) complexity per instance. Unfortunately, the trees built by CVFDT do not have the same theoretical guarantees of Hoeffding trees.

Bifet and Gavaldà (2009a) introduced the Hoeffding Adaptive Tree (HAT), an adaptive extension to the Hoeffding Tree that uses ADWIN as change detector and error estimator. As opposed to CVFDT, HAT has theoretical guarantees of performance and requires no parameters related to change control. CVFDT requires parameters as the example window size, or the numbers of examples that are used to build or to test the accuracy of the alternate tree. Choosing these parameters implies preconceptions on how fast or how often the data are going to evolve. Since we could make the wrong choices or, even more, the stream may experience a combination of different types of changes (making any fixed choice wrong), HAT is aimed at adapting to the scale of change in the data, instead of relying on a priori assumptions made by the user. In addition, ADWIN has rigorous guarantees of performance that can be transferred to decision tree learners as follows: if a change is followed by a long enough stable period, the classification error of the learner will tend, and the same rate, to the error rate of VFDT.

Most strategies for dealing with time change contain hardwired constants, or else require input parameters, concerning the expected speed or frequency of the change; some examples are a priori definitions of sliding window lengths, values of decay or forgetting parameters, explicit bounds on maximum drift, etc. These choices represent preconceptions on how fast or how often the data are going to evolve and, of course, they may be completely wrong. Even more, no fixed choice may be right, since the stream may experience any combination of abrupt changes, gradual ones, and long stationary periods.

Other proposals have aimed to improve VFDT classification performance via the use of Naive Bayes learners at the leaves instead of majority class classifier. Gama et al. (2003) introduced VFDT_{C} that extends the Hoeffding Tree to handle numeric attributes and concept drifts, and incorporates the Naive Bayes learners at the leaves for prediction. VFDT_{C} handles numeric values using binary trees and uses DDM as change detector. The trees contain a Naive Bayes classifier at each node. Nonetheless, Holmes et al. (2005) identified situations where the standard VFDT is overtaken initially but eventually outperforms the alternative using Naïve Bayes at the leaves, and proposed a hybrid adaptive method: when making a prediction for a test example, the leaf will return the prediction of Naïve Bayes or the majority class classifier depending on which was more accurate overall. The only overhead needed is maintaining the two counts for the number

of times each method has been right.

Despite Hoeffding inequality has been widely used for estimating measures like information gain or the Gini index, these measures cannot be expressed as a sum of independent random variables and, therefore, Rutkowski et al. (2012) argue Hoeffding bound to be the wrong tool. They proposed the use of the McDiarmid inequality, a generalization of Hoeffding inequality that works explicitly on functions of the data. Thus, they presented The McDiarmid Tree algorithm (Rutkowski et al., 2012); a version of the Hoeffding Tree algorithm (Domingos and Hulten, 2000) that replaces the use of Hoeffding bound in the computation of the splitting attribute by the McDiarmid bound. Rutkowski et al. (2014) introduces the dsCART algorithm that, inspired by VFDT, adapts the CART offline algorithm to data streams. The authors compared the accuracy of dsCART with their previous proposal the McDiarmid Tree algorithm. The trees obtained by both algorithms are similar, so the final accuracy tends to the same value. However, dsCART needs fewer data elements to make a split, therefore, requiring less examples to reach the top accuracy levels. This is especially important in some particular concept drifting cases.

The use of McDiarmid bound represents a relatively new result. Although Hoeffding trees may, in the future, be demonstrated to be mistaken in the sense of being based on assumptions that do not hold, they are still very effective in practice and widely used in different implementations. Their reasonable results may be due to the fact that Hoeffding bound provides, in most cases, an overestimation of the true probability error.

Apart from decision trees, there are many other classification methods, but only a few can be applied to the configuration of data streams without losing accuracy and efficiency. k-Nearest Neighbors (kNN) (Read et al., 2012) may be the most obvious batch method to try in the streaming context. Instance based learning is inherently incremental. The change to streaming is simply achieved by limiting the search space for determining the k-nearest neighbors to a sliding window. Indeed, in Read et al. (2012) the method using a sliding window with the 1,000 most recent instances was found to be significantly effective. If implemented ingenuously, the method can be inefficient at prediction time due to the neighbor search. However, it is possible to improve prediction efficiency by indexing the instances in the sliding window (Zhang et al., 2011).

Thanks to the lazy learning scheme, no model that could get outdated is generated. Thus, the method can naturally respond to concept drift as the window slides. However, different types of concept drift (abrupt or gradual) can appear and the lazy learner should respond differently to them (Beringer and Hüllermeier, 2007). In Losing et al. (2016), this issue is addressed by proposing the use of two different memories: a short-term memory for the current concept, and a long-term memory to keep knowledge from past concepts. The authors claimed that the method is useful in practice since it does not need any meta-parameters to be tuned. Furthermore, they were able to obtain very competitive results in benchmark tests.

More recently, IBLStreams was introduced by Shaker and Hüllermeier (2012), an instance-based approach that can be applied to both classification and regression problems.

IBLStreams is based on adding and removing instances from the case base (instances than form the classifier). An instance should be retained as long as it is useful to improve the predictive performance. IBLStreams based its decision on retaining or not an instance on three indicators of usefulness: temporal relevance, spatial relevance and consistency. As in traditional kNN, the neighborhood is defined by the k_{cand}-nearest neighbors. The idea is that the most recent examples are the most relevant. When a new example is received, it is added to the case base and neighboring (redundant) examples are checked for removal. The most recent examples are not included in the candidates for removal to prevent removing as noise what is actually the beginning of a new concept. The algorithm is intended to have a more or less uniform coverage of the data space, and to eliminate data that are believed to be inconsistent with the current concept. Moreover, IBLStreams employs a drift detection method to detect abrupt changes. In case a change is detected, a large number of instances is removed from the base.

Very Fast Decision Rules (VFDR) (Kosina and Gama, 2015) algorithm is an online, any-time and one-pass method for learning decision rules in the context of data streams. The paths from the root of a tree to the leaves can be expressed as a set of unordered IF-THEN rules. This set of rules encapsulates the main characteristics of the decision problem. Rule sets present, though, some advantages over decision trees. They are not hierarchically structured, i.e., each rule can be handled independently from the rest of the rules in the set. Therefore, the set of rules can be altered more easily. Individual rules which are considered outdated can be just removed without hardly affecting the learning efficiency, with no need to rebuild the classifier from scratch or execute a complicated change in the tree structure. VFDR is partially inspired by VFDT. Each rules contains the sufficient information needed to expand the rule and to classify the test instances. These statistics are continuously updated based on the training examples that the rule covers. The number of observations required to expand a rule or induce a new one is determined by the Hoeffding bound. Two different strategies can be used to classify test examples: majority class or Naïve Bayes classifier.

Support Vector Machines (SVMs) achieve prominent performance in many offline machine learning problems although their application in large-scale datasets is costly due to their high time and memory complexity. Different incremental approaches have been proposed to make viable the used of SVM techniques in problems where the amount of incoming data is extremely large, such as data stream environments. LASVM (Bordes et al., 2005) is an online algorithm that converges to the SVM solution. Experimental evidence shown by the authors indicates that it reaches competitive accuracy rates after one single pass over the training data. Rai et al. (2009) developed StreamSVM, a one-pass SVM approach for data streams. It is a streaming extension of CVM (Tsang et al., 2005). CVM uses Minimum Enclosing Balls (MEBs), hyper-spheres that represent the set of examples inside the, to reduce its complexity. In StreamSVM, the radius of a MEB is flexible and it is updated every time a new training example is added. The authors performed experimental results that showed StreamSVM to be able to learn efficiently in just one pass and reach accuracies comparable to other SVM approaches (batch and

online). Nonetheless, neither LASVM or StreamSVM are prepared to react to concept changes.

In Domeniconi and Gunopulos (2001), the authors proposed an incremental SVM approach that needs the incoming stream to be divided in chunks of a given size. A set of w models representative of the last w chunks is kept in memory. Every time a new chunk is received, the oldest model is removed from memory, the rest of the models are incrementally updated to take into account the new chunk, and a new model is created based exclusively in the new data chunk. At each step, the oldest model kept in memory, the one trained in the last w chunks, is the one used to predict the labels of new data. Several techniques are discussed for the incremental updates. This algorithm is aimed at maintaining an accurate representation of recent data. However, the chunk size can be a critical parameter to get the algorithm to properly react to change.

We mention two more single-model proposals that have the potential to adapt to data stream setting. Last et al. (2002) proposes an Info-Fuzzy Network (IFN) classification system, which uses a fuzzy information network as the base classifier. IFN is a network-based classification model, designed to minimize the total number of prediction attributes. The underlying principle of the IFN method is to build a multi-layered network to test the mutual information between input and output attributes. A standard decision tree can easily be extracted from the IFN structure by re-moving the target layer and associating a single classification rule with each terminal node in the network. Each hidden layer is related to a specific input attribute and represents the interaction between this input attribute and the others. The IFN algorithm uses the previous pruning strategy: a node is divided if this procedure produces a statistically significant decrease in the entropy value (or an increase in mutual information) of the target attribute. If none of the remaining input attributes provide a statistically significant increase in mutual information, the construction of the network is stopped. The stability of the IFN algorithm is ensured by restricting the tree structure to using the same feature for all nodes of the same tree level and by the built-in statistical significance tests.

AWSOM (Arbitrary Window Stream mOdeling Method) is a method to discover interesting patterns from sensors proposed by Papadimitriou et al. (2003). It is a one-step algorithm that updates the patterns incrementally. This method requires only O(logn) memory where n is the length of the sequence. It uses wavelet coefficients as compact information representation and correlation structure detection, applying a linear regression model in the wavelet domain.

However, ensembles are currently the most often studied classifiers in the data stream research field. Krawczyk et al. (2017) surveys research on ensembles for data stream classification and regression tasks. In addition, the paper discusses several advanced learning concepts, as well as open research problems and future research lines. According to Krawczyk et al. (2017), ensemble classifiers for data streams can be categorized according to different criteria but the following categorizations are the most common ones:

- Stationary versus non-stationary stream classifiers: approaches for stationary environ-

ments do not contain any mechanism to react to concept drifts, while approaches for non-stationary environments are specifically designed to address possible conceptual deviations.

- Active versus passive approaches: approaches to address concept drift are generally distinguished between active and passive approaches (Ditzler et al., 2015; Stefanowski, 2015; Žliobaitė, 2010). Active algorithms contain drift detectors that trigger changes in classifiers (Gama et al., 2014). Passive approaches, on the other hand, do not use any special techniques to detect concept drift. Instead, they continuously update the classifier every time a new data instance is received (regardless whether a real drift is occurring or not). Most of the current ensembles follow a passive adaptation scheme.

- Chunk based versus online learning modes: chunk-based approaches process data into chunks, where each chunk contains a fixed number of training examples, while online learning approaches process training examples one by one, upon arrival. In chunk-based approaches, the learning algorithm may be able to iterate over the training examples in each chunk more than once. This allows to use batch algorithms to learn base classifiers. Online learning approaches are meant for applications with strict time and memory limitations, or, in general, applications where it is not affordable to process each training example more than once, for instance, due to a very large amount of incoming data.

- Differentiating techniques for updating base classifiers and aggregating their predictions: four basic strategies are distinguished (Kuncheva, 2004), namely, dynamic combiners, updating training data, updating ensemble members, and structural changes of the ensemble. In the first case, the ensemble adapts by changing the combination phase, for example, by tuning the classifier weights inside the voting rule (Jacobs et al., 1991; Littlestone and Warmuth, 1994). The updating training data approach is based on using recent training examples to online-update the base classifiers (Oza, 2005; Bifet et al., 2010b; Wang et al., 2014). Some approaches online update the ensemble members or retrain them in batch mode (using chunks) (Bifet and Gavaldà, 2009b; Fern and Givan, 2003; Kolter and Maloof, 2007; Oza and Russell, 2001; Rodríguez and Kuncheva, 2008), while others adopt an strategy based on making structural changes on the ensemble, e.g., dynamically evaluating the classifiers and replacing the worst one by a new one trained on the most recent data (Jackowski, 2014; Kotler and Maloof, 2003).

Krawczyk et al. argue that the main criteria for categorizing classification ensemble approaches are the data processing method, that is, if examples are processed in chunks or one-by-one, and whether the approaches are designed for stationary or non-stationary streams. These two criteria determine the type of data stream applications the approaches tackle. Thus, they proposed a taxonomy with the following four main categories: chunk-based ensembles for stationary streams, online ensembles for stationary streams, chunk-based ensembles for non-stationary streams, and online ensembles for non-stationary

streams. Within some of these categories, further criteria are used to distinguish between existing classifiers.

Chunk-based ensembles for stationary data streams have received less attention from the research community than their online counterparts and, therefore, are not so well developed. Chunk-based approaches are also related to batch processing of larger datasets, and usually do not explicitly refer to this as data stream mining (Polikar et al., 2001; Minku et al., 2009; Zhao et al., 2010). Due to a general popularity of online learning and its various real-world applications, online ensembles for stationary streams have received significantly more attention (Oza, 2005; Bifet et al., 2009a, 2010b; Gama, 2010; Gama et al., 2005; Saffari et al., 2009; Denil et al., 2013).

When addressing non-stationary environments, chunk-based approaches usually adapt to concept drifts by creating new base classifiers from new chunks of training data. Constructing a new component on the most recent chunk is a natural way of adapting to drifts (Žliobaitė, 2010). In general, the different base classifiers are learned from chunks that correspond to different parts of the data stream. Thus, the ensemble may combine representation of different concepts. In addition, learning base classifiers from complete chunks allows applying standard, batch algorithms. Old classification knowledge can be forgotten by removing classifiers that are performing too poorly. This allows to limit the amount of memory required for the ensemble model but, on the other hand, impedes that the eliminated classifiers can be reused in case their corresponding concept reoccurs. Therefore, some chunk-based approaches keep an additional buffer to store old classifiers as a way to handle potential recurring concepts as the classifiers stored in the buffer could be reused if needed. It is common among chunk-based ensembles to periodically assess their base classifiers on the newest chunk. This assessment is often used to update the weights associated to the components so they can emphasize the classifiers that best reflect the data distribution of the most recent chunk, or they can be used to decide which useless classifiers should be discarded.

Two different kinds of chunk-based ensembles for non-stationary environments can be distinguished based on whether or not they always create new classifiers for each new incoming data chunk to deal with concept drifts. On the one hand, typical chunk-based approaches always build a new classifier on each incoming data chunk. This makes these approaches especially sensitive to proper tuning of chunk size. A too large chunk size would mean slow adaptation to drifts, while too small chunk size would not be enough to properly learn an entire stable concept, resulting in poor classification performance, and would increase computational costs. Examples of approaches following this learning scheme are Streaming Ensemble Algorithm (SEA) (Street and Kim, 2005) and Accuracy Weighted Ensemble (AWE) (Wang et al., 2003a). The key idea of AWE is to assign weights to each component classifier based on their prediction error on the newest training chunk. This approach assumes that the newest training chunk is likely to present a more similar distribution to the current test examples.

On the other hand, some researchers proposed alternative chunk-based approaches

that deviate from the traditional chunk-based learning schema in an attempt to reduce the sensitivity of the approaches to chunk size tuning, or cut down the potential unnecessary learning overhead from learning every new data chunk even when the existing classifiers may be considered good enough for the current concept. These approaches establish some criteria to decide whether it is necessary or not to create a new classifier to learn the new incoming chunk (Deckert, 2011; Brzeziński and Stefanowski, 2011; Brzezinski and Stefanowski, 2013). Furthermore, some approaches specifically aimed at addressing recurring concepts have been studied (Ramamurthy and Bhatnagar, 2007; Sobolewski and Woźniak, 2017; Jackowski, 2014).

Finally, online ensembles are able to learn the data stream in one pass, potentially being faster and with lower memory requirements than chunk-based approaches. They are able to do so because they learn each incoming example individually, instead of in chunks, and then discard it. This also allows them to avoid the need for choosing a proper chunk size. However, online approaches would often have other parameters affecting the speed of reaction to concept drifts (e.g., sliding window or fading factors parameters).

As discussed above, one of the main features for distinguishing between different online learning approaches for non-stationary environments is the use of concept drift detection methods, distinguishing between passive (concept change adaptation) or active (concept change detection) categories. Most of the passive approaches contain mechanisms to continuously adapt to concept drifts that may appear, whereas the speed of the adaptation and its sensitivity to noise often depends on parameters (Kolter and Maloof, 2007, 2005; Brzezinski and Stefanowski, 2014; Yoshida et al., 2011). Dynamic Weighted Majority (DWM) (Kolter and Maloof, 2007) is one of the most well known passive approaches. It maintains a weight for each of the component classifiers that is reduced by a multiplicative constant every time the classifier makes a prediction error. The active approaches are much less frequent, but there are some methods such as Adaptive Classifiers-Ensemble (ACE) (Nishida and Yamauchi, 2007), Todi (Nishida, 2008), ADWINBagging (Bifet et al., 2009b) or Diversity for Dealing with Drifts (DDD) (Minku and Yao, 2011).

II.2 Frequent pattern mining and association rules in data streams

In broad terms, a *pattern* can be defined as an entity that is present (or absent) with a frequency that deviates from the random. In frequent pattern mining, the input to the data mining process is a dataset (or stream) of transactions D, where each transaction can be viewed as a pattern with an associated id. It is said that a transaction t supports a pattern p, if p is a subpattern of the pattern defined by t. The support of a pattern p in a set (or stream) of transactions D is the number of transactions in D that support p. Thus, given a dataset D and a support threshold σ in $[0, 1]$, the frequent pattern mining problem can be defined as finding all σ-frequent patterns in D, that is, finding all the

patterns with a support in D equal or greater than σ.

Frequent pattern mining is an important unsupervised learning task, which has multiple application fields. It is a widely studied field in the literature, in both batch mining and data stream mining areas. Probably, the most naïve implementation of a solution to the frequent pattern mining problem would consist on going through the entire dataset, keeping track of the frequency of every pattern in the dataset. However, this approach is very inefficient and since the number of subpatterns in the dataset tends to rapidly grow with dataset size, it quickly turns unfeasible. Thus, other more efficient approaches are required and have been proposed.

Apriori (Agrawal and Srikant, 1994) relies on the antimonoticity property (or Apriori property) to restrict the search for frequent patterns to only a subset of the subpatterns present in the dataset. This property stipulates that any subpattern of a frequent pattern is also frequent. This is equivalent to saying that any superpattern of an infrequent pattern will also be infrequent, which allows Apriori to narrow down the list of frequent pattern candidates without risking losing any truly frequent patterns. Another well-known approach to batch itemset mining is the FP-growth algorithm (Han et al., 2004). FP-growth avoids the expensive candidate generation phase thanks to the use of an FP-Tree. A data structure that allows FP-Growth to store the dataset in a compact way employing two passes, and from which the frequent itemsets can be directly retrieved. The Eclat algorithm (Zaki et al., 1997) uses a depth-first search and is able to find the frequent patterns performing only one pass over the dataset.

Nonetheless, as it happens in classification and other data mining domains, data streams raise new challenges for which batch-oriented proposals for frequent pattern mining are not prepared. The approaches outlined above are not designed to return results in an anytime way, and the amount of patterns they store is too high for streaming settings. Therefore, over the last years different proposals have been developed to address the problem of frequent pattern mining in streams.

Algorithms for extracting frequent patterns from data streams can be classified according to different criteria (Bifet et al., 2018; Sancho-Asensio et al., 2016). We use the categorization proposed in Sancho-Asensio et al. (2016) as a guideline to review the proposals that can be found in the literature of frequent pattern mining in data streams. This categorization is based on the following criteria:

- **Category**. The pattern mining algorithms can be categorized according to the specific problem they address: (1) heavy hitters—find the singleton items with a support greater than the given threshold—, (2) top-k frequent itemsets—find the k most frequent itemsets in a stream—, (3) frequent pattern mining—find all the itemsets (of any length) with a support greater than the given threshold—, (4) mining closed frequent itemsets—find those itemsets that do not have any frequent superset with the same frequency (avoiding reduncancy (Bifet et al., 2010a))—, (5) rare itemset mining—itemsets that do not occur frequently—, (6) ratio rules—find the quantitative knowledge between distinct itemsets inside a rule—, (7) frequent closed

graph mining—find those graphs that have no frequent supersets with the same frequency—, (8) frequent sequence mining—find sequences of itemsets with a support greater than a given threshold—, (9) probabilistic pattern mining—find those itemsets with a probability greater than a minimum threshold—, (10) association rules—find all the frequent (quantitative) rules—, and (11) fuzzy association rules—find all the frequent fuzzy rules.

- **Type**. The approaches employ different types of algorithms to extract the frequent patterns from the streams. Some of the most extended ones are: (1) counting-based—an iterative counting algorithm—, (2) tree-based—a tree structure is built for pattern identification and extraction—, and (3) hashing-based—hash tables are employed for frequent itemset discovery. Although queue-based, graph-based and hyper-structure-based proposals are also present in the literature.

- **Data**. Different types of data can form the streams. The approaches can manage the following data types: (1) categorical, (2) real or continuous, (3) uncertain and (4) fuzzy data. The categorical data is the most commonly supported by far.

- **Approach**. As mentioned before, algorithms for mining frequent patterns in data stream can also be distinguished depending on whether they consider the frequency of the patterns from the beginning of the stream (landmark-window), or they confer more importance on recent items (sliding window, tilted-time window or decay factor).

- **Rules**. Most frequent pattern mining methods do not produce rules. Nonetheless, a small fraction of these methods can generate rules naturally.

- **Experiments**. The experimental setups in the literature are basically two: (1) synthetic environments and (2) real-world datasets. Both setups are often combined.

We would argue that, from the mentioned classification criteria, *Category* can be considered the main one. It allows to differentiate the proposals according to the type of task they intend to solve, i.e., proposals in different categories pursue different objectives. Along the following subsections, we review the main techniques proposed in the literature of data streams for each of these categories of pattern mining. Moreover, Table II.1 shows a survey of the proposals in the area.

II.2.1 Heavy hitters

Heavy hitters is a reduction of the frequent pattern mining problem, where only frequent singleton items are targeted. Thus, given a threshold σ and a stream, having read a segment of length t of the stream, the set of heavy hitters consists of all those items whose relative frequency exceeds σ. The relative frequency is understood as the absolute frequency of the item divided by t.

Table II.1: Characteristics of frequent pattern mining algorithms for data streams.

Reference	Category	Type	Data	Approach	Rules	Experiments
(Manku and Motwani, 2002)	Heavy hitters, Frequent pattern mining	Counting	Categorical	-	No	Mixed
(Karp et al., 2003)	Heavy hitters	Counting	Categorical	-	No	-
(Jin et al., 2003)	Heavy hitters	Hashing	Categorical	-	No	Mixed
(Chang and Lee, 2003)	Frequent pattern mining	Tree	Categorical	Decay factor	No	Synthetic
(Giannella et al., 2003)	Frequent pattern mining	Tree	Categorical	Tilted-time window	No	Synthetic
(Charikar et al., 2004)	Heavy hitters	Hashing	Categorical	-	No	-
(Metwally et al., 2005)	Heavy hitters	Counting	Categorical	-	No	Synthetic
(Chi et al., 2006)	Closed itemsets mining	Tree	Categorical	Sliding window	No	Mixed
(Wong and Fu, 2006)	Top-k frequent itemsets	Counting	Categorical	Sliding window	No	Synthetic
(Marascu and Masseglia, 2006)	Sequential pattern mining	Tree	Categorical	Tilted-time window	No	Mixed
(Yang et al., 2007)	Frequent pattern mining	Tree	Categorical	-	No	Synthetic
(Raïssi and Poncelet, 2007)	Sequential pattern mining	Counting	Categorical	Sliding window	No	Mixed
(Cormode et al., 2008)	Heavy hitters	Counting	Categorical	-	No	Mixed
(Cheng et al., 2008)	Closed itemsets mining	Inverted index structure	Categorical	Sliding window	No	Synthetic
(Wang and Chen, 2009)	Frequent pattern mining	Hashing	Categorical	-	No	Mixed
(Leung and Hao, 2009)	Frequent pattern mining	Tree	Uncertain	Sliding window	No	Synthetic
(Yen et al., 2009)	Closed itemsets mining	Tree	Categorical	-	No	Synthetic
(Fan et al., 2009)	Ratio rules	Counting	Quantitative*	Sliding window	Yes	Mixed
(Chen et al., 2010)	Fuzzy association rules	Tree	Fuzzy	Sliding window	Yes	Mixed
(Tu et al., 2010)	Frequent pattern mining	Tree	Categorical	Sliding window	No	Mixed
(Cormode and Muthukrishnan, 2011)	Heavy hitters	Counting	Categorical	-	No	Mixed
(Wang and Chen, 2011)	Frequent pattern mining	Hashing	Categorical	-	No	Mixed
(Memar et al., 2011)	Frequent pattern mining	Queue	Categorical	Sliding window	No	Synthetic
(Bifet et al., 2011)	Frequent closed graphs mining	Graph	Categorical	Sliding window	No	Mixed
(Huang et al., 2012)	Rare itemsets mining	Tree	Categorical	Sliding window	No	Mixed
(HewaNadungodage et al., 2013)	Frequent pattern mining	Hyper-structure	Uncertain	Sliding window	No	Mixed
(Akbarinia and Masseglia, 2013)	Probabilistic pattern mining	Counting	Uncertain	Sliding window	No	Mixed
(Zihayat and An, 2014)	Top-k frequent itemsets	Tree	Categorical	Sliding window	No	Mixed
(Braverman et al., 2016)	Heavy hitters	Hashing	Categorical	-	No	-
(Woodruff, 2016)	Heavy hitters	Hashing	Categorical	-	No	-
(Roy et al., 2016)	Heavy hitters	Hashing	Categorical	-	No	Mixed
(Sancho-Asensio et al., 2016)	Fuzzy association rules	Counting	Fuzzy	-	Yes	Mixed
(Basat et al., 2017)	Heavy hitters	Counting	Categorical	-	No	Real
(Dawar et al., 2017)	Top-k frequent itemsets	List structure	Categorical	Sliding window	No	Mixed
(Zihayat et al., 2017)	Sequential pattern mining	Tree	Categorical	-	No	Mixed
(Kusumakumari et al., 2017)	Frequent pattern mining	Tree	Categorical	Sliding window	No	Real
(Basat et al., 2018)	Heavy hitters	Counting	Categorical	Sliding window	No	Mixed
(Yun et al., 2018)	Frequent pattern mining	Tree	Categorical	Decay factor	No	Real
(Li et al., 2018)	Probabilistic pattern mining	Tree	Uncertain	Sliding window	No	Mixed
(Liu et al., 2018)	Frequent pattern mining	Tree	Uncertain	Sliding window	No	Mixed
(Bustio-Martínez et al., 2019)	Frequent pattern mining	Hashing	Categorical	Sliding window	No	Mixed
(Ovi et al., 2019)	Frequent pattern mining	Tree	Uncertain	Sliding window	No	Mixed
(Xie and Tan, 2019)	Frequent pattern mining	List structure	Uncertain	Sliding window	No	Mixed
(Choi and Park, 2019)	Frequent pattern mining	Tree	Categorical	Sliding window	No	Real
(Ventruto et al., 2020)	Heavy hitters	Hashing	Categorical	-	No	Synthetic
(Xiao et al., 2020)	Heavy hitters	Hashing	Categorical	-	No	Real
(Veloso et al., 2020)	Heavy hitters	Counting	Categorical	Decay factor	No	Real
(Yang et al., 2020)	Closed itemsets mining	Tree	Uncertain	Decay factor	No	Synthetic
(Goyal et al., 2020)	Frequent pattern mining	Tree	Categorical	Decay factor	No	Mixed

Several algorithms have been proposed in the literature to tackle the heavy hitters problem. There are two main approaches: counter-based, and hash-based. Counter-based methods maintain counters for a certain set of elements of the stream. Thus, only this limited number of elements is monitored. If an item of the stream arrives and it is being already monitored, the associated counter is incremented. Otherwise, the algorithm decides whether the item is included in the set or discarded.

Historically, most of the counter-based algorithms are evolutions of the method proposed by Boyer and Moore (1991) to find the majority element, that is, the one with a frequency of at least 0.5, if it exists. FREQUENT (Karp et al., 2003; Misra and Gries, 1982; Demaine et al., 2002) is an improvement of Boyer and Moore's method that is able to obtain a list of elements among which all σ-heavy hitters are guaranteed to be included. However, the main drawback of FREQUENT is that it does not provide any reliable estimation of the frequency of these σ-heavy hitters. Examples of counter-based algorithms that are able to provide approximations of such frequencies are the proposals by Manku and Motwani (2002), Lossy Counting and Sticky Sampling, and the proposal by Metwally et al. (2005), Space Saving.

Lossy Counting is a deterministic algorithm that conceptually divides the incoming stream into buckets of width $w = \frac{1}{\epsilon}$ transactions. The algorithm maintains a set of entries $\mathcal{D}$, each of which has the form (e, f, Δ), where e is an element from the stream, f represents its estimated frequency (counter) and Δ is the maximum possible error for f. When an item arrives, if the item already exists in $\mathcal{D}$, its estimated frequency f is incremented, else a new entry is created. At the end of each bucket, $\mathcal{D}$ is pruned by deleting every entry for which $f + \Delta \leq b_{current}$, where $b_{current}$ is the index of the current bucket. The output of the algorithm is a list of items composed by those entries in $\mathcal{D}$ with $f \geq (\sigma - \epsilon)/N$. The authors show that Lossy Counting uses at most $1/\epsilon \cdot \log(\epsilon \cdot N)$ space where N denotes the current length of the stream.

As opposed to Lossy Counting, Sticky Sampling is a probabilistic algorithm. It maintains a set S of pairs item-frequency. Each time an item from the stream is received, if the item was already included in S, its frequency (counter) is incremented, else the new item is added to S with probability $1/r$. The sampling rate r evolves over the lifetime of the input stream. Each time the sampling rate is updated, all the entries of S are revisited and some of them deleted, so S is transformed to the state it would have been in if the new rate had been used from the beginning. Like Lossy Counting, the algorithm returns those items of S for which $f \geq (\sigma - \epsilon)/N$. Additionally, the authors compared them and experimentally showed that Lossy Counting performs better in practice, even though it has a theoretically worse worst-case bound.

SpaceSaving keeps in memory a maximum of k different elements together with their occurrence counters. When a new element that is not one of the elements included in that set is received from the stream, the new element replaces the element in the set that had the counter with the lowest value and that counter is incremented by one. SpaceSaving is a simple algorithm but at the same time presents rigorous guarantees on the quality

of the approximations. The frequency approximations estimated by SpaceSaving do not underestimate the true frequencies and do not overestimate them by more than t/k, where t is the number of elements received so far. The *Stream-Summary* data structure proposed in Metwally et al. (2005) implements the sketch ensuring constant time per update. In addition, it has been claimed in various occasions to achieve a better performance than several other heavy hitter algorithms (Cormode and Hadjieleftheriou, 2009; Liu et al., 2011; Manerikar and Palpanas, 2009).

On the other hand, hash-based (also called sketch-based) methods employ hashing techniques to map items to a reduced set of counters. They maintain approximate frequency counts of all elements in the stream.Therefore, as opposed to counter-based methods, hash-based methods are able to monitor all elements in the stream, instead of just a limited set of them. Two of the most well-known hash-based sketches are Count-Min Sketch (Cormode and Muthukrishnan, 2011) and CountSketch (Charikar et al., 2004). They support both item additions and subtractions and can be used to solve various problems related to item frequencies, including heavy hitters.

CountSketch (Charikar et al., 2004) employs an array A of w counters and two hash functions (both assumed to be random enough). Function h hashes items to positions in the array and σ maps items to the set +1,-1, i.e., mapping to addition or subtraction. Therefore, for a given item all the updates are either additions or subtractions. The algorithm takes as estimated frequency of an item x $f_x = \sigma(x) \cdot A[h(x)]$. Some approaches have been presented to improve the memory efficiency of CountSketch (Braverman et al., 2016; Woodruff, 2016).

Count-Min sketch (Cormode and Muthukrishnan, 2011) consists of a two-dimensional array of counters with d rows and w columns. Furthermore, d independent hash functions, chosen at random, map each item to a column in the sketch. Thus, when an item arrives, one counter in each row is incremented. The output of the hash functions points to the counter to be incremented in each row. For any item, the minimum of its associated counters is taken as its estimated frequency. The j-th counter of the i-th row contains the sum of the frequencies of all those items mapped to it by the i-th hash function. Therefore, the frequency of an item can only be overestimated. hCount, presented in Jin et al. (2003), implements a Count-Min Sketch to keep an approximation of the frequency of every item. In this work, a brute force approach, which checks the estimated frequency of every possible item, is used to extract the heavy hitters from the Count-Min Sketch. Nonetheless, this approach is unviable for large item universes. There is no obvious efficient way to locate the heavy hitters inside the Count-Min Sketch. Keeping additional information about the frequencies of groups of items or using a hierarchical implementation can help speed up the query process, at the expense of increasing the space requirement (Manerikar and Palpanas, 2009).

Some approaches have been proposed to try to reduce the error rate of Count-Min Sketch. In Roy et al. (2016), Augmented Sketch (ASketch) is proposed to increase the frequency estimation accuracy of the most frequent items and reduce the misclassification

of low-frequency items. The proposed solution is based on trying to reduce the collisions with frequent items by removing the frequent items from the sketch. This is done by means of a pre-filtering stage in which frequent items are dynamically identified and removed from the main sketch into a second data structured, called a filter. The filter is formed by a set of k counters shaped as (item[i], new_count[i], old_count[i]), where $i = 1, 2, \ldots, k$ and item[i] is the item monitored by the i-th counter. The sketch can take different forms depending on the frequency estimation algorithm on which ASketch rests, what adds to the solution a degree of generalization. The algorithm can generate false negatives if the filter is not sized properly. Furthermore, ACMSS (Ventruto et al., 2020) has been recently proposed to improve the accuracy of ASketch. Like ASketch, ACMSS uses two data structures: a filter and a sketch. The frequent items are inserted into the filter after being identified as such in the sketch. The main differences of ACMSS over ASketch are: (1) the fact that the sketch is based on a space optimized version of the CMSS sketch (Cafaro et al., 2019), (2) a conservative sketch update policy (Goyal and Daumé III, 2011), and (3) a different swap policy to determine which items must be moved from the filter data structure to the sketch.

Hash-based methods provide useful information in addition to the heavy hitters. However, following Manerikar and Palpanas (2009), if the aim is strictly limited to discovering the frequent items (heavy hitters), counter-based approaches are probably preferable. They are likely to perform better in time, memory, and accuracy. In Manerikar and Palpanas (2009), Count-Min Sketch and CountSketch were found to be less stable and perform worse for some parameter ranges. Moreover, counter-based methods are normally easier to implement.

Mining heavy hitters from a stream is an important problem from both theoretical and application perspective. One of the most relevant application fields of heavy hitters is network traffic monitoring. In the last years, several approaches have been presented for this application field (Cormode et al., 2008; Xiao et al., 2020; Basat et al., 2017, 2018). Many of these studies manage the concept of Hierarchical Heavy Hitters (HHH) (Cormode et al., 2008). The hierarchy is defined based on the type of prefixes in a certain application (e.g., IP prefixes), i.e., frequent flow aggregates based on common prefixes values. Cormode et al. (2008) introduces deterministic methods for both single-hierarchical and multi-hierarchical problems. In the case of the multi-dimensional problem, the proposed algorithms exploit the mathematical lattice structure, resulting from the product of hierarchical dimensions, which allows them to track approximate HHHs using a fixed number of statistics per stored item, independently of the number of dimensions. Basat et al. (2018) propose randomized constant time algorithm for mining HHHs. This method uses a matrix of H independent heavy hitters algorithms, where each node is responsible for a single prefix pattern. For each packet, the approach updates at most a single randomly selected heavy hitter algorithm. These two previous algorithms give identical importance to each item from the beginning of the stream. A family of methods for mining both HHs and HHHs in the single-device and network-wide settings is presented in Basat et al. (2018).

For a similar application field, two different algorithmic solutions are proposed

in Veloso et al. (2020). The specific target problem is called the Interconnect Bypass Fraud and the aim is to rapidly detect numbers with abnormal behaviors (bursts of calls, repetitions, mirror behaviors...). The proposed solutions use the heavy hitters to detect such abnormal behaviors. The first solution explored is based on the incorporation of forgetting factors in the Lossy Counting algorithm. The other solution explored is a single pass algorithm for mining hierarchical heavy hitters that also adopts a fast forgetting mechanism. This second proposal is based on the offline algorithm proposed in Cormode et al. (2004).

II.2.2 Frequent pattern mining

Schemes designed for heavy hitters are often extensible to other problems related to frequent pattern mining. In fact, Manku and Motwani (2002) extends Lossy Counting to sets of items (or *itemsets*). This extension of Lossy Counting maintains, as the heavy hitter approach, a data structure $\mathcal{D}$. However, in this case, each entry of $\mathcal{D}$ has the form (set, f, Δ), where set is a subset of items instead of a singleton item, and the incoming stream is not processed transaction by transaction but divided in batches. The algorithm tries to fill the available main memory with as many transactions as possible, and then process such batch of transactions together. If an itemset that is not already in $\mathcal{D}$ occurs β or more times in the current batch, where β is the number of buckets in the current batch, the itemset is added to $\mathcal{D}$. The amount of memory available may increase/decrease over time. Small values of β can cause more spurious subsets get into $\mathcal{D}$. Nevertheless, when it comes to finding frequent itemsets, counting and hashing are not the only types of solutions that can be found in the literature. Tree-based approaches (Giannella et al., 2003; Chi et al., 2006; Yang et al., 2007; Yun et al., 2018; Ovi et al., 2019) are numerous, but queue-based (Memar et al., 2011) or hyper-structure-based approaches (HewaNadungodage et al., 2013) have also been introduced.

In general terms, most of the proposals for mining frequent itemsets in the literature follow a strategy in which the frequency of itemsets is estimated while monitoring the incoming transactions and a data structure is maintained with those itemsets that are estimated to be frequent or close to frequent (they are not frequent but may end up becoming frequent). When a query is performed to return the frequent itemsets, those itemsets that currently exceed the specified minimum support threshold are extracted from the data structure. This scheme is referred to in some publications as a two-steps method (Yang et al., 2007) or as “immediate” mining mode (Leung and Hao, 2009).

Wang and Chen (2009) presented hMiner. A hashing-based algorithm that employs a data structure, referred as hSynopsis, to summarize the data stream. This data structure comprises a hash table and frequent nodes. Each entry of the hash table is linked to a list of f-nodes (frequent nodes), and stores the total number of accesses to the entry and the time stamp of the last access. The f-nodes are exploited to keep the information of the frequent itemsets. When a new transaction arrives, all the itemsets contained in the current transaction are enumerated, sorted by increasing length and sequentially hashed.

When an itemset is hashed into an entry, the total accesses and last access fields of the entry are updated. If the itemset is identified by one of the f-nodes linked to the entry, the true count of the node is increased by one. Otherwise, a new f-node to keep track of the itemset will be created and linked to the entry only if the itemset is estimated to be frequent enough. Once all the itemsets in the current transaction are hashed all the accessed entries in hSynopsis are checked for removing the f-nodes for which the sum of the counter of occurrences since its insertion plus the estimated previous frequency is less than σN, where N is the number of transactions received so far. Later, in Wang and Chen (2011), the authors propose a distributed computation framework to extend hMiner approach to mine global frequent itemsets from a collection of data streams distributed at distinct remote sites. They use hSynopsis (Wang and Chen, 2009) to summarize the local streams, and present communication strategies rooted in hSynopsis.

The frequent pattern mining approaches mentioned consider all the transactions received equally relevant. Nonetheless, in the frequent pattern mining literature, it is common to find proposals that give more importance to the most recent transactions by means of techniques such as sliding window (Memar et al., 2011; Akbarinia and Masseglia, 2013; Bustio-Martínez et al., 2019), tilted-time window (Giannella et al., 2003; Marascu and Masseglia, 2006) or decay factor (Chang and Lee, 2003; Yun et al., 2018; Goyal et al., 2020).

Chang and Lee (2003) propose estDec, an approximate algorithm for mining recent frequent itemsets. The proposal employs a decay factor, $d \in (0, 1)$, to reduce the weight of old transactions in the results. A prefix tree lattice structure (Brin et al., 1997; Agarwal et al., 2000) (monitoring lattice) is used to maintain the different combinations of items generated by the stream of transactions. estDec estimates the decayed frequency of a new n-itemset X, such that $n \geq 2$, based on the frequencies of its (n-1)-subsets. When the updated support of an itemset in a monitoring lattice becomes lower than a predefined threshold, the itemset is pruned from the monitoring lattice. Except for 1-itemsets, which are not pruned from the monitoring lattice because it would be impossible to estimate their count later. Then, the algorithm tries to find any new itemset that has a high possibility to become a frequent itemset in the near future according to its estimated frequency. In Chang and Lee (2005), the authors of estDec explore several frequency estimation methods that could be used in the last step (delayed-insertion) of estDec. The methods are analyzed in terms of mining accuracy, memory usage and processing time.

A prefix tree structure is also used in FP-Stream (Giannella et al., 2003). FP-stream employs a FP-Tree structure as in FP-growth and a tilted-time window to maintain the set of frequent itemsets. Each itemset in the FP-Tree is represented by a root-to-node path. The node at the end of the path has a titled-time window that keeps track of the frequency of the itemset at a finer granularity for more recent time frames and at a less smooth granularity for older time frames. The algorithm processes the data stream in batches. Every time a new batch is collected, a new FP-Tree is computed and added to the global FP-Tree. FP-Stream computes frequent and subfrequent (its support is greater than σ' but lower than σ). Pattern occurrences in a batch with a frequency below σ' will

not be added to the global FP-Tree, so they will be undercounted and may eventually become false negatives. However, this leaves out of consideration a large amount of truly infrequent patterns, saving memory and time. As the sliding window model, the tilted-time window concedes more importance to recent data than to old data. Nonetheless, it does not completely lose the information in the historical data. Hence, FP-stream allows answering more expressive time-sensitive queries at the expense of storing more than one frequency record per itemset.

The proposals for frequent pattern mining visited so far follow an "immediate" mining mode, i.e., the frequencies of the itemsets are estimated the moment a new transaction or batch is received in order to decide which itemsets are worth keeping. However, this strategy may involve wasting computational effort. Thus, other proposals opt to follow a "delayed" or lazy mining mode. In this case, the computation or estimation of the frequency of the itemsets is delayed until it is completely indispensable, i.e., until the algorithm is required to return the list of frequent itemsets. The algorithms using this mining mode keep in memory a compressed representation of the data stream (or of a window of the stream), from which it is possible to infer the frequency of the itemsets when required.

A delayed method for data streams, called DELAY, is presented in (Yang et al., 2007). DELAY first just stores necessary information from the incoming transactions. The algorithm employs two main data structures: a list to keep track of items and a tree for the itemsets. The list has a fix length and its updating procedure is inspired by Space Saving (Metwally et al., 2005). In order to save space, some itemsets are pruned based on the count of occurrences of their single items (i.e., without performing any actual frequency estimation). The frequency is not calculated until the query for frequent itemsets is submitted. When this happens, a pattern fragment growth step analogous to the second step of FP-growth is triggered to answer the query. A similar logic to save space is depicted in Bustio-Martínez et al. (2019), where the top-k frequent 1-itemsets detection is used as preprocessing and all the single items detected as infrequent are removed.

A queue-based approach is proposed in Memar et al. (2011), MFI-CBSW (Mining Frequent Itemset within Circular Block Sliding Window). This method employs a sliding window approach and considers the incoming transactions to be grouped in blocks. To improve the efficiency of the window sliding process, the authors propose a new technique referred as Circular Block Sliding (CBS). The algorithm uses a blocked-bit-sequence representation of items with a queue of non-zero block numbers to store all the transactions in the current window in a compressed format. Each bit in the blocked-bit-sequence corresponds to a transaction in the window and indicates if the item is present in such transaction. The queue of non-zero block numbers maintains the indices of the blocks of the current window where the item appears in at least one transaction (non-zero blocks). The bit sequence of an itemset in a block can be calculated as the conjunction of the bit sequences of its single items. Thus, to compute the support of an itemset, its blocked-bit-sequence is constructed using its subsets. For extracting frequent itemsets within the current window, MFI-CBSW first identifies frequent single items and then follows a depth

first method of traversing the prefix tree of itemsets.

MPM (Yun et al., 2018), designed to mine high utility itemsets, employs a tree-based data structure (DAT) to keep track of the incoming transactions. Each transaction is arranged in a lexicographic order of item identifiers so the arrived data processing is done in a single data scan. The average utility information is continually accumulated in this data structure until the user submits a mining request. It is when a mining request is performed that the data structure is updated to reflect recent average utility information according to a damped window model and MPM conducts its mining process over DAT based on a pattern growth approach. The algorithm recursively constructs conditional trees for selected prefixes in order to generate candidate patterns. Finally, one additional data scan is conducted for the candidate validation step, when the actual damped average utilities are calculated.

II.2.2.1 Managing data uncertainty

Sometimes, due to the presence of noise or to the nature of the data, there is a certain degree of uncertainty present in the data from the stream. In the context of frequent pattern mining, the difference between precise and uncertain data is that an uncertain transaction contains an existential probability for each item, which indicates the likelihood of the item being actually present in the transaction. Algorithms designed for precise (or deterministic) data are not directly applicable in uncertain (or probabilistic) data. Two main support measures are used for uncertain data in the literature: (1) *expected support* (Chui et al., 2007), which is an approximate measure of support, and (2) *probabilistic support* (Bernecker et al., 2009), which is an exact measure of support in probabilistic data.

Leung and Hao (2009) present two (one approximate and one exact) tree-based proposals for mining frequent itemsets from streams of uncertain data. The first one, UF-streaming, is an approximate method based on the algorithm for static datasets UF-growth (Leung et al., 2007, 2008), and the method for precise data streams FP-stream (Giannella et al., 2003). Similarly to FP-stream, when a new batch is received, UF-streaming mines those itemsets that are frequent or subfrequent in the current batch, and they are added to a prefix tree structure called UF-stream. UF-growth is used to mine such "frequent" itemsets in the current batch. Every node in the UF-stream includes both an item and a window table containing one expected support value per batch of transactions in the window. Every time a new batch of transactions is received, the window slides and the expected support values of each node in the tree shift. The authors put forward some potential problems associated to this approximate proposal, such as: the need of a post-processing step to find the truly frequent itemsets (discarding the subfrequent ones); the possibility of missing truly frequent itemsets if the subfrequent threshold is too close to the frequent threshold; the necessity of an additional data structure (the UF-stream) to store the mined itemsets; and the potential waste of computation derived from using an "immediate" mode of mining, especially when many batches are processed before the

mining results (frequent itemsets) are requested. Hence, they present an exact proposal, SUF-growth, which tries to overcome these limitations and potential problems. Being an exact algorithm, SUF-growth returns all and only those truly frequent itemsets (no false positives or false negatives are returned). Furthermore, SUF-growth does not follow an "immediate" mining mode but a "delayed" one and, therefore, does not need the UF-stream structure to maintain the mined itemsets. It builds a global tree called the SUF-tree which is always kept up-to-date with the sliding window. Given an appropriate minimum support threshold, the frequent itemsets can be mined from this up-to-dated SUF-tree in a similar manner to the UF-growth algorithm. Tree-based proposals for data uncertainty settings are also proposed in Liu et al. (2018); Ovi et al. (2019).

HewaNadungodage et al. (2013) argue that the consideration of existential probabilities causes FP-growth (Han et al., 2004) to lose its compression power on uncertain data. Thus, as opposed to tree-based proposals, two hyper-structure-based algorithms are proposed in HewaNadungodage et al. (2013) to efficiently mine frequent itemsets from streams of uncertain data: UHS-Stream and TFUHS-Stream. The main difference between both algorithms is that while UHS-Stream is designed to find all frequent itemsets up to the current moment, TFUHS-Stream does it in a time-fading manner. UHS-Stream processes the incoming stream in batches, applies the UH-mine algorithm to find potentially frequent and subfrequent itemsets in each batch, and stores these itemsets in a global tree structure referred as IS-tree. The main novelty is that instead of a FP-tree, UH-mine uses a hyper-linked array structure called the UH-struct. UH-mine first scans the input database and remove the infrequent items from the transactions. The frequent single items left are sorted following a certain global order. An array structure stores the transformed database, where each row corresponds to one transaction and each entry in a row has three fields: an item-id (corresponding to a frequent item present in the transaction), the existential probability of the item in the transaction, and a hyper-link pointing to the next transaction containing that item. A header table is constructed with each frequent item entry having three fields: an item-id, a expected support count, and a hyper-link to the starting point of the projected transactions. UH-mine can find the frequent itemsets by scanning the projected transactions linked together by the hyper-links. TFUHS-Stream also employs the UH-struct to mine frequent itemsets and stores them in the IS-tree structure. Nonetheless, it fades the recorded estimated frequency count and the maximum possible error by a decay factor λ.

Akbarinia and Masseglia (2013) presents FEMP (Fast and Exact Mining of Probabilistic data streams), a proposal for exact PFI (Probabilistic Frequent Itemsets) mining in data streams based on sliding window approach. Instead of expected support (Leung and Hao, 2009; HewaNadungodage et al., 2013), FEMP maintains the probabilistic support of the itemsets. This means that the support of an itemset is given as a probability distribution function, i.e., each possible value s for the support (from 0 to the size of the window) of an itemset X is associated to a probability (the probability of s being the support of X in the database). FEMP is able to obtain the exact probabilistic frequency distribution function for any monitored itemset, at any time. FEMP uses a recursion

on transactions to update the probabilistic support of the itemsets. Every time a new transaction is added to the window, the probabilistic support of an itemset X in the set of transactions T $= t_1, ...t_{n-1}, t_n$ is computed based on the probabilistic support in $T - t_n$. Similarly, for the case of transaction deletion, the algorithm computes the probabilistic support in $T - t_n$ applying an equation on the probabilistic support of the itemset in T.

Due to the high cost of computing probabilistic support, a method to estimate the range of probabilistic support based on the support and the expected support is proposed in Li et al. (2018). This work introduces an in-memory index named PFIT (Probabilistic Frequent Itemset Tree), to store the data synopsis of probabilistic frequent itemsets in a bottom-up manner, and presents the PFIMoS (Probabilistic Frequent Itemset Mining over Streams) algorithm to incrementally discover the probabilistic frequent itemsets over a sliding window. PFIMoS estimates the upper and lower bounds of probabilistic support, therefore, reducing the probabilistic support computing cost. Nonetheless, when the minimum support is low or the data are dense, massive probabilistic supports have to be computed. An improved version of PFIMoS, PFIMoS+ (Li et al., 2018), incorporates a heuristic rule to reduce the count of the probabilistic support computing that is not pruned by the bounds. PFIMoS+ receives an error parameter ω, that is the main factor conditioning the improvement achieved by PMFIoS+. This error parameter determines during how many sliding window updates, the computation of the probabilistic support is avoided. If ω is always 0, the PMFIoS+ algorithm will have the same performance as PMFIoS.

II.2.3 Mining closed frequent itemsets

In practice, the patterns obtained by frequent pattern mining algorithms can be redundant or non-relevant in obvious or subtle ways. Let p and q be two different patterns such that $p \prec q$ and both have the same or very similar support. Then, if we know that q is frequent, knowing that p is frequent does not add much information and could be considered redundant. Some works focus on looking for frequent closed patterns without computing all frequent patterns as a more efficient way, in both time and memory, of obtaining the same knowledge. A pattern is closed if it has higher support than every one of its superpatterns. Given all frequent closed patterns, we can infer all frequent patterns. Furthermore, if the frequency of all closed patterns is known, the frequency of every frequent pattern can be deduce.

Similarly to the way in which frequent pattern mining methods often maintain not only the frequent itemsets but also the subfrequent (or promissing) ones, frequent closed itemset mining algorithms monitor other selected itemsets, in addition to Frequent Closed Itemsets (FCIs), in order to be able to detect new itemsets when they become frequent (and/or closed). Moment (Chi et al., 2006) is an exact algorithm for mining closed frequent itemsets on a sliding window. This method uses a compact data structure called a *Closed Enumeration Tree* (CET) to keep all the itemsets needed at any moment, which include: (1) closed frequent itemsets, and (2) itemsets on the boundary between closed frequent

itemsets and the rest of the itemsets. Concretely, four types of nodes are distinguished in the tree: infrequent gateway node, unpromising gateway node, intermediate node and closed node. In addition, the transactions in the sliding window are stored in a FP-tree. To build a CET, a depth-first procedure visits the itemsets in lexicographical order. For a certain itemset, the FP-tree is consulted to obtain the support of the itemset. Then, the type of the node conditions if the algorithm further explores the node. For instance, if a node is found to be infrequent, then the algorithm marks it as an infrequent gateway node and it does not explore the node further but it still stores the support of the node because it will provide important information during a CET update when an infrequent itemset can potentially become frequent. Every time the window slides, Moment traverses the parts of the CET that are related with the new transaction added to the window and with the transaction that is to be removed from the window, and updates each related node. If the type of a node changes, re-exploring originally pruned branches (transaction addition) or pruning certain branches (transaction deletion) may be required. The addition algorithm will not decrease the number of nodes in a CET, in the same way that the deletion algorithm will not increase the number of nodes in a CET.

IncMine (Cheng et al., 2008) is an approximate algorithm for mining frequent closed itemsets. As Moment, IncMine follows a sliding window approach and it is aimed at reporting the frequent closed itemsets in the window. However, as opposed to Moment, IncMine does not process the transactions from the stream one by one but groups them in batches of size b. Thus, the window stores a set of w batches ($w \cdot b$ transactions). The two keys of IncMine are: (1) the notion of semi-FCIs, and (2) the inverted index structure. IncMine stores a superset of the FCIs, referred as semi-FCIs. The minimum support threshold for an itemset to be considered a semi-FCI progressively increases as the itemset is retained longer in the window. Let σ be the desired minimum support for an FCI, IncMine sets up a schedule of augmented supports $r(i)$ for $i = 1 \ldots w$, such that $r(1) < r(2) < \ldots < r(w-1) < r(w) = 1$. If a pattern in the ith batch of the window has a frequency lower than $r(i) \cdot \sigma \cdot i$, the itemset is removed from the current set of semi-FCIs C because it is considered unlikely that its frequency will reach $\sigma \cdot w \cdot b$ after $w - i$ more batches.

When a new batch of transactions arrives, IncMine incrementally updates C. Roughly speaking, it first mines the set of FCIs in the new batch $C2$, and then updates C based on $C2$ and according to a set of rules that also implementes the forgetting of the transactions in the oldest batch of the window. $C2$ is generated using an existing non-streaming FCI mining algorithm (Pasquier et al., 1999; Zaki and Hsiao, 2002; Wang et al., 2003b). To improve the efficiency of the update operations, an inverted index structure is used to store the semi-FCIs. To build such structure, the set of semi-FCIs over the last window L is partitioned according to the size of the semi-FCIs in L, so all the semiFCIs of the same size belong to the same partition. An array called FCI-array stores each partition. Each entry of an FCI-array stores: a semi-FCI; an assigned ID, which corresponds to the position of the semi-FCI in the array, and the approximate support of a semi-FCI computed over each time unit. A *garbage-queue* is also associated which each

FCI-array, so that the ID of every semi-FCI deleted from a FCI-array is pushed into the corresponding garbage-queue.

CloStream (Yen et al., 2009) employs two in-memory data structures, called *Closed Table* and *Cid List*, to mine frequent closed itemsets from data streams. *Closed Table* maintains the information of the closed itemsets. Thus, each entry of the table contains three fields: the closed itemset, a unique id assigned to the itemset and the support of the itemset. *Cid List* is used to keep for each single item, the list of closed itemsets that include the item. In addition, when a new transaction arrives, the algorithm also uses a hash table to temporarily store those itemsets to be updated. Every time a new transaction is received, CloStream creates a new empty temporal table *Temp*. Each entry of *Temp* will include an itemset and a *Closure Id* that points to the closed superset with the highest support. In a second phase, CloStream updates the support of the itemsets included in Temp. For each record in *Temp*, if the itemset was already included in *Closed Table* its support is increased by one. On the other hand, if the itemset was not included in *Closed Table*, the itemset is assigned an identifier and a new entry is added to *Closed Table* with a support value equal to one plus the support of the closed itemset indicated by *Closure Id*. All frequent closed itemsets can be found by scanning Closed Table once. CloStream does not incorporate any forgetting mechanisms (sliding window or decay factor). It generates the list of closed itemsets from the beginning of the stream. In some settings, this could derived in a really high number of closed itemsets to be maintained.

II.2.4 Rare itemsets mining

In some domains, patterns representing events that are unusual are considered more useful than frequent patterns. Examples of this are the detection of computer attacks or of fraudulent credit transactions. The input data for these problems (network logs and banking transactions) are often received in the form of flows (or streams). An itemset is considered a rare itemset if its support is lower than a certain threshold (minimum frequent support threshold). One of the challenges associated with rare itemset mining is the difficulty to differentiate between noisy itemsets and the actual rare itemsets.

SRP-Tree (Streaming Rare Pattern Tree) (Huang et al., 2012) is a tree-based approach for mining rare itemsets from data streams using a sliding window, which defines a noise filter threshold (minimum rare support) to discard the noisy itemsets. Thus, an itemset is considered to be rare in a window if its support in the window is below the minimum frequent support threshold but surpasses the minimum rare support. SRP-Tree relies on three main data structures: (1) a list to maintain the frequency count of every single item in the window; (2) an item list called the *Connection Table* to keep track of every single item in the window and the items that co-occur with them along with their respective frequencies; and (3) a tree structure that allows to capture the content of the incoming transactions, using appearance order as a canonical order to build the tree. The Connection Table is designed using a hash map which allows for $O(1)$ access. In a given window, for each incoming transaction, the algorithm first updates the list of

item frequencies. After that, the *Connection Table* and the tree are also updated. The transactions are considered to be organized in blocks (or batches). At the end of each block, SRP-Tree lists all the rare items and all the items they co-occur with which also surpass the minimum rare support, build the conditional FP-Tree of each of these items, and then uses each conditional tree and the corresponding item as arguments for FP-Growth. The union of the outputs from all these calls to FP-Growth is a set of rare item itemsets.

II.2.5 Top-k frequent itemsets

A variation of the problem of mining frequent itemsets is the problem of mining the top-k frequent itemsets. The main argument in favor of this problem refactoring is that it may be easier for the user to set a bound on the size of the result to be obtained, rather than specifying an appropriate minimum support threshold (Wong and Fu, 2006; Zihayat and An, 2014; Dawar et al., 2017). The number of highly useful patterns can be quite high so finding only the top-k patterns can be more attractive. However, this is done at the expense of adding complexity to the problem. The methods need to estimate the minimum support threshold that allows to find the k most frequent itemsets. This means that there is an extra dimension of guessing in the algorithms and, therefore, a source of error.

Two different methods for mining top-k frequent itemsets from data streams are proposed in Wong and Fu (2006). Both methods process the data stream in batches. The first one is based on the Chernoff bound. For every batch, the algorithm estimates the support threshold s_k in such batch. Then, based on s_k and on the Chernoff bound, the itemsets in the batch are distinguished between potential k-frequent itemsets and unpromising itemsets, and a local pool P_l is formed with the potential k-frequent l-itemsets in the batch. After that, the local pool P_l is combined with the global pool F_l and the algorithm updates the support of each entry in the global pool. If the maximum size of the global pool is reached, the unpromising itemsets are pruned according to the support threshold estimated based on all the transactions seen so far. The algorithm is an any-time response method that returns the top k frequent itemsets in the current global pool. The Chernoff-based algorithm assumes data independency. However, the authors introduce some techniques that allow the algorithm to handle data dependencies.

The second proposal is an adaptation of Lossy Counting (Manku and Motwani, 2002). It follows the same pool-based approach as the Chernoff-based proposal. The main difference is the criterion used to distinguish between potential k-frequent itemsets and unpromising itemsets. In this case, any itemset with a frequency of, at least, β, where β is the number of buckets in the batch, is added to the local pool P_l, and an entry of the global pool is considered unpromising if $f + \Delta \leq \lceil \frac{n}{w} \rceil$, where f is the estimated frequency, Δ is the maximum estimation error, n the number of transactions seen so far, and w the bucket size. Furthermore, adaptations of both Chernoff-based algorithm and Top-k Lossy Counting algorithm to incorporate a sliding window approach are also introduced in Wong and Fu (2006). Basically, this adaptations consist of keeping the local pools for all the batches in the current window. Every time the window slides, the supports of the entries

of the global pool are updated according to both the new batch and the batch that drops the window.

T-HUDS (Zihayat and An, 2014) addresses the problem of mining the top-k High Utility Itemsets (HUI) over sliding windows. The algorithm relies on a prefix tree structure to maintain an up-to-dated compressed version of the transactions in the sliding window, and two auxiliary lists that are used to estimate the support bound to distinguish between itemsets that are Potential Top-k HUIs (PTKHUIs) and the unpromising ones. A pattern growth approach similar to FP-Growth is used to identify PTKHUIs. Then, the transactions in the current window are scanned to obtain the exact utility of each PTKHUI, so the true top-k HUIs can be identified.

The same problem is addressed in Dawar et al. (2017), where a data structure called *iList* and an algorithm based on such data structure are proposed for mining top-k high-utility itemsets from a data stream. iList is an adaptation of utility-list (Liu and Qu, 2012), proposed for static transaction database scenarios. It captures the utility information associated with an itemset across windows by maintaining a FIFO (First In First Out) queue of the batches in the current window. Each batch contains a list of $\langle Tid, EU, RU\rangle$ tuples, where Tid is the transaction identifier which contains an itemset, EU is the exact utility of an itemset and RU is the remaining utility of an itemset. The iList data structure is built by scanning the sliding window twice. In the first scan, Transaction Weighted Utility (TWU) of items is computed. During the second scan, items in each transaction are sorted according to ascending order of TWU and an iList for each item is created. The iList for an itemset is generated by intersecting the iLists of individual items. The proposed algorithm is a single-phase method that obtains the top-k utility itemsets based on an iList data structure and does not generate any candidates in the mining process.

At the beginning of this section, we have defined patterns as entities that are present or absent with a frequency that deviates from the random. So far, we have identified patterns with itemsets. However, other combinatorial structures such as sequences and graphs are also often used to embody this broad definition of pattern.

II.2.6 Sequential pattern mining

In its most basic form, a sequence is an ordered lists of items: $S = \langle i_3, i_7, i_2, i_{10}, i_6\rangle$, where both $\langle i_7, i_{10}\rangle$ and $\langle i_3, i_7, i_6\rangle$ are subsequences of S. A generalization of this basic idea derives on every element of the ordered list being an itemset: $S = \langle I_1, I_2, \ldots, I_n\rangle$, where each I_i is a subset of the set of items $I = i_1, \ldots, i_n$. Sequential pattern mining allows the discovery of frequent sequences and can be useful to identify relations between itemsets. Nonetheless, it is a difficult task given the great size of the search space (Zaki, 2001). Indeed, only a few proposals for mining sequential patterns from data streams are found in the literature.

SMDS (Sequence Mining in Data Streams) (Marascu and Masseglia, 2006) is an

algorithm for mining frequent sequential patterns from web usage data streams that relies on a greedy clustering algorithm associated to an alignment method and on the prefix tree structure of PSP (Masseglia et al., 1998) for managing the frequent sequences mined. The algorithm processes the data stream in fixed-size batches and employs a prefix tree structure to store the approximate frequent sequential patterns mined from the incoming batches. Each navigation sequence in the batch is compared to each cluster and inserted in the one that, among those that meet a series of conditions, have the closest centroid to the sequence. If no such cluster is found, a new cluster is created. Every time a new sequence is inserted into a cluster, the centroid of the cluster is incrementally updated using the alignment technique presented in Kum et al. (2003), and then filtered according to parameter k to obtain the approximate sequential pattern. At the end of each batch, the filtered aligned sequence (or approximate sequential pattern) from each cluster, considered as a summary of the cluster, is inserted into the prefix tree. The method is provided with a logarithmic tilted-time window. An intermediate window system allows to merge windows when needed and tail pruning is implemented to delete the oldest records.

In Raïssi and Poncelet (2007), a method based on reservoir sampling is proposed to address sequential pattern mining over data streams. In reservoir sampling (Vitter, 1985) the probability of the insertion of a data point in the reservoir decreases as the data set length increases, which is a clear disadvantage for data stream contexts. Instead, the approach proposed in Raïssi and Poncelet (2007) uses an exponential bias function to regulate the sampling. Each data point is defined as a pair formed by a customer and its associated transaction. The algorithm starts with an empty reservoir of capacity $\frac{1}{\lambda}$, where λ is the bias rate of the exponential bias function. Each new data point from the stream is deterministically added to the reservoir by flipping a coin: the point is simply inserted into the reservoir or it replaces a customer and all its transactions. To bound the size of the list of transactions associated to each customer included in the reservoir, the method uses a sequence-based sliding window to maintain the most recent transactions for a given customer in the sample.

II.2.7 Frequent closed graph mining

A graph G is a pair composed by a set of nodes V together with a set of edges E among nodes. We say that a graph $G = (V, E)$ is a subgraph (or graph subpattern) of another graph $G' = (V', E')$ if $V \subseteq V'$ and $E \subseteq (V \times V) \cap E'$. When addressing frequent graph mining from data streams, each element of the stream is in itself a graph and the algorithms are aimed at mining the subgraphs of these incoming graphs that are frequent and closed.

Bifet et al. (2011) presents two coreset-based algorithms for mining the approximate set of closed frequent graphs over a sliding window: WinGraphMiner and AdaGraphMiner. Given a problem, a coreset of a set C can be defined as a small subset of C such that solving the problem on the coreset gives an approximate solution for the problem on C. Both WinGraphMiner and AdaGraphMiner exploit the concept of Δ-support in order to improve the time efficiency of the operations of adding and removing patterns from

the summary (or coreset). The Δ-support (or relative support (Bifet et al., 2011)) of a pattern can be calculated as its support minus the sum of the absolute supports of all its closed, proper superpatterns. Conversely, the absolute support of a closed pattern can be calculated as the sum of the Δ-supports of all its closed superpatterns (including itself). Hence, keeping the Δ-support of each closed pattern in the summary, to add a pattern p to the summary, we just need to add 1 to the Δ-support of p and we will be implicitly adding 1 to the (regular) support of all its subpatterns. The same logic applies for removals. The algorithm will need to compute the actual regular supports only when a query to output the frequent closed graphs is submitted.

WinGraphMiner employs a fixed-size sliding window. The algorithm maintains at all times a summary G that contains the approximate set of frequent closed subgraphs in the current window.Every time a new batch of graphs is received, it is mined for closed frequent graphs using the batch miner CloseGraph (Yan and Han, 2003), and transformed to the relative support representation. In addition, WinGraphMiner subtracts the coreset corresponding to the batch that drops from the sliding window.

AdaGraphMiner is an extension of WinGraphMiner that is able to adapt to changes in the stream. The algorithm is aimed at outputting the graphs that are frequent and closed in the current distribution. In Bifet et al. (2011) two different versions of AdaGraphMiner are presented. The simplest one monitors the total number of closed graphs. It uses ADWIN as change detector and shrinks the window if change is detected. The other version uses an individual ADWIN instance to monitor the support of every frequent closed subgraph. Thanks to this, the algorithm is more sensitive to changes in individual graphs and avoids keeping all the batches in a sliding window in memory. This second version experimentally requires less memory even though an ADWIN instance is maintained for every graph.

II.2.8 Mining rules

When the interest resides in the associations between items, beyond knowing that they co-occur with some frequency, rule mining can be highly useful. We can distinguish different categories of rule mining depending on the kind of associations extracted. Ratio rule mining, for instance, is aimed at capturing quantitative association knowledge. Using the classical shopping cart example, a ratio rule milk, diapers, beer = 1 : 2 : 1 can be translated as: "if a customer spends 1 amount on milk, then he/she is likely to spend 2 amounts on diapers and 1 amount on beer". Fan et al. (2009) presents a method to mine ratio rules at changing data streams in an incremental and adaptive way. Two key steps can be identified in the proposal: the detection of emerging trends and the actual mining of ratio rules. The algorithm uses the technique for data stream evolution diagnosis proposed in Aggarwal (2003) to detect partially coagulated intervals in the data distribution as emerging trend intervals. Then, the algorithm adopts an automated Incremental Principal Component Analysis (IPCA) to mine ratio rules at these emerging trends. In addition, a generalized multiple regression measurement is used to evaluate how good the generated

ratio rules are at each new incoming data point.

Association rule mining is a well-studied field which is aimed at extracting associations between variables in the form of production rules. An association rule R is an implication of the form $X \rightarrow Y$, where both X and Y are frequent itemsets and $X \cap Y = \emptyset$. Association rules are typically obtained by first mining frequent itemsets from datasets or data streams, then building the frequent rules from the mined frequent itemsets. This relegates the rule generation to a second place in an offline process. Since there is nothing essentially specific to the streaming setting in this offline rule generation process, in the streaming context, efforts have been mainly focused on the frequent itemset mining step.

Moreover, despite in real-world applications the streaming data sources often include quantitative attributes, most of the algorithms that have been proposed for mining frequent itemsets and association rules in data streams can only handle categorical data. In order to deal with continuous attributes, two different strategies were initially explored for generating quantitative association rules: (1) discretize the features and then deal with them in a purely qualitative fashion (Wang et al., 1998), and (2) using an interval-based representation (Martínez-Ballesteros et al., 2011). Later on, fuzzy modeling was introduced. The use of fuzzy sets allows to build highly legible models, and to avoid the loss of information that can be derived from discretization and the unnatural boundaries caused by interval-based representation (Dubois et al., 2006; Hong et al., 2001).

FFI-Stream (Chen et al., 2010) is a method for mining fuzzy association rules from data streams over a sliding window. This proposal employs clustering to determine the fuzzy sets. Concretely, some of the techniques included in SWEM (Dang et al., 2009) are used. First, it applies the micro-clustering stage ensuring enough information about the data distribution is collected and then, in the second stage, the macro-clusters and the correspoding fuzzy sets are obtained. In addition, Selectively Updating Mechanism and Projected Summaries are proposed to update the fuzzy sets dynamically. Due to the possibility of concept drift, Membership Function Bias measure is introduced to evaluate the membership function in each sliding window and detect significant changes. To find frequent itemsets with fuzzy sets, FFI-Stream adapts UF-Streaming (Leung and Hao, 2009), which was, as we detailed before, originally proposed for mining frequent itemsets in uncertain data streams. The existential probability from data uncertainty context is altered into the membership degree in the context of mining frequent itemsets with fuzzy sets. When the user submits a query, FFI-Stream finds the frequent itemsets in the current window and generates the association rules.

However, if we are interested on extracting the interesting associations among the forming attributes of the data and keep track of their evolution as the dynamics of the data flow change, the typical offline rule generation is unpractical. In the association stream mining field, the mined rules have to be present immediately and adapt to the changing dynamics of the data stream. Although association stream mining is closely related to mining frequent itemsets in data streams and both share similar challenges (single pass limitation, memory constraints, handling concept drift (Bifet et al., 2010a)), methods for

the latter do not identify the whole pattern while the process is running and, therefore, are usually ill-suited for handling association stream problems.

Fuzzy-CSar (Sancho-Asensio et al., 2016) is an online genetic fuzzy system designed to mine interesting association rules from streams of data in a single step, i.e., it does not build any list of frequent itemsets. Instead, Fuzzy-CSar directly evolves the set of fuzzy association rules. Fuzzy-CSar-AFP (Ruiz and Casillas, 2018), one of the proposals of this thesis, extends Fuzzy-CSar to be able to handle adaptive fuzzy partitions, better adapt to the requirements of real-world environments and increase the diversity of the mined rules.

II.3 Semi-Supervised Learning in data streams

II.3.1 Label scarcity in data streams

In many different fields (from Web mining to bioinformatics), it is significantly easier to get unlabeled than labeled data since it requires less effort, expertise and time consumption. Labeling can be costly due to different reasons ranging from great amount of human labor to required expensive, intrusive or destructive laboratory tests. In this context, supervised learning is limited to using labeled training data to build a model.

The conditions associated to data streams (high arrival rate, potential infinite length, changing distribution...) favor the existence of many real-world problems where it may be unreasonable or unaffordable to require true labels for all incoming instances (Matuszyk and Spiliopoulou, 2015; Noorbehbahani et al., 2017; Tang et al., 2017; Iosifidis and Ntoutsi, 2019). Hence, proposals prepared to deal with label scarcity in data streams have gained attention over the past few years. Various types of approaches have been explored. Two of the main fields of research are: Active Learning (AL) and Semi-Supervised Learning (SSL). In both cases the proposals assume that, in addition to unlabeled data, a certain (although maybe small) amount of labeled data will be readily available (SSL) or could be requested (AL).

Active Learning (Masud et al., 2010; Žliobaitė et al., 2011; Dyer et al., 2013) is based on the premise that it is not affordable to label every instance. The main aim of AL is to reduce training cost selecting those key instances whose labels - if available - would provide the higher learning profit. Thus, these techniques require few labeled instances for training but they do not use unlabeled data for training the classification model. The most important limitation of AL is, however, the implicit requirement that the true label be provided for any instance for which the algorithm requests it; this is not always a realistic assumption. On the other hand, the semi-supervised learning paradigm (Zhu and Goldberg, 2009) is grounded in the assumption that unlabeled examples can also contribute to the learning process. SSL is an extension of unsupervised and supervised learning by including additional information typical of the other learning paradigm. Hence, both unlabeled and labeled data are involved in training and work together to improve

system performance.

Offline semi-supervised classification methods are often classified depending on the assumptions they make about the relationship between the underlying distributions of labeled and unlabeled data. In broad terms, two different assumptions are distinguished:manifold assumption and cluster assumption. The first assumption is met if data lie approximately on a manifold of lower dimensionality than the input space. The cluster assumption states that similar examples should be likely to be of the same class. Different type of approaches can be used to implement any of these assumptions. For instance, the manifold assumption is commonly implemented through graph-based models, and generative models or semi-supervised support vector machines are examples of models based on the cluster assumption. Furthermore, there are also other semi-supervised techniques that do not make any specific assumptions about the input data, e.g., self-labeled techniques. Over the past few years, proposals based on several of the aforementioned approaches have been made for semi-supervised classification in data streams, such as graph-based (Bertini et al., 2012), cluster-then-label (Castellano and Fanelli, 2016) or self-labeled (Wu et al., 2006; Loo and Marsono, 2015; Feng et al., 2016; Wang and Li, 2018) models.

Table II.2 shows a survey of the proposals in the literature and categorizes them according to the following characteristics: (1) Strategy—indicates if there is only one single classifier or if an ensemble of classifiers is built—, (2) Type—references the type of classification method used—, (3) Clustering—whether clustering techniques are employed or not—, (4) Scheme—the incoming instances can be process independently as they arrived or grouped in chunks or batches—, and (5) Experiments—the experimental setup may include synthetic environments, real-world datasets or a combination of both.

As shown in the table, data stream classification proposals can be categorized according to the way they process the incoming flow of data, distinguishing between: chunk-based and instance-based proposals. In the first case, the data stream is divided into batches or chunks, normally of fixed size, and the algorithm processes each of them as a unit. In the case of instance-based (also referred as *online*) proposals, each instance is processed individually at the moment of its arrival, i.e., the model can be updated every time a new instance is received. Strategies for chunk-based learning may not be applicable to online settings (e.g., if they rely on offline learning algorithms).

II.3.2 Chunk-based proposals

Within the field of semi-supervised learning in data stream, there are some proposals that explore the Extreme Verification Latency (EVL) case. These proposals take to the limit the constraint of the existence of a certain delay in the reception of the labels. In EVL the duration of this lag is set to infinity. Hence, the proposals work on the assumption that the only labeled data they receive are part of a starting set which allows the system to be initialized, and that afterwards they will just receive a flow of unlabeled data. In Dyer et al. (2013) and Ferreira et al. (2019) two EVL approaches are proposed. Both of

Table II.2: Characteristics of semi-supervised classification methods for data streams.

Reference	Strategy	Type	Clustering	Scheme	Experiments
(Wu et al., 2006)	-	-	Yes	Chunk-based	Mixed
(Masud et al., 2008)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Yu et al., 2008)	Single-model	SSC	Yes	Chunk-based	Real-world
(Woolam et al., 2009)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Zhang et al., 2010)	Ensemble	-	Yes	Chunk-based	Real-world
(Li et al., 2010)	Single-model	Tree	Yes	Online	Mixed
(Xu et al., 2011)	Single-model	Tree	Yes	Online	Mixed
(Ditzler and Polikar, 2011)	Ensemble	-	Yes	Chunk-based	Synthetic
(Bertini et al., 2012)	Single-model	Graph-based	No	Chunk-based	Mixed
(Masud et al., 2012)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Zhang et al., 2012)	Single-model	SVM	Yes	Chunk-based	Mixed
(Wu et al., 2012)	Single-model	Tree	Yes	Online	Mixed
(Ahmadi and Beigy, 2012)	Ensemble	Tree	No	Chunk-based	Mixed
(Dyer et al., 2013)	-	-	No	Chunk-based	Synthetic
(Loo and Marsono, 2015)	-	-	Yes	Chunk-based	Mixed
(Castellano and Fanelli, 2016)	Single-model	SSC	Yes	Chunk-based	Real-world
(Haque et al., 2016)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Hosseini et al., 2016)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Shao et al., 2016)	Single-model	Lazy learning	Yes	Online	Mixed
(Feng et al., 2016)	Single-model	Lazy learning	No	Chunk-based	Mixed
(Tang et al., 2017)	Single-model	SSC	Yes	Chunk-based	Real-world
(Noorbehbahani et al., 2017)	Single-model	Lazy learning	Yes	Online	Real-world
(Wang and Li, 2018)	Ensemble	Tree	No	Chunk-based	Mixed
(Qin and Wen, 2018)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Ferreira et al., 2019)	-	-	No	Chunk-based	Mixed
(Wen and Liu, 2020)	Ensemble	SSC	Yes	Chunk-based	Mixed
(Din et al., 2020)	Single-model	Lazy learning	Yes	Online	Mixed

them are designed for gradual drift scenarios and follow the same framework structure where an offline semi-supervised algorithm is trained on the starting set (which contains labeled and unlabeled data) and is used to label the remaining upcoming samples divided in equal-sized chunks. Once the samples in a chunk have been labeled, a filtered is applied to select a subset of optimal instances that will pass as labeled training data for the next chunk of unlabeled samples. The base semi-supervised classifier is retrained each time a new chunk is received. However, both approaches differ in the strategy used to select the optimal instances from each chunk.

COMPOSE (Dyer et al., 2013) uses geometric techniques in order to map core support regions. It first builds an α-shape (a generalization of convex hull) for each class, where an α-shape is a set of connected faces (simplexes) creating a hull that describes a finite set of points at a certain level of detail (defined by the parameter $\alpha \geq 0$). Then, the algorithm compacts (shrinks) these shapes and extracts the instances from the compacted shape that represent the geometric center (core support region) of each class distribution. This compaction of the α-shapes is achieved by iteratively removing a layer of simplexes from the edges of the α-shape, until a certain compaction percentage is reached. Each time a layer of simplexes is removed, the number of instances in the compacted α-shape is decreased. The data remaining in the α-shape after compaction are considered the core supports and will be used as labeled training samples for the next chunk.

In the case of AMANDA (Ferreira et al., 2019), a density-based algorithm measures the relevance of the classified samples and weights them, and only those considered to be the most representative ones are selected. The aim is, as in COMPOSE, to identify which samples are included in the core region of the existing class distributions. Thus, the kernel density estimation method employed calculates density curves that take into account the distance of each point from a central value (kernel). Each instance is associated with a probability density function value, which indicates the denser instances in the distribution. The α-most-dense instances from the set of weighted instances are selected. Two different versions of AMANDA are proposed in Ferreira et al. (2019): AMANDA-FCP (AMANDA Fixed Cutting Percentage) and AMANDA-DCP (AMANDA Dynamic Cutting Percentage). In the case of AMANDA-FCP, α is a user-defined parameter with a fixed value, while AMANDA-DCP dynamically calculates α for every batch based on the distance between the past distribution and the distribution in the new batch.

Other data stream semi-supervised approaches, outside EVL area, assume that labeled and unlabeled samples will alternate throughout the flow of data. Some of these approaches adapt traditional semi-supervised paradigms to stream environments, such as self-labeled methods. These methods iteratively seek to obtain one (or several) enlarged labeled set(s), based on their most confident predictions to properly represent the training set. Self-labeled techniques can use one or more classifiers (of the same or different type of learners) during the enlarging phase of the labeled set. They also can be distinguished between single-view or multi-view depending on how the input feature space is taken into consideration by the self-labeled technique. Thus, the most basic self-labeled technique is self-training. In self-training, the training set is expanded using the most-confident

predictions of the uniquely used classifier, which uses the full input feature space for training. Therefore, a hypothesis is initially learned from labeled data; such hypothesis is then used to classify unlabeled data, and the most confident unlabeled data, along with the labeled instances, are added into the training set. The hypothesis is repeatedly refined with the updated training set. In Wu et al. (2006) and Loo and Marsono (2015) self-training cluster-based methods are proposed. In both cases, k-prototypes (Wu et al., 2006) and k-means (Loo and Marsono, 2015) clustering approaches are used to expand labels to unlabeled instances and select the most confident ones to join the training set for retraining or updating the classifier. The proposal presented in Wu et al. (2006), bases the instance selection in the comparison between the label predicted by the k-prototypes and the one inferred by a supervised classifier. The method assumes the existence of an initial labeled training set that allows to initialize this classifier, which would be re-trained every time new confident instances are added to the training set.

The main drawback of self-training is the possibility of error propagation since the own predictions of the classifier are later used to train it (He and Zhou, 2011; Zimmerann et al., 2014). To avoid this problem co-training was introduced (Blum and Mitchell, 1998), whose intrinsic intuition is that different classifiers will make different mistakes and, therefore, they can learn from each other. In co-training two classifiers are trained on two different attribute subsets (multi-view), which are supposed to be sufficient and redundant, and each classifier labels unlabeled data. The most confident predicted data from one classifier are used to improve the training set of the other classifier, and each classifier refines its hypothesis with the updated training set. Co-training avoids the error propagation risk associated with self-training but verifying that the assumptions made by co-training are met in a real-world problem is not trivial. These assumptions are: (1) the view used to train each classifier is sufficient to fulfill the learning task, and (2) both views are independent given the class. Nevertheless, co-training have achieved good performance in certain real-world problems where the mentioned conditions were not ensured.

Co-training techniques are employed in Feng et al. (2016) to select the most confident exemplars that will be used to augment the training set. An incremental self-representative data selection strategy is first applied to find the most representative exemplars from the sequential data chunks. Then, these exemplars are labeled by means of co-training. The features of each sample are randomly split in two views and k-Nearest Neighbors classifiers are used to estimate the labels of the exemplars. The most confident ones along with their predicted labels are added to the training set. Based on this training set, the testing samples are classified using kNN. In Ahmadi and Beigy (2012), the authors propose an ensemble learning method in which, for each classifier in the ensemble, the majority vote of other classifiers is used to label a subset of unlabeled instances that is then employed to update the classifier. This idea is further developed in Wang and Li (2018) with the proposal of SCo-Forest, which combines co-training paradigm with Random Forest (Breiman, 2001) for data streams. Furthermore, it employs ADWIN2 (Bifet and Gavaldà, 2007) for concept drift detection. For every data chunk received, the concept drift detector is adopted. If change is detected, the low accuracy classifiers in

the ensemble are replaced by new ones. Otherwise, the ensemble is kept in its current state. To build new classifiers, an online bootstrap sampling method is applied to sample the labeled set in the current window and generate the training set for the new random tree. In addition, SCo-Forest tries to improve the performance of the individual random trees by augmenting the labeled set with the most confident unlabeled examples. Thus, for each random tree h_j, the algorithm employs the concomitant ensemble H_j (generated by excluding h_j from the global ensemble) to select the most confident examples in the unlabeled set, which are added to the labeled training set used to update h_j.

Nonetheless, self-labeled techniques are not the only traditional offline semi-supervised strategies whose adaptation to data stream has been explored. Bertini et al. (2012) presents a graph-based semi-supervised approach that extends the offline classifier based on the k-associated optimal graph. This approach maintains a dynamic principal graph that is repeatedly updated over time. For each new data chunk, the system generates the k-associated optimal graph and some of its components are added to the principal graph. Labeled and unlabeled examples constitute the graph vertices and the similarity measurements between vertices correspond to the graph edges. Each vertex v_i connects to its k-nearest neighbors whose classes are not different from the class of v_i. This means that vertices corresponding to labeled samples connect to those, among their k-nearest neighbors, which belong to their same class or are unlabeled, and unlabeled samples connect to all their k-nearest neighbors without considering their classes. The graph can be seen as a set of groups of connected samples (components). Since unlabeled vertices can be connected to labeled vertices of any class, components with more than one class may be formed. These components are split by cutting edges based on the purity of the vertex. The cutting process finishes when the component is separated into single class components. The class labels are spread within every component. Finally, the purity measure is calculated for all components and each component of the current k-associated graph is compared to the corresponding components in the principal graph. If the new component increases or maintains the purity, it will substitute the old ones. The process continues by increasing k and generating new graphs until the number of components matches the number of classes.

Cluster-then-Label methods, closely related to generative models, have also been explored for semi-supervised classification in data streams. Instead of using a probabilistic model, these techniques apply a previous clustering step, and then they label each cluster with the help of labeled data. The proposal in Castellano and Fanelli (2016) applies SSFCM (Pedrycz and Waletzky, 1997), an offline semi-supervised fuzzy k-means algorithm, to cluster the instances on every chunk. The cluster prototypes obtained from a chunk are added as pre-labeled data points to the next chunk, and SSFCM is applied again to derive k clusters and k labeled prototypes. After every run of SSFCM, data included in the k-th cluster are predicted as belonging to the class of the k-th prototype. Clustering techniques allow to take advantage of unlabeled examples to try to discern the true class boundaries, although they present some limitations. In high dimensional space, sparse data can cause the proximity measured between instances by distance metrics to be hardly

ever meaningful (Aggarwal et al., 2001; Beyer et al., 1999). Furthermore, the approaches based on k-means or similar are restricted to spherical clusters.

Ensemble techniques present some appealing characteristics to deal with data streams. Hence, as well as in the case of supervised classification, several ensemble methods have also been proposed for semi-supervised classification in data streams. In Ditzler and Polikar (2011), the labeled data are used to build the base classifiers. The voting weights of such classifiers are determined based on the distances between Gaussian mixture model components trained on both labeled and unlabeled data in an environment with limited gradual drift.

Ensemble proposals for semi-supervised classification in data streams often combine the ensemble scheme with cluster-based techniques. This combination was first explored by Masud et al. (2008); Woolam et al. (2009); Masud et al. (2012). In Masud et al. (2012), they propose an ensemble classification model where each base model is built as a set of micro-clusters using semi-supervised impurity based k-means. The ensemble is composed of L different models, each of which has been built on a different data chunk. Each time a new data chunk is received, a new base model is generated. The first step to build such model is to apply clustering on the last data chunk generating k clusters. Then, each of these k clusters is divided into *pure* micro-clusters, where a pure micro-cluster contains unlabeled instances or labeled instances of only one class. This new set of micro-clusters is combined with the *labeled* micro-clusters (those corresponding to true labeled instances) from the previous r chunks, and a label propagation technique is adopted to classify the unlabeled micro-clusters. Due to concept evolution, a new particular class, which is not present in any of the current L models in the ensemble, may be present in the new model. If this happens, the ensemble is refined by injecting micro-clusters from the new model into the models currently included in the ensemble. Finally, the best L models from the $L + 1$ existing ones are selected to form the new ensemble. The decision is based on the accuracy of the models on the labeled instances of the present chunk.

Impurity based k-means clustering is also exploited by SAND (Haque et al., 2016), which also considers the concept evolution scenario. SAND requires an initial *warm up* training set. Once the *warm up* period is over, each instance received is checked to determine whether it is an outlier. If the instance is not considered an outlier, the instance is labeled based on the majority vote of the classifiers in the ensemble. Otherwise, it is temporarily stored in a buffer. When the buffer reaches a certain size, a novel class detector is applied to determine if the instances in the buffer are labeled as belonging to a new class or if, on the contrary, they should be labeled by the existing ensemble. Each model in the ensemble is represented by a collection of k pseudopoints. Each of these pseudopoints summarizes a cluster and can be seen as a hypersphere in the feature space. The class predicted by a specific classifier in the ensemble for a sample x, will be the most common class in the cluster represented by the closest pseudopoint to x. SAND delimits the size of each chunk based on the detection of a change in the distribution, instead of assuming fixed-size chunks. This change detection is based on the confidence of the predictions of the ensemble. When change is detected, a chunk boundary is set and the

ensemble model needs to be updated to adapt to the changed concept. To that aim, a new model is trained and inserted in the ensemble. On the other hand, if no significant change in the confidence scores is found, the current ensemble is retrained and the size of the current chunk continues increasing until the maximum size (based on available memory) is reached, and the model has to be updated and the chunk reinitialized.

Hosseini et al. (2016) proposes SPACS, an ensemble of cluster-based classifiers focused on detecting recurring concepts and exploiting information on previously known concepts. Each classifier in the ensemble has an associated weight that is dynamically updated based on the accuracy of its predictions on the labeled incoming samples. Every time a new data chunk is received, the classifier with the maximum weight is chosen to sequentially classify the instances in the chunk. SPACS assumes data chunks to be single-concept. A similarity-based method is proposed for detecting the recurring concept-drifts. If no recurring concept-drift is detected and the pool of classifiers is not full, a new classifier is built on the current chunk and added to the pool. Otherwise, the most similar classifier is incrementally updated. A extended version of SPACS, which incorporates local component replacement to update the classifier pool in case the detected concept-drift is not recurrent, is presented in Qin and Wen (2018). Along the same lines, another ensemble of cluster-based models is also proposed in Wen and Liu (2020). In this case, a BIRCH-like method (Zhang et al., 1996) is used to build the base models. To perform concept detection, local structure mapping strategy, based on Gao et al. (2008), is combined with semi-supervised Bayesian method.

In Zhang et al. (2012), a framework to categorize the incoming chunks according to the labeling ratio and the potential presence of concept drift is proposed. Depending on the categorization of the chunk a different learning scheme is adopted, although offline algorithms are applied on each chunk in any scenario. Four different learning cases are distinguished: (1) labeling rate is high and concept drift probability is low, (2) both labeling rate and concept drift probability are low, (3) both labeling rate and concept drift probability are high, and (4) labeling rate is low and the concept-drift probability is high. In the first case, a generic SVM model is trained using only the labeled data that belong to the target domain. In the second scenario, both labeled and unlabeled samples from the target distribution are used to train a semi-supervised SVM model. In the third case, three different types of samples (both labeled and unlabeled from the target domain, and labeled samples from a similar domain) are used and a novel transfer semi-supervised SVM model is presented. Finally, all samples are used in the fourth case to train a model that combines relational k-means model (Zhu and Jin, 2009) and the transfer semi-supervised SVM proposed for the third scenario.

Meanwhile most of the previously mentioned works (except for EVL proposals) consider semi-supervised scenarios where there are both unlabeled and labeled instances in every chunk, Zhang et al. (2010) contemplates a different semi-supervised scenario where instances in a chunk might be all labeled or all unlabeled. An ensemble model is constructed by combining classifiers and clusters together. Each time a new data chunk is received, the incoming examples are clustered. If the chunk is labeled a new base classifier

is also built. Class label of each cluster is inferred through a label propagation method that takes into account both class labels from classifiers and clusters internal structure. A weighted average mechanism is employed to combine together all classifiers and clusters for prediction.

Chunk-based approach can offer some advantages since each time the models are trained or updated it is done based on a subset of data instead of on a single instance but they can also present some important drawbacks. The time for retraining the model (or training new base models) is highly dependent on batch size. Too large chunks may result on high learning times while too small chunks may reduce model reliability. Moreover, most of the chunk-based models assume that every batch contains both labeled and unlabeled data, therefore, linking chunk size and arrival frequency of labeled data.

II.3.3 Online proposals

Instance-based approaches are significantly less frequent than chunk-based ones, and most of them employ clustering techniques. Clustering Feature Decision Tree (CFDT) (Xu et al., 2011) extends the supervised method VFDT (Domingos and Hulten, 2000) by combining micro-clustering and clustering feature techniques (Zhang et al., 1996) on tree leaves to maintain the statistics needed for the splitting heuristic evaluation and to conduct the label induction process on test samples. Each leaf of CFDT comprises at most k entries, where k is the number of classes. Two main parameters affect the structure of the leaves: a threshold that defines the maximum radius of the micro-clusters and the maximum number of micro-clusters per entry. Since CFDT works in a pure online way, each instance is inspected only once. First, the example traverses the tree from the root to a leaf. If the example is labeled, an entry whose class label matches the one of the example is chosen, and then the micro-cluster with the closest centroid is selected. On the other hand, if the example is unlabeled, the micro-cluster whose centroid is the closest from all the entries in the leaf is selected. If the selected micro-cluster can incorporate the example, its clustering feature vector is updated. Otherwise, a new micro-cluster is generated based on the example. When a sufficient number of examples have been seen at a certain leaf, the heuristic function is calculated to decide the best cut-point. CFDT uses a recursive binary partition strategy and applies an exhaustive method at each stage of this process, i.e., all attributes and all possible cut-points for each attribute are evaluated. Regarding classification, the label predicted for an example is the class label of the nearest micro-cluster in the corresponding leaf.

REDLLA (Li et al., 2010) and SUN (Wu et al., 2012) also combine incremental decision trees and clustering. In both schemes, all instances in each leaf of the tree are clustered and majority-class method is used to infer the labels of unlabeled instances. If the number of examples in a leaf exceeds a limit, split-test is evaluated in a heuristic way based on information gain and Hoeffding bounds inequality (similarly to CVFDT (Hulten et al., 2001)). k-means (Li et al., 2010) or k-modes (Wu et al., 2012) clustering is periodically used to produce concept clusters at leaves, which are used to distinguish

potential concept drifts from noise. The new set of concept clusters is compared with past concept clusters to detect concept drifts. The estimation of the deviation between both sets of clusters is based on the radius of each set of clusters and on the average distance between them. Based on the values of these measures, three different possible scenarios are distinguished: (1) a potential concept drift is considered, (2) the deviation between concepts is considered noise, and (3) a true concept drift is considered. In the case of SUN, if a potential concept drift is considered, the new set of clusters is incorporated into the historic set of concept clusters. In case the deviation is considered as noise, the historic set of clusters is not modified and the new set is discarded. If an actual abrupt drift is detected, the historic clusters are replaced by the new ones. Meanwhile, when a potential or actual drift is detected by REDLLA, the method judges whether the new concept is actually a recurring concept. If the concept is brand new, the current clustering is added to a *concept list*. Otherwise, the current clustering is integrated into the historic set of concept clusters.

An incremental semi-supervised stream-based Intrusion Detection System (IDS) is presented in Noorbehbahani et al. (2017). The system includes two different learning phases: offline and online. First, in the offline phase, an initial set of clusters and an initial classification model are generated based on an initial labeled training set. The classification model is composed of a set of prototypes. During the online phase, the set of clusters is incrementally updated with the new data. Labeled samples are also used to update the prototypes of the classification model. During this phase, the system can label new instances by means of the set of prototypes and the nearest neighbor classification method. The approach assumes that future labeled training sets could be available for periodical applications of the offline phase, generating up-to-date models.

In Din et al. (2020), an algorithm that dynamically maintains a set of micro-clusters is proposed for semi-supervised classification on evolving data streams. This method has three main parts: (1) the initialization of the learning model, (2) classification, and (3) online data maintenance. The proposal assumes the existence of an initial training set that is used to initialize a learning model based on micro-clusters. Afterwards, a kNN classifier is used to predict the class label for every incoming instance. Any instance (labeled or unlabeled) received is employed to update the model. If the instance is labeled, it is used to assess the prediction performance of the model and, based on that, to infer the reliability of neighboring micro-clusters. Moreover, the instance is added to an existing micro-cluster or, if needed, a new micro-cluster is created and incorporated to the model. During the online maintenance phase, micro-clusters with low reliability or considered outdated are removed from the model so it can evolve along with the concepts in the incoming stream. A similar approach, which is also based on the application of kNN classifier on micro-clusters, had been previously presented in Shao et al. (2016).

Chapter III

CLAST: Learning rules for classification in data streams

III.1 Introduction

Research efforts on data stream classification have mainly focused on improving the accuracy of the predictive models, disregarding other aspects. This has led to the development of models with very limited descriptive power. Considering a data stream classification approach which assumes that all the examples coming from the stream are labeled, one may wonder how much sense it makes to focus exclusively on the accuracy of a predictive model which aspires to be, in the best case, hardly as accurate as the entity (human, community, machine...) that has been labeling each data in the first place.

On the other hand, a descriptive classification model would be able to provide insights on the problem and on how it is determined to which class belongs an example. The model still needs to predict accurately, otherwise we would be dealing with a model that does not well represent the reality of the problem. Here, we explore the use of rules as knowledge representation in data stream classification.

The chapter is devoted to present CLAST (CLAssification in data STreams), a system that works under a supervised learning paradigm to address data stream classification. The algorithm builds and dynamically maintains a population of rules, which jointly represent a solution to the classification task. The examples are processed one by one as they arrive, without need to store them or repeat the full learning process on all the observed examples. The only information needed in memory is the set of rules itself, the rules contain a series of parameters sufficient for evolving the population.

The remainder of this chapter is composed of two sections: Section III.2 gives a detailed description of the proposed CLAST algorithm, and Section III.3 presents the different experiments along with their results.

III.2 CLAST: CLAssification in data STreams

CLAST operates in two different modes: *exploration* and *exploitation*. Figure III.1 schematically represents the functioning of the system during a *exploration* (or training) interaction. In exploration mode the algorithm aims at evolving a maximally general rule population that minimizes the prediction error. The algorithm makes use of a set of histograms, which are incrementally updated as new data are received, to summarize data distribution information. On the other hand, in the exploitation mode the algorithm uses the knowledge harbored by the evolved population to infer the class of unlabeled test examples. Thus, every rule in the population that matches the test sample with a sufficient degree casts a weighted vote, and the most voted class is predicted.

This kind of ensemble design in which each rule works in a cooperative way with the rest of the population to build a solution is specially suitable for data stream environments. The easiness to add, remove or replace individual classifiers (rules) allows to incrementally evolve the ensemble (population) as new data are received in a very natural and efficient way.

The algorithm functions in a completely online fashion. Indeed, *CLAST* meets all the main requirements that, according to Bifet et al. (2009b), a classification algorithm for data stream must meet. The following can be considered the most significant ones: First, process one example at a time and inspect it at most once. Each example must be accepted in the order in which it is received and once inspected or ignored, the example is discarded and never retrieved. Data streams typically have very high incoming rates and, therefore, there is not enough time to reprocess examples several times. Second, memory usage must be limited since memory will be easily exhausted without limiting its allocation due to the potentially infinite nature of data. Third, the training complexity must be linear to the number of samples. Fourth, the system must be able to perform classification at any time, i.e., the induction model can be applied at any point between training examples.

III.2.1 Knowledge representation

As mentioned, the algorithm evolves a population [P] of classifiers that jointly represent the solution to the classification problem. Each classifier is composed of a weighted fuzzy classification rule, whose condition is in *conjunctive normal form*, and a set of parameters. Hence, the structure of the rule can be represented as:

$$R_k : \textbf{IF } X_1 \text{ is } \widehat{A_1^k} \text{ and } \ldots \text{ and } X_n \text{ is } \widehat{A_n^k} \textbf{ THEN } c^k \textbf{ WITH } w^k \qquad \text{(III.1)}$$

where $\widehat{A_i^k}$ is the fuzzy set used for the ith input variable in the kth rule; c^k, in the consequent of the rule, is the class the rule itself advocates, and the weight w^k in [0,1] signifies the soundness with which the rule advocates class c^k. This kind of rules which include a weight in the consequent are known as fuzzy rules of type II (Cordón et al.,

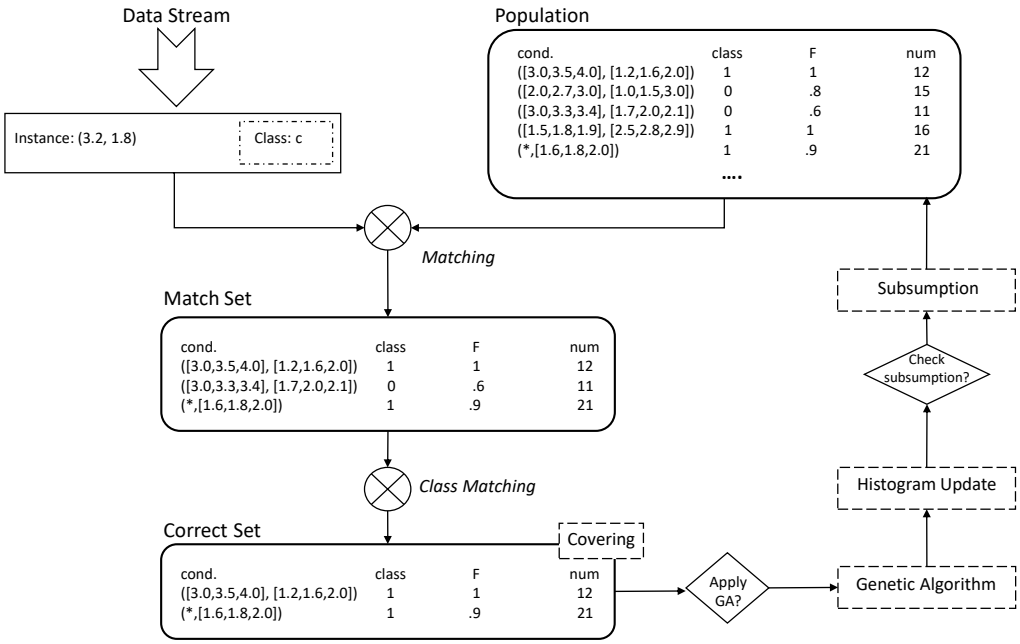


Figure III.1: Schematic illustration of the learning interaction of CLAST.

2001). The four main parameters that form part of every classifier are: 1) fitness F, which estimates the accuracy of the rule; 2) correct set size cs, the average size of the correct sets in which the classifier has been included; 3) experience exp^k, keeps count of the number of times fitness has been updated, and 4) numerosity num which denotes how many copies of the rule there are in the population.

The fuzzy sets employed in the condition of the rule make a direct use of fuzzy variables instead of using linguistic terms. Thus, each fuzzy rule presents its own semantics, i.e, the variables take different fuzzy sets as values instead of linguistic terms from a global term set. In this case, triangular-shaped fuzzy sets are used and, therefore, each variable in a rule k is represented by three continuous values a_k (left vertex), b_k (middle vertex) and c_k (right vertex). Since Radial Basis Function (RBF) Networks are functionally equivalent to certain types of fuzzy systems (Jang and Sun, 1993) and considering that the membership function of the fuzzy sets used is represented by a radial basis function, such fuzzy sets can be referred as RBF fuzzy sets (Shimojima et al., 1995). Concretely, the membership function $\mu_{A^k}(e_i)$ of the input sample e_i to the fuzzy set A_i^k (or matching degree between e_i and A_i^k) is computed as:

$$\mu_{A^k}(e_i) = \begin{cases} \dfrac{e_i - a_k}{b_k - a_k} & \text{if} \quad a_k \leq e_i < b_k \\ \dfrac{c_k - e_i}{c_k - b_k} & \text{if} \quad b_k < e_i \leq c_k \\ 0 & \text{otherwise} \end{cases} \tag{III.2}$$

This type of fuzzy systems, which are equivalent to fuzzy graphs (Alcalá et al., 2001; Zadeh, 1965), allows to ignore the restriction imposed when using linguistic terms by which the membership function in each fuzzy rule must belong to a common set (Alcalá et al., 2001). Furthermore, it also enables the algorithm to deal with both continuous and categorical variables in a very direct way. In the case of categorical variables singleton fuzzy sets are used, i.e., the three vertices of the fuzzy set are assigned the same category of the variable, being $\mu_{A^k}(e_i) = 1$ if e_i represents the same category as the vertices and $\mu_{A^k}(e_i) = 0$ otherwise. Moreover, RBF fuzzy sets allow generalization on their own when dealing with continuous variables, since each variable can be represented by a different fuzzy set with different vertices and shape.

On the other hand, since no global semantic is used, these fuzzy sets cannot be linguistically interpreted. RBF representation allows maximum flexibility by permitting tuning each individual fuzzy set of each rule thus resulting more complex and less interpretable. Nevertheless, interpretability is a controversial and subjective concept that is usually not guaranteed by just producing simple fuzzy rule sets since, in these situations, explanation capability may be degraded (Ishibuchi et al., 2009). Moreover, this structure allows the model to be more flexible. Despite the deriving process is more complex due to the higher freedom degrees and, therefore, this additional flexibility does not ensure the obtainment of more accurate results (Alcalá et al., 2001), it is expected to do so.

As mentioned before, the system keeps a set of histograms that summarize data distribution information that may be useful during the evolution process of the population. The system maintains a histogram per class and variable which is updated according to each new sample received. In a problem with n input variables $\{X_1, ..., X_i, ..., X_n\}$ and m different classes $\{c_1, ..., c_j, ..., c_m\}$, $n \times m$ histograms H_{ij} are kept, i.e., m histograms (one per class) are maintained for each input variable i. If X_i is a continuous variable, the bins of its histograms are defined as:

$$G_l^i = \begin{cases} [g_{l-1}^i, g_l^i), & \forall l \in \{1, ..., \eta - 1\} \\ [g_{l-1}^i, g_l^i] & l = \eta \end{cases} \tag{III.3a}$$

$$g_l^i = min_i + l \cdot s_i, \quad s_i = \frac{max_i - min_i}{\eta} \tag{III.3b}$$

where $[min_i, max_i]$ delimits the range of the ith variable and η is the number of bins (a user-defined parameter). If X_i is categorical, its histograms have one bin per category of the variable.

The data distribution information summarized by these histograms is used at several points of the learning interaction. Concretely, it affects the decisions of: which variables to include i the condition of a rule generated by covering; or to what extent we can expand (covering or mutation) or contract (mutation) a certain fuzzy set.

III.2.2 Exploration mode

Under exploration mode, each time a datum is received a learning (training) iteration is performed. At each learning iteration, a series of steps are taken in order to update the population based on the information contained in the received sample.

First, the matching degree between the input sample e and every rule in the population is checked and all those rules with a matching degree greater than zero are grouped to create the *matchset* [M]. Once [M] is built, those rules from [M] that advocates the true class of e are used by the system to create the *correctset* [C]. If none of the rules in [C] matches e with a sufficient degree for all the input variables, the covering operator is launched. Through the covering operator the system creates a new classifier that matches e with maximum degree and which is added to [C], [M] and [P]. Next, the fitness of those classifiers in [M] (which cover a region of the feature space that includes e) is updated. Afterwards, the rule discovery component may be applied. The rule discovery component allows the system to discover new promising rules via a genetic mechanism applied to the classifiers included in [C]. To control runtime and reduce the risk of overfitting, this component only acts if the average time since its last application upon the classifiers in [C] exceeds a certain threshold. Finally, histograms are updated and subsumption is checked in [P]. Checking subsumption is aimed at reducing the number of rules in [P], pushing toward maximally generalized rules and reducing redundancies between rules. The user-defined parameter θ_{sub} allows to control the frequency with which this subsumption check is performed. The different components involved in the exploratory behavior of the algorithm are further described below.

III.2.2.1 Matchset creation

Given an input sample e, those classifiers from [P] that match e with a degree greater than zero become part of the matchset [M]. To compute the matching degree between an input sample e and a classifier k, the algorithm calculates the membership degree $\mu_{A_i^k}(e_i)$ for each input variable x_i in the condition of the classifier. Then, the matching degree of the classifier is determined by the T-norm (conjunction) of such membership degrees. In our case the product ($\prod \mu_{A_i^k}(e_i)$) is used as T-norm. The system is enabled to deal with missing values by considering that if the value of the feature e_i is not known, then $\mu_{A^k}(e_i) = 1$.

From the classifiers forming [M], those which advocate for the true class of e become part of the correctset [C].

III.2.2.2 Covering operator

The covering operator generates a new classifier whose condition is generalized based on e and that advocates the class associated to e. This mechanism is only triggered when there is no classifier k in [C] which satisfies that: for any variable x_i in-

cluded in its condition, the membership degree of e_i to A_i^k is greater or equal than θ_μ ($\min([\mu_{A_1^k}(e_1), ..., \mu_{A_i^k}(e_i), ..., \mu_{A_n^k}(e_n)]) \geq \theta_\mu$).

To build the new classifier, the covering operator has to decide which variables are included in its condition and which not. Each variable has a probability of being selected equals to the relative frequency of the input value e_i among the seen samples of c^e according to the histogram. At least one variable must be included in the condition of the classifier. For each variable x_i selected to be included in the classifier, a fuzzy set has to be generalized from the corresponding input value e_i. This generalization process also relies on the histograms maintained by the algorithm for each class-variable pair. The underlying idea is to lead the generalization process towards the direction that allows to cover a higher ratio of c^e examples.

The relative frequency $h_{ic^e}(e_i)$ of the value e_i in the histogram maintained for class c^e and variable x_i and the associated probability $\widehat{h}_{ic^e}(e_i)$ of x_i being selected for the condition of the new classifier are calculated as:

$$\phi_i = l \text{ s.t. } e_i \in G_l^i \tag{III.4a}$$

$$\varphi_i = \begin{cases} l-1, & e_i \in G_l^i, l > 1, e_i < \frac{g_{l-1}^i + g_l^i}{2} \\ l+1, & e_i \in G_l^i, l < \eta, e_i > \frac{g_{l-1}^i + g_l^i}{2} \\ l, & \text{otherwise} \end{cases} \tag{III.4b}$$

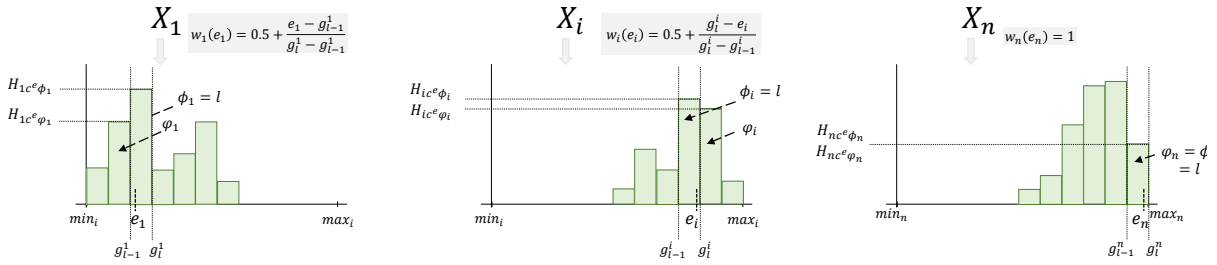
$$w_i(e_i) = \begin{cases} 0.5 + \frac{e_i - g_{l-1}^i}{g_l - g_{l-1}}, & \varphi_i = l-1 \\ 0.5 + \frac{g_l^i - e_i}{g_l - g_{l-1}}, & \varphi_i = l+1 \\ 1, & \text{otherwise} \end{cases} \tag{III.5a}$$

$$h_{ic^e}(e_i) = \frac{w_i(e_i) H_{ic^e\phi_i} + (1 - w_i(e_i)) H_{ic^e\varphi_i}}{\sum_{j=1}^m w_i(e_i) H_{ic_j\phi_i} + (1 - w_i(e_i)) H_{ic_j\varphi_i}} \tag{III.5b}$$

$$\widehat{h}_{ic^e}(e_i) = \frac{h_{ic^e}(e_i)}{\sum_{k=1}^n h_{kc^e}(e_k)} \tag{III.5c}$$

where ϕ_i indicates the bin where e_i lies and φ_i is the next closest bin. Note that, with the purpose of achieving better estimations, an interpolation (with weight w_i) is performed between bins ϕ_i and φ_i. Figure III.2 depicts three different sample scenarios for the computation of w_i. In the case of categorical variables, the number of bins matches the number of categories and no interpolation is applied, bin ϕ_i is the only one taken into account.

Algorithm 1, describes the process followed to generalized a fuzzy set from a single input sample. Starting from the core of bin ϕ_i where e_i lies and b_k is placed, in an iterative



(a) (b) (c)

Figure III.2: Illustration of the computation of $w_i(e_i)$ in different scenarios.

way, every bin core is being tested as a potential fuzzy set (left or right) vertex. That is, until a vertex is found that would provoke the relative frequency of c^e in the hypothetical fuzzy set to be lower than the one in ϕ_i. Hence, if placing the vertex in the bin core being tested means that the relative frequency of c^e in the fuzzy set would be lower that that in bin ϕ_i, the searching process stops and the previous bin core is established as expansion limit. Once both expansion limits (left and right) have been found, a_k and c_k are randomly assigned a value between b_k and the corresponding expansion limit. Figure III.3 illustrates the different steps of this generalization process. Note that the vertices of the fuzzy set may exceed the range of the input variable. This works as a way to enable the coverage of the minimum and maximum values of the variable, when appropriate. This process is much simpler when dealing with categorical variables. In such case, the three vertices of the fuzzy set are assigned the same value: e_i.

III.2.2.3 Fitness update

Each classifier internally maintains a vector of class weights $\{\omega_1^k, ..., \omega_m^k\}$. Such weights are incrementally updated during the learning process. This class weight vector allows to determine which class the classifier advocates, as well as, to compute the fitness of the classifier. At each exploration iteration, the class weights of those classifiers in [M] are updated. The class c^k with maximum weight ω_j^k is the one advocated by rule k. Given that these class weights are updated along the learning process, the class advocated by a classifier is not set when the classifier is born but it may change as the rule matches different examples and its parameters are consequently updated. The process followed to update the class weights of each classifier in [M] is now detailed.

First, the sum of correct matchings for each class is calculated:

$$cm_{j_{t+1}}^k = cm_{j_t}^k + m(k,j) \cdot \lambda_{c^e} \tag{III.6}$$

where λ_{c^e} can be specified as a user-defined parameter or estimated based on the class

Algorithm 1: Description of the generalization process followed by the covering operator to define the fuzzy set $\widehat{A_i^k}$ for the ith input variable based on the value of the received training sample. The fuzzy set $\widehat{A_i^k}$ is defined by three continuous values: a_k (left vertex), b_k (middle vertex) and c_k (right vertex).

Function `hInFuzzySet`(v_1, v_2, v_3, c^e)**:**

$h \leftarrow h_{ic^e}(v_2)$ `/* Frequency of` c^e `for value` v_2 `according to the histograms */`

for $x \leftarrow v_1; x < v_2; x = x + s_i$ **do**

$line_1 \leftarrow$ `line(point(`$x, 0$`), point(`$x, 1$`))`

$line_2 \leftarrow$ `line(point(`$v_1, 0$`), point(`$v_2, 1$`))`

$h \leftarrow h + h_{ic^e}(x)\cdot$`IntersectionHeight(`$line_1$, $line_2$`)`

end

for $x \leftarrow v_2 + s_i; x <= v_3; x = x + s_i$ **do**

$line_1 \leftarrow$ `line(point(`$x, 0$`), point(`$x, 1$`))`

$line_2 \leftarrow$ `line(point(`$v_2, 1$`), point(`$v_3, 0$`))`

$h \leftarrow h + h_{ic^e}(x)\cdot$`IntersectionHeight(`$line_1$, $line_2$`)`

end

`return(`h`)` `/* Return` c^e `frequency in the fuzzy set defined by` $\{v_1, v_2, v_3\}$ `*/`

Function `FuzzySetCoveringInit` *(i, ϕ_i, c^e, start, end)*:

`/*` i `is the index of the input variable,` ϕ_i `the bin where the input sample lies and` c^e `the class of the input sample */`

$a_k, b_k, c_k \leftarrow g^i_{\phi_i - 1} + \frac{s_i}{2}$

$limit \leftarrow b_k$

for $x_1 \leftarrow b_k + s_i; x_1 \leq end; x_1 = x_1 + s_i$ **do** `/* Look for right expansion limit */`

$aux \leftarrow$`hInFuzzySet(`a_k, b_k, x_1, c^e`)`

if $aux \geq$ `min(`$h_{c^e}, h_{ic^e}(b_k)$`)` **then**

$limit \leftarrow x_1$

else

`break`

end

end

$c_k \leftarrow$`rand(`$[b_k, limit]$`)` `/* Right vertex definition */`

$limit \leftarrow b_k$

for $x_1 \leftarrow b_k - s_i; x_1 \geq start; x_1 = x_1 - s_i$ **do** `/* Look for left expansion limit */`

$aux \leftarrow$ `hInFuzzySet(`x_1, b_k, c_k, c^e`)`

if $aux \geq$ `min(`$h_{c^e}, h_{ic^e}(b_k)$`)` **then**

$limit \leftarrow x_1$

else

`break`

end

end

$a_k \leftarrow$`rand(`$[limit, b_k]$`)` `/* Left vertex definition */`

`return(`a_k, b_k, c_k`)` `/* Return the three vertices defining the fuzzy set */`

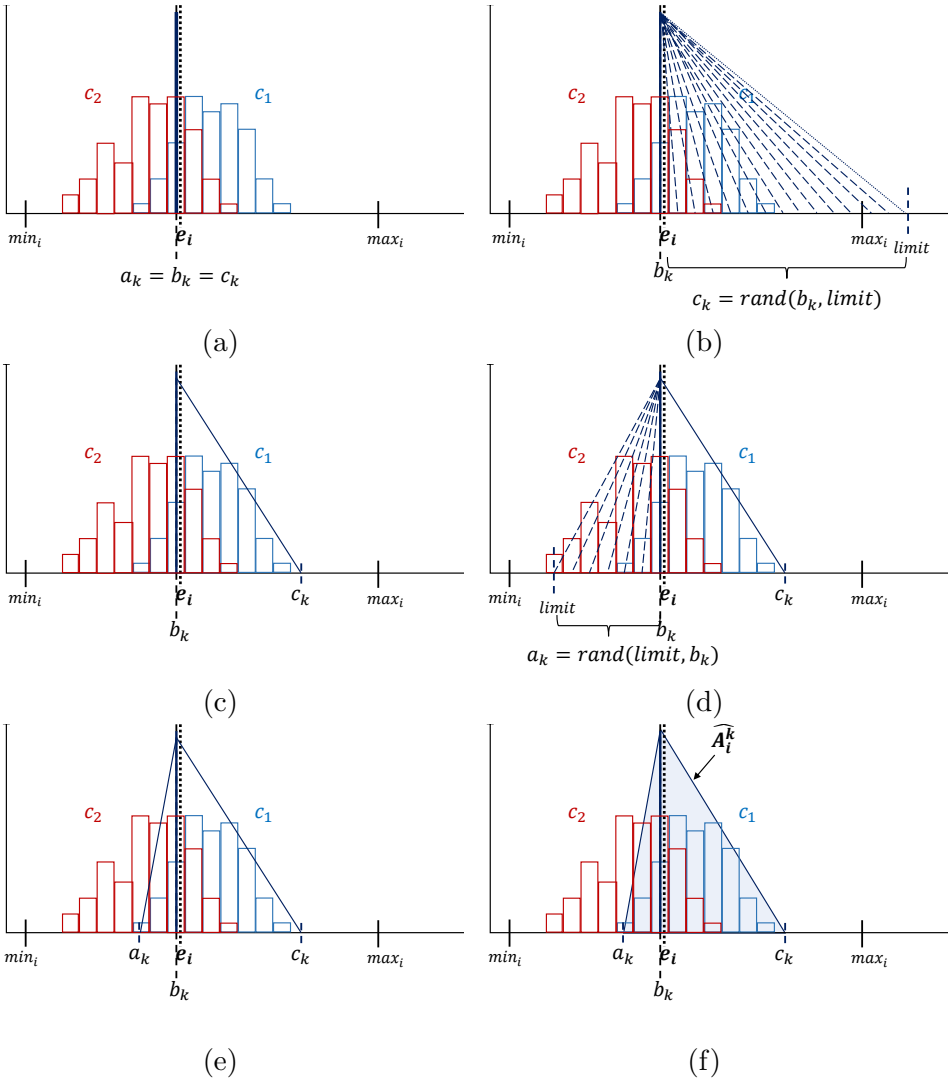


Figure III.3: Schematic example of the process followed in covering to define a fuzzy set based on the input value e_i.

balance among the processed samples, and

$$m(k,j) = \begin{cases} \mu_{A^k}(e) & \text{if } j = c^e \\ 0 & \text{otherwise} \end{cases} \tag{III.7}$$

λ_{c^e} adjusts the weight e has on the fitness based on its class and the class balance observed in the samples received so far. Next, each weight $\omega^k_{j_{t+1}}$ is computed based on $cm^k_{j_{t+1}}$:

$$\forall j : \omega^k_{j_{t+1}} = \frac{cm^k_{j_{t+1}}}{\sum_i cm^k_{i_{t+1}}} \tag{III.8}$$

Thus, if a rule k has only matched examples from class j, ω_j^k will be 1 and the remaining weights 0. Those rules which match examples from different classes will have weights ranging from 0 to 1, the sum of all the weights always remaining 1.

Lastly, the class weights previously updated are used to compute the fitness of the rule F_{t+1}^k as follows:

$$F_{t+1}^k = \omega_{max_{t+1}}^k - \sum_{j|j\neq max} \omega_{j_{t+1}}^k \tag{III.9}$$

where we subtract the values of the other weights from the weight with maximum value ω_{max}^k. It is worth noting that: (1) this way of computing fitness is aimed at favoring classifiers which match samples of one single class, and (2) zero or negative fitness is possible (e.g., if there are more than two classes and all the class weights are the same).

Once fitness has been updated, the experience of the rule exp^k is increased by one and the correct set size of the classifiers in [C] is updated. This correct set size is computed as the arithmetic average of the sizes of all the correct sets in which the classifier has participated.

III.2.2.4 Rule discovery component

This component is aimed at discovering new promising rules. For this purpose, a steady-state niche-based *genetic algorithm* (GA) is used. The GA is applied to the classifiers included in [C]. The niching is, therefore, provided by the GA being applied to rules that match the same input with a degree greater than zero and advocate the same class.

The GA is triggered only when the average time since its last application on the classifiers in [C] surpasses the threshold θ_{GA} (user-defined parameter). This discovery component selects two parents p_1 and p_2 from [C] to generate offspring ch_1 and ch_2 by means of crossover and mutation. Note that the non-generational but steady-state character of this GA allows it to function in an online way.

Proportionate selection (Godberg, 1989) is employed for parent selection, and the probability for a classifier k of being selected is:

$$p_{sel}^k = \begin{cases} \dfrac{(F^k)^v \cdot \mu_{A^k}(e)}{\sum\limits_{i\in[C]|F^i>0} (F^i)^v \cdot \mu_{A^i}(e)} & \text{if } F^k > 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{III.10}$$

where $v > 0$ is a constant that regulates the pressure towards maximally accurate rules. The crossover operator works in a parent-centric way (García-Martínez et al., 2008; Lozano et al., 2004) creating one child in the neighborhood of each of the parents and defining the range of the neighborhood based on the distance between both parents. The condition of the child includes the same variables as the condition of the main parent. Lets $\{a_{p_1}, b_{p_1}, c_{p_1}\}$

be the left, middle and right vertices of the fuzzy set of the ith variable in the condition of p_1; $\{a_{p_2}, b_{p_2}, c_{p_2}\}$ the vertices of that same variable in p_2, and ch_1 the offspring centered on p_1. The parent-centric crossover operator generates a fuzzy set for ch_1 defined by the following a_{ch_1}, b_{ch_1} and c_{ch_1} vertices:

$$I_v = v_{p_2} - v_{p_1}; \quad v \in \{a, b, c\} \tag{III.11a}$$

$$b_{ch_1} = \text{rand}([\min(b_{p_1}, b_{p_1} + I_b \cdot \alpha), \max(b_{p_1}, b_{p_1} + I_b \cdot \alpha)]) \tag{III.11b}$$

$$a_{ch_1} = \text{rand}([\min(b_{ch_1}, a_{p_1}, a_{p_1} + I_a \cdot \alpha), \min(b_{ch}, \max(a_{p_1}, a_{p_1} + I_a \cdot \alpha))]) \tag{III.11c}$$

$$c_{ch_1} = \text{rand}([\max(\min(c_{p_1}, c_{p_1} + I_c \cdot \alpha), b_{ch_1}), \max(c_{p_1}, c_{p_1} + I_c \cdot \alpha, b_{ch_1})]) \tag{III.11d}$$

where $\alpha \in [0.5, 1]$ determines the spread associated with the probability distributions used to create offspring. If a variable is included in p_1 but not in p_2, the fuzzy set from p_1 is just copied into ch_1. For ch_2, p_1 and p_2 roles are interchanged.

Then, the offspring created through the crossover operator may be mutated. Each variable in the condition of the new classifier is impacted by mutation with probability μ. If selected, the variable is affected in one of the two following ways: expansion or contraction. In expansion, new vertices a'_{ch} and c'_{ch} are generated such that $a'_{ch} <= a_{ch}$ and $c'_{ch} >= c_{ch}$, where a_{ch} and c_{ch} are the vertices before mutation. In the case of contraction, instead of looking for increasing the area covered by the fuzzy set we are looking for decreasing it. Therefore, new vertices a'_{ch} and c'_{ch} are generated such that $a'_{ch} >= a_{ch}$ and $c'_{ch} <= c_{ch}$. In both cases, expansion and contraction, the logic of the process followed to generate new external vertices is analogous to the one of the fuzzy set generalization process used in covering. In addition, the mutation operator can also add or remove variables from the condition of the classifier, provided that the condition is not left empty. The probability of adding or removing one variable is based on the class ratios of the variables forming the condition of the classifier. Such class ratios are estimated based on the histograms maintained by the algorithm.

Before being inserted in the population, the new offspring is compare with both its parents. If it is not identical to any of them, we look for the most general classifier from [C] that can subsume the offspring. If no subsumer is found, the new classifier is inserted in the population. Subusmption mechanism (Section III.2.2.7) prevents the creation of classifiers with specific conditions when there already are more general and, at least, equally accurate rules in the population which cover the same region of the feature space.

III.2.2.5 Replacement mechanism

If the maximum size of the population has been reached, classifiers need to be removed from [P]. Each classifier in [P] has a deletion probability proportional to its numerosity num and its average correct set size cs. Furthermore, if the power of its fitness $(F^k)^v$, taking into account on how many samples it is based (exp^k), is lower than the population

average fitness power ($(F^k + \varepsilon^k)^v < F_{[P]}$ where $F_{[P]} = (1/N)\sum_{j\in[P]}(F^j)^v$), the deletion probability of the classifier is further increased. Thereby, the deletion probability p^k_{del} of each classifier k is computed as follows:

$$\varepsilon^k = \sqrt{\frac{R^2 \cdot \log(1/\delta)}{2 \cdot exp^k}} \tag{III.12a}$$

$$d_k = \begin{cases} \frac{cs \cdot num \cdot F_{[P]}}{(F^k)^\nu}, & \text{if } (F^k + \varepsilon^k)^\nu < F_{[P]} \\ cs \cdot num, & otherwise \end{cases} \tag{III.12b}$$

$$p^k_{del} = \frac{d_k}{\sum\limits_{\forall j\in[P]} d_j} \tag{III.12c}$$

Note that a Hoeffding-kind bound (Hulten et al., 2001; Hoeffding, 1994) is used to decide if the deletion probability of the classifier should be further increased or not. In the computation of ε^k, R is the range of the fitness, and δ and ν are user-defined parameters. δ establishes the confidence level $(1-\delta)$ with which $F^k \in [F^k - \varepsilon^k, F^k + \varepsilon^k]$. The inclusion of ε allows to take into account the experience of the rule and, therefore, the soundness of its fitness. Hence, it is possible to avoid penalizing certain apparently poorly fit rules based on unreliable fitness values. The threshold is more demanding when the fitness of the classifier is consider sounder. Moreover, rigid user-defined threshold are avoided.

This kind of deletion probability pushes toward the removal of classifiers belonging to large correct sets at the same time that it encourages the search of highly fit classifiers by increasing the deletion probability of those rules whose fitness is significantly lower than the population average. Therefore, it pursues to avoid the search for fit classifiers being done at the expense of significant diversity reduction, which would mean ceasing to cover certain areas of the feature space.

III.2.2.6 Histogram update

At the end of each learning iteration, the algorithm updates the histograms of class c^e to which e belongs. The histogram of c^e for each input variable is updated as follows:

$$H^{t+1}_{ic^e\phi_i} = H^t_{ic^e\phi_i} + 1,\ \forall i \in \{1, ..., n\} \tag{III.13}$$

where ϕ_i is the bin where e_i lies.

III.2.2.7 Subsumption Mechanism

As mentioned before, this mechanism allows to reduce redundancies between the rules included in the population and acts as a mean to push [P] towards generalization (Wilson,

1998). For a rule r_i to be considered as a candidate subsumer of another rule r_j, it must meet the following two requirements: (1) r_i must have a similar or better fitness than r_j (that is, $F^i - \varepsilon^i \geq 0.9 \cdot (F^j + \varepsilon^j)$), and (2) r_i needs to be more general than r_j. A rule r_i is more general than r_j if all the variables in the condition of r_i are also included in r_j, and the area covered by the fuzzy set in r_j is included in the area of the fuzzy set in r_i for each one of its variables. Every time a rule r_i subsumes a rule r_j, the numerosity of the subsumer r_i increases while r_j is removed from the population.

This mechanism is employed under exploration mode at two different points: (1) in GA before inserting the offspring into [P], and (2) at the end of learning iterations when each rule in [P] is checked for subsumption with each other rule in [P].

III.2.3 Exploitation mode

Under exploitation mode, the rules in [P] work together to predict the class of a test sample. The population is searched for those classifiers that meet the following two conditions: (1) the membership degree of the test sample for each of the variables in the condition of the classifier is greater than θ_μ, and (2) the fitness of the classifier can be said to be greater than zero with a high level of confidence ($F^k - \varepsilon^k > 0$). Each classifier k in [P] that meets these two conditions emits a weighted vote v_c^k for the class c it advocates:

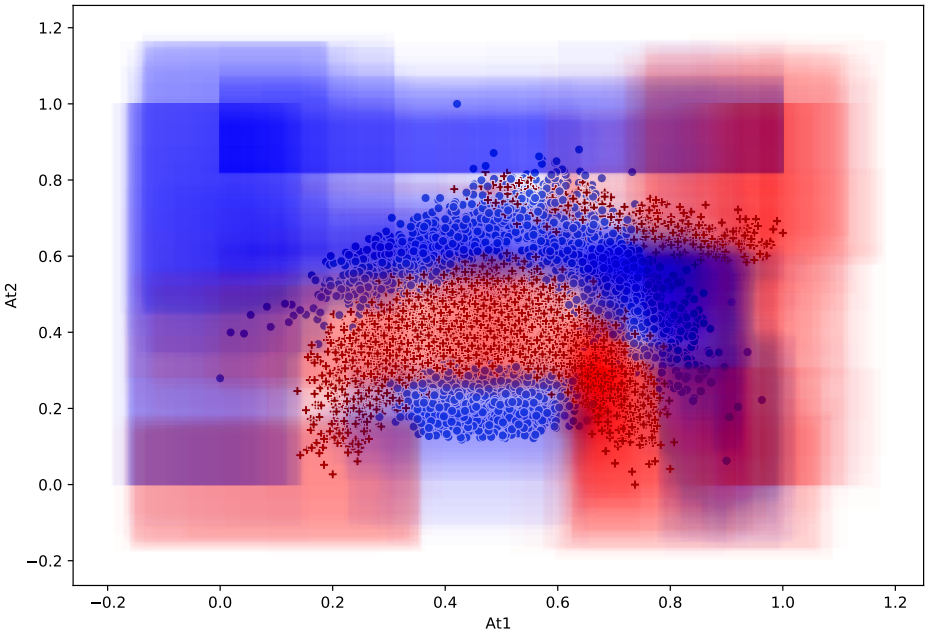
$$v_c^k = (F^k - \varepsilon^k) \cdot \mu_{A^k}(e) \tag{III.14}$$

The votes of all the classifiers are counted and the most voted class is returned as the predicted one. In case of a tie between classes the fittest rule decides.

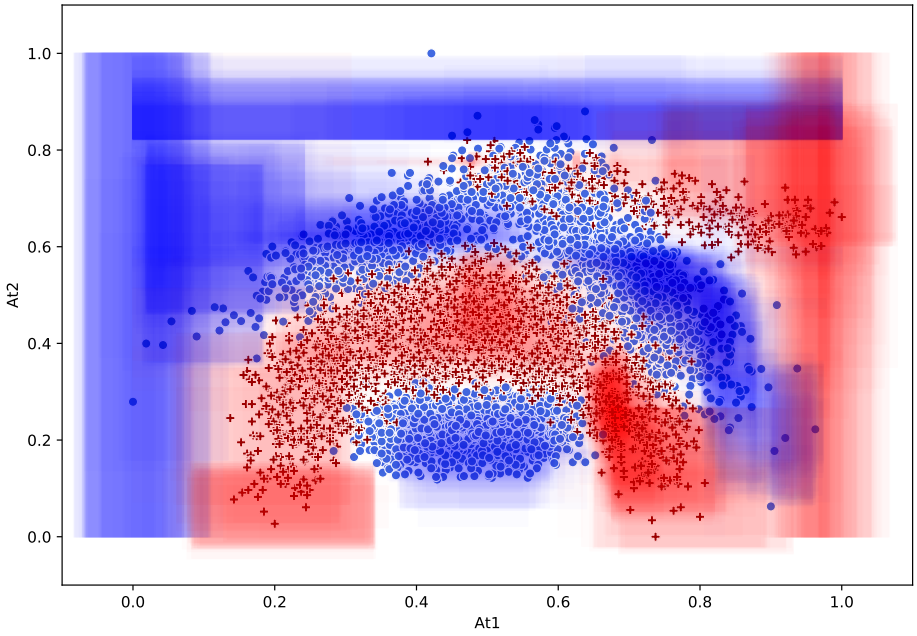
Figure III.4 represents which areas of the feature space for the *banana* dataset (KEEL Repository (Alcalá-Fdez et al., 2011)) are covered by the rule population that CLAST generates for this dataset. The area covered by each rule is colored based on which class the rule advocates. In Figure III.4 (a) the whole area covered by the rule is colored, that is, the membership degree of any sample in the colored area will be greater than zero for both input variables. In Figure III.4 (b), the colored area match that in which the membership degree for both variables is greater than 0.5. Therefore, each sample of the *banana* dataset is represented covered by those rules that, in exploitation mode, would vote to predict its class if (a) $\theta_\mu = 0$ or (b) $\theta_\mu = 0.5$. The color balance gives us an idea of which class would be the most voted one at each point.

III.3 Comparison of CLAST to several machine learning techniques

In this section, we analyze the performance of CLAST in comparison with different machine learning techniques. For this purpose, we compare CLAST with two sets of classifiers:



(a)



(b)

Figure III.4: Rule covering area for *banana* dataset (KEEL Repository (Alcalá-Fdez et al., 2011)): (a) full covering area of the rules, and (b) central half of the covering area of the rules ($\mu_{A^k}(e_i) \geq 0.5$). The color of the covering area of a rule is set according to which class it advocates.

data stream classifiers and traditional batch classifiers. With the former comparison, we analyze the behavior of CLAST with respect to other data stream approaches that also function in an incremental way, which may limit the maximum performance that can be attained in some domains. With the latter comparison, we analyze whether, even with the inherent limitations that an incremental learning design may impose, CLAST is competitive with several of the most representative batch learners. Finally, we analyze the behavior of CLAST and other data stream classifiers on four different real-world data streams.

Below, we first describe the experimental methodology for each case, and then present and analyze the results of the different algorithms.

III.3.1 Comparison with other data stream approaches

Throughout this section we compare the performance of CLAST with other data stream classifiers based on different types of models.

III.3.1.1 Experimental setup

The behavior of CLAST is compared with the following seven data stream learners: CVFDT, CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, HAT, VFDR, AWEC and DWM. CVFDT (Hulten et al., 2001) is an extension of the well known Very Fast Decision Tree (VFDT) (Domingos and Hulten, 2000) suitable for data streams in which there is concept changing. CVFDT_{NB} incorporates to CVFDT the Naïve Bayes classification strategy at leaves as proposed by Gama for VFDT (Gama et al., 2003). $\text{CVFDT}_{\text{NBA}}$ (Bifet et al., 2010a) is an extended version of CVFDT which monitors the error rate of majority class and Naïve Bayes decisions at every leaf and decides to employ the option that has been more accurate in past cases. HAT (Bifet and Gavaldà, 2009a) is based on an adaptive Hoeffding Tree that uses ADWIN to monitor performance of branches on the tree and replace them with new branches when their accuracy decreases in case the new branches are more accurate. VFDR (Kosina and Gama, 2015) is an incremental rule learning classifier that follows a similar learning process to VFDT. AWEC (Wang et al., 2003a) is an ensemble approach where each classifier is weighted based on its expected classification accuracy. DWM (Kolter and Maloof, 2007) is also an ensemble approaches; it uses four different mechanisms to cope with concept drift.

The performance of the different approaches is compared based on their results in 22 different datasets. A summary of the main characteristics of each of these datasets is shown in Table III.1.

To evaluate and compare the performance of the classifiers, a test-then-train scheme is followed. This evaluation scheme is designed specifically for streaming settings. Using test-then-train evaluation each example has a double function: first it is used for testing and then for training the classifier. The examples are processed sequentially, in the same

Table III.1: Properties of the datasets. The columns of the table describe: the name of the dataset (Dataset), the number of instances (#Inst), the number of attributes (#Att), the number of real attributes (#Re), the number of integer attributes (#In), the number of nominal attributes (#No), the number of classes (#Cl), the proportion of instances of the minority class (%Min), and the proportion of instances of the majority class (%Maj).

Alias	Dataset	#Inst	#Att	#Re	#In	#No	#Cl	%Min	%Maj
app	appendicitis	106	7	7	0	0	2	19.80	80.20
aut	automobile	156	25	7	8	10	5	8.30	30.80
ban	banana	5300	2	2	0	0	2	44.80	55.20
bnd	bands	365	19	11	8	0	2	37.00	63.00
bre	breast	277	9	0	1	8	2	29.20	70.80
car	car	1728	6	0	0	6	4	3.80	70.00
cov	covtypeNorm	581012	54	10	44	0	7	0.50	48.80
eco	ecoli	332	6	0	6	0	6	1.51	43.10
hay	hayes-roth	160	4	0	4	0	3	19.40	40.60
hea	heart	270	13	0	13	0	2	44.40	55.60
kdd	kddcup	494043	41	15	23	3	23	1.6e-3	56.80
mam	mammographic	961	4	0	4	0	2	46.30	53.70
pag	page-blocks	5472	10	4	6	0	5	0.50	89.80
pim	pima	768	8	2	6	0	2	34.90	65.10
sah	saheart	462	9	5	3	1	2	34.60	65.40
skn	skin	245057	3	0	3	0	2	20.80	79.30
tae	tae	151	5	0	5	0	3	32.50	34.40
tic	tic-tac-toe	958	9	0	0	9	2	34.70	65.30
tit	titanic	891	9	1	3	5	2	38.38	61.62
veh	vehicle	846	18	0	18	0	4	23.50	25.80
yea	yeast	1484	8	8	0	0	10	0.30	31.20
wdb	wdbc	569	30	30	0	0	2	37.26	62.74

order they are received. In this way, the model is tested in all the available examples and testing is always done on examples that the algorithm has not yet seen, from which it has not been able to learn. For each dataset, the classifiers are tested on a random sample of 100,000 examples. Thus, the average number of times the algorithms will see each example will depend on the original size of the dataset.

The performance is assessed based on two measures: (1) the test accuracy rate, i.e., the ratio of correct predictions in previously unseen instances; and (2) the G-mean. G-mean is included as evaluation measure due to the class imbalance ratios present in some of the datasets. In a binary problem, G-mean is calculated as the geometric mean of the True Positive Rate (TPR) and the True Negative Rate (TNR):

$$G\text{-}mean = \sqrt{\frac{TP}{TP+FN} \cdot \frac{TN}{TN+FP}} = \sqrt{TPR \cdot TNR} \tag{III.15}$$

where TP or True Positives is the number of correct positive predictions; FN or False Negatives is the number of positive examples wrongly classified as negative; TN or True Negatives is the number of correct negative predictions, and FP is the number of negative examples wrongly classified as positive. In an imbalance problem where the negative class is the minority one, classifiers are keen on increasing TP at the expense of also increasing FP. G-mean tries to maximize the success in both classes with a good balance between them. Moreover, G-mean can be easily defined for a multiclass problem as:

$$G\text{-}mean = \Big(\frac{\text{true } c_1}{|c_1|} \cdot \frac{\text{true } c_2}{|c_2|} \cdot \ldots \cdot \frac{\text{true } c_m}{|c_m|}\Big)^{1/m} \tag{III.16}$$

where $truec_i$ is the number of i^{th} class examples correctly classified; $|c_i|$ the total number of examples in class c_i, and m is the number of classes.

All the classifiers, except for CLAST, were run using Scikit-multiflow (Montiel et al., 2018), a python library that implements several data stream classifiers (many of them based on MOA (Bifet et al., 2010a)). We followed the recommended parameter values given in Scikit-multiflow for each classifier. Likewise, we also maintain the same CLAST parameters for all the experiments: $N = 5000$ (population maximum size), $\eta = 20$, $\theta_{sub} = 500$, $\theta_{GA} = 50$, $\delta = 0.05$ and $\nu = 10$. The results shown below are averages over thirty runs with different seeds

The results were statistically analyzed according to the considerations and recommendations pointed out in Derrac et al. (2011). We used non parametric statistical tests to compare the results achieved by the different learning approaches. Parametric tests are based on assumptions which must be satisfied for the tests to be effective. Such assumptions are usually violated when analyzing the performance of stochastic algorithms (Sheskin, 2020; García et al., 2009). Therefore, nonparametric statistical tests are recommended (Derrac et al., 2011; Demšar, 2006) since they relax the requirements on the input data and provide a practical tool to be used when the assumptions required by the parametric tests cannot be satisfied.

We apply multiple comparison statistical procedures to test the null hypothesis which states that all the algorithms perform equally on average. In particular, the well-known Friedman's test (Friedman, 1937, 1940) is used. The Friedman's test is the equivalent of the repeated measures ANOVA in nonparametric statistical procedures; thus, it is a multiple comparisons test that seeks to detect significant differences between the behavior of two or more algorithms. If Friedman's null hypothesis is rejected, we employ the nonparametric post-hoc Finner procedure (Finner, 1993) to compare CLAST (control-method) to every other learner. From the ranking obtained through Friedman's test, the p-values of the required family of hypotheses for such comparisons can be computed. However, these p-values are not valid for multiple comparisons (Derrac et al., 2011), because they do not take into account the remaining comparisons belonging to the family. Adjusted p-values can deal with this problem. The Finner procedure adjusts the value of α in a step-down manner. It rejects hypotheses H_1 to H_{i-1} if i is the smallest integer so that $p_i > 1 - (1 - \alpha)^{(k-1/i)}$ where p_i is the unadjusted p-value of H_i, and the adjusted p-value of each hypothesis H_i is computed as $APV_i = minv, 1$, where $v = max1 - (1 - p_j)^{(k-1)/j} : 1 \leq j \leq i$. Finner test was considered a good option because, being easy to comprehend, it usually offers better results than other post-hoc tests like Bonferroni-Dunn test (excessively conservative) or, even, Holm, Hochberg or Rom (Derrac et al., 2011).

Pairwise comparison tests, such as Wilcoxon signed ranks test, are not recommended to extract conclusions involving more than one pairwise comparison because an accumulated error coming from its combination will be obtained. However, in a multiple comparison, the set of algorithms included can determine the results of the analysis, whereas a pairwise comparison is not influenced by any external factor. Hence, we also compare the performance of CLAST with each other learner by means of the nonparametric Wilcoxon singed-ranks test (Wilcoxon, 1992) to complement the statistical analysis.

III.3.1.2 Results

Below, we compare CLAST to the mentioned set of data stream classifiers. Tables III.2 and III.3 show the performance of the classifiers on the collection of datasets in terms of accuracy and G-mean, respectively. Both tables also include the ranks of the learners according to their performance in each of the problems.

Moreover, the results shown in these two first tables are summarized in Figure III.5 and Table III.4. Figure III.5 represents the distribution of the values of accuracy and G-mean for each learner, while Table III.4 shows the average per algorithm. The range of the accuracy or G-mean values obtained by the classifiers varies considerably depending on the dataset. To avoid that the particularities of the different problems can affect the weight they have on the global comparison, we normalized the accuracy and G-mean values for each problem. Both Figure III.5 and Table III.4 are based on such normalized values.

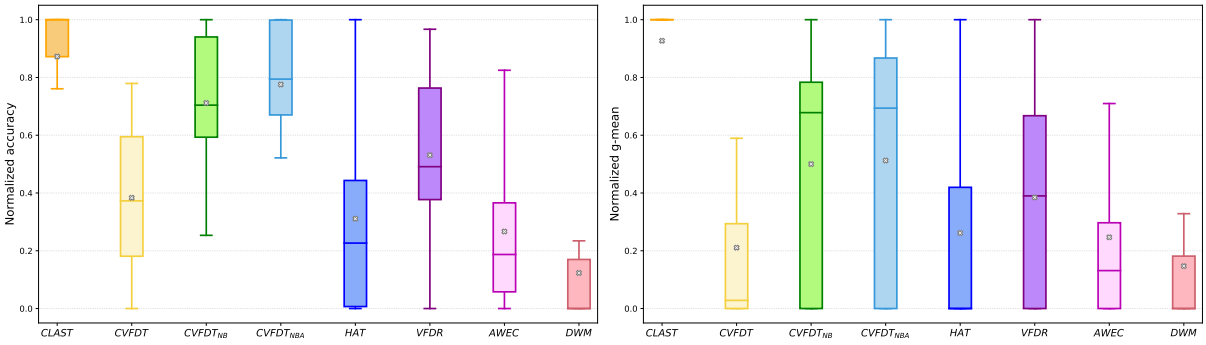
Some remarks can be commented based on the experimental results. First, CLAST is the classifier that exhibits the highest level of performance. It achieves both the best

Table III.2: Test accuracy and ranking positions of the different online approaches for each dataset (Data). The results collected in the table are referred to the final time stamp when all the examples in the stream have been used for both test and train.

Data	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
app	93.71 (3)	91.97 (5)	95.73 (2)	95.78 (1)	88.13 (7)	93.51 (4)	88.55 (6)	86.32 (8)
aut	98.11 (1)	60.33 (7)	83.84 (2)	83.83 (3)	73.64 (4)	72.01 (5)	71.49 (6)	53.70 (8)
ban	85.01 (1)	44.83 (2)	44.68 (7)	44.69 (6)	44.64 (8)	44.72 (5)	44.74 (4)	44.83 (3)
bnd	93.33 (1)	72.35 (5)	82.47 (2)	82.39 (3)	67.93 (8)	73.87 (4)	70.17 (6)	68.12 (7)
bre	95.04 (1)	82.30 (5)	90.30 (3)	90.39 (2)	78.77 (6)	82.95 (4)	75.57 (7)	74.66 (8)
car	96.59 (1)	81.43 (5)	92.37 (2)	92.33 (3)	79.44 (6)	90.67 (4)	76.40 (7)	71.47 (8)
cov	68.09 (1)	59.57 (6)	64.17 (4)	65.54 (3)	63.37 (5)	65.92 (2)	57.50 (7)	54.17 (8)
eco	94.81 (1)	71.09 (8)	86.90 (3)	87.35 (2)	81.21 (6)	86.33 (4)	84.19 (5)	75.87 (7)
hay	86.71 (4)	83.48 (5)	88.22 (1)	88.21 (2)	77.66 (6)	87.18 (3)	74.11 (8)	77.34 (7)
hea	91.53 (3)	84.00 (7)	92.80 (1)	92.80 (1)	83.40 (8)	87.89 (4)	84.27 (6)	85.60 (5)
kdd	98.16 (1)	86.51 (7)	96.70 (4)	97.87 (2)	54.24 (8)	96.70 (3)	96.66 (5)	96.56 (6)
mam	80.25 (2)	79.26 (4)	79.29 (3)	81.62 (1)	75.91 (8)	78.92 (6)	79.13 (5)	78.67 (7)
pag	93.79 (2)	91.88 (5)	86.76 (7)	94.02 (1)	91.95 (4)	84.30 (8)	92.27 (3)	88.59 (6)
pim	86.09 (1)	76.98 (6)	81.60 (3)	81.96 (2)	75.02 (8)	77.88 (4)	77.04 (5)	75.76 (7)
sah	90.72 (1)	75.08 (5)	81.18 (3)	81.47 (2)	71.91 (7)	76.07 (4)	72.42 (6)	71.39 (8)
skn	93.23 (6)	95.87 (4)	97.82 (2)	97.86 (1)	93.80 (5)	97.29 (3)	91.97 (8)	92.41 (7)
tae	88.82 (1)	62.41 (5)	72.98 (3)	73.05 (2)	54.70 (6)	65.64 (4)	52.47 (7)	52.19 (8)
tic	96.31 (1)	79.93 (5)	88.23 (3)	88.21 (4)	73.87 (6)	90.90 (2)	72.10 (7)	70.27 (8)
tit	88.01 (6)	94.14 (4)	98.92 (3)	98.92 (2)	98.99 (1)	88.21 (5)	79.48 (7)	77.96 (8)
veh	87.60 (1)	59.49 (6)	72.98 (2)	72.93 (3)	60.27 (5)	65.31 (4)	55.04 (7)	43.49 (8)
wdb	94.98 (6)	95.15 (5)	98.08 (3)	98.35 (2)	98.77 (1)	96.17 (4)	94.80 (7)	93.86 (8)
yea	63.91 (1)	46.18 (7)	55.37 (3)	55.35 (4)	52.07 (6)	54.40 (5)	58.32 (2)	31.98 (8)

Table III.3: G-mean and ranking positions of the different online approaches for each dataset (Data). The results collected in the table are referred to the final time stamp when all the examples in the stream have been used for both test and train.

Data	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
app	93.27 (2)	84.59 (5)	93.37 (1)	92.83 (3)	78.54 (8)	89.09 (4)	81.61 (6)	79.56 (7)
aut	98.24 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	69.72 (2)	0.00 (3)
ban	84.84 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
bnd	92.80 (1)	59.71 (6)	79.51 (2)	78.87 (3)	46.38 (8)	64.80 (4)	60.27 (5)	47.10 (7)
bre	93.00 (1)	72.21 (5)	86.74 (2)	86.23 (3)	69.28 (6)	76.19 (4)	66.64 (8)	66.73 (7)
car	90.93 (1)	27.14 (8)	75.48 (2)	75.13 (3)	54.35 (5)	73.07 (4)	31.19 (7)	34.23 (6)
cov	26.63 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
eco	93.98 (1)	1.33 (8)	84.23 (3)	84.18 (4)	76.50 (6)	85.26 (2)	83.66 (5)	69.85 (7)
hay	86.46 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
hea	91.40 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
kdd	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
mam	80.38 (2)	79.35 (4)	79.36 (3)	81.77 (1)	75.87 (8)	79.04 (6)	79.32 (5)	78.87 (7)
pag	60.26 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
pim	84.43 (1)	71.80 (6)	78.55 (2)	78.19 (3)	66.83 (8)	73.57 (4)	71.94 (5)	71.25 (7)
sah	89.81 (1)	64.96 (7)	77.08 (2)	76.13 (3)	63.48 (8)	70.92 (4)	68.72 (6)	68.87 (5)
skn	95.10 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
tae	88.65 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.0 (2)
tic	95.51 (1)	71.62 (5)	85.72 (3)	85.54 (4)	64.90 (6)	89.76 (2)	50.77 (7)	48.83 (8)
tit	86.91 (6)	94.04 (4)	98.87 (3)	98.90 (2)	98.96 (1)	87.79 (5)	78.47 (7)	75.89 (8)
veh	87.28 (1)	53.51 (6)	72.41 (2)	72.31 (3)	58.45 (5)	64.43 (4)	52.36 (7)	39.32 (8)
wdb	93.91 (6)	94.29 (5)	98.11 (3)	98.17 (2)	98.66 (1)	95.91 (4)	93.86 (7)	93.0 (8)
yea	58.49 (1)	0.0 (8)	42.0 (3)	41.47 (4)	39.27 (5)	42.85 (2)	35.69 (6)	19.21 (7)



(a) (b)

Figure III.5: Distribution of the results achieved by each classifier in terms of: (a) normalized accuracy, and (b) normalized G-mean.

Table III.4: Comparison of the average performance of CLAST with the performance of the online learners.

		$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	Avg	0.87 (1)	0.38 (5)	0.71 (3)	0.78 (2)	0.31 (6)	0.53 (4)	0.27 (7)	0.12 (8)
	Rank	2.09 (1)	5.36 (5)	3.00 (3)	2.36 (2)	5.86 (6)	4.14 (4)	5.95 (7)	7.18 (8)
G-mean	Avg	0.93 (1)	0.21 (7)	0.5 (3)	0.51 (2)	0.26 (6)	0.38 (4)	0.25 (5)	0.15 (8)
	Rank	1.55 (1)	4.32 (6)	2.23 (2)	2.55 (3)	4.23 (5)	3.05 (4)	4.45 (7)	5.00 (8)

accuracy and G-mean in most of the problems (accuracy: 14/22, G-mean: 17/22). Whether we look at the distribution shown in the boxplots or just at the average, the advantage of CLAST over the next best performing classifiers ($\text{CVFDT}_{\text{NBA}}$ and CVFDT_{NB}) is clear.

Figure III.5 shows how the variability of the results is significantly greater in terms of G-mean than in terms of accuracy for several algorithms, which also see their average performance diminished. Some of the datasets employed present a certain degree of class imbalanced (see Table III.1). Classifiers that obtain good results in terms of accuracy, such as $\text{CVFDT}_{\text{NBA}}$, CVFDT_{NB} or VFDR, fail to classify the instances of the minority class of these problems correctly. This penalizes their G-mean. Nonetheless, CLAST improves its relative results when using G-mean as evaluation metric. This points to the fact that the classifier is capable of learning all classes, even minority ones, and therefore maintains a good balance in the accuracy with which it classifies the examples of all of them.

We conducted a statistical test analysis on the results. First, we applied Friedman's test which rejected ($\alpha = 0.05$) the median performance of all the algorithms being equivalent. Having Friedman's test rejected its null hypothesis, we applied the post-hoc Finner test ($\alpha = 0.05$) to compare the performance of CLAST with that of each other classifier. According to Finner test, CLAST improves the performance of the other seven data stream classifiers but such improvement is not statistically significant for CVFDT_{NB} or, in terms of accuracy, for $\text{CVFDT}_{\text{NBA}}$. Finally, to complement our analysis, we also carried out pairwise comparisons between CLAST and every other approach by means

of Wilcoxon signed ranks test. While Finner test found the improvement achieved by CLAST non-significant in some cases, Wilcoxon test finds that CLAST significantly improves the performance of each one of the other seven learners for both accuracy and G-mean. Tables III.5 and III.6 sum up the results of each test for the comparisons conducted between CLAST and the rest of the data stream classifiers. The symbols $\oplus$ and $\ominus$ mean that CLAST significantly improved/degraded the performance of the method in the corresponding column. Likewise, the symbols $+$ and $-$ denote a non-significant improvement/degradation.

Table III.5: Comparison of the performance of CLAST with the remaining online learners by means of a post-hoc Finner test after Friedman's test rejected null hypothesis of equality of all learners. The same comparison is conducted for both accuracy and G-mean.

	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	$\oplus$	$+$	$+$	$\oplus$	$+$	$\oplus$	$\oplus$
G-mean	$\oplus$	$+$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$

[1] $\oplus/\ominus$: CLAST significantly improves/degrades the performance of the method in the column
$+/-$: CLAST improves/degrades the performance of the method in the column

Table III.6: Pairwise comparison of the performance of CLAST with the remaining online learners by means of a Wilcoxon signed ranks test. Same comparison is conducted for both accuracy and G-mean.

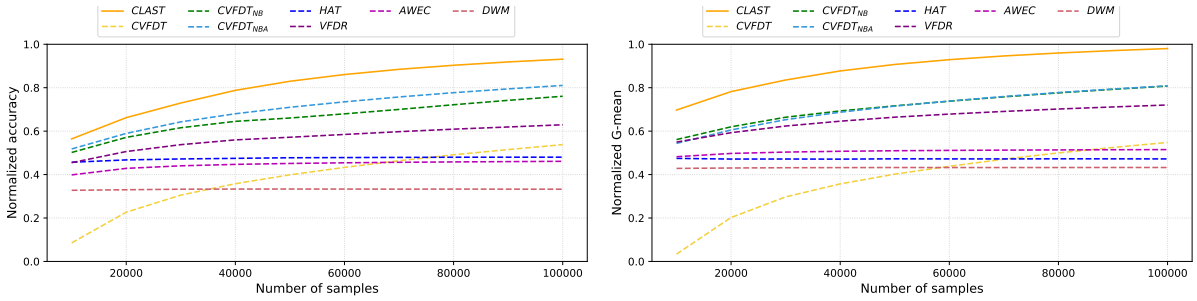
	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
G-mean	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$

[1] $\oplus/\ominus$: CLAST significantly improves/degrades the performance of the method in the column
$+/-$: CLAST improves/degrades the performance of the method in the column

Previously we have compared the performance of the different approaches based on the results obtained once all the examples have been processed. In a test-then-train scheme, every example is first used for evaluation and, then, for learning, and the accuracy (or G-mean) at a certain instant refers to all the examples received up to that instant. Therefore, it is possible to monitorize the performance as the examples are processed. Figure III.6 illustrates the evolution of the average normalized performance of each classifier. CLAST obtains the best performance, maintaining a practically constant advantage over the next classifier. The same effect is observed for both accuracy and G-mean. In general, the algorithms keep identical positions all the time.

Note that the performances of all the algorithms follow a constant trend. This behavior is not strange given that the majority of the addressed problems are not data stream problems and, in addition, the figure shows the average performance of twenty datasets. Hence, the objective of the figure is not to analyze the performance swings that

would be expected in a real-world data stream problem, but to check whether the scenario found once all the examples have been processed is an anomaly or the dominant pattern.



(a) (b)

Figure III.6: Evolution of the average performance in the collection of datasets as the amount of data processed increases. Performance is assessed by: (a) accuracy and (b) G-mean.

III.3.2 Comparison with batch approaches

In this section we compare the performance of CLAST with six widely used learning algorithms. These six learners are general purpose classifiers, i.e., not designed for data streams, that do not implement an incremental learning strategy but look at the dataset as a whole.

III.3.2.1 Experimental setup

The methodology followed is similar to the one presented in the previous section. However, some aspects had to be adapted to the use of static classifiers. We selected 19 out of the collection of 22 problems, whose characteristics are summed up in Table III.1. Skin, covtypeNorm and kddcup are left out of this set of experiments due to their large size. Both test accuracy rate and G-mean were used to measure the performance of the different approaches.

Traditional offline classifiers are not designed to process data sample by sample in a flow manner. They are not prepared to learn in an incremental way. Hence, the test-then-train experimental methodology used in the previous section is discarded here. In this case, the experiments ran on 90/10 train-test partitions. CLAST uses the same train-test partitions than the rest of the methods but we allow the algorithm to perform 10 rounds over the training set, i.e., it can see each training example 10 times. Thirty partitions of each dataset are generated based on thirty different seeds. Hence, each sample is used on average three times for testing and the remaining 27 times for training. The results shown below are averages over the thirty runs.

The performance of CLAST is compared with the following six widely used learning algorithms: Gaussian Naive Bayes (NB), Decision Tree (DT), k-Nearest Neighbors (kNN), Support Vector Classification (SVC), Random Forest (RF) and Bagging. These six learners are representative of different types of knowledge representation and learning strategies. NB (Zhang, 2004) is a Naive Bayes classifier in which the likelihood of the features is assumed to be Gaussian. DT is an optimized version of the CART algorithm (Breiman et al., 1984). kNN (Cover and Hart, 1967), instance-based learner where each example is classified based on a simple majority vote of the k-nearest neighbors, no internal model is built by the learner. SVC (Chang and Lin, 2011) is a Support Vector Machine approach for classification problems which implements "one-versus-one" strategy for multi-class scenarios. RF (Breiman, 2001) and Bagging (Breiman, 1996) are both ensemble meta-estimators that fit a number of base classifiers on different sub-samples of the original dataset and then aggregate their predictions to decide a final prediction. In RF the base classifiers are always decision trees and each sub-sample implies a selection of both samples and features. In contrast to the original publication, the scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class. Different types of learners can be used as base classifiers in Bagging, although we also use decision trees, and the random sub-samples are drawn as random subset of samples with replacement.

All these learners were run using Scikit-learn (Pedregosa et al., 2011). We followed the recommended parameter values given in Scikit-multiflow for each classifier. We only changed the number of neighbors k in kNN and the number of estimators n in RF and Bagging. We tried several values of both k and n for all the datasets, and selected $k = 3$ and $n = 5$ since they generally allowed to achieve higher performance ratios. For CLAST, we use the same configuration as in the previous section: $N = 5000$ (population maximum size), $\eta = 20$, $\theta_{sub} = 500$, $\theta_{GA} = 50$, $\delta = 0.05$ and $\nu = 10$. We did not introduce the same learner with different parameter settings in the comparison to avoid biasing the statistical analysis of the results.

For the statistical analysis of the results we followed the same methodolgy employed before. Therefore, the performance of the different algorithms was compared in terms of both accuracy and G-mean by means of Friedman's test (Friedman, 1937, 1940), post-hoc Finner test (Finner, 1993), and Wilcoxon signed ranks test (Wilcoxon, 1992).

III.3.2.2 Results

In the following, we compare CLAST to the mentioned set of general-purpose learners. Table III.7 shows the accuracy of the classifiers on the collection of datasets, while Table III.8 shows the G-mean computed on the same experimental results. Both tables also include the rank of each algorithm according to its performance in each of the problems.

Additionally, Figure III.7 and Table III.9 sum up the behavior of the classifiers. The former shows the performance distribution of each algorithm on the collection of

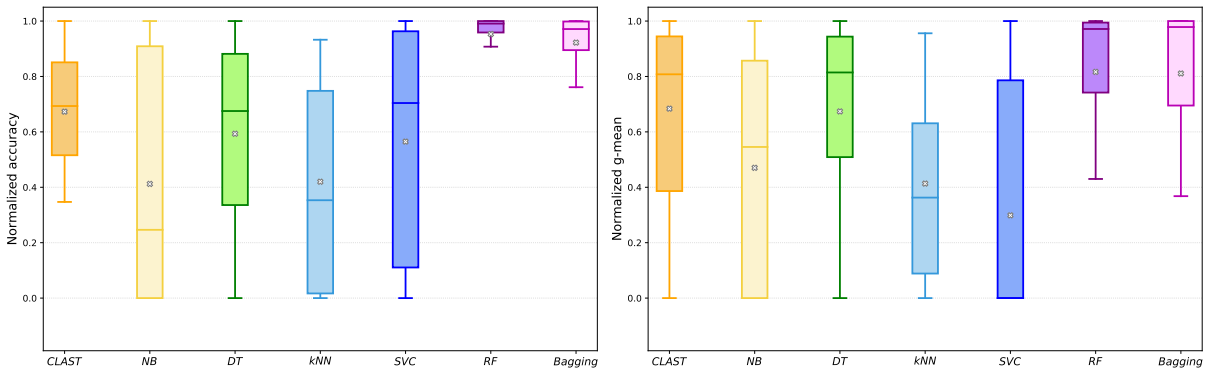
Table III.7: Test accuracy and rank of CLAST and the different offline approaches for each dataset.

Data	*CLAST*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
app	77.64 (7)	86.15 (4)	80.97 (6)	85.15 (5)	87.00 (2)	87.09 (1)	86.82 (3)
aut	62.99 (4)	52.58 (5)	83.35 (3)	41.76 (6)	40.78 (7)	88.68 (1)	87.82 (2)
ban	87.29 (5)	61.29 (7)	87.11 (6)	88.35 (4)	90.30 (1)	89.33 (2)	89.16 (3)
bnd	59.67 (6)	46.01 (7)	62.28 (5)	65.69 (3)	63.03 (4)	74.08 (1)	72.88 (2)
bre	72.56 (3)	73.53 (1)	64.23 (7)	67.51 (6)	70.78 (4)	72.67 (2)	70.16 (5)
car	91.40 (5)	62.65 (7)	98.30 (1)	85.15 (6)	93.60 (4)	97.98 (3)	98.19 (2)
eco	74.00 (6)	60.54 (7)	78.01 (5)	81.71 (4)	84.92 (2)	85.83 (1)	82.84 (3)
hay	77.50 (5)	67.50 (6)	81.67 (4)	66.67 (7)	82.29 (1)	81.88 (2)	81.88 (2)
hea	80.86 (4)	84.32 (2)	75.93 (5)	67.65 (7)	69.14 (6)	84.44 (1)	82.10 (3)
mam	78.22 (1)	77.90 (2)	74.44 (7)	76.87 (6)	77.49 (3)	77.25 (4)	77.04 (5)
pag	93.87 (5)	88.63 (7)	96.24 (3)	95.77 (4)	90.47 (6)	97.37 (1)	97.13 (2)
pim	75.78 (2)	75.66 (3)	69.88 (7)	69.96 (6)	76.04 (1)	75.65 (4)	75.48 (5)
sah	67.10 (4)	71.35 (1)	61.62 (6)	58.07 (7)	66.30 (5)	67.17 (3)	68.18 (2)
tae	50.38 (5)	51.00 (4)	63.67 (3)	40.92 (6)	36.03 (7)	63.86 (2)	64.76 (1)
tic	95.75 (2)	71.19 (7)	87.51 (5)	79.48 (6)	89.53 (4)	94.92 (3)	96.14 (1)
tit	79.16 (3)	78.23 (4)	78.16 (5)	68.13 (7)	68.76 (6)	82.98 (1)	82.98 (2)
veh	63.43 (5)	45.19 (7)	71.35 (3)	65.10 (4)	49.17 (6)	74.75 (2)	75.10 (1)
wdb	93.15 (4)	94.14 (3)	93.03 (5)	92.56 (6)	91.68 (7)	95.90 (2)	95.90 (1)
yea	52.82 (5)	14.74 (7)	52.40 (6)	55.59 (4)	60.30 (3)	62.08 (1)	60.72 (2)

Table III.8: G-mean and rank of CLAST and the different offline approaches for each dataset.

Data	*CLAST*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
app	63.53 (4)	72.22 (1)	60.07 (6)	62.36 (5)	59.84 (7)	65.16 (2)	64.39 (3)
aut	28.97 (4)	5.550 (5)	58.38 (3)	0.00 (6)	0.00 (6)	76.16 (1)	73.01 (2)
ban	87.25 (5)	51.67 (7)	86.99 (6)	88.06 (4)	89.74 (1)	89.05 (2)	88.92 (3)
bnd	58.57 (3)	36.18 (6)	57.67 (4)	56.42 (5)	0.00 (7)	65.88 (2)	66.32 (1)
bre	64.79 (1)	63.92 (2)	52.52 (5)	48.55 (6)	47.25 (7)	55.24 (3)	54.71 (4)
car	70.47 (5)	0.00 (7)	95.41 (2)	52.12 (6)	75.58 (4)	94.54 (3)	96.89 (1)
eco	25.99 (6)	0.00 (7)	32.51 (5)	65.00 (2)	72.45 (1)	59.43 (3)	52.59 (4)
hay	77.75 (5)	68.15 (6)	82.79 (2)	55.71 (7)	82.44 (4)	82.99 (1)	82.77 (3)
hea	80.30 (4)	83.59 (2)	75.02 (5)	65.59 (6)	63.50 (7)	83.75 (1)	81.38 (3)
mam	78.32 (1)	77.95 (2)	74.01 (7)	76.61 (6)	77.48 (3)	77.10 (4)	76.87 (5)
pag	58.20 (5)	53.21 (6)	78.95 (3)	68.15 (4)	0.00 (7)	82.24 (1)	80.64 (2)
pim	73.75 (1)	70.84 (2)	65.60 (6)	64.87 (7)	66.06 (5)	69.28 (4)	70.53 (3)
sah	66.32 (2)	68.55 (1)	56.31 (5)	46.95 (6)	19.47 (7)	57.06 (4)	58.53 (3)
tae	35.14 (4)	31.77 (6)	61.64 (2)	34.85 (5)	20.68 (7)	61.15 (3)	62.42 (1)
tic	94.51 (1)	44.30 (7)	85.19 (4)	71.89 (6)	84.07 (5)	92.83 (3)	94.30 (2)
tit	76.92 (4)	75.97 (5)	76.95 (3)	64.10 (6)	55.87 (7)	80.87 (2)	80.98 (1)
veh	55.88 (5)	37.78 (7)	68.60 (3)	60.18 (4)	44.01 (6)	70.86 (2)	71.84 (1)
wdb	91.57 (5)	93.01 (3)	92.84 (4)	91.38 (6)	89.07 (7)	95.27 (2)	95.45 (1)
yea	0.00 (4.5)	0.00 (4.5)	1.42 (1)	0.00 (4.5)	0.00 (4.5)	0.00 (4.5)	0.00 (4.5)

problems by means of *boxplots*, while the latter presents the average performance and rank of each classifier. As in the previous section, we first normalized the accuracy and G-mean values for each problem and then averaged them or illustrated their distribution. This way, we prevent the different characteristics of the problems from affecting their weight on the global comparison. Different problems may present great differences in the minimum, maximum and variance of classification performance.



(a) (b)

Figure III.7: Distribution of the results achieved by each classifier in terms of: (a) normalized accuracy, and (b) normalized G-mean.

Table III.9: Comparison of the average performance of CLAST with the performance of the offline learners.

		CLAST	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
Accuracy	Avg	0.67 (3)	0.41 (7)	0.59 (4)	0.42 (6)	0.56 (5)	0.95 (1)	0.92 (2)
	Rank	4.26 (4)	4.79 (5)	4.84 (6)	5.47 (7)	4.16 (3)	1.95 (1)	2.47 (2)
G-mean	Avg	0.68 (3)	0.47 (5)	0.67 (4)	0.41 (6)	0.30 (7)	0.82 (1)	0.81 (2)
	Rank	3.53 (3)	4.42 (5)	4.00 (4)	5.21 (6)	5.26 (7)	2.37 (1.5)	2.37 (1.5)

Several observations can be drawn from the results. First, it is worth highlighting the good average performance presented by CLAST. It is the third best method in accuracy, G-mean and G-mean ranking, and the fourth in accuracy ranking. Hence, the incremental learning process does seem not to limit the capabilities of CLAST for learning the underlying class distribution of the problems. Its average performance is only clearly surpassed by RF and Bagging. The two ensemble approaches exhibit a clear advantage over the rest of the learners. In average, the behavior of CLAST is similar to DT and SVC, and better than NB and kNN. Some of these observations can be extended, beyond the average, to the distribution of performance across the problem collection (Figure III.7). RF and Bagging are clearly outperforming the rest of the learners. CLAST performs similarly to DT. This similarity is especially noticeable if the reference measure is G-mean. NB and SVC present a particularly high variance. NB, SVC and kNN perform worse than CLAST on most of the problems.

The results of the statistical tests support some of the observations outlined above. Friedman's test rejected the null hypothesis of equality of all learners. Post-hoc Finner test found that the performance of CLAST can be considered statistically equivalent to that of all the other learners. According to the results of Friedman and Finner tests, CLAST outperforms NB, DT and kNN for both accuracy and G-mean, and SVC only for G-mean. However, this improvement is not statistically significant. On the other hand, if CLAST is compared in a pairwise way with each of the batch learners, Wilcoxon signed ranks test considers that CLAST significantly improves NB, kNN and SVC (for G-mean), and significantly degrades the performance of RF and Bagging.

Tables III.5 and III.6 sum up the results of Finner and Wilcoxon tests, respectively. The symbols $\oplus$ and $\ominus$ imply that CLAST significantly improved/degraded the performance of the method in the corresponding column. Likewise, the symbols $+$ and $-$ denote a non-significant improvement/degradation.

Table III.10: Comparison of the performance of CLAST with the set of offline learners by means of a post-hoc Finner test after Friedman's test rejected null hypothesis of equality of all learners. The same comparison is conducted for both accuracy and G-mean.

	NB	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
Accuracy	$+$	$+$	$+$	$-$	$-$	$-$
G-mean	$+$	$+$	$+$	$+$	$-$	$-$

[1] $\oplus/\ominus$: CLAST significantly improves/degrades the performance of the method in the column
$+/-$: CLAST improves/degrades the performance of the method in the column

Table III.11: Pairwise comparison of the performance of CLAST with the set of offline learners by means of a Wilcoxon signed ranks test. Same comparison is conducted for both accuracy and G-mean.

	NB	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
Accuracy	$\oplus$	$-$	$\oplus$	$+$	$\ominus$	$\ominus$
G-mean	$\oplus$	$-$	$\oplus$	$\oplus$	$\ominus$	$\ominus$

[1] $\oplus/\ominus$: CLAST significantly improves/degrades the performance of the method in the column
$+/-$: CLAST improves/degrades the performance of the method in the column

III.3.3 Real world data stream problems

In this section, we present the experiments conducted on different Real-World (RW) data streams, where data are received in a flow manner and data distribution may vary over time (*concept drift*), and compare the results achieved by CLAST with those of other data stream classifiers.

III.3.3.1 Experimental setup

The methodology followed is equivalent to the one used in Section III.3.1. CLAST is compared to the same set of seven online learners: CVFDT, CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, HAT, VFDR, AWEC and DWM. By the same token, a test-then-train scheme is employed, and the algorithms are compared according to both accuracy and G-mean. Nonetheless, in this case, the algorithms are tested on four real-world data stream problems. Furthermore, we analyze each problem independently, including the performance evolution of the algorithms along each stream. Since we are dealing with real data stream problems, it is expected for the performance evolution of the learners not to describe a steadily rising curve but to suffer ups and downs.

Table III.12 summarizes the main characteristics of each dataset. Below, we introduce each of the real-world problems and present the corresponding comparative analysis between CLAST and the other online learners.

Table III.12: Properties of the RW datasets. The columns of the table describe: the name of the dataset (Dataset), the number of instances (#Inst), the number of attributes (#Att), the number of real attributes (#Re), the number of integer attributes (#In), the number of nominal attributes (#No), the number of classes (#Cl), the proportion of instances of the minority class (%Min), and the proportion of instances of the majority class (%Maj)

Alias	Dataset	#Inst	#Att	#Re	#In	#No	#Cl	%Min	%Maj
eeg	EEG Eye State	14980	14	14	0	0	2	44.90	55.10
pow	Powersupply	29928	2	2	0	0	4	25.00	25.00
lon	London Shared Bikes	17414	11	4	1	6	4	24.53	25.47
bik	Bike Rental	17379	11	4	1	6	4	24.85	25.05

III.3.3.2 Detecting open/close eyes through EEG

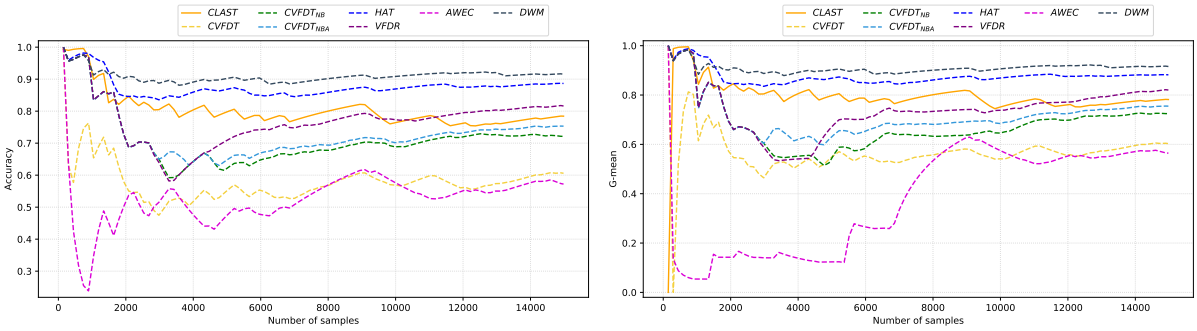
We employ the *EEG Eye State* dataset (Roesler, 2013; Dua and Graff, 2017). The dataset consists of fourteen input EEG values and an output variable indicating the eye state (open/close). The dataset is quite balanced between both classes, around 55% of the samples are labeled with eye-open state and the remaining 45% as eye-close state. The EEG values form one continuous EEG measurement recorded with the Emotiv EEG

Neuroheadset. The eye state was detected through a camera during the EEG recording and added later manually to the file after analyzing the video frames. The EEG measurement lasted 117 seconds what lead to a dataset formed by 14980 instances. The dataset respects the original chronological order of the samples.

Table III.13 details the test accuracy and G-mean obtained with each data stream classifier. For both accuracy and G-mean, the corresponding rank of each algorithm is indicated in brackets. The values in the table match the end of the stream. Moreover, Figure III.8 illustrates the performance evolution of the classifiers along the entire stream.

Table III.13: Comparison of performance between CLAST and seven other online learners in EEG Eye State problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	78.44 (4)	60.59 (7)	72.15 (6)	75.31 (5)	91.31 (2)	81.81 (3)	57.21 (8)	91.52 (1)
G-mean	78.18 (4)	60.30 (7)	72.50 (6)	75.49 (5)	91.30 (2)	82.22 (3)	56.44 (8)	91.56 (1)



(a) (b)

Figure III.8: Performance evolution as the amount of data processed increases for EEG Eye State data stream. Performance is assessed by: (a) accuracy and (b) G-mean.

The experimental results show that at the end of the stream, once all the examples have been processed, CLAST is ranked as the fourth best learner. Its performance is close to AWEC. DWM and HAT are the two best ranked method, their performances are quite similar and quite distanced from those of the rest of approaches. If we pay attention to the performance evolution throughout the flow, there are some additional aspects that are worth to mention.

The particular characteristics of this dataset cause an alternation between classes along the stream. The stream is composed of periods of open and closed eyes of variable duration that alternate over time. This explains why at the beginning all the classifiers predict correctly 100% of the examples. It corresponds to the first open-eye period, during which all the examples received belong to the open-eye class. DWM and HAT are still the two algorithms that present the best behavior. They perform well and remain very

stable along time. CLAST, despite being overtaken by VFDR towards the end, occupies the third position for a significant amount of time. In addition, it demonstrates, as DWM and HAT, a very stable performance. However, VFDR suffers a quite important performance decrease during the first part of the stream. Just behind CLAST and VFDR are $CVFDT_{NB}$ and $CVFDT_{NBA}$, which follow a similar evolution to VFDR but with a slightly lower performance. Although CLAST is not the learner that obtains the best performance in this problem, it obtains competitive results. It achieves a performance superior to most of the other classifiers and exhibits a stable behavior throughout time. Its performance is not abruptly affected by the changes in data distribution.

III.3.3.3 Powersupply

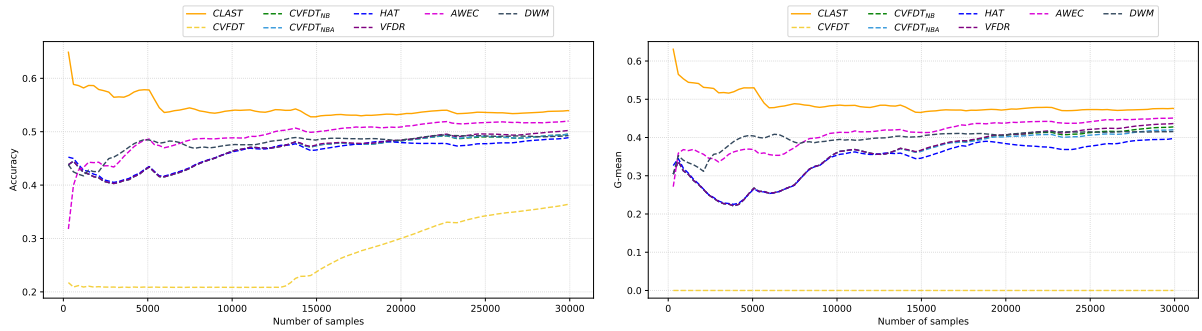
Powersupply stream (Zhu, 2010) contains hourly power supply of an Italian electricity company which records the power from two sources: power supply from main grid and power transformed from other grids. This stream contains three year power supply records from 1995 to 1998. The learning task is aimed at predicting which part of the day the current power supply belongs to. The concept drifting in this stream is mainly driven by the issues such as the season, weather, hours of a day (e.g., morning and evening), and the differences between working days and weekend.

The test accuracy and G-mean obtained by each data stream classifier at the end of the data stream are shown in Table III.14, along with the ranking position of each classifier. The evolution of the performance of the classifiers across the stream is illustrated in Figure III.9.

Table III.14: Comparison of performance between CLAST and seven other online learners in Powersupply problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	53.94 (1)	36.4 (8)	49.53 (4)	49.52 (5)	48.86 (7)	50.22 (3)	51.98 (2)	49.27 (6)
G-mean	47.59 (1)	0.00 (8)	42.76 (4)	42.00 (5)	39.71 (7)	43.64 (3)	45.07 (2)	41.55 (6)

As shown in Table III.14, at the end of the stream, CLAST obtains the highest accuracy and G-mean, followed by AWEC and VFDR. Figure III.9 allows us to verify that this first position of CLAST is not limited to the final timestamp of the stream. In fact, the greatest difference between CLAST and the other approaches is observed at early stages of the stream. CLAST, as opposed to most of the rest of classifiers, gets the highest accuracy and G-mean values at the beginning of the stream. When around half of the instances from the stream have been processed, most of the algorithms, including CLAST, seem to reach a steady level of performance that continues until the end.



(a) (b)

Figure III.9: Performance evolution as the amount of data processed increases for Power-supply data stream. Performance is assessed by: (a) accuracy and (b) G-mean.

III.3.3.4 London Bike Sharing

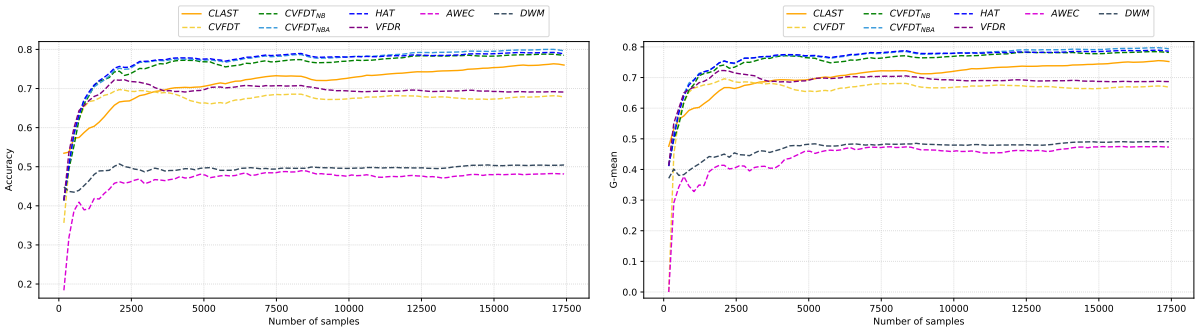
London Bike Sharing (Mavrodiev, 2020), powered by TfL Open Data, is composed by the combination of information about: (1) shared bikes demand in London, (2) weather data and (3) bank holidays in United Kingdom. The aim of the classification task addressed is to be able to predict if the bike demand will be very low, low, high or very high based on: weather conditions, season, and whether it is a bank holiday or weekend. It is worth to mention that this problem could be also addressed as a regression task. The dataset includes a continuous variable with the bike share count. We have transformed this continuous variable into a categorical one to approach the problem from a classification perspective.

Table III.15 compares the test accuracy and G-mean obtained by CLAST with that of the other seven data stream classifiers at the end of the stream. This table also includes the rank of each algorithm for each evaluation metric. Figure III.10 complements Table III.15 by showing the performance evolution of the algorithms as the stream is processed.

Table III.15: Comparison of performance between CLAST and seven other online learners in London Bike Sharing problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	76.00 (4)	67.81 (6)	78.52 (3)	79.67 (1)	78.88 (2)	69.08 (5)	48.12 (8)	50.41 (7)
G-mean	75.23 (4)	66.87 (6)	78.22 (3)	79.41 (1)	78.59 (2)	68.66 (5)	47.28 (8)	49.03 (7)

In this case, the performance of CLAST is improved by that of three other algorithms: $\text{CVFDT}_{\text{NBA}}$, HAT and CVFDT_{NB}. Figure III.10 shows that there are no strong variations in the performance of the different algorithms once the initial learning curve has been overcome. All the algorithms exhibit fairly stable behavior once the first 2500 examples are



(a) (b)

Figure III.10: Performance evolution as the amount of data processed increases for London Bike Sharing data stream. Performance is assessed by: (a) accuracy and (b) G-mean.

exceeded and until the end of the stream. This figure also shows that there are two classifiers that clearly lag behind: AWEC and DWM. While CVFDT, which had shown rather poor performance in several of the previous real-world problems, shows a better behavior in this scenario. Finally, we can see how the three best-ranked classifiers ($\text{CVFDT}_{\text{NBA}}$, HAT and CVFDT_{NB}) obtain almost identical results throughout the stream. CLAST is the next best performer and shows an ascending trend that allows it to continuously close the gap.

III.3.3.5 Bike Rental

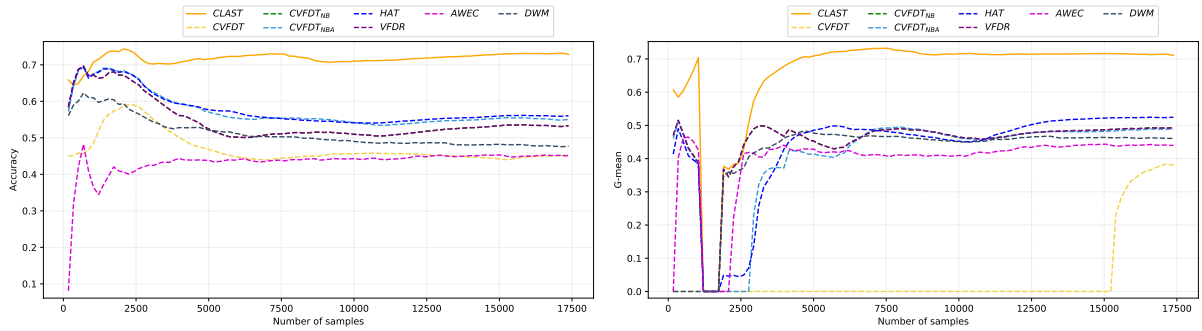
Bike Rental (Bansal, 2020; Fanaee-T and Gama, 2014) includes hourly bike-sharing data for a 2-year long period (from January 1 2011 to December 31 2012) from Capital Bikeshare System (Washington D.C., USA) along with weather and seasonal information. The prediction task addressed consists on correctly estimating if the number of riders for each hour of each day is very low, low, high or very high. Such prediction is based on input variables related to hour, weekday, month and season, as well as, weather conditions. As in the case of London Bike Sharing, the dataset includes the bike sharing count as a continuous variable and we turned this variable into a categorical variable so the problem can be addressed as a classification task.

Table III.16 presents the results of the classifiers in terms of test accuracy, G-mean and ranking positions. These values correspond to the end of the stream, once all the instances have been processed. Furthermore, Figure III.11 illustrates the performance evolution of the classifiers along the entire stream.

In Table III.16, we observe that CLAST obtains the highest accuracy and G-mean values at the end of the stream, i.e., based on the labels predicted for all the instances in the stream. Moreover, the improvement of CLAST over the second best ranked classifier, HAT, is noteworthy. It is evident from Figure III.11 that this advantage of CLAST over the other proposals is maintained for almost the entire duration of the stream.

Table III.16: Comparison of performance between CLAST and seven other online learners in Bike Rental problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	72.84 (1)	44.90 (8)	53.28 (4)	54.99 (3)	56.05 (2)	53.28 (4)	45.20 (7)	47.70 (6)
G-mean	71.08 (1)	38.02 (8)	49.23 (3)	48.85 (5)	52.48 (2)	49.23 (3)	44.09 (7)	46.11 (6)



(a) (b)

Figure III.11: Performance evolution as the amount of data processed increases for Bike Rental data stream. Performance is assessed by: (a) accuracy and (b) G-mean.

Moreover, if we compare the evolution of both evaluation metrics, we can observe a clear difference between them. Accuracy does not describe any abrupt variation throughout the stream and all the algorithms describe relatively stable trends. However, all the classifiers suffer an abrupt drop in G-mean that leads them to register null values shortly after having exceeded 1000 processed instances. This is probably due to the arrival of instances belonging to a new class, hitherto unknown to the algorithms: "very high" bike sharing demand. Since the algorithms have not had the opportunity to learn instances of that class, they misclassify these instances. We can observe the ability of the different algorithms to learn this new class and recover their previous G-mean levels. It is worth noting the behavior of CLAST, which quickly manages to reach high G-mean levels.

Chapter IV

Adaptive Fuzzy Partitions for association stream mining

IV.1 Introduction

In real-world problems with very high arrival rates and immense volumes of data is often difficult to find data which are completely labeled and structured. Probably, it would be more realistic to generate descriptive models with good interpretability which enables system monitoring.

In general, unsupervised learning is more directly applicable to real-world data stream problems. In particular, discovering frequent patterns and association rules is considered highly suitable to address many data stream problems where the purpose is to supervise or monitor (not predict) using independent, significant, readable, and simple models. Among the proposals in the literature, Fuzzy-CSar (Sancho-Asensio et al., 2016) is a steady-state genetic algorithm designed to discover interesting association rules from data streams in a dynamic and pure online way. Its learning process is designed to manage huge amounts of data and to adapt its knowledge to concept drifts. Furthermore, thanks to the use of fuzzy logic, it can deal with both categorical and continuous variables.

The aim of this chapter is to present an advanced version of the Fuzzy-CSar algorithm. Two main improvements are included in this new version: (1) new representation and new genetic operators to allow the use of fuzzy partitions with different granularities (number of linguistic terms), and (2) a mechanism to update the range of each attribute in an online way (this mechanism makes unnecessary to know a priori the domain of each attribute), which is a demanding feature in real-world data stream environments. Furthermore, the proposal is applied in a new challenging real-world problem. The results of this new algorithm, henceforth referred as Fuzzy-CSar-AFP (Fuzzy-CSar with Adaptive Fuzzy Partitions), are validated in a real-world Psychophysiology problem where associations between different signals from electroencephalogram electrodes are analyzed online while the subjects are subjected to undergo different activities and stimuli.

The remainder of this chapter is structured as follows. Section IV.2 describes our association stream mining proposal. In Section IV.3, we describe the real-world data streams that have been used to validate the functionality of Fuzzy-CSar-AFP, detail the experiments conducted and present their results.

IV.2 Fuzzy-CSar-AFP: Fuzzy-CSar with Adaptive Fuzzy Partitions

Figure IV.1 schematically illustrates the learning process of Fuzzy-CSar-AFP. The algorithm learns from a stream of data samples in an incremental way, performing a learning iteration every time a new sample e is received. At the beginning of each learning iteration, the system builds a *match set* $[M]$ with all those individuals from the population that match e with a degree greater than 0. If the number of individuals in $[M]$ is lower than θ_{mna} (a user-defined parameter), the *covering operator* is applied until the size of $[M]$ reaches θ_{mna}. After that, the individuals in $[M]$ are grouped by their antecedent into different *association set candidates* $[A]_i$. Each $[A]_i$ has a probability of being selected proportional to the average confidence of its individuals. The selected *association set* goes through a *subsumption process*. Next, the parameters of all the individuals in $[M]$ are updated in an incremental way. Finally, a steady-state (not generational) genetic algorithm is applied to $[A]$ if the average time since its last application on the individuals in $[A]$ is greater than θ_{GA} (also a user-defined parameter). These steps are repeated for each input data sample. Hence, the system keeps constantly updating the parameters of existing individuals and creating new promising rules online, i.e., dynamically evolving the rule population.

Fuzzy-CSar-AFP innovations on Fuzzy-CSar revolve around two main axes: (1) knowledge representation, and (2) dynamic attribute domain update. In Fuzzy-CSar-AFP, the variables in the association rules represent continuous attributes by means of fuzzy partitions which granularity can vary between rules. This allows adapting to the precision requirements of each variable in each rule. Genetic operators are newly designed for this new representation. In addition, Fuzzy-CSar-AFP incorporates a new mechanism that allows the algorithm to dynamically evolve the range of each attribute according to the progression of the input stream. In many real-world data stream problems, it is likely for attribute domains not to be known a priori. They can oscillate along the stream or simply not be bound to a static interval. With Fuzzy-CSar-AFP there is no need to specify the working range of each variable, the minimum and maximum values of each attribute are updated in real-time as data are being processed. In what follows, both the knowledge representation and the learning process of Fuzzy-CSar-AFP are further explained.

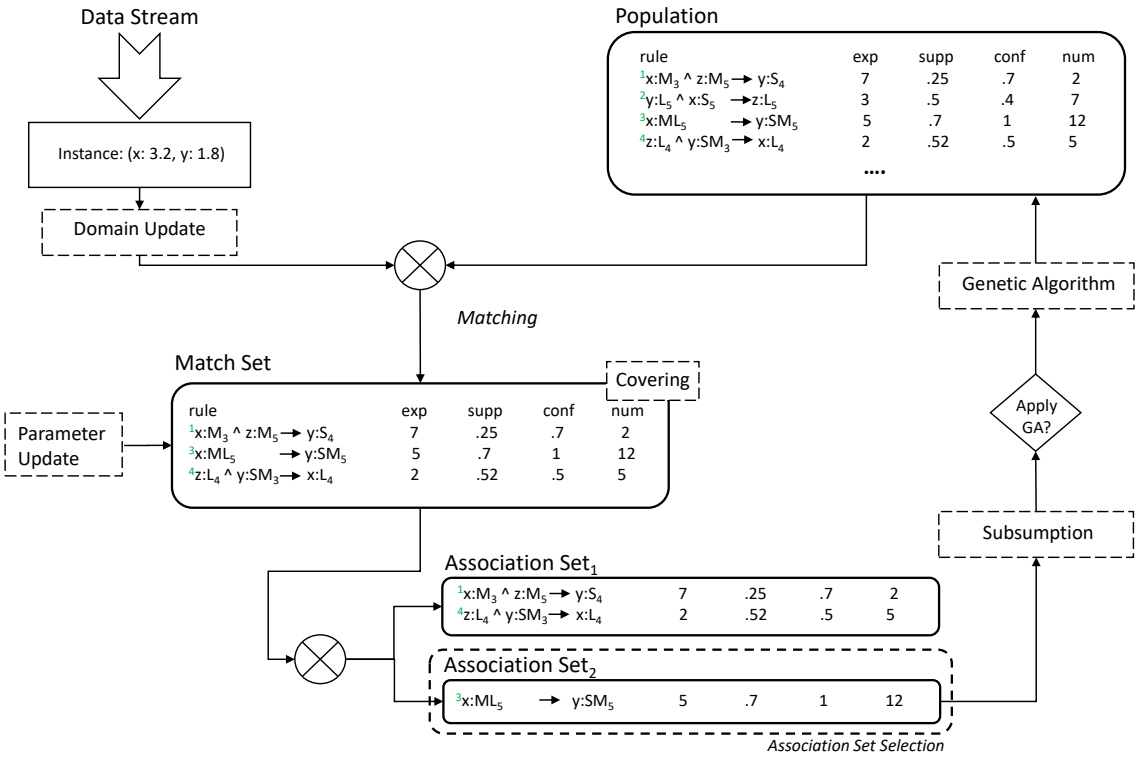


Figure IV.1: Schematic illustration of the learning interaction of Fuzzy-CSar-AFP.

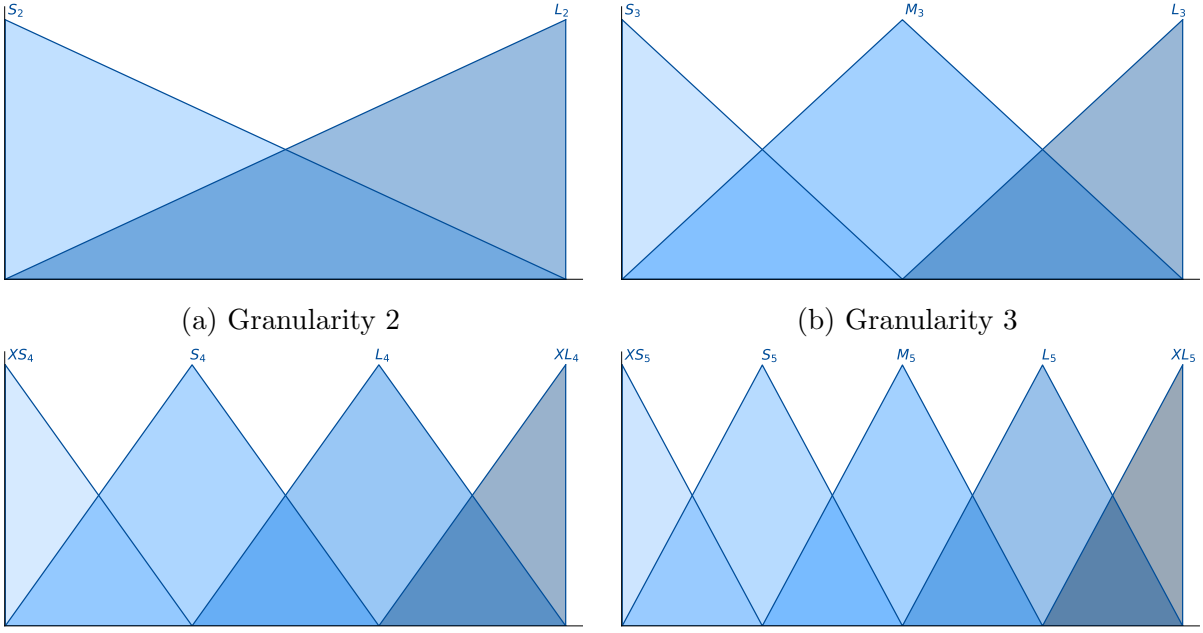
IV.2.1 Knowledge representation

Fuzzy-CSar-AFP is designed for mining association rules from data streams that contain both quantitative and categorical attributes. To that aim, it dynamically maintains a population of fuzzy rules, which is incrementally updated as the data stream is processed. Each individual in such *population* $[P]$ is formed of: (1) an association rule, and (2) a set of parameters that assess the quality of the rule. Concretely, the following eight parameters accompany each rule: (1) the support sup, (2) the confidence con, (3) the lift lif, (4) the fitness F, (5) the accuracy acc, (6) the experience exp, (7) the numerosity num, which totals the number of copies of the individual in the population, and (8) the average size as of the action sets in which the individual has been included. The rule takes the form:

$$R_k : \textbf{IF } X_i \text{ is } \tilde{A}_i^k \text{ and } \ldots \text{ and } X_j \text{ is } \tilde{A}_j^k \textbf{ THEN } X_c \text{ is } \tilde{A}_c^k \tag{IV.1}$$

where the antecedent part contains a set of l_a input variables ($0 < l_a < l$ and l is the number of variables), and the consequent consists of a single variable X_c, not included in the antecedent. Each variable in a rule is represented by a disjunction of linguistic terms $\tilde{A}_i^k = A_{i1} \vee ... \vee A_{in_i}$, where n_i is the total number of linguistic terms. Each linguistic term is an uniformly distributed triangular-shaped membership function, given its interpretability tradeoff (Casillas and Martínez-López, 2009). The semantics of the variables are defined by means of Ruspini's strong fuzzy partitions which satisfy the

equality $\sum_{j=1}^{n_i} \mu_{A_{ij}}(x) = 1$, $\forall x_i$ (for more information refer to (Pedrycz, 2018)). Since Fuzzy-CSar-AFP is able to allocate patterns with different granularities per attribute, the total number of linguistic terms n_i may vary between different patterns. This mechanism allows the algorithm to precisely adapt the generality degree of each fuzzy association rule for an optimum covering of the data. The advantage is twofold, as richer or poorer vocabulary can be used depending on performance demands and, at the same time, different support ranges and cores can be used to better locate the fuzzy set where required. This approach allows the system to evolve not only symbolic structure of the fuzzy rules but also the fuzzy set parameters.



(a) Granularity 2

(b) Granularity 3

(c) Granularity 4

(d) Granularity 5

Figure IV.2: Illustration of fuzzy partitions with four different granularities (from 2 to 5) where VS stands for *very small*, S for *small*, M for *medium*, L for *large*, and VL for *very large*.

In Figure IV.2, the fuzzy partitions for four possible granularities (from 2 to 5 fuzzy sets) are represented independently, while in Figure IV.3 we can observe how the fuzzy partitions of such four granularities overlap, making some cores from different partitions match (even if the complete linguistic terms do not). The idea of maintaining different granularities per attribute was originally proposed for classification tasks in Ishibuchi and Yamamoto (2004) and has been widely applied by the same authors since then. In classification, this approach may suffer from an important lack of interpretability as the expert expects to understand the classification policy by viewing the fuzzy rule set as a whole. However, in unsupervised tasks, the target is to provide the expert with independent patterns that explain individual association relationships among different attributes and, therefore, the fact of using different granularities for the same attribute in different rules

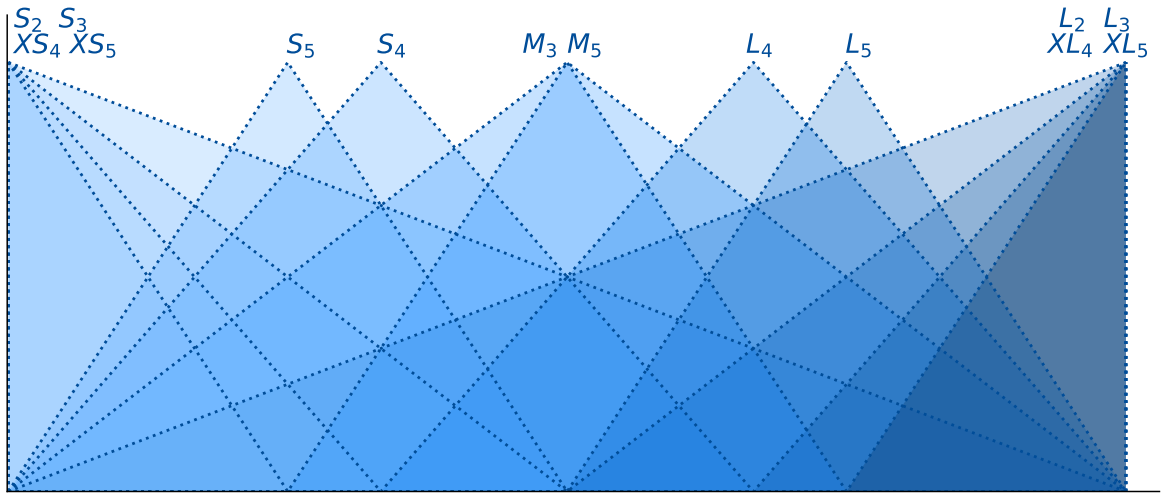


Figure IV.3: Illustration of the overlapping areas between four different granularity options (from 2 to 5).

is compatible with expert's understanding.

Every time a new datum is received, the algorithm determines which rules from the population match such datum. The matching degree between an input example e and an individual (rule) R_k is estimated as followed. First, the matching degree $\mu_{\tilde{A}_i^k}$ of each antecedent variable x_i is computed as the T-conorm (disjunction) of the membership degrees $\mu_{A_j^k}(e_i)$ of all of its linguistic terms. In this case, the *bounded sum* $sum(min[1, \mu_{A_i}(x) + \mu_{A_j}(x)])$ is employed as T-conorm ($\mu_{\tilde{A}_i^k} = \wedge \mu_{A_j^k}$). If the value of e_i is unknown (missing value), the system considers $\mu_{\tilde{A}_i^k}(e_i) = 1$. Then, the matching degree of the antecedent part of the rule is determined by the T-norm (conjunction) of the matching degrees of all the input variables. In this work, the product $\prod \mu_{\tilde{A}_i^k}(e_i)$ is used as T-norm. In the same way, the consequent matching degree is computed as the T-conorm of the membership degrees of all the linguistic terms of the consequent variable ($\mu_{\tilde{A}_c^k}$). Lastly, the matching degree of the rule is computed by means of the *Dienes implication* of the antecedent and consequent matching degree ($max 1 - \mu_{\tilde{A}(x)}, \mu_{\tilde{C}(x)}$).

IV.2.2 Learning process

During the course of each learning iteration, the system employs a set of operators to evolve the learned knowledge. Such operators need to be further explained in order to fully understand how Fuzzy-CSar-AFP works.

IV.2.2.1 Domain update

At the beginning of each learning iteration, the range of each attribute is updated according to the new incoming data sample. In Algorithm 2 the procedure followed is depicted. This domain updating algorithm is based on the incremental computation of mean (μ) and standard deviation (σ) (Welford, 1962). Given the streaming and online character of Fuzzy-CSar-AFP, this procedure had to meet two requirements: (1) inspect each data sample just once and then forget it, and (2) evolve the domain rapidly enough, being able to deal with the changing nature of data stream.

Algorithm 2: Incremental algorithm to update the domain of an attribute

`procedure Stream-Domain-Update(` *sum* `at time` t, μ `at time` t, M_2 `at time` t, n `at time` t, α, β, x `)`

Data: n is the number of samples processed at time t
x is the value of a new sample for a certain attribute
μ and σ are the stream mean and standard deviation
sum, M_2, α and β are real values
Result: *min* and *max* at time $t+1$

begin

$temp \leftarrow sum \cdot \alpha + 1$
$diff \leftarrow x - \mu$
$R \leftarrow diff \ / \ temp$
$\mu \leftarrow \mu + R$
$M_2 \leftarrow M_2 \cdot \alpha + (sum \cdot \alpha \cdot diff \cdot R)$
$sum \leftarrow temp$
$\sigma^2 \leftarrow (M_2 \cdot n) \ / \ (sum \cdot (n-1))$
$(min, max) \leftarrow (\mu - \beta \cdot \sigma^2, \mu + \beta \cdot \sigma^2)$

end

Two control parameters are included in Algorithm 2: β, which allows to control the percentage of samples covered by the range, and a decay factor α, which allows to increase the influence of the most recent input values, penalizing the oldest ones. If $\alpha = 1$, no penalization is applied. The lower α is, the higher the relevance of the most recent data. In case of $\alpha = 0$, μ matches the last value received and $\sigma = 0$. Note that a traditional weighted approach was not considered an appropriate choice since a priori all examples are equally relevant; the weight of the examples decreases as they get older. Different formulas could be used to update μ and σ when a new input value arrives. However, many of them suffer from numerical instability (Chan et al., 1983; Ling, 1974).

Thanks to this mechanism, every attribute minimum and maximum values are incrementally updated without needing to know the domain of each input attribute a priori or storing any sliding window from the data stream.

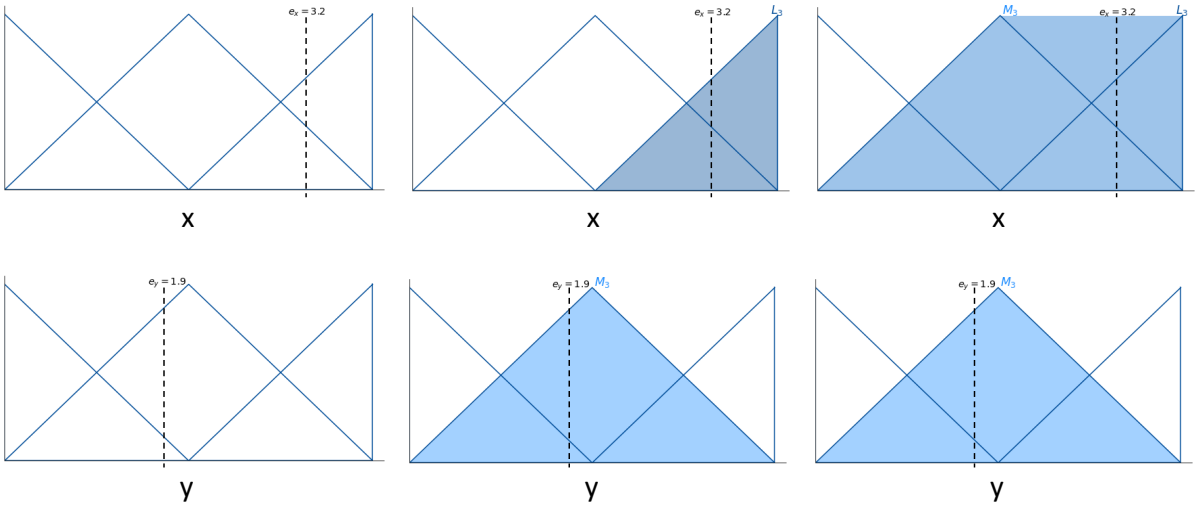
IV.2.2.2 Covering operator

This operator is aimed at generating new association rules when there are not enough matching rules for the new data sample in the current pool. Given an input data sample e, this operator creates a new individual that matches e with *maximum* degree. Each variable e_i is assigned a probability $1 - P_\#$ (where $P_\#$ is a user-defined parameter) of being in the antecedent of the rule, taking into account the following two constraints: (1) the antecedent can not be empty, i.e., at least, one variable has to be included; and (2) not all variables can be included in the antecedent, i.e., at most, $l - 1$ variables can be selected for the antecedent. Afterward, one of the variables not included in the antecedent is selected to be in the consequent. If the value of one (or several) of the input variables is unknown, the algorithm ignores the corresponding input variable(s). Each of the variables included in both the antecedent and consequent of the rule, are initialized with a random granularity and the linguistic term that maximizes the matching degree with e_i. The selected granularity must be between two linguistic terms and η (user-defined parameter). Then, rule generalization is included by adding any other linguistic term with probability $P_\#$, being the maximum number of linguistic terms that a variable can contain restricted by a configuration parameter. Lastly, individual's parameters are initialized as: $sup = exp = 0$, $con = num = 1$, and as is set to the actual size of $[A]$.

As an illustrative example and following the schematic of Figure IV.1, suppose that a new data sample $e = (x : 3.2, y : 1.9)$ is received and there are no matching individuals in the population. Consequently, the covering operator is triggered to generate a new matching individual. In the first place, the operator decides which variable is going to be in the antecedent part of the rule (e has only two attributes, so just one of them can be in the antecedent): assume that with probability $1 - P_\#$ the selected variable is x and, therefore, y is selected to be in the consequent. Let us assume three linguistic terms for both x and y. Figure IV.4 graphically represents the initialization and subsequent generalization of both variables x and y according to this example. Each variable is initialized with the linguistic label that maximizes the matching degree (we can assume: $x : L$ and $y : M$). Then, the rule generalization process is triggered and it adds other linguistic terms with probability $P_\#$. After the generalization process the final individual may be **if** x is $M \vee L$ **then** y is M (the generalization process has only added extra linguistic terms to variable x). At the end, the individual's parameters are set to its initial values.

IV.2.2.3 Association set candidates and selection

Association set candidates represent niches where rules expressing similar knowledge are grouped and made to compete. The best individuals will take over their niche and, therefore, highly fit individuals will evolve. Two rules are considered similar if they have the exact same variables in their antecedents, regardless of the granularity and linguistic terms of those variables. Note that this means rules with different variables in their consequent can be grouped under the same $[A]_i$. One of the *association set* candidates is



(a) Maximizing matching degree (b) Variable initialization (c) Variable generalization

Figure IV.4: Example of the definition process of the representation in a rule of two variables X and Y by the covering operator, schematically illustrated in three steps: (a) the linguistic terms that maximizes the matching degree is found, (b) such linguistic term is added to the variable, and (3) the generalization process may add extra linguistic terms to the variables.

selected following a roulette-wheel strategy. Each $[A]_i$ has a probability of being selected proportional to its accumulated confidence:

$$p^k_{sel} \leftarrow \frac{\sum_{i \in [A]_k} w_i \cdot con_i}{\sum_{j \in [M]} w_j \cdot con_j} \tag{IV.2}$$

where $w_i = 1$ if $exp_i > \theta_{exp}$ (user-defined parameter) and 1^{-10} otherwise.

IV.2.2.4 Association set subsumption

The selected *association set* $[A]$ undergoes a subsumption process aimed at reducing the number of rules that represent similar or redundant conditions.

Each rule in $[A]$ is checked for subsumption with each other rule in $[A]$. For the rule r_i to be considered a candidate subsumer of r_j, two conditions have to be met: (1) r_i has a similar confidence to r_j and it has enough experience ($con^i \geq 0.9 \cdot con^j$ and $exp^i > \theta_{exp}$); and (2) r_i is more general than r_j. r_i being more general than r_j means that all the variables of r_i are also included in r_j and, for each of these variables, its representation in r_i is also more general than in r_j. Since in Fuzzy-CSar-AFP the same variable can have fuzzy partitions with different granularity for r_i and for r_j, r_i is considered more general than r_j if: (1) all the variables of r_i are also defined in r_j, (2) the granularity of each of these variables has a lower number of fuzzy sets in r_i than in r_j, and (3) all the cores of

the linguistic terms of r_j are included in the cores of the linguistic terms of r_i. Each time a rule r_i actually subsumes a rule r_j, the numerosity of r_i is increased and r_j is removed from the rule population.

IV.2.2.5 Parameter update

At each learning iteration, once the association set subsumption process has been carried out, the parameters of the individuals in $[M]$ are updated. First of all, the experience of each individual is increased. Afterward, their support and confidence are updated.

In association rule mining, support is a measure of how frequently something appears in the data. Concretely, the support of a rule $X \to Y$ is computed as the frequency of antecedent and consequent occurring together in the database:

$$supp(X \to Y) = \frac{|X \cup Y|}{N} \tag{IV.3}$$

where X and Y are the antecedent and consequent of the rule, respectively; $|X \cup Y|$ is the number of samples in which both antecedent and consequent are met, and N is the total number of samples in the database. Furthermore, confidence indicates the frequency with which the if-then statement of the rule is found true. It is used as a way to evaluate the strength of the implication denoted in the association rule:

$$conf(X \to Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{IV.4}$$

A fuzzy association rule is, like before, an implication of the form $X \to Y$ in which:

$$X = \bigwedge_{i_i \in A} \mu_{\tilde{A}}(i_i) \quad \text{and} \quad Y = \bigwedge_{i_j \in C} \mu_{\tilde{C}}(i_j), \tag{IV.5}$$

where $\mu_{\tilde{C}(i_j)}$ is the membership degree of features in the consequent and $\mu_{\tilde{A}(i_i)}$ is the membership degree of features in the antecedent. In this situation, support is extended by using the product T-norm and confidence is extended by using the *Dienes implication* (Dubois et al., 2006):

$$supp(X \to Y) = \frac{1}{N} \sum \mu_{\tilde{A}}(X) \cdot \mu_{\tilde{C}}(Y) \tag{IV.6}$$

$$conf(X \to Y) = \frac{\sum(\mu_{\tilde{A}}(X) \cdot \max\{1 - \mu_{\tilde{A}}(X), \mu_{\tilde{C}}(Y)\})}{\sum \mu_{\tilde{A}}(X)} \tag{IV.7}$$

where $\mu_{\tilde{A}}(X)$ is the membership degree of the antecedent part of the rule and $\mu_{\tilde{C}}(Y)$ is the membership degree of the consequent part of the rule.

Given the incremental character of Fuzzy-CSar-AFP, support is not computed directly. At each learning iteration, the support of every rule in $[M]$ is updated as:

$$sup_{t+1} \leftarrow sup_t + \frac{\mu_{\tilde{A}(e)} \cdot \mu_{\tilde{C}(e)} - sup_t}{exp} \tag{IV.8}$$

being $\mu_{\tilde{A}(e)}$ and $\mu_{\tilde{C}(e)}$ the matching degree of the antecedent and consequent of the rule, respectively. And, after that, confidence is computed as:

$$con_{t+1} \leftarrow \frac{imp_{t+1}}{ant\_mat_{t+1}} \tag{IV.9}$$

where $imp_{t+1} \leftarrow imp_t + \mu_{\tilde{A}(e)} \cdot max\{1 - \mu_{\tilde{A}(e)}, \mu_{\tilde{C}(e)}\}$, and $ant\_mat_{t+1} \leftarrow ant\_mat_t + \mu_{\tilde{A}(e)}$. Initially, both imp_t and ant_mat_t are set to 0. Then, the lift is calculated as:

$$lif_{t+1} \leftarrow \frac{sup_{t+1}}{ant\_mat_{t+1} \cdot con\_mat_{t+1}} \tag{IV.10}$$

where $con\_mat_{t+1} \leftarrow con\_mat_t + \mu_{\tilde{C}(e)}$ and con_mat_t is initially set to 0. Afterward, the accuracy is updated as:

$$acc_{t+1} \leftarrow sup_{t+1} + 1 - (ant\_mat_{t+1} + con\_mat_{t+1} - sup_{t+1}) \tag{IV.11}$$

Thereafter, the fitness is computed as:

$$F \leftarrow \left(\frac{sup_{t+1} \cdot lif_{t+1} + acc_{t+1}}{2}\right)^{v} \tag{IV.12}$$

Finally, the estimated association set size of all the rules belonging to the selected $[A]$ is computed as the average size of all the *association sets* in which the rule has participated.

IV.2.2.6 Rule discovery component

The system uses a niche-based steady-state genetic algorithm to discover new rules. This genetic algorithm is applied to the selected $[A]$ but only when the average time from its last application upon the individuals in $[A]$ surpasses θ_{GA} (user-defined parameter). Two parents p_1 and p_2 are selected from $[A]$ using tournament selection, and they are copied into offspring ch_1 and ch_2. The offspring undergo crossover and mutation. Regarding the first one, a uniform crossover operator, which randomly chooses from which of the two parents each antecedent variable is inherited, is applied with probability P_χ. The outcoming offspring may go through one out of four different mutation types: (1) mutation of the antecedent variables $P_{I/R}$, which randomly chooses whether a new variable has to be added to the antecedent of the rule or one of the variables in the antecedent has to be removed from it (Fig. IV.5 (a)); (2) mutation of the consequent variable (with probability P_C), which selects one of the variables in the antecedent of the rule and exchanges it with the variable in the consequent (Fig. IV.5 (b)); (3) mutation of the linguistic terms of the variables (with probability P_μ), which selects one of the existing variables of the rule and mutates its value in one of three possible ways (Fig. IV.5 (c)): expansion, contraction or shift; and (4) granularity mutation, which selects one of the variables in the rule and changes the number of fuzzy sets in its fuzzy partition (Fig. IV.5 (d)).

Regarding the mutation of the linguistic terms of a variable (third type of mutation): expansion adds to the corresponding variable a new linguistic term that was not previously

represented in it; contraction chooses a linguistic term represented in the variable and removes it; so, it can only be applied to variables that have, at least, two linguistic terms; and shift switches a linguistic term by the next inferior or superior one. With respect to the granularity mutation (fourth type of mutation), the granularity of the variable can be replaced by the immediate superior or inferior option. Furthermore, if the cores of all the fuzzy sets in the linguistic partition match the ones from two granularity levels up or down, mutation to such matching level is also allowed. If the mutation implies increasing the number of fuzzy sets, the support of the new value (linguistic term) of the variable has to be contained in the support of the value before mutation. Otherwise, the operation is reversed, i.e., the support of the new value of the variable has to contain the support of the value before mutation. Note that, the minimum allowed granularity is two fuzzy sets. Therefore, a variable represented by a fuzzy partition of two fuzzy sets can only mutate its granularity in one way, that is, increasing the number of fuzzy sets. Fuzzy-CSar-AFP considers an adjacency matrix that gathers the possible mutations to be made. Table IV.1 illustrates such adjacency matrix for a maximum granularity of 5. Number 1 marks which mutations are allowed and 0 which are not, considering rows as initial linguistic terms and columns as the ones after mutation. Based on this matrix, the granularity of the variable is mutated to a new randomly selected one and the linguistic terms of the variable are replaced by the corresponding ones in the new fuzzy partition. It can be observed in the matrix that only one exception is made to the specific guidelines: mutation from M_3 to S_2 or L_2 is not allowed because the meaning of M_3 is considered very different from S_2 or L_2, despite they all share the same support.

Table IV.1: Adjacency matrix for granularity mutation (with a maximum granularity of 5 fuzzy sets).

	S_2	L_2	S_3	M_3	L_3	VS_4	S_4	L_4	VL_4	VS_5	S_5	M_5	L_5	VL_5
S_2	0	0	1	0	0	0	0	0	0	0	0	0	0	0
L_2	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S_3	1	0	0	0	0	1	0	0	0	0	0	0	0	0
M_3	0	0	0	0	0	0	1	1	0	0	0	1	0	0
L_3	0	1	0	0	0	0	0	0	1	0	0	0	0	0
VS_4	0	0	1	0	0	0	0	0	0	1	0	0	0	0
S_4	0	0	0	1	0	0	0	0	0	0	1	0	0	0
L_4	0	0	0	1	0	0	0	0	0	0	0	0	1	0
VL_4	0	0	0	0	1	0	0	0	0	0	0	0	0	1
VS_5	0	0	0	0	0	1	0	0	0	0	0	0	0	0
S_5	0	0	0	0	0	0	1	0	0	0	0	0	0	0
M_5	0	0	0	1	0	0	0	0	0	0	0	0	0	0
L_5	0	0	0	0	0	0	0	1	0	0	0	0	0	0
VL_5	0	0	0	0	0	0	0	0	1	0	0	0	0	0

The system initializes the parameters of the offspring as follows: if the crossover is not applied, the parameters are copied from the selected parent. Otherwise, the parameters are set to the average value between the corresponding parameters in both parents. In both cases, fitness is decreased to 10% of the parental fitness; experience is set to 0, and

numerosity to 1.

IV.2.2.7 Replacement mechanism

Before the just born offspring are introduced into the population, each child is checked for subsumption with their parents. If any of the parents is a candidate subsumer, its numerosity is increased and the offspring is not inserted into the population. Otherwise, subsumption with every rule in $[A]$ is checked. If one or more candidate subsumers are found, the offspring is not inserted into the population and the numerosity of the most general candidate subsumer is increased. Finally, if no subsumer is found, the new individual is inserted into $[P]$.

In case the maximum population size has already been reached, the exceeding individuals are removed from the population. The deletion probability of each individual in $[P]$ is directly proportional to their estimated association set size and inversely proportional to its fitness. In addition, if the individual R_k is experienced enough but its fitness F^k is significantly lower than the average fitness of the individuals in $[P]$, its deletion probability is further increased:

$$p^k \leftarrow \frac{d^k}{\sum_{\forall j \in [P]} d^j}, \tag{IV.13}$$

where

$$d^k \leftarrow \begin{cases} \frac{as \cdot num \cdot F_{[P]}}{F^k} & \text{if } exp^k > \theta_{del} \text{ and } F^k < \delta F_{[P]}, \\ as \cdot num & \text{otherwise,} \end{cases} \tag{IV.14}$$

where θ_{del} and δ are user-defined parameters. Thus, the algorithm pushes toward the removal of rules belonging to large association sets and, therefore, balances the individual's allocation in the different $[A]$'s. Whilst it favors seeking highly fit individuals, given that the deletion probability is increased for those rules whose fitness is much smaller than the average fitness.

IV.3 Physiological signals analysis through association stream mining

In order to test, analyze and validate the behavior of Fuzzy-CSar-AFP we employ a original real-world data stream problem. In this section, we address a challenging psychophysiological problem in which the associations between electroencephalographic signals from different electrodes recorded in subjects who are subjected to different stimuli are to be analyzed.

An extensive analysis of the results obtained is conducted. The performance of Fuzzy-CSar-AFP is widely compared with the that of Fuzzy-CSar (Sancho-Asensio et al.,

2016) and Fuzzy-Apriori (Hong et al., 1999). Furthermore, the results are interpreted from a psychophysiology perspective.

IV.3.1 Some background on the problem of exploring networks instead of single physiological signals

The physiological adaptation to an ever-changing internal and external environment is the result of a complex interaction between physiological systems who have shown to exhibit non-stationary, intermittent, scale-invariant and nonlinear behaviors. Moreover, physiological dynamics are in constant flux, responding to changes in the underlying control mechanisms caused by different physiological states or pathologic conditions. Here we employ the novel methodology of association stream mining in order to dynamically obtain association rules that explain in an online fashion the relationships between signals derived from the recording of electrophysiological brain activity at distinct electrode sites.

IV.3.1.1 Method

The database consists of physiological recordings from 50 young adult participants, divided into two groups of 25 participants per group based on fitness level. Since the consequences of a sedentary lifestyle reach far beyond the development of chronic diseases as they also directly influence brain plasticity and function (Voss et al., 2013). Numerous studies have repeatedly shown that exercise improves learning and memory, counteracts the mental decline that comes with age, and facilitates functional recovery from brain injury, disease, and depression (Vaynman and Gomez-Pinilla, 2006). The average age of the high-fit (trained - TRA) group was 22 years (age range: 21–24 years old) and of the low-fit (sedentary - SED) group 23 years (age range: 22–24 years old). Here we compare two subjects that were randomly selected as representative from their corresponding groups.

Continuous EEG data were recorded using a BioSemi Active Two system (Biosemi, Amsterdam, Netherlands) and were digitized at a sample rate of 1024 Hz with 24-bit A/D conversion and subsequently resampled at 256 Hz. The 64 active scalp Ag/AgCl electrodes were arranged according to the international standard 10—20 system for electrode placement using a nylon head cap.

The behavioural task was designed to measure vigilance by recording participants' reaction times (RT) to visual stimuli in a computer screen. Participants were instructed to respond as fast as they could once they had detected the presentation of the stimuli. They had to respond with their dominant hand by pressing the space bar on the keyboard. The task comprised a single block of 60 minutes of total duration.

The explained problem on EEG analysis has some properties that makes it an ideal benchmark for association stream mining by Fuzzy-CSar-AFP: (1) the variables (electrode signals) are continuous (which justifies the use of fuzzy logic) and the variation interval unknown (so the proposed attribute domain update mechanism makes sense here); (2) the

problem does not have any dependent or output variable, so it needs to be addressed by unsupervised learning; (3) the rate of incoming data is very high (256 per second), which justifies the need of processing data on-the-fly instead of storing them or using sliding window; and (4) there is interest from experts regarding the analysis of relationships among variables as a complementary study to their conventional time series approach.

IV.3.2 Addressing the difficulty of evaluating association stream mining

Unlike other types of problems such as supervised learning problems where there are standard measures and mechanisms to evaluate the goodness of the results of algorithms and to establish comparisons between them; in the association rules field there is no standard way to fairly establish comparisons between algorithms. Association rules discovery is an unsupervised learning problem so we do not know what the perfect association rules might be. There are several parameters for individual association rule evaluation like support, confidence, lift, etc., but even with them it is hard to reliably compare different rules sets resulting from two different algorithms or experiments. These issues increase when we talk about association streams, we have to deal with the additional difficulty of evaluating the ability of the algorithm to adapt to concept drifts. Again unlike in supervised and clustering stream fields, for association stream mining, there is no formal way to quantitatively evaluate what happens with the learned model when a concept drift occurs. To evaluate and better understand the results we have designed a set of original graphs and visualizations which would help interpret the results in two different ways: (1) comparison between the performance of Fuzzy-CSar-AFP, Fuzzy-CSar and Fuzzy-Apriori; and (2) visual tools to represent the associations discovered by Fuzzy-CSar-AFP in the data. This set of graphics and visualizations are detailed in the following.

IV.3.2.1 Attribute domain evolution

One of the novel aspects of Fuzzy-CSar-AFP is its ability to evolve the range of the input variables in real time while processing the data stream. Therefore, as part of our analysis we want to study how these dynamic limits of the algorithm are adapting to the real evolution of the variables.

For each attribute we represent a graphic plot with three functions: (1) the evolution of the bottom limit of the range (minimum value) dynamically evolved by the algorithm for the attribute (represented in red), (2) the evolution of the actual values of this attribute in the received data stream (represented in blue), and (3) the evolution of the top limit of the range (maximum value) dynamically updated by the algorithm (represented in red). With these plots we get a clearer picture of the data stream received and check how well the domains that the algorithm maintains fit the real data. This type of plot is shown in Section IV.3.4.1.

IV.3.2.2 Number of rules: minimum confidence vs. minimum support

As mentioned, part of our analysis focuses on a comparison between the performance of Fuzzy-CSar-AFP and that of other algorithms also based on fuzzy association rules. This comparison comprises different types of analysis.

Given a minimum confidence value of c, we can count the number of rules whose support is equal to or greater than s. If we plot the output of this operation for a sufficient number of s values, the resulting curve is often used to analyze the results of association rules algorithms as this type of graph represents the quality of the rules obtained.

IV.3.2.3 Evolution of the amount of good rules

In this case, the flow character of the problem is addressed. A minimum confidence threshold and a minimum support threshold are set. For each algorithm, the number of rules whose support and confidence exceed these thresholds at each moment of the experiment is represented. This graph complements the one described in the previous section by showing the dynamical behavior of the algorithms.

IV.3.2.4 Multidimensional Scaling to analyze dispersion of association rules

We consider the previous plots concerning the amount of rules obtained according to different quality criteria are not enough for a real comparison between two approaches as an algorithm can find many rules which are very similar to each other (e.g., with slight differences on the variables used in the antecedent), which in practice is not so useful for expert decision making. On the contrary, an algorithm that generates association rules with more diversity is preferable. However, this has not been thoroughly analyzed in the specialized literature yet although some visualizations have been proposed (Yamada et al., 2015; Trevisan et al., 2015).

Here we propose a new method to assess how diverse the rules obtained by the algorithm are. If the rules were illustrated as points in a two-dimensional coordinate system, then we could easily distinguish by color n groups of points matching the rules obtained by n different algorithms, and so it would be pretty simple to visually compare which group spreads their points more evenly and which has the majority of its points (association rules) concentrated in certain small zones of the plot.

To get to visualize the rules in such 2D plots, it is necessary to conduct a process that can be summarized in the following four main steps: (1) obtaining the output set of rules of each of the algorithms involved in the comparative analysis (applying some kind of filter if needed); (2) joining the m rule sets coming from the m different algorithms in one common rule set; (3) compute the distance matrix containing the distance between each pair of association rules in the common set; (4) apply MultiDimensional Scaling (MDS) (Meulman, 1992) to such distance matrix. As a result, a set of 2D points is obtained. Each

of these resulting 2D points represents an individual association rule. These points can now be easily graphically displayed in an interpretable way using, for instance, a scatter plot.

However, the distance between two association rules may not be trivial. We need to define a distance function between rules and this definition may depend on the kind of association rules that we are using. In this case, the three algorithms that we aim to compare are based on fuzzy association rules where knowledge is represented as described in Section IV.2.1. Thus, each fuzzy set can be considered trapezoidal-shaped and defined by four vertices.

Figure IV.6 shows the position of these four vertices in different scenarios depending on whether or not the variable is used in the antecedent/consequent part of the rule and, in case the variable is actually used, on the position and amount of linguistic fuzzy labels that compose the corresponding fuzzy set.

From these vertices, we estimate the distance between two rules for a given variable as the average of the absolute value difference between the vertices of the two rules. Given this distance function between variables, we define the distance between two rules as the Euclidean distance of the different variables in antecedent and consequent. In Algorithm 3, along with Algorithm 4, this distance function between association rules and all the steps carried out for its calculation are described.

The distance matrix used for MDS is built by applying the distance function described in Algorithm 3 on every pair (r_i, r_j) of fuzzy association rules in the set.

In addition to the graphic visualization, a quantitative measure to numerically assess the scattering of each group of rules is also proposed. We denote this rule diversity measure as δ_{MDS} and it can be defined as the mean of the Euclidean distances between every pair of points that represent association rules resulting from a specific algorithm. That is, to compute δ_{MDS} for each one of the n algorithms we have to: (1) cluster the points based on which algorithm produced the rule the point represents; (2) in each of the created subsets, calculate the Euclidean distance between each pair of points and compute the average of such Euclidean distances.

IV.3.2.5 Streamgraphs

A Streamgraph (or Stream graph) is a variation of the stacked area chart, where the evolution of a numeric variable (ordinate axis) following another numeric variable (axis of abscissas) is represented. As in a stacked area chart, this evolution is represented for several groups, using a distinct color for each of them. Areas are usually displaced around a central axis, what gives a nice impression of flow. This kind of plot is quite useful to study the relative proportions of a whole.

The flow look-and-feel of the streamgraph along with the fact that they are specially suitable for representing the evolution of numeric variables and to study the relative

Algorithm 3: Description of the fuzzy association rule distance function used for MDS

`function FuzzyRulesDistance( fuzzy rule` r_i`, fuzzy rule` r_j `)`

Result: Distance between r_i and r_j

begin

$sum = 0$;

foreach *variable* X_k **do**

$[S_k^{i_{ant}}] \leftarrow$ GetTrapezoidalVertices(r_i, X_k, *antecedent*);

$[S_k^{j_{ant}}] \leftarrow$ GetTrapezoidalVertices(r_j, X_k, *antecedent*);

$[S_k^{i_{con}}] \leftarrow$ GetTrapezoidalVertices(r_i, X_k, *consequent*);

$[S_k^{j_{con}}] \leftarrow$ GetTrapezoidalVertices(r_j, X_k, *consequent*);

$d_A \leftarrow$ FuzzyVariableDistance($[S_k^{i_{ant}}]$, $[S_k^{j_{ant}}]$);

$d_C \leftarrow$ FuzzyVariableDistance($[S_k^{i_{con}}]$, $[S_k^{j_{con}}]$);

Estimate the maximum possible distance d_{max} between two fuzzy sets for X_k;

$\overline{d}_A = \frac{d_A}{d_{max}}$; `// Normalized antecedent distance for` X_k

$\overline{d}_C = \frac{d_C}{d_{max}}$; `// Normalized consequent distance for` X_k

$sum = sum + \overline{d}_A + \overline{d}_C$

end

$d = \sqrt{\frac{sum}{2 \cdot n}}$;

Return d;

end

proportions of the components, make it a really good fit to represent the evolution of different subsets of association rules over time.

Given a set of rules with a certain consequent variable, a streamgraph can be used to display the evolution of the relative frequency of each of the antecedent variables along the data stream. Algorithm 5 describes the process follows to, given a certain consequent variable, obtain the information needed to generate the streamgraph from the output of an association stream mining algorithm. This process returns the weight of each antecedent variable at each point of the data stream. Such weight represents the size of the corresponding colored area. In Section IV.3.5 several streamgraphs are displayed as part of the results analysis. In each of them, a color strip (colored area) is shown for each one of the antecedent variables and its size is proportional to the relative relevance of the antecedent variable at each specific point of the stream. The sum of all these subareas corresponds to the amount of rules that include the selected consequent variable.

Algorithm 4: Description of the functions used in Algorithm 3 to get the trapezoidal representation of a variable and compute the distance between trapezoidal-shaped fuzzy sets

`function FuzzyVariableDistance( set of vertices [`S_k^i`], set of vertices [`S_k^j`] )`
Result: Distance between r_i and r_j for X_k

begin
  Extract the vertices $\{a_k^i, b_k^i, c_k^i, d_k^i\}$ from $[S_k^i]$;
  Extract the vertices $\{a_k^j, b_k^j, c_k^j, d_k^j\}$ from $[S_k^j]$;
  $d = \frac{|a_k^i - a_k^j| + |b_k^i - b_k^j| + |c_k^i - c_k^j| + |d_k^i - d_k^j|}{4}$;
  Return d;
end

`function GetTrapezoidalVertices( fuzzy rule` r_i`, variable` X_k`, rule part` p `)`
Data: *part* is either *antecedent* or *consequent*
Result: Set of vertices of the trapezoidal-shaped X_k in the p part of r_i

begin
  Check if X_k is included in the p part of r_i;
  if X_k *is in* p *of* r_i **then**
    Get the four vertices $\{a_k^i, b_k^i, c_k^i, d_k^i\}$ from the trapezoidal representation of the fuzzy set $\tilde{A}_k^i$;
  else
    $a_k^i = b_k^i = min(X_k)$;
    $c_k^i = d_k^i = max(X_k)$;
  end
  Save the four vertices as the set $[S_k^i]$;
  Return $[S_k^i]$;
end

IV.3.2.6 Dependency wheels

A Chord diagram, also known as Dependency wheel, allows to visualize weighted relationships between several entities (called nodes). Nodes are arranged along a circle, i.e., each entity is represented by a fragment on the outer part of a circular layout. Arcs are used to connect the nodes to each other. The size of both the arcs and the nodes is proportional to the weight of the corresponding associations. Color is often used to group data into different categories, which facilitates making comparisons and differentiating between groups.

The use of dependency wheels allows us to summarize in a single image much of the information contained in the set of association rules. By looking at the graph it is possible

Algorithm 5: Process followed to transform the output of an association stream algorithm into the input data of a streamgraph

Data:
Array of time stamps [T] and target consequent variable X_k
Result: [W] contains the weight of every antecedent variable at each t in [T]

```
Initialize [W] to zero;
foreach t in [T] do
    Retrieve the rule set [R] from the algorithm;
    foreach r in [R] do
        Let X_c be the consequent variable of r;
        if X_c == X_k then
            Let l_a be the number of variables in the antecedent of r;
            foreach X_i in the antecedent of r do
                Add l_a^{-1} to the weight of X_i at t;
            end
        end
    end
end
```

to identify whether there are some variables that have a particularly high/low weight and, in general, to get an idea of the strength of the association between each pair of variables. This type of graph takes a shot of how the variables are associated at a particular point of the stream. Since the set of rules evolves dynamically over time, two pictures taken at different times may not show the same scenery.

In Section IV.3.5, we include examples of dependency wheels obtained from the results of Fuzzy-CSar-AFP, which help us analyze the rules and draw conclusions. The used wheels present a double ring. While the outer ring represents the different variables of the problem, the inner ring distinguishes when the variable appears as antecedent (dashed sectors) or consequent (dotted sectors). Therefore, the arcs always connect antecedent and consequent sectors. The width of these sectors for each variable is proportional to the number of rules in which this variable appears. Likewise, the width of the arcs represent the importance of these connections. The color of each arc corresponds to the one assigned to the antecedent variable of such link. Algorithm 6 describes how the adjacency matrix required for these dependency wheels is built from a rule set.

IV.3.3 Experimental setup

Fuzzy-CSar-AFP has several configuration parameters which enable the user to adjust the behavior of the system. For most of these configuration parameters, we took as a reference the values used in Sancho-Asensio et al. (2016) for the algorithm Fuzzy-CSar,

Algorithm 6: Building process of the adjacency matrix for a dependency wheel from a set of association rules

Data: The rule set [R]
Result: The lxl adjacency matrix m (where l is the number of variables)

```
Initialize m to zero;
foreach rule r in [R] do
    Let X_c be the variable in the consequent of r;
    foreach variable X_i in the antecedent of r do
        Add l_a^{-1} to m[i, c];
    end
end
```

which were obtained experimentally following the recommendations found in Orriols-Puig et al. (2008a). Thus, here, for both Fuzzy-CSar and Fuzzy-CSar-AFP: $\upsilon = 1$, $P_{\#} = 0.2$, $P_{\chi} = 0.8$, $\{P_{I/R}, P_{\mu}, P_{C}\} = 0.1$, $\delta = 0.1$, $\theta_{GA} = 12$, $maxLingTermsPerVariable = 3$, and θ_{mna} is automatically set to the number of variables. However, here we are dealing with a challenging real-world data stream problem in which 256 samples are analyzed every second during 60 minutes per subject (i.e., about a million of samples are processed in each experiment). Due to this high rate of incoming data, it was considered convenient for the values of those parameters related to experience thresholds, as well as, the size of the population to be adjusted: $\theta_{exp} = 10000$ (a rule that has been updated for about 40 seconds is experimented enough as to consider the performance estimation reliable), $\theta_{del} = \theta_{sub} = 4000$ (after about 15 seconds the rules can be deleted or subsumed), and the population size was set to 5000 individuals.

As we have pointed before, Fuzzy-CSar-AFP is also analyzed in comparison with Fuzzy-Apriori (Hong et al., 1999), which integrates the Apriori algorithm and fuzzy sets concepts to discover interesting fuzzy association rules among quantitative values. Because Fuzzy-Apriori is not a data stream algorithm the dataset is partitioned into 30 subsets, each one containing 30720 samples (data registered during two minutes of experiment) and Fuzzy-Apriori is applied on each subset. So we can obtain results from Fuzzy-Apriori for different stretches from the original dataset, and not only at the end.

Regarding the fuzzy sets, all the variables use Ruspini's strong fuzzy semantics with 5 linguistic terms (XS, S, M, L, XL) for both Fuzy-CSar and Fuzzy-Apriori, and between 2 and 5 linguistic terms in the case of Fuzzy-CSar-AFP. The same configuration is applied to all the experiments conducted and, therefore, all the results shown in the following Section IV.3.4 have been obtained following this procedure.

Furthermore, with respect to the domain update mechanism. In our experimentation, we use $\beta = 2.5$ so about 98% of the samples are included in the range, and $\alpha = 10^{\log_{10}(0.5)/(sps \cdot s)}$ being $sps = 256$ (samples per second, i.e., Hz) and $s = 40$ (i.e., after 40 seconds the decay factor α will be faded out to 0.5).

IV.3.4 Results

The EEG data streams described in Section IV.3.1 are used in a series of experiments in order to corroborate our hypothesis that Fuzzy-CSar-AFP is able to better adapt to the peculiarities of each variable, thus obtaining better and more heterogeneous rules that could help understand the associations between the input features. We thoroughly compare Fuzzy-CSar-AFP with Fuzzy-Apriori and Fuzzy-CSar.For this comparison we employ two different versions of the mentioned data streams. The full data streams with their 64 variables are used to assess the time-performing of the algorithms. However, to analyze and compare the sets of rules obtained by the different algorithms, EEG data from six scalp locations ((Bartsch et al., 2015)) is used in order to enable a more conscious analysis. These six scalp locations are: frontal left (Fp1), frontal right (Fp2), occipital left (O1), occipital right (O2), central left (C3) and central right (C4).

The experiments can be clustered in two different categories depending on whether the algorithms employ the absolute domain of each input feature or the incremental domain evolution mechanism (Section IV.2.2.1). Comparisons are done separately for these two categories of experiments due to the difficulties of achieving a fair comparison (specially in terms of dispersion) between algorithms using different attribute domains. In any case, the main aim of the incremental domain evolution mechanism is to make the algorithm useful in real-world problems in which it is common not to know the range of such domains or where the domains may vary significantly along the data stream.

IV.3.4.1 Domain update

The domain update mechanism described in Section IV.2.2.1, allows the range that the algorithm maintains for each attribute to be incrementally updated in real time as the algorithm processes the incoming examples. Thus, each time a new example is received, the upper and lower limits of that range are updated accordingly. In Figure IV.7 and Figure IV.8 we can see the evolution of these upper and lower limits along the whole data stream for each of the subjects included in our experiments.

The domain update mechanism described in Section IV.2.2.1, allows the range that the algorithm maintains for each attribute to be incrementally updated in real time as the algorithm processes the incoming examples. Thus, each time a new example is received, the upper and lower limits of that range are updated accordingly. In Figure IV.7 and Figure IV.8 we can see the evolution of these upper and lower limits along the whole data stream for each of the subjects included in our experiments.

Delimiting the range of each variable based on the incrementally computed μ and σ allows the range to evolve along time but to avoid overreacting to noise and outliers. Moreover, in this case, using the actual maximum and minimum values would imply that several linguistic labels would not ever be used. For instance, in the case of the trained subject (Figure IV.7) for electrode C4 some linguistic labels would cover values that are only reached once in the whole stream.

IV.3.4.2 Number of rules stratified by support

In this section, we compare the amount of rules obtained through Fuzzy-CSar-AFP and Fuzzy-Apriori according to a minimum support threshold. Since Fuzzy-CSar-AFP does not conduct any kind of support filtering during its learning process, we are filtering the output rule sets from the algorithms based on the specified minimum support thresholds.

Figure IV.9 shows how the amount of rules evolve as the demanded minimum support increases (from 0.0 to 1.0). This is shown for both Fuzzy-CSar-AFP and Fuzzy-Apriori, for three different confidence thresholds (0.75, 0.80 and 0.85), and for two different scenarios: (1) both algorithms use the full range of each attribute during the whole stream, and (2) both algorithms incrementally update the range o feach input attribute using the method described in Section IV.2.2.1.

Table IV.2: Number of rules ($\#R$) obtained by Fuzzy-Apriori with the lowest minimum supports. The amount of rules is shown for the three confidence thresholds (0.75, 0.80 and 0.85) included in Figure IV.9 and for both trained (TRA) and sedentary (SED) subjects when using evolving domains.

min. supp.	**0**			**0.01**			**0.02**		
min. conf.	**0.75**	**0.8**	**0.85**	**0.75**	**0.8**	**0.85**	**0.75**	**0.8**	**0.85**
TRA #R	317514	274020	212082	2608	794	30	846	42	0
SED #R	156852	135884	104812	1282	334	17	395	23	1

When using static domains and low support thresholds, the number of rules generated by Fuzzy-Apriori is much higher but it decreases very quickly, falling below Fuzzy-CSar-AFP before the support reaches 0.2. Fuzzy-CSar-AFP gets a lower number of rules when no minimum support is required but maintains that level of rules for significantly higher supports. However, when evolving domains are used, Fuzzy-CSar-AFP obtains a larger number of rules from very close to zero support values to the end. If we focus on support values between 0.0 and 0.1 our attention is drawn to the fact that the number of Fuzzy-Apriori's rules starts descending quickly, then seems to stabilize for a moment and finally sharply descends again. Before the minimum support reaches 0.1 the number of rules from Fuzzy-Apriori falls to zero for both subjects and for the three possible confidence thresholds. This evolution in the amount of rules discovered by Fuzzy-Apriori is overriding to the choice of rules quality thresholds for other comparatives and analysis shown in this paper. Unlike Fuzzy-Apriori, Fuzzy-CSar-AFP draws a smoother curve and continues generating rules even for quite high minimum supports, and for both attribute range approaches.

IV.3.4.3 Evolution of the amount of quality rules along the stream

Figure IV.10 represents the evolution of the number of rules beating minimum support and confidence values as the amount of samples processed by the algorithm grows. Algorithms are distinguished by color. Note that Fuzzy-Apriori is not a data stream approach and, therefore, it is executed in consecutive data batches. Each batch comprises two minutes of information recording, i.e., 30720 samples. As in the previous section, we consider two different experimental setups: static domains and evolving domains.

Based on Figure IV.10, minimum supports are selected for which the number of rules for Fuzzy-Apriori and Fuzzy-CSar-AFP tends to be similar and not excessively low. Thus, the chosen minimum supports are: 0.02 for evolving domains, 0.25 for static domains on the trained subject and 0.10 for static domains on the sedentary subject. Furthermore, rules are also filtered according to two minimum confidences (0.75 and 0.85).

In general, it is observed that for a minimum confidence of 0.75, the number of Fuzzy-Apriori and Fuzzy-CSar-AFP rules remains at similar levels throughout the experiment. The main differences between them are observed when 0.85 is used as the confidence threshold.

Since Fuzzy-Apriori is not an incremental algorithm the trend of the evolution on the number of different rules is not the same as for Fuzzy-CSar and Fuzzy-CSar-AFP, and there are noticeable differences among the four graphics. While the differences observed between Fuzzy-CSar-AFP and Fuzzy-CSar maintain the same pattern regardless of the subject, in the case of Fuzzy-Apriori there is considerable difference between them. When the domains are kept static, Fuzzy-Apriori is able to maintain a higher number of rules than Fuzzy-CSar-AFP for both confidence thresholds in the trained subject. However, in the case of the sedentary subject, when the confidence threshold rises to 0.85, the number of Fuzzy-Apriori rules collapses, exceeding Fuzzy-CSar-AFP only for a short interval.

We also appreciate differences between subjects in the case of evolving domains. For the trained subject and 0.75 as confidence threshold, the amount of filtered rules from Fuzzy-Apriori is always greater than the one from Fuzzy-CSar-AFP. Nonetheless, if the confidence threshold is raised up to 0.85, Fuzzy-Apriori only beats Fuzzy-CSar-AFP during a few short intervals. Meanwhile, the amount of filtered rules from Fuzzy-Apriori and Fuzzy-CSar-AFP are quite similar in the sedentary subject when the minimum confidence is 0.75. When increasing the minimum confidence, the number of rules falls significantly for Fuzzy-Apriori, remaining below both Fuzzy-CSar-AFP and Fuzzy-CSar throughout almost the entire experiment. In summary, it can be said that the number of rules from Fuzzy-CSar-AFP surpasses the number of rules from Fuzzy-CSar. However, the relationship between the number of different rules obtained by Fuzzy-CSar-AFP and Fuzzy-Apriori is not so clear even though in most cases the difference between confidence thresholds (0.75 and 0.85) is more significant in Fuzzy-Apriori. Anyway, the comparative analysis is expanded throughout the following section.

IV.3.4.4 MultiDimensional Scaling to analyze dispersion of association rules

In this section we use the proposal described in Section IV.3.2.4 to carry out a comparison in terms of dispersion between the sets of rules generated by Fuzzy-CSar-AFP, Fuzzy-CSar and Fuzzy-Apriori.

The rules included in the comparison are filtered according to support and confidence thresholds. In Section IV.3.2.4 we could ascertain how much impact the minimum support has on the number of rules generated by Fuzzy-Apriori.Therefore, to allow a more fair comparison in terms of dispersion between Fuzzy-CSar-AFP and Fuzzy-Apriori we decided to choose minimum support and confidence thresholds for which both algorithms obtain a very close number of rules. Thus, the results included in this section refer to rules filtered according to: (a) minimum support of 0.02 in experiments where evolving domains are used; (b) minimum support of 0.10 in experiments with static domains for the sedentary subject, or (c) 0.25 as minimum support in experiments with static domains for the trained subject. Minimum confidence was set to 0.75.

Through the process described in Section IV.3.2.4 we are able to represent each rule as a bidimensional point. Hence, we can visually display the set of rules generated by different algorithms in scatter plots as the ones shown in Figures IV.11 and IV.12. In each of these two figures, the rules generated by the algorithms are divided in different plots based on their consequent variable. The sets of rules used correspond to the ones at the end of the stream, i.e.,when 60 minutes of data have been processed (921600 samples). Tables IV.3 and IV.4 include the corresponding number of rules and δ_{MDS}.

Based on these figures and tables, we can both visually and quantitatively compare how dispersed the rules of each algorithm are distributed. Visually, we can observe that for both trained (Figure IV.11) and sedentary (Figure IV.12) subjects the rules obtained by Fuzzy-CSar-AFP (blue points) tend to be more distanced from each other and are quite spread, covering a wide area and presenting rules for every consequent variable. If we check Tables IV.3 and IV.4, δ_{MDS} values confirm that in most of the analyzed cases Fuzzy-CSar-AFP is the algorithm whose rules are distributed more widely, with a higher average distance between each other. It is worth highlighting those cases in which Fuzzy-Apriori obtains more rules but Fuzzy-CSar-AFP gets a higher δ_{MDS}, for instance, when O2 is the consequent variable. In these examples, Fuzzy-Apriori obtains more rules than Fuzzy-CSar-AFP but δ_{MDS} is clearly greater for Fuzzy-CSar-AFP, i.e., Fuzzy-CSar-AFP manages to discover with less association rules a more diverse and widespread knowledge. An extremely low number of rules is neither desirable, not even if δ_{MDS} is higher. The aim is to find a balance between the number of quality rules and how sparse they are.

De momento/ Por el momento, hemos puesto el foco únicamente en el instante final del flujo de datos, when all the samples have been processed. No obstante, al disponer de una medida que nos permite evaluar cuantitativamente la dispersión de las reglas, we can get a more global vision of the evolution of δ_{mds} throughout the full stream. In Figure IV.13 we represent δ_{mds} at 30 different points of the experiment. δ_{mds} value for each algorithm's rules is computed everytime two new minutes of data (30720 new samples)

Table IV.3: Number of rules ($\#R$) and dispersion (δ_{mds}) comparison for the TRA subject. The rules included in this table have overcome support and confidence thresholds ($conf \geq 0.75$ and different minimum supports). The results are shown for both static and evolving domains.

		Fp2		**C4**		**O2**		**O1**		**C3**		**Fp1**	
		δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$
Static	F-A priori	0.144	35	0.00	0	0.152	36	0.00	0	0.148	37	0.00	0
	FCSar	**0.161**	7	0.00	0	**0.156**	8	0.00	0	**0.162**	9	0.00	0
	FCSar-AFP	0.095	4	**0.132**	19	0.107	8	**0.124**	16	0.106	12	**0.134**	27
Evolving	F-A priori	0.180	64	0.184	75	0.182	71	**0.186**	62	0.183	58	0.175	62
	FCSar	0.163	12	0.150	10	0.150	9	0.079	2	0.170	12	0.153	6
	FCSar-AFP	**0.183**	62	**0.190**	60	**0.190**	53	0.174	54	**0.190**	56	**0.207**	38

Table IV.4: Number of rules ($\#R$) and dispersion (δ_{mds}) comparison for the SED subject. The rules included in this table have overcome support and confidence thresholds ($conf \geq 0.75$ and different minimum supports). The results are shown for both static and evolving domains.

		Fp2		**C4**		**O2**		**O1**		**C3**		**Fp1**	
		δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$	δ_{mds}	$\#R$
Static	F-Apriori	0.147	28	0.150	74	0.137	40	0.144	7	0.00	0	0.153	22
	FCSar	0.122	7	0.00	0	0.137	4	0.160	11	**0.179**	15	0.156	12
	FCSar-AFP	**0.159**	19	**0.151**	23	**0.159**	15	**0.187**	8	0.126	16	**0.162**	20
Evolving	F-Apriori	0.171	45	**0.187**	68	0.171	76	0.168	67	**0.194**	54	**0.193**	59
	FCSar	0.089	10	0.240	6	0.126	6	0.135	11	0.137	19	0.140	13
	FCSar-AFP	**0.179**	80	0.165	54	**0.203**	53	**0.203**	64	0.181	52	0.178	52

have been processed by the algorithms. The association rules are filtered by the same quality parameters as in the previous figures and tables ($conf \geq 0.75$ and $supp \geq 0.02$).

In the plots contained in Figure IV.13 we can appreciate how Fuzzy-CSar-AFP obtains the highest values of δ_{mds} for the main part of the stream in all cases. If we focus on the comparative between Fuzzy-CSar-AFP and Fuzzy-Apriori, Fuzzy-CSar-AFP equals or improves Fuzzy-Apriori in all the experiments (both subjects and both domains). The improvement of Fuzzy-CSar-AFP over Fuzzy-Apriori is more noticeable in the case of absolute domains. But even with evolving domains Fuzzy-CSar-AFP beats Fuzzy-Apriori in most of the occasions. Furthermore, it is important to recall that Fuzzy-Apriori is not really an incremental online algorithm but we apply it on different batches of data. If we would need to get a real time update of the state of the population of rules each time a new data is received, this would be possible with Fuzzy-CSar or Fuzzy-CSar-AFP but not with Fuzzy-Apriori. When comparing Fuzzy-CSar with Fuzzy-CSar-AFP, it is clear that Fuzzy-CSar-AFP achieves better results for three out of four experiments, and with a smaller difference also gets better results in the remaining one. Furthermore, if we also check Figure IV.10, we observe that the number of rules from Fuzzy-CSar is lower for 0.75 is the minimum confidence. So we can conclude that Fuzzy-CSar-AFP tends to extract a sufficiently high number of quality rules with a greater diversity degree than the ones obtained by other algorithms in most of the conducted experiments, i.e., Fuzzy-CSar-AFP generates rules that represent different knowledge rather than a lot of overlapping or redundant rules expressing the same information. As explained before, an algorithm capable of generating rules with a higher diversity level is preferable.

IV.3.4.5 Efficiency analysis

In order to check and compare the efficiency of the algorithms, two different types of experiments have been performed. Firstly, the computation time spent by each algorithm in processing every two new minutes of data (30 720 sampling) is registered. Since the minimum support value specified as input argument of Fuzzy-Apriori determines its number of frequent itemsets and, therefore, influences its execution time, we carry out several tests varying the support threshold. Fuzzy-CSar and Fuzzy-CSar-AFP do not need minimum support as input parameter. The results of these tests are shown in Figure IV.14. The experiments are performed in a Intel® Core™ i7-4790 3.60 GHz, RAM 16 GB DDR3 (1600 Mhz) computer.

In both figures, we can observe how the support threshold is a key factor in the execution time of Fuzzy-Apriori. Relating these results with Figure IV.9, as the number of rules obtained by Fuzzy-Apriori quickly decreases if the minimum support value increases a little, its execution time follows the same trend. Comparing the performances of Fuzzy-CSar-AFP and Fuzzy-CSar, the latter gets better efficiency since it maintains a significantly lower number of rules (see Figure IV.10). In other words, in exchange for getting many more quality rules, Fuzzy-CSar-AFP needs to evolve a wider pool of rules that increases processing time. In addition, we can observe how in the first subsets both Fuzzy-CSar and

Fuzzy-CSar-AFP register lower time values. This is due to the incremental and online character of these algorithms. Initially, the population of association rules maintained by these algorithms is empty, then the population starts to grow up and, finally, it stabilizes, as well as the execution time of the algorithm.

Another key factor in the time efficiency of these algorithms is the number of attributes forming the input data. In order to test the influence of this factor on the three algorithms, we test them on the full original EEG dataset (before feature selection) including its 64 attributes. The average execution times per sample registered along with those recorded using the 6-attribute dataset are shown in Table IV.5. This table shows the average of three runs. In the table we can appreciate how Fuzzy-CSar-AFP scales much better than Fuzzy-Apriori. With the 6-attribute dataset, both Fuzzy-CSar-AFP and Fuzzy-Apriori can manage a sampling frequency of 1000 Hz. However, with the 64-attribute dataset, Fuzzy-CSar-AFP can manage 200 Hz while Fuzzy-Apriori needs about a second and a half to process each sample. That is, for a sampling frequency of 200 Hz, Fuzzy-Apriori may spend more than 44 hours to process the data recorded during 10 minutes of experiment, which is completely unfeasible in a real-world data stream problem. Fuzzy-CSar-AFP, on the contrary, is able to process the data on-the-fly.

In Table IV.5 we can also discover another interesting fact: the time registered by Fuzzy-CSar is lower than the time registered by Fuzzy-CSar-AFP for the 6-attribute dataset but higher in the case of the 64-attribute dataset. This is due to he number of rules generated. Fuzzy-CSar obtains a lower amount of different rules than Fuzzy-CSar-AFP for the 6-attribute dataset (as shown above) but gets more different rules than Fuzzy-CSar-AFP for the 64-attribute dataset (1919 vs. 1547). Nevertheless, these higher number of rules does not imply more quality. Indeed, Fuzzy-CSar-AFP obtains more high-quality rules (with minimum confidence of 0.85) than Fuzzy-CSar. The 64-attribute dataset is a very sparse problem so Fuzzy-CSar does not enhance quality properly, thus generating a high number of different rules but with poorer quality.

Table IV.5: Average execution time (in seconds) per sample of Fuzzy-Apriori, Fuzzy-CSar and Fuzzy-CSar-AFP applied on 6-electrodes and 64-electrodes dataset.

	6-attribute dataset		**64-attribute dataset**	
	TRA	**SED**	**TRA**	**SED**
Fuzzy-Apriori	0.000022	0.000018	1.337197	1.265868
Fuzzy-CSar	0.000203	0.000166	0.007194	0.007203
Fuzzy-CSar-AFP	0.001092	0.001057	0.004616	0.005056

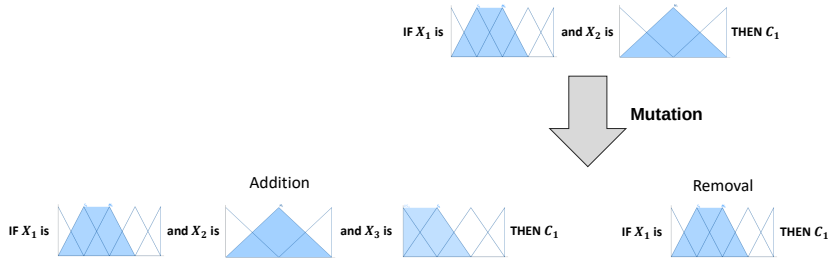
IV.3.5 Interpretation of obtained results in psychophysiology

Figures IV.15 and IV.16 show the association streams for each subject: trained and sedentary. In both cases, the figures revealed a stable increase in the number of association

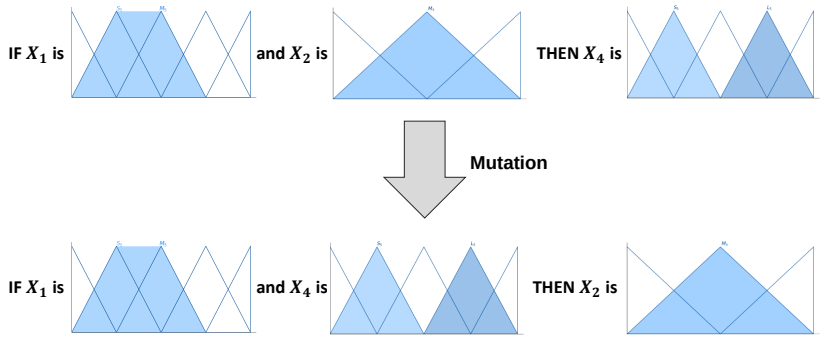
rules throughout the experimental session. This is an expected behavior for biological signals that fluctuate in response to variations in common underlying physiological sources. The result confirms the sensitivity of the method when detecting natural associations between signals from electrodes at distinct scalp locations. In addition, the method shows robust to mild physiological changes induced by mental fatigue during a behavioral task with low cognitive demands. This robustness of the algorithm to the factor of time and to modest cognitive effort is important for further experimentation with more demanding conditions marked by pronounced physiological changes (e.g., sleep vs. awake and distinct sleep stages, tasks with significant cognitive load, etc.)

Although the algorithm proved robust to changes induced by the length and the particular behavioral demands of the task, it also proved capable of tracking the evolution of specific association rules throughout the experimental session. Differences in the dynamical pattern of concrete association rules between subjects seem like a promising methodological tool for detecting minute which would be impossible to assess with current averaging or pair-wise correlation techniques.

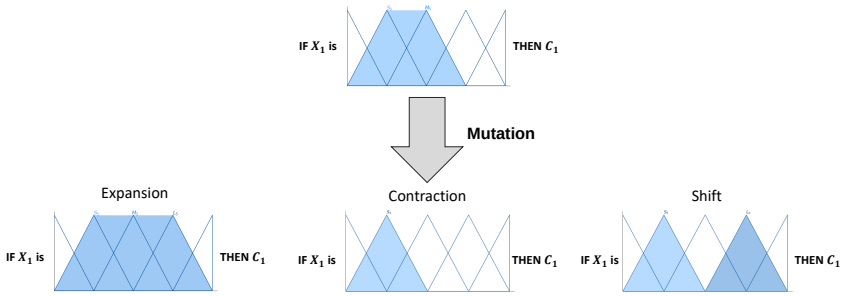
Figure IV.17 shows the dependency wheels of high-quality fuzzy association rules ($supp \geq 0.05$ and $conf \geq 0.85$) obtained for both TRA and SED subjects at the end of the experiment (921 600 samples processed, i.e., after 60 minutes). At this level of performance, we can observe differences between the fuzzy association rules in these two cases. We can perform a zone analysis of these two graphics and highlight certain contrasts between them. Looking at the frontal electrodes (Fp1 and Fp2), in the case of TRA, Fp1 is the frontal electrode with greater presence as consequent (dotted sectors) while Fp2 appears mainly as antecedent (dashed sectors). This behavior is completely reversed in SED. On the one hand, Fp2 has a much larger total occurrence in SED and most of it as consequent. On the other hand, Fp1 loses relevance, particularly, its consequent sector. In the central electrodes (C3 and C4), there are not so significant differences between subjects, even though the relevance of C3 as consequent (dotted sectors) is clearly bigger in TRA. Finally, looking at the last electrodes (O1 and O2) we appreciate again significant variances between subjects, specially in O1 that has a pretty important consequent sector in TRA while in SED its consequent sector hardly exists. Instead of focusing on each individual electrode, we also can pay attention to the relationships and dependencies established between them. Many of these associations vary depending on the subject while others seem steady and more independent to subject change such as those built between Fp1 and Fp2, or Fp2 and C4.



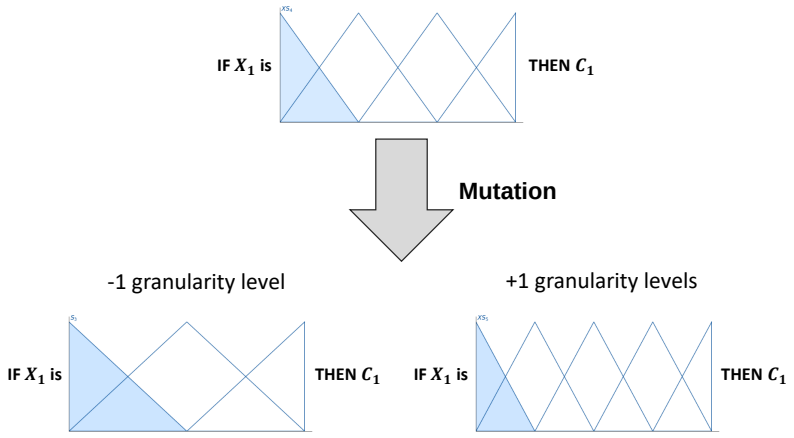
(a) Mutation of the antecedent variables



(b) Mutation of the consequent variable



(c) Mutation of linguistic terms



(d) Mutation of granularity

Figure IV.5: Graphical examples of the four different types of mutations that rules can undergo.

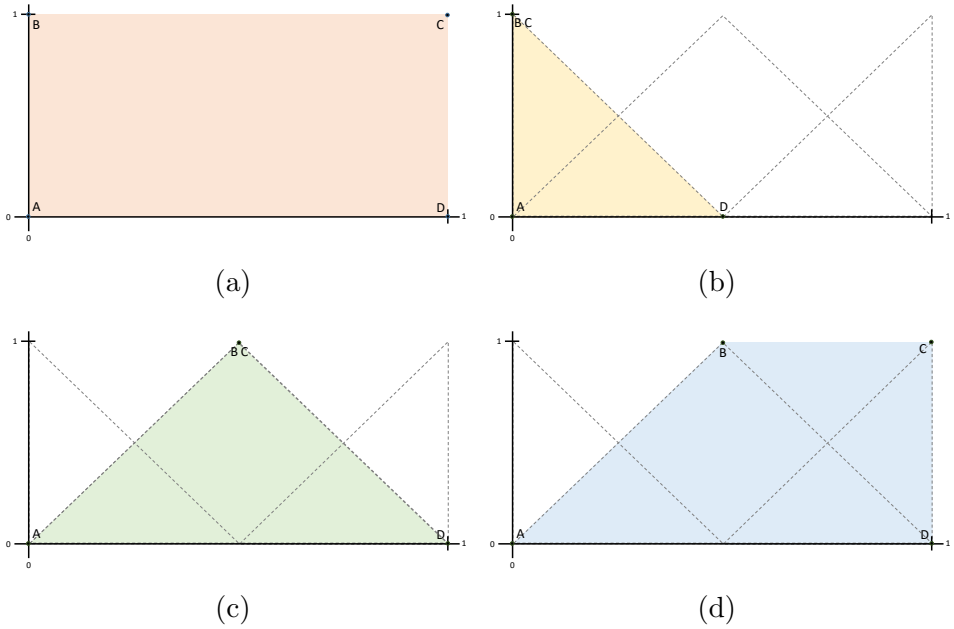


Figure IV.6: Location of the four vertices (noted as A,B,C,D) that define the trapezoidal representation of a fuzzy set in different scenarios: (a) the variable is not used, (b) the fuzzy set is composed by the label S_3, (c) the fuzzy is composed by the label M_3, and (d) the fuzzy set is composed by the disjunction of labels M_3 and L_3.

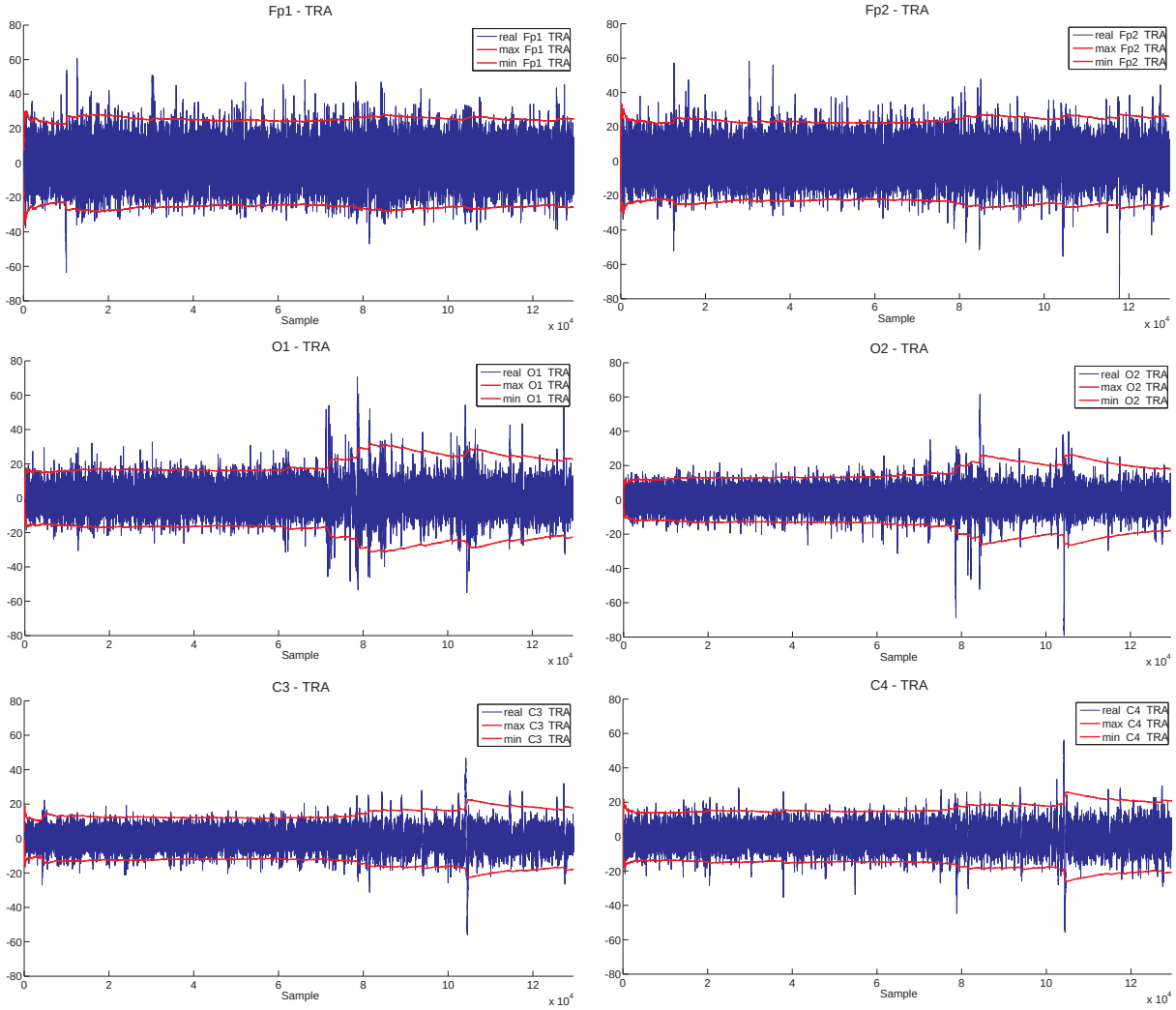


Figure IV.7: Evolution of the real value (blue), algorithm maximum value (red) and algorithm minimum value (red) for each input attribute (Fp1, Fp2, O1, O2, C3 and C4) as data are processed. The values shown in this figure correspond to the subject used as representative of the group of trained (TRA) subjects. Because of the high sampling rate and to reduce the weight of the images, only 1 in 20 data is represented in these plots.

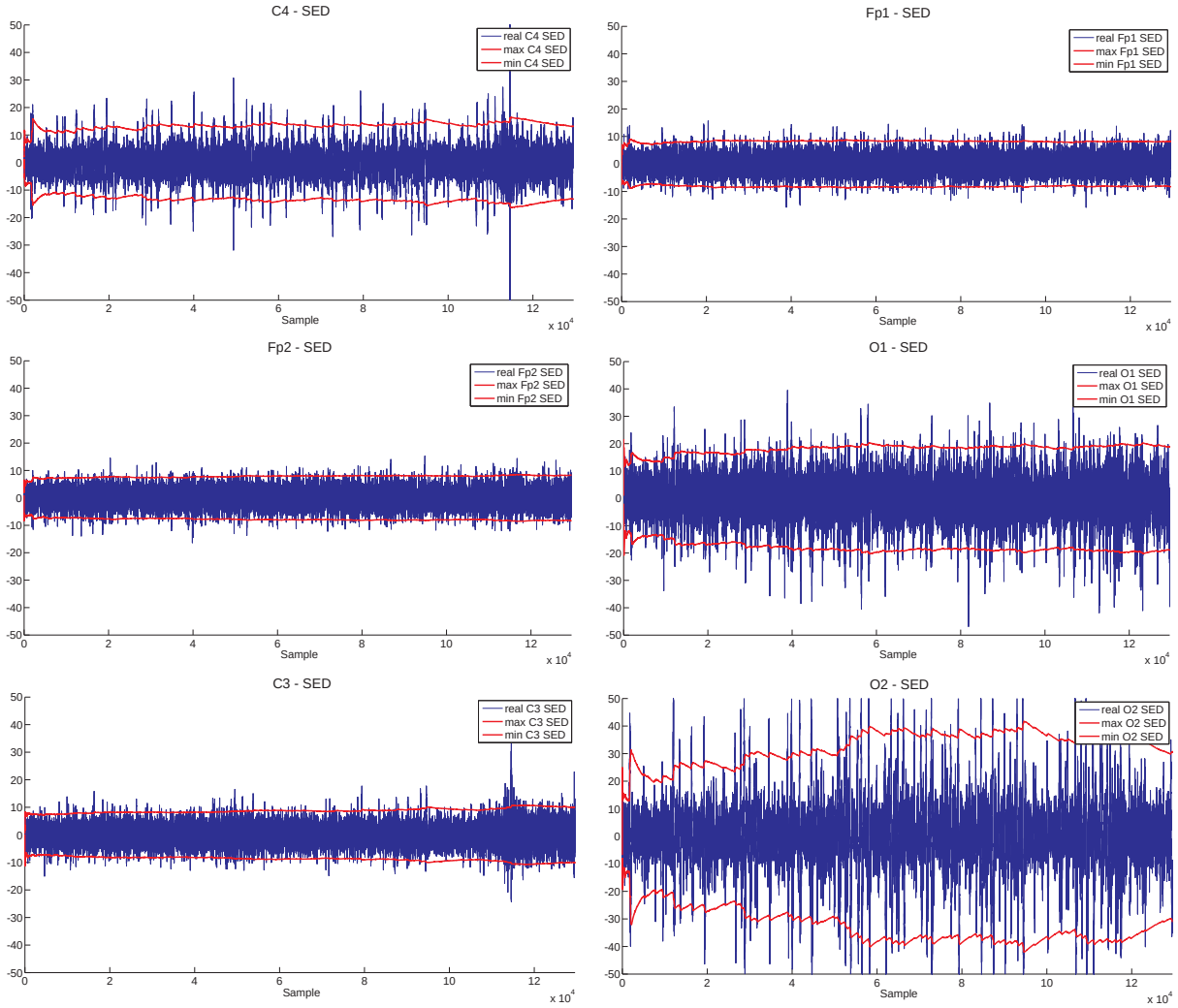
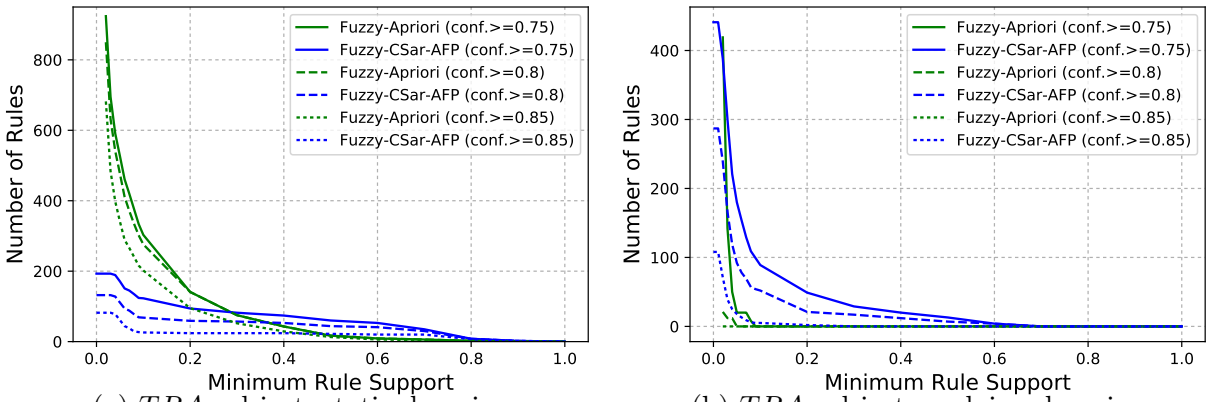


Figure IV.8: Evolution of the real value (blue), algorithm maximum value (red) and algorithm minimum value (red) for each input attribute (Fp1, Fp2, O1, O2, C3 and C4) as data are processed. The values shown in this figure correspond to the subject used as representative of the group of sedentary (SED) subjects. Because of the high sampling rate and to reduce the weight of the images, only 1 in 20 data is represented in these plots.



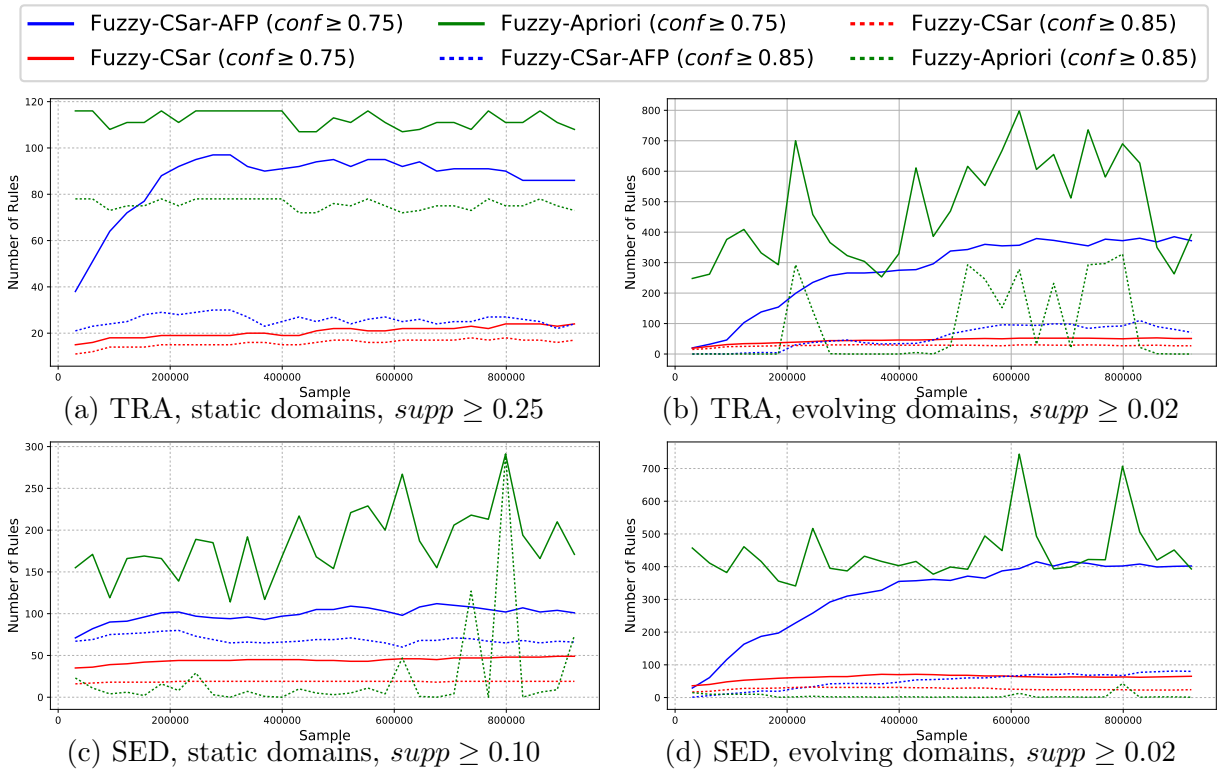
(a) TRA subject, static domains

(b) TRA subject, evolving domains

(c) SED subject, static domains

(d) SED subject, evolving domains

Figure IV.9: Amount of rules from Fuzzy-CSar-AFP (blue) and Fuzzy-Apriori (green) which get over three different confidence thresholds ($conf \geq 0.75$, $conf \geq 0.8$ and $conf \geq 0.85$) and minimum support thresholds from 0.0 to 1.0. The results are shown for both the experiments with absolute domains (left) and evolving attribute's domains (right).



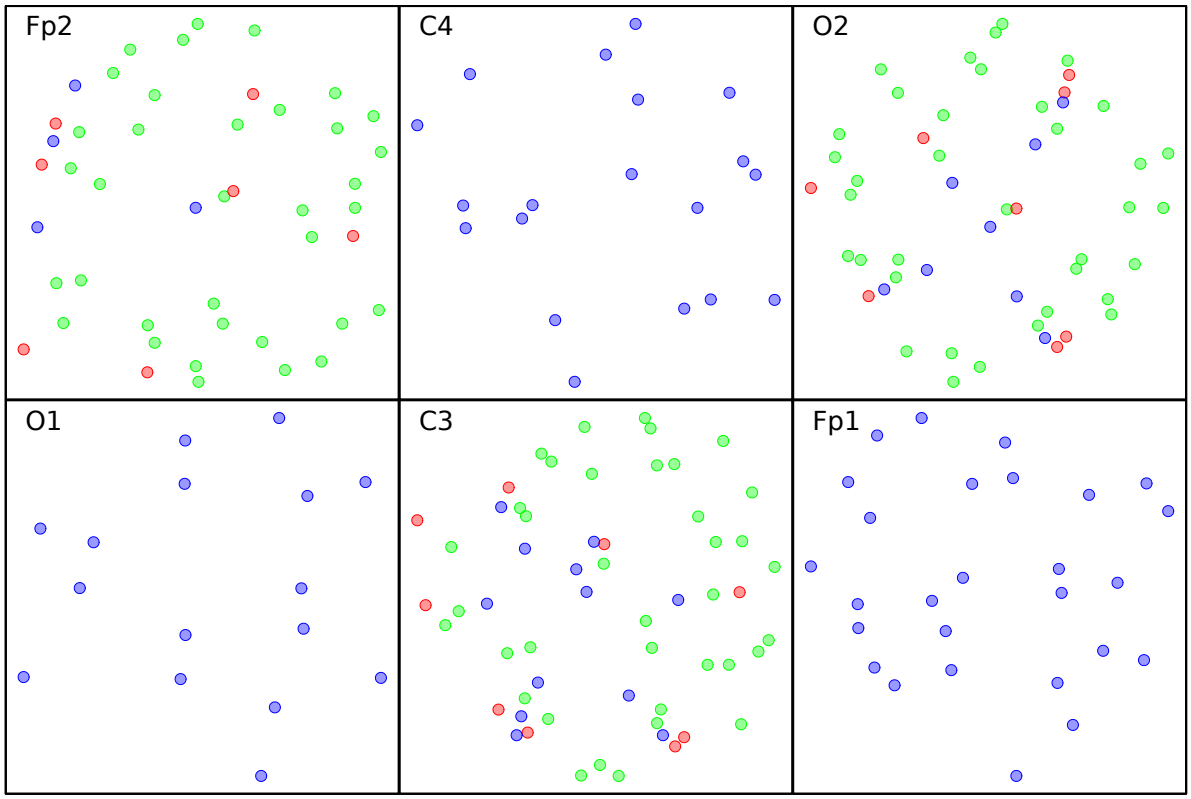
(a) TRA, static domains, $supp \geq 0.25$

(b) TRA, evolving domains, $supp \geq 0.02$

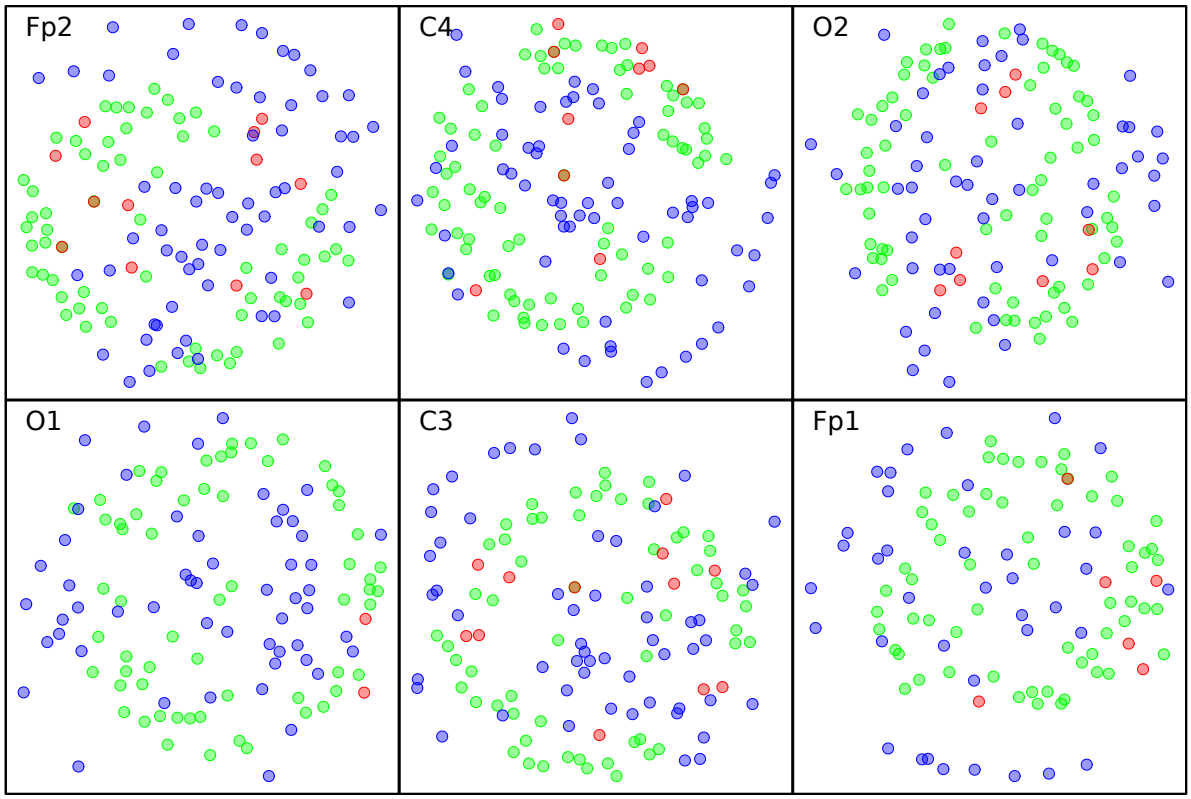
(c) SED, static domains, $supp \geq 0.10$

(d) SED, evolving domains, $supp \geq 0.02$

Figure IV.10: Evolution on the amount of different rules along the stream for both trained (TRA) and sedentary (SED) subject. The results are shown for experiments with static domains (left) and evolving domains (right).

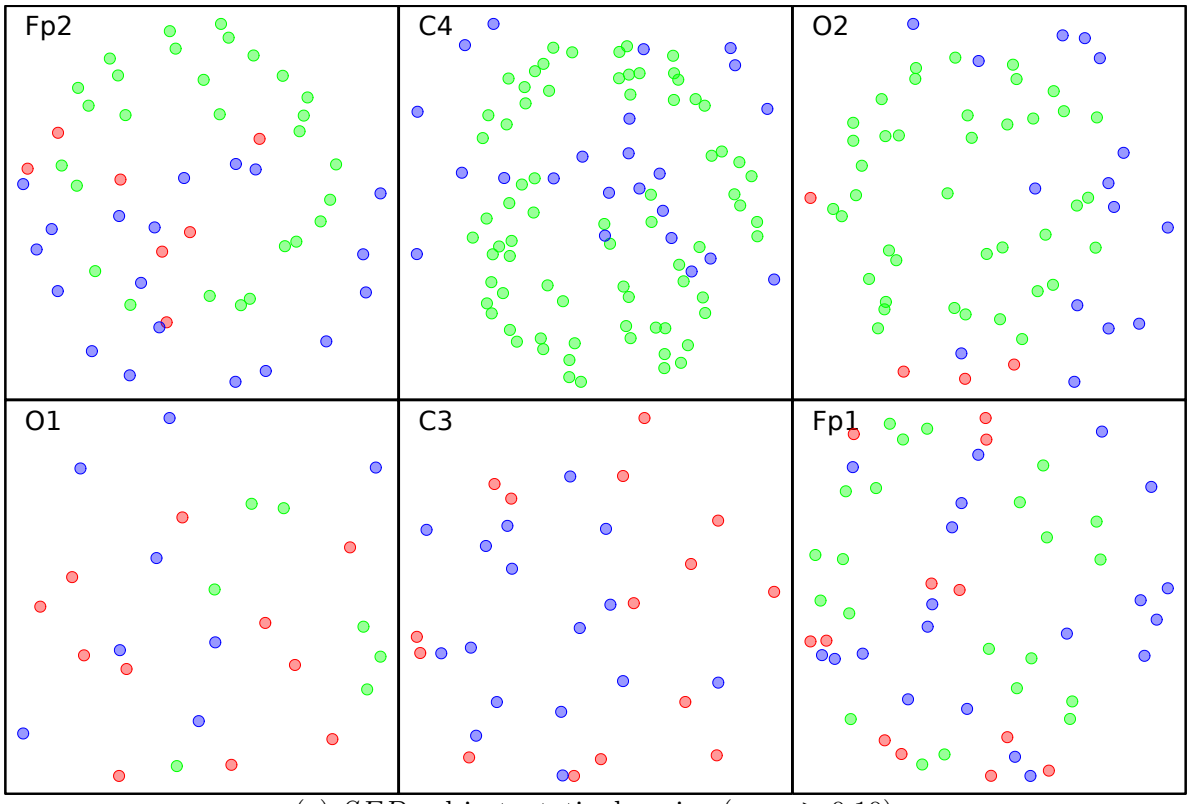


(a) TRA subject, static domains ($supp \geq 0.25$)

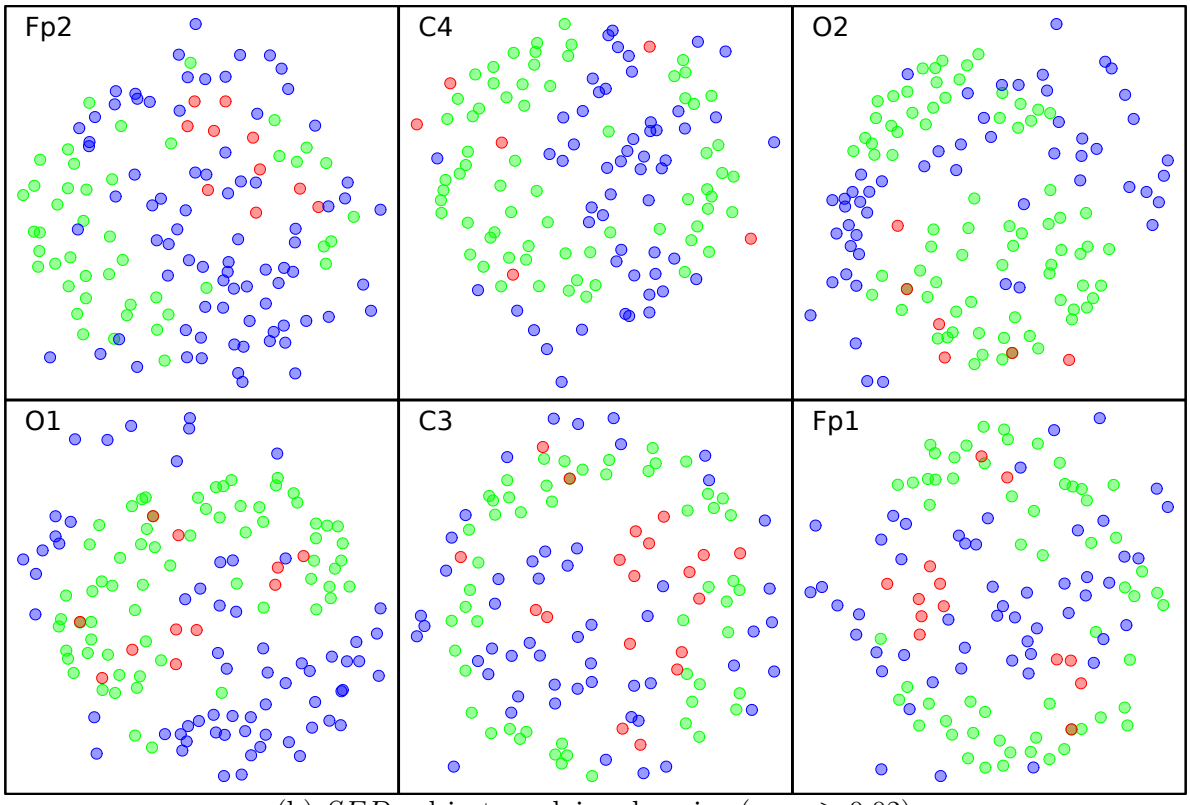


(b) TRA subject, evolving domains ($supp \geq 0.02$)

Figure IV.11: Two-dimensional representations of how the rules are distributed in Fuzzy-CSar-AFP (blue points), Fuzzy-CSar (red) and Fuzzy-Apriori (green) for trained subject (TRA) data. The results are shown for both the experiments with absolute domains (up) and evolving domains (down).

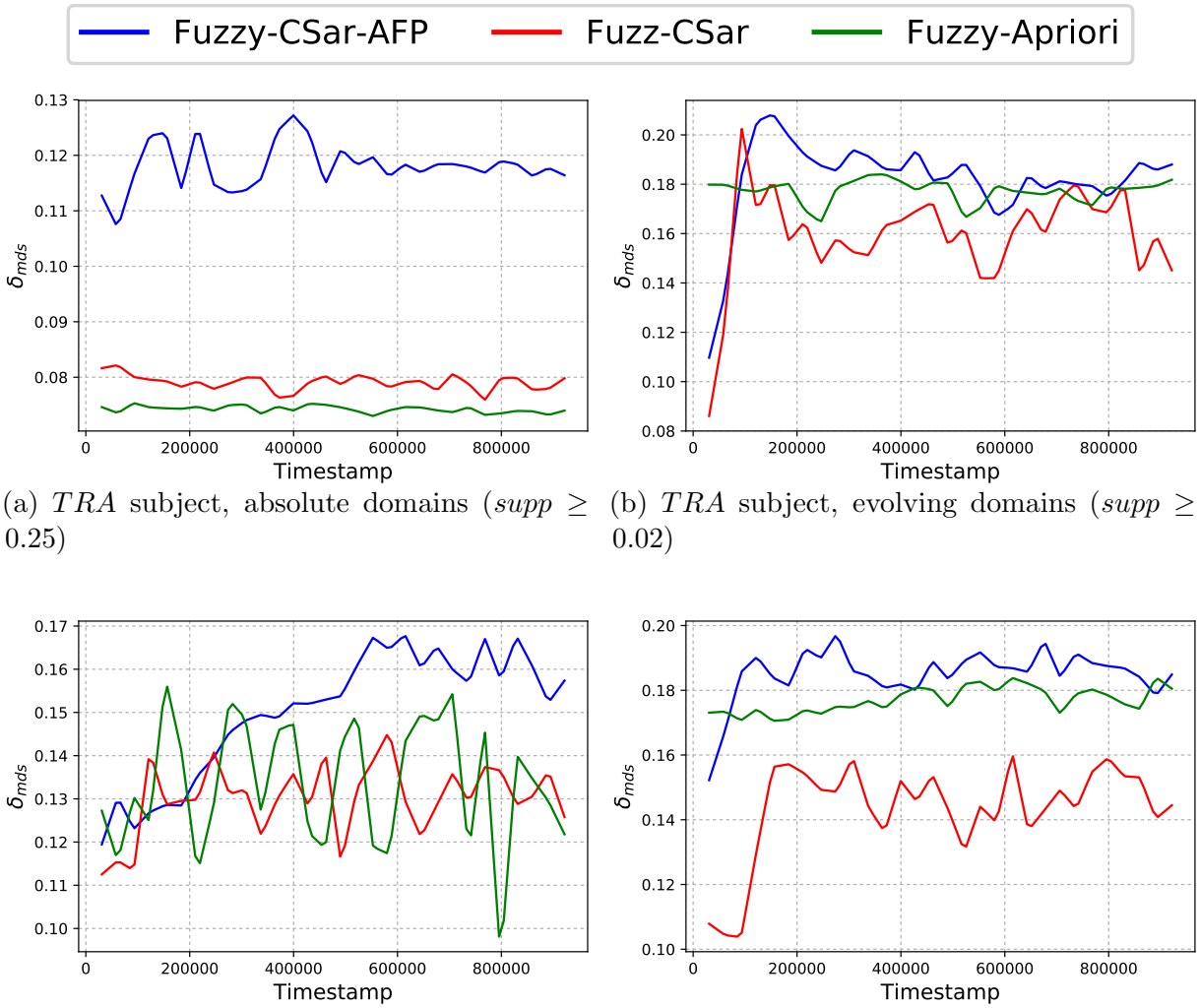


(a) *SED* subject, static domains ($supp \geq 0.10$)



(b) *SED* subject, evolving domains ($supp \geq 0.02$)

Figure IV.12: Two-dimensional representations of how the rules are distributed in Fuzzy-CSar-AFP (blue points), Fuzzy-CSar (red) and Fuzzy-Apriori (green) for sedentary subject (*SED*) data. The results are shown for both the experiments with absolute domains (up) and evolving domains (down).



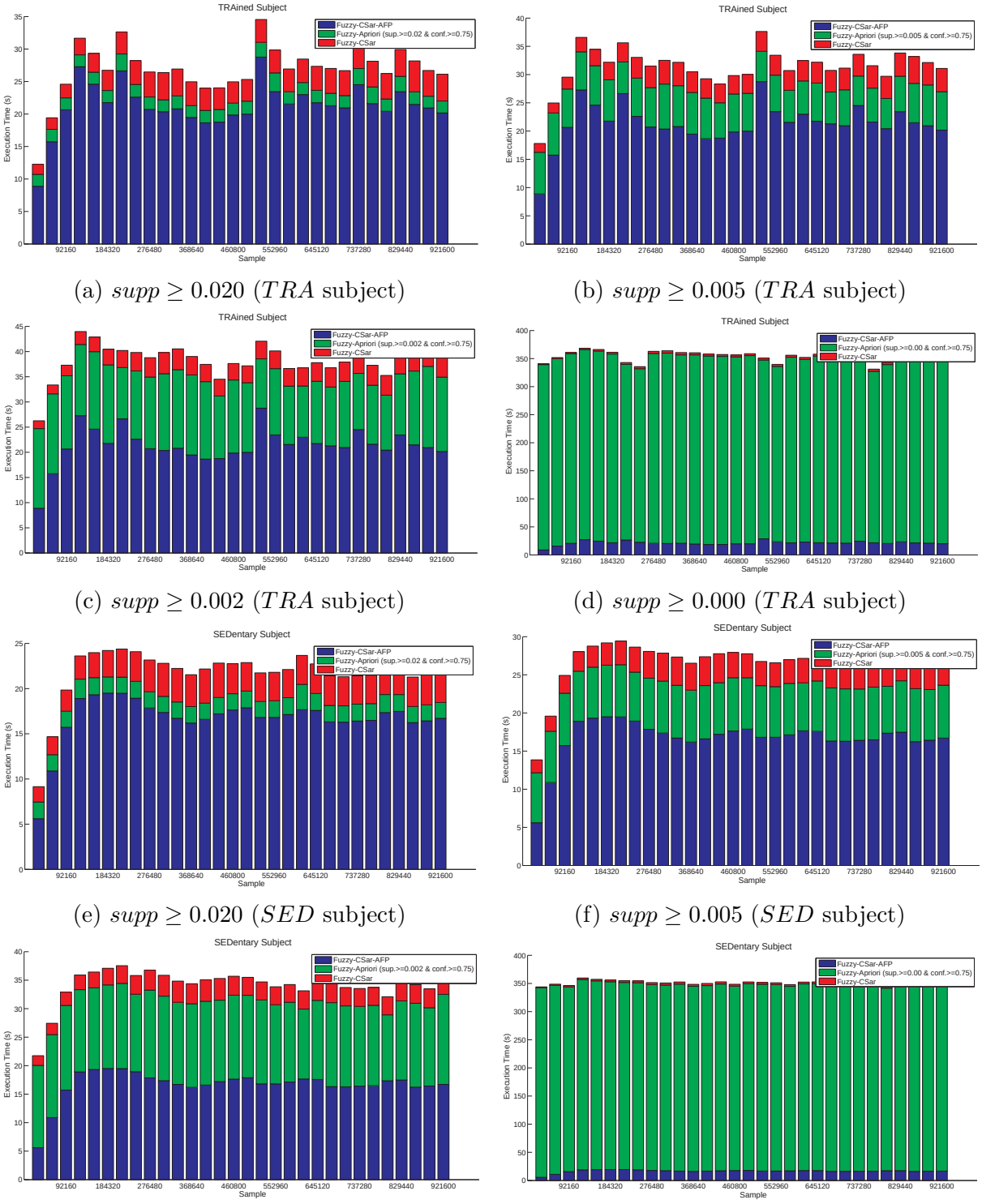
(a) *TRA* subject, absolute domains ($supp \geq 0.25$)

(b) *TRA* subject, evolving domains ($supp \geq 0.02$)

(c) *SED* subject, absolute domains ($supp \geq 0.10$)

(d) *SED* subject, evolving domains ($supp \geq 0.02$)

Figure IV.13: Evolution of δ_{mds} for the rules (filtered according to $conf \geq 0.75$ and different support thresholds) obtained by Fuzzy-CSar-AFP (blue), Fuzzy-CSar (red) and Fuzzy-Apriori (green) for trained *TRA* (up) and sedentary *SED* (down) subjects. The results are shown for both absolute domains (left) and evolving domains (right).



(a) $supp \geq 0.020$ (TRA subject)

(b) $supp \geq 0.005$ (TRA subject)

(c) $supp \geq 0.002$ (TRA subject)

(d) $supp \geq 0.000$ (TRA subject)

(e) $supp \geq 0.020$ (SED subject)

(f) $supp \geq 0.005$ (SED subject)

(g) $supp \geq 0.002$ (SED subject)

(h) $supp \geq 0.000$ (SED subject)

Figure IV.14: Execution times spent by Fuzzy-CSar-AFP (blue), Fuzzy-CSar (red) and Fuzzy-Apriori (green) to process each new subset (30720 sampling, 2 minutes of data recording) varying the minimum support value used by Fuzzy-Apriori. A minimum confidence value of 0.75 is used by Fuzzy-Apriori.

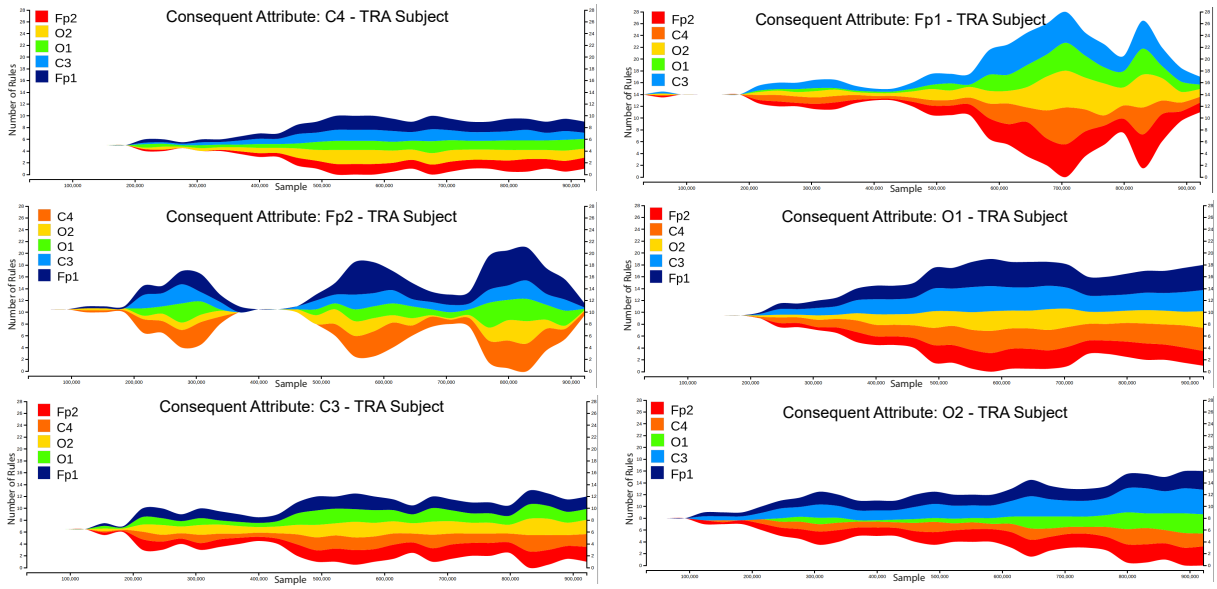


Figure IV.15: Streamgraphs for TRA (trained subject) $supp \geq 0.02$ and $conf \geq 0.8$

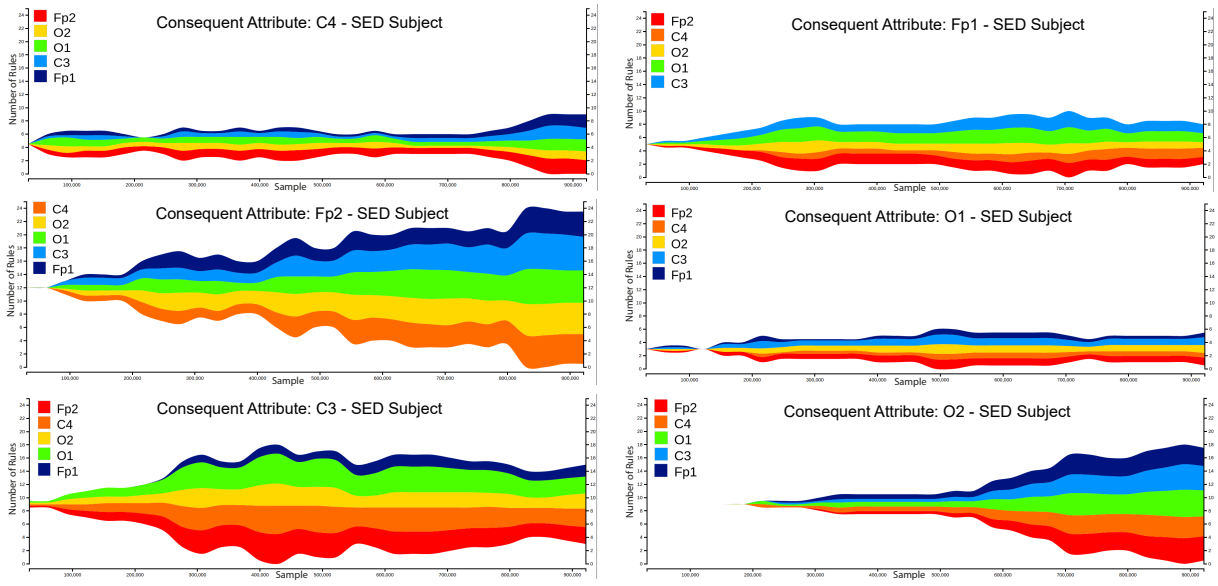
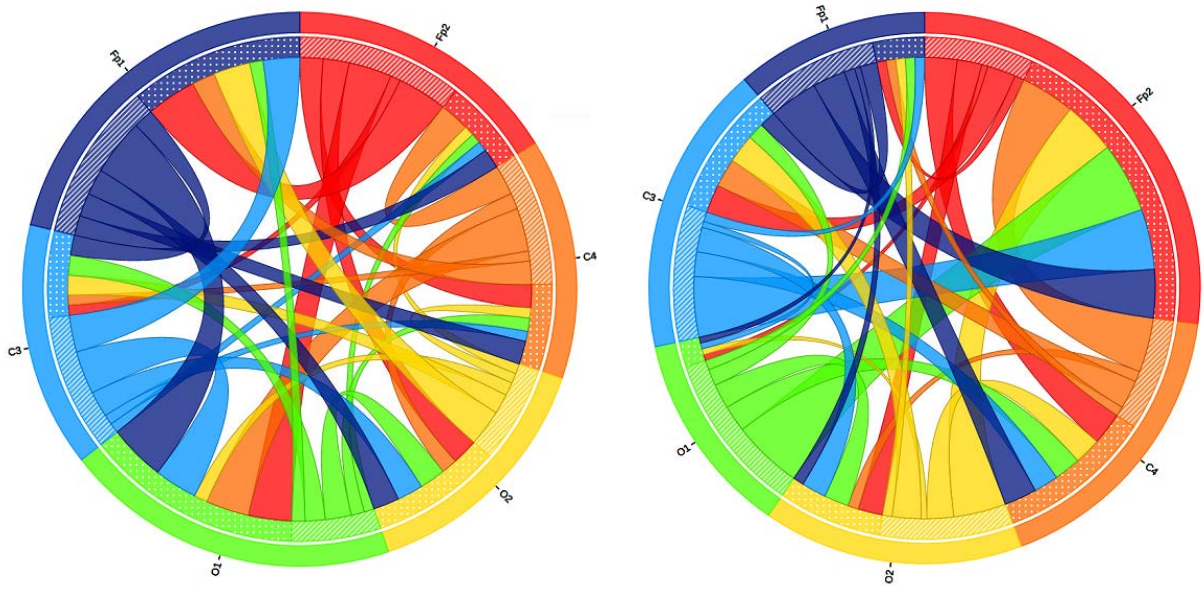


Figure IV.16: Streamgraphs for SED (sedentary subject) $supp \geq 0.02$ and $conf \geq 0.8$.



(a) *TRA* Subject (b) *SED* Subject

Figure IV.17: Dependency wheels for (a) *TRA* and (b) *SED* subjects at the end of the sampling with $supp \geq 0.05$ and $conf \geq 0.85$.

Chapter V

PAST: Learning rules in data stream semi-supervised learning

V.1 Introduction

A supervised approach is able to exploit the labels associated with the data but is not able to capitalize on data with no associated class label. An unsupervised learning approach is prepared to obtain knowledge from data without any associated labels, but does not contemplate the possibility of exploiting labeling information in case it exists. A semi-supervised learning approach allows to obtain knowledge both from labels received associated to some examples and from those examples that do not have any associated label.

Despite the high utility of unsupervised learning techniques when the main interest does not lie in performing a predictive task, in data streams, as in offline environments, there are problems where a model able to predict the values of a target variable is needed. A very important part of the research effort in data stream classification have been focused on supervised methods, which require a massive amount of labeled data and, consequently, are dependent on timely available labels. Most of these studies assume that data streams arrive completely labeled and that such labels can be utilized at hand. Nonetheless, in many applications, obtaining labeled data is extremely complicated or costly, whereas unlabeled data are easily available (Matuszyk and Spiliopoulou, 2015; Noorbehbahani et al., 2017; Tang et al., 2017; Dal et al., 2018; Iosifidis and Ntoutsi, 2019). The very essence of data streams favors the existence of environments where it is difficult (or impossible) to have great amounts of labels. The potentially infinite nature of the stream, coupled with the high generation speed, makes it virtually impossible in many real-world applications to label data completely and on time.

Hence, it is likely that only a small fraction of data can be labeled in many real streaming environments. Since supervised models can only be trained with labeled examples, just a limited fraction of the data stream could be used for training and updating

the classification models, leading to poorly trained classifiers. Probably, a more realistic approach would be to proceed on the basis that labels are only going to be available for a certain percentage of the data. The data stream classification problem is further complicated by this label scarcity and the fact that the solution must rely partially on unlabeled data.

Semi-supervised learning techniques rely on the fact that the massive amount of unlabeled examples that form the data stream can also be seen as a source of information about the underlying data distribution and, hence, they can be used, in combination with labeled data, to improve the accuracy of the model. Semi-supervised learning capitalizes a small amount of labeled instances and uses a great amount of unlabeled ones to train the model.

In this chapter, a fully online semi-supervised approach for classification in data streams is presented. The proposal, hereafter *PAST* (PArtially labeled data STream mining), is based on the use of a population of fuzzy rules to extract knowledge from both labeled and unlabeled data. The algorithm does not require any initial set of labeled samples to initialize the system nor assumes the labeled samples will arrive at a specific frequency. The novelty of the work is twofold: (1) a new purely online semi-supervised algorithm for data stream classification is presented, and (2) the potential of a rule-based approach for semi-supervised classification in data streams is explored. Rule-based approaches for semi-supervised classification in data streams have been hardly studied.

This chapter is organized as follows. In Section V.2, we describe our semi-supervised proposal to tackle the issue of the lack of labeled data in data stream classification problems. Section V.3 present the experiments and their results.

V.2 PAST: PArtially labeled data STream mining

PAST is a semi-supervised learning system designed for dealing with label scarcity in data stream classification problems. PAST is an adaptation of the supervised approach CLAST and, therefore, it works in two different modes: *exploration* or training, and *exploitation* or testing. In the exploration or training mode, the main objective of the algorithm remains to build a population of rules, as general as possible, which accurately represent the class division that underlies the data stream. Each sample processed in exploration mode, labeled or not, is used to update this rule population. Figure V.1 schematically illustrates the learning process of PAST during an exploratory iteration. In the exploitation mode, the algorithm gathers the rules of the population to work together to try to successfully predict to which class the example belongs. Hence, PAST maintains the cooperative design employed by CLAST, a design that allows to incrementally evolve the model (population) over time in a very natural and efficient way.

PAST meets all the main requirements put forward by Bifet et al. (2009b) for data stream classification algorithms. Thus, PAST accepts examples in the order they

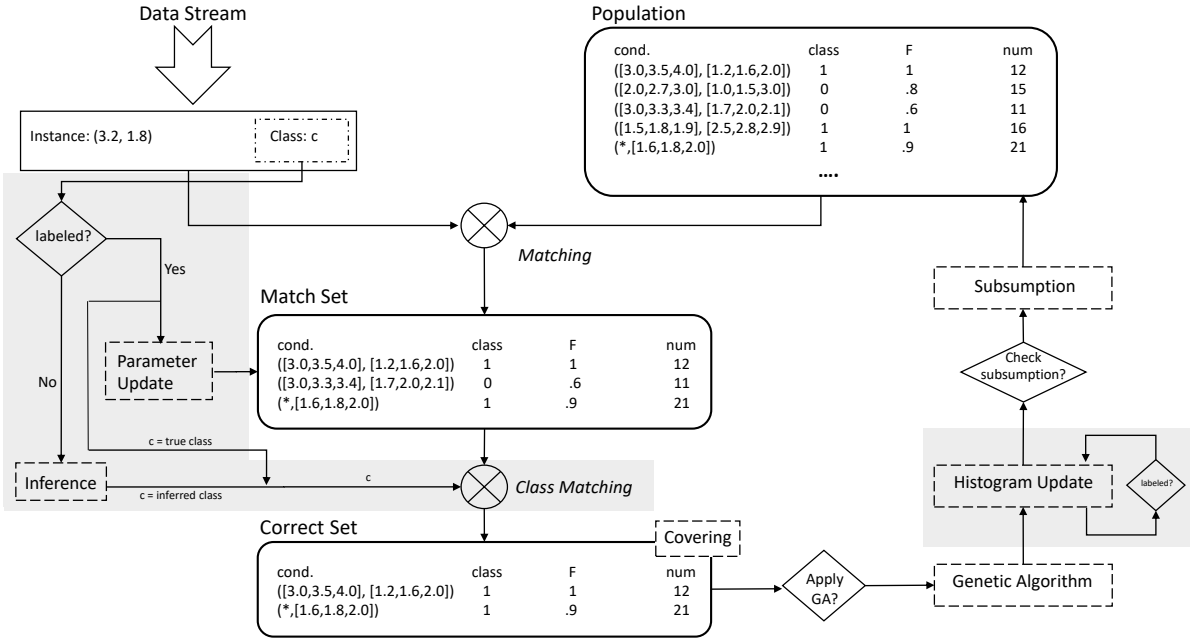


Figure V.1: Schematic illustration of the learning interaction of PAST. The main areas where the differences between PAST and CLAST are concentrated are highlighted with a gray background.

are received; each example is visited only once; memory usage is controlled; learning complexity is linear to the number of examples, and the induction model can be applied at any time between training examples. In fact, unlike CLAST where the two modes of the algorithm (exploration and exploitation) work completely independently; in the case of PAST, the inference process is part of the learning iteration if the input example is received unlabeled.

To the qualities claimed by Bifet et al., we could add the capacity of the system to extract knowledge from unlabeled examples, allowing it to adapt to environments where labeled examples are scarce. A very common scenario in real-world data streams given the high volume, speed and infinite character of such data sources.

V.2.1 Knowledge Representation

The differences between PAST and CLAST are focused on the exploration process, since the PAST learning scheme must consider the possibility of not knowing the class of the example received. However, the way of representing knowledge is identical in PAST and CLAST, and the inference process carried out to predict the class of an example is also shared between both approaches (regardless of whether PAST uses that mechanism in scenarios that are not contemplated in CLAST).

Thus, each classifier included in the population maintained by PAST contains a

fuzzy rule of type II (Cordón et al., 2001) with the form:

$$R_k : \textbf{IF } X_1 \text{ is } \widehat{A_1^k} \text{ and } \ldots \text{ and } X_n \text{ is } \widehat{A_n^k} \textbf{ THEN } c^k \textbf{ WITH } w^k \tag{V.1}$$

where the condition is composed by the conjunction of a series of fuzzy sets $\widehat{A_i^k}$, each of them associated to an input variable i; c^k is the advocated class, and $w^k \in [0, 1]$ is the strength with which the rule advocates class c^k. Along with the fuzzy rule, each classifier also contains a set of parameters.

V.2.2 Exploration Mode

As can be seen in Figure V.1, much of the learning iteration of PAST matches that of CLAST. However, in the case of PAST, depending on whether the example received is labeled or not the course of the learning iteration varies.

When an input example e is received, the first step is to build the *matchset* [M] with all those rules that match e with a degree greater than zero. This first step is identical regardless of whether e is labeled or not since the class of e is irrelevant to the construction of [M]. However, the second step consists on defining the *correctset* [C], which includes all those rules of [M] that advocate the class of e. If e is an unlabeled example, the system does not know the real class to which it belongs. In this case, a class is inferred for e based on the rules in [P], and this predicted class will be used as the class of the example during the rest of the learning iteration. This is where the PAST inference process becomes part of the learning process. Therefore, in the case of a labeled example, [C] groups all the rules in [M] that advocate the class of the label while, in the case of an unlabeled example, [C] is formed by those rules in [M] that advocate the class inferred for the example. If none of the rules in [C] matches e with a sufficient degree for all the input variables, the covering operator is launched. The covering operator creates a new classifier that matches e with maximum degree. This new classifier is then added to [C], [M] and [P]. In Section III.2.2.2 we detailed how the covering operator of CLAST generalizes the condition of the new classifier from e, and how the class of e affects such generalization process. PAST covering operator follows a generalization process analogous to that of CLAST, with the difference that if e is unlabeled, the class used as reference in the generalization process will be the one inferred by the system itself. Afterwards, if e is labeled, the fitness of those classifiers in [M] is updated taking into account the new example received e. Next, the rule discovery component, which enables the system to discover new promising rules via a genetic mechanism, may be applied to the classifiers included in [C]. In order to control runtime and reduce overfitting risk, this component is only launched when the average time since its last application upon the classifiers in [C] surpasses a certain threshold. Finally, the last two steps of the learning iteration are: histogram update and subsumption check.

The method for updating the histograms varies depending on whether e was received labeled or not. In the first case, only the histograms of the class c^e to which e belongs are

updated. The histogram of c^e for each variable is updated in the following way:

$$H^{t+1}_{ic^e\phi_i} = H^t_{ic^e\phi_i} + 1, \ \forall i \in \{1, ..., n\} \tag{V.2}$$

where ϕ_i is the bin where e_i lies.

However, if e was received as an unlabeled example, the histograms of all classes are updated. The histogram of each class for each variable is updated as

$$H^{t+1}_{ic_j\phi_i} = H^t_{ic_j\phi_i} + \frac{1}{m}, \quad \forall i \in \{1, ..., n\} \ \forall j \in \{1, ..., m\} \tag{V.3}$$

Histograms only count an example as belonging to a class when a label indicating so has been received. Therefore, the unlabeled examples influence the weight that the different bins have on operators such as covering or mutation while, at the same time, we avoid introducing inaccurate information in the histograms. This same principle of avoiding entering inaccurate information governs the decision of only updating the parameters of the classifiers in [M] when the input example is labeled. Subsumption check in [P], as rule discovery component, is only conducted when the time since its last application exceeds a user-defined threshold (θ_{sub}).

V.2.3 Exploitation mode

The system is prepared to conduct the class inference process at any time, that is, any example of the data stream can be processed by the system either in exploration mode or in exploitation mode. As shown in Figure V.1 and detailed in Section V.2.2, if we are facing an unlabeled example, it is necessary to predict a class for such example before creating the correct set. Of course, this same inference process can also be conducted in an isolated way (not as part of a training iteration) when the example in question is considered a test example.

As for the internal functioning of the inference mechanism, the one used by CLAST (Section III.2.3) is maintained. No changes need to be made to adapt it to a semi-supervised scenario. Hence, those rules in [P] that meet the following two conditions cooperate to infer the class of the input example: (1) the membership degree for each of the variables in the condition of the rule is greater than θ_μ, and (2) the fitness of the rule can be said to be greater than zero with a high level of confidence ($F^k - \varepsilon^k > 0$). Each of these rules emits a weighted vote v^k_c for the class c it advocates, and the most voted class is the one predicted for the example. The weighted vote v^k_c can be formulated as:

$$v^k_c = (F^k - \varepsilon^k) \cdot \mu_{A^k}(e) \tag{V.4}$$

Figure V.2 illustrates how the rule population covers the feature space of the *banana* dataset (KEEL Repository (Alcalá-Fdez et al., 2011)). In the pseudo-heatmaps included in this figure, the area covered by each rule is colored based on which class the rules

advocated once the whole *banana* dataset had been processed. Therefore, each example is covered by the colored areas of those rules that match it with a degree greater than zero. This representation gives us an idea of how well the rule population built by the algorithm is reflecting the underlying class distribution of the dataset. In Figure V.2, the rule populations generated by PAST and CLAST in three scenarios with low percentages of labeled data (5, 10 and 25%) are visually represented. On the basis of this representation, it seems that PAST is managing to develop populations that are better suited to the actual class distribution when there is a low number of labels available.

V.3 Comparison of PAST to several machine learning techniques

In this section, we analyze the behavior of PAST as compared to several other approaches based on different types of machine learning techniques. We compare PAST with two main sets of classifiers: data stream classifiers and static classifiers. In the first case, we study the performance of PAST and other data stream approaches that also follow an incremental online learning process. CLAST is included in such set of learners, so we can test if PAST is able to improve its predictions thanks to the knowledge extracted from unlabeled examples. In the second case, we assess if PAST is competitive with several offline machine learning techniques, despite the constraints derived from an incremental learning strategy. Finally, we use four different real-world data streams to analyze the performance of PAST under actual data stream conditions.

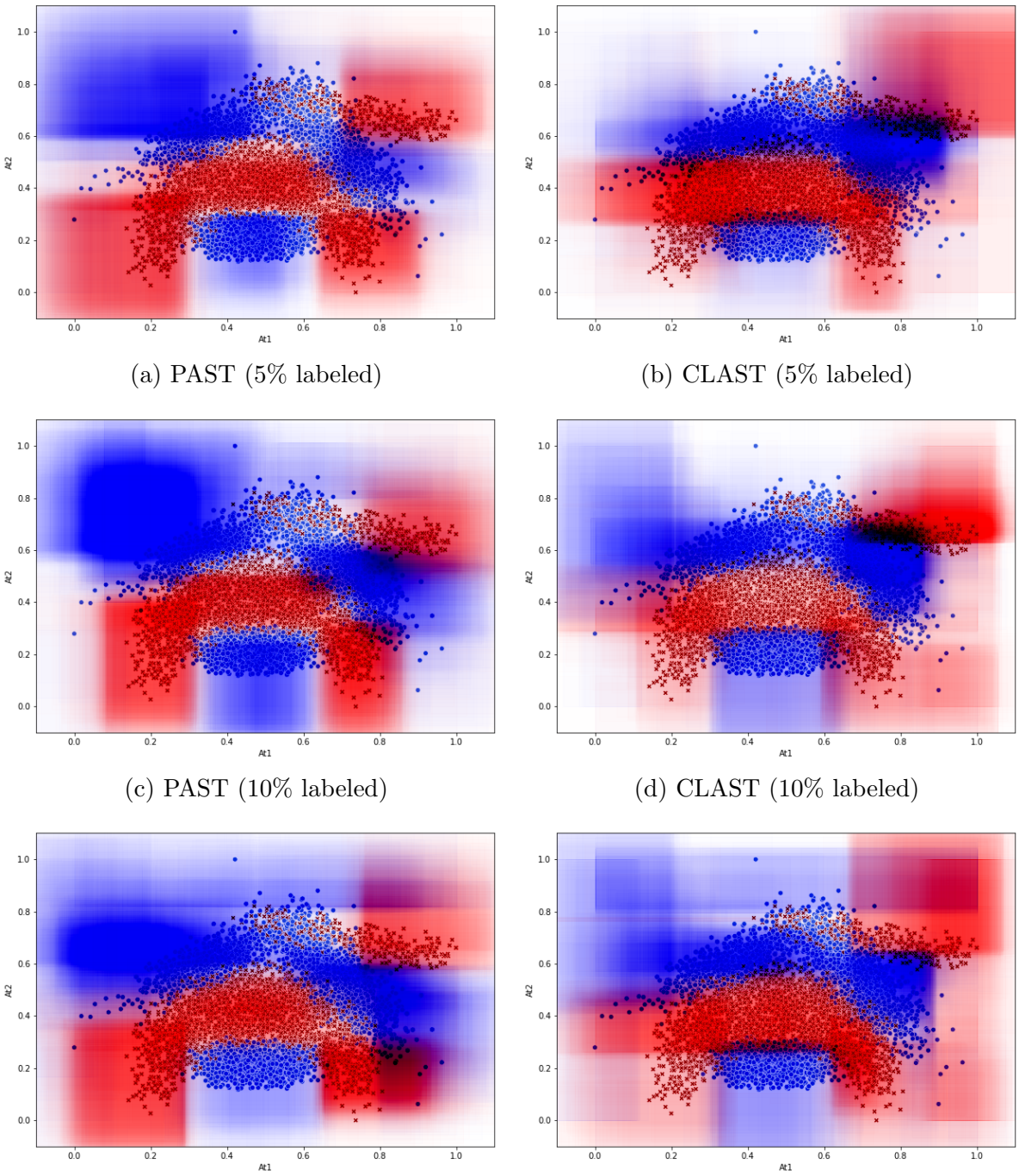
Next, we first detail the experimental methodology followed in each case, and then present and analyze the obtained results.

V.3.1 Comparison with other data stream approaches

In this section, PAST is compared with other approaches for data stream classification in different labeling scenarios.

V.3.1.1 Experimental setup

We add PAST to the list of classifiers discussed in Section III.3.1 of Chapter III. Thus, we compare the performance of PAST with our supervised proposal CLAST and with the following data stream classifiers: CVFDT, CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, HAT, VFDR, AWEC and DWM. In general, the methodology followed is analogous to that used in the analysis of Section III.3.1.1. Hence, we use the same collection of 22 datasets, the classifiers are applied on a random sample of 100,000 examples of each dataset, a test-then-train scheme is followed and the performance of the classifiers is measured in terms of accuracy and G-mean. Likewise, the results shown in this section are averages over thirty runs with



(a) PAST (5% labeled)

(b) CLAST (5% labeled)

(c) PAST (10% labeled)

(d) CLAST (10% labeled)

(e) PAST (25% labeled)

(f) CLAST (25% labeled)

Figure V.2: Rule covering area for *banana* dataset (KEEL Repository (Alcalá-Fdez et al., 2011)) for (a) PAST and (b) CLAST with three different labeling percentages: 5%, 10% and 25%.

different seeds. All the classifiers maintained the same parameter configuration used in Chapter III. PAST is set with the same parameter configuration as CLAST. For further information about the datasets or the parameter setting of the learners, the reader is referred to Chapter III (Section III.3.1).

However, in this case, the comparison is not limited to environments where 100% of the data received is assumed to be labeled. We compared the performance of the algorithms in six different label scarcity scenarios: 5, 10, 25, 50, 75 and 100% of labeled examples. Thus, the 100,000-instance sample is divided into two subsets: L and U. The instances in L are received accompanied by their true class label. The rest of the instances, included in U, are received unlabeled, i.e., the classifiers do not know to which class they truly belong. Which instances belong to L and which to U is decided in a random but stratified way, where as far as possible the original balance between classes will be kept both in L and in U. A labeling percentage of 5% means that 5,000 out of the 100,000 instances in the sample are labeled (L) while the remaining 95,000 instances are not (U). When the labeling percentage reaches 100%, the total of the 100,000 instances will be included in L and U will be an empty set.

Using test-then-train evaluation implies that each example has a double function: first it is used for testing and then for training the classifier. Algorithms that follow an exclusively supervised learning approach will only perform a learning interaction when the example received belongs to L; the algorithm in question will first predict a label for the received example and then use it to learn. If the received example belongs to U, supervised algorithms will predict a label for that example but will not use it to learn, since they are not prepared to do so as long as the class of the datum is unknown. The algorithms that follow an SSL-based approach will complete both steps (predict first, then learn) whether the example is received labeled or not.

We do know the real labels of all the examples from every dataset but the percentage of them that we make accessible to the algorithms varies to simulate conditions of label scarcity. Therefore, although not all labels are used for training, all of them are always used for testing. The performance of an algorithm at instant t can be defined as the proportion of the first t examples that have been correctly classified.

We analyzed the results in each of the labeling scenarios through non-parametric statistical tests. Firstly, we studied whether the average performance of all the classifiers can be considered equivalent according to the Friedman test (Friedman, 1937, 1940). If this hypothesis is rejected, we compare PAST (control-method) with each of the other approaches by means of the post-hoc Finner test (Finner, 1993). Finally, we complement the study with the use of the Wilcoxon signed ranks test (Wilcoxon, 1992) to conduct pairwise comparisons between PAST and the rest of online learners. All tests are applied for $\alpha = 0.05$.

V.3.1.2 Results

Tables V.1-V.4 gather the results of the nine data stream classifiers for the six labeling scenarios in each one of the twenty datasets. The results are assessed as accuracy (Tables V.1-V.2) and G-mean (Tables V.3-V.4). Moreover, the rank of the learners in each problem is also specified. The results in these four tables correspond to the end of the sample, once all the examples have been used for testing.

The large amount of information included in Tables V.1-V.4 is summarized in Table V.5 along with Figures V.3-V.4. Table V.5 shows the average performance of each of the online learners depending on the percentage of labeled data. Likewise, Figures V.3-V.4 illustrate, also for each labeling ratio, the performance distribution of each algorithm in the problem collection. In both cases, we normalized performance, either accuracy or G-mean, before computing average or building the boxplots, so all the problems have the same weight.

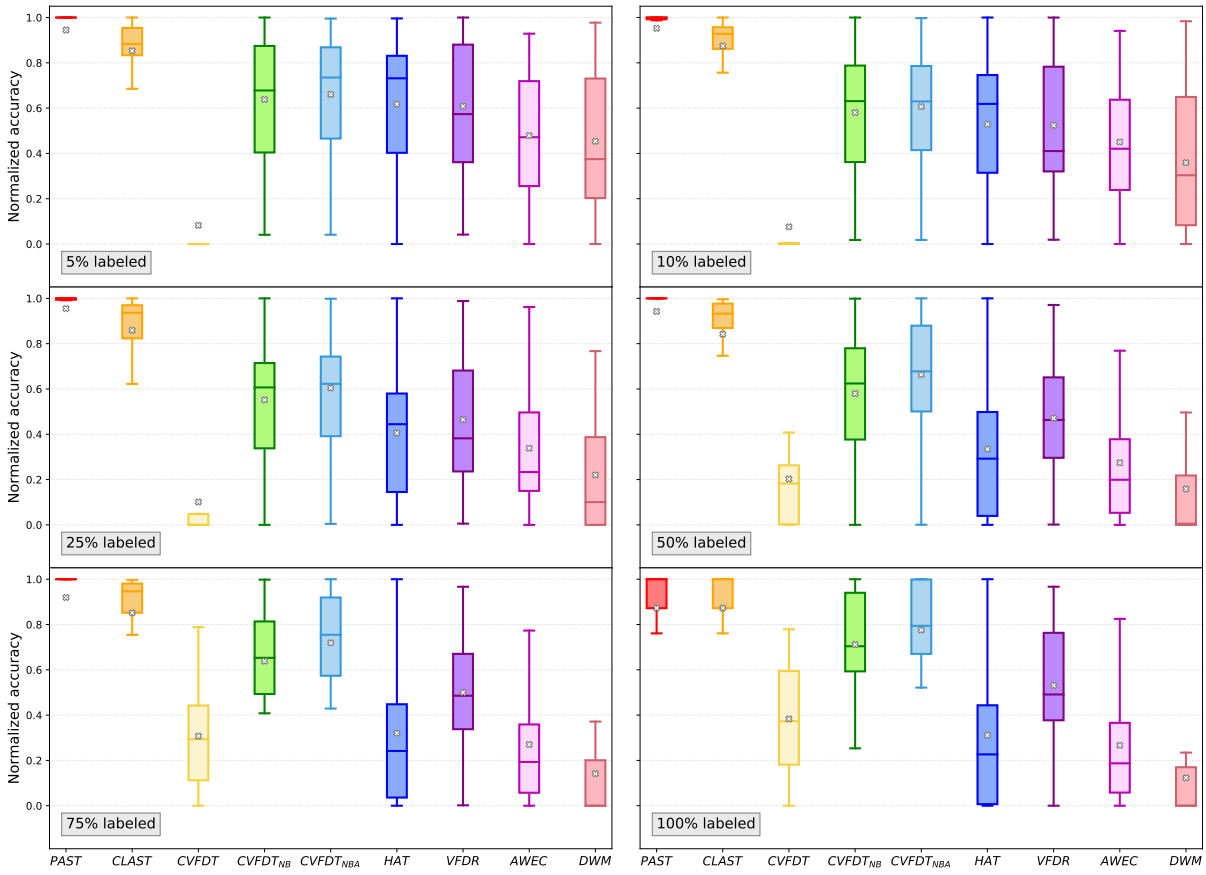


Figure V.3: Distribution of the normalized test accuracy achieved by each classifier in the different labeling scenarios.

Based on these results, we can comment on some observations. PAST is the best positioned classifier in the majority of cases; it achieves the highest accuracy in 98 of 132

Table V.1: Test accuracy and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). This table contains half of the datasets included in the experimentation, the information regarding the remaining datasets is in Table V.2.

Data	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
app	5	90.77 (1)	88.30 (2)	81.49 (9)	86.61 (7)	86.67 (5)	87.26 (3)	86.61 (6)	86.86 (4)	86.30 (8)
	10	92.52 (1)	90.38 (2)	83.70 (9)	87.54 (5)	87.71 (3)	87.59 (4)	87.22 (7)	87.33 (6)	86.28 (8)
	25	94.98 (1)	92.50 (2)	86.69 (8)	90.48 (4)	90.65 (3)	87.99 (6)	89.08 (5)	87.90 (7)	86.30 (9)
	50	96.28 (1)	93.76 (2)	89.21 (6)	93.14 (4)	93.22 (3)	88.10 (8)	91.26 (5)	88.26 (7)	86.31 (9)
	75	96.49 (1)	93.99 (4)	90.63 (6)	94.46 (3)	94.53 (2)	88.07 (8)	92.26 (5)	88.35 (7)	86.31 (9)
	100	93.71 (3)	93.71 (3)	91.97 (6)	95.73 (2)	95.78 (1)	88.13 (8)	93.51 (5)	88.55 (7)	86.32 (9)
aut	5	92.64 (1)	85.32 (2)	44.65 (9)	75.30 (4)	75.27 (5)	79.65 (3)	64.74 (7)	66.03 (6)	53.42 (8)
	10	95.71 (1)	91.35 (2)	47.29 (9)	76.71 (4)	76.67 (5)	78.19 (3)	66.80 (7)	68.12 (6)	53.43 (8)
	25	97.53 (1)	95.53 (2)	50.68 (9)	79.04 (3)	79.02 (4)	75.51 (5)	69.30 (7)	69.57 (6)	53.52 (8)
	50	98.19 (1)	97.16 (2)	53.58 (8)	80.99 (3)	80.97 (4)	74.30 (5)	71.00 (6)	70.70 (7)	53.58 (9)
	75	98.36 (1)	97.85 (2)	57.03 (8)	82.67 (3)	82.65 (4)	73.76 (5)	71.27 (6)	71.18 (7)	53.62 (9)
	100	98.11 (1)	98.11 (1)	60.33 (8)	83.84 (3)	83.83 (4)	73.64 (5)	72.01 (6)	71.49 (7)	53.70 (9)
ban	5	82.52 (1)	77.63 (2)	44.82 (3)	44.65 (8)	44.65 (7)	44.67 (6)	44.68 (5)	43.03 (9)	44.80 (4)
	10	83.70 (1)	80.81 (2)	44.83 (3)	44.65 (7)	44.65 (6)	44.63 (8)	44.69 (5)	43.94 (9)	44.81 (4)
	25	85.03 (1)	83.38 (2)	44.83 (3)	44.64 (7)	44.65 (6)	44.64 (8)	44.69 (5)	44.47 (9)	44.82 (4)
	50	85.77 (1)	84.82 (2)	44.83 (3)	44.66 (7)	44.67 (6)	44.64 (9)	44.70 (5)	44.65 (8)	44.83 (4)
	75	85.64 (1)	84.89 (2)	44.83 (3)	44.67 (8)	44.68 (7)	44.64 (9)	44.71 (5)	44.71 (6)	44.83 (4)
	100	85.01 (1)	85.01 (1)	44.83 (3)	44.68 (8)	44.69 (7)	44.64 (9)	44.72 (6)	44.74 (5)	44.83 (4)
bnd	5	79.52 (1)	77.73 (2)	63.14 (9)	68.51 (3)	68.30 (4)	67.91 (6)	68.07 (5)	67.28 (8)	67.81 (7)
	10	85.90 (1)	84.30 (2)	63.48 (9)	69.45 (3)	69.29 (4)	67.61 (8)	68.79 (5)	68.26 (6)	67.92 (7)
	25	90.84 (1)	89.27 (2)	64.91 (9)	73.02 (3)	72.92 (4)	67.83 (8)	70.48 (5)	69.50 (6)	68.08 (7)
	50	92.29 (1)	91.41 (2)	67.97 (8)	77.02 (3)	76.94 (4)	67.87 (9)	71.70 (5)	70.08 (6)	68.10 (7)
	75	92.95 (1)	92.62 (2)	70.34 (6)	80.31 (3)	80.20 (4)	67.87 (9)	72.85 (5)	70.08 (7)	68.11 (8)
	100	93.33 (1)	93.33 (1)	72.35 (6)	82.47 (3)	82.39 (4)	67.93 (9)	73.87 (5)	70.17 (7)	68.12 (8)
bre	5	85.06 (1)	83.11 (2)	70.83 (9)	75.44 (6)	75.89 (3)	75.89 (4)	75.84 (5)	74.59 (7)	74.57 (8)
	10	88.55 (2)	88.70 (1)	72.41 (9)	78.24 (4)	78.75 (3)	77.52 (5)	76.34 (6)	74.78 (7)	74.59 (8)
	25	92.46 (2)	92.58 (1)	75.52 (7)	83.25 (4)	83.52 (3)	78.50 (6)	78.60 (5)	75.04 (8)	74.65 (9)
	50	94.39 (1)	94.06 (2)	78.93 (6)	87.17 (4)	87.32 (3)	78.53 (7)	80.85 (5)	75.39 (8)	74.66 (9)
	75	95.05 (1)	94.61 (2)	80.78 (6)	89.05 (4)	89.19 (3)	78.76 (7)	82.13 (5)	75.41 (8)	74.65 (9)
	100	95.04 (1)	95.04 (1)	82.30 (6)	90.30 (4)	90.39 (3)	78.77 (7)	82.95 (5)	75.57 (8)	74.66 (9)
car	5	81.18 (6)	84.33 (3)	70.77 (9)	83.34 (4)	83.23 (5)	84.95 (2)	85.09 (1)	73.31 (7)	71.05 (8)
	10	85.85 (3)	88.81 (1)	71.82 (8)	85.17 (4)	85.13 (5)	84.89 (6)	86.04 (2)	74.73 (7)	71.21 (9)
	25	92.07 (2)	92.89 (1)	74.85 (8)	87.63 (4)	87.60 (5)	81.40 (6)	87.73 (3)	75.70 (7)	71.33 (9)
	50	95.10 (1)	94.98 (2)	77.30 (7)	89.94 (3)	89.89 (4)	80.14 (6)	88.90 (5)	76.14 (8)	71.31 (9)
	75	96.17 (1)	96.01 (2)	79.53 (7)	91.31 (3)	91.28 (4)	79.68 (6)	89.82 (5)	76.39 (8)	71.43 (9)
	100	96.59 (1)	96.59 (1)	81.43 (6)	92.37 (3)	92.33 (4)	79.44 (7)	90.67 (5)	76.40 (8)	71.47 (9)
cov	5	64.36 (2)	65.47 (1)	48.68 (9)	61.57 (5)	61.53 (6)	63.18 (4)	63.56 (3)	55.35 (7)	53.89 (8)
	10	66.33 (2)	66.80 (1)	51.02 (9)	62.08 (5)	62.07 (6)	63.06 (4)	63.67 (3)	56.51 (7)	53.92 (8)
	25	67.66 (2)	67.71 (1)	54.11 (8)	62.97 (6)	63.38 (4)	63.33 (5)	64.91 (3)	57.15 (7)	54.08 (9)
	50	68.14 (1)	68.09 (2)	57.45 (7)	63.67 (5)	64.50 (4)	63.28 (6)	65.30 (3)	57.33 (8)	54.14 (9)
	75	68.39 (1)	68.03 (2)	58.82 (7)	64.28 (5)	65.49 (4)	63.36 (6)	65.62 (3)	57.23 (8)	54.14 (9)
	100	68.09 (1)	68.09 (1)	59.57 (7)	64.17 (5)	65.54 (4)	63.37 (6)	65.92 (3)	57.50 (8)	54.17 (9)
eco	5	88.35 (1)	79.06 (7)	43.94 (9)	79.17 (5)	79.15 (6)	79.40 (4)	79.77 (3)	81.21 (2)	75.53 (8)
	10	91.34 (1)	85.46 (2)	51.73 (9)	80.73 (5)	80.68 (6)	80.30 (7)	81.71 (4)	82.66 (3)	75.83 (8)
	25	94.21 (1)	90.51 (2)	62.68 (9)	83.62 (5)	83.91 (4)	81.03 (7)	84.56 (3)	83.40 (6)	75.84 (8)
	50	95.32 (1)	93.23 (2)	66.58 (9)	85.06 (5)	85.62 (4)	81.23 (7)	85.79 (3)	83.85 (6)	75.72 (8)
	75	95.54 (1)	94.12 (2)	69.68 (9)	86.21 (4)	86.62 (3)	81.09 (7)	85.80 (5)	84.02 (6)	75.80 (8)
	100	94.81 (1)	94.81 (1)	71.09 (9)	86.90 (4)	87.35 (3)	81.21 (7)	86.33 (5)	84.19 (6)	75.87 (8)
hay	5	87.97 (2)	88.41 (1)	40.44 (9)	78.46 (3)	78.45 (5)	77.95 (6)	78.46 (4)	72.07 (8)	76.14 (7)
	10	88.36 (2)	88.92 (1)	43.29 (9)	78.52 (3)	78.49 (4)	77.72 (6)	78.46 (5)	70.53 (8)	77.01 (7)
	25	88.64 (1)	87.96 (2)	63.22 (9)	81.27 (3)	81.22 (4)	77.79 (6)	80.52 (5)	72.73 (8)	76.98 (7)
	50	89.08 (1)	87.24 (2)	76.31 (8)	85.47 (3)	85.45 (4)	77.60 (6)	84.38 (5)	73.36 (9)	77.17 (7)
	75	89.20 (1)	86.76 (4)	81.01 (6)	87.25 (2)	87.23 (3)	77.65 (7)	86.17 (5)	73.13 (9)	77.15 (8)
	100	86.71 (4)	86.71 (4)	83.48 (6)	88.22 (1)	88.21 (2)	77.66 (7)	87.18 (3)	74.11 (9)	77.34 (8)
hea	5	87.51 (1)	84.24 (3)	62.28 (9)	84.11 (4)	84.06 (5)	83.52 (7)	84.05 (6)	80.93 (8)	85.09 (2)
	10	89.30 (1)	85.85 (2)	70.05 (9)	84.19 (4)	84.10 (5)	83.50 (7)	83.94 (6)	82.57 (8)	85.33 (3)
	25	91.69 (1)	87.95 (2)	75.77 (9)	86.64 (4)	86.68 (3)	83.54 (8)	84.96 (6)	84.17 (7)	85.51 (5)
	50	92.41 (1)	89.60 (4)	79.24 (9)	89.61 (3)	89.65 (2)	83.39 (8)	86.08 (5)	83.99 (7)	85.57 (6)
	75	92.20 (1)	90.74 (4)	81.64 (9)	91.38 (3)	91.40 (2)	83.42 (8)	87.18 (5)	84.07 (7)	85.56 (6)
	100	91.53 (3)	91.53 (3)	84.00 (8)	92.80 (1)	92.80 (1)	83.40 (9)	87.89 (5)	84.27 (7)	85.60 (6)
kdd	5	96.99 (3)	97.19 (2)	78.46 (8)	96.72 (6)	96.75 (5)	61.44 (9)	97.73 (1)	93.09 (7)	96.91 (4)
	10	97.55 (2)	97.45 (3)	80.27 (8)	97.07 (5)	97.26 (4)	57.14 (9)	97.66 (1)	95.27 (7)	97.00 (6)
	25	97.92 (1)	97.68 (2)	82.30 (8)	96.92 (5)	97.36 (4)	54.39 (9)	97.41 (3)	96.26 (7)	96.80 (6)
	50	98.15 (1)	97.84 (2)	82.63 (8)	96.78 (4)	97.65 (3)	50.00 (9)	96.74 (5)	96.55 (7)	96.62 (6)
	75	98.16 (1)	98.01 (2)	83.89 (8)	96.72 (4)	97.77 (3)	47.78 (9)	96.49 (7)	96.59 (6)	96.59 (5)
	100	98.16 (1)	98.16 (1)	86.51 (8)	96.70 (5)	97.87 (3)	54.24 (9)	96.70 (4)	96.66 (6)	96.56 (7)

Table V.2: Test accuracy and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). This table contains the second half of the datasets included in the experimentation, the information regarding the first half of the datasets is in Table V.1.

Data	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
mam	5	79.64 (1)	79.26 (2)	60.21 (9)	78.10 (6)	78.86 (3)	75.54 (8)	78.78 (4)	77.83 (7)	78.54 (5)
	10	80.02 (1)	79.45 (2)	68.33 (9)	78.35 (6)	79.30 (3)	76.12 (8)	78.16 (7)	78.36 (5)	78.64 (4)
	25	80.34 (1)	79.80 (3)	73.12 (9)	78.29 (6)	79.91 (2)	76.16 (8)	78.04 (7)	78.84 (4)	78.66 (5)
	50	80.47 (1)	79.96 (3)	76.00 (8)	77.49 (7)	80.42 (2)	75.89 (9)	78.43 (6)	78.96 (4)	78.67 (5)
	75	80.45 (2)	80.12 (3)	78.03 (8)	78.17 (7)	81.06 (1)	75.56 (9)	78.64 (6)	79.07 (4)	78.67 (5)
	100	80.25 (2)	80.25 (2)	79.26 (5)	79.29 (4)	81.62 (1)	75.91 (9)	78.92 (7)	79.13 (6)	78.67 (8)
pag	5	92.26 (1)	91.37 (2)	89.76 (5)	88.82 (7)	90.97 (3)	90.94 (4)	88.82 (6)	87.00 (9)	88.65 (8)
	10	92.68 (1)	92.20 (2)	89.78 (8)	90.13 (7)	91.03 (4)	91.08 (3)	90.13 (6)	90.31 (5)	88.69 (9)
	25	92.99 (2)	93.07 (1)	90.06 (6)	88.68 (9)	91.49 (3)	91.33 (4)	88.81 (7)	90.44 (5)	88.72 (8)
	50	93.50 (1)	93.38 (2)	90.73 (5)	84.02 (9)	92.64 (3)	91.77 (4)	85.31 (8)	89.91 (6)	88.73 (7)
	75	93.75 (1)	93.71 (2)	91.38 (6)	82.54 (9)	93.50 (3)	91.80 (4)	84.19 (8)	91.67 (5)	88.60 (7)
	100	93.79 (2)	93.79 (2)	91.88 (6)	86.76 (8)	94.02 (1)	91.95 (5)	84.30 (9)	92.27 (4)	88.59 (7)
pim	5	78.14 (1)	77.68 (2)	72.00 (9)	75.07 (7)	75.16 (5)	74.77 (8)	75.08 (6)	75.65 (3)	75.50 (4)
	10	80.32 (1)	79.31 (2)	73.14 (9)	75.85 (5)	76.39 (3)	75.13 (8)	75.38 (7)	76.25 (4)	75.72 (6)
	25	83.38 (1)	81.62 (2)	74.42 (9)	77.26 (4)	77.83 (3)	75.02 (8)	76.21 (6)	76.58 (5)	75.71 (7)
	50	85.24 (1)	83.86 (2)	75.07 (8)	78.95 (4)	79.44 (3)	75.05 (9)	77.13 (5)	76.83 (6)	75.74 (7)
	75	85.92 (1)	85.24 (2)	76.11 (7)	80.34 (4)	80.74 (3)	74.92 (9)	77.67 (5)	76.97 (6)	75.76 (8)
	100	86.09 (1)	86.09 (1)	76.98 (7)	81.60 (4)	81.96 (3)	75.02 (9)	77.88 (5)	77.04 (6)	75.76 (8)
sah	5	75.79 (1)	74.00 (2)	65.51 (9)	71.34 (5)	71.55 (4)	71.05 (8)	71.63 (3)	71.34 (6)	71.19 (7)
	10	80.20 (1)	79.46 (2)	67.23 (9)	72.00 (5)	72.59 (3)	71.34 (7)	72.09 (4)	71.73 (6)	71.32 (8)
	25	86.46 (1)	85.41 (2)	69.71 (9)	74.30 (4)	75.06 (3)	71.75 (7)	73.25 (5)	72.17 (6)	71.35 (8)
	50	89.91 (1)	88.50 (2)	72.03 (7)	77.26 (4)	77.80 (3)	71.90 (8)	74.62 (5)	72.34 (6)	71.37 (9)
	75	90.93 (1)	89.92 (2)	73.81 (6)	79.35 (4)	79.76 (3)	71.90 (8)	75.22 (5)	72.43 (7)	71.37 (9)
	100	90.72 (1)	90.72 (1)	75.08 (6)	81.18 (4)	81.47 (3)	71.91 (8)	76.07 (5)	72.42 (7)	71.39 (9)
skin	5	95.20 (1)	91.59 (7)	83.72 (9)	94.79 (2)	94.75 (3)	93.87 (5)	94.69 (4)	88.44 (8)	92.18 (6)
	10	95.87 (1)	92.75 (6)	85.35 (9)	95.21 (2)	95.18 (3)	93.63 (5)	94.43 (4)	90.45 (8)	92.33 (7)
	25	96.26 (3)	93.14 (6)	90.73 (9)	96.30 (1)	96.27 (2)	93.68 (5)	96.12 (4)	91.50 (8)	92.39 (7)
	50	96.17 (4)	93.00 (7)	93.83 (5)	97.01 (2)	97.04 (1)	93.81 (6)	96.71 (3)	91.61 (9)	92.40 (8)
	75	95.17 (4)	93.44 (7)	95.05 (5)	97.55 (2)	97.58 (1)	93.82 (6)	97.02 (3)	91.88 (9)	92.41 (8)
	100	93.23 (6)	93.23 (6)	95.87 (4)	97.82 (2)	97.86 (1)	93.80 (5)	97.29 (3)	91.97 (9)	92.41 (8)
tae	5	81.97 (1)	74.63 (2)	34.18 (9)	52.77 (3)	52.74 (5)	52.68 (6)	52.77 (4)	51.01 (8)	51.96 (7)
	10	86.19 (1)	80.80 (2)	36.07 (9)	53.50 (4)	53.42 (5)	53.62 (3)	53.37 (6)	51.83 (8)	52.06 (7)
	25	89.01 (1)	85.71 (2)	45.53 (9)	60.25 (3)	60.20 (4)	54.25 (6)	58.84 (5)	52.48 (7)	52.08 (8)
	50	90.06 (1)	87.68 (2)	51.92 (9)	64.96 (3)	64.91 (4)	54.71 (6)	62.98 (5)	52.69 (7)	52.15 (8)
	75	90.08 (1)	88.24 (2)	57.54 (6)	69.26 (4)	69.32 (3)	54.61 (7)	64.62 (5)	52.30 (8)	52.19 (9)
	100	88.82 (1)	88.82 (1)	62.41 (6)	72.98 (4)	73.05 (3)	54.70 (7)	65.64 (5)	52.47 (8)	52.19 (9)
tic	5	86.79 (1)	85.73 (2)	67.22 (9)	72.39 (5)	72.49 (4)	73.19 (3)	71.41 (6)	69.43 (8)	70.11 (7)
	10	92.92 (1)	91.92 (2)	68.57 (9)	73.59 (5)	73.69 (3)	73.64 (4)	72.93 (6)	70.41 (7)	70.24 (8)
	25	96.88 (1)	95.57 (2)	71.33 (8)	79.21 (3)	79.17 (4)	73.63 (6)	77.74 (5)	71.38 (7)	70.20 (9)
	50	97.73 (1)	96.06 (2)	75.77 (6)	83.79 (4)	83.77 (5)	73.82 (7)	84.98 (3)	71.79 (8)	70.26 (9)
	75	97.67 (1)	96.13 (2)	77.62 (6)	85.76 (4)	85.74 (5)	73.83 (7)	88.88 (3)	72.08 (8)	70.27 (9)
	100	96.31 (1)	96.31 (1)	79.93 (6)	88.23 (4)	88.21 (5)	73.87 (7)	90.90 (3)	72.10 (8)	70.27 (9)
tit	5	80.02 (6)	78.69 (7)	89.28 (4)	92.70 (1)	92.63 (3)	92.64 (2)	81.40 (5)	77.84 (8)	77.58 (9)
	10	81.55 (5)	79.60 (7)	91.66 (4)	95.87 (1)	95.82 (2)	95.82 (3)	81.00 (6)	78.41 (8)	77.73 (9)
	25	84.72 (5)	83.35 (7)	93.20 (4)	97.86 (2)	97.84 (3)	97.87 (1)	83.62 (6)	78.97 (8)	77.93 (9)
	50	87.05 (5)	85.96 (6)	93.73 (4)	98.54 (3)	98.55 (2)	98.57 (1)	85.91 (7)	79.27 (8)	77.92 (9)
	75	88.04 (5)	87.16 (7)	93.95 (4)	98.78 (3)	98.79 (2)	98.82 (1)	87.26 (6)	79.46 (8)	77.95 (9)
	100	88.01 (6)	88.01 (6)	94.14 (4)	98.92 (3)	98.92 (2)	98.99 (1)	88.21 (5)	79.48 (8)	77.96 (9)
veh	5	72.44 (1)	68.29 (2)	25.53 (9)	46.62 (4)	46.61 (7)	46.62 (6)	46.62 (5)	48.86 (3)	43.24 (8)
	10	77.35 (1)	74.07 (2)	25.62 (9)	47.24 (5)	47.23 (7)	47.73 (4)	47.24 (5)	51.39 (3)	43.53 (8)
	25	83.77 (1)	80.85 (2)	38.83 (9)	57.96 (3)	57.90 (4)	56.11 (5)	55.32 (6)	52.92 (7)	43.45 (8)
	50	87.08 (1)	84.76 (2)	51.21 (8)	66.53 (3)	66.48 (4)	58.85 (6)	59.12 (5)	54.35 (7)	43.51 (9)
	75	87.84 (1)	86.45 (2)	56.23 (7)	70.22 (3)	70.17 (4)	59.53 (6)	62.48 (5)	54.94 (8)	43.53 (9)
	100	87.60 (1)	87.60 (1)	59.49 (7)	72.98 (3)	72.93 (4)	60.27 (6)	65.31 (5)	55.04 (8)	43.49 (9)
wdbc	5	94.47 (1)	93.49 (7)	71.82 (9)	93.76 (5)	93.82 (3)	94.10 (2)	93.76 (4)	92.78 (8)	93.63 (6)
	10	94.91 (2)	93.69 (6)	82.98 (9)	93.34 (7)	94.89 (3)	95.37 (1)	92.73 (8)	93.73 (5)	93.78 (4)
	25	95.28 (4)	94.39 (6)	89.89 (9)	95.67 (3)	96.55 (2)	97.13 (1)	94.47 (5)	94.33 (7)	93.83 (8)
	50	95.34 (4)	94.42 (7)	93.20 (9)	97.02 (3)	97.55 (2)	97.96 (1)	95.25 (5)	94.58 (6)	93.84 (8)
	75	95.12 (5)	94.77 (6)	94.48 (8)	97.62 (3)	98.02 (2)	98.47 (1)	95.71 (4)	94.72 (7)	93.86 (9)
	100	94.98 (6)	94.98 (6)	95.15 (5)	98.08 (3)	98.35 (2)	98.77 (1)	96.17 (4)	94.80 (8)	93.86 (9)
yea	5	56.48 (1)	56.04 (2)	31.00 (9)	49.28 (6)	49.25 (7)	49.70 (5)	49.93 (4)	54.66 (3)	31.97 (8)
	10	59.56 (1)	58.06 (2)	32.11 (8)	50.02 (6)	49.97 (7)	50.93 (4)	50.61 (5)	56.19 (3)	31.97 (9)
	25	63.49 (1)	60.65 (2)	38.11 (8)	51.14 (6)	51.09 (7)	51.39 (4)	51.15 (5)	57.39 (3)	31.95 (9)
	50	65.69 (1)	62.52 (2)	41.00 (8)	52.24 (4)	52.21 (6)	51.97 (7)	52.23 (5)	57.91 (3)	32.02 (9)
	75	65.72 (1)	63.46 (2)	44.22 (8)	54.08 (4)	54.05 (5)	51.99 (7)	53.66 (6)	58.04 (3)	31.82 (9)
	100	63.91 (1)	63.91 (1)	46.18 (8)	55.37 (4)	55.35 (5)	52.07 (7)	54.40 (6)	58.32 (3)	31.98 (9)

Table V.3: G-mean and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). This table contains the second half of the datasets included in the experimentation, the information regarding the first half of the datasets is in Table V.3.

Data	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
app	5	87.36 (1)	83.81 (2)	34.66 (9)	79.47 (4)	77.76 (8)	79.04 (6)	79.46 (5)	78.48 (7)	79.65 (3)
	10	89.59 (1)	87.36 (2)	56.76 (9)	79.96 (4)	77.38 (8)	79.08 (7)	80.05 (3)	79.75 (5)	79.47 (6)
	25	93.15 (1)	90.88 (2)	70.42 (9)	84.24 (3)	82.39 (4)	78.36 (8)	81.58 (5)	80.80 (6)	79.55 (7)
	50	95.55 (1)	92.83 (2)	77.42 (9)	88.85 (3)	87.79 (4)	78.77 (8)	85.54 (5)	81.41 (6)	79.51 (7)
	75	95.68 (1)	93.32 (2)	81.98 (6)	91.16 (3)	90.32 (4)	78.41 (9)	87.20 (5)	81.43 (7)	79.54 (8)
	100	93.27 (2)	93.27 (2)	84.59 (6)	93.37 (1)	92.83 (4)	78.54 (9)	89.09 (5)	81.61 (7)	79.56 (8)
aut	5	91.77 (1)	83.63 (2)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	63.68 (3)	0.00 (4)
	10	95.32 (1)	90.85 (2)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	65.89 (3)	0.00 (4)
	25	97.46 (1)	95.45 (2)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	67.37 (3)	0.00 (4)
	50	98.19 (1)	97.23 (2)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	68.84 (3)	0.00 (4)
	75	98.40 (1)	97.92 (2)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	69.41 (3)	0.00 (4)
	100	98.24 (1)	98.24 (1)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	0.00 (4)	69.72 (3)	0.00 (4)
ban	5	81.99 (1)	76.74 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	83.05 (1)	80.33 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	84.50 (1)	83.08 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	85.40 (1)	84.67 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	85.38 (1)	84.74 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	84.84 (1)	84.84 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
bnd	5	77.19 (1)	73.55 (2)	9.69 (9)	51.68 (4)	49.01 (5)	44.74 (8)	47.39 (7)	57.43 (3)	47.80 (6)
	10	84.31 (1)	81.64 (2)	18.32 (9)	54.91 (4)	52.72 (5)	45.25 (8)	49.26 (6)	58.06 (3)	47.37 (7)
	25	90.19 (1)	87.85 (2)	33.62 (9)	64.70 (3)	62.98 (4)	45.82 (8)	54.53 (6)	59.75 (5)	47.10 (7)
	50	91.79 (1)	90.64 (2)	47.11 (7)	72.24 (3)	71.05 (4)	46.03 (9)	59.26 (6)	59.95 (5)	47.09 (8)
	75	92.13 (1)	92.02 (2)	54.90 (7)	76.76 (3)	75.98 (4)	46.08 (9)	62.58 (5)	60.03 (6)	46.92 (8)
	100	92.80 (1)	92.80 (1)	59.71 (7)	79.51 (3)	78.87 (4)	46.38 (9)	64.80 (5)	60.27 (6)	47.10 (8)
bre	5	76.72 (1)	76.26 (2)	37.84 (9)	68.33 (3)	66.68 (5)	63.76 (8)	67.75 (4)	65.06 (7)	66.66 (6)
	10	82.75 (2)	84.23 (1)	52.85 (9)	71.66 (3)	68.41 (4)	67.05 (6)	67.47 (5)	65.68 (8)	66.67 (7)
	25	89.12 (2)	89.56 (1)	59.06 (9)	77.41 (3)	75.60 (4)	68.41 (6)	69.88 (5)	66.19 (8)	66.76 (7)
	50	92.23 (1)	91.62 (2)	65.74 (9)	82.46 (3)	81.38 (4)	67.98 (6)	72.98 (5)	66.58 (8)	66.77 (7)
	75	93.13 (1)	92.35 (2)	69.85 (6)	84.96 (3)	84.25 (4)	68.81 (7)	74.72 (5)	66.56 (9)	66.72 (8)
	100	93.00 (1)	93.00 (1)	72.21 (6)	86.74 (3)	86.23 (4)	69.28 (7)	76.19 (5)	66.64 (9)	66.73 (8)
car	5	34.66 (6)	42.70 (5)	0.00 (9)	53.20 (3)	51.10 (4)	55.90 (2)	57.94 (1)	27.62 (8)	34.08 (7)
	10	51.28 (6)	63.41 (1)	0.00 (9)	57.11 (3)	56.09 (4)	56.00 (5)	58.97 (2)	27.19 (8)	34.12 (7)
	25	74.59 (2)	78.98 (1)	0.00 (9)	63.89 (3)	63.06 (4)	54.97 (6)	61.36 (5)	29.98 (8)	34.34 (7)
	50	85.20 (2)	85.59 (1)	7.29 (9)	68.38 (3)	67.81 (4)	54.61 (6)	64.43 (5)	30.68 (8)	34.15 (7)
	75	88.82 (1)	88.75 (2)	19.87 (9)	72.06 (3)	71.61 (4)	54.57 (6)	68.36 (5)	30.86 (8)	34.21 (7)
	100	90.93 (1)	90.93 (1)	27.14 (9)	75.48 (3)	75.13 (4)	54.35 (6)	73.07 (5)	31.19 (8)	34.23 (7)
cov	5	11.04 (1)	4.59 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	12.88 (1)	4.11 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	17.52 (1)	11.11 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	21.86 (1)	20.78 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	24.65 (1)	24.38 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	26.63 (1)	26.63 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
eco	5	83.94 (1)	67.08 (8)	0.00 (9)	74.13 (5)	74.04 (6)	76.59 (4)	78.09 (3)	78.49 (2)	68.74 (7)
	10	88.96 (1)	78.83 (4)	0.43 (9)	76.10 (6)	75.89 (7)	76.47 (5)	79.59 (3)	81.01 (2)	69.64 (8)
	25	92.71 (1)	87.44 (2)	0.33 (9)	80.03 (5)	79.82 (6)	76.53 (7)	82.78 (3)	82.51 (4)	69.81 (8)
	50	94.17 (1)	91.71 (2)	1.67 (9)	82.15 (6)	82.20 (5)	76.53 (7)	84.30 (3)	83.12 (4)	69.62 (8)
	75	94.68 (1)	93.12 (2)	0.00 (9)	83.27 (5)	83.16 (6)	76.10 (7)	84.64 (3)	83.41 (4)	69.74 (8)
	100	93.98 (1)	93.98 (1)	1.33 (9)	84.23 (4)	84.18 (5)	76.50 (7)	85.26 (3)	83.66 (6)	69.85 (8)
hay	5	89.43 (2)	89.91 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	89.90 (2)	90.47 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	90.27 (1)	88.68 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	90.71 (1)	87.33 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	90.78 (1)	86.53 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	86.46 (1)	86.46 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
hea	5	87.39 (1)	83.92 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	89.24 (1)	85.60 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	91.69 (1)	87.75 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	92.33 (1)	89.44 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	92.14 (1)	90.60 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	91.40 (1)	91.40 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
kdd	5	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	10	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	25	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	50	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	75	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	100	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)

Table V.4: G-mean and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). This table contains the second half of the datasets included in the experimentation, the information regarding the first half of the datasets is in Table V.4.

Data	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
mam	5	79.67 (1)	79.42 (2)	53.51 (9)	78.27 (6)	79.06 (3)	75.43 (8)	78.94 (4)	77.95 (7)	78.74 (5)
	10	80.06 (1)	79.59 (2)	67.15 (9)	78.52 (6)	79.49 (3)	76.08 (8)	78.31 (7)	78.53 (5)	78.83 (4)
	25	80.42 (1)	79.91 (3)	72.98 (9)	78.46 (6)	80.09 (2)	76.08 (8)	78.20 (7)	79.02 (4)	78.85 (5)
	50	80.58 (2)	80.08 (3)	75.99 (8)	77.59 (7)	80.60 (1)	75.81 (9)	78.58 (6)	79.15 (4)	78.86 (5)
	75	80.55 (2)	80.24 (3)	78.12 (8)	78.25 (7)	81.23 (1)	75.47 (9)	78.77 (6)	79.26 (4)	78.86 (5)
	100	80.38 (2)	80.38 (2)	79.35 (5)	79.36 (4)	81.77 (1)	75.87 (9)	79.04 (7)	79.32 (6)	78.87 (8)
pag	5	20.11 (1)	1.12 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	35.11 (1)	6.96 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	48.90 (1)	35.07 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	58.86 (1)	50.51 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	62.57 (1)	56.68 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	60.26 (1)	60.26 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
pim	5	74.44 (1)	72.74 (2)	55.61 (9)	70.80 (4)	69.17 (7)	66.75 (8)	70.74 (5)	70.26 (6)	71.11 (3)
	10	77.26 (1)	75.85 (2)	61.62 (9)	71.43 (3)	69.73 (7)	67.34 (8)	71.06 (6)	71.13 (5)	71.17 (4)
	25	81.48 (1)	79.16 (2)	63.93 (9)	73.07 (3)	71.97 (5)	67.40 (8)	72.02 (4)	71.56 (6)	71.21 (7)
	50	83.62 (1)	81.86 (2)	67.61 (8)	75.10 (3)	74.44 (4)	66.97 (9)	72.88 (5)	71.76 (6)	71.21 (7)
	75	83.88 (1)	83.52 (2)	70.18 (8)	76.93 (3)	76.46 (4)	66.78 (9)	73.36 (5)	71.94 (6)	71.24 (7)
	100	84.43 (1)	84.43 (1)	71.80 (7)	78.55 (3)	78.19 (4)	66.83 (9)	73.57 (5)	71.94 (6)	71.25 (8)
sah	5	72.91 (1)	70.81 (2)	9.66 (9)	67.92 (5)	67.10 (7)	65.16 (8)	68.30 (4)	67.45 (6)	68.64 (3)
	10	77.90 (1)	77.56 (2)	40.44 (9)	67.46 (6)	65.80 (7)	64.26 (8)	67.74 (5)	67.93 (4)	68.83 (3)
	25	85.14 (1)	84.10 (2)	55.37 (9)	69.03 (3)	67.35 (7)	63.59 (8)	68.20 (6)	68.49 (5)	68.85 (4)
	50	88.86 (1)	87.36 (2)	59.81 (9)	72.22 (3)	70.85 (4)	63.60 (8)	69.44 (5)	68.59 (7)	68.85 (6)
	75	89.87 (1)	88.96 (2)	62.68 (9)	74.74 (3)	73.45 (4)	63.53 (8)	70.01 (5)	68.65 (7)	68.87 (6)
	100	89.81 (1)	89.81 (1)	64.96 (8)	77.08 (3)	76.13 (4)	63.48 (9)	70.92 (5)	68.72 (7)	68.87 (6)
skn	5	96.34 (1)	91.03 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	97.02 (1)	92.87 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	97.40 (1)	94.29 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	97.34 (1)	94.77 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	96.76 (1)	95.15 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	95.10 (1)	95.10 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
tae	5	81.70 (1)	74.24 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	10	85.98 (1)	80.52 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	25	88.86 (1)	85.51 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	50	89.93 (1)	87.49 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	89.92 (1)	88.06 (2)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	100	88.65 (1)	88.65 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
tic	5	81.49 (2)	83.73 (1)	39.20 (9)	60.45 (5)	59.51 (6)	63.76 (3)	62.52 (4)	50.68 (7)	48.85 (8)
	10	90.72 (2)	90.82 (1)	57.31 (7)	65.20 (4)	64.62 (5)	64.58 (6)	67.13 (3)	50.26 (8)	48.78 (9)
	25	96.21 (1)	94.83 (2)	57.51 (7)	73.36 (4)	72.79 (5)	64.55 (6)	74.20 (3)	50.60 (8)	48.97 (9)
	50	97.35 (1)	95.30 (2)	64.17 (7)	79.85 (4)	79.58 (5)	64.84 (6)	82.86 (3)	50.60 (8)	48.85 (9)
	75	97.30 (1)	95.31 (2)	67.03 (6)	82.53 (4)	82.31 (5)	64.80 (7)	87.41 (3)	51.22 (8)	48.92 (9)
	100	95.51 (1)	95.51 (1)	71.62 (6)	85.72 (4)	85.54 (5)	64.90 (7)	89.76 (3)	50.77 (8)	48.83 (9)
tit	5	78.09 (6)	76.27 (7)	87.13 (4)	90.89 (1)	90.77 (2)	90.71 (3)	79.25 (5)	75.75 (8)	75.46 (9)
	10	79.80 (5)	77.99 (7)	90.66 (4)	95.09 (1)	95.03 (2)	94.98 (3)	79.54 (6)	77.05 (8)	75.75 (9)
	25	83.21 (6)	82.22 (7)	92.77 (4)	97.61 (1)	97.57 (3)	97.58 (2)	83.38 (5)	77.82 (8)	75.86 (9)
	50	85.87 (5)	84.80 (7)	93.52 (4)	98.42 (3)	98.44 (2)	98.45 (1)	85.55 (6)	78.20 (8)	75.84 (9)
	75	86.95 (5)	85.98 (7)	93.83 (4)	98.72 (3)	98.75 (2)	98.76 (1)	86.89 (6)	78.40 (8)	75.86 (9)
	100	86.91 (6)	86.91 (6)	94.04 (4)	98.87 (3)	98.90 (2)	98.96 (1)	87.79 (5)	78.47 (8)	75.89 (9)
veh	5	69.68 (1)	64.43 (2)	6.24 (9)	40.54 (4)	40.52 (7)	40.53 (6)	40.54 (5)	45.19 (3)	38.85 (8)
	10	75.68 (1)	71.55 (2)	4.55 (9)	41.31 (5)	41.30 (7)	42.72 (4)	41.31 (5)	48.06 (3)	39.36 (8)
	25	83.12 (1)	79.70 (2)	31.06 (9)	57.06 (3)	56.98 (4)	53.93 (5)	53.84 (6)	49.92 (7)	39.27 (8)
	50	86.74 (1)	84.16 (2)	43.57 (8)	66.04 (3)	65.94 (4)	56.98 (6)	58.08 (5)	51.49 (7)	39.29 (9)
	75	87.54 (1)	86.04 (2)	48.94 (8)	69.72 (3)	69.61 (4)	57.65 (6)	61.57 (5)	52.25 (7)	39.35 (9)
	100	87.28 (1)	87.28 (1)	53.51 (7)	72.41 (3)	72.31 (4)	58.45 (6)	64.43 (5)	52.36 (8)	39.32 (9)
wdb	5	93.42 (2)	92.42 (7)	50.87 (9)	93.19 (4)	93.19 (5)	93.44 (1)	93.20 (3)	91.36 (8)	92.81 (6)
	10	93.91 (3)	92.79 (6)	75.91 (9)	93.23 (4)	94.02 (2)	94.76 (1)	92.36 (8)	92.51 (7)	92.94 (5)
	25	94.60 (4)	93.53 (6)	87.41 (9)	95.73 (3)	96.04 (2)	96.79 (1)	94.03 (5)	93.30 (7)	92.99 (8)
	50	94.48 (5)	93.30 (7)	91.91 (9)	97.10 (3)	97.24 (2)	97.78 (1)	94.87 (4)	93.59 (6)	92.99 (8)
	75	93.92 (5)	93.68 (7)	93.49 (8)	97.70 (3)	97.77 (2)	98.31 (1)	95.37 (4)	93.74 (6)	93.00 (9)
	100	93.91 (6)	93.91 (6)	94.29 (5)	98.11 (3)	98.17 (2)	98.66 (1)	95.91 (4)	93.86 (8)	93.00 (9)
yea	5	38.00 (1)	28.27 (4)	0.00 (9)	26.27 (6)	26.23 (7)	28.62 (3)	26.97 (5)	32.69 (2)	18.70 (8)
	10	43.43 (1)	35.47 (2)	0.00 (9)	28.34 (6)	27.87 (7)	34.70 (3)	29.89 (5)	34.35 (4)	18.29 (8)
	25	50.63 (1)	46.10 (2)	0.00 (9)	34.65 (5)	33.70 (6)	38.36 (3)	33.11 (7)	35.52 (4)	19.23 (8)
	50	54.55 (1)	52.80 (2)	0.00 (9)	36.01 (5)	35.43 (6)	39.20 (3)	38.05 (4)	35.38 (7)	18.98 (8)
	75	54.09 (2)	56.15 (1)	0.00 (9)	38.80 (5)	37.51 (6)	38.94 (4)	41.68 (3)	34.72 (7)	18.90 (8)
	100	58.49 (1)	58.49 (1)	0.00 (9)	42.00 (4)	41.47 (5)	39.27 (6)	42.85 (3)	35.69 (7)	19.21 (8)

Table V.5: Comparison of the average performance of PAST with CLAST and the rest of data stream learners for each percentage of labeled samples (%Lab).

	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
Accuracy	5	0.95 (1.64)	0.85 (2.91)	0.08 (8.27)	0.64 (4.82)	0.66 (4.68)	0.62 (5.05)	0.61 (4.41)	0.48 (6.55)	0.45 (6.68)
	10	0.95 (1.50)	0.87 (2.45)	0.08 (8.32)	0.58 (4.64)	0.61 (4.27)	0.53 (5.32)	0.52 (5.23)	0.45 (6.18)	0.36 (7.05)
	25	0.95 (1.59)	0.86 (2.45)	0.10 (8.00)	0.55 (4.18)	0.60 (3.68)	0.41 (5.86)	0.46 (5.05)	0.34 (6.59)	0.22 (7.59)
	50	0.94 (1.45)	0.84 (2.77)	0.20 (7.09)	0.58 (4.09)	0.66 (3.45)	0.33 (6.55)	0.47 (4.95)	0.27 (6.86)	0.16 (7.77)
	75	0.92 (1.55)	0.85 (2.95)	0.31 (6.64)	0.64 (4.05)	0.72 (3.23)	0.32 (6.64)	0.50 (5.09)	0.27 (6.91)	0.14 (7.95)
	100	0.87 (2.09)	0.87 (2.09)	0.38 (6.23)	0.71 (3.73)	0.78 (3.00)	0.31 (6.73)	0.53 (4.95)	0.27 (6.95)	0.12 (8.18)
G-mean	5	0.94 (1.59)	0.82 (2.82)	0.08 (6.27)	0.52 (3.64)	0.51 (4.45)	0.50 (4.27)	0.49 (3.68)	0.49 (4.50)	0.43 (4.77)
	10	0.95 (1.64)	0.85 (2.27)	0.09 (6.18)	0.47 (3.68)	0.46 (4.27)	0.44 (4.45)	0.43 (4.09)	0.43 (4.32)	0.36 (5.05)
	25	0.96 (1.45)	0.89 (2.32)	0.09 (6.18)	0.44 (3.23)	0.44 (3.73)	0.35 (4.64)	0.37 (4.23)	0.35 (4.77)	0.27 (5.45)
	50	0.95 (1.45)	0.89 (2.41)	0.10 (5.95)	0.43 (3.41)	0.45 (3.41)	0.28 (4.77)	0.35 (4.00)	0.28 (4.95)	0.18 (5.64)
	75	0.93 (1.45)	0.90 (2.41)	0.16 (5.59)	0.46 (3.36)	0.48 (3.45)	0.26 (4.95)	0.36 (3.91)	0.25 (5.09)	0.16 (5.77)
	100	0.93 (1.55)	0.93 (1.55)	0.21 (5.18)	0.50 (3.05)	0.51 (3.36)	0.26 (5.09)	0.38 (3.91)	0.25 (5.41)	0.15 (5.95)

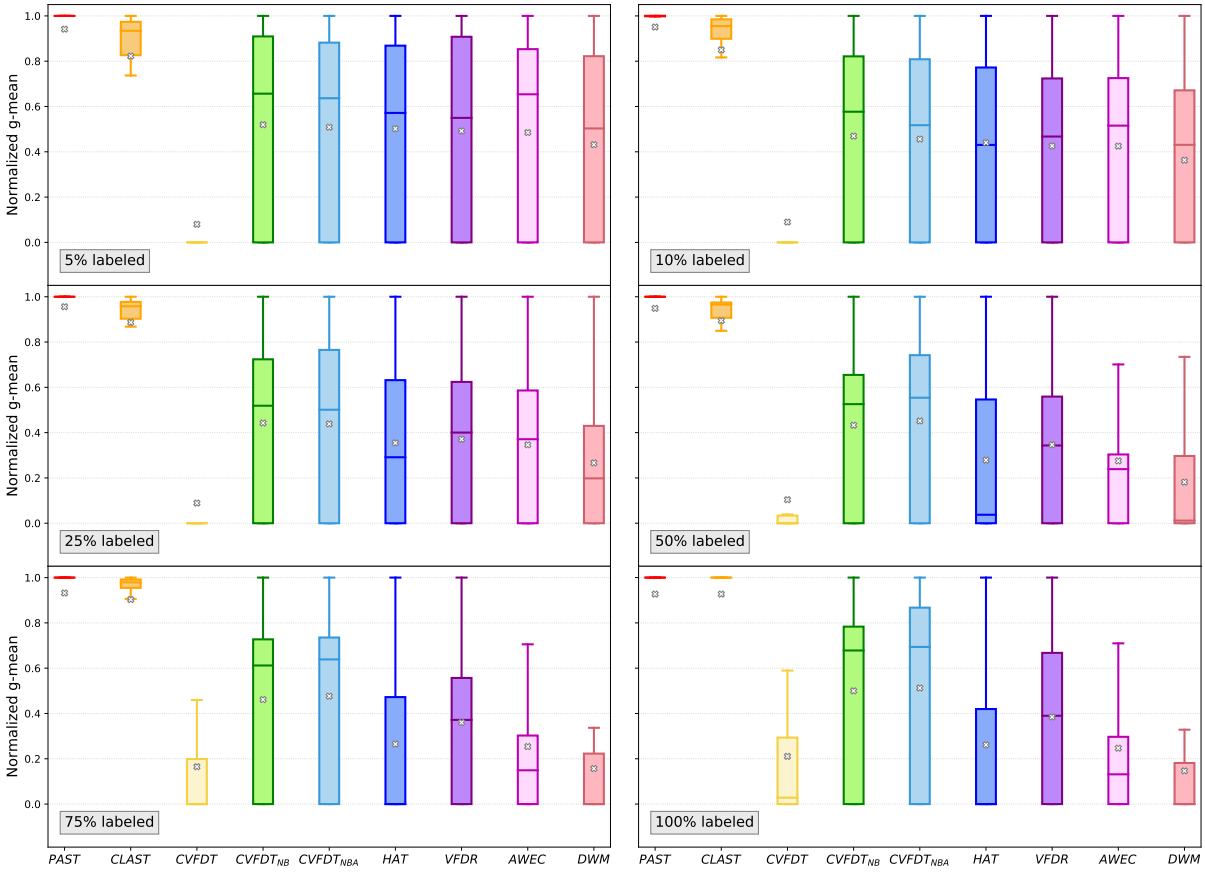


Figure V.4: Distribution of the normalized test G-mean achieved by each classifier in the different labeling scenarios.

scenarios (74%) and the highest G-mean in 99 of 126 scenarios (79 %, excluding kddcup dataset). Furthermore, in the remaining scenarios, second place is the most common position. The next best performer is CLAST. Half of the times CLAST reaches the first position correspond to 100% labeled scenarios and, therefore, both PAST and CLAST are tied in first place. The cases in which CLAST obtains a better result than PAST for low labeling percentages are circumscribed to a small set of 3-4 datasets.

PAST is the classifier that obtains the best average performance for the six labeling percentages (tied with CLAST in the case of 100% labeled). This is true regardless of whether the measurement used is accuracy or G-mean. In fact, the good performance of PAST is even more remarkable for G-mean. While other classifiers worsen their results considerably with respect to accuracy, PAST maintains its good performance in terms of G-mean in the vast majority of problems, even when there is class imbalance. This shows that, as CLAST, PAST's learning strategy allows it to learn every class. Moreover, in this case we see that this ability is preserved even in very challenging environments, where label scarcity is coupled with class imbalance.

This good average performance of PAST is also sustained if we look at the distribution of results (Figures V.3-V.4). PAST gets the highest value for all quartiles, followed by CLAST. In general, the variability of the results is greater in the scenarios with fewer labels. However, the difference between high and low labeling percentages is lower for PAST than for proposals such as $\text{CVFDT}_{\text{NBA}}$, HAT or AWEC. Its variability hardly increases when the percentage of labeled data decreases. Furthermore, it is worth to mention the better adaptation of CLAST to low levels of labeled data as compared to the other supervised methods.

We applied Friedman's test to the results of each percentage of labeled data and, in the six cases, it rejected the null hypothesis that the medians of performance of all the algorithms were equivalent. Then, we applied the post-hoc Finner test to compare the performance of PAST with every other classifier. Moreover, we complete our analysis using the Wilcoxon signed ranks test to perform the pairwise comparisons between PAST and each of the other data stream classifiers. Both Finner and Wilcoxon tests found PAST to perform significantly better than CVFDT, CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, HAT, VFDR, AWEC and DWM for all the labeling scenarios, with only a few exceptions in the case of Finner test. As part of the multiple comparison, PAST is found to outperform CLAST but the improvement is not considered significant. Nonetheless, the pairwise comparison conducted through Wilcoxon signed ranks test found that PAST significantly improves the performance of CLAST for the five scenarios that included unlabeled data. Tables V.6 and V.7 sum up the results of the comparisons conducted by means of post-hoc Finner test (after Friedman's test rejected the null hypothesis) and pairwise Wilcoxon signed ranks test, at a significance level of 0.05. The symbols $\oplus$ and $\ominus$ indicate that *PAST* significantly improves/degrades the performance obtained with the method in the column. Likewise, the symbols $+$ and $-$ denote a non-significant improvement/degradation, while symbol $=$ indicates that both algorithms obtained the exact same results.

Table V.6: Multiple comparison of the performance of PAST with the remaining data stream learners by means of a post-hoc Finner test after Friedman's test rejected null hypothesis of equality of all learners. The same comparison is conducted for both accuracy and G-mean.

	%Lab	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	5	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	10	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	25	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	50	+	⊕	⊕	+	⊕	⊕	⊕	⊕
	75	+	⊕	⊕	+	⊕	⊕	⊕	⊕
	100	=	⊕	+	+	⊕	⊕	⊕	⊕
G-mean	5	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	10	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	25	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	50	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	75	+	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	100	=	⊕	⊕	⊕	⊕	⊕	⊕	⊕

[1] ⊕/⊖: PAST significantly improves/degrades the performance of the method in the column
+/−: PAST improves/degrades the performance of the method in the column
=: PAST obtains the same results that the method in the column

Table V.7: Pairwise comparison of the performance of PAST with the remaining data stream learners by means of a Wilcoxon signed ranks test. Same comparison is conducted for both accuracy and G-mean.

	%Lab	$CLAST$	$CVFDT$	$CVFDT_{NB}$	$CVFDT_{NBA}$	HAT	$VFDR$	$AWEC$	DWM
Accuracy	5	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	10	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	25	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	50	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	75	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	100	=	⊕	⊕	⊕	⊕	⊕	⊕	⊕
G-mean	5	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	10	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	25	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	50	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	75	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
	100	=	⊕	⊕	⊕	⊕	⊕	⊕	⊕

[1] ⊕/⊖: PAST significantly improves/degrades the performance of the method in the column
+/−: PAST improves/degrades the performance of the method in the column
=: PAST obtains the same results that the method in the column

So far we have compared the different data stream learners based on their final performance once all the examples have been processed. The number of training examples for most the learners depends on the number of labels. However, every received example is always first used for testing and, therefore, all the classifiers are tested on the 100,000 examples regardless of the labeling percentage. Figures V.5-V.6 graphically represent the evolution of performance as the amount of tested examples increases over time. PAST maintains the best performance during the entire duration of the experiment for the six labeling scenarios. The difference of PAST over CLAST is observed for all the labeling percentages below 100%, although it is more noticeable for the lowest labeling percentages (5-10%) and decreases as this percentage grows. The steady trend followed by all the learners may be explained as the result of the combination of two factors: (1) the addressed problems are not real data streams, and (2) we are representing the average for a collection of 22 datasets and, therefore, the particularities of each problem are dispelled.

V.3.2 Comparison with batch approaches

Throughout this section we compare the performance of PAST with CLAST and eight offline batch classifiers, both supervised and semi-supervised, which are based on different machine learning techniques. These offline classifiers do not see the dataset as a stream but they address it as a complete and static entity.

V.3.2.1 Experimental setup

Again, we add PAST to the experimentation carried out in Chapter III (Section III.3.2) and extend it by covering scenarios where not all the data received are labeled. Thus, for the most part, the methodology followed is analogous to the one used in Section III.3.2. Nevertheless, we expanded the set of learners by including two proposals for offline Semi-Supervised Learning (SSL). Hence, we compare the performance of PAST with CLAST and eight batch learning algorithms: Gaussian Naive Bayes (NB), Decision Tree (DT), k-Nearest Neighbors (kNN), Support Vector Classification (SVC), Random Forest (RF), Bagging, Label Propagation (LP) and Label Spreading (LS). Both LP (Delalleau et al., 2005) and LS (Delalleau et al., 2005) are semi-supervised approaches based on label propagation. LP uses the raw similarity matrix constructed from the data with no modifications while LS minimizes a loss function that has regularization properties, therefore, it is often more robust to noise. The algorithm iterates on a modified version of the original graph and normalizes the edge weights by computing the normalized graph Laplacian matrix.

We used the same collection of datasets as in Section III.3.2 and the classifiers are compared based on the accuracy and G-mean obtained. The classifiers are also set with the same parameter configuration used in Chapter III. PAST is configured with the same parameter set as CLAST. For further information about the characteristics of the employed datasets or the parameter setting of the learners, we refer the reader to Chapter

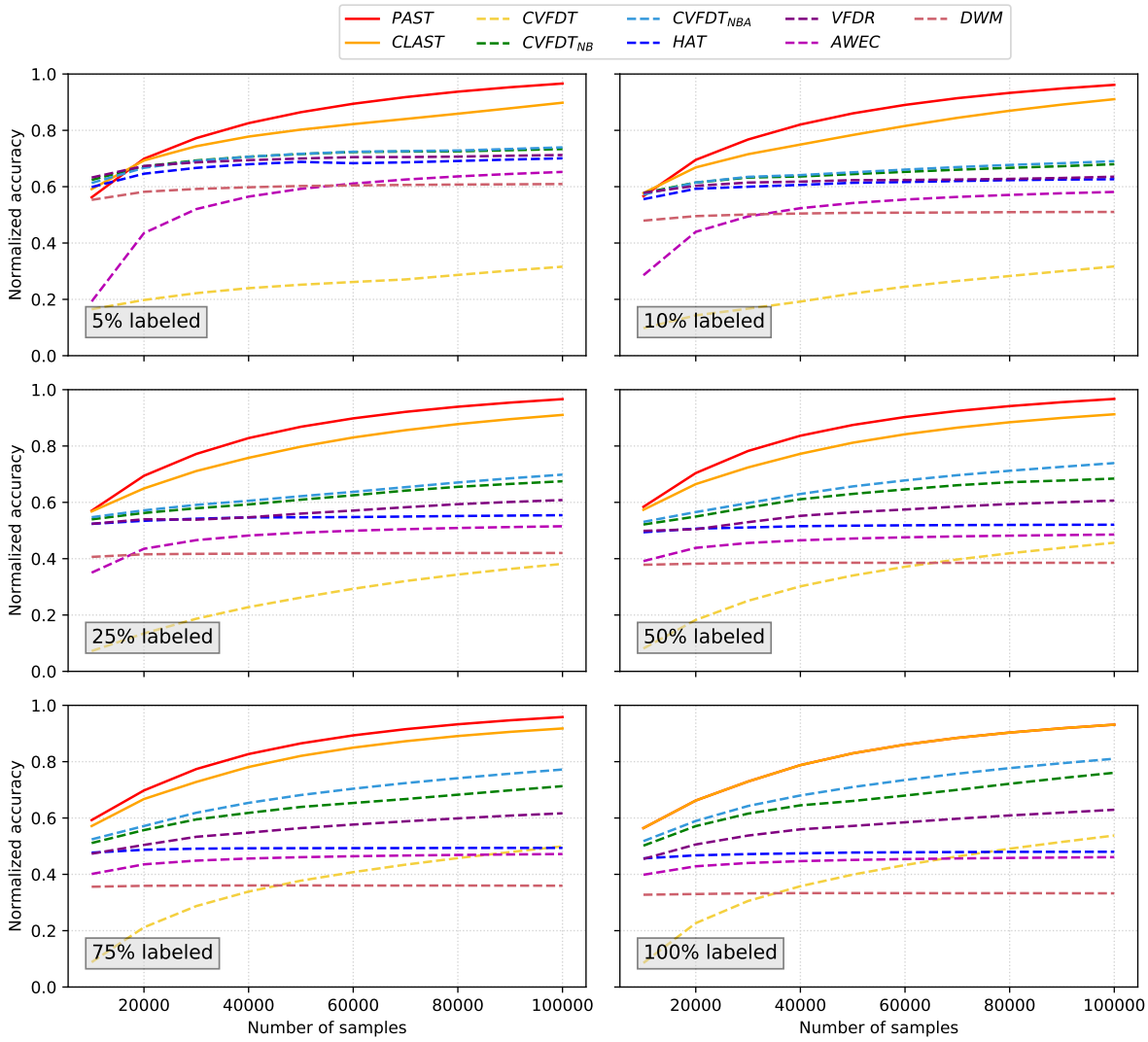


Figure V.5: Average normalized accuracy evolution of the different data stream approaches as the amount of data received increases for the 22 datasets studied.

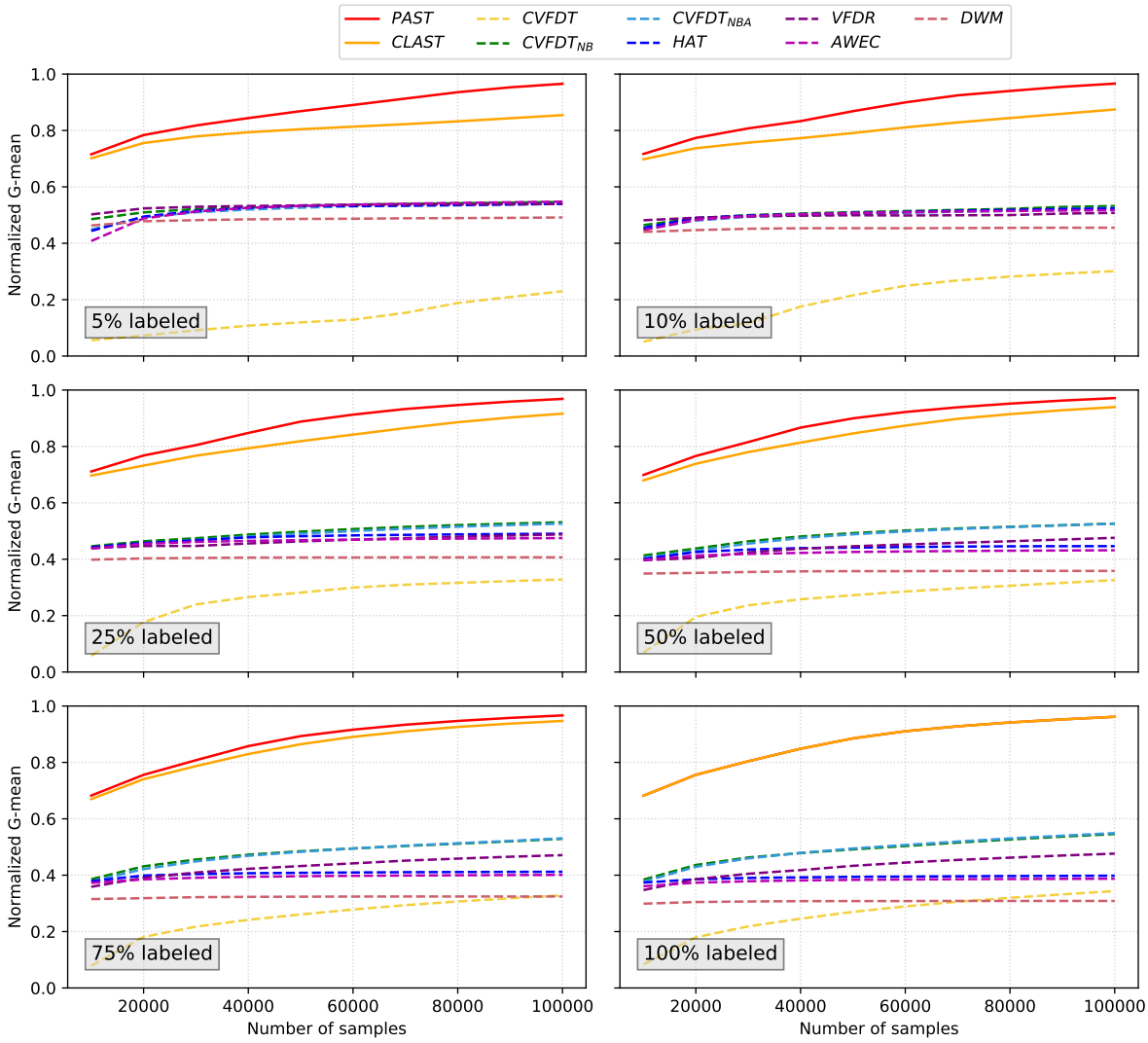


Figure V.6: Average normalized G-mean evolution of the different data stream approaches as the amount of data received increases for the 22 datasets studied.

III.

Likewise, since offline proposals are not designed for incremental learning, we discarded test-then-train scheme. The datasets are divided into train-test partitions, where 90% of the dataset is used for training and the remaining 10% for testing. PAST and CLAST are allowed to iterate 10 times over the training set. The experiments are executed with 30 different seeds and the average results are shown.

As mentioned, we extended the experimentation to cover label scarcity scenarios. We conducted experiments with the same six labeling percentages used in the previous section: 5, 10, 25, 50, 75 and 100%. Therefore, each training partition is divided in two subsets: L (labeled instances) and U (unlabeled instances). L and U are built in a random but stratified way, preserving the original class balance. All the algorithms received the same labeled data. Nonetheless, algorithms that follow a supervised learning approach will only learn from such labeled data (L), while semi-supervised approaches will take advantage of the full training set (L+U).

We conduct a statistical analysis of the results based on three non-parametric tests: Friedman's test (Friedman, 1937, 1940), post-hoc Finner test (Finner, 1993) and Wilcoxon signed ranks test (Wilcoxon, 1992).

V.3.2.2 Results

Hereafter, we compare PAST to the enumerated offline learners. Tables V.8-V.11 show the accuracy and G-mean of the classifiers on the different datasets and for each ratio of labeled data. In addition to PAST and the set of offline learners, CLAST is also included in the tables. As in previous section, along with accuracy and G-mean, the rank of each algorithm is also specified.

Table V.12 and Figures V.7-V.8 aggregate the results shown in the previous tables by the amount of labeled data. This table gathers the average normalized performance of each classifier on each labeling case, while in the figures, the distribution of the normalized performance is represented in boxplots.

From the experimental results, we can note the good performance of PAST. It achieves the best accuracy in nine cases and the best G-mean in nineteen. It improves the average performance of CLAST for all the labeling percentages below 100%, achieving the third highest average for both accuracy and G-mean for the six labeling scenarios. RF and Bagging are the only two methods that outperform PAST. The performance of PAST surpasses that of the two offline semi-supervised approaches. In fact, LP and LS perform quite poorly, with very high variance in their results. They are able to reach competitive results in some of the problems but they are left behind in many other cases.

As we already observed in previous sections, both PAST and CLAST have demonstrated great capacity to make accurate predictions for minority classes in imbalanced problems, clearly improving the behavior of other approaches. This is also observed here.

Table V.8: Test accuracy and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). The classifiers are grouped according to whether they are online or offline, and supervised (SL) or semi-supervised (SSL). This table contains half of the datasets included in the experimentation, the information regarding the remaining datasets is in Table V.9.

		Online SSL	Online SL	Offline SSL		Offline SL					
Data	%Lab	*PAST*	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
app	5	69.52 (9)	64.52 (10)	82.39 (2)	82.79 (1)	80.18 (6)	74.61 (8)	80.18 (6)	82.33 (3)	81.15 (5)	81.76 (4)
	10	74.06 (10)	78.09 (8)	84.27 (3)	85.21 (1)	80.39 (7)	77.42 (9)	84.00 (4)	84.27 (2)	82.67 (5)	81.97 (6)
	25	74.45 (10)	80.97 (9)	83.24 (7)	83.85 (6)	86.00 (1)	81.33 (8)	85.39 (2)	84.85 (3)	84.45 (4)	84.45 (4)
	50	79.97 (9)	80.09 (8)	86.73 (3)	87.09 (2)	85.85 (5)	79.58 (10)	84.00 (7)	88.64 (1)	86.73 (3)	84.24 (6)
	75	79.97 (9)	80.15 (8)	85.45 (6)	86.09 (3)	85.82 (5)	76.64 (10)	85.09 (7)	87.00 (2)	88.67 (1)	85.85 (4)
	100	77.64 (9)	77.64 (9)	86.42 (4)	86.42 (4)	86.15 (6)	80.97 (8)	85.15 (7)	87.00 (2)	87.09 (1)	86.82 (3)
aut	5	38.24 (4)	35.61 (6)	12.83 (9)	12.83 (9)	35.86 (5)	38.93 (3)	31.42 (8)	35.42 (7)	46.88 (2)	48.79 (1)
	10	42.56 (5)	34.82 (7)	12.83 (9)	12.83 (9)	43.67 (4)	47.25 (3)	34.53 (8)	38.06 (6)	52.19 (1)	51.72 (2)
	25	55.79 (4)	47.13 (6)	12.83 (9)	12.83 (9)	48.35 (5)	60.06 (3)	37.99 (8)	39.31 (7)	63.51 (1)	61.96 (2)
	50	60.40 (4)	57.96 (5)	12.83 (9)	12.83 (9)	54.21 (6)	72.06 (3)	37.38 (8)	42.07 (7)	77.06 (2)	78.86 (1)
	75	61.36 (4)	58.17 (5)	12.83 (9)	12.83 (9)	55.29 (6)	79.58 (3)	39.43 (8)	40.58 (7)	81.67 (2)	84.04 (1)
	100	62.99 (4)	62.99 (4)	12.83 (9)	12.83 (9)	52.58 (6)	83.35 (3)	41.76 (7)	40.78 (8)	88.68 (1)	87.82 (2)
ban	5	83.31 (8)	71.92 (9)	85.42 (6)	89.00 (1)	59.26 (10)	83.50 (7)	86.77 (3)	88.22 (2)	86.44 (4)	85.77 (5)
	10	85.19 (8)	74.43 (9)	88.29 (3)	89.48 (1)	60.16 (10)	85.25 (7)	88.15 (4)	88.96 (2)	87.69 (5)	87.19 (6)
	25	86.77 (7)	76.36 (9)	89.91 (2)	90.26 (1)	60.79 (10)	86.34 (8)	88.57 (5)	89.85 (3)	88.72 (4)	88.32 (6)
	50	87.32 (7)	81.72 (9)	90.20 (2)	90.21 (1)	61.59 (10)	86.64 (8)	88.41 (5)	90.08 (3)	88.96 (4)	88.40 (6)
	75	87.45 (7)	86.51 (9)	90.25 (2)	90.26 (1)	61.19 (10)	86.96 (8)	88.38 (6)	90.19 (3)	89.10 (4)	88.57 (5)
	100	87.29 (7)	87.29 (7)	90.33 (1)	90.26 (3)	61.29 (10)	87.11 (9)	88.35 (6)	90.30 (2)	89.33 (4)	89.16 (5)
bnd	5	53.61 (8)	53.77 (7)	36.97 (9)	36.97 (9)	56.43 (6)	57.87 (5)	58.32 (4)	63.03 (1)	59.16 (2)	58.42 (3)
	10	56.70 (4)	55.34 (7)	36.97 (9)	36.97 (9)	54.26 (8)	56.47 (5)	56.28 (6)	63.03 (3)	63.93 (1)	63.38 (2)
	25	56.79 (8)	57.33 (7)	36.97 (9)	36.97 (9)	59.34 (5)	60.08 (4)	59.01 (6)	63.03 (3)	66.38 (1)	66.03 (2)
	50	56.61 (6)	55.80 (7)	36.97 (9)	36.97 (9)	52.19 (8)	60.91 (5)	62.01 (4)	63.03 (3)	71.34 (1)	69.17 (2)
	75	58.28 (6)	58.18 (7)	36.97 (9)	36.97 (9)	52.63 (8)	64.78 (3)	62.76 (5)	63.03 (4)	74.19 (1)	72.16 (2)
	100	59.67 (6)	59.67 (6)	45.83 (9)	45.83 (9)	46.01 (8)	62.28 (5)	65.69 (3)	63.03 (4)	74.08 (1)	72.88 (2)
bre	5	68.72 (2)	64.26 (9)	64.48 (8)	66.04 (7)	66.87 (5)	62.76 (10)	66.41 (6)	71.10 (1)	68.46 (3)	67.13 (4)
	10	68.73 (2)	63.06 (10)	67.28 (6)	65.25 (7)	63.96 (9)	65.22 (8)	68.34 (5)	70.75 (1)	68.61 (3)	68.47 (4)
	25	69.32 (5)	68.84 (7)	65.10 (10)	65.33 (9)	72.82 (1)	66.30 (8)	68.94 (6)	70.63 (4)	71.36 (2)	71.12 (3)
	50	73.08 (2)	72.56 (3)	68.12 (8)	68.11 (9)	74.25 (1)	66.07 (10)	68.95 (7)	71.38 (5)	71.96 (4)	70.29 (6)
	75	72.21 (3)	72.80 (2)	67.50 (8)	66.18 (9)	73.28 (1)	63.32 (10)	69.31 (7)	71.48 (5)	71.94 (4)	69.78 (6)
	100	72.56 (3)	72.56 (3)	68.95 (7)	66.90 (9)	73.53 (1)	64.23 (10)	67.51 (8)	70.78 (5)	72.67 (2)	70.16 (6)
car	5	77.84 (3)	76.66 (5)	73.32 (7)	74.21 (6)	62.94 (10)	77.60 (4)	71.22 (8)	70.31 (9)	79.57 (2)	79.92 (1)
	10	80.19 (4)	79.01 (5)	75.69 (6)	75.02 (7)	62.83 (10)	83.31 (2)	73.92 (8)	70.91 (9)	82.97 (3)	84.22 (1)
	25	85.21 (4)	84.43 (5)	81.64 (6)	80.29 (8)	63.48 (10)	91.53 (2)	80.50 (7)	76.31 (9)	90.22 (3)	92.57 (1)
	50	89.74 (4)	88.31 (5)	87.11 (6)	85.96 (7)	61.96 (10)	95.91 (2)	85.49 (8)	84.94 (9)	94.48 (3)	96.08 (1)
	75	91.15 (5)	90.66 (6)	91.71 (4)	89.91 (8)	62.83 (10)	97.49 (2)	85.73 (9)	90.08 (7)	96.66 (3)	97.76 (1)
	100	91.40 (7)	91.40 (7)	93.77 (4)	93.73 (5)	62.65 (10)	98.30 (1)	85.15 (9)	93.60 (6)	97.98 (3)	98.19 (2)
eco	5	59.54 (5)	51.41 (8)	43.07 (9)	43.07 (9)	58.64 (6)	61.52 (4)	57.23 (7)	62.55 (3)	69.38 (1)	68.07 (2)
	10	65.05 (6)	59.52 (8)	43.07 (9)	43.07 (9)	66.85 (5)	65.04 (7)	67.36 (4)	70.78 (3)	75.41 (1)	73.53 (2)
	25	71.96 (6)	67.85 (8)	43.07 (9)	43.07 (9)	70.87 (7)	73.09 (5)	73.29 (4)	77.91 (2)	79.54 (1)	77.61 (3)
	50	74.77 (6)	71.19 (7)	43.07 (9)	43.07 (9)	69.47 (8)	76.01 (5)	79.00 (4)	82.01 (2)	83.02 (1)	81.51 (3)
	75	75.97 (7)	76.08 (6)	43.07 (9)	43.07 (9)	64.47 (8)	78.22 (5)	81.61 (4)	84.42 (2)	85.34 (1)	81.93 (3)
	100	74.00 (6)	74.00 (6)	44.68 (9)	44.68 (9)	60.54 (8)	78.01 (5)	81.71 (4)	84.92 (2)	85.83 (1)	82.84 (3)
hay	5	44.38 (8)	47.29 (3)	41.67 (10)	45.42 (7)	47.29 (3)	50.21 (1)	46.04 (6)	42.71 (9)	47.08 (5)	49.58 (2)
	10	54.58 (4)	52.50 (6)	46.67 (9)	51.46 (7)	53.33 (5)	60.21 (1)	44.38 (10)	48.12 (8)	56.46 (3)	59.17 (2)
	25	64.79 (4)	60.21 (5)	54.37 (9)	56.46 (8)	57.71 (6)	66.67 (3)	50.21 (10)	57.29 (7)	67.29 (1)	66.88 (2)
	50	68.33 (5)	67.92 (6)	62.71 (9)	63.75 (8)	65.62 (7)	79.38 (1)	56.25 (10)	70.42 (4)	78.33 (2)	78.33 (2)
	75	73.54 (5)	71.46 (6)	68.96 (8)	69.17 (7)	67.71 (9)	81.46 (1)	63.33 (10)	78.33 (4)	81.04 (3)	81.46 (1)
	100	77.50 (5)	77.50 (5)	69.17 (8)	69.58 (7)	67.50 (9)	81.67 (4)	66.67 (10)	82.29 (1)	81.88 (2)	81.88 (2)
hea	5	73.95 (4)	71.60 (5)	55.56 (8)	55.56 (8)	75.43 (2)	68.15 (6)	59.26 (7)	55.31 (10)	77.78 (1)	75.43 (2)
	10	71.98 (5)	73.95 (4)	55.56 (9)	55.56 (9)	79.01 (1)	68.64 (6)	63.95 (7)	55.93 (8)	78.64 (2)	74.69 (3)
	25	78.89 (4)	75.93 (5)	55.56 (9)	55.56 (9)	82.59 (1)	71.73 (6)	65.19 (7)	59.51 (8)	81.73 (2)	79.26 (3)
	50	78.02 (5)	78.40 (4)	55.56 (9)	55.56 (9)	83.58 (1)	72.72 (6)	67.53 (7)	65.68 (8)	81.85 (2)	80.62 (3)
	75	80.99 (3)	80.25 (4)	55.56 (9)	55.56 (9)	83.70 (1)	72.22 (6)	66.67 (8)	68.02 (7)	81.48 (2)	79.51 (5)
	100	80.86 (4)	80.86 (4)	57.04 (9)	57.04 (9)	84.32 (2)	75.93 (6)	67.65 (8)	69.14 (7)	84.44 (1)	82.10 (3)

Table V.9: Test accuracy and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). The classifiers are grouped according to whether they are online or offline, and supervised (SL) or semi-supervised (SSL). This table contains half of the datasets included in the experimentation, the information regarding the first half of datasets is in Table V.8.

		Online SSL	Online SL	Offline SSL		Offline SL					
Data	%Lab	*PAST*	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
mam	5	77.73 (1)	77.31 (2)	53.69 (10)	54.49 (9)	72.84 (5)	69.10 (7)	70.83 (6)	67.98 (8)	73.67 (3)	72.88 (4)
	10	78.70 (1)	78.21 (2)	54.42 (10)	58.44 (9)	76.24 (3)	70.41 (7)	72.63 (6)	68.85 (8)	73.43 (4)	72.91 (5)
	25	79.64 (1)	78.63 (2)	57.72 (10)	65.55 (9)	78.56 (3)	70.03 (8)	75.10 (4)	71.38 (7)	74.99 (5)	72.88 (6)
	50	79.22 (1)	78.77 (2)	68.57 (10)	70.02 (9)	77.80 (3)	71.90 (8)	76.10 (4)	75.44 (6)	75.79 (5)	75.06 (7)
	75	79.02 (1)	78.42 (2)	73.84 (8)	73.32 (9)	77.87 (3)	71.80 (10)	76.00 (5)	76.90 (4)	75.51 (6)	75.37 (7)
	100	78.22 (1)	78.22 (1)	77.35 (5)	76.38 (9)	77.90 (3)	74.44 (10)	76.87 (8)	77.49 (4)	77.25 (6)	77.04 (7)
pag	5	92.12 (5)	90.55 (6)	89.78 (7)	89.78 (7)	87.08 (10)	93.98 (3)	92.42 (4)	89.77 (9)	95.13 (1)	95.06 (2)
	10	92.56 (5)	91.87 (6)	89.78 (8)	89.78 (8)	90.02 (7)	94.77 (3)	93.45 (4)	89.78 (10)	95.75 (1)	95.69 (2)
	25	93.05 (5)	92.78 (6)	89.78 (8)	89.78 (8)	86.21 (10)	95.62 (3)	94.51 (4)	89.90 (7)	96.63 (1)	96.49 (2)
	50	93.28 (5)	92.86 (6)	89.78 (8)	89.78 (8)	89.61 (10)	96.24 (3)	95.14 (4)	90.09 (7)	96.92 (2)	96.93 (1)
	75	93.76 (5)	93.54 (6)	89.78 (8)	89.78 (8)	86.96 (10)	96.27 (3)	95.69 (4)	90.24 (7)	97.34 (1)	97.00 (2)
	100	93.87 (5)	93.87 (5)	92.08 (7)	92.07 (8)	88.63 (10)	96.24 (3)	95.77 (4)	90.47 (9)	97.37 (1)	97.13 (2)
pim	5	68.36 (5)	66.41 (8)	65.11 (9)	65.11 (9)	71.62 (3)	67.37 (6)	69.06 (4)	67.37 (7)	71.97 (1)	71.71 (2)
	10	71.31 (4)	69.14 (6)	65.11 (9)	65.11 (9)	72.88 (3)	67.62 (7)	67.44 (8)	69.27 (5)	73.83 (1)	73.13 (2)
	25	72.22 (6)	72.39 (5)	65.11 (9)	65.11 (9)	74.66 (1)	67.71 (8)	70.74 (7)	73.39 (4)	74.35 (2)	74.30 (3)
	50	73.43 (6)	74.09 (5)	65.11 (9)	65.11 (9)	75.18 (2)	67.57 (8)	70.31 (7)	75.00 (3)	75.39 (1)	74.87 (4)
	75	76.95 (1)	76.31 (2)	65.11 (9)	65.11 (9)	75.40 (5)	69.18 (8)	69.19 (7)	75.82 (3)	75.52 (4)	75.35 (6)
	100	75.78 (2)	75.78 (2)	65.76 (9)	65.76 (9)	75.66 (4)	69.88 (8)	69.96 (7)	76.04 (1)	75.65 (5)	75.48 (6)
sah	5	63.58 (7)	57.07 (10)	65.37 (2)	65.37 (2)	64.08 (5)	58.79 (9)	61.32 (8)	65.44 (1)	64.85 (4)	63.85 (6)
	10	65.61 (5)	61.55 (10)	65.37 (6)	65.37 (6)	68.40 (1)	63.29 (9)	63.74 (8)	65.87 (4)	66.30 (2)	66.04 (3)
	25	65.51 (4)	64.52 (8)	65.37 (6)	65.37 (6)	69.98 (1)	63.71 (10)	64.07 (9)	65.51 (5)	68.83 (2)	68.53 (3)
	50	67.25 (2)	65.88 (6)	65.37 (7)	65.37 (7)	70.99 (1)	61.47 (10)	62.50 (9)	66.30 (5)	66.37 (4)	67.18 (3)
	75	68.26 (2)	67.54 (3)	65.37 (7)	65.37 (7)	71.43 (1)	61.64 (9)	59.60 (10)	65.51 (6)	67.39 (4)	66.17 (5)
	100	67.10 (4)	67.10 (4)	64.36 (7)	64.36 (7)	71.35 (1)	61.62 (9)	58.07 (10)	66.30 (6)	67.17 (3)	68.18 (2)
skn	5	97.27 (6)	95.90 (7)	N/A	N/A	92.41 (8)	99.64 (4)	99.85 (1)	99.38 (5)	99.81 (2)	99.72 (3)
	10	97.41 (6)	95.44 (7)	N/A	N/A	92.40 (8)	99.75 (4)	99.89 (1)	99.45 (5)	99.87 (2)	99.82 (3)
	25	97.13 (6)	95.21 (7)	N/A	N/A	92.39 (8)	99.86 (4)	99.93 (1)	99.50 (5)	99.92 (2)	99.88 (3)
	50	96.43 (6)	95.28 (7)	N/A	N/A	92.39 (8)	99.90 (4)	99.95 (1)	99.73 (5)	99.94 (2)	99.92 (3)
	75	95.39 (6)	95.06 (7)	N/A	N/A	92.39 (8)	99.92 (4)	99.95 (2)	99.83 (5)	99.95 (1)	99.94 (3)
	100	94.97 (6)	94.97 (6)	N/A	N/A	92.39 (8)	99.93 (4)	99.96 (1)	99.83 (5)	99.96 (2)	99.94 (3)
tae	5	38.89 (2)	36.65 (4)	32.46 (9)	32.46 (9)	39.76 (1)	35.79 (5)	34.04 (8)	35.15 (6)	37.81 (3)	34.47 (7)
	10	39.19 (5)	38.22 (6)	32.46 (9)	32.46 (9)	44.01 (1)	40.25 (3)	35.63 (8)	36.89 (7)	39.63 (4)	41.15 (2)
	25	49.56 (2)	45.89 (5)	32.46 (9)	32.46 (9)	44.15 (6)	47.44 (4)	40.18 (7)	36.18 (8)	50.78 (1)	47.92 (3)
	50	53.29 (3)	46.04 (6)	32.46 (9)	32.46 (9)	50.14 (5)	51.86 (4)	41.83 (7)	38.18 (8)	54.40 (1)	54.10 (2)
	75	45.51 (6)	47.11 (5)	34.68 (9)	34.46 (10)	49.04 (4)	56.82 (2)	40.67 (7)	37.36 (8)	56.21 (3)	59.04 (1)
	100	50.38 (7)	50.38 (7)	58.17 (4)	57.28 (5)	51.00 (6)	63.67 (3)	40.92 (9)	36.03 (10)	63.86 (2)	64.76 (1)
tic	5	69.76 (2)	60.57 (10)	67.50 (4)	75.40 (1)	61.96 (9)	62.81 (8)	63.61 (7)	65.80 (6)	67.71 (3)	66.80 (5)
	10	76.51 (2)	69.48 (8)	71.78 (5)	78.67 (1)	64.24 (10)	71.39 (6)	68.85 (9)	70.11 (7)	75.51 (4)	76.03 (3)
	25	83.15 (3)	75.50 (8)	79.68 (5)	81.14 (4)	67.99 (10)	78.57 (6)	72.41 (9)	75.51 (7)	83.75 (2)	85.56 (1)
	50	89.18 (2)	81.14 (8)	82.53 (6)	81.98 (7)	70.57 (10)	84.45 (4)	80.86 (9)	82.91 (5)	89.11 (3)	91.82 (1)
	75	92.48 (3)	87.79 (5)	83.26 (7)	82.74 (8)	70.77 (10)	88.62 (4)	80.55 (9)	87.09 (6)	93.18 (2)	94.43 (1)
	100	95.75 (2)	95.75 (2)	84.03 (7)	83.33 (8)	71.19 (10)	87.51 (6)	79.48 (9)	89.53 (5)	94.92 (4)	96.14 (1)
tit	5	77.26 (1)	74.19 (4)	61.62 (10)	61.65 (9)	73.85 (5)	70.60 (6)	64.30 (7)	63.86 (8)	74.60 (2)	74.30 (3)
	10	77.78 (1)	76.76 (2)	61.62 (9)	60.91 (10)	74.45 (5)	73.51 (6)	62.70 (8)	65.81 (7)	76.76 (3)	76.20 (4)
	25	78.49 (3)	77.25 (4)	61.62 (9)	61.24 (10)	76.69 (5)	74.14 (6)	65.62 (8)	67.26 (7)	79.31 (1)	78.60 (2)
	50	78.90 (3)	78.37 (4)	61.62 (9)	61.36 (10)	77.78 (5)	77.22 (6)	68.24 (8)	68.35 (7)	81.07 (1)	80.96 (2)
	75	78.07 (4)	78.34 (3)	61.62 (9)	61.28 (10)	77.52 (5)	77.07 (6)	68.35 (8)	68.35 (7)	82.64 (1)	81.82 (2)
	100	79.16 (3)	79.16 (3)	62.40 (9)	62.40 (9)	78.23 (5)	78.16 (6)	68.13 (8)	68.76 (7)	82.98 (1)	82.98 (2)
veh	5	49.80 (4)	40.70 (7)	25.77 (9)	25.77 (9)	46.89 (6)	52.49 (3)	47.00 (5)	38.60 (8)	59.47 (1)	58.79 (2)
	10	56.88 (4)	50.00 (6)	25.77 (9)	25.77 (9)	47.00 (7)	57.80 (3)	53.27 (5)	39.87 (8)	66.71 (2)	67.18 (1)
	25	62.14 (4)	55.79 (6)	25.77 (9)	25.77 (9)	45.15 (7)	65.84 (3)	56.70 (5)	40.86 (8)	72.70 (1)	71.76 (2)
	50	62.70 (4)	61.43 (6)	25.77 (9)	25.77 (9)	45.11 (7)	68.48 (3)	61.47 (5)	44.79 (8)	73.25 (2)	73.56 (1)
	75	62.96 (5)	62.33 (6)	25.77 (9)	25.77 (9)	45.31 (8)	69.39 (3)	64.15 (4)	47.91 (7)	74.86 (1)	73.95 (2)
	100	63.43 (5)	63.43 (5)	25.93 (9)	25.93 (9)	45.19 (8)	71.35 (3)	65.10 (4)	49.17 (7)	74.75 (2)	75.10 (1)
wdb	5	91.27 (4)	84.60 (8)	62.74 (9)	62.74 (9)	93.27 (1)	87.70 (7)	88.34 (5)	87.94 (6)	91.39 (3)	91.51 (2)
	10	92.44 (3)	88.04 (8)	62.74 (9)	62.74 (9)	93.44 (1)	90.10 (6)	91.63 (5)	88.99 (7)	93.27 (2)	92.09 (4)
	25	91.86 (5)	91.16 (7)	62.74 (9)	62.74 (9)	93.73 (3)	91.51 (6)	92.33 (4)	90.28 (8)	94.15 (1)	93.85 (2)
	50	93.62 (4)	91.74 (7)	62.74 (9)	62.74 (9)	93.85 (3)	92.44 (6)	92.45 (5)	90.51 (8)	95.32 (1)	95.03 (2)
	75	92.86 (4)	91.92 (7)	62.74 (9)	62.74 (9)	94.03 (3)	92.56 (6)	92.57 (5)	90.86 (8)	95.90 (1)	95.20 (2)
	100	93.15 (4)	93.15 (4)	62.74 (9)	62.74 (9)	94.14 (3)	93.03 (6)	92.56 (7)	91.68 (8)	95.90 (2)	95.90 (1)
yea	5	45.06 (4)	37.87 (9)	31.92 (10)	38.92 (8)	44.30 (6)	44.70 (5)	48.09 (3)	43.41 (7)	51.14 (2)	51.84 (1)
	10	46.58 (5)	41.35 (7)	32.14 (10)	40.12 (8)	38.81 (9)	44.81 (6)	49.17 (4)	52.13 (3)	53.84 (1)	52.90 (2)
	25	50.33 (5)	48.26 (6)	36.81 (9)	41.51 (8)	34.64 (10)	47.86 (7)	51.89 (4)	57.37 (1)	57.17 (2)	56.38 (3)
	50	52.19 (5)	50.98 (6)	41.49 (9)	43.15 (8)	23.72 (10)	49.19 (7)	53.64 (4)	59.72 (2)	59.88 (1)	57.55 (3)
	75	51.61 (6)	53.47 (5)	44.43 (9)	44.50 (8)	18.24 (10)	51.01 (7)	54.31 (4)	60.01 (2)	60.94 (1)	59.25 (3)
	100	52.82 (5)	52.82 (5)	47.80 (8)	46.31 (9)	14.74 (10)	52.40 (7)	55.59 (4)	60.30 (3)	62.08 (1)	60.72 (2)

Table V.10: G-mean and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). The classifiers are grouped according to whether they are online or offline, and supervised (SL) or semi-supervised (SSL). This table contains half of the datasets included in the experimentation, the information regarding the remaining datasets is in Table V.11.

		Online SSL	Online SL	Offline SSL		Offline SL					
Data	%Lab	*PAST*	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
app	5	47.31 (2)	41.83 (3)	23.31 (7)	41.36 (4)	0.00 (9)	47.63 (1)	0.00 (9)	19.80 (8)	35.99 (6)	39.41 (5)
	10	59.34 (4)	61.32 (2)	37.72 (8)	55.69 (5)	22.39 (10)	59.95 (3)	52.67 (7)	35.18 (9)	55.47 (6)	63.60 (1)
	25	70.91 (1)	68.48 (3)	45.12 (9)	50.25 (8)	69.83 (2)	63.89 (6)	55.65 (7)	43.92 (10)	67.34 (4)	66.34 (5)
	50	68.65 (1)	65.84 (5)	61.44 (9)	66.01 (4)	68.08 (2)	62.58 (7)	59.62 (10)	66.82 (3)	64.47 (6)	61.82 (8)
	75	67.05 (4)	68.72 (2)	60.20 (8)	62.59 (6)	72.33 (1)	56.90 (10)	59.58 (9)	61.17 (7)	68.29 (3)	63.92 (5)
	100	63.53 (6)	63.53 (6)	63.56 (4)	63.56 (4)	72.22 (1)	60.07 (9)	62.36 (8)	59.84 (10)	65.16 (2)	64.39 (3)
aut	5	1.41 (4)	0.00 (5)	0.00 (5)	0.00 (5)	0.00 (5)	1.83 (3)	0.00 (5)	0.00 (5)	5.10 (1)	2.13 (2)
	10	2.13 (3)	1.22 (5)	0.00 (6)	0.00 (6)	0.00 (6)	8.49 (2)	0.00 (6)	0.00 (6)	2.03 (4)	8.77 (1)
	25	7.00 (4)	6.88 (5)	0.00 (7)	0.00 (7)	3.92 (6)	19.14 (1)	0.00 (7)	0.00 (7)	14.48 (2)	11.49 (3)
	50	17.13 (6)	21.49 (4)	0.00 (7)	0.00 (7)	17.79 (5)	30.84 (3)	0.00 (7)	0.00 (7)	50.82 (1)	49.91 (2)
	75	33.80 (4)	19.39 (5)	0.00 (8)	0.00 (8)	12.51 (6)	50.43 (3)	1.94 (7)	0.00 (8)	54.32 (2)	57.86 (1)
	100	28.97 (4)	28.97 (4)	0.00 (7)	0.00 (7)	5.55 (6)	58.38 (3)	0.00 (7)	0.00 (7)	76.16 (1)	73.01 (2)
ban	5	82.58 (8)	70.76 (9)	82.90 (7)	88.43 (1)	51.40 (10)	83.36 (6)	86.47 (3)	87.72 (2)	86.16 (4)	85.50 (5)
	10	84.83 (8)	73.41 (9)	86.71 (6)	88.73 (1)	51.77 (10)	85.03 (7)	87.84 (3)	88.43 (2)	87.33 (4)	86.84 (5)
	25	86.45 (7)	75.48 (9)	88.92 (3)	89.59 (1)	51.73 (10)	86.20 (8)	88.32 (5)	89.32 (2)	88.46 (4)	88.06 (6)
	50	87.04 (7)	81.56 (9)	89.43 (3)	89.50 (2)	52.24 (10)	86.48 (8)	88.16 (5)	89.53 (1)	88.67 (4)	88.10 (6)
	75	87.29 (7)	86.52 (9)	89.56 (3)	89.57 (2)	51.64 (10)	86.78 (8)	88.14 (6)	89.62 (1)	88.80 (4)	88.30 (5)
	100	87.25 (7)	87.25 (7)	89.71 (2)	89.61 (3)	51.67 (10)	86.99 (9)	88.06 (6)	89.74 (1)	89.05 (4)	88.92 (5)
bnd	5	46.53 (6)	50.36 (2)	0.00 (8)	0.00 (8)	44.82 (7)	52.63 (1)	48.81 (4)	0.00 (8)	47.54 (5)	50.05 (3)
	10	51.10 (5)	51.15 (4)	0.00 (8)	0.00 (8)	42.14 (7)	51.83 (2)	46.61 (6)	0.00 (8)	51.48 (3)	53.63 (1)
	25	54.34 (4)	52.73 (5)	0.00 (8)	0.00 (8)	45.18 (7)	54.73 (3)	46.60 (6)	0.00 (8)	55.34 (2)	56.31 (1)
	50	56.15 (4)	53.99 (5)	0.00 (8)	0.00 (8)	46.71 (7)	57.59 (3)	53.99 (6)	0.00 (8)	62.15 (2)	62.33 (1)
	75	56.16 (5)	56.31 (4)	0.00 (8)	0.00 (8)	46.28 (7)	61.15 (3)	54.04 (6)	0.00 (8)	66.54 (1)	64.79 (2)
	100	58.57 (3)	58.57 (3)	36.59 (7)	36.59 (7)	36.18 (9)	57.67 (5)	56.42 (6)	0.00 (10)	65.88 (2)	66.32 (1)
bre	5	54.39 (3)	54.54 (2)	37.35 (9)	47.83 (6)	54.85 (1)	50.57 (4)	43.32 (8)	24.41 (10)	45.49 (7)	50.28 (5)
	10	49.11 (4)	52.79 (1)	44.12 (7)	44.73 (6)	52.51 (2)	51.53 (3)	39.70 (9)	23.30 (10)	42.13 (8)	47.33 (5)
	25	56.70 (3)	59.29 (2)	43.93 (9)	50.63 (6)	63.36 (1)	53.35 (5)	45.43 (8)	30.67 (10)	47.61 (7)	53.93 (4)
	50	64.82 (1)	63.85 (3)	48.95 (8)	51.70 (7)	64.80 (2)	54.21 (5)	47.86 (9)	41.91 (10)	54.62 (4)	53.65 (6)
	75	63.25 (3)	63.70 (1)	52.78 (6)	53.66 (5)	63.28 (2)	51.33 (8)	48.39 (9)	45.15 (10)	55.36 (4)	52.48 (7)
	100	64.79 (1)	64.79 (1)	53.84 (6)	53.26 (7)	63.92 (3)	52.52 (8)	48.55 (9)	47.25 (10)	55.24 (4)	54.71 (5)
car	5	22.25 (4)	17.47 (8)	23.24 (3)	17.64 (7)	18.18 (6)	44.18 (1)	12.91 (9)	0.00 (10)	21.71 (5)	38.02 (2)
	10	24.99 (6)	28.95 (5)	22.83 (7)	15.24 (9)	17.30 (8)	59.92 (1)	30.58 (4)	0.00 (10)	43.00 (3)	54.62 (2)
	25	43.46 (5)	35.97 (7)	38.84 (6)	29.96 (8)	4.54 (9)	80.28 (1)	44.97 (4)	1.61 (10)	67.13 (3)	78.16 (2)
	50	61.60 (4)	45.93 (8)	58.32 (5)	52.34 (7)	3.02 (10)	90.12 (1)	54.65 (6)	10.18 (9)	84.91 (3)	89.89 (2)
	75	69.86 (5)	69.44 (6)	74.07 (4)	65.84 (7)	0.00 (10)	94.14 (2)	53.71 (8)	49.00 (9)	91.41 (3)	95.78 (1)
	100	70.47 (7)	70.47 (7)	80.75 (4)	80.73 (5)	0.00 (10)	95.41 (2)	52.12 (9)	75.58 (6)	94.54 (3)	96.89 (1)
eco	5	3.22 (5)	3.76 (4)	0.00 (6)	0.00 (6)	0.00 (6)	5.14 (2)	0.00 (6)	0.00 (6)	9.69 (1)	5.03 (3)
	10	7.16 (4)	5.60 (6)	0.00 (9)	0.00 (9)	4.59 (7)	16.68 (3)	6.62 (5)	3.58 (8)	17.53 (2)	20.68 (1)
	25	30.09 (4)	16.30 (6)	0.00 (9)	0.00 (9)	24.56 (5)	30.82 (3)	13.08 (7)	12.82 (8)	31.31 (2)	31.66 (1)
	50	23.56 (6)	16.35 (7)	0.00 (9)	0.00 (9)	10.06 (8)	33.29 (5)	62.12 (1)	57.94 (2)	51.20 (3)	50.62 (4)
	75	31.22 (7)	33.29 (6)	0.00 (9)	0.00 (9)	3.94 (8)	41.65 (5)	65.74 (2)	71.68 (1)	51.96 (3)	44.66 (4)
	100	25.99 (6)	25.99 (6)	0.00 (8)	0.00 (8)	0.00 (8)	32.51 (5)	65.00 (2)	72.45 (1)	59.43 (3)	52.59 (4)
hay	5	14.80 (5)	17.34 (3)	13.18 (7)	27.81 (2)	1.52 (8)	31.12 (1)	0.00 (10)	1.52 (8)	13.35 (6)	15.65 (4)
	10	40.29 (4)	38.19 (6)	20.07 (9)	38.99 (5)	41.91 (3)	48.69 (1)	16.41 (10)	21.48 (8)	38.00 (7)	45.97 (2)
	25	58.03 (4)	54.36 (6)	34.28 (9)	45.80 (8)	54.67 (5)	61.91 (1)	29.36 (10)	49.45 (7)	61.88 (2)	59.02 (3)
	50	66.41 (6)	67.59 (5)	49.24 (9)	52.36 (8)	65.37 (7)	78.90 (1)	44.01 (10)	68.97 (4)	78.46 (2)	77.86 (3)
	75	72.81 (5)	70.59 (6)	57.37 (9)	62.46 (8)	68.27 (7)	82.42 (2)	52.39 (10)	78.54 (4)	81.98 (3)	82.67 (1)
	100	77.75 (5)	77.75 (5)	59.35 (9)	60.41 (8)	68.15 (7)	82.79 (2)	55.71 (10)	82.44 (4)	82.99 (1)	82.77 (3)
hea	5	71.09 (4)	68.96 (5)	0.00 (8)	0.00 (8)	73.06 (2)	66.14 (6)	51.42 (7)	0.00 (8)	73.87 (1)	72.61 (3)
	10	70.99 (5)	71.89 (4)	0.00 (9)	0.00 (9)	77.90 (1)	67.64 (6)	61.46 (7)	4.49 (8)	76.39 (2)	73.00 (3)
	25	78.18 (3)	75.15 (5)	0.00 (9)	0.00 (9)	81.69 (1)	70.36 (6)	62.53 (7)	30.25 (8)	80.71 (2)	78.05 (4)
	50	77.63 (4)	76.55 (5)	0.00 (9)	0.00 (9)	82.68 (1)	71.59 (6)	64.91 (7)	56.48 (8)	80.56 (2)	79.20 (3)
	75	80.16 (3)	78.90 (4)	0.00 (9)	0.00 (9)	83.10 (1)	71.39 (6)	64.57 (7)	62.01 (8)	80.28 (2)	78.36 (5)
	100	80.30 (4)	80.30 (4)	10.42 (9)	10.42 (9)	83.59 (2)	75.02 (6)	65.59 (7)	63.50 (8)	83.75 (1)	81.38 (3)

Table V.11: G-mean and ranking positions for each dataset (Data) and percentage of labeled samples (%Lab). The classifiers are grouped according to whether they are online or offline, and supervised (SL) or semi-supervised (SSL). This table contains half of the datasets included in the experimentation, the information regarding the first half of datasets is in Table V.10.

		Online SSL	Online SL	Offline SSL		Offline SL					
Data	%Lab	*PAST*	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
mam	5	77.71 (1)	77.21 (2)	0.00 (10)	4.32 (9)	71.28 (5)	68.49 (7)	70.03 (6)	63.76 (8)	73.05 (3)	72.19 (4)
	10	78.75 (1)	78.22 (2)	2.50 (10)	19.07 (9)	76.07 (3)	69.83 (7)	72.12 (6)	67.49 (8)	72.87 (4)	72.39 (5)
	25	79.71 (1)	78.72 (2)	16.43 (10)	45.58 (9)	78.60 (3)	69.30 (8)	74.57 (5)	70.69 (7)	74.66 (4)	72.38 (6)
	50	79.32 (1)	78.84 (2)	52.32 (10)	59.09 (9)	77.83 (3)	71.12 (8)	75.71 (4)	75.01 (6)	75.53 (5)	74.78 (7)
	75	79.12 (1)	78.50 (2)	68.42 (10)	69.64 (9)	77.91 (3)	71.00 (8)	75.80 (5)	76.68 (4)	75.17 (6)	75.08 (7)
	100	78.32 (1)	78.32 (1)	77.32 (5)	76.35 (9)	77.95 (3)	74.01 (10)	76.61 (8)	77.48 (4)	77.10 (6)	76.87 (7)
pag	5	0.00 (5)	1.74 (4)	0.00 (5)	0.00 (5)	0.00 (5)	28.44 (1)	0.00 (5)	0.00 (5)	12.19 (3)	27.02 (2)
	10	4.50 (6)	1.98 (7)	0.00 (8)	0.00 (8)	46.10 (1)	41.83 (2)	6.09 (5)	0.00 (8)	36.94 (4)	41.67 (3)
	25	31.83 (5)	9.07 (7)	0.00 (8)	0.00 (8)	56.04 (4)	72.37 (2)	27.29 (6)	0.00 (8)	62.34 (3)	74.19 (1)
	50	47.95 (6)	18.63 (7)	0.00 (8)	0.00 (8)	58.00 (4)	76.50 (2)	48.84 (5)	0.00 (8)	75.79 (3)	79.86 (1)
	75	61.72 (5)	41.16 (7)	0.00 (8)	0.00 (8)	52.42 (6)	75.01 (3)	66.60 (4)	0.00 (8)	79.31 (1)	76.59 (2)
	100	58.20 (5)	58.20 (5)	0.00 (8)	0.00 (8)	53.21 (7)	78.95 (3)	68.15 (4)	0.00 (8)	82.24 (1)	80.64 (2)
pim	5	60.61 (6)	60.62 (5)	0.00 (9)	0.00 (9)	65.24 (1)	62.70 (3)	59.44 (7)	29.80 (8)	62.28 (4)	64.36 (2)
	10	67.47 (1)	63.16 (6)	0.00 (9)	0.00 (9)	67.20 (2)	63.76 (5)	60.00 (7)	50.23 (8)	64.81 (4)	66.09 (3)
	25	69.82 (1)	69.24 (3)	0.00 (9)	0.00 (9)	69.78 (2)	63.74 (7)	64.94 (6)	61.13 (8)	67.48 (5)	68.70 (4)
	50	72.24 (1)	71.43 (2)	0.00 (9)	0.00 (9)	70.17 (3)	64.11 (7)	64.15 (6)	63.63 (8)	68.59 (5)	68.71 (4)
	75	74.92 (1)	73.21 (2)	0.00 (9)	0.00 (9)	70.55 (3)	65.21 (7)	63.28 (8)	65.70 (6)	69.04 (5)	69.48 (4)
	100	73.75 (1)	73.75 (1)	16.77 (9)	16.77 (9)	70.84 (3)	65.60 (7)	64.87 (8)	66.06 (6)	69.28 (5)	70.53 (4)
sah	5	53.97 (2)	51.43 (3)	0.00 (9)	0.00 (9)	55.57 (1)	50.39 (5)	50.57 (4)	0.83 (8)	48.48 (6)	48.27 (7)
	10	61.67 (2)	56.95 (3)	0.00 (9)	0.00 (9)	64.06 (1)	56.16 (4)	51.53 (6)	5.60 (8)	50.37 (7)	54.61 (5)
	25	63.65 (2)	62.45 (3)	0.00 (9)	0.00 (9)	66.89 (1)	58.00 (6)	53.08 (7)	7.23 (8)	58.49 (5)	59.81 (4)
	50	65.75 (2)	63.79 (3)	0.00 (9)	0.00 (9)	67.94 (1)	55.86 (5)	51.93 (7)	14.02 (8)	55.64 (6)	57.16 (4)
	75	67.21 (2)	66.54 (3)	0.00 (9)	0.00 (9)	68.51 (1)	55.31 (6)	48.73 (7)	14.05 (8)	55.79 (4)	55.50 (5)
	100	66.32 (2)	66.32 (2)	3.28 (9)	3.28 (9)	68.55 (1)	56.31 (6)	46.95 (7)	19.47 (8)	57.06 (5)	58.53 (4)
skn	5	98.20 (6)	97.08 (7)	N/A	N/A	84.67 (8)	99.49 (5)	99.89 (1)	99.61 (4)	99.82 (2)	99.65 (3)
	10	98.34 (6)	96.80 (7)	N/A	N/A	84.63 (8)	99.66 (4)	99.92 (1)	99.65 (5)	99.87 (2)	99.77 (3)
	25	98.16 (6)	96.66 (7)	N/A	N/A	84.62 (8)	99.80 (4)	99.95 (1)	99.68 (5)	99.92 (2)	99.85 (3)
	50	97.70 (6)	96.86 (7)	N/A	N/A	84.62 (8)	99.86 (4)	99.96 (1)	99.83 (5)	99.95 (2)	99.91 (3)
	75	97.01 (6)	96.65 (7)	N/A	N/A	84.61 (8)	99.90 (4)	99.96 (1)	99.89 (5)	99.96 (2)	99.92 (3)
	100	96.61 (6)	96.61 (6)	N/A	N/A	84.62 (8)	99.91 (4)	99.97 (1)	99.89 (5)	99.96 (2)	99.94 (3)
tae	5	24.56 (1)	20.19 (4)	0.00 (9)	0.00 (9)	15.64 (7)	19.17 (6)	10.48 (8)	20.05 (5)	22.34 (2)	21.82 (3)
	10	23.96 (4)	16.61 (6)	0.00 (9)	0.00 (9)	22.78 (5)	27.93 (2)	16.54 (7)	7.61 (8)	27.83 (3)	28.88 (1)
	25	36.30 (4)	31.79 (5)	0.00 (9)	0.00 (9)	16.24 (8)	38.81 (3)	28.60 (6)	18.54 (7)	44.87 (1)	41.83 (2)
	50	47.81 (3)	32.85 (6)	0.00 (9)	0.00 (9)	32.27 (7)	47.44 (4)	33.83 (5)	21.99 (8)	48.74 (2)	49.79 (1)
	75	31.35 (7)	36.74 (4)	4.40 (9)	4.20 (10)	32.79 (5)	53.23 (1)	32.33 (6)	24.12 (8)	51.00 (3)	53.07 (2)
	100	35.14 (6)	35.14 (6)	54.93 (4)	52.72 (5)	31.77 (9)	61.64 (2)	34.85 (8)	20.68 (10)	61.15 (3)	62.42 (1)
tic	5	61.85 (2)	58.57 (3)	53.45 (8)	65.23 (1)	49.24 (9)	58.23 (5)	54.23 (7)	33.20 (10)	57.75 (6)	58.42 (4)
	10	68.67 (2)	66.21 (5)	55.28 (8)	69.27 (1)	48.90 (9)	67.56 (4)	58.69 (7)	48.07 (10)	65.47 (6)	68.19 (3)
	25	77.09 (2)	73.11 (5)	68.40 (7)	72.77 (6)	47.70 (10)	75.05 (4)	61.24 (8)	59.68 (9)	76.99 (3)	79.93 (1)
	50	85.20 (2)	79.00 (5)	74.10 (6)	73.95 (7)	47.91 (10)	82.19 (4)	72.58 (9)	73.75 (8)	84.87 (3)	88.50 (1)
	75	89.79 (3)	85.48 (5)	74.73 (8)	74.73 (7)	44.58 (10)	86.40 (4)	72.98 (9)	80.34 (6)	90.42 (2)	92.01 (1)
	100	94.51 (1)	94.51 (1)	75.34 (7)	74.92 (8)	44.30 (10)	85.19 (5)	71.89 (9)	84.07 (6)	92.83 (4)	94.30 (3)
tit	5	73.66 (1)	66.42 (6)	0.00 (10)	5.62 (9)	70.46 (4)	68.28 (5)	56.43 (7)	24.98 (8)	70.65 (3)	71.24 (2)
	10	74.67 (1)	71.61 (5)	0.00 (10)	6.27 (9)	72.31 (4)	71.26 (6)	54.63 (7)	44.90 (8)	73.86 (2)	73.63 (3)
	25	76.02 (3)	72.93 (5)	0.00 (10)	9.82 (9)	74.71 (4)	72.18 (6)	60.83 (7)	53.72 (8)	76.68 (1)	76.28 (2)
	50	76.50 (3)	75.14 (6)	0.00 (10)	12.22 (9)	75.40 (5)	75.67 (4)	64.22 (7)	54.14 (8)	78.64 (1)	78.48 (2)
	75	76.55 (3)	76.27 (4)	0.00 (10)	13.18 (9)	75.13 (6)	75.59 (5)	63.87 (7)	55.31 (8)	80.55 (1)	79.71 (2)
	100	76.92 (4)	76.92 (4)	43.61 (9)	43.61 (9)	75.97 (6)	76.95 (3)	64.10 (7)	55.87 (8)	80.87 (2)	80.98 (1)
veh	5	42.86 (4)	33.81 (7)	0.00 (9)	0.00 (9)	39.31 (6)	49.36 (3)	41.22 (5)	7.66 (8)	54.07 (2)	54.72 (1)
	10	50.63 (4)	45.02 (6)	0.00 (9)	0.00 (9)	39.79 (7)	54.67 (3)	47.92 (5)	14.49 (8)	61.53 (2)	63.03 (1)
	25	56.43 (4)	49.23 (6)	0.00 (9)	0.00 (9)	33.17 (7)	63.20 (3)	51.99 (5)	19.30 (8)	68.59 (1)	67.94 (2)
	50	55.90 (5)	54.30 (6)	0.00 (9)	0.00 (9)	37.39 (7)	65.25 (3)	56.82 (4)	27.86 (8)	69.00 (2)	69.94 (1)
	75	55.78 (5)	55.52 (6)	0.00 (9)	0.00 (9)	36.32 (8)	66.54 (3)	59.32 (4)	36.61 (7)	71.24 (1)	70.34 (2)
	100	55.88 (5)	55.88 (5)	0.00 (9)	0.00 (9)	37.78 (8)	68.60 (3)	60.18 (4)	44.01 (7)	70.86 (2)	71.84 (1)
wdb	5	89.17 (4)	82.79 (7)	0.00 (9)	0.00 (9)	92.79 (1)	85.91 (6)	85.92 (5)	82.76 (8)	89.93 (3)	90.09 (2)
	10	90.45 (4)	86.18 (7)	0.00 (9)	0.00 (9)	92.71 (1)	89.40 (6)	89.69 (5)	84.71 (8)	92.25 (2)	91.40 (3)
	25	90.84 (6)	89.81 (7)	0.00 (9)	0.00 (9)	92.80 (3)	91.09 (4)	90.96 (5)	86.46 (8)	93.44 (1)	93.14 (2)
	50	92.35 (4)	90.52 (7)	0.00 (9)	0.00 (9)	92.69 (3)	91.77 (5)	91.20 (6)	87.13 (8)	94.52 (1)	94.35 (2)
	75	91.04 (6)	90.00 (7)	0.00 (9)	0.00 (9)	92.89 (3)	92.27 (4)	91.32 (5)	87.67 (8)	95.30 (1)	94.70 (2)
	100	91.57 (5)	91.57 (5)	0.00 (9)	0.00 (9)	93.01 (3)	92.84 (4)	91.38 (7)	89.07 (8)	95.27 (2)	95.45 (1)
yea	5	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	10	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	25	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)	0.00 (1)
	50	0.00 (3)	1.57 (2)	0.00 (3)	0.00 (3)	2.76 (1)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)	0.00 (3)
	75	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	3.28 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)
	100	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)	1.42 (1)	0.00 (2)	0.00 (2)	0.00 (2)	0.00 (2)

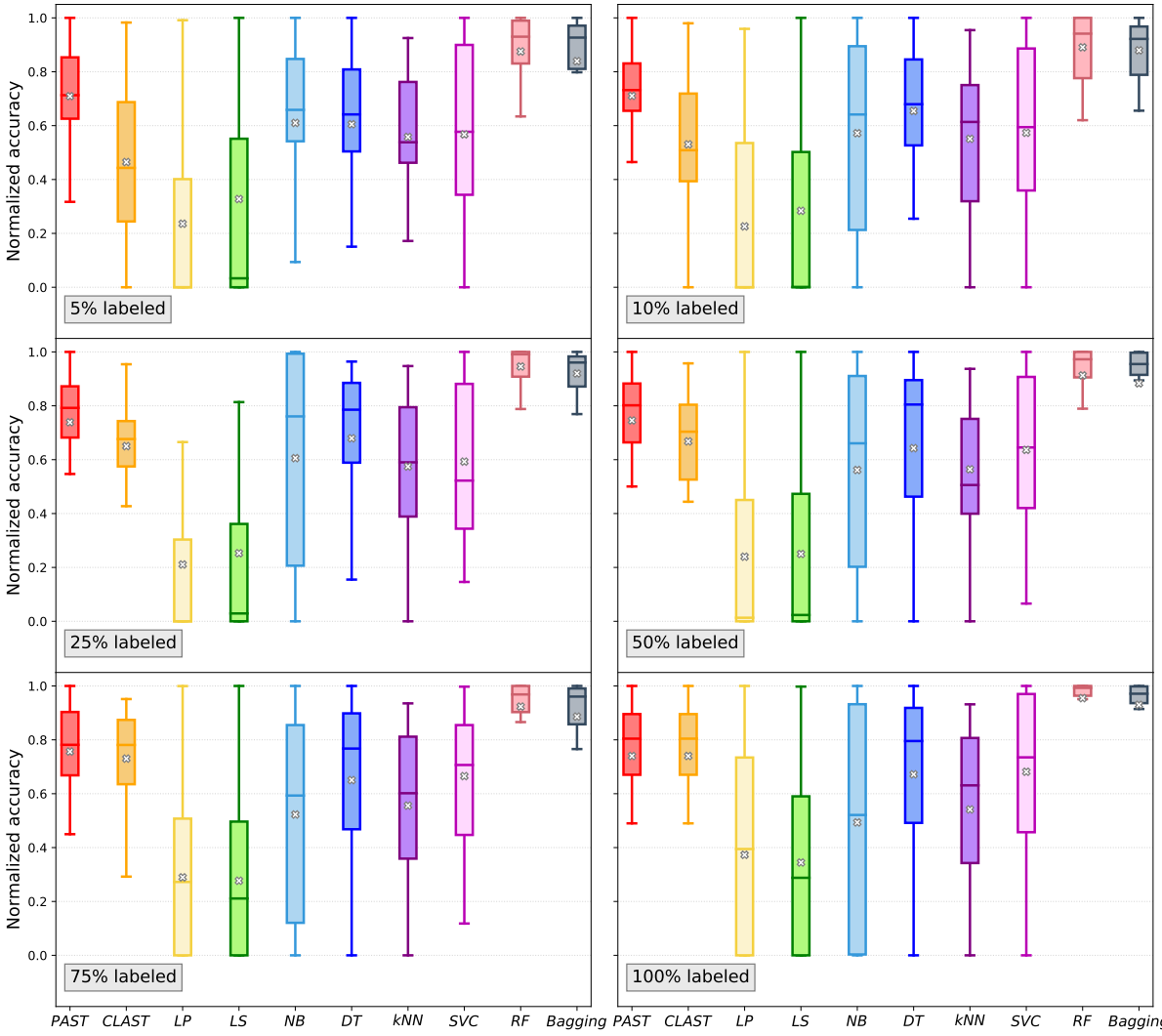


Figure V.7: Distribution of the normalized test accuracy achieved by each classifier in the different labeling scenarios.

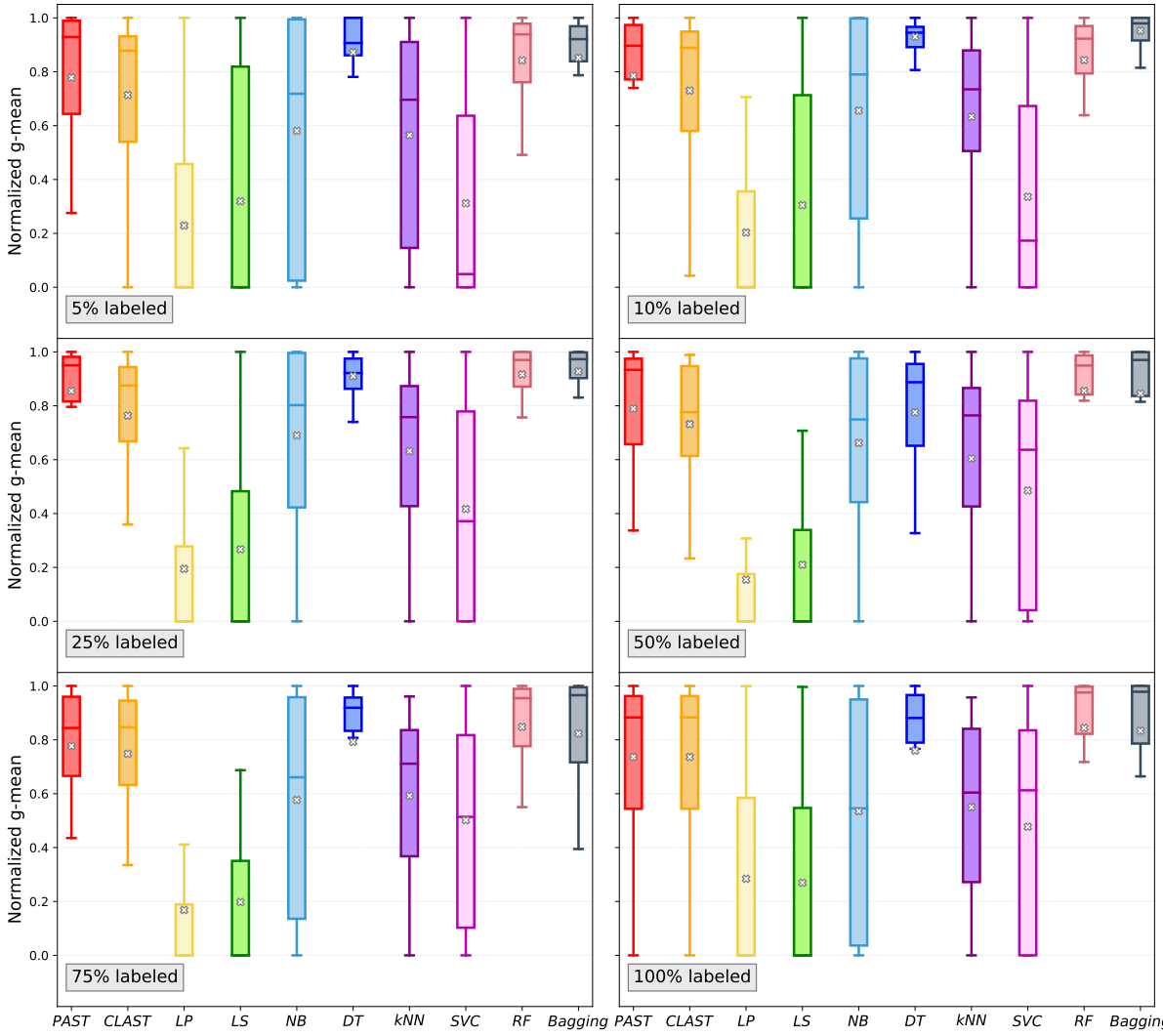


Figure V.8: Distribution of the normalized test G-mean achieved by each classifier in the different labeling scenarios.

Table V.12: Comparison of the average performance of PAST with CLAST and the different offline learners for each percentage of labeled samples (%Lab). The classifiers are grouped according to whether they are online or offline, and supervised (SL) or semi-supervised (SSL).

		Online SSL	Online SL	Offline SSL		Offline SL					
	%Lab	*PAST*	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
Accuracy	5	0.71 (4.53)	0.46 (6.84)	0.24 (7.74)	0.33 (6.79)	0.61 (5.47)	0.61 (5.63)	0.56 (5.89)	0.57 (5.84)	0.88 (2.53)	0.84 (3.05)
	10	0.71 (4.37)	0.53 (6.58)	0.23 (7.74)	0.28 (7.21)	0.57 (5.58)	0.65 (5.47)	0.55 (6.37)	0.57 (5.68)	0.89 (2.53)	0.88 (2.95)
	25	0.74 (4.74)	0.65 (6.21)	0.21 (8.05)	0.25 (7.84)	0.61 (5.37)	0.68 (5.68)	0.57 (6.11)	0.59 (5.68)	0.95 (1.95)	0.92 (2.79)
	50	0.75 (4.37)	0.67 (5.79)	0.24 (7.84)	0.25 (7.79)	0.56 (5.89)	0.64 (5.74)	0.56 (6.42)	0.64 (5.32)	0.91 (2.26)	0.88 (2.95)
	75	0.76 (4.53)	0.73 (5.11)	0.29 (7.79)	0.28 (7.95)	0.52 (6.16)	0.65 (5.58)	0.56 (6.68)	0.67 (5.21)	0.92 (2.37)	0.89 (3.11)
	100	0.74 (4.68)	0.74 (4.68)	0.37 (7.05)	0.34 (7.68)	0.49 (6.32)	0.67 (5.89)	0.54 (6.95)	0.68 (5.11)	0.95 (2.26)	0.93 (2.79)
G-mean	5	0.78 (3.58)	0.71 (4.37)	0.23 (7.32)	0.32 (6.16)	0.58 (4.95)	0.87 (3.53)	0.56 (6.11)	0.31 (7.05)	0.84 (3.63)	0.85 (3.16)
	10	0.79 (3.63)	0.73 (4.74)	0.20 (7.95)	0.30 (6.89)	0.66 (4.63)	0.93 (3.58)	0.63 (5.95)	0.34 (7.58)	0.84 (3.89)	0.95 (2.58)
	25	0.86 (3.37)	0.76 (4.84)	0.19 (7.89)	0.27 (7.47)	0.69 (4.68)	0.91 (4.11)	0.63 (6.16)	0.42 (7.47)	0.92 (2.79)	0.93 (2.84)
	50	0.79 (3.63)	0.73 (5.11)	0.15 (7.84)	0.21 (7.47)	0.66 (5.05)	0.78 (4.42)	0.60 (6.16)	0.49 (6.58)	0.86 (3.05)	0.85 (3.21)
	75	0.78 (4.11)	0.75 (4.47)	0.17 (7.74)	0.20 (7.53)	0.58 (5.21)	0.79 (4.68)	0.59 (6.37)	0.50 (6.37)	0.85 (2.68)	0.82 (3.16)
	100	0.74 (3.95)	0.74 (3.95)	0.28 (6.68)	0.27 (7.11)	0.53 (5.68)	0.76 (4.89)	0.55 (6.74)	0.48 (6.53)	0.84 (2.79)	0.83 (2.79)

Both the average performance and average rank of PAST and CLAST are higher for G-mean. On the contrary, the results of LP, LS, SVC, RF and Bagging are lower for G-mean. This allows PAST to shorten distances with RF and Bagging. Despite the inherent limitations of online incremental learning, PAST is able to beat several offline widely used learners; its performance is close to that of the best positioned ensemble offline classifiers; it behaves specially well when the class imbalanced is taken into account, and it improves the average results of CLAST for all those cases in which unlabeled samples are received.

Finally, as we did in the previous section, we conducted a statistical analysis on the results. First, Friedman's test rejected the hypothesis of all the algorithms performing equivalently. Then, we apply post-hoc Finner test and also pairwise comparisons by means of Wilcoxon signed ranks test. Tables V.13 and V.14 summarize the differences encountered between PAST and the rest of the approaches according to post-hoc Finner test and Wilcoxon signed ranks test, respectively. Finner test found PAST to perform significantly better than LP and LS in all the cases for both accuracy and G-mean. Using G-mean as evaluation metric, Finner test also considered PAST to significantly improve the performance of kNN and SVC in four of the six labeling scenarios. On the other hand, PAST significantly degrades the accuracy reached by RF and Bagging for some specific labeling scenarios (RF: 25% and 100% labeled, and Bagging: 100%). The pairwise comparisons yielded slightly different results. PAST is found to significantly improve CLAST, LP, LS and kNN in all the cases, and NB and SVC in many of them. Meanwhile, RF and Bagging significantly outperform PAST in terms of accuracy and for the highest labeling percentages also in terms of G-mean.

Table V.13: Multiple comparison of the performance of PAST with the remaining data stream learners by means of a post-hoc Finner test after Friedman's test rejected null hypothesis of equality of all learners.

		Online SL	Offline SSL		Offline SL					
	%Lab	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
Accuracy	5	+	⊕	⊕	+	+	+	+	-	-
	10	+	⊕	⊕	+	+	+	+	-	-
	25	+	⊕	⊕	+	+	+	+	⊖	-
	50	+	⊕	⊕	+	+	+	+	-	-
	75	+	⊕	⊕	+	+	+	+	-	-
	100	=	⊕	⊕	+	+	+	-	⊖	⊖
G-mean	5	+	⊕	⊕	+	-	⊕	⊕	-	-
	10	+	⊕	⊕	+	-	+	⊕	+	-
	25	+	⊕	⊕	+	+	⊕	⊕	-	-
	50	+	⊕	⊕	+	+	+	⊕	-	-
	75	+	⊕	⊕	+	+	⊕	+	-	-
	100	=	⊕	⊕	+	+	⊕	+	-	-

[1] ⊕/⊖: PAST significantly improves/degrades the performance of the method in the column
+/−: PAST improves/degrades the performance of the method in the column
=: PAST obtains the same results that the method in the column

Table V.14: Pairwise comparison of the performance of PAST with CLAST and the set of offline learners by means of Wilcoxon signed ranks test.

		Online SL	Offline SSL		Offline SL					
	%Lab	*CLAST*	*LP*	*LS*	*NB*	*DT*	*kNN*	*SVC*	*RF*	*Bagging*
Accuracy	5	⊕	⊕	⊕	+	+	⊕	+	⊖	⊖
	10	⊕	⊕	⊕	+	+	⊕	+	⊖	⊖
	25	⊕	⊕	⊕	⊕	+	⊕	⊕	⊖	⊖
	50	⊕	⊕	⊕	⊕	+	⊕	+	⊖	⊖
	75	⊕	⊕	⊕	⊕	-	⊕	+	⊖	⊖
	100	=	⊕	⊕	⊕	-	⊕	+	⊖	⊖
G-mean	5	⊕	⊕	⊕	⊕	-	⊕	⊕	-	-
	10	⊕	⊕	⊕	+	-	⊕	⊕	-	⊖
	25	⊕	⊕	⊕	⊕	-	⊕	⊕	-	-
	50	⊕	⊕	⊕	⊕	-	⊕	⊕	⊖	⊖
	75	⊕	⊕	⊕	⊕	-	⊕	⊕	⊖	⊖
	100	=	⊕	⊕	⊕	-	⊕	⊕	⊖	⊖

[1] ⊕/⊖: PAST significantly improves/degrades the performance of the method in the column
+/−: PAST improves/degrades the performance of the method in the column
=: PAST obtains the same results that the method in the column

V.3.3 Real world data stream problems

In this section, we conduct experiments on different Real-World (RW) data stream problems and compare the performance of PAST with CLAST and other data stream classifiers. Below, we describe the experimental methodology and present the results obtained on each problem.

V.3.3.1 Experimental setup

We followed the same methodology described in Section V.3.1.1. The performance of PAST is compared with the same eight data stream classifiers: CLAST, CVFDT, CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, HAT, VFDR, AWEC and DWM. Likewise, the datasets are processed following a test-then-train setup, performance is assessed in terms of accuracy and G-mean, and six different labeling percentages are explored for each problem. The real-world data stream problems are the same that were used in Section III.3.3: EEG Eye State, Powersupply, London Bike Sharing and Bike Rental. See Table III.12 for details on the characteristics of each data stream. The results on the different problems are analyzed independently. As we are dealing with real data stream problems, the data distribution may suffer changes along time and, therefore, the performance evolution of the learners may not draw a stable increasing trend.

V.3.3.2 Detecting open/close eyes through EEG

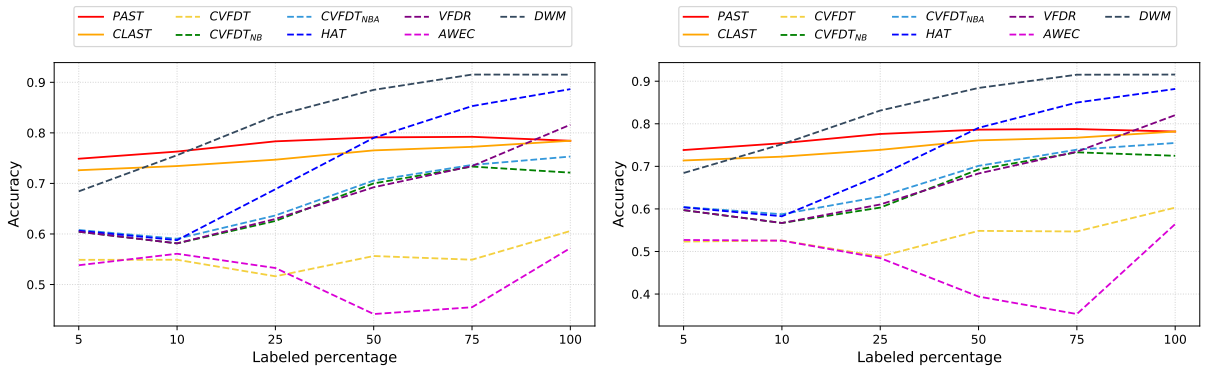
EEG Eye State data stream (Roesler, 2013) is composed of a continuous flow of 14980 samples from 14 EEG channels where each sample is labeled according to whether it matches an eye-open or an eye-close state.

Table V.15 includes the performance achieved by each online learner in the EEG Eye State dataset for each one of the six labeling scenarios. The rank positions of the algorithms are specified in brackets. The accuracy and G-mean values included in the table are calculated at the end of the stream and, therefore, are based on the predictions for all the examples in the data stream. Figure V.9 illustrates how this final performance of each algorithm correlates with the percentage of labeled examples. Figures V.10 and V.11 add context about the evolution along time of accuracy and G-mean, respectively.

At the end of the stream, PAST is the best ranked classifier for the two lowest labeling percentages. Once the ratio of labeled examples reaches 25%, PAST is surpassed by DWM and, for percentages superior to 50%, also by HAT. It is worth noting, the high performance of CLAST with high ratios of unlabeled examples. CLAST suffers from less wear due to the lack of labels than other proposals. It is the second best approach in the scenarios with higher label scarcity. Still, PAST obtains better performance than CLAST in all those scenarios where the labeling percentage is below 100% (when all the examples are labeled the behaviors of PAST and CLAST are equivalent). This advantage of PAST over CLAST is observed throughout the entire duration of the data flow, and becomes

Table V.15: Comparison of performance between PAST, CLAST and seven other online learners in EEG Eye State problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
Accuracy	5	74.88 (1)	72.61 (2)	54.88 (8)	60.44 (6)	60.79 (4)	60.64 (5)	60.44 (6)	53.82 (9)	68.43 (3)
	10	76.30 (1)	73.44 (3)	54.90 (9)	58.14 (6)	59.05 (4)	58.78 (5)	58.14 (6)	56.09 (8)	75.58 (2)
	25	78.32 (2)	74.69 (3)	51.63 (9)	62.57 (7)	63.66 (5)	68.85 (4)	62.97 (6)	53.27 (8)	83.40 (1)
	50	79.10 (2)	76.54 (4)	55.64 (8)	70.01 (6)	70.57 (5)	79.02 (3)	69.24 (7)	44.16 (9)	88.51 (1)
	75	79.22 (3)	77.24 (4)	54.91 (8)	73.33 (7)	73.66 (5)	85.29 (2)	73.42 (6)	45.51 (9)	91.53 (1)
	100	78.44 (4)	78.44 (4)	60.59 (8)	72.13 (7)	75.31 (6)	88.65 (2)	81.61 (3)	57.21 (9)	91.52 (1)
G-mean	5	73.83 (1)	71.38 (2)	52.33 (9)	59.72 (6)	60.45 (4)	60.37 (5)	59.72 (6)	52.71 (8)	68.44 (3)
	10	75.44 (1)	72.28 (3)	52.57 (8)	56.69 (6)	58.75 (4)	58.28 (5)	56.69 (6)	52.54 (9)	75.19 (2)
	25	77.61 (2)	73.87 (3)	48.82 (8)	60.31 (7)	62.89 (5)	67.90 (4)	61.04 (6)	48.45 (9)	83.11 (1)
	50	78.62 (3)	76.10 (4)	54.84 (8)	69.28 (6)	70.11 (5)	79.04 (2)	68.32 (7)	39.41 (9)	88.42 (1)
	75	78.76 (3)	76.73 (4)	54.70 (8)	73.29 (7)	73.92 (5)	84.99 (2)	73.44 (6)	35.31 (9)	91.53 (1)
	100	78.18 (4)	78.18 (4)	60.30 (8)	72.48 (7)	75.49 (6)	88.18 (2)	82.03 (3)	56.44 (9)	91.56 (1)



(a) (b)

Figure V.9: Performance evolution of the data stream algorithms as percentage of labeled data increases in EEG Eye State data stream.

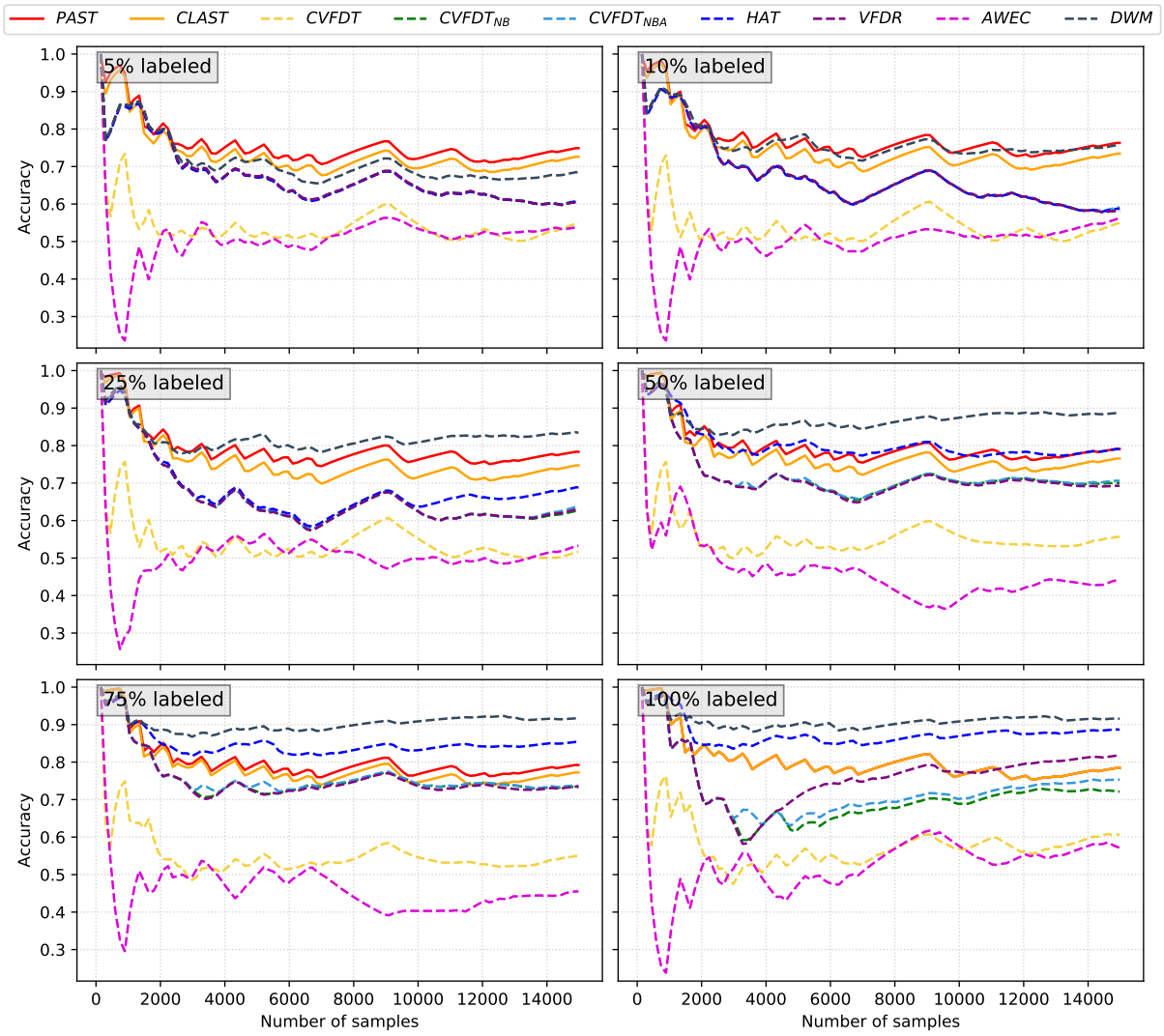


Figure V.10: Test accuracy evolution as the amount of data processed increases in EEG Eye State data stream.

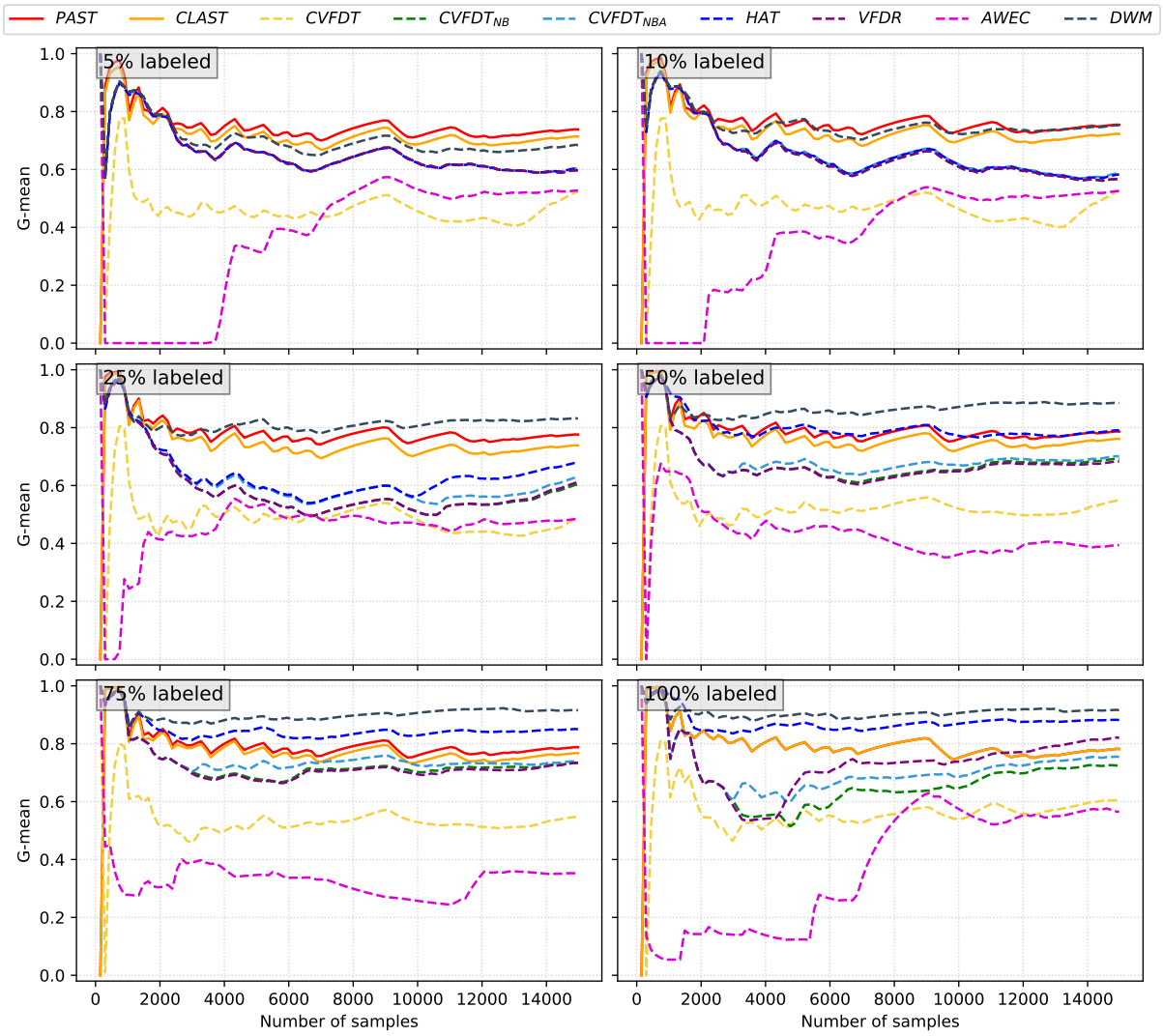


Figure V.11: G-mean evolution as the amount of data processed increases in EEG Eye State data stream.

more significant as the labeling rate decreases. All classifiers suffer performance ups and downs caused by changes in the distribution of the incoming data. Both CLAST and PAST maintain a fairly stable performance over time. These changes do not cause major alterations in the algorithm's ability to make accurate predictions. Furthermore, they perform better than CVFDT, CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, VFDR and AWEC during almost the entire duration of the stream for the seven labeling scenarios.

If accuracy and G-mean are compared, no major differences are observed between the evolutions described by most of the algorithms. This is to be expected in a problem with little imbalance between classes. Nonetheless, in the particular case of AWEC, some differences are observed depending on the measure used, obtaining zero G-mean values in different occasions.

V.3.3.3 Powersupply

Powersupply data stream (Zhu, 2010) is formed by 29928 instances that represent hourly power supply records of an Italian electricity company. The target classification task is aimed at predicting to which part of the day belongs the current power supply.

The results achieved at the end of the stream by the different learners for the six labeling ratios are gathered in Table V.16. Furthermore, the correlation between the ratio of labeled data and this final performance is visually represented in Figure V.12 for each of the classifiers. Figures V.13 and V.14 how the performance of each algorithm evolves along the stream.

Table V.16: Comparison of performance between PAST, CLAST and seven other online learners in Powersupply problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
Accuracy	5	49.05 (1)	48.79 (2)	24.91 (9)	47.34 (3)	47.29 (5)	46.92 (6)	47.34 (3)	43.08 (8)	46.37 (7)
	10	50.67 (1)	50.51 (2)	24.84 (9)	48.11 (3)	48.09 (5)	47.75 (6)	48.11 (3)	45.44 (8)	46.81 (7)
	25	52.47 (1)	52.28 (2)	24.73 (9)	48.83 (5)	48.82 (7)	48.94 (4)	48.83 (5)	49.16 (3)	48.00 (8)
	50	53.36 (1)	53.22 (2)	27.33 (9)	48.33 (7)	48.46 (6)	49.54 (4)	48.33 (7)	50.34 (3)	48.77 (5)
	75	53.67 (1)	53.56 (2)	33.01 (9)	48.96 (8)	49.09 (6)	49.25 (4)	48.96 (7)	51.48 (3)	49.13 (5)
	100	53.94 (1)	53.94 (1)	36.40 (9)	49.53 (5)	49.52 (6)	48.86 (8)	50.22 (4)	51.98 (3)	49.27 (7)
G-mean	5	42.22 (1)	41.84 (2)	11.55 (9)	33.56 (6)	33.60 (5)	34.00 (4)	33.56 (6)	31.91 (8)	36.96 (3)
	10	43.80 (1)	43.69 (2)	13.05 (9)	34.46 (6)	34.48 (5)	33.77 (8)	34.46 (6)	35.52 (4)	38.37 (3)
	25	46.03 (1)	45.65 (2)	15.41 (9)	36.51 (7)	36.52 (6)	36.76 (5)	36.51 (7)	40.79 (3)	39.62 (4)
	50	46.96 (1)	46.78 (2)	16.81 (9)	36.79 (6)	36.54 (8)	41.28 (4)	36.79 (6)	42.71 (3)	40.41 (5)
	75	47.34 (1)	47.21 (2)	22.30 (9)	41.12 (4)	39.86 (8)	40.60 (7)	41.12 (5)	44.19 (3)	41.05 (6)
	100	47.59 (1)	47.59 (1)	0.00 (9)	42.76 (5)	42.00 (6)	39.71 (8)	43.64 (4)	45.07 (3)	41.55 (7)

In Table V.16, PAST is ranked as the best classifier for the six labeling percentages (tied with CLAST for 100%). The next best performing classifier is CLAST. In this case, although PAST improves the outcome of CLAST, the results obtained by both are quite similar. This can be clearly appreciated in Figures V.12-V.14.

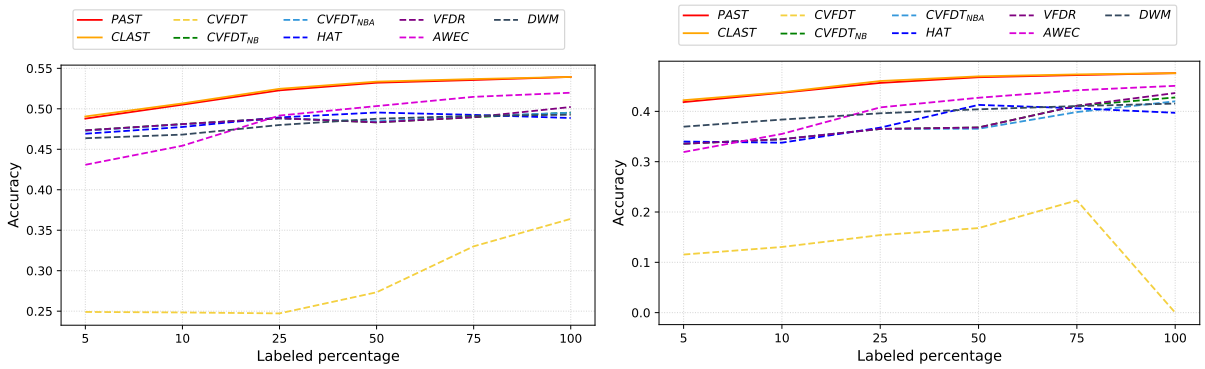


Figure V.12: Performance evolution of the data stream algorithms as percentage of labeled data increases in Powersupply data stream.

The improvement of PAST and CLAST over the rest of the algorithms is observed for both evaluation metrics, for the different labeling percentages and over the entire length of the data stream. Nevertheless, the difference is more noticeable for G-mean than for accuracy. In general, the evolution of the performance of the algorithms shows some oscillations but no abrupt changes. After PAST and CLAST, the following positions in the ranking are fairly evenly distributed, since CVFDT_{NB}, $\text{CVFDT}_{\text{NBA}}$, HAT, VFDR and DWM have quite similar results in most cases. AWEC lags behind for the lowest labeling percentages, requiring a larger number of examples to get competitive results.

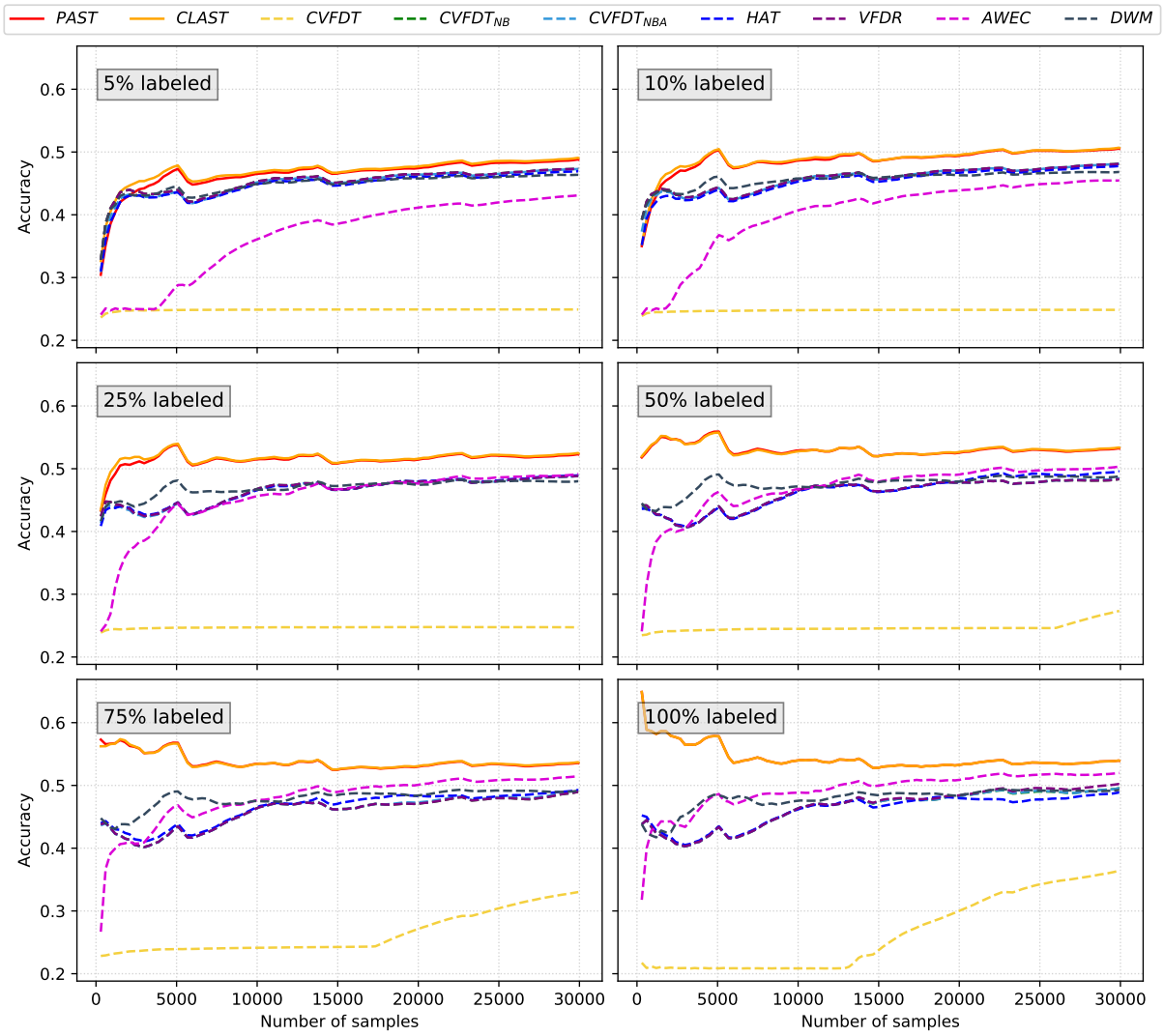


Figure V.13: Test accuracy evolution as the amount of data processed increases in Powersupply data stream.

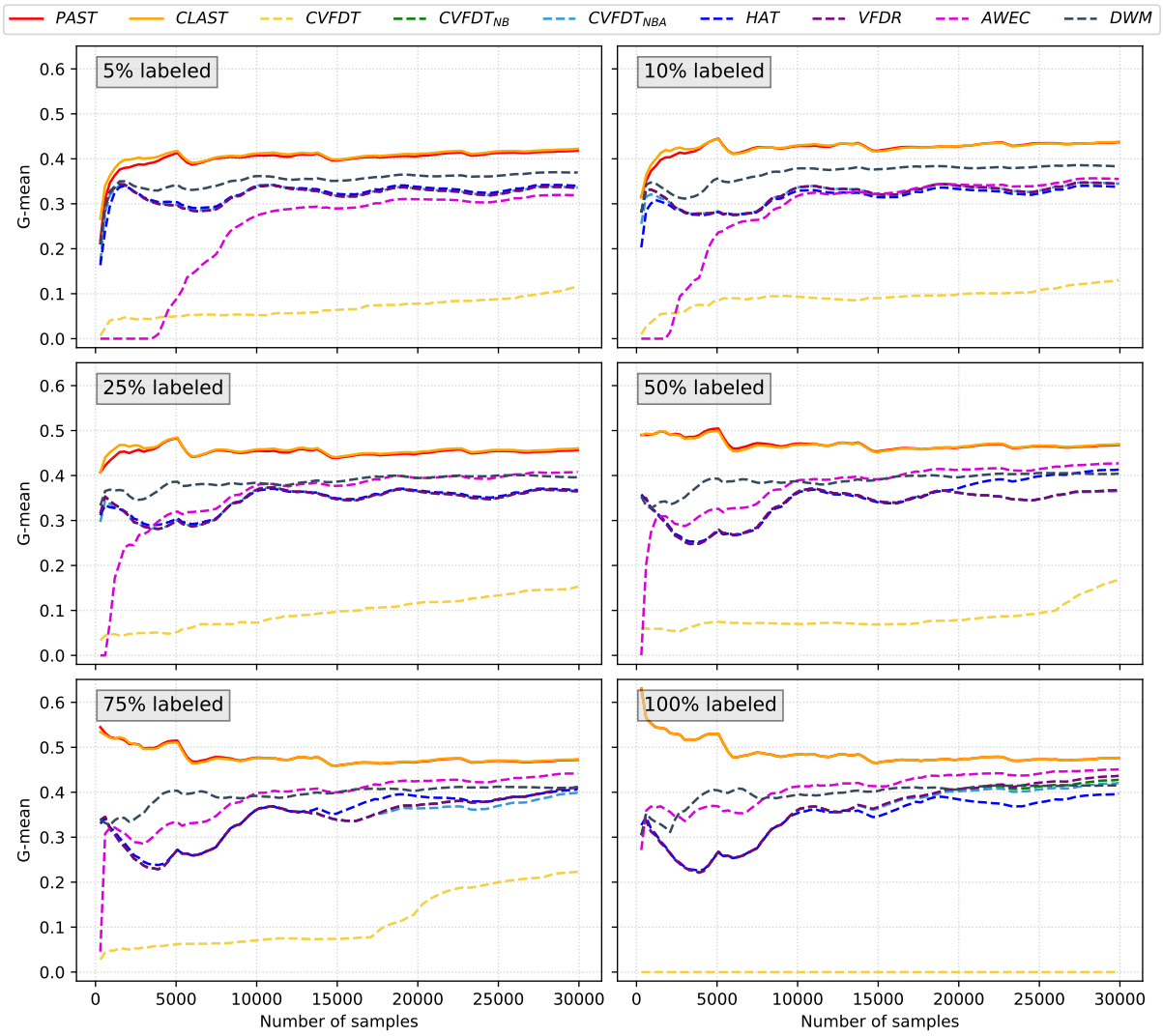


Figure V.14: G-mean evolution as the amount of data processed increases in Powersupply data stream.

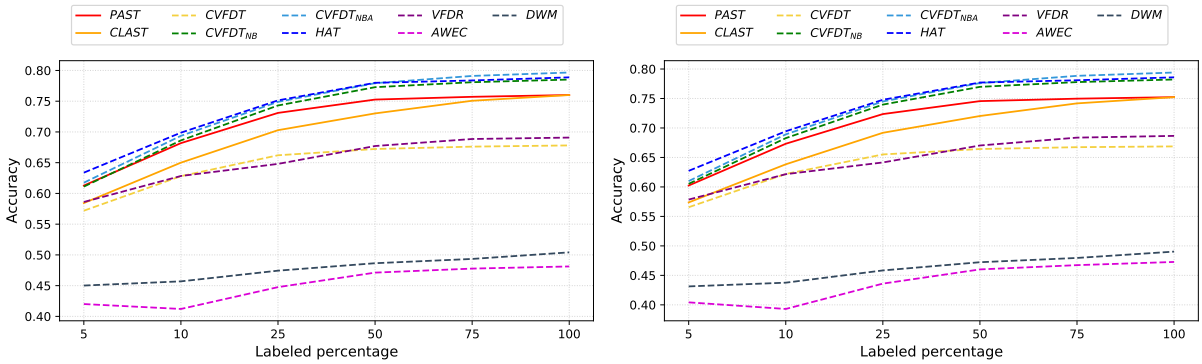
V.3.3.4 London Bike Sharing

London Bike Sharing data stream (Mavrodiev, 2020) combines information about weather conditions, bank holidays and hourly bike sharing in London. The aim of the classifiers is to predict the level of bike demand.

Following the same structure as in the previous cases, Table V.17 displays the performance (for the evaluation metrics accuracy and G-mean) of each data stream learner based on the whole London Bike Sharing data stream under different labeling conditions. In Figure V.15, we can observe the correlation of such performance and the amount of labeled data for each of the algorithms. Finally, the evolution of the behavior of the different approaches along the stream is shown in Figures V.16 and V.17.

Table V.17: Comparison of performance between PAST, CLAST and seven other online learners in London Bike Sharing problem. The results contained in the table are referred to the end of the stream.

	%Lab	*PAST*	*CLAST*	*CVFDT*	*CVFDT$_{NB}$*	*CVFDT$_{NBA}$*	*HAT*	*VFDR*	*AWEC*	*DWM*
Accuracy	5	61.28 (3)	58.43 (6)	57.19 (7)	61.11 (4)	61.74 (2)	63.38 (1)	58.61 (5)	42.01 (9)	45.02 (8)
	10	68.18 (4)	65.00 (5)	62.81 (7)	68.62 (3)	69.37 (2)	69.88 (1)	62.83 (6)	41.21 (9)	45.70 (8)
	25	73.10 (4)	70.29 (5)	66.20 (6)	74.30 (3)	74.89 (2)	75.13 (1)	64.80 (7)	44.76 (9)	47.44 (8)
	50	75.27 (4)	73.01 (5)	67.23 (7)	77.28 (3)	77.93 (2)	77.99 (1)	67.71 (6)	47.12 (9)	48.65 (8)
	75	75.71 (4)	75.07 (5)	67.61 (7)	78.09 (3)	79.11 (1)	78.37 (2)	68.85 (6)	47.77 (9)	49.34 (8)
	100	76.00 (4)	76.00 (4)	67.81 (7)	78.52 (3)	79.67 (1)	78.88 (2)	69.08 (6)	48.12 (9)	50.41 (8)
G-mean	5	60.24 (4)	57.37 (6)	56.56 (7)	60.53 (3)	60.99 (2)	62.73 (1)	57.86 (5)	40.42 (9)	43.13 (8)
	10	67.32 (4)	63.84 (5)	62.20 (6)	68.27 (3)	68.91 (2)	69.44 (1)	62.16 (7)	39.30 (9)	43.76 (8)
	25	72.36 (4)	69.18 (5)	65.51 (6)	73.96 (3)	74.50 (2)	74.78 (1)	64.15 (7)	43.6 (9)	45.84 (8)
	50	74.55 (4)	72.03 (5)	66.43 (7)	76.97 (3)	77.62 (2)	77.71 (1)	67.03 (6)	46.01 (9)	47.23 (8)
	75	74.97 (4)	74.17 (5)	66.74 (7)	77.78 (3)	78.84 (1)	78.09 (2)	68.36 (6)	46.74 (9)	47.95 (8)
	100	75.23 (4)	75.23 (4)	66.87 (7)	78.22 (3)	79.41 (1)	78.59 (2)	68.66 (6)	47.28 (9)	49.03 (8)



(a) (b)

Figure V.15: Performance evolution of the data stream algorithms as percentage of labeled data increases in London Bike Sharing data stream.

HAT obtains the best results for the four lowest labeling percentages. While CVFDT$_{\text{NBA}}$ does it for the two remaining scenarios. PAST obtains the third highest

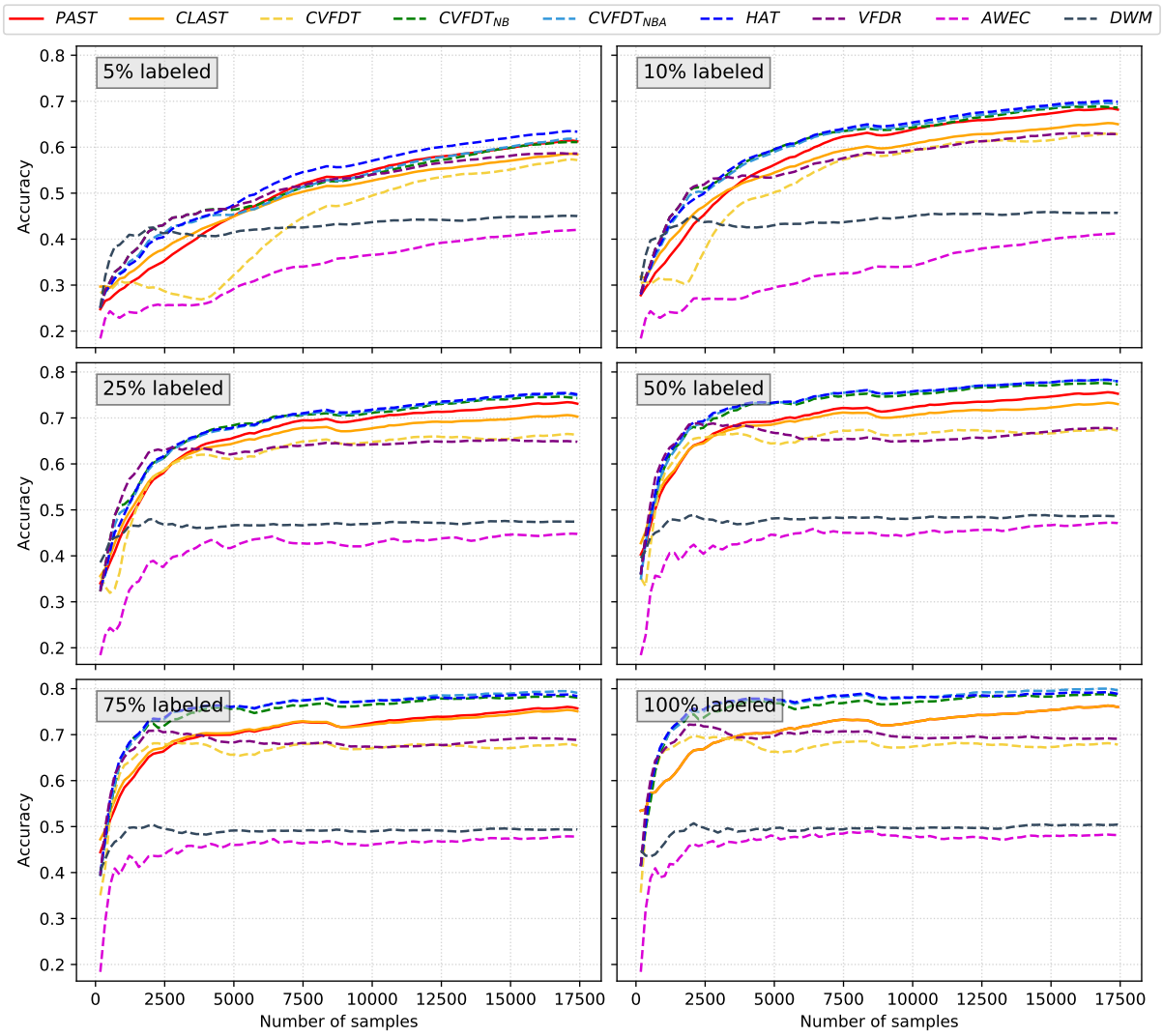


Figure V.16: Test accuracy evolution as the amount of data processed increases in London Bike Sharing data stream.

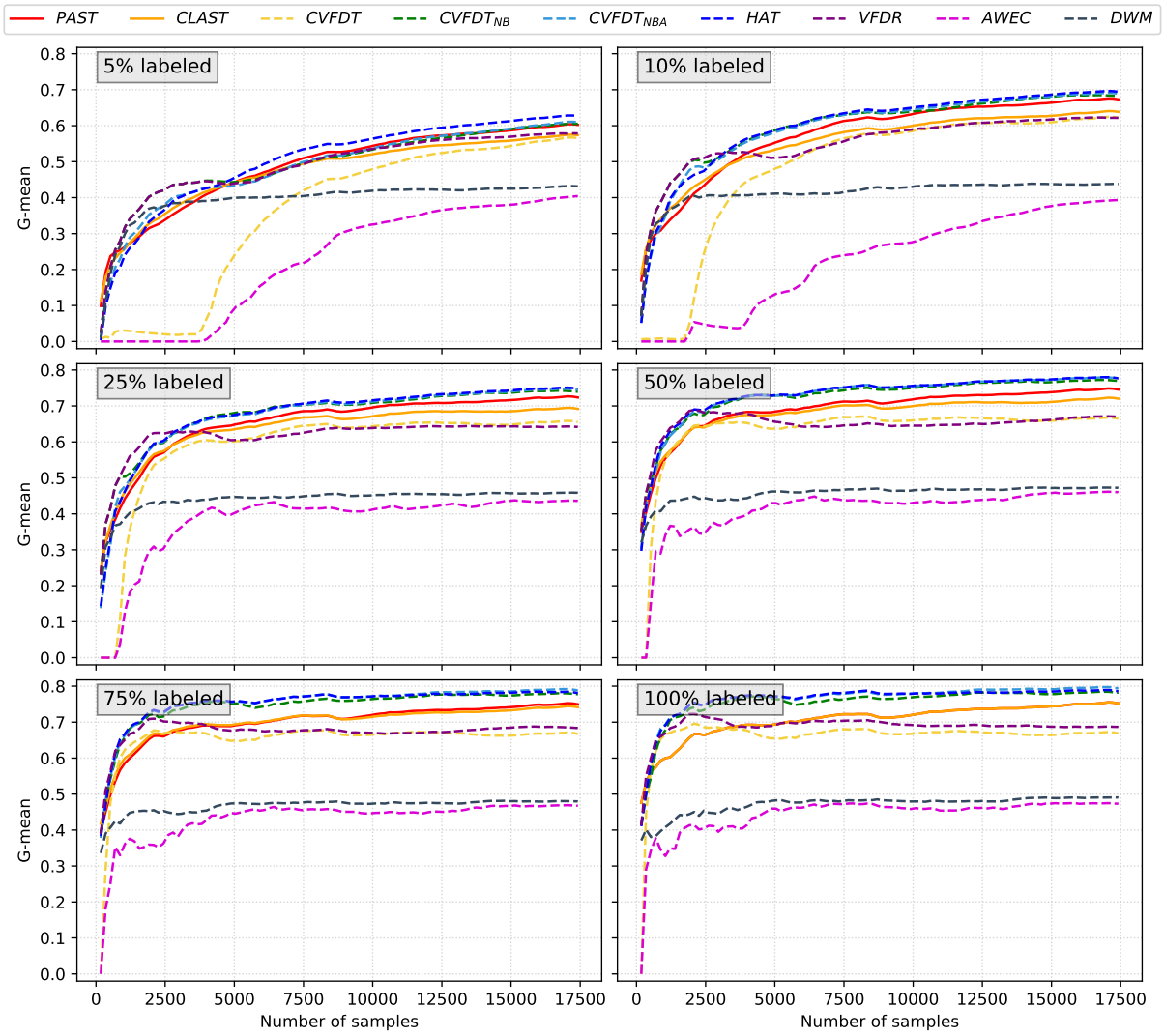


Figure V.17: G-mean evolution as the amount of data processed increases in London Bike Sharing data stream.

accuracy when 5% of the examples are labeled. In the rest of the cases, for both accuracy and G-mean, it occupies the fourth position. The third algorithm able to improve the performance of PAST in several scenarios is CVFDT_{NB}.

If we focus on the comparison with CLAST, we observe that PAST improves the accuracy and G-mean values obtained by CLAST in all cases with missing labels Figure V.15 shows how this advantage of PAST over CLAST remains fairly stable for the first few labeling percentages (5-25%) and begins to converge after 50% of the labels are available.

Analyzing the evolution of the performance of the algorithms along the stream (Figures V.16-V.17), we observe that PAST and CLAST tend to obtain very similar results at the beginning of the stream but soon PAST starts to improve the results of CLAST. In general, there are no major variations in ranking throughout the stream once the initial learning curve is overcome. Thus, the snapshot shown in Table V.17 is extensible to most of the stream. In addition, when comparing accuracy and G-mean, there are no major differences between the learning curves described by the algorithms in either case.

V.3.3.5 Bike Rental

As in the previous problem, the aim in the case of *Bike Rental* (Bansal, 2020; Fanaee-T and Gama, 2014) is to predict the level of bike demand based on the hour, weekday, month and season, as well as, weather conditions. As in the previous cases, we compare the performance of the algorithms at the end of the stream, we analyze how the number of available labels affects their performance and also how their behavior evolves along the data stream. Thus, Table V.18 shows the performance of the algorithms at the end of the stream for each of the labeling scenarios; Figure V.18 illustrates how this final performance correlates with the amount of labels, and Figures V.10 and V.11 depict the performance evolution as data are being received.

Table V.18: Comparison of performance between PAST, CLAST and seven other online learners in London Bike Sharing problem. The results contained in the table are referred to the final time stamp when all the examples in the stream have been used for both testing and training.

	%Lab	*PAST*	*CLAST*	*CVFDT*	$CVFDT_{NB}$	$CVFDT_{NBA}$	*HAT*	*VFDR*	*AWEC*	*DWM*
Accuracy	5	58.00 (1)	52.12 (2)	25.40 (9)	49.01 (3)	48.89 (5)	48.38 (6)	49.01 (3)	37.68 (8)	41.99 (7)
	10	63.71 (1)	58.83 (2)	28.91 (9)	49.31 (5)	49.42 (4)	49.71 (3)	49.31 (5)	36.91 (8)	42.64 (7)
	25	69.15 (1)	65.33 (2)	36.85 (9)	51.53 (5)	51.98 (4)	52.54 (3)	50.72 (6)	41.36 (8)	44.33 (7)
	50	71.50 (1)	69.24 (2)	40.29 (8)	52.53 (5)	54.10 (4)	54.55 (3)	51.43 (6)	40.02 (9)	45.32 (7)
	75	72.34 (1)	71.26 (2)	42.65 (8)	53.86 (5)	54.29 (4)	55.77 (3)	53.47 (6)	42.64 (9)	46.47 (7)
	100	72.84 (1)	72.84 (1)	44.90 (9)	53.28 (5)	54.99 (4)	56.05 (3)	53.28 (5)	45.20 (8)	47.70 (7)
G-mean	5	56.09 (1)	49.66 (2)	6.94 (9)	46.35 (3)	46.28 (5)	46.07 (6)	46.35 (3)	31.78 (8)	39.86 (7)
	10	61.86 (1)	56.12 (2)	13.56 (9)	46.46 (4)	46.28 (6)	46.54 (3)	46.46 (4)	33.93 (8)	40.07 (7)
	25	67.45 (1)	62.91 (2)	30.68 (9)	47.57 (3)	46.93 (5)	47.15 (4)	46.55 (6)	40.07 (8)	41.85 (7)
	50	69.81 (1)	67.17 (2)	33.24 (9)	48.57 (4)	48.40 (5)	49.71 (3)	47.13 (6)	37.57 (8)	43.33 (7)
	75	70.59 (1)	69.40 (2)	36.36 (9)	50.32 (4)	49.79 (5)	52.22 (3)	49.79 (6)	41.24 (8)	44.60 (7)
	100	71.08 (1)	71.08 (1)	38.02 (9)	49.23 (4)	48.85 (6)	52.48 (3)	49.23 (4)	44.09 (8)	46.11 (7)

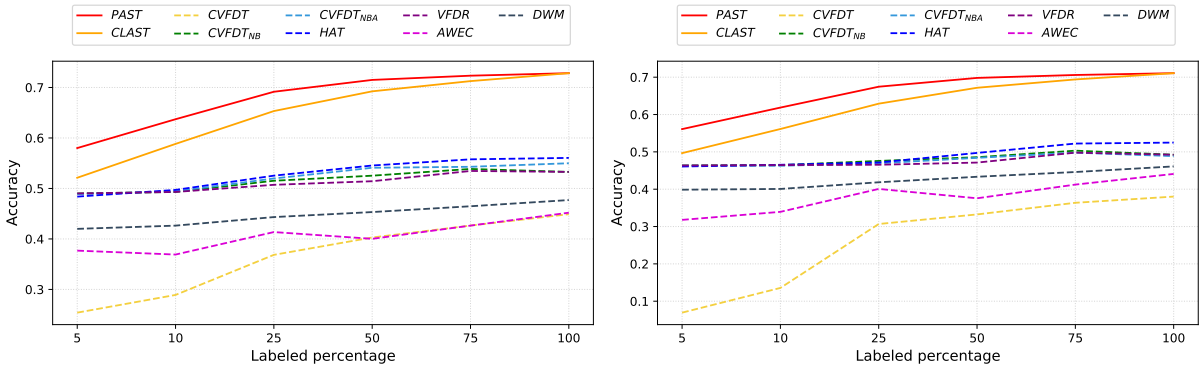


Figure V.18: Performance evolution of the data stream algorithms as percentage of labeled data increases in Bike Rental data stream.

PAST and CLAST are the best performing algorithms in this problem. In Table V.18 we can see how they are the best positioned classifiers for all labeling percentages and for both evaluation metrics. The next best positioned algorithms are CVFDT_{NB} and HAT, although the advantage of CLAST and PAST over them is quite noticeable in most cases. Furthermore, PAST significantly improves the results of CLAST in those scenarios with greater scarcity of labels. Figures V.19 and V.20 show that this improvement is not achieved at the end of the stream but starts after about 2500 examples have been received.

As we discussed in Section III.3.3, one of the four classes of this problem appears for the first time after more than 1000 instances have been received. This causes the algorithms to misclassify instances of this class, previously unknown to them, which results in the sudden decrease in G-mean observed in Figure V.20. As soon as the algorithms start to learn this new class, the G-mean values increase again. This figure shows how PAST and CLAST are able to learn this new class at the same velocity as most of the other algorithms but, in addition, they are able to reach significantly higher G-mean values.

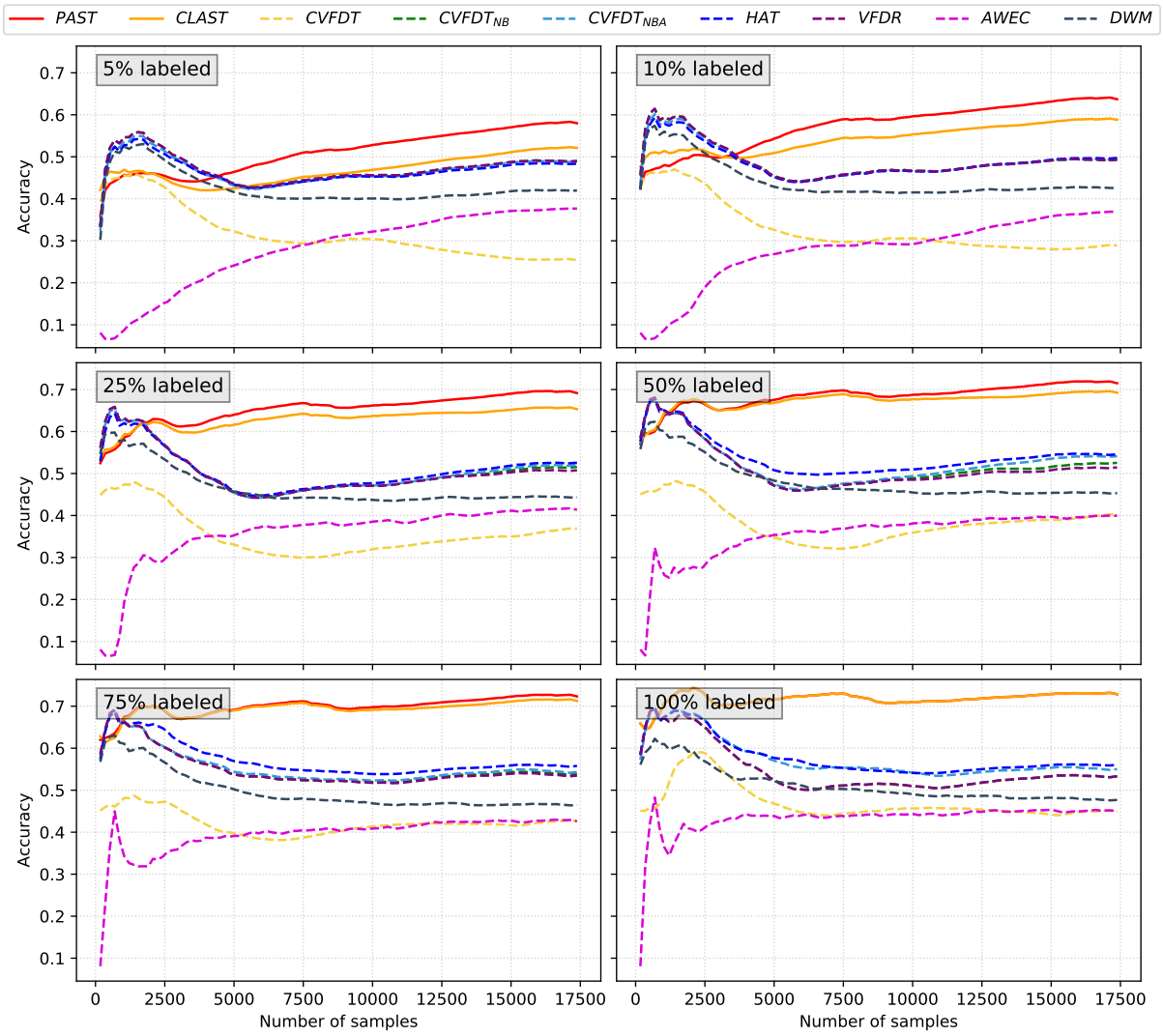


Figure V.19: Test accuracy evolution as the amount of data processed increases in Bike Rental data stream.

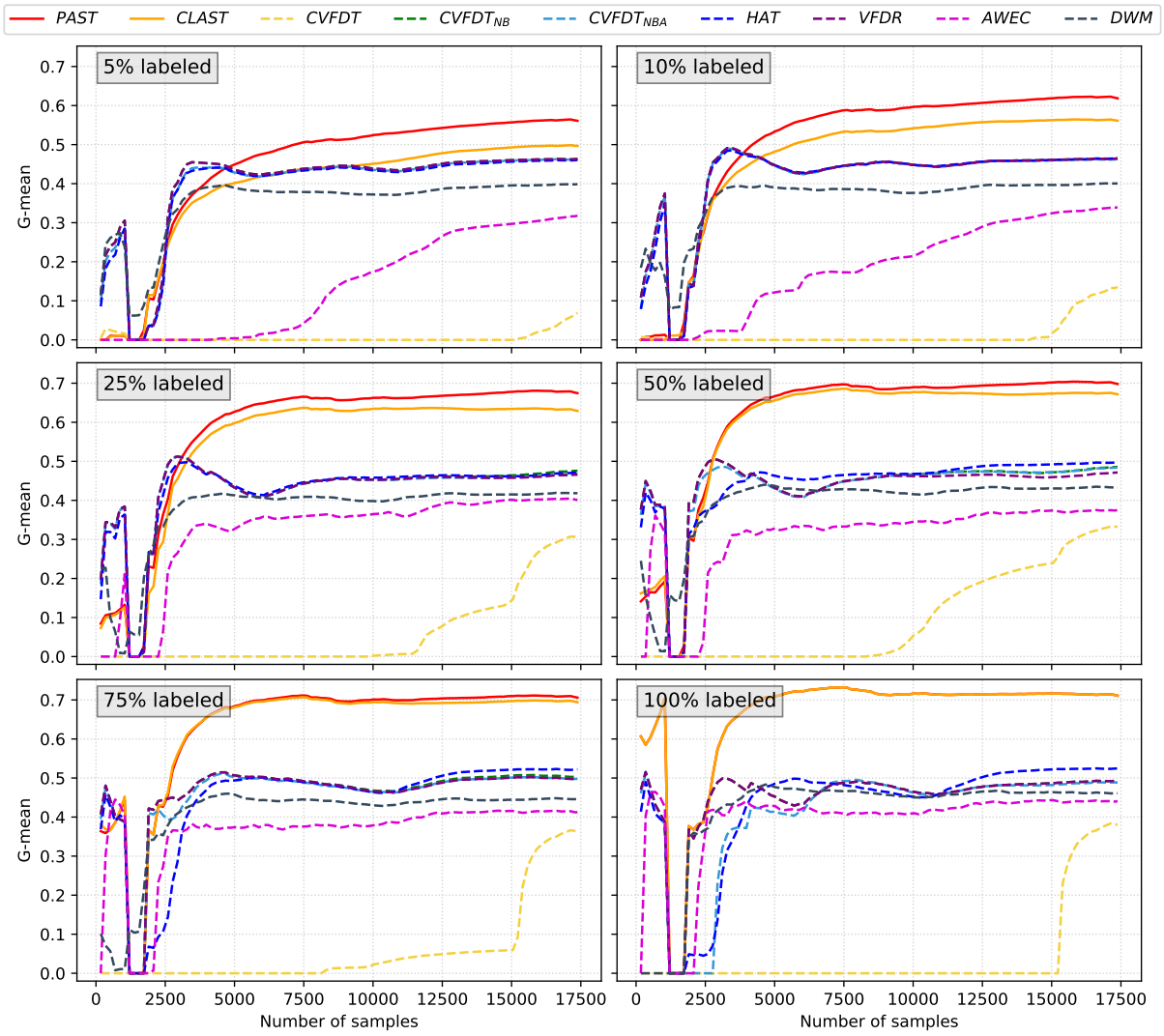


Figure V.20: G-mean evolution as the amount of data processed increases in Bike Rental data stream.

Chapter VI

Applications

VI.1 Smartphone usage analysis through association stream mining

The number of sensors surrounding us have significantly increased over the past few years, pervading multitude of activities from our daily life. Pervasive computing and the Internet of Things (IoT) are swiftly colonizing our daily routines. Thus, the number of cellular IoT connections is forecast to increase from 1 billion in 2018 to 4.1 billion in 2024, while the total IoT connections would raise from 8.6 billion to 22.3 billion (Ericsson, 2019). Part of this phenomenon is the rising interest in wearable monitoring devices, systems that accompany us during every single moment of our day and night and which are able to continuously gather a great amount of data. According to the technological consulting company Gartner, 178.91 million wearable devices were sold in 2018, a 25.8 increased was expected for 2019, i.e., 225.12 million of shipments were predicted worldwide; an amount that would reach 453.19 million in 2022 (Gartner, 2018).

From the analysis of data collected by such wearable devices we can extract insights about life style and behavior patterns that are valuable for a wide range of use cases and applications. Proof of this are the numerous studies from different areas that use them (Oliver and Flores-Mangas, 2006; Sanchez-Valdes and Trivino, 2015; Gravenhorst et al., 2015; Konsolakis et al., 2018; Ahmad et al., 2017; Wang et al., 2019; Casilari-Pérez and García-Lagos, 2019). Many of these studies focus on achieving advances in both physical and mental health, either by trying to encourage healthier life styles (Aharony et al., 2011b) or by detecting patterns that can act as risk state alarm indicators (Oliver and Flores-Mangas, 2006; Frost et al., 2013; Muaremi et al., 2014; Gravenhorst et al., 2015; Casilari-Pérez and García-Lagos, 2019).

There is a wide range of wearable devices available; being smartphones and smartwatches the current mainstream options. Smartphones, even though they are not wearable in the strict sense, are often included in this category since they have become omnipresent

up to the point that it is common to constantly carry them with us, both inside and outside home. Furthermore, smartphones offer certain advantages over other wearable devices. The number of smartphone connections reached 5.1 billion worldwide in 2018 and it is expected to increase up to 7.16 billion for 2024 (Ericsson, 2019). Opposite to other technologies, the fast extension of smartphones is not restricted to developed countries (Ericsson, 2019; World Health Organization, 2011). This may be partially due to the fact that smartphone cost has notably decreased while their functionality continued expanding. Indeed, a smartphone currently constitutes a powerful technological platform with a significant computation capacity, communication functionality, sensors, etc. This turns smartphones into both an ideal tool to bring certain beneficial services and applications straight to the user at any time and a valuable information source.

Smartphones are able to gather a wide range of different kind of data, from calls and messages, through app usage to data collected by the sensors integrated in them. Particularly useful is the accelerometer, a sensor included in virtually every current smartphone, which records are fundamental, for instance, in many activity recognition studies (Kwon et al., 2014; Sysoev et al., 2015; Ronao and Cho, 2016; Wang et al., 2019) or in physical activity monitoring studies aimed at preventing physical and mental disorders (Puiatti et al., 2011; Sanchez-Valdes and Trivino, 2015). Some studies based on smartphone sensing data have focused on extracting patterns or association rules (Sarker and Salim, 2018), and some of them have been able to relate certain patterns obtained from such data to risk situations. MONARCA project (Frost et al., 2013), centered on patients suffering bipolar disorder, performed an analysis of accelerometer values from smartphones (Puiatti et al., 2011) which revealed a correlation between physical activity levels and psychiatric assessment of depression (Osmani et al., 2013). Moreover, further research has concluded that certain assumptions about the emotional state of the user can be made based on statistical analysis of mobile sensing data, such as use-pattern changes (Gravenhorst et al., 2015). For instance, in Muaremi et al. (2014) several phone call parameters were examined and it was shown that call duration and accumulated talking time can be used to predict bipolar disorder episodes. In Sysoev et al. (2015) stress level is determined based on behavioral and contextual data obtained exclusively from smartphones.

Despite a great part of these studies perform an offline a posteriori analysis of the sensing data; the nature of this kind of data perfectly matches what is known as data streams. Wearable sensing devices generate chronologically sorted information in real-time and in a continuous way; data that are better modeled based not on persistent relations but on changing streams. To address this kind of problems and bring out knowledge in real-time we can use data stream mining. The potential of data stream mining in wearable sensing data field has begun to be exploited through a few research works. In Gomes et al. (2012) an activity recognition system on-board the mobile device itself learns through ubiquitous data stream mining in an incremental way. Moreover, smartphones pool a great amount of personal information and, therefore, they are a frequent cyber-attack target. In Mirsky et al. (2017) data stream clustering and anomaly detection techniques are combined in order to automatically detect attacks while they are in progress.

In this section, we address a real-world association stream mining application. The stream comprises smartphone activity data gathered during several months. The association stream mining algorithm dynamically maintains and constantly updates a set of association rules that explain the user activity at any time in a very efficient way. Figure VI.1 schematically represents the data stream scenario presented. The used algorithm is Fuzzy-CSar-AFP. In Section IV.3, Fuzzy-CSar-AFP was already applied to a real-world data stream problem. However, on that occasion the real-world problem was used as a benchmark to conduct a comparative analysis between the performance of Fuzzy-CSar-AFP and other proposals. In this case, the aim is not to analyze the performance of Fuzzy-CSar-AFP but to focus on the analysis at different levels of the extracted rules and the knowledge they provide.

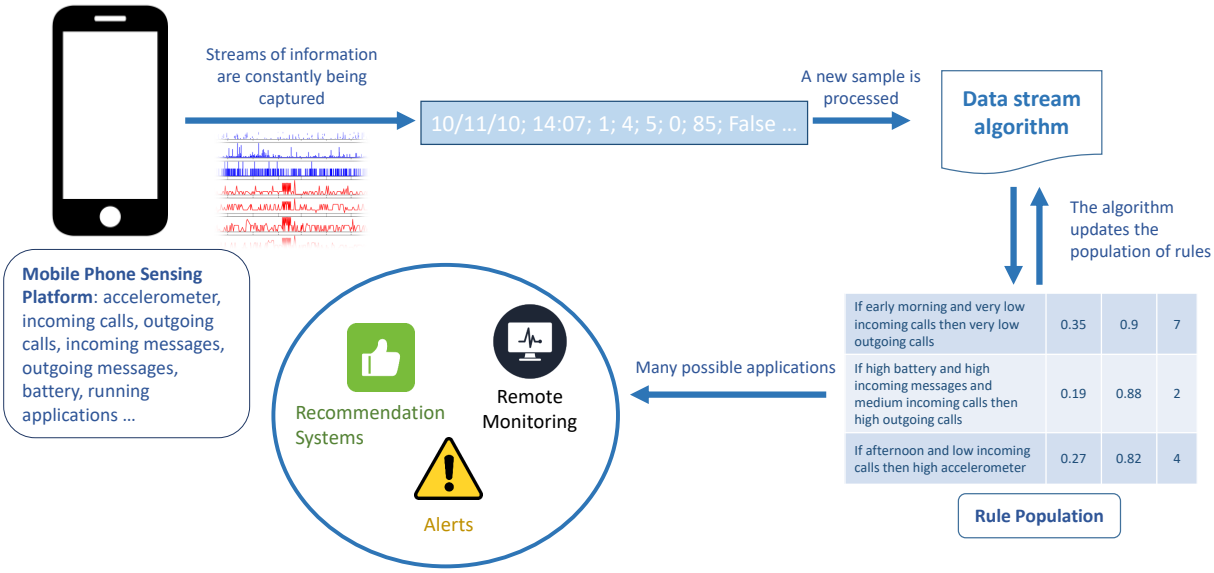


Figure VI.1: Schema of the association stream mining scenario presented.

The data were gathered during a sociological experiment in the course of which, in addition to the smartphone activity data employed by the algorithm, subjective information about emotional state, collected through periodic surveys, was also available and is used to complement the analysis of the discovered rules. Thus, the evolution over time of the levels of happiness, stress, productivity and health expressed in the surveys (but not shown to the algorithm) is compared with the evolution of the different rules generated automatically in real time.

VI.1.1 Some context on the real-world data: Friends and Family Study

Friends and Family Study is a research work conducted by the Media Lab, Massachusetts Institute of Technology over the years 2010 and 2011 (Aharony et al., 2011b,a). This study transformed a residential community nearby to a well-known North American university into a longitudinal living laboratory for over 15 months. For nearly a year, every behavior, communication, and social detail from the lives of a large number of members of the mentioned community were recorded while they went about their daily tasks as normal. A total of 130 subjects were part of the study. During the study period, a huge amount of data was collected giving rise to an unique and very rich longitudinal dataset, known as *Friends and Family* dataset. Such dataset includes a large collection of phone-based signals including location, communication activities (calls and text messages), installed applications, running applications, accelerometer information, nearby Bluetooth devices, ... The study was divided into two phases. The first phase consisted of a pilot stage that started in March 2010 and lasted 6 months, 55 subjects took part in the study during this phase. The second phase of the study was launched in September 2010 with 130 people participating in it (about 64 families). These 130 participants were selected out from approximately 200 applicants following diversity criteria in order to obtain a representative sample of the community and the different sub-communities. Due to the origin and peculiarities of the information included in the dataset, the study was performed under strict protocols that ensured the privacy of every participant.

If we focus on the data collection obtained from the study and, specially, on the part that was published and has served as starting point dataset for our particular study, we found a collection whose size exceeds 7 GB and which is distributed into several files from different sources and with different formats. In general terms, in this data collection we can differentiate the next components: (1) data from mobile phone sensing platform, and (2) surveys. On the one hand, Android OS based mobile phones were provided to the participants under the condition that these phones should be their main phones during the study. These devices were supposed to make the role of wearable social sensors to record subjects activity features. This information is the core of the data collection. On the other hand, surveys were completed by the participants at regular intervals, combining web-based and on-phone surveys. In the monthly surveys, subjects answered questions related to self perception of relationships, group affiliation, and interactions, along with standard scales such as the Big Five personality test (John et al., 1999). Meanwhile, daily surveys included questions about sleep, mood and the amount of time spent on certain activities.

Several studies have used this same dataset for a wide range of aims, such as: inferring some characteristics of the subjects (personality traits, age, nationality, marital status, etc.) (Mønsted et al., 2018; Altshuler et al., 2012, 2013); better understanding social systems, friendship and human behavior (Shmueli et al., 2014), or building social networks to study diffusion and interaction dynamics (Pan et al., 2011). Despite this type

of data fits perfectly the concept of data stream, none of these studies explores the option of treating this data collection as a stream. In fact, just a few of the cited studies exploit the temporal dimension of the data collection (Altshuler et al., 2012, 2013) and they are limited to offline traditional machine learning algorithms. This forces them to carry out series of repetitive experiments in which the period covered by the samples is increased from one experiment to the next in order to get a result evolution (the algorithms used are not capable of incremental learning). Furthermore, they often restrict the experiments to a small part of the full data collection, not reaching to cover the totality of the period for which data are available.

VI.1.2 Data stream preparation

The *Friends and Family* study was not aimed at data stream mining. Hence, although the nature of the smartphone activity data matches a data stream, the way the published data collection is organized and structured does not fit with the input of a data mining algorithm. It is necessary to undertake a certain preparation process to structure the data as a unique input stream to a data stream mining algorithm.

The Friends and Family study had a sociological perspective, studying different types of interactions between community members and their consequences. Therefore, data collection combines data from different subjects while dividing the data according to their type. In our case, we will treat the data generated by each subject as an independent flow in which the different types of information from the mobile phone are combined. Thereby, the data will be processed as if they were being received by the algorithm in real time directly from the phone.

Each information type was gathered with its own peculiarities. Depending on information type and collection method, data format, organization and sampling frequency varied. For instance, information about calls or text messages were only registered when a new call or message was sent or received. However, records of accelerometry were generated once each 2 minutes and each one represented an agglutinated measure of what has been occurring during that couple of minutes. By the same token, apps present in the phone were scanned with a 10-minute frequency and information about app usage gathered every 30 seconds. In order to unify and integrate all the types of data, a sampling frequency of one minute was chosen trying not to lose too much detail but, at the same time, avoiding data granularity needlessly small.

Moreover, an incremental and completely online algorithm presents some peculiarities derived from the fact that each data is going to be processed only once and the algorithm does never deal with the full dataset. For instance, this can make data that appear only for a moment of time to end up disappearing for the algorithm despite they could provide relevant information. Trying to minimize this risk, modifications related to call logs, message logs, and running applications were made, generating variables in the way of "how many calls have been registered in the last X minutes".

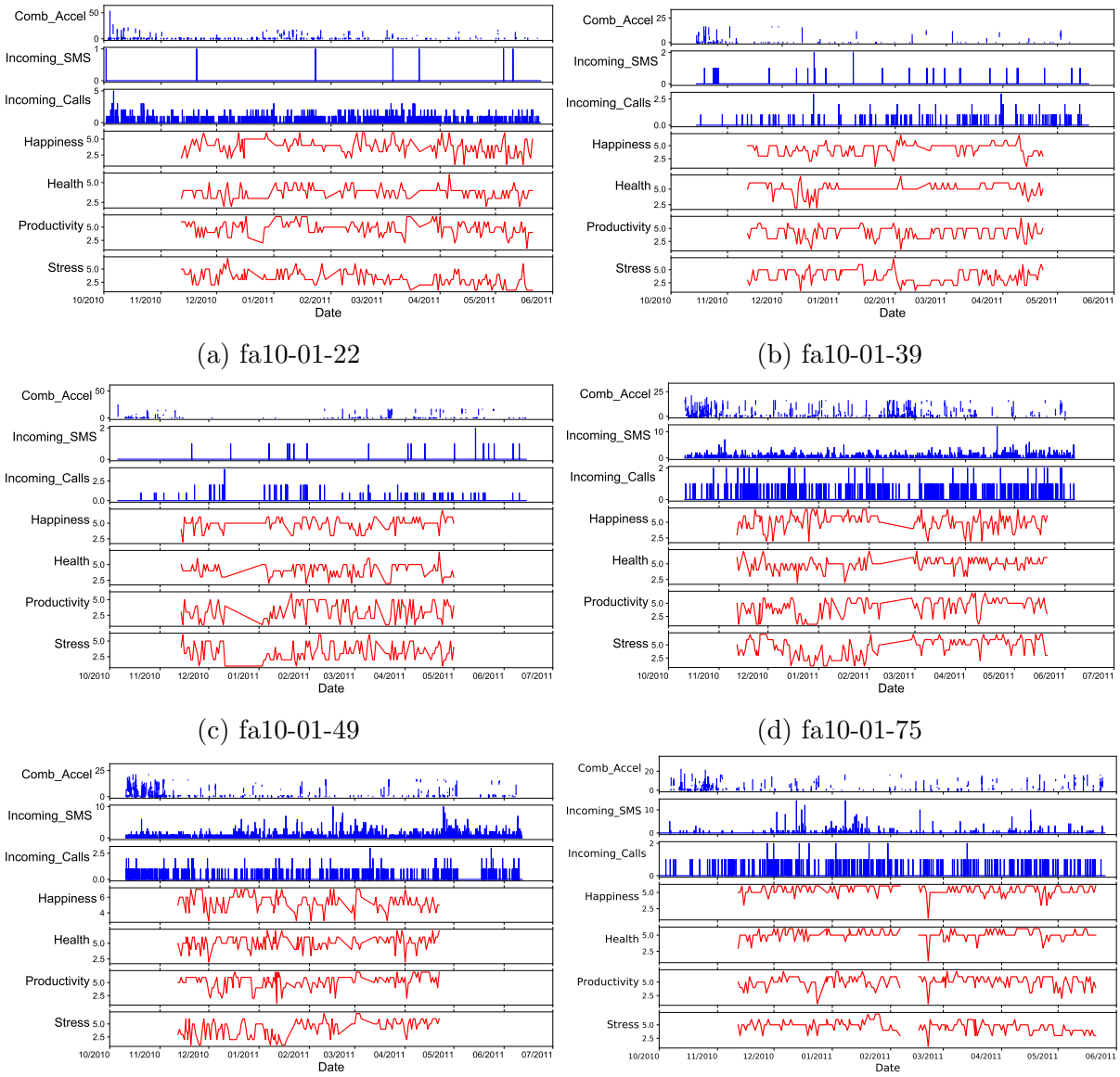
Likewise, not all the information contained in the raw data turned out to be useful for our goals. All those data whose main purpose is to highlight the relationships between participants (all of them members of the same community) were likely to be excluded from our study, since our goal is to monitor each participant's individual behavior and not to study the social relationships between them. Other data were also found irrelevant to our study due to different reasons (e.g., battery technology type).

As it has been mentioned, the study was divided in two phases; the first one, a pilot phase in which a lower number of participants took part and where the researchers experimented with different data collection options. Hence, for reasons of coherence and consistency we decided to use only data collected during the second phase of the study. Finally, duplicated information was cleaned up, and data was grouped by participant and chronologically sorted.

Note that this preparation process is only needed because the published data collection was structured in a way that matches the requirements of the sociological study. In fact, the output of this process is much more similar to the raw data generated by the mobile phones. The process from those raw data to the input stream for the algorithm would be much more trivial and could be done in a real-time online manner.

Although the information from the surveys is not provided to the algorithm, it is also necessary to carry out the part of the process corresponding to duplicate cleaning, aggregation by participant and chronological sorting, in order to use this information to complement the rule analysis. Furthermore, taking into account both the scope of the surveys and their periodicity, we only use information collected through the daily surveys. These daily surveys assessed interesting aspects about the lives of the participants that could potentially complement the information from the mobile phone sensing platform. They gathered information about happiness, health perception, productivity sensation, stress, sleeping hours and hanging out time. Meanwhile, the monthly surveys are focus on social aspects that do not fit the individual approach we are taking. Moreover, the daily surveys were on-phone. Compliance rate usually gets better when the subjects can use their own smartphone to fulfill the survey since they usually have it on hand and memory errors are more likely avoided (Bardram et al., 2013).

Figure VI.2 represents the evolution over time of the main attributes after the data preparation process. The data presented in such figure correspond to six different subjects who are also included as examples in different parts of the result analysis described in Section VI.1.4. The plots have a 1-day resolution, so it must be kept in mind that variables which represent call and message counters indicate for each minute the number of calls/messages registered in the last ten minutes. Thus, the values shown for these accumulative variables (call and message counters) in the graphics for a given day are ten times the actual number of calls/messages that have occurred that day. This figure helps us to understand how complicated it would be to extract useful information and discover interesting associations between different features without the assistance provided by the association stream mining algorithm.



(a) fa10-01-22

(b) fa10-01-39

(c) fa10-01-49

(d) fa10-01-75

(e) fa10-01-76

(f) sp10-01-32

Figure VI.2: Evolution over time of part of the data gathered through the mobile phone sensing platform and surveys for six different subjects, after data preparation process.

VI.1.3 Experimental setup

The dataset which is finally provided to Fuzzy-CSar-AFP is formed by 11 input attributes: (1) weekday; (2) day minute (from 0 to 1439); (3) battery level; (4) number of incoming text messages; (5) number of outgoing text messages; (6) number of incoming calls; (7) number of outgoing calls; (8) number of missed calls; (9) accelerometer; (10) any application removed (uninstalled) from the device (boolean); and (11) any application running (boolean). Hence, for each participant in the second phase of the *Friends and Famility* study, a data stream composed by the former eleven attributes is provided as input to Fuzzy-CSar-AFP and the evolution over time of the resulting rule populations is analyzed below.

Regarding the configuration of Fuzzy-CSar-AFP, for most of its configuration parameters, we took as a reference the values used in Sancho-Asensio et al. (2016) for the algorithm Fuzzy-CSar, which were obtained experimentally following the recommendations claimed in Orriols-Puig et al. (2008a). Thus, $\upsilon = 1$, $P_{\#} = 0.2$, $P_{\chi} = 0.8$, $\{P_{I/R}, P_{\mu}, P_{C}\} = 0.1$, $\delta = 0.1$, and θ_{mna} is automatically set to the number of variables. Nonetheless, other parameters were adjusted based on the particular characteristics of the problem we are addressing here, therefore: $\theta_{GA} = 25$, $\{\theta_{del} = \theta_{sub}\} = 10$, $\theta_{exp} = 20$ and *maxLingTermsPerVariable* was automatically set to half of the total number of linguistic terms for the used granularity. The population size was set to 10000 individuals. All the variables use Ruspini's strong fuzzy semantics with between 2 and 5 linguistic terms. The same configuration is applied to the streams of all the subjects and, therefore, all the results shown in the following sections have been obtained using these settings.

VI.1.4 Association rule analysis

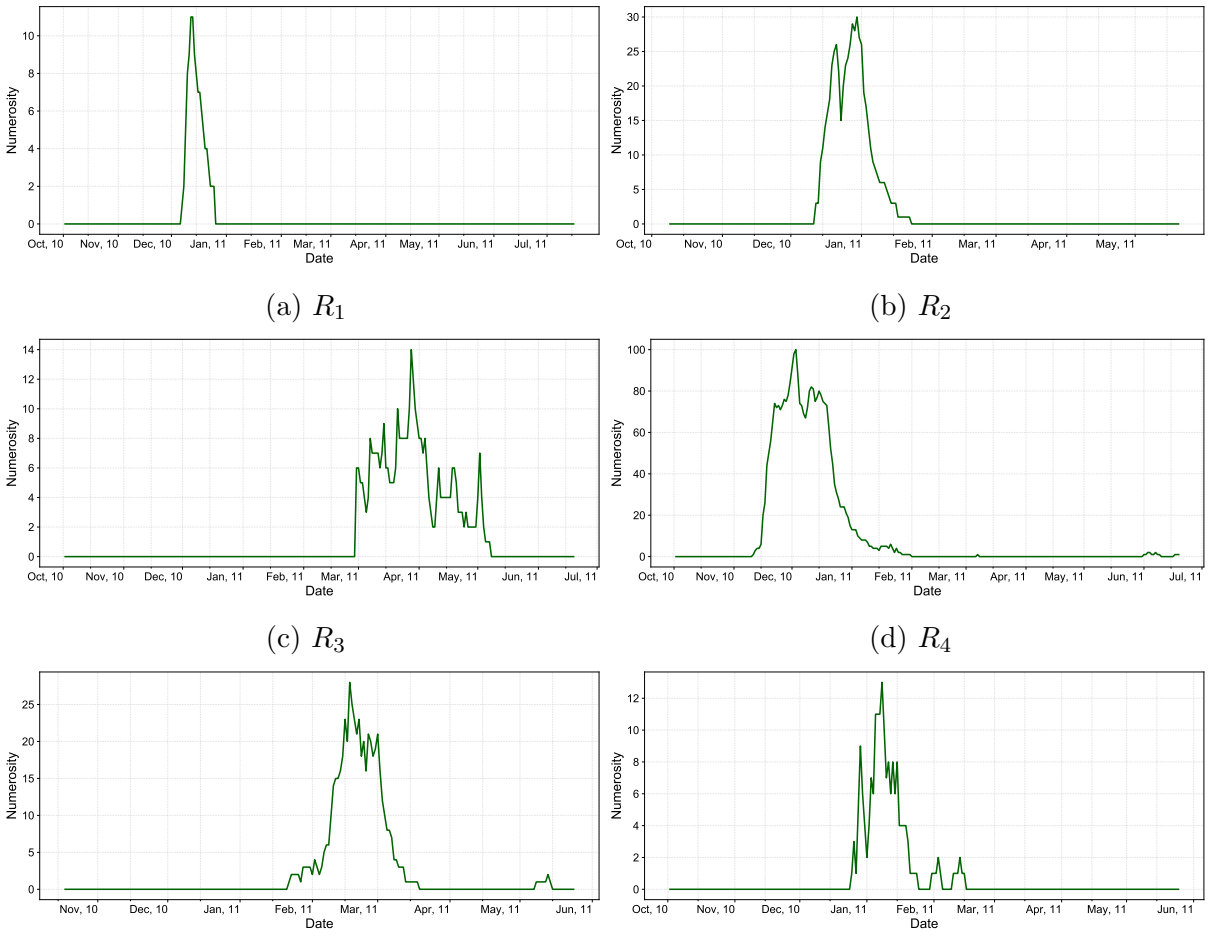
Fuzzy-CSar-AFP continuously maintains a population of association rules. As part of our analysis, we study how such population evolves over time for different subjects. In order to be able to do that, we need the algorithm to periodically print the content of such population. Furthermore, given the difficulty of trying to analyze the entire population at once, along our analysis we will be focusing on different set of rules, consequents or antecedents.

The nomenclature used to refer the linguistic terms is the same as in Chapter IV, which was illustrated in Figure IV.3. Therefore, the possible sets of linguistic terms go from $\{S_2, L_2\}$ with granularity 2 to $\{XS_5, S_5, M_5, L_5, XL_5\}$ for granularity 5. Note that the sub-index denotes the granularity.

VI.1.4.1 Better understanding rules through real examples

Table VI.1 shows six rules obtained by the algorithm. These sample rules help to understand and visualize in a better way the structure of the association rules generated by the

algorithm. In addition to the structure of the rules, Table VI.1 also presents some information about them. Concretely, for each rule the following attributes are presented: (1) *Rule Id*, identifier used to refer the rule; (2) *Subject Id*, identifier of the subject to which the rule corresponds; (3) *Rule*, structure of the rule; (4) *supp*, rule support at the time of maximum numerosity; (5) $conf$, rule confidence at the time of maximum numerosity; and (6) *num* maximum numerosity of the rule. Additionally, we can see the evolution of the number of copies stored in the pool (numerosity) for each one of the rules from Table VI.1 in Figure VI.3. Numerosity is a representation of the relative importance of a rule at any time.



(a) R_1 (b) R_2

(c) R_3 (d) R_4

(e) R_5 (f) R_6

Figure VI.3: Numerosity evolution for six different rules. R_1 was discovered for subject fa10-01-06, R_2 for fa10-01-35, R_3 and R_4 for sp10-01-12, R_5 for fa10-01-67, and R_6 for fa10-01-13.

The rules included in Table VI.1 are different from each other in their meaning, in their temporal evolution and in the subject for which they are true. These rules exemplify both the capacity of the algorithm to adapt to the particularities of each subject and

Table VI.1: Six example rules obtained for the data streams of five different subjects (numerosity evolution of these rules is shown in Fig. VI.3). Support, confidence and numerosity values shown in the table are referred to the maximum numerosity time stamp.

Rule Id	Subject Id	Rule						*supp*	*conf*	*num*
R_1	fa10-01-06	**IF**	Battery Level	is		and		0.110	0.796	11
			Incoming Calls	is		and				
			Missed Calls	is		and				
			Accelerometer	is						
		THEN	Outgoing Calls	is						
R_2	fa10-01-35	**IF**	Minute	is		and		0.017	0.938	30
			Battery Level	is		and				
			Incoming Messages	is		and				
			Outgoing Calls	is		and				
			Missed Calls	is		and				
			Accelerometer	is						
		THEN	Incoming Calls	is						
R_3	sp10-01-12	**IF**	Weekday	is	Sunday	and		0.103	1.0	14
			Minute	is		and				
			Missed Calls	is						
		THEN	Accelerometer	is						
R_4	sp10-01-12	**IF**	Minute	is		and		0.644	1.0	100
			Missed Calls	is						
		THEN	Accelerometer	is						
R_5	fa10-01-67	**IF**	Incoming Messages	is		and		0.485	0.951	28
			Outgoing Messages	is		and				
			Missed Calls	is						
		THEN	Accelerometer	is						
R_6	fa10-01-13	**IF**	Minute	is		and		0.055	0.783	13
			Incoming Messages	is		and				
			Outgoing Messages	is		and				
			Incoming Calls	is		and				
			Accelerometer	is		and				
			Any Running App	is	False					
		THEN	Outgoing Calls	is						

the evolution the rule population experiences at the same time as the input data stream itself evolves. Moreover, all these rules, despite representing diverse patterns, can be easily interpreted. Thus, R_1 describes how the number of outgoing calls tends to be low or very low when few or no incoming and missed calls are registered and high levels of accelerometer are being recorded even though battery level is medium. Later on, we will explain how accelerometer can be interpreted as an indicator of the amount of physical activity performed. According to this interpretation, it would make sense for the subject to not be making too many calls during high physical activity periods. Other rules that include variables related to calls and accelerometer are R_2, R_4 and R_5. R_2 implies that the number of incoming calls is high when battery level is low, a high number of incoming messages are being received and high levels of accelerometer are being registered. R_4 and R_5, although corresponding to different subjects, share the same consequent (accelerometer $[L_5, XL_5]$) and have similar antecedents. Both rules imply that the presence of missed calls (combined with a certain time interval, in the case of R_4, or with the existence of incoming messages and the scarcity of outgoing messages, in the case of R_5) coincides with high levels of accelerometer (the subject could be engaged in physical activity). Regarding their numerosity evolution, these four rules fit the same general pattern: they appear at a certain point of the stream, their numerosity rapidly increases, reaches a peak and start descending shortly afterwards. This behavior is particularly clear in the case of R_1.

On the other hand, we have rules like R_3 whose numerosity suffers periodic ups and downs. R_3 points that during Sunday afternoons and evenings, high values of accelerometer are registered. The fact that the rule refers to something that occurs only once a week causes the cyclical numerosity increases and decreases observed in Figure VI.3. R_3 reaches a certain number of copies in the population during Sunday. During the next days, copies of the rule are removed from the population (its numerosity decreases) until Sunday arrives again when the numerosity of the rule begins to increase. This occurs every week from the last week of February to mid-April (for subject sp10-01-32). Lastly, some ups and downs can also be appreciated in the evolution of R_6 but without a fix temporal cycle (as the one observed for R_3). R_6 links high number of outgoing calls with occasions where there is a shortage of incoming messages and calls, some messages are sent, high levels of accelerometer are being recorded and no application is running during the first half of the day.

VI.1.4.2 Rule aggregation by consequent

Studying groups of rules rather than specific rules can allow us to generalize our analysis and facilitate finding common ground between different subjects. Following this idea, we aggregate rules based on their consequent and analyze the evolution of different groups of rules.

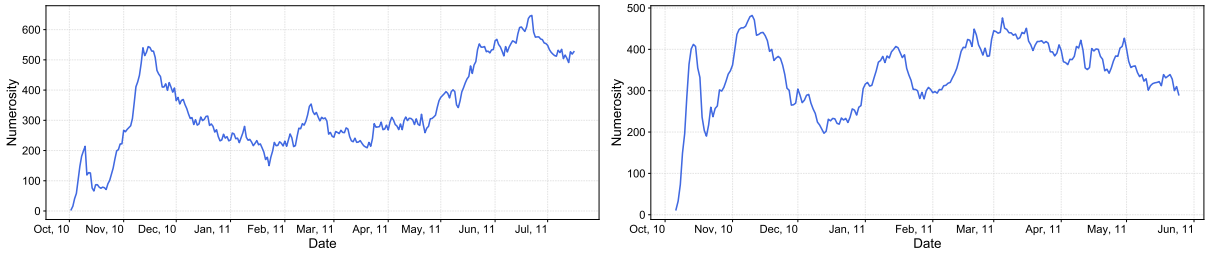
As mentioned above, accelerometer records can be linked to the physical activity performed by the subject. Over the last years, several studies have established and refined accelerometer as a tool for tracing physical activity (Bouten et al., 1997; Troiano et al.,

2008; Anderson et al., 2007; Toscos et al., 2008; Saponas et al., 2008; Puiatti et al., 2011; Kwon et al., 2014; Wang et al., 2019). One of the main parts of the study presented in Aharony et al. (2011b) was a fitness-centered intervention. Its principal aim was to analyze social influence and motivation in the context of health and wellness activities. For that purpose they tested different social strategies to try to encourage physical activity and implemented an accelerometer-based algorithm to estimate physical activity levels. The goal was not to discern the specific type of physical activity the subjects were performing but only to identify the intensity of the activity so they decided to implement a less accurate but more robust algorithm, which allowed more flexibility in the way participants could carry the phone. As described in Aharony et al. (2011b), the fitness-centered intervention was carried out between October and December 2010 and it was divided into three periods: a baseline period before the beginning of the intervention was officially announced (from October 5 to October 27); and two other periods after the intervention actually began (from October 28 to November 15, and from November 16 to December 5). A total of 108 subjects elected to participate, part of them assigned to a control group.

Along these lines, if we focus on the group of rules whose consequent indicates high (L) or very high (XL) accelerometer values and represent its numerosity evolution over time, we observe how a significant rise is presented for multiple subjects coinciding with the second and third periods of the fitness intervention. Some examples are included in Figure VI.4. It is worth mentioning that the use of linguistic labels to build the fuzzy sets allows the term "high or very high accelerometer" (also referred as accelerometer $L - XL$) to have an independent meaning for each subject. This meaning depends on the range of the accelerometer values for each particular subject. Figure VI.5 helps us to understand the distribution of the accelerometer values during the second period of the fitness intervention using one of the subjects included in Figure VI.4 as example. o special concentration of high or very high values are appreciated in Figure VI.5. Nor can we appreciate significant increases in the values registered. Thus, we can infer the higher number of rules with accelerometer L-XL in consequent is due to the fact that the algorithm finds association relationships that explain these accelerometer values and, consequently, the confidence of such rules increases and reaches the quality thresholds.

Delving deeper into the analysis of these same rules, we examine which relationships are being discovered by the algorithm for those values of accelerometer. Despite the fact that accelerometer has been already proved as a trustworthy indicator of the amount of physical activity (Aharony et al., 2011b; Puiatti et al., 2011; Kwon et al., 2014; Wang et al., 2019), examining those relationships can also reinforce the assumption that the increases in rule numerosity are not caused by other activities or types of phone usage.

Figure VI.6 shows the disaggregation of the antecedents of the analyzed rules into four different groups, i.e., each one of the antecedents of each individual rule is classified into one out of four possible groups: (1) antecedents related to weekday or time, (2) antecedents that indicate the phone is not being used much (few outgoing calls, few outgoing messages...), (3) antecedents meaning the phone is being used (several outgoing calls or messages, applications running...), and (4) other antecedents not directly related to



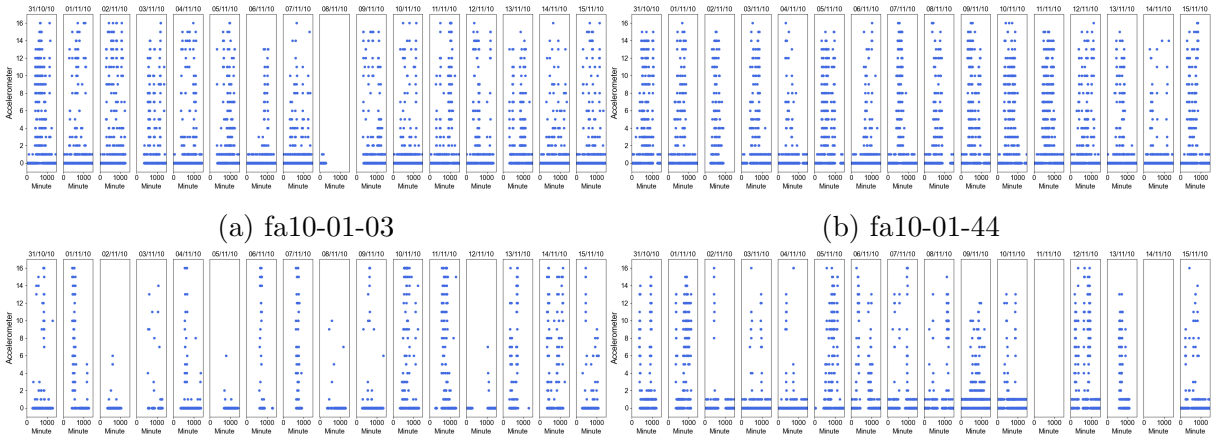
(a) fa10-01-03

(b) fa10-01-44

(c) fa10-01-82

(d) sp10-01-12

Figure VI.4: High-very high accelerometer consequent rules: numerosity evolution over time of rules (filtered by $supp \geq 0.2$ and $conf \geq 0.75$) with accelerometer L-XL in their consequent for four different subjects.



(a) fa10-01-03

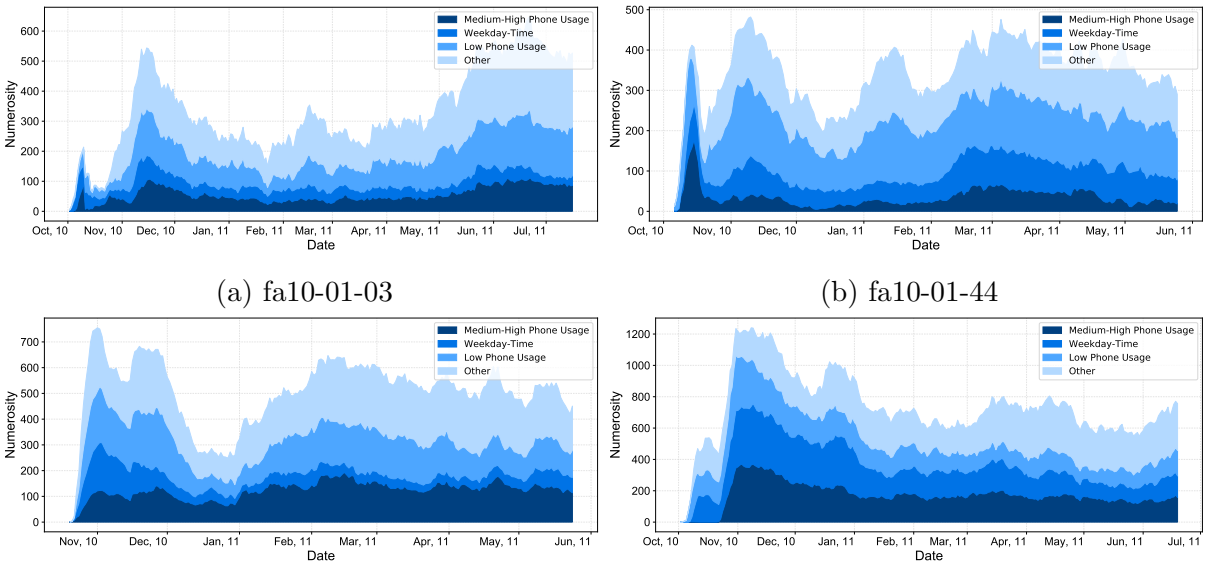
(b) fa10-01-44

(c) fa10-01-82

(d) sp10-01-12

Figure VI.5: Daily scatter plots of the accelerometer values (for four different subjects) during part of the period in which the highest numerosity of quality rules are observed with L-XL accelerometer in their consequent.

active usage of the phone (battery level, missed calls, incoming messages...). For instance, suppose the algorithm discovers the rule "If hour is S_4 and weekday is *Sunday* and outgoing calls is XS_5 then accelerometer is XL_5" and at a certain time stamp its numerosity is 4 (there are 4 copies of the rule in the pool). This means that two-thirds of the antecedents are related to weekday or time (first group) and one-third to low level of a certain usage (second group). Therefore, $\frac{2}{3} \cdot 4$ is added to the first group counter and $\frac{1}{3} \cdot 4$ to the second one, for that particular time stamp; and we continue doing the same for every rule that has accelerometer L-XL in the consequent at every time stamp.



(a) fa10-01-03

(b) fa10-01-44

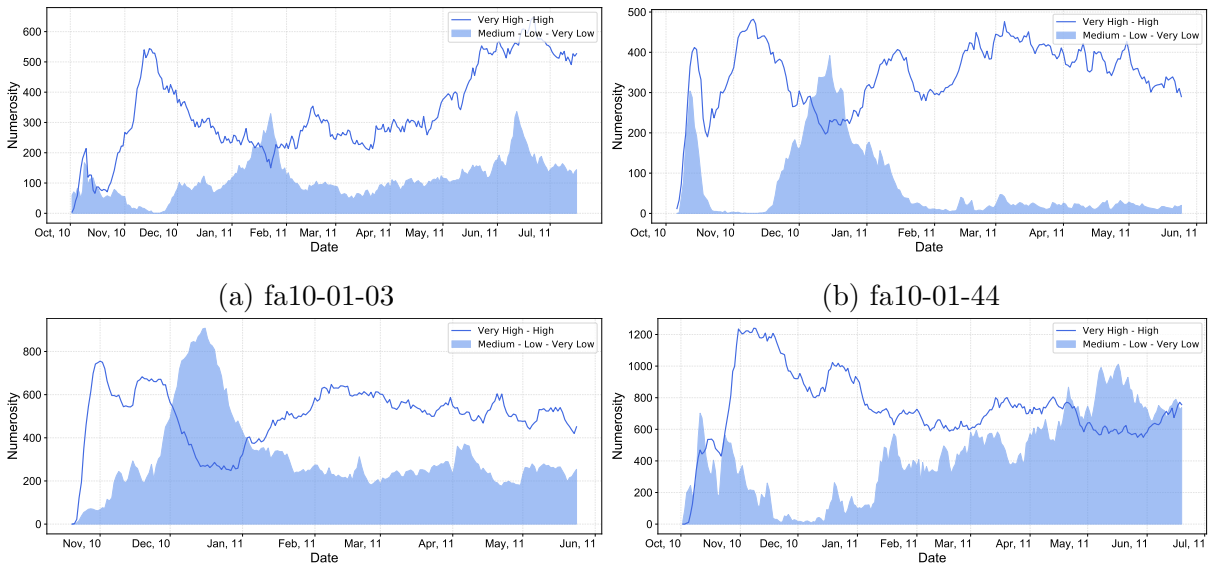
(c) fa10-01-82

(d) sp10-01-12

Figure VI.6: High-very high accelerometer consequent rules grouped by antecedents: numerosity evolution over time of rules (filtered by $supp \geq 0.2$ and $conf \geq 0.75$) with L-XL accelerometer in their consequent, distinguishing the proportion of antecedents that belong to one of three different groups: (1) weekday and time; (2) low phone usage, and (3) high phone usage.

Following this line of analysis, we also compare the evolution of the rules whose consequent indicates high or very high accelerometer values with those rules whose consequent indicates other accelerometer values. Figure VI.7 illustrates this comparison for the same subjects shown in the two previous figures. We can observe how the rise of the first group of rules (accelerometer L-XL) during the fitness-intervention (October 28 - December 5) does not overlap with a similar rise of those rules with other values of accelerometer. Indeed, the number of rules with other values of accelerometer is specially low during such period of time.

Nonetheless, accelerometer is not the only variable whose evolution in the association rules might be worth analyzing. Figure VI.8 shows the evolution of the accumulated numerosity of rules whose consequent implies high or very high amount of incoming/outgoing



(a) fa10-01-03

(b) fa10-01-44

(c) fa10-01-82

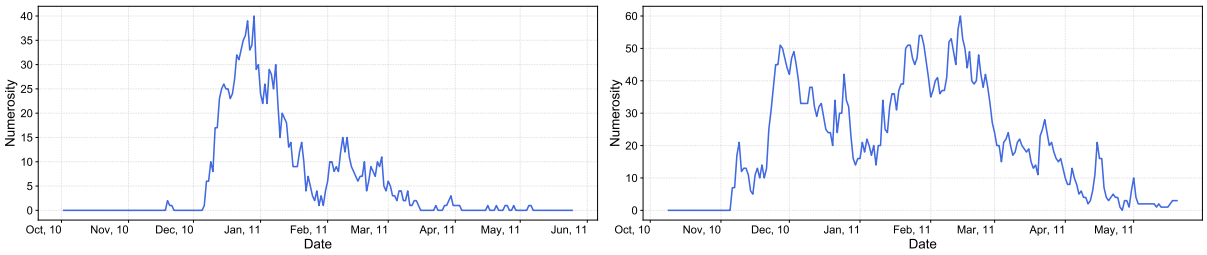
(d) sp10-01-12

Figure VI.7: High-very high accelerometer versus other accelerometer consequents: comparison between numerosity evolution of rules (filtered by $supp \geq 0.2$ and $conf \geq 0.75$) with L-XL accelerometer as consequent and of rules which have accelerometer as consequent with any other label (XS-S-M).

calls for two subjects. For the subject on the left, the high numerosities reached during part of December and January are specially noteworthy. During this period, several association rules explaining the high number of outgoing calls are discovered. Those same rules are not sufficiently confident outside this specific period. Hence, we can infer the subject is behaving differently during such period. Note that numerosity reaches its highest levels around Christmas Day. In the case of the subject to whom the right plot corresponds, confident enough rules are found during almost the entire duration of the study. However, significant increases and decreases in numerosity can be noticed. A great part of the patterns that are true at a given time may not continue being true after a while. In addition, some of the numerosity peaks match special dates, such as Christmas or Saint Valentine's Day. These are just two more examples of how monitoring the evolution of certain groups of association rules can help detect changes in behavioral patterns.

VI.1.4.3 Correlation with emotional state

Several studies have explored the use of wearable sensing data to obtain information about mental health or emotional state, being some of them able to discover interesting connections (Puiatti et al., 2011; Frost et al., 2013; Osmani et al., 2013; Gravenhorst et al., 2015; Muaremi et al., 2014). In Sanchez-Valdes and Trivino (2015) the level of physical activity is monitorized through smartphone accelerometer to provide therapists with an



(a) fa10-01-13 (b) fa10-01-35

Figure VI.8: High-very high incoming/outgoing calls rules: numerosity evolution of rules (filtered by $supp \geq 0.05$ and $conf \geq 0.75$) with (a) L-XL outgoing calls and (b) L-XL incoming calls in their consequent for two different subjects.

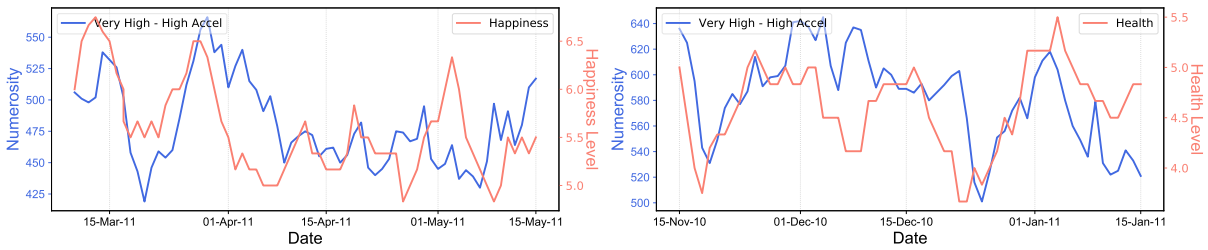
objective and quantitative measurement tool to be applied when developing patient-specific treatments.

As mentioned before, in addition to the information collected through the mobile phone sensing platform, the data collection from the *Friends and Family* study also included information gathered by means of surveys about different aspects of the lives of the participants. Part of these surveys could be daily completed by the participants from their own phones. These daily on-phone surveys asked the subjects to assess quantitatively different variables related to their mood, such as happiness, stress, productivity or health. A numerical scale from 1 to 7 was used to evaluate each variable, where 1 is the lowest level and 7 the highest. In addition, these surveys also asked the participants about the number of hours they spent sleeping or hanging out.

In Bogomolov et al. (2013) and Bogomolov et al. (2014) the mobile phone sensing data part of the *Friends and Family* data collection is employed to predict happiness and stress. Although this is done in an offline a posteriori process and not from a data stream perspective, these studies can still be seen as an indicator of the possible links between smartphone usage and subject's mood. In the next part of our analysis, we incorporate survey information as a way to complement the results obtained by Fuzzy-CSar-AFP. Hence, we analyze possible correlations between the evolution of certain rules and the answers to the surveys.

Figure VI.9 shows together the evolution of the numerosity of rules whose consequent is accelerometer L-XL and the evolution of two aspects assessed by the surveys (happiness and health). In both two graphics it is possible to appreciate certain parallelism between both evolutions. Certainly, such correlation is not perfect, a person perception of his/her level of happiness or health can be influenced by a bunch of factors. Furthermore, happiness (as well as, health sensation, stress, productivity...) is habitually not expressed in absolute terms. That is, each time a person reaches level 5 of happiness does not mean that this person is just as happy as the past times that he/she said his/her happiness was at level 5. The scale changes with the pass of time and the context. Every time we say "it is the happiest day of my life" we would have to look back and correct all our previous feelings of

happiness. What is really meaningful is trend. Increases or decreases in happiness (health sensation, stress, productivity...) is what really matters, and these trends show sync with the analyzed groups of rules.

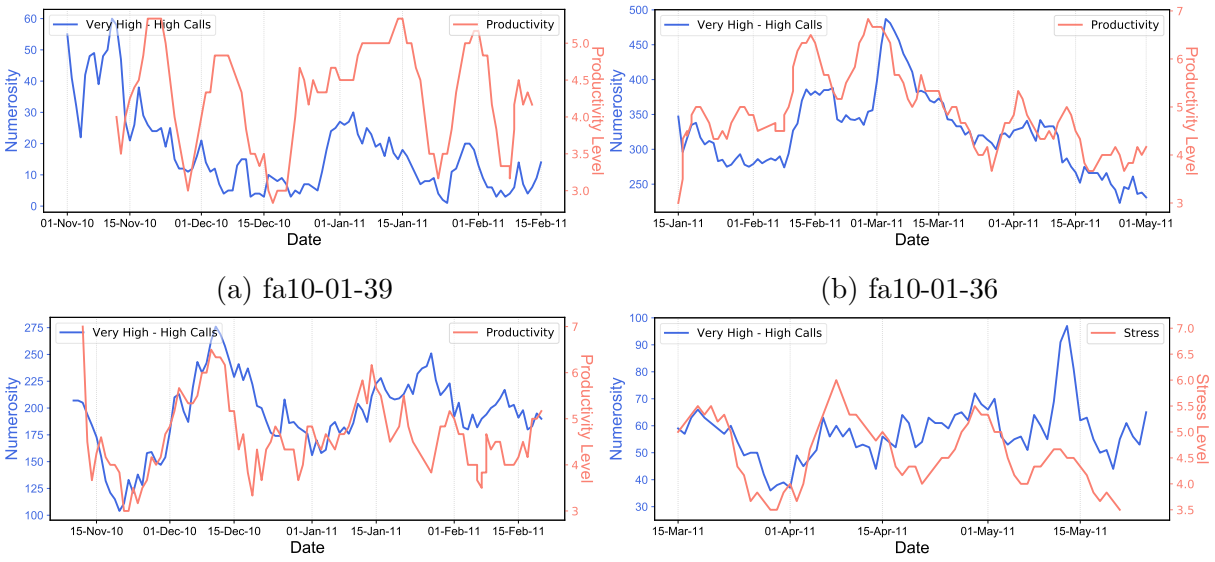


(a) sp10-01-34 (b) fa10-01-45

Figure VI.9: High-very high accelerometer, happiness and health: numerosity evolution of rules (filtered by $supp \geq 0.2$ and $conf \geq 0.75$) with L-XL accelerometer as consequent along with evolution of (a) happiness and (b) health perception gathered through surveys.

But this is not the only group of rules whose evolution seems to present similarities with subjective perceptions collected through the surveys. Figure VI.10 contains four graphics showing the numerosity evolution of those rules whose consequent implies a high or very high number of calls along with the evolution of productivity perception ((a)-(c)) and stress (b). Once more, similarity between both lines for all four subjects is noteworthy. This would point out that for the first three subjects some of the moments with higher sensation of productivity coincide with some of the occasions when more high-confidence patterns related to high number of calls are being detected by the algorithm (and equivalent for the most stressful periods in the case of the fourth subject).

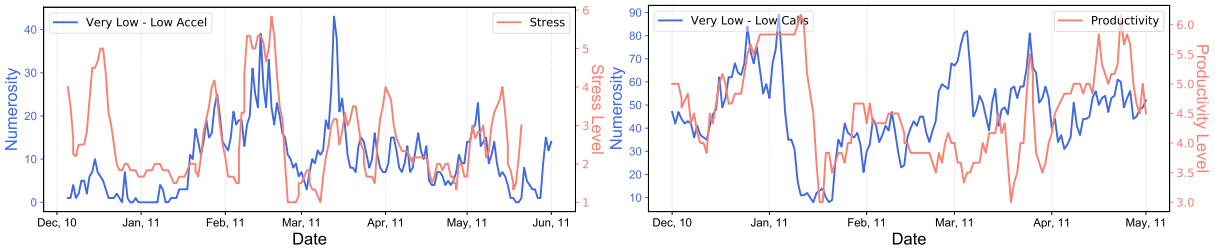
As mentioned before, thanks to the use of fuzzy labels the concept of "high number of calls" can be different for each subject, the algorithm does not assume any a priori data structure and adapts itself to each particular case. This allows to discover completely different sets of rules for different subjects. Figure VI.11 includes two more plots showing the numerosity evolution of a certain set of rules along with that of the answers given by the subject to a specific question of the daily surveys. The left plot illustrates this for those rules with consequents point out low levels of accelerometer and for the evolution in the level of stress claimed in the surveys. The right plot represents, for a different participant, the evolution of those rules whose consequent refers to low amount of calls together with the evolution of the productivity perception. If we delve into the meaning of the parallelisms detected here between groups of rules and subjective aspects, they are found to be logical. Taking accelerometer as an estimator of physical activity, it seems reasonable that periods of higher physical activity coincide with periods of greater happiness or health sensation. In the same way, the parallelisms between high number of calls and productivity or stress, between low physical activity and stress, or low number of calls and productivity, seem equally reasonable.



(a) fa10-01-39 (b) fa10-01-36

(c) sp10-01-30 (d) sp10-01-42

Figure VI.10: High-very high number of calls, productivity and stress: numerosity evolution of rules (filtered by $supp \geq 0.05$ and $conf \geq 0.75$) with L-XL calls as consequent along with evolution of ((a), (b), (c)) productivity and (d) stress perception gathered through surveys.



(a) fa10-01-24 (b) fa10-01-12

Figure VI.11: Numerosity evolution of rules (filtered by $supp \geq 0.2$ and $conf \geq 0.75$) with (a) XS-S accelerometer and (b) XS-S calls as consequent, along with the evolution of (a) stress and (b) productivity perception gathered through surveys.

VI.2 Real-Time relational analysis on Twitter

Twitter is a clear example of the continuous generation of huge amounts of chronologically ordered data. These data contain interesting and very rich information that can be analyzed. The immediacy of social networks should also be transferred to the automatic analysis of their information, providing answers that explain what is happening, when it is happening. If we add to this a clearly non-stationary nature where reality is constantly evolving and changing, we find ourselves with an ideal laboratory for analyzing data stream mining algorithms. We focus on tweets related to political events because politics usually generates a lively debate that adapts to different circumstances that could be considered as concept drift. We analyze two use cases. On the one hand, a dataset concerning the 2016 U.S. presidential election. On the other, the 2019 investiture process in Spain. For the latter, we analyze the messages on that topic published during July and August 2019.

The application of data mining techniques on data from social networks can reveal patterns about the individuals immersed in the shared environment and produce knowledge that was previously not feasible to find due to the variety and complexity of the information. Thus, these social media have become a fundamental element to be considered by large companies when analyzing the quality of their products, defining their marketing strategies and even at the time of decision-making (Chamlertwat et al., 2012).

Twitter stands out as the most prominent social network for obtaining concise and accurate information over time. Moreover, the micro-blogging structure provides an easy mechanism for communication and discussions between users, as well as a tool within marketing strategies for companies (Java et al., 2007). Since its appearance in 2006, it has emerged as a massive social network. With more than 180 million active users and 656 million tweets per day at the end of 2020. The ease and speed with which users can share their opinions and feelings makes Twitter a channel of communication with an ever-increasing role in socio-political areas.

The information generated on Twitter is extremely abundant, making it an appealing source to collect data in real time. However, the format in which the collected information is presented entails some challenges. This is mainly due to the fact that the tweets are written in natural language, so they do not have an adequate structure to be used in a straightforward way to obtain quality knowledge. Therefore, although our main objective will be the real-time analysis of frequent associations, this requires the processing and simplification of the text present in the tweets studied. In this context, text mining and natural language processing (NPL) techniques are very useful tools that allow us to simplify and standardize the text of the tweets.

Thus, we need to apply an important preprocessing consisting of tokenization (splitting the sentence into words), lowercase transformation, cleaning of repeated characters (used by users for emphasis), *stopwords*, punctuation marks, numbers and links, lemmatization (to give common form to words with different morphological and verbal derivations), *stemming* (reduction of the word to its base or root), *part-of-speech* tagger

(tagging of words to indicate their grammatical category, adding in our case the entities user and hashtag) and, finally, identification of N-grams (groups of words that have a meaning of interest as a whole). In addition, we will also try to analyze the positivity or negativity of the tweets considered in order to be able to detect the presence of approval or repulsion towards specific political terms or parties.

Social media and especially Twitter are playing an increasingly important role in modern societies. Although there are different topics with great appeal, Twitter is closely linked to politics. Indeed, political parties themselves use this platform actively to convey their ideas, being a key medium in election campaigns.

The importance of social media in election campaigns was demonstrated after Barack Obama's successful campaign for the 2008 White House presidential election (Williams and Gulati, 2008). Therefore, during the last decades, research has been conducted focused on analyzing the use of Twitter as a medium to promote ideas or debate political issues (Vilares et al., 2015). Thus, most of the research based on tweets has focused on the prediction of election results (Ceron et al., 2014; Golbeck and Hansen, 2011), as well as the monitoring of election campaigns, mainly in the United States (Jensen and Anstead, 2013; Wang et al., 2012), although research has also been conducted in Europe (Tumasjan et al., 2010; Caldarelli et al., 2014).

VI.2.1 Text mining

A significant portion of the data generated by social networks is textual. In this context, Text Mining arises within a new paradigm, called KDT (Knowledge Discovery in Textual-Databases), which was introduced in Feldman and Dagan (1995) and is different from the existing KDD (Knowledge Discovery in Databases), which was focused on the search for patterns within completely structured data. Thus, this new approach aims to analyze and extract useful and non-explicit information from textual data, lacking structure and homogeneity, in which the attributes are unknown, which requires the use of machine learning algorithms on textual data (Bloehdorn et al., 2011).

Thus, Text Mining is a subcategory of Data Mining, consisting of obtaining information and discovering and identifying patterns, entities and relationships in unstructured text data. This subcategory is relatively new and its interest and usefulness is increasing significantly. In addition, it is a highly interdisciplinary area, which combines other important branches such as natural language processing, information retrieval and information extraction, as well as the use of machine learning techniques.

The structuring of textual data by means of Text Mining techniques makes it possible to give the text a form that allows it to be studied by means of common machine learning techniques.

Text preprocessing is one of the essential phases in any text mining task. Textual communication offers the user a wide range of possibilities and a great freedom of expression, which makes texts data with a greater amount of intrinsic information. However, this

freedom means that there is not a common solution for text preprocessing, but that there are several possibilities and each type of text requires an appropriate analysis (Rangu et al., 2017). Thus, Text Mining has its own preprocessing techniques, not applicable in other areas, and tremendously characteristic (Moreno and Redondo, 2016). In the following, we list the preprocessing techniques used in this application:

- **Tokenization**. Tokenization appears as a technique that allows to divide every sentence into words, making these as the minimal meaningful entity (Hofmann and Chisholm, 2016). This minimal entity is known as a token.
- **Transformation to lowercase**. Transforming texts to lowercase is a common text mining preprocessing technique.
- **Elimination of repeated characters**. There is a tendency for users to seek emphasis by repeating (more than twice) the same letter within a word successively (e.g., "Greeeat" in English or "Bieeeen" in Spanish). The objective is to eliminate the distortion that this produces, identifying the actual word without the repeated characters (i.e., "Great" or "Bien").
- **Filtering stopwords**. Stopwords serve as connectors, quantifiers and other functions (e.g., pronouns, prepositions, etc.), which provide practically no additional information to the content of the text in terms of semantics. It is a common measure to filter such stopwords.
- **Removal of numbers and punctuation marks**. In the context of association analysis and sentiment analysis, numbers do not seem to be an element that is worth taking into account. On the other hand, the removal of punctuation marks is a technique commonly applied during preprocessing. This is because it results in a significant reduction of the problem, although sometimes the presence of punctuation marks can be interpreted as intensifiers (Effrosynidis et al., 2017).
- **Link filtering**. The presence of links is a common element in different types of texts. Although there is the possibility of accessing the linked web, obtaining the text contained in it and associating it with the tweet in question, we considered that this would complicate the search for associations without providing a clear benefit. In addition, it would defeat the spirit of analyzing short text, so we decided to filter such links.
- **Lemmatization**. This technique allows to give a common form to words with different morphological derivations (gender and number) and different verb forms. The target is to obtain the morphological information of the word and focus the study on its semantic information. Lemmatization is a costly technique, since it requires a deep knowledge of the linguistics of the language in question. However, its usefulness is remarkable, since it involves a substantial reduction of noise, improving the results of Stemming, although taking a longer execution time.

- **Stemming**. The aim of Stemming is also to transform a word to a common form, although in this case it is achieved by reducing the word to its base or root form. Thus, once again, the semantic section of the word is emphasized, forgetting its morphology, allowing the dimensionality of the problem to be reduced.

- **Named-Entity-Recognition**. It allows the extraction of proper entities in the text, such as names of people, organizations, locations, expressions of times, etc. This allows the addition of a large amount of semantic knowledge that helps to understand the subject of a text.

- **POS (Part-Of-Speech) Tagger**. It is a technique used to indicate the grammatical category of each of the words within the sentence. Thus, this technique is used to confront words with several possible meanings. It analyzes the position of the word in the sentences to define the real meaning of the word, checking its situation and comparing it with frequent grammatical sequences in different languages. In general, the usefulness of this type of techniques for micro-blogging texts (Twitter case) is debatable, mainly due to the brevity of the text (Go et al., 2009; Kouloumpis et al., 2011). In this particular case, it is applied with the objective of eliminating words that are not verbs, adjectives, names, entities, users or hashtags, thus reducing the dimensionality of the problem.

- **Identification of N-grams**. We define a N-gram as a series of words that can be grouped together because they have a meaning of interest that makes their appearance in the database specially frequent. In the case of association analysis, this technique becomes a fundamental point since, if any N-gram is not identified, it could lead to the appearance of an association between the terms that make up the unidentified N-gram. This could hide other associations of interest.

VI.2.2 Sentiment analysis

Sentiment analysis or opinion mining is a research area within the field of Natural Language Processing (NLP). It is a discipline whose main objective is the automatic detection of opinions, feelings and subjectivity within a text. Thus, it pursues the recognition of the emotions behind the analyzed texts, seeking to determine the polarity of the text and its strength.

Thus, different types of techniques are applied with the aim of representing the subjectivity or opinion existing in a text by means of a quantitative value on a certain scale, which makes it possible to classify such text in a certain type of feeling or opinion. In this context, an opinion is a positive or negative evaluation about a product, service, organization, person or any other type of entity about which a given text is expressed. Nowadays, the popularization of micro-blogging social networks such as Twitter has increased interest in this area, so that it is a question of achieving real-time monitoring of the opinions of thousands of people.

When performing sentiment analysis, three possible levels of study of a text are distinguished. These three levels are defined on the basis of the granularity, depth and detail required:

- **Document-level analysis**: The overall sentiment of a document is analyzed as an indivisible whole. It is assumed that the document expresses an assessment of a single entity.
- **Sentence-level analysis**: The document is divided into individual sentences in order to subsequently extract the opinion contained in each one of them.
- **Aspect-level analysis**: An entity is considered to be made up of different elements or aspects and an opinion is expressed on each of them, the polarity of which may be different in each case.

Given the brief nature of the tweets, in this case we consider a document-level analysis, assuming that each message expresses the opinion of the user, taking that there is only a single entity.

Although the number of tools available for sentiment classification in English is remarkable and there are some quite accurate ones, for the case of other languages not everything is so advanced. Therefore, when facing sentiment analysis for the case of tweets about Spanish politics there appears the difficulty of making an adequate determination of the sentiment resulting from the messages. Therefore, the analysis of this problem for other languages has become a topic of interest in research. In the case of Spanish, novel analyses have appeared over the last decade (Brooke et al., 2009; Martínez Cámara et al., 2011).

In this case, we analyzed different options including: the use of Lexicons of words labeled as positive and negative; the use of a corpus to create a learner ensemble capable of classifying tweets; and the use of existing software tools capable of performing sentiment classification of a text. In general, such software tools have been trained using a corpus of messages or clusters of lexicons. The existing tools analyzed are: senti-py (Hofman, 2018), MeaningCloudClassifier (MeaningCloud, 2018), AutoCop, SentiStrength (Culpeper et al., 2018) Lingmotif (Sentitext) (Moreno-Ortiz and Hernández, 2013), and apiculture (Sogo, 2018). After studying the advantages and disadvantages of each of them, we chose to use SentiStrength (Culpeper et al., 2018). This decision is also supported by the results of several theoretical studies (Thelwall et al., 2010; Gonçalves et al., 2013).

VI.2.3 Experimental setup

As mentioned, the study conducted has been applied on two datasets, both with political character: one focused on the 2016 US political elections and the other one related to the Spanish investiture process during the summer of 2019. The first is a smaller dataset

and already labeled, so it has been used for the initial study. The second one presents a dataset with real data, extracted from Twitter for one month. Thus, we proceed to describe the study carried out, together with the different considerations taken. After that, the corresponding results obtained are presented.

In both cases, the same experimental scheme is followed. The aim has been to follow a fixed analysis structure, starting with text processing and sentiment analysis (if required), and then moving on to data stream analysis in search of association rules. For this last step, two paths are explored. Figure VI.12 summarizes the sequence of steps involved in this experimental scheme.

As shown in the figure, the first step is to perform a language filtering of the tweets if necessary. For this, a language parser is applied, so that if the message is considered, with a sufficiently high probability, to be in a different language than expected, it will be labeled as *foreign* using a *Language* variable, based on which a subsequent filtering is performed.

The next step is to check if the tweets have a sentiment tag associated with them. If not, the SentiStrength classifier (Vilares et al., 2015) is applied, which allows extracting the sentiment present in the text and reflecting it in the Sentiment attribute. Of the two use cases addressed, these first two steps will only be necessary for the collection of tweets about the 2019 Spanish investiture process.

Once, if necessary, the language filtering and sentiment identification have been completed, a text analysis is applied on the tweets. Thus, we started by removing links, numbers, punctuation (except for the # and @ that refer to hashtags and users, respectively), emoticons and laughter from the messages. We have also eliminated the appearance of the same letter multiple times (more than two) and consecutively, reducing these to simply two consecutive appearances.

Next, SpaCy (Honnibal et al., 2020) was used as a tool for text analysis. Therefore, we started by eliminating stopwords and short words (length less than 3). Likewise, the POS-Tagger technique was applied, performing a grammatical tagging on the words. In the case of hashtags and users, two new grammatical tags have been defined. Subsequently, lemmatization was applied to the text.

Following this, a simplification of the text is performed by eliminating certain words based on their grammatical category. While verbs tend to quantify the activity-passivity in the content of the message, adjectives tend to be useful in identifying the feeling present in the text. Since this approach aims to find relationships between terms, only nouns and entities are relevant, in addition to users and hashtags. However, since adjectives can sometimes be useful in discerning concepts, they have also been finally considered. When working with texts in Spanish, certain errors were detected in the SpaCy tagger that cause some nouns to be wrongly identified as verbs. This led us not to remove verbs in the case of the collection of tweets related to the investiture process in Spain in 2019.

The next step is the analysis of N-grams in the text, mainly looking at bigrams

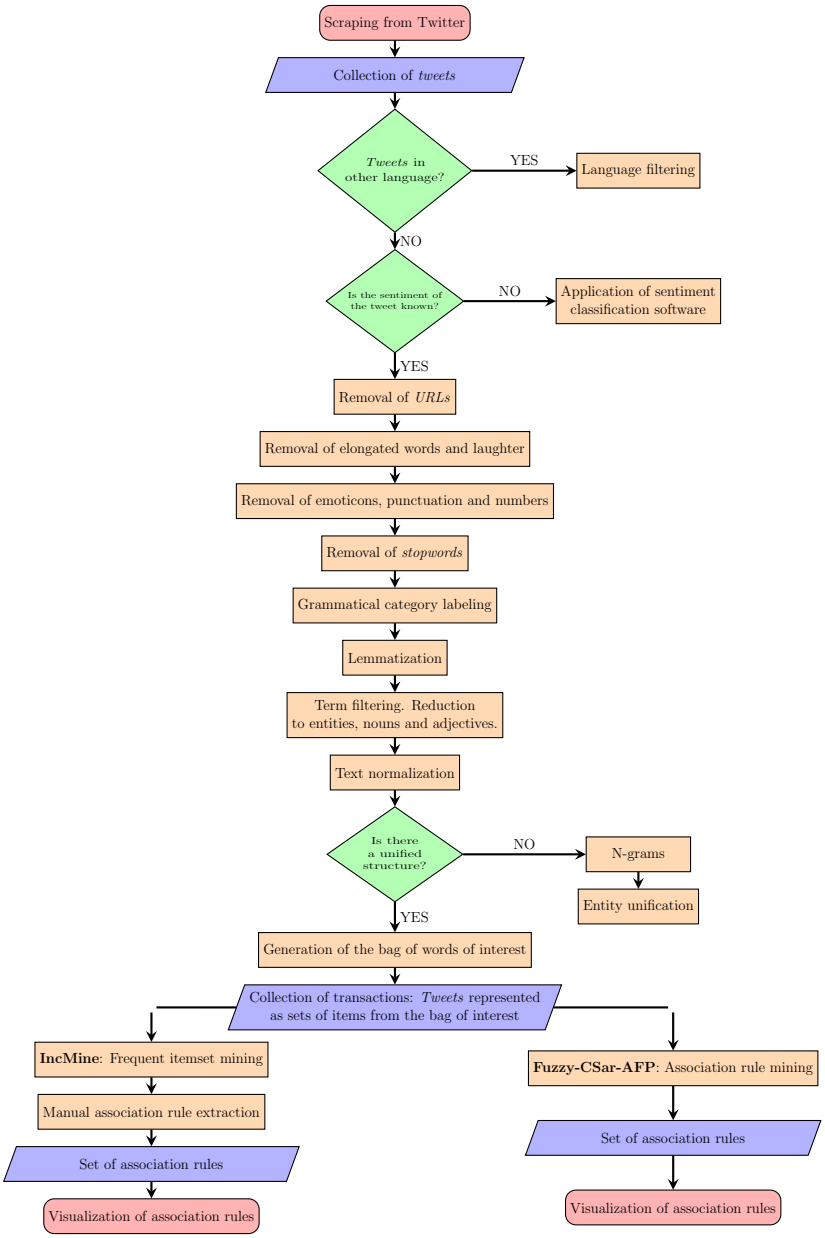


Figure VI.12: Flow chart of the experimental scheme followed in the study.

and trigrams. After that, we proceeded to the unification of entities, since there are often different ways of citing or referring to entities of interest in the dataset using different terms.

Once the terms and elements of the messages in the database had been unified, we proceeded to create a bag of elements of interest, in which interesting terms were defined within the database based on the concept to which they refer and their frequency of occurrence. The elements included in the bag have been considered of certain relevance in the debates conducted in the set of tweets analyzed.

After that, each tweet in the database is represented by a sequence of words in the bag of interest. Thus, at this point each tweet can be seen as a transaction, while each of the words in the bag of interest is seen as an item, which can appear in sets, forming itemsets. Therefore, it is possible to work on this set of transactions with the terms of interest in search of the frequent itemsets and the association rules existing between the different terms.

At this point, once we have managed to structure the textual data and generate a transactional database based on the representation of the tweets by the terms of the defined bag of interest, we move on to the last stage of the process. In this stage, the association rules between terms will be extracted, performing real-time learning by applying data stream mining algorithms. As shown in Figure VI.12, two different approaches are explored in this stage.

On the one hand, an approach based on the incremental extraction and update of frequent closed itemsets (FCIs) over data streams is applied using IncMine (Cheng et al., 2008). Performing the extraction of association rules from these frequent itemsets in an offline fashion.

On the other hand, it is possible to perform the direct extraction of the association rules by applying a fuzzy and completely online technique, as is the case of the Fuzzy-CSar-AFP algorithm (Ruiz and Casillas, 2018).

After collecting the rules, they are analyzed to look for links between the presence of terms and sentiments. Thus, a count of the number of rules generated for each antecedent plus consequent pair will be made, seeking to analyze the existence of term-consequent relationships (where the consequent is the sentiment in this case). For this, the rules will be filtered to keep only quality rules.

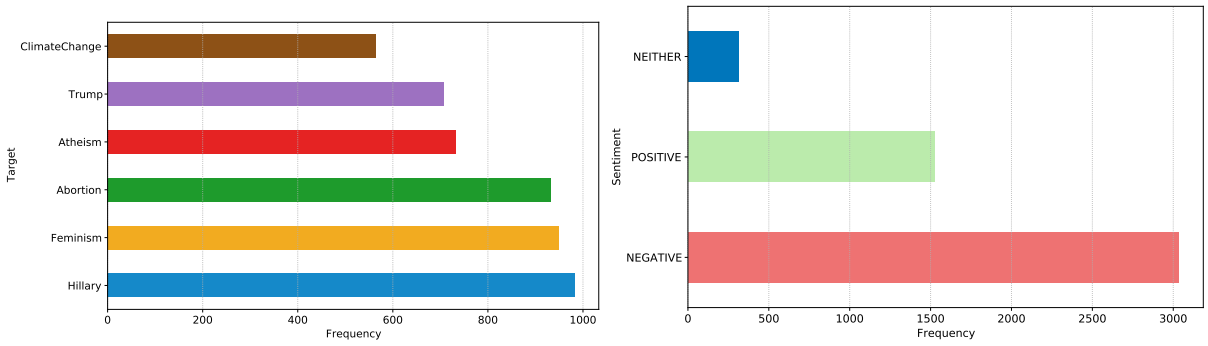
VI.2.4 Case 1: 2016 United States presidential election

For this case study, the SemEval-2016 Stance Dataset, related to the 2016 US political elections, is used (Mohammad et al., 2017). In this dataset each of the tweets already comes with an associated sentiment label. Specifically, the attributes associated with each of the tweets are: (1) *ID*, identifying number of the tweet; (2) *Target*, unit of interest in which the tweet encompasses; (3) *Tweet*, content of the tweet; (4) *Stance*, refers to

the attitude of the user who wrote the tweet; (5) *Opinion_towards*, this is the target of the opinion expressed in the tweet; and (6) *Sentiment*. Each of the last three take three possible values. In the case of Stance the possible values are: *Favor*, the user supports the objective; *Against*, the user is against the objective, and *None*. As for the objective of the opinion expressed (*Opinion_towards*), it can be the entity of interest itself (*Target*), other (*Other*) and none, in case the purpose of the text is not to give an opinion but to provide other information. Finally, three possible sentiments are considered: positive, negative or neither.

VI.2.4.1 Exploratory data analysis

Figure VI.13 shows the distribution of the values of the *Target* and *Sentiment* variables. We can observe that while three targets are specially frequent, the overall set of tweets is fairly evenly distributed among the different targets and no target is extremely rare or infrequent. However, regarding sentiment, negative messages clearly predominate and there is a very small amount of neutral messages.



(a) (b)

Figure VI.13: Plots with the distributions of the (a) target and (b) sentiment present in the tweets to be studied.

The left plot of this figure summarizes the distribution of topics (targets) in the total collection of tweets. However, it does not take into account the temporal component. The distribution of the topics of interest may not be constant over time. A multitude of factors, such as news published in the media or statements made by public figures, can cause the popularity of topics to vary over time. The evolution of the frequency of appearance of these targets is shown in Figure VI.14.

We can see how, indeed, the popularity of the different targets varies over time. Thus, the topics “Trump”, “Hilary”, “Atheism” or “ClimateChange” are prominent in the conversation throughout the entire period analyzed. However, “Feminism” and “Abortion” only gain relevance towards the latter part of the period

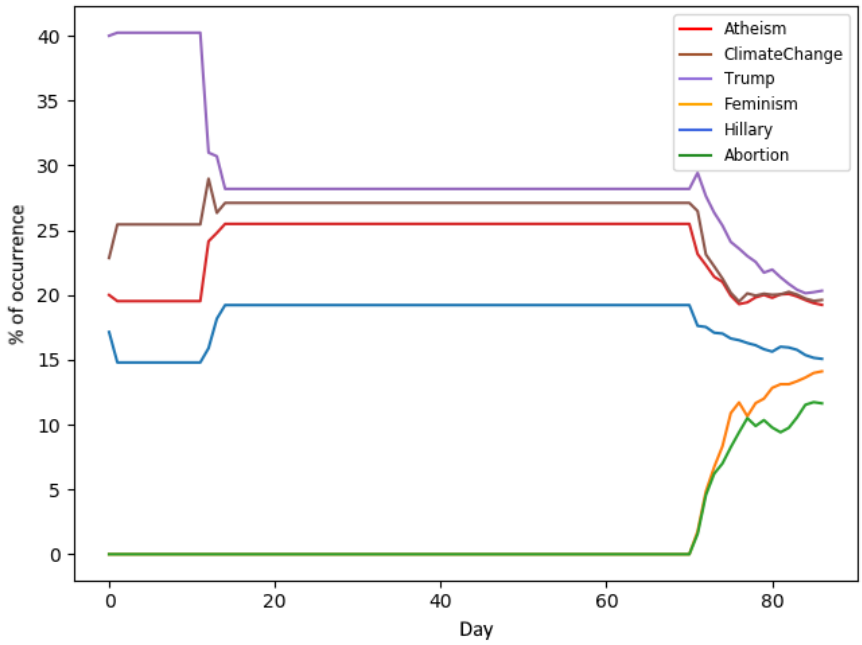


Figure VI.14: Temporal evolution of the frequency of publication on the topics of interest present in the tweets related to the 2016 US election.

VI.2.4.2 Natural language processing

As described in Section VI.2.3, before applying the algorithms for frequent itemset extraction and association rules, we perform a series of preprocessing steps that culminate in obtaining a bag of terms of interest that we use to represent the tweets in transactional format.

First, links, numbers, punctuation marks (except # and @), emoticons, laughter and repeated letters are removed from the text. Likewise, stopwords and short words are also removed; certain words are discarded based on their grammatical category, and lemmatization is applied to standardize the text.

After that, also as part of the preprocessing stage, we analyzed the N-grams present in the tweets, focusing on bigrams and trigrams. After analyzing the trigrams obtained, the only term of interest is anti_choice_law, which refers to the law against free abortion. In the case of bigrams, a number of concepts have been found, which are shown in Table VI.2.

After this, we proceeded to the unification of entities, with the aim of identifying the different ways of citing or referring to the same entity of interest. Tables VI.3 and VI.4 show, respectively, the different entities considered and unified terms along with the terms that each one encompasses.

Once the terms and entities of the messages in the database were unified, we

Bigram	Concept
(gay, marriage)	Gay marriage.
(supreme, court)	U.S. Supreme Court.
(united, states)	United States.
(birth, control)	Birth control.
(marriage, equality)	Marriage equality.
(confederate, flag)	Confederate flag.
(sea, level)	Sea level.
(david, attenborough)	British broadcaster and natural historian.
(human, being)	Human being.
(social, medium)	Social medium.
(mother, teresa)	Mother Teresa of Calcuta.
(death, penalty)	Death penalty.
(gender, equality)	Gender equality.
(preborn, child)	Preborn childs.

Table VI.2: Table with the different bigrams found in the case of the US presidential election and their corresponding concept.

Entity	Associated terms
donald_trump	*@realdonaldtrump*, *donald_trump*, *mr_trump*, *trump*, *#trump*, *donaldtrump*, *#donaldtrump*
hillary_clinton	*#hillaryclinton*, *hillaryclinton*, *hillary*, *clinton*, *hilary*, *#stophillary*, *#stophillarypac*, *hillary_clinton*, *#readyforhillary*, *#nohillary*, *#killary*
barack_obama	*@barackobama*, *barackobama*, *obama*, *#obamas*
democrat	*@thedemocrats*, *@vademocrats*, *democrat*, *#democrats*, *#democrat*
republican	*republican*, *#republican*, *#republicans*, *#republicanvalues*, *republicans*
USA	*unitedstates*, *united_states*, *usa*, *@unitedstates*, *#unitedstates*
SCOTUS	*scotus*, *#supremecourt*, *supremecourt*, *#scotus*, *supreme_court*
david_attenboroug	*david_attenboroug*, *@sir_attenboroug*, *davidattenboroug*, *attenborough*

Table VI.3: Table with the different entities considered and the terms included in them.

proceeded to create a bag of items of interest. The list of words included in the bag of interest is gathered in Table VI.5.

These elements of interest are the items in our transactional database. Figure VI.15 shows them within a word cloud and Figure VI.16 represents the frequency of each of them. In both figures, we can observe how the two presidential candidates appear with a high frequency, along with the hottest topics and their most discussed policy proposals.

VI.2.4.3 Real-time association rule analysis

On the database obtained after preprocessing, in which each tweet is represented in transactional format by a set of items included in the bag of interest, we conducted an association rule analysis on data streams.

In the following, the results obtained through two different approaches are presented and analyzed. In the first case, the IncMine algorithm is used to mine FCIs and, from these

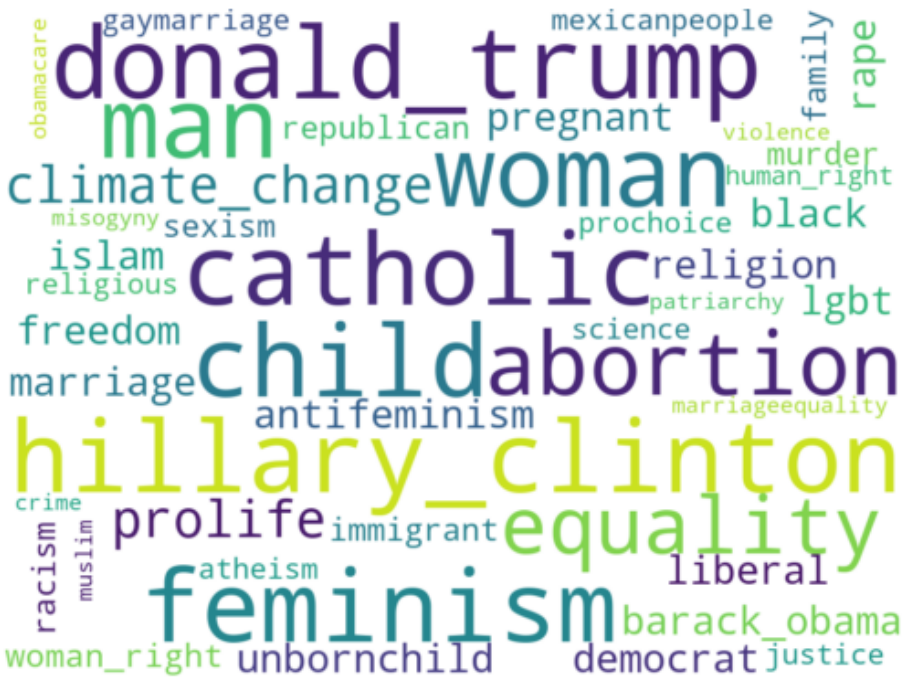


Figure VI.15: *Word cloud* with the words chosen for the interest bag of the US election problem.

Concept	Associated terms
prolife	*prolife*, *pro_life*, *prolifeyouth*, *prolifegen*, *alllivesmatter*
prochoice	*prochoice*, *pro_choice*, *prowomanchoice*
feminism	*feminist*, *feminism*, *genderequality*, *gender_equality*
antifeminism	*antifeminism*, *antifeminist*, *feminazi*, *notafeminist*, *meninist*, *spankafeminist*
misogyny	*misogyny*, *misogynism*, *misogynst*, *misogynist*, *misogynistic*, *misogynyisugly*
immigrant	*immigrant*, *immigration*, *latino*, *hispanic*
child	*child*, *kid*, *baby*
unbornchild	*unborn_child*, *preborn_child*, *unborn*, *preborn*, *fetus*
pregnant	*pregnant*, *pregnancy*
equality	*equality*, *equalityforall*, *equal_right*, *equalright*, *eaquality*
marriageequality	*marriageequality*, *marriageequaility*, *marriage_equality*
gaymarriage	*scotusmarriage*, *gaymarriage*, *gay_marriage*
woman_right	*woman_right*, *womensright*
lgbt	*lgbt*, *gay*, *homosex*, *homosexual*, *homosexuality*, *lesbian*
rape	*rape*, *rapeculture*, rapist, *maritalrape*, *maritalrapedebate*
sexism	*sexism*, *sexist*
racism	*racism*, *racist*
climate_change	*climate_change*, *global_warming*, *climate*, *climatehope*, *climatenexus*, *emission*, *climatechangeisreal*, *mychangeforclimate*, *ecologyaction*
freedom	*freedom*, *liberty*
science	*science*, *sciencerule*, *scientist*
anti_choice_law	*anti_choice_law*, *antichoice*
catholic	*catholic*, *romancatholic*, *church*, *christian*, *christ*, *christianity*, *bible*, *jesus*, *teamjesus*
atheism	*atheism*, *agnostic*, *atheist*, *atheistq*
islamic	*islamic*, *islam*, *isis*, *islamicstate*
man	*man*, *male*
woman	*woman*, *girl*, *female*, *yesallwoman*, *yesallwomen*
mexicanpeople	*mexicanpeople*, *wearemexico*, *mexico*, *mexican*

Table VI.4: Table with the different terms of interest unified under the same concept.

FCIs, all possible rules are generated offline. In the second case, the Fuzzy-CSar-AFP algorithm will be applied, which directly obtains and evolves the association rules in a fully online way. In both cases, we rely on different visualizations to analyze the rules obtained.

Our objective is not to compare the performance of both algorithms but to analyze two possible ways of dynamically obtaining association rules from a stream of tweets. In any case, the comparison between the performance of both algorithms is not directly applicable since the quality measures of the rules (support, confidence, lift, etc.) are different in one algorithm and the other. To perform a fair comparison between both methods a more in-depth study would be required.

Table VI.5: Set of elements of interest considered for the 2016 US presidential election database.

hillary_clinton	woman	donald_trump	feminism	man
catholic	child	abortion	equality	climate_change
prolife	barack_obama	rape	unbornchild	religion
islam	pregnant	freedom	marriage	liberal
democrat	black	antifeminism	woman_right	republican
family	racism	sexism	immigrant	murder
mexicanpeople	science	religious	gaymarriage	prochoice
atheism	justice	human_right	marriageequality	crime
obamacare	misogyny	patriarchy	violence	muslim
anti_choice_law	conception	humanist	lgbt	birth_control

Association stream mining with Fuzzy-CSar-AFP

Unlike IncMine and most frequent pattern mining methods from data streams, the Fuzzy-CSar-AFP algorithm allows direct mining of association rules. Instead of focusing on the identification of frequent itemsets and relegating rule construction to an offline stage, Fuzzy-CSar-AFP directly generates and evolves a population of association rules along with their respective quality measures. It performs online learning as data is received from the data stream.

To apply the algorithm on this dataset, we configure the algorithm so that the antecedent of the rules is generated from the variables referring to the part of the day and the variables corresponding to the terms of the stock market of interest, while the consequent is based on the sentiment of the message.

The algorithm is able to return the rules that form the population at any time. In this case, we decided to print the set of rules every 487 tweets (which corresponds to 10% of the data). As for other configuration parameters of the algorithm: $\theta_{mna} = 3$, the mutation parameters p_C and p_M have been defined as 0.4, while p_S turns out to be irrelevant in this case (no variable changes in the consequent are possible).

In Figures VI.17-VI.18, three Sankey diagrams are shown to summarize the linkage patterns in the association rules obtained by Fuzzy-CSar-AFP at different times in the data stream. These diagrams represent the relationships found between the terms of interest (antecedent) and the sentiment manifested in the tweets (consequent). The strength of these relationships is given by the frequency with which they occur and is represented in the diagram by the width of the link between the elements.

We can see how, the larger the amount of data examined, the more term-sentiment relationships appear. Figure VI.18 illustrates the relationships present in the population of rules corresponding to the final instant of the tweet stream.

It is clear from Figure VI.17 that at the beginning of the stream the rule set generated by Fuzzy-CSar-AFP is quite poor. The purely online approach of Fuzzy-CSar-

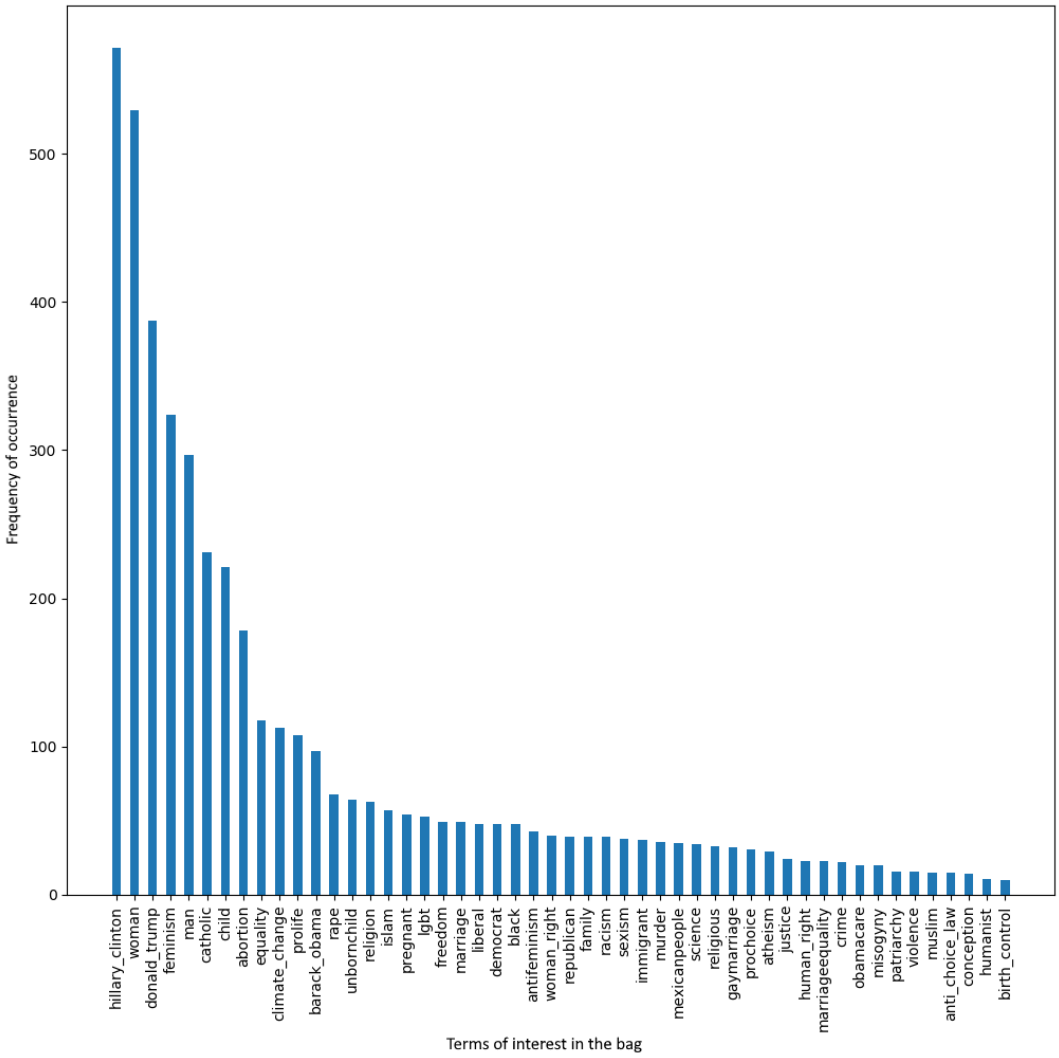
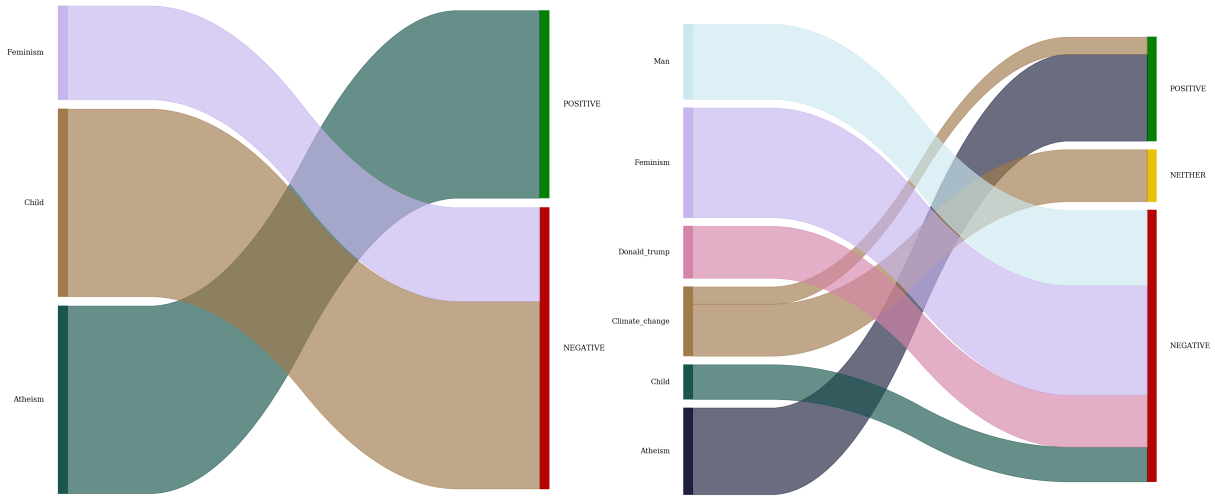


Figure VI.16: Histogram showing the frequency of occurrence of each of the words chosen for the interest bag in the US election problem.

AFP, in which the data stream is processed in an instance-based manner, has clear advantages but also brings certain limitations, such as the fact that the algorithm needs to receive a certain amount of data to start obtaining quality results. The amount of data is not a problem in real data stream environments, but this is a limited dataset and 487 tweets are insufficient for Fuzzy-CSar-AFP to obtain competitive results.

If we focus on Figure VI.18 we can see how terms such as `Feminism`, `Climate_change` and `Atheism` present a considerable weight. Among them, `Climate_change` stands out as it is the only term that appears associated with tweets that are neither linked to a positive nor a negative sentiment.



(a) (b)

Figure VI.17: *Sankey* diagram illustrating the set of term-sentiment links obtained based on the rules generated by *Fuzzy-CSarAFP* after analyzing (a) 487 and (b) 2,435 tweets.

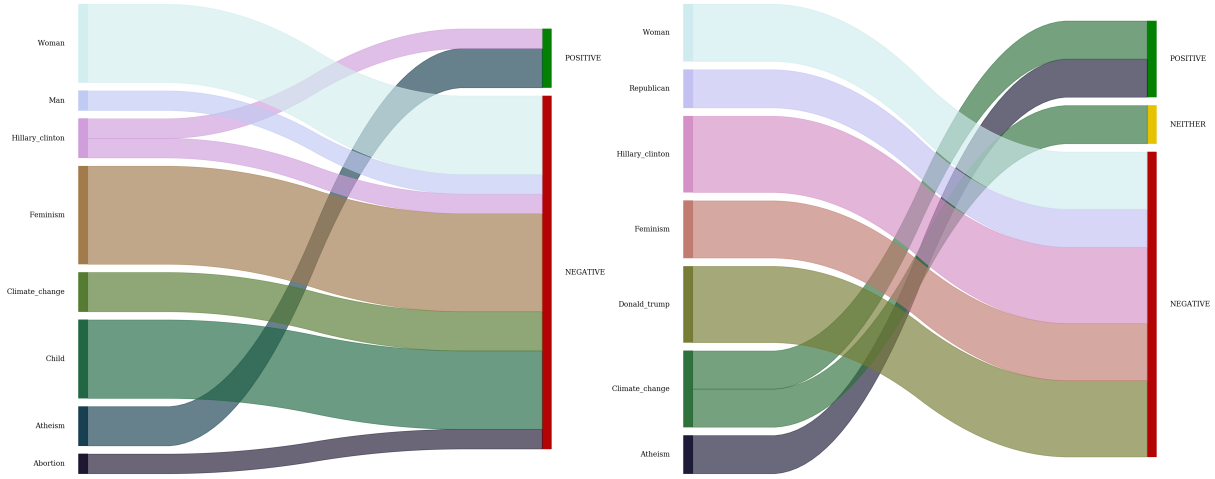
Dynamic extraction of association rules with IncMine

To carry out this analysis we use the IncMine extension for MOA (Bifet et al., 2010a). The use of IncMine requires the definition of some common parameters such as the minimum support (always necessary to obtain frequent itemsets) but also others specific to the algorithm, such as the relaxation rate. Thus, a threshold support of 0.05 and a relaxation rate of 0.3 have been set. In addition, the segment length is set to 487 so that it matches the frequency with which Fuzzy-CSar-AFP prints the rules. In the offline extraction of the association rules, all possible rules according to the FCIs of each data batch will be generated. By analyzing their lift value, the most interesting rules will be determined. On the resulting set, we will search for indications of relationships between terms of interest and sentiments.

In Figures VI.19-VI.20 Sankey diagrams are used to represent, based on the rules generated from IncMine, the relationships between different terms of interest and message sentiment. Each of the diagrams corresponds to a learning point, i.e., to the rules obtained offline after different batches of data. Figure VI.20 corresponds to the association rules obtained after analyzing the whole stream.

It can be seen that the links represented by the rules obtained are directed to a specific sentiment and almost never neutral. This fits with the context of Twitter, a social network employed by users to vindicate their opinions and where there is usually a tendency of dominance by those extreme users or attitudes. Polarization that becomes more intense when it comes to political issues.

Figure VI.18: *Sankey* diagram illustrating the set of term-sentiment links obtained based on the rules generated by *Fuzzy-CSarAFP* after analyzing all the 4870 tweets.



(a) (b)

Figure VI.19: *Sankey* diagram illustrating the set of term-sentiment links present on the rules obtained through the use of *IncMine* after the (a) first and (b) fifth batch of tweets have been processed.

At the end of the stream (Figure VI.20), there is a considerable variety of terms present in the relationships, and many of them, with similar weights. One of the most prominent is `Donald_trump`, which appears strongly linked to negative messages at this final stage. However, it also appears related to positive messages. Other examples of rules with a large number of appearances are those containing `Hillary_clinton` $\rightarrow$ Negative relations. Few terms appear related to both negative and positive sentiment. In addition to the case of `Donald_trump`, other terms that also relate to both sentiments in this final stage are `Woman` and `Feminism`.

If we compare the different images, we can clearly appreciate evolution both in the terms present and in the relationship of some of them with sentiments. This comparison allows us to observe, for example, the parallelism that exists between the evolution of the terms `Woman` and `Feminism`. Both terms present similar relevance at the three timestamps illustrated in the figures. Initially, they are related only to negative sentiments, but in Figure VI.20 they also appear linked to positive sentiment.

Another term that shows an interesting evolution is `Climate_change`, which begins with an important weight but with a negative connotation; it still maintains a certain weight in the conversation towards the middle of the stream but moving from that initial negative association to a link with neutral and positive messages; and, finally, it appears linked only to positive sentiment but having lost much weight in the conversation in the face of the new topics of discussion that have surfaced.

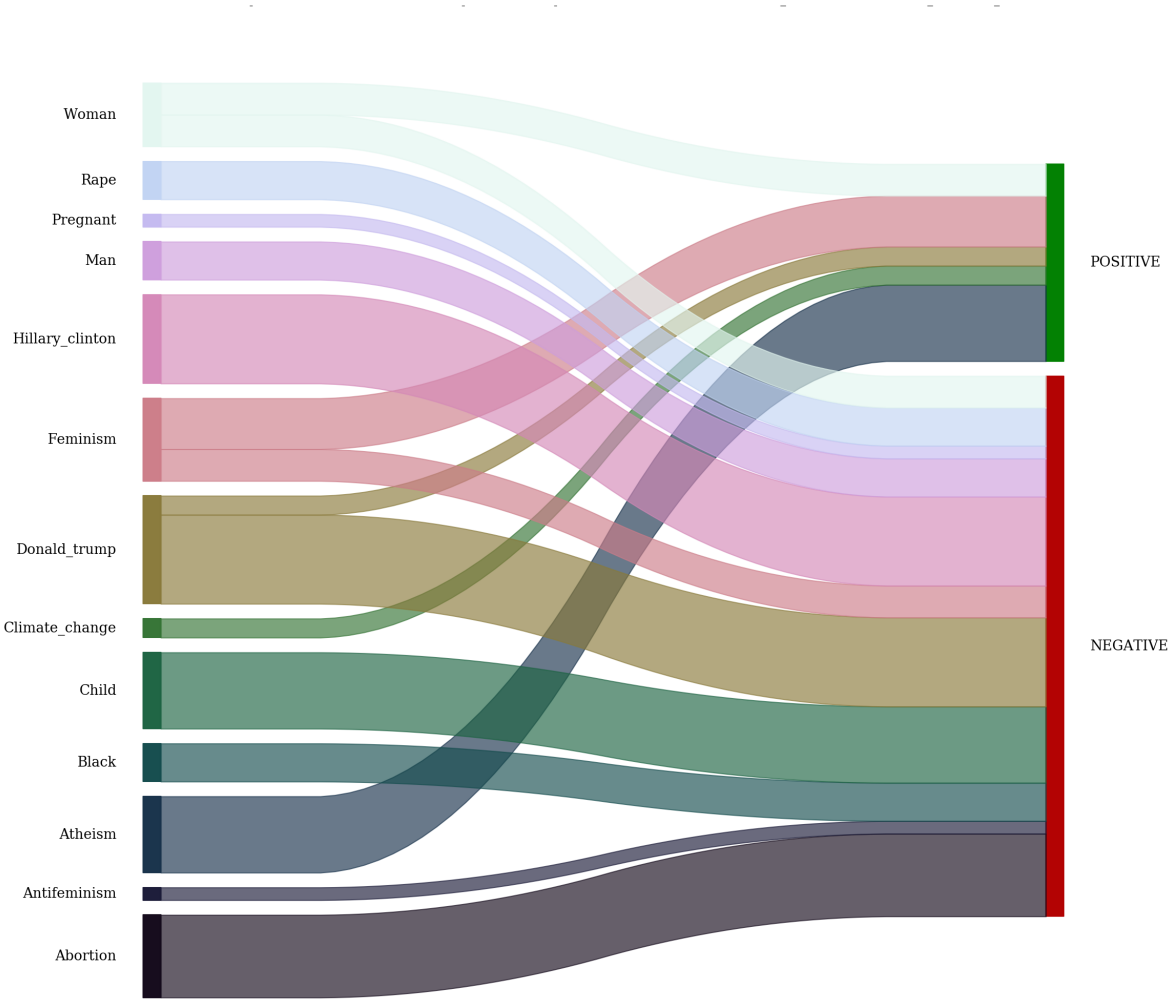


Figure VI.20: *Sankey* diagram illustrating the set of term-sentiment links present on the rules obtained through the use of *IncMine* after the last batch of tweets has been processed.

VI.2.5 Case 2: 2019 Spain investiture process

After conducting the initial study on the previous dataset, we decided to apply the same experimental setup on an original set of real data, extracted from Twitter through the use of its API. The tweets studied are also connected politics but in this case the messages are related to the 2019 Spain investiture process, which have been published in the period between July 15, 2019 and August 29, 2019. Moreover, in this case we will analyze tweets in Spanish, unlike the dataset used previously which contains only tweets in English.

In total, 261,080 tweets were extracted. For this purpose, a scraping task was performed using a series of hashtags as a reference for the searches. The hashtags used were: `#PSOE`, `#PP`, `#UnidasPodemos`, `#CiudadanosCs`, `#VOX`, `#InvestiduraCongreso19` and `#InvestiDudaARV`.

The tool provided by Twitter for extracting tweets offers numerous options. Taking advantage of this, several metadata are obtained to describe each message. Thus, the attributes associated with each of the tweets collected are: (1) *Created_at*, date of creation of the tweet; (2) *ID*, identification number of the tweet; (3) *Text*, message contained in the tweet; (4) *User_ID*, identification of the user posting the tweet; (5) *User_Name*, name of the user posting the tweet; (6) *Entities_Hashtags*, contains the various hashtags present in the tweet; (7) *RT*, indicates whether the posted message is a retweet; (8) *RT_Count*, count of the number of times the message has been retweeted; and (9) *Favorite_Count*, count of the number of times the message has been marked as a favorite (*favs*).

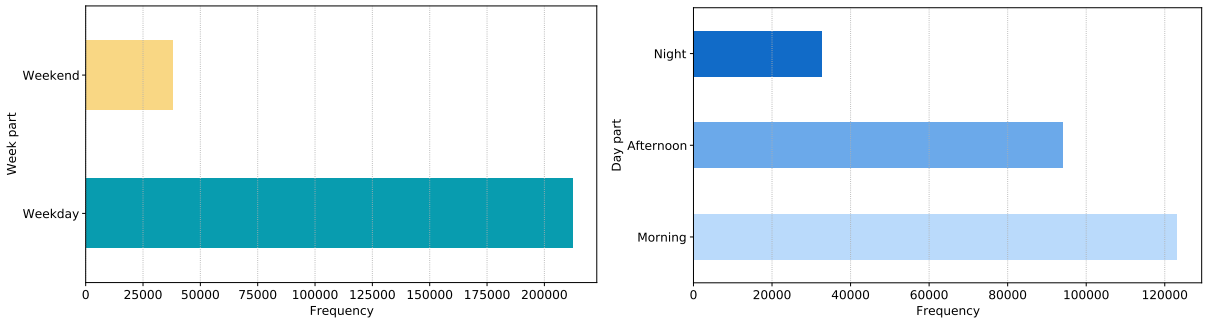
Furthermore, we generate two additional variables from the date of publication of the tweets: *Weekday*, which indicates whether the message was published during the week or at the weekend; and *DayPart*, which refers to the part of the day in which the message was published. The parts of the day are defined as:*morning* (from 7 AM to 2 PM), *afternoon-evening* (from 2 PM to 9 PM), *night* (from 9 PM to 7 AM).

Moreover, since we intend to extract rules with the sentiment of the message as a consequent, we need a variable that contains this sentiment. For this purpose, we have applied the *SentiStrength* classifier previously introduced, which will allow us to extract in a *Sentiment* attribute the sentiment present in the text. Likewise, it has been detected that some of the hashtags used gave rise to messages in other languages. In view of this, it has been decided to apply a language analyzer, so that if the message is considered to be in a language other than Spanish, it is classified as foreign. This is collected in the *Language* variable, which allows further filtering. The filtering derived from the application of this language analyzer reduces the total number of messages to 250,152, meaning that more than 10,000 tweets are discarded for being in a language other than Spanish.

VI.2.5.1 Exploratory data analysis

As we did in the previous case study, we analyzed the distribution of the values of some attributes of interest. Figures VI.21-VI.22 show the distributions of the variables *Weekday*,

DayPart and *Sentiment*. Figure VI.21 shows that the part of the day in which more tweets are published is the morning, this matches the part of the day in which both politicians and journalists tend to be more active. Along the same lines, in proportion to the number of days, there is greater activity on weekdays than on weekends. On the other hand, the variable whose distribution presents a priori greater interest is the sentiment expressed in the messages. In Figure VI.22, it can be seen that the *neutral* is the most common sentiment, followed by *positive*, and that the imbalance between *positive* and *negative* is limited.



(a) (b)

Figure VI.21: Distribution of values of the variables (a) *week part* and (b) *day part*.

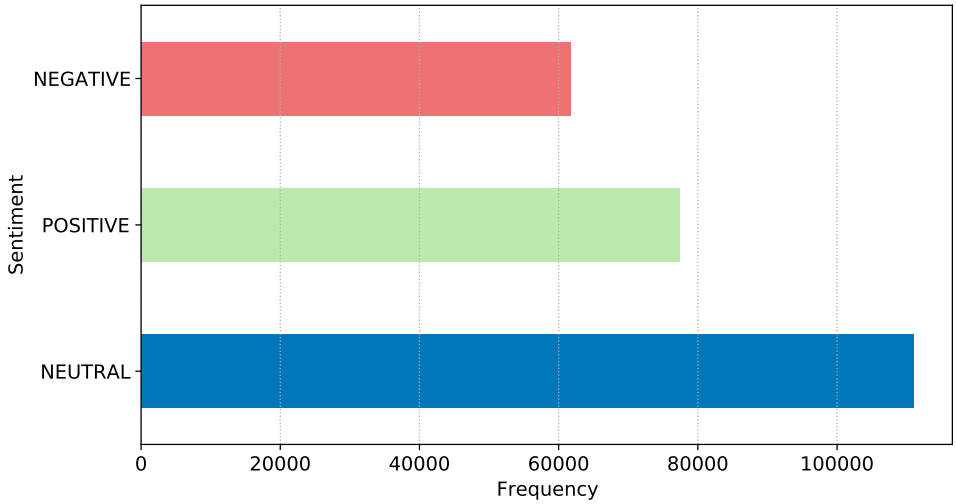


Figure VI.22: Distribution of the *Sentiment* variable in the set of tweets studied.

In addition, the influence of the political groups has been analyzed based on the mentions to the parties themselves and their members in the tweets published throughout the timeline. The individual names tracked have been chosen based on their relevance in the historical messages. Thus, while for the PSOE (Partido Socialista Obrero Español), names such as Pedro Sánchez (leader), Carmen Calvo and Adriana Lastra have been

considered, in the case of the PP (Partido Popular), relevant members such as Pablo Casado (leader) or Isabel Díaz Ayuso appear. For Ciudadanos, Inés Arrimadas and Albert Rivera (leader) have been highlighted, while for VOX, Santiago Abascal (leader) and Ortega Smith have been taken into account. Finally, in Unidas Podemos, Pablo Iglesias (leader), Irene Montero and Pablo Echenique were considered. The evolution of the amount of mentions of the different parties and their members over the analyzed time period is shown in Figure VI.23.

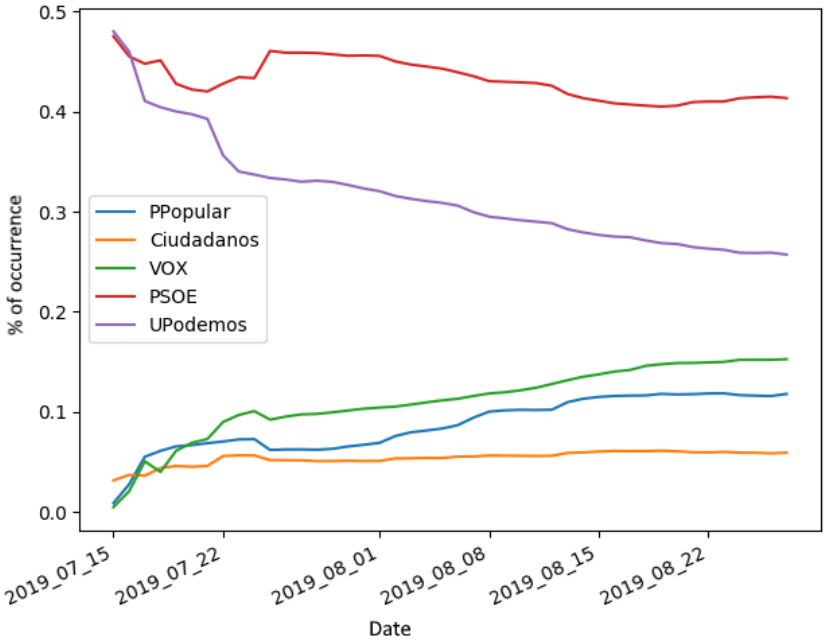


Figure VI.23: Evolution over time of the frequency of publication on the topics of interest in the tweets related to the 2019 Spanish investiture process.

We can observe how the two parties that were in the midst of negotiations to form a government, PSOE and Unidas Podemos, initially monopolize practically all the attention. The relevance of both parties remains fairly close until the days prior to the start of the investiture debate (July 22, 2019). From that moment on, the number of mentions linked to Unidas Podemos declines while those of the PSOE remain fairly stable. As for the three remaining parties, as the days go by, a rise in their number of mentions can be seen, thus balancing the situation of all parties, with the exception of the PSOE, which continues to stand out clearly. This prominent position of the PSOE is not surprising, since it had been the most voted party and its leader was the candidate to be sworn in as president. In general, the evolution in the mentions moves from a scenario of concentration of the discussion around only two parties (PSOE and Unidas Podemos) towards a more open scenario. This is consistent with the fact that at the beginning of the period analyzed there were intense negotiations to try to form a coalition government between PSOE and Unidas Podemos, while as the summer progresses the possibility of a repeat election is

gaining more and more strength.

VI.2.5.2 Natural language processing

A text analysis very similar to the one discussed for the US election dataset has been applied. Thus, links, numbers, punctuation (with the exception of # and @), emoticons and laughter have been removed from the messages, as well as possible repetitions of consecutive letters. Common text mining techniques have also been applied, starting with the removal of stopwords and short words, and applying grammatical tagging (again considering that hashtags and users have their own grammatical tags) and the lemmatization technique to standardize the text. Again, SpaCy has been used as a tool for text analysis because, although its speed of analysis is not the best, it presents a better library for the analysis of Spanish texts. In addition, a simplification of the text according to the grammatical category of the words has been conducted. A strategy similar to the one applied to the US election database has been followed, except that this time verbs have been considered.

After that, we analyzed the identifiable N-grams, looking for bigrams, trigrams and quatrigrams. After analyzing the obtained quatrigrams, the only term of interest is `partido_socialista_obrero_español`, which refers to the political party PSOE. On the other hand, in the case of trigrams, a total of 6 have been found that may be of interest. Five of them refer to political personalities (`isabel_diaz_ayuso`, `cayetana_alvarez_toledo`, `pedro_sanchez_castejon`, `javier_ortega_smith`, `miguel_angel_blanco`) and two (`ley_patrimonio_natural` and `ley_violencia_genero`) refer to laws. In particular, the first refers to a law that defends the conservation of the natural heritage and biodiversity of ecosystems (which is probably related to views on a serious forest fire in Gran Canaria) while the second refers to a 2004 law about violence against women (probably cited in the context of the news of the murders of women or the claims made by the VOX political party against feminism and against this law).

In the case of the bigrams, numerous terms of interest have been found, which are listed in Table VI.6. As can be seen, most of the bigrams refer to political personalities or Autonomous Communities. However, there are also groups or public institutions (such as `guardia civil`, `sanidad_publica`, `seguridad_social` or `audiencia_nacional`) and programs or measures (such as `politica_social`, `servicio_publico` or `ley electoral`), as well as interesting concepts related to political discussion (such as `preso_politico`, `violencia_genero`, `libertad_expresion` or `agresion_sexual`).

Again, it is necessary to carry out a unification of entities since there is a great variety of terms to refer to personalities, political groups or other entities of interest. Table VI.7 shows the entities considered together with the terms that come under each of them.

Once this is done, the texts are as standardized as possible, so we proceed to the creation of a bag of terms of interest from the database, which we deem relevant based on the concept to which they refer and their frequency of occurrence. On this occasion, this

(pedro, sanchez)	*(pablo, iglesias)*	*(unidas, podemos)*	*(carmen, calvo)*	*(violencia, genero)*
(mayoria, absoluto)	*(adriana, lastra)*	*(albert, rivera)*	*(partir, politico)*	*(susana, diaz)*
(pablo, casado)	*(mocion, censura)*	*(open, arms)*	*(santiago, abascal)*	*(guardia, civil)*
(comunidad, madrid)	*(reformar, laboral)*	*(irene, montero)*	*(repetir, eleccion)*	*(castilla, leon)*
(diaz, ayuso)	*(consejo, ministro)*	*(aitor, esteban)*	*(felipe, gonzalez)*	*(ivan, redondo)*
(señor, sanchez)	*(repeticion, electoral)*	*(campaña, electoral)*	*(politico, social)*	*(ley, mordaza)*
(gabriel, rufian)	*(congreso, diputados)*	*(preso, politico)*	*(partido, popular)*	*(grupo, parlamentario)*
(antonio, martinez)	*(rocio, monasterio)*	*(esperanza, aguirre)*	*(señor, iglesias)*	*(violencia, machista)*
(ana, oramas)	*(repeticion, eleccion)*	*(ortega, smith)*	*(alvarez, toledo)*	*(reforma, laboral)*
(seguridad, social)	*(pais, vasco)*	*(ortega, lara)*	*(ana, botella)*	*(guerra, civil)*
(isabel, #diazayuso)	*(señor, rivera)*	*(servicio, publico)*	*(señora, calvo)*	*(mariano, rajoy)*
(javier, maroto)	*(junta, andalucia)*	*(audiencia, nacional)*	*(alberto, garzon)*	*(lopez, miras)*
(raquel, romero)	*(sanchez, castejon)*	*(iñigo, errejon)*	*(region, murcia)*	*(libertad, expresion)*
(santi, abascal)	*(cristina, cifuentes)*	*(partido, socialista)*	*(ines, arrimadas)*	*(pablo, montesinos)*
(pablo, echenique)	*(julio, anguita)*	*(sra, calvo)*	*(maria, claver)*	*(agresion, sexual)*
(sanidad, publico)	*(laura, borras)*	*(ley, electoral)*	*(noelia, vera)*	*(ana, beltran)*
(señor, pedro)	*(coalicion, canaria)*	*(diaz, #ayuso)*	*(policia, nacional)*	*(yolanda, diaz)*

Table VI.6: Table with the different bigrams found in the set of tweets in Spanish.

bag contains a total of 40 terms of interest, which are shown in Table VI.8.

Figures VI.24 and VI.25, graphically represent the frequency of each of the terms included in the bag. These 40 terms are the items on which we rely to represent the tweets in transactional format.

VI.2.5.3 Real-time association rule analysis

Once the transaction database has been generated based on the terms of interest, the association rules between these terms and the links between terms and sentiments are obtained. As in the previous case, a study will be conducted following the two approaches considered: an incremental learning of data streams to extract the FCIs by using *IncMine* (Cheng et al., 2008), followed by *offline* rule learning; and a direct extraction of the association rules present by applying a fuzzy and *online* technique such as *Fuzzy-CSar-AFP* (Ruiz and Casillas, 2018).

After that, these rules are analyzed to look for the presence of relations between terms and sentiments in them. After obtaining all the rules, the corresponding analysis has to be carried out to look for relations of the *term of interest-sentiment* type. It is necessary to pay attention only to those that are sufficiently valid, for which the *lift* is again used as a quality measure: we look for rules with *lift* greater than 1.2.

Association stream mining with Fuzzy-CSar-AFP

At this point, we apply the *online* learning algorithm. *Fuzzy-CSarAFP* proceeding similarly to the previous case. Thus, it is established that the sentiment variable is the one that acts as the consequent, while the rest of the variables (part of the day, part of the week, retweet and the variables of the 40 terms of the bag of interest) will be part of the antecedents of the rules. Likewise, it has been defined that the writing of the rules

Entity	Associated terms
PSOE	*partido_socialista_obrero_español, psoe, #psoe, #partidosocialistaobreroespañol, @psoe, partido_socialista*
pedro_sanchez	*sanchez_castejon, @sanchezcastejon, #sanchez, #pedrosanchez, pedro_sanchez_castejon, #pedrosanchezcastejon, #sanchezcastejon, pedro_sanchez, señor_pedro, #pedrosanchezenlaser, #sanchezsi, #siapedro, #pedronoseatreve, #pedronoquiere, #sanchezdimision*
carmen_calvo	*carmen_calvo, señor_calvo, sra_calvo, #carmencalvo, @carmencalvo, @carmencalvo_*
VOX	*vox, @voxes,@vox, #vox, #sentidocomunvox, #voxutil, #voxverdaderaoposicion, #yovolvereavotarvox, #voxextremanecesidad*
adriana_lastra	*adriana_lastra, adrianalastra, #adrianalastra, @adrilastra, lastra*
santiago_abascal	*santiago_abascal, @santiabascal, @santi_abascal, santi_abascal, abascal, #abascal, señor_abascal, #santiabascal*
ortega_smith	*ortega_smith, javier_ortega_smith, @ortegasmith, @ortega_smith, #ortegasmith*
unidas_podemos	*unidas_podemos, podemos, #unidaspodemos, @unidas_podemos, @unidaspodemos,@ahorapodemos, #podemos, #ahorapodemos*
pablo_iglesias	*pablo_iglesias, señor_iglesias, @pabloiglesias, iglesias, @pablo_iglesias_, #pabloiglesias, #iglesias*
irene_montero	*irene_montero, @irenemontero, #irenemontero, @irene_montero_, #montero, irenemontero*
pablo_echenique	*pablo_echenique, echenique, @pnique, #pabloechenique, #echenique*
iñigo_errejon	*iñigo_errejon, errejon, #errejon, #iñigoerrejon, @ierrejon*
alberto_garzon	*alberto_garzon, #garzon, @garzon, #albertogarzon, @agarzon*
ciudadanos	*#ciudadanoscs, @ciudadanoscs, #cs, @ciudadanos, #ciudadanos*
ines_arrimadas	*ines_arrimadas, #inesarrimadas, #arrimadas, arrimadas, @inesarrimadas*
albert_rivera	*albert_rivera, señor_rivera, #albertrivera, albertrivera, #rivera, @albert_rivera, @albertrivera*
gabriel_rufian	*gabriel_rufian, señor_rufian, #rufian, rufian, @rufian, @gabrielrufian, #gabrielrufian*
partido_popular	*partido_popular, #partidopopular, partidopopular, @partidopopular, #pp, @ppopular, #ppopular, @populares*
pablo_casado	*pablo_casado, señor_casado, #casado, casado, @pablocasado, @pablocasado_, #pablocasado*
isabel_diaz_ayuso	*diaz_ayuso, @idiazayuso, #isabeldiazayuso, #diazayuso, #ayuso, isabel_diaz_ayuso, isabeldiazayuso, @isabeldiazayuso*
cayetana_alvarez_toledo	*cayetana_alvarez_toledo, #cayetanaalvarez, @cayetanaat, alvarez_toledo, #cayetanaalvareztoledo*
investidura	*#investidurapublico, #mvtinvestidura, #investiduraarv, #debatedeinvestidura, #mvtinvestidura, investidura, #investidurave, #investidura, #investiduracongreso #investidurafallida, #debateinvestiduraespv, #investiduravej, #sesiondeinvestidura*
elecciones	*elecciones, #elecciones, electoral, eleccion, #eleccionesespaña*
derecha	*derecha, #derecha, ultraderechista, ultraderecha, #ultraderecha, #trifachito, trifachito*
izquierda	*izquierda, izquierdo, progresista, #izquierda, #progresista, #socialista, #sociolisto, #sociolista*
independentismo	*independentismo, independentista, #independentista, #independencia, independencia*
abstencion	*abstencion, #abstencion, abstenerse, abstenido, abstener, #abstener*
repeticion_electoral	*repeticion_electoral, repeticion_eleccion, repeticion_elecciones, repetir_elecciones, repetir_eleccion, #repeticionelectoral, #repeticionelecciones, #repeticioneleccion*
negociacion	*dialogo, #dialogo, dialogar, negociar, negociacion, #negociacion*
pacto	*pacto, #pacto, pactar*
gobierno	*gobierno, #gobierno, gobernar, #construirgobierno*
feminismo	*feminismo, #feminismo, feminista, #feminista, feminazi*
open_arms	*open_arms, #openarms, @openarms*
democracia	*#democracia, democratico, democracia*
inmigracion	*open_arms, #openarms, @openarms, inmigracion, #inmigracion, inmigrante*
violencia_genero	*#violenciamachista, violencia_genero, agresion_sexual, #violenciadegenero, #misoginia*
corrupcion	*corrupto, #corrupcion, corrupcion*

Table VI.7: Table with the different entities considered and the terms included in them.

investidura	*PSOE*	*pedro_sanchez*	*pablo_iglesias*	*gobierno*
unidas_podemos	*VOX*	*partido_popular*	*izquierda*	*izquierda*
ciudadanos	*elecciones*	*coaliccion*	*negociacion*	*pacto*
albert_rivera	*carmen_calvo*	*santiago_abascal*	*isabel_diaz_ayuso*	*democracia*
pablo_casado	*gabriel_rufian*	*abstencion*	*adriana_lastra*	*irene_montero*
reforma_laboral	*feminismo*	*violencia_genero*	*iñigo_errejon*	*corrupcion*
pablo_echenique	*independentismo*	*ortega_smith*	*bipartidismo*	*alberto_garzon*
inmigracion	*mayoria_absoluta*	*repeticion_electoral*	*regeneracion*	*ines_arrimadas*

Table VI.8: Set of elements of interest considered for the 2019 Spanish investiture process case.

Figure VI.24: *Word cloud* with the words chosen for the interest bag of the Spanish investiture process problem.

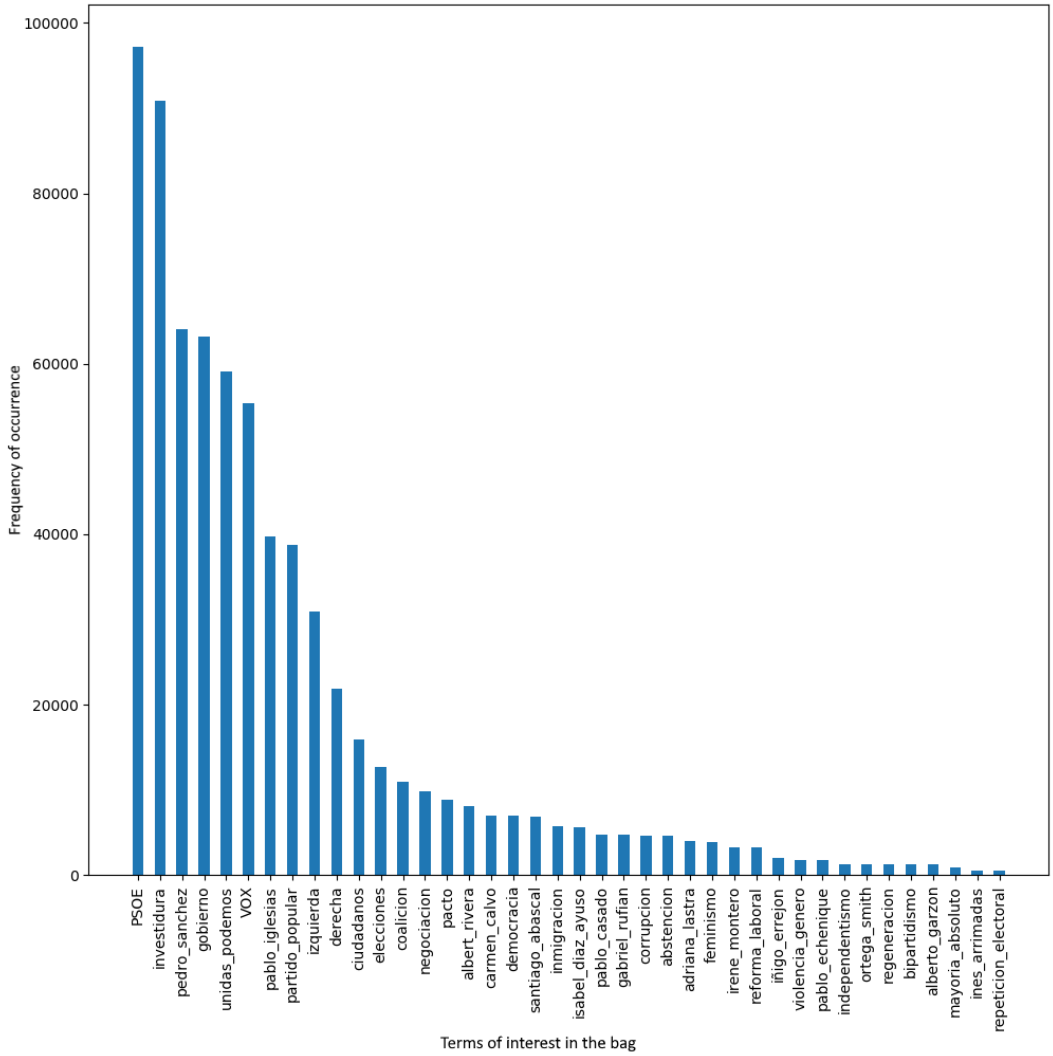
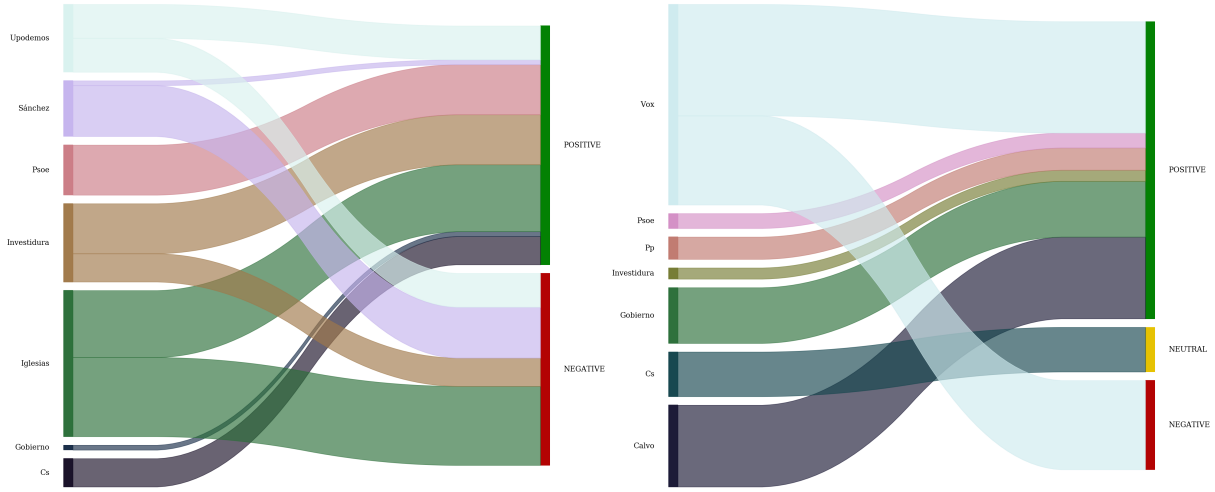


Figure VI.25: Histogram showing the frequency of occurrence of each of the words chosen for the interest bag in the Spanish investiture process problem.

obtained will be done every 5,600 elements, which corresponds to the average number of tweets per day. On the other hand, given the increase in the number of variables to consider, we set $\theta_{mna} = 6$. Meanwhile, the mutation parameters p_C and p_M have been set to 0.1, whereas, given its definition, p_S is irrelevant as there is only one possible variable for the consequent.

Once more, we analyze the resulting rules by focusing on the links between the defined terms of interest and the sentiments expressed in the tweets. Figures VI.26-VI.27 show three examples of the *Sankey* diagrams generated for such relations based on the rules obtained by *Fuzzy-CSar-AFP* at three different times in the stream. Of particular note is Figure VI.27, which presents the final rules, i.e., reflects the state of the *Fuzzy-CSar-AFP* rule population once the algorithm has processed all the tweets.



(a) (b)

Figure VI.26: *Sankey* diagram illustrating the set of term-sentiment links obtained based on the rules generated by *Fuzzy-CSarAFP* after analyzing (a) 5,600 and (b) 128,800 tweets.

These figures show variations in the number and content of the links found over time. There are, therefore, concept drifts, as would be expected in a real dataset such as the one studied. Over time, there are changes in the relations present in the data received, with terms or sentiments that are different from those of previous instants. These changes cause the population of rules maintained by Fuzzy-CSar-AFP to evolve and this is reflected in the *Sankey* diagrams shown.

Comparing the two diagrams included in Figure VI.26, the aforementioned evolution of the sets of relationships obtained (result of the concept changes discussed) can be appreciated. In spite of these variations, a stable link of the PSOE with positive messages can be appreciated, although the weight of this type of rules is reduced. However, certain changes can be seen in the case of other parties. Such is the case of Ciudadanos, which

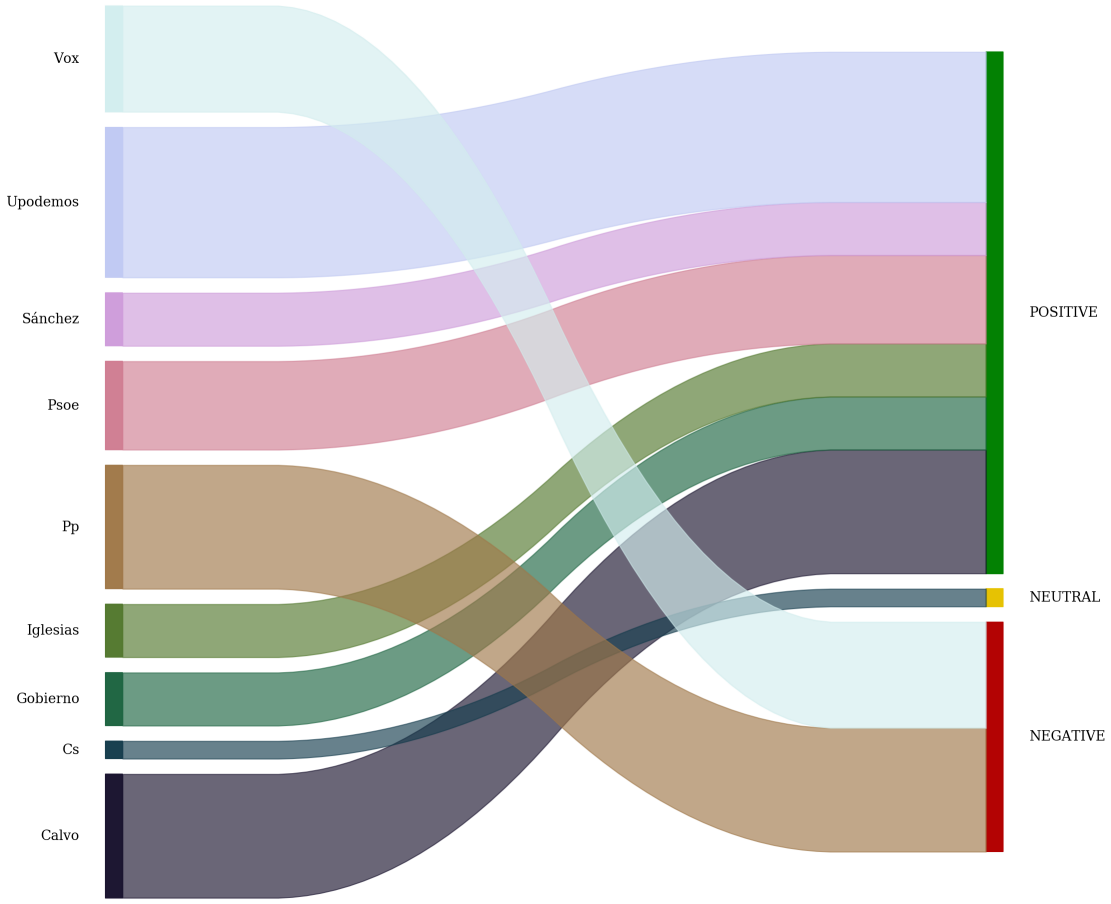


Figure VI.27: *Sankey* diagram illustrating the set of term-sentiment links obtained based on the rules generated by *Fuzzy-CSarAFP* after analyzing 250,147 tweets.

goes from having a link (albeit weak) with positive sentiment to a significant relationship with neutral sentiment. Likewise, in the case of the PP we see how it evolves from being linked to a positive sentiment to the disappearance of this link and to an important weight of rules that associate it with a negative sentiment.

Another interesting case is that of VOX, which initially does not appear in the rules but gradually gains weight in the messages and begins to be included in the rules generated. It is noteworthy that at the midpoint of the stream it appears linked to the positive and negative sentiments with almost equal strength, and presenting a great weight with respect to the rest of the terms, since it is present in the antecedents of around half of the rules. This can be clearly seen in Figure VI.26.

The analysis of Figure VI.27 presents a very interesting scenario, in which all political parties appear associated with a sentiment, giving us a summary of the sentiment generated by the different political groups once all the tweets have been visited. Thus, while PSOE and Podemos present a high number of rules with positive sentiment; Ciudadanos appears related to neutral sentiment, and PP and VOX get linked to negative sentiment.

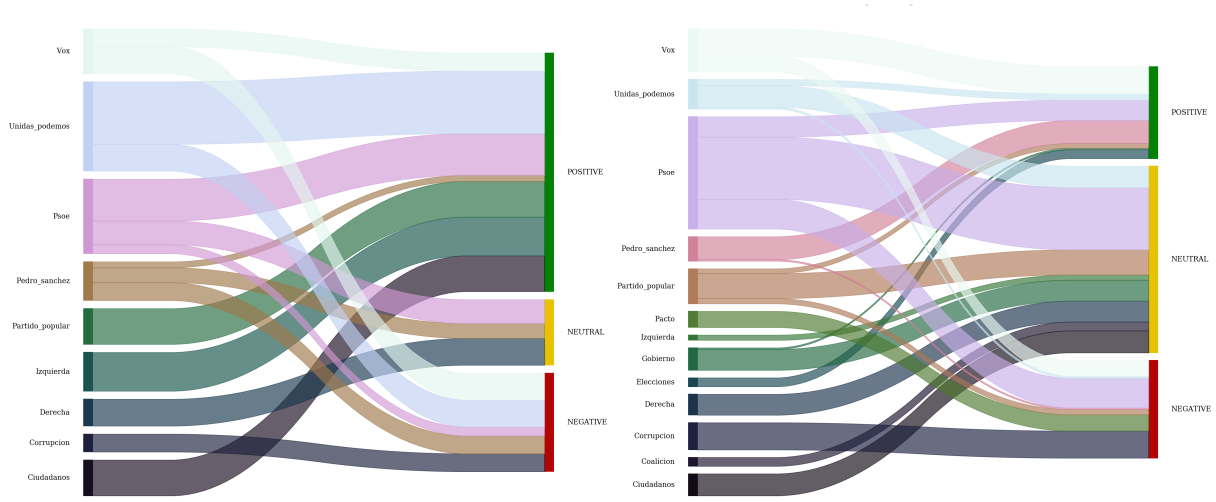
Dynamic extraction of association rules with IncMine

As in the previous case, we applied IncMine available in MOA (Bifet et al., 2010a) to extract FCIs from the data stream and then, at the end of each data batch, generate the association rules offline. For the IncMine configuration, we keep the threshold support of 0.05, but set a relaxation ratio of 0.3. Again, the segment length matches the rule writing frequency of Fuzzy-CSar-AFP. After obtaining the FCIs and the subsequent offline extraction of association rules from them, the presence of relationships between terms of interest and sentiments is analyzed based on the existing relationships in the set of association rules filtered according to the *lift* threshold.

Figures VI.28-VI.29 show the link patterns present in the association rules obtained using the *IncMine* algorithm for this second case study based on Spanish tweets. Each of the plots corresponds to a different learning point, i.e., to the rules generated after different *batches* of data have been processed. The association rules obtained after IncMine processes the last batch of data yield Figure VI.20.

We can appreciate how the number of rules obtained at different points in the stream is uneven, so that the number and type of relationships between terms vary over time. The analysis of Figure VI.29 shows us a very similar situation to that obtained for *Fuzzy-CSarAFP*, where the five political parties are associated to sentiments. However, the links created are different. This can be explained by the differences between the two algorithms when it comes to processing the data and evolving the models.

The analysis of the different stages and their comparison makes it possible to identify many relationships that remain constant over time, such as the CORRUPCION $\rightarrow$ NEGATIVE link. However, there are also other rules that present clear variations, so that



(a) (b)

Figure VI.28: *Sankey* diagram illustrating the set of term-sentiment links obtained based on the rules obtained through the use of *IncMine* after analyzing (a) 5,600 and (b) 128,800 tweets.

the phenomenon of *concept drift* is made evident again. An example is the case of *PP*, which initially appears linked to a positive sentiment but soon evolves to a relationship of the type PP $\rightarrow$ NEUTRAL, although maintaining links of lesser weight with the other two sentiments. Thus, along the timeline it relates with sentiments of all kinds, whereby messages received in the data stream show changes in the opinions of users, which give rise to rules implying different consequents. Towards the end of the stream, the sentiment with which it appears most strongly linked is the negative one.

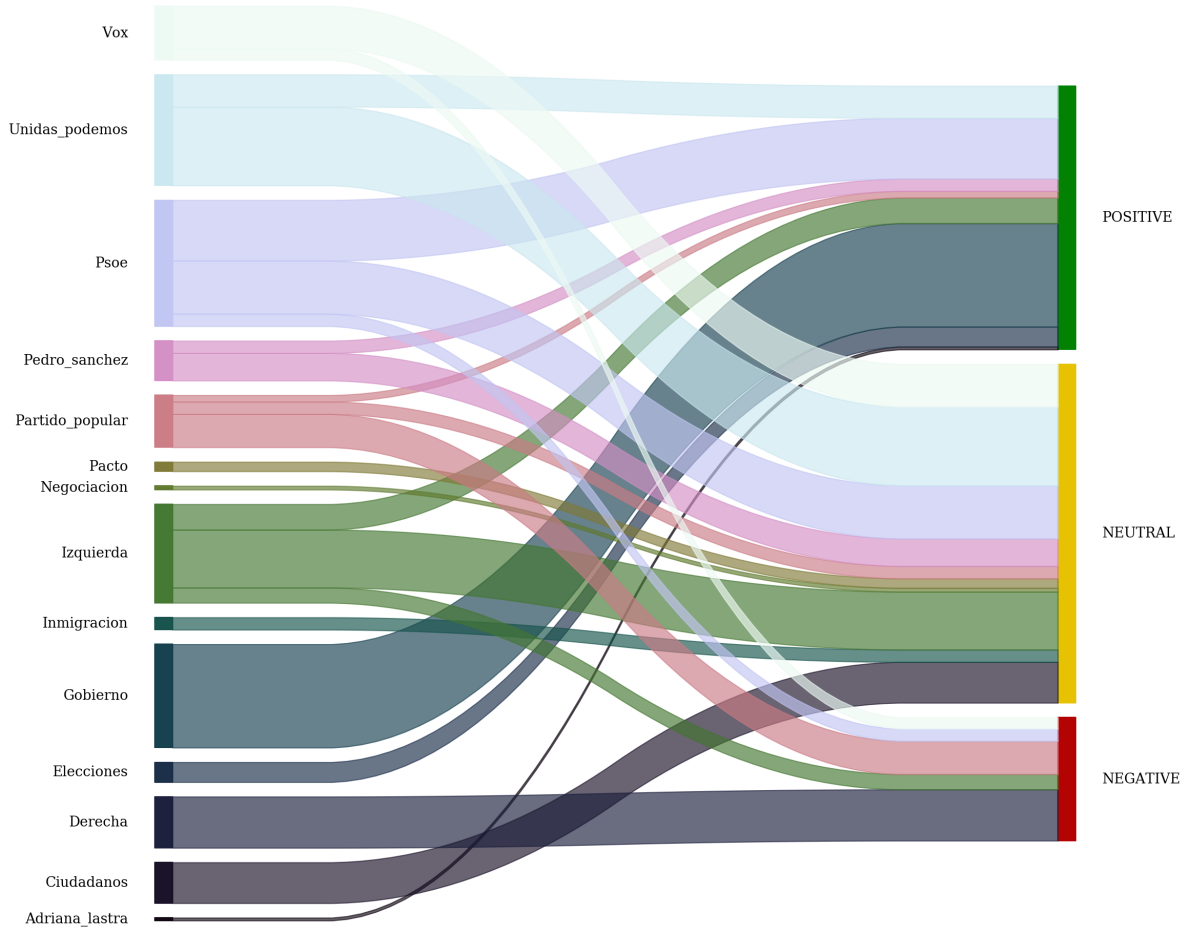


Figure VI.29: *Sankey* diagram illustrating the set of term-sentiment links obtained based on the rules obtained through the use of *IncMine* after analyzing 250,147 tweets.

Chapter VII

Conclusions and future work

VII.1 Concluding remarks

The main objective of this thesis is to contribute to the existence of online learning proposals for data stream mining that are useful and applicable in real problems. In this sense, we advocate the development of descriptive models, based on interpretable knowledge representations, that allow to understand what is happening in the system. We study the use of rule learning in classification problems, but we focus on other learning paradigms that do not require large amounts of labeled data and that we believe may be more directly applicable to many real data stream contexts.

Three algorithmic proposals are presented throughout this thesis. Each of them is based on a different learning paradigm. First, a supervised approach for classification in data streams, CLAST, is presented. CLAST learns rules that allow it to approach the classification task with competitive results, while generating descriptive models that can be analyzed and interpreted to understand the basis for classification decisions. The method present a purely online design approach, where each example is visited just once upon its arrival. The population of rules evolves dynamically and a steady-state genetic algorithm is used for the discovery of new rules. A Hoeffding bound is incorporated for some of the decisions made in the evolution of the population, which allows, among other things, to reduce the number of parameters required. In addition, the system maintains a set of histograms that are also updated online and provide additional information about the underlying class distribution. At any time, the rules in the population can work together to predict the class label of a received example. Compared to other data stream classifiers, CLAST obtains the best average results in the extensive benchmark experiments conducted and present very competitive performance in various real-world data stream scenarios.

Given the high cost of maintaining a high level of labeling in data streams, we consider that other learning paradigms may be more applicable than the supervised paradigm. In this line, an unsupervised learning algorithm for association stream mining is proposed. Unlike most methods in the literature, our proposal, Fuzzy-CSar-AFP, is

not limited to the identification of frequent itemsets nor does it relegate the generation of association rules to an offline phase. The use of an evolutionary algorithm allows Fuzzy-CSar-AFP to evolve the association rules directly from the data without the need for the classical two-step process of rule mining algorithms. Thus, the algorithm dynamically maintains a set of fuzzy association rules that explain what is happening in the system at any given time. Fuzzy-CSar-AFP is an advanced version of an earlier proposal that incorporates new mechanisms which adapt membership functions and fuzzy partitions so that the algorithm might be endowed with more flexibility to fit the features of each variable. The better behavior of Fuzzy-CSar-AFP as compared with Fuzzy-CSar and Fuzzy-Apriori is evidenced through the conducted experiments. These experiments are conducted on complex original real-world data streams related with a Psychophysiology problem. Moreover, we tackle the difficulties of fairly evaluating association stream mining by incorporating a new methodology to assess and compare results from different algorithms. This new methodology comprises both new quantitative metrics and visual representations. Thus, we are able to show how the quality rules generated by Fuzzy-CSar-AFP are distributed in a more spacious way. This more spaced distribution means that the rules are more relevant as they represent different knowledge.

The last one of the algorithmic proposals brings together the supervised and unsupervised worlds. PAST, is a semi-supervised approach to deal with label scarcity in data stream classification. It takes advantage of unlabeled examples (much more frequent and easily available in data streams) to alleviate the shortcomings resulting from label scarcity. Given the good performance shown by CLAST and the difficulty to obtain labeled data in many data stream environments, we present this adaptation of CLAST that operates under the semi-supervised learning paradigm. The algorithm continues to be fully online and maintains the ability to predict the class label of an example at any time, but, in this case, both labeled and unlabeled examples are used to update the population of rules and the set of histograms, i.e., to evolve the knowledge of the system. We compare the performance of PAST with that of CLAST and other classifiers both online and offline. The experiments are conducted on different datasets among which real data stream problems are included. For each of the problems, the behavior of the algorithms is studied under different labeling conditions, ranging from only 5% to 100% labeled data. PAST obtains the best average results in the collection of benchmark datasets. It significantly improves the performance of the rest of data stream classifiers tested. It also obtains highly competitive results in the real data stream problems included in the experimentation, being the best performing algorithm for low labeling percentages in three of the four problems addressed. PAST improves the results obtained by CLAST in all cases where there is unlabeled data, although this improvement is more noticeable for low labeling percentages (5-10%).

In addition, two real use cases are presented in which the value of the knowledge extracted through association stream mining is demonstrated. The first of these applications uses smartphone usage data collected over several months. The association stream mining algorithm helps to explain what is happening at any time. The results obtained show the

evolution experienced by the population of rules as the data stream is received. These results exemplify how Fuzzy-CSar-AFP and the association rules discovered by it make possible to find data properties out, properties that otherwise would not have been detected. Since the algorithm does not assume any a priori structure for the problem, it is able to adapt itself to the specific characteristics of each subject's data. Moreover, this is achieved in a very efficient way, it takes only 1.97ms to Fuzzy-CSar-AFP to process each datum, i.e., with this specific data structure an input rate of about 500Hz could be handled.

The analysis of the rules discovered by the algorithm is complemented with periodical subjective information about emotional state. Thus, the evolution over time of the levels of happiness, stress, productivity and health expressed by the subjects (but not shown to the algorithm) is compared with the evolution of the different rules generated automatically in real time. Interestingly, an important correlation is detected between the user's emotional state and certain patterns in the use of the smartphone.

The second application focuses on the use of association rules to analyze streams of tweets. The obtained tweets are related to political topics. Specifically, two cases are analyzed: a first smaller collection of tweets linked to the 2016 US presidential election, and a second original real collection with more than 250,000 tweets related to the investiture process in Spain during the summer of 2019. In both cases, after extracting the tweets, natural language processing techniques are applied to identify a collection of terms of interest (items) based on which the tweets are represented as transactions. In addition, in the second case, sentiment analysis techniques are applied to associate each tweet with a sentiment label. On these data, we apply the IncMine frequent itemset mining algorithm and the Fuzzy-CSar-AFP association stream mining algorithm. The analysis of the obtained association rules focuses on the term-sentiment links present in the rules. Through this analysis, it is possible to observe how certain political parties, personalities or themes tend to maintain stable associations with a given sentiment, while for others it is observed how the type of associated sentiment changes over time. In some cases, parallelism between the evolutions of the sentiments associated with two terms is observed.

In summary, we have managed to design solutions based on evolutionary learning of rules from data streams, which successfully address different types of data mining problems. The good performance and usefulness of the proposals has been evidenced in real-world applications.

VII.2 Future work

The work developed in this thesis and its outcomes enable new open challenges, improvements and research works. Some future research lines related to the previously drawn conclusions are detailed below:

- Concept-drift and Non-stationary environments. We plan to conduct a more extensive study of the performance of the proposals, especially CLAST and PAST, under

concept-drift conditions. From the results of this study, we could analyze the possibility of incorporating techniques based on decay factor to favor a greater influence of the current data allowing a faster adaptation to changes that may occur in the data. Another possibility to be studied would be the adaptation of techniques on concept change detection in supervised or semi-supervised learning (Žliobaitė et al., 2013; Casillas et al., 2018). Indeed, the proposal presented in Casillas et al. (2018) is based on structures that are already incorporated in the learning iteration of CLAST and PAST.

- Improvement of our unsupervised learning proposal. Along the same lines that have driven the novelties already introduced by Fuzzy-CSar-AFP (i.e., favoring rule diversity and better adaptation of the algorithm to the conditions of real problems), the use of diversity mechanisms to guarantee the generation of heterogeneous rules (Rodríguez et al., 2013), as well as, the use of online preprocessing in problems with a high number of input variables (Žliobaitė and Gabrys, 2012) could be explored.

- Exploring new real-world applications. Apply these techniques to other real problems where they may be useful, working in an interdisciplinary way with experts from other areas that can benefit from the knowledge extracted by the algorithms. In this PhD dissertation, we have shown examples of real applications of the proposals. In this sense, researchers in psychophysiology at the University of Granada have already shown their interest in applying this type of techniques to the study of the dynamic causal model of EEG in different scenarios. In any case, as has been demonstrated with the applications presented, there is a wide range of potential fields of application.

- Dissemination. It is our intention to share the developed algorithms with the community. Different options are being considered, from publishing the developed algorithms in general open source repositories (e.g. GitHub) to incorporating them in specific software for data mining or data stream mining (such as MOA (Bifet et al., 2010a) or Spark Streaming (Spark, 2021)). Another of the possibilities contemplated is to enable a web tool through which users can enter their data and make use of some of the techniques. Some steps have been taken in this direction, but for the moment they are still premature and limited to rule visualization techniques.

- Further exploiting the descriptive power of rules. As the number of rules grows, interpreting the results becomes more challenging. However, since the rules represent relationships between variables, they offer the ability to reach a level of interpretability detail that cannot be approached with other techniques. In this sense, we consider interesting to investigate methods that allow us to deepen the interpretation of the rules. One of the possible paths to explore is the development of specialized visualizations. Nevertheless, other types of approaches aimed at synthesizing the knowledge present in the rules are also worth studying.

- Proposal of a greedy algorithm for association stream mining. A greedy proposal, which is not based on optimization paradigms such as evolutionary computation and which does not use fuzzy representation of variables, present two main potential advantages: its simplicity and the possibility of increasing efficiency. These two potential advantages could also facilitate its incorporation into tools or repositories of data mining methods, increasing the usefulness and impact of the proposal. Part of the objective would also be to minimize the number of parameters to try to ensure the greatest possible ease of use.

Bibliography

Agarwal, R. C., Aggarwal, C. C., and Prasad, V. (2000). Depth first generation of long patterns. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 108–118.

Aggarwal, C. C. (2003). A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 575–586.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434. Springer.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago de Chile, Chile*, pages 487–499.

Aharony, N., Pan, W., Ip, C., Khayal, I., and Pentland, A. (2011a). Friends and family dataset. Reality Commons. `http://realitycommons.media.mit.edu/friendsdataset.html`. Accessed February 21, 2021.

Aharony, N., Pan, W., Ip, C., Khayal, I., and Pentland, A. (2011b). Social fMRI: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659.

Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147.

Ahmadi, Z. and Beigy, H. (2012). Semi-supervised ensemble learning of data streams in the presence of concept drift. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 526–537. Springer.

Akbarinia, R. and Masseglia, F. (2013). Fast and exact mining of probabilistic data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 493–508. Springer.

Alcalá, R., Casillas, J., Cordón, O., and Herrera, F. (2001). Building fuzzy graphs: features and taxonomy of learning for non-grid-oriented fuzzy rule-based systems. *Journal of Intelligent & Fuzzy Systems*, 11(3, 4):99–119.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17.

Altshuler, Y., Aharony, N., Fire, M., Elovici, Y., and Pentland, A. (2012). Incremental learning with accuracy prediction of social and individual properties from mobile-phone data. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 969–974. IEEE.

Altshuler, Y., Fire, M., Aharony, N., Volkovich, Z., Elovici, Y., and Pentland, A. S. (2013). Trade-offs in social and behavioral modeling in mobile networks. In *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*, pages 412–423. Springer.

Anderson, I., Maitland, J., Sherwood, S., Barkhuus, L., Chalmers, M., Hall, M., Brown, B., and Muller, H. (2007). Shakra: tracking and sharing daily activity levels with unaugmented mobile phones. *Mobile Networks and Applications*, 12(2-3):185–199.

Bansal, A. (2020). London bike sharing dataset. `https://www.kaggle.com/architt9406/bike-sharing`. Accessed February 21, 2021.

Bardram, J. E., Frost, M., Szántó, K., Faurholt-Jepsen, M., Vinberg, M., and Kessing, L. V. (2013). Designing mobile health technology for bipolar disorder: a field trial of the MONARCA system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2627–2636. ACM.

Bartsch, R. P., Liu, K. K., Bashan, A., and Ivanov, P. C. (2015). Network physiology: how organ systems dynamically interact. *PLOS ONE*, 10(11):e0142143.

Basat, R. B., Einziger, G., Friedman, R., Luizelli, M. C., and Waisbard, E. (2017). Constant time updates in hierarchical heavy hitters. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 127–140.

Basat, R. B., Einziger, G., Keslassy, I., Orda, A., Vargaftik, S., and Waisbard, E. (2018). Memento: Making sliding windows efficient for heavy hitters. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, pages 254–266.

Beringer, J. and Hüllermeier, E. (2007). Efficient instance-based learning on data streams. *Intelligent Data Analysis*, 11(6):627–650.

Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., and Zuefle, A. (2009). Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128.

Bertini, J. R., de Andrade Lopes, A., and Zhao, L. (2012). Partially labeled data stream classification with the semi-supervised k-associated graph. *Journal of the Brazilian Computer Society*, 18(4):299–310.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *International Conference on Database Theory*, pages 217–235. Springer.

Bifet, A. (2010). *Adaptive stream mining: Pattern learning and mining from evolving data streams*, volume 207. IOS Press.

Bifet, A. and Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference On Data Mining*, pages 443–448.

Bifet, A. and Gavaldà, R. (2009a). Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer.

Bifet, A. and Gavaldà, R. (2009b). Adaptive xml tree classification on evolving data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 147–162. Springer.

Bifet, A., Gavaldà, R., Holmes, G., and Pfahringer, B. (2018). *Machine learning for data streams: with practical examples in MOA*. MIT press.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010a). MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11:1601–1604.

Bifet, A., Holmes, G., and Pfahringer, B. (2010b). Leveraging bagging for evolving data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–150. Springer.

Bifet, A., Holmes, G., Pfahringer, B., and Gavaldà, R. (2009a). Improving adaptive bagging methods for evolving data streams. In *Asian Conference on Machine Learning*, pages 23–37. Springer.

Bifet, A., Holmes, G., Pfahringer, B., and Gavaldà, R. (2011). Mining frequent closed graphs on evolving data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 591–599.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009b). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148.

Bloehdorn, S., Blohm, S., Cimiano, P., Giesbrecht, E., Hotho, A., Lösch, U., Mädche, A., Mönch, E., Sorg, P., Staab, S., et al. (2011). Combining data-driven and semantic approaches for text mining. In *Foundations for the Web of Information and Services*, pages 115–142. Springer.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100.

Bogomolov, A., Lepri, B., Ferron, M., Pianesi, F., and Pentland, A. S. (2014). Pervasive stress recognition for sustainable living. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops*, pages 345–350. IEEE.

Bogomolov, A., Lepri, B., and Pianesi, F. (2013). Happiness recognition from mobile phone data. In *2013 International Conference on Social Computing*, pages 790–795. IEEE.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(Sep):1579–1619.

Bouten, C. V., Koekkoek, K. T., Verduin, M., Kodde, R., and Janssen, J. D. (1997). A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity. *IEEE Transactions on Biomedical Engineering*, 44(3):136–147.

Boyer, R. S. and Moore, J. S. (1991). MJRTY—a fast majority vote algorithm. In *Automated Reasoning*, pages 105–117. Springer.

Braverman, V., Chestnut, S. R., Ivkin, N., and Woodruff, D. P. (2016). Beating countsketch for heavy hitters in insertion streams. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, pages 740–753.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press.

Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 255–264.

Brooke, J., Tofiloski, M., and Taboada, M. (2009). Cross-linguistic sentiment analysis: From english to spanish. In *Proceedings of the International Conference RANLP-2009*, pages 50–54.

Brzeziński, D. and Stefanowski, J. (2011). Accuracy updated ensemble for data streams with concept drift. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 155–163. Springer.

Brzezinski, D. and Stefanowski, J. (2013). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94.

Brzezinski, D. and Stefanowski, J. (2014). Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265:50–67.

Bustio-Martínez, L., Letras-Luna, M., Cumplido, R., Hernández-León, R., Feregrino-Uribe, C., and Bande-Serrano, J. M. (2019). Using hashing and lexicographic order for frequent itemsets mining on data streams. *Journal of Parallel and Distributed Computing*, 125:58–71.

Cafaro, M., Epicoco, I., and Pulimeno, M. (2019). Cmss: Sketching based reliable tracking of large network flows. *Future Generation Computer Systems*, 101:770–784.

Caldarelli, G., Chessa, A., Pammolli, F., Pompa, G., Puliga, M., Riccaboni, M., and Riotta, G. (2014). A multi-level geographical study of italian political elections from twitter data. *PLOS ONE*, 9(5):e95809.

Casilari-Pérez, E. and García-Lagos, F. (2019). A comprehensive study on the use of artificial neural networks in wearable fall detection systems. *Expert Systems with Applications*, 138:112811.

Casillas, J. and Martínez-López, F. J. (2009). Mining uncertain data with multiobjective genetic fuzzy systems to be applied in consumer behaviour modelling. *Expert Systems with Applications*, 36(2):1645–1659.

Casillas, J., Wang, S., and Yao, X. (2018). Concept drift detection in histogram-based straightforward data stream prediction. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW 2018)*, pages 878–885. IEEE.

Castellano, G. and Fanelli, A. M. (2016). Classification of data streams by incremental semi-supervised fuzzy clustering. In *International Workshop on Fuzzy Logic and Applications*, pages 185–194. Springer.

Ceron, A., Curini, L., and Iacus, S. (2014). Using social media to forecast electoral results. a meta-analysis. *UNIMI-Research Papers in Economics, Business, and Statistics*, page 62.

Chamlertwat, W., Bhattarakosol, P., Rungkasiri, T., and Haruechaiyasak, C. (2012). Discovering consumer insight from twitter via sentiment analysis. *Journal of Universal Computer Science*, 18(8):973–992.

Chan, T. F., Golub, G. H., and LeVeque, R. J. (1983). Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27.

Chang, J. H. and Lee, W. S. (2003). Finding recent frequent itemsets adaptively over online data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 487–492.

Chang, J. H. and Lee, W. S. (2005). Effect of count estimation in finding frequent itemsets over online transactional data streams. *Journal of Computer Science and Technology*, 20(1):63–69.

Charikar, M., Chen, K., and Farach-Colton, M. (2004). Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15.

Chen, P., Su, H., Guo, L., and Qu, Y. (2010). Mining fuzzy association rules in data streams. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 4, pages V4–153. IEEE.

Cheng, J., Ke, Y., and Ng, W. (2008). Maintaining frequent closed itemsets over a sliding window. *Journal of Intelligent Information Systems*, 31(3):191–215.

Chi, Y., Wang, H., Philip, S. Y., and Muntz, R. R. (2006). Catch the moment: maintaining closed frequent itemsets over a data stream sliding window. *Knowledge and Information Systems*, 10(3):265–294.

Choi, H.-J. and Park, C. H. (2019). Emerging topic detection in twitter stream based on high utility pattern mining. *Expert Systems with Applications*, 115:27–36.

Chui, C.-K., Kao, B., and Hung, E. (2007). Mining frequent itemsets from uncertain data. In *Pacific-Asia Conference on Knowledge Discovery and Data mining*, pages 47–58. Springer.

Cordón, O. et al. (2001). *Genetic Fuzzy Systems: evolutionary tuning and learning of fuzzy knowledge bases*, volume 19. World Scientific.

Cormode, G. and Hadjieleftheriou, M. (2009). Finding the frequent items in streams of data. *Communications of the ACM*, 52(10):97–105.

Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. (2004). Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 155–166.

Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. (2008). Finding hierarchical heavy hitters in streaming data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(4):1–48.

Cormode, G. and Muthukrishnan, M. (2011). Approximating data with the count-min sketch. *IEEE Software*, 29(1):64–69.

Corral, G., Garcia-Piquer, A., Orriols-Puig, A., Fornells, A., and Golobardes, E. (2011). Analysis of vulnerability assessment results based on caos. *Applied Soft Computing*, 11(7):4321–4331.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Culpeper, J., Findlay, A., Cortese, B., and Thelwall, M. (2018). Measuring emotional temperatures in shakespeare's drama. *English Text Construction*, 11(1):10–37.

Dal, A. P., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3784–3797.

Dang, X. H., Lee, V. C., Ng, W. K., and Ong, K. L. (2009). Incremental and adaptive clustering stream data over sliding window. In *International Conference on Database and Expert Systems Applications*, pages 660–674. Springer.

Datar, M., Gionis, A., Indyk, P., and Motwani, R. (2002). Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813.

Dawar, S., Sharma, V., and Goyal, V. (2017). Mining top-k high-utility itemsets from a data stream under sliding window model. *Applied Intelligence*, 47(4):1240–1255.

Deckert, M. (2011). Batch weighted ensemble for mining data streams with concept drift. In *International Symposium on Methodologies for Intelligent Systems*, pages 290–299. Springer.

Delalleau, O., Bengio, Y., and Roux, N. L. (2005). Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Barbados*, pages 96–103. Society for Artificial Intelligence and Statistics.

Demaine, E. D., López-Ortiz, A., and Munro, J. I. (2002). Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*, pages 348–360. Springer.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Denil, M., Matheson, D., and Freitas, N. (2013). Consistency of online random forests. In *International Conference on Machine Learning*, pages 1256–1264.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Din, S. U., Shao, J., Kumar, J., Ali, W., Liu, J., and Ye, Y. (2020). Online reliable semi-supervised learning on evolving data streams. *Information Sciences*, 525:153–171.

Ditzler, G. and Polikar, R. (2011). Semi-supervised learning in nonstationary environments. In *The 2011 International Joint Conference on Neural Networks*, pages 2741–2748. IEEE.

Ditzler, G., Roveri, M., Alippi, C., and Polikar, R. (2015). Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25.

Domeniconi, C. and Gunopulos, D. (2001). Incremental support vector machine construction. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 589–592. IEEE.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80.

Dua, D. and Graff, C. (2017). UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`. Accessed February 21, 2021.

Dubois, D., Hüllermeier, E., and Prade, H. (2006). A systematic approach to the assessment of fuzzy association rules. *Data Mining and Knowledge Discovery*, 13(2):167–192.

Dyer, K. B., Capo, R., and Polikar, R. (2013). Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):12–26.

Effrosynidis, D., Symeonidis, S., and Arampatzis, A. (2017). A comparison of pre-processing techniques for twitter sentiment analysis. In *International Conference on Theory and Practice of Digital Libraries*, pages 394–406. Springer.

Ericsson (2019). Ericsson mobility report (june 2019). `https://www.ericsson.com/49d1d9/assets/local/mobility-report/documents/2019/ericsson-mobility-report-june-2019.pdf`. Accessed February 21, 2021.

Fan, W., Watanabe, T., and Asakura, K. (2009). Ratio rules mining in concept drifting data streams. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 2.

Fanaee-T, H. and Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127.

Feldman, R. and Dagan, I. (1995). Knowledge discovery in textual databases (kdt). In *Knowledge Discovery in Databases*, volume 95, pages 112–117.

Feng, Z., Wang, M., Yang, S., and Jiao, L. (2016). Incremental semi-supervised classification of data streams via self-representative selection. *Applied Soft Computing*, 47:389–394.

Fern, A. and Givan, R. (2003). Online ensemble learning: An empirical study. *Machine Learning*, 53(1-2):71–109.

Ferreira, R. S., Zimbrão, G., and Alvim, L. G. (2019). Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency. *Information Sciences*, 488:219–237.

Finner, H. (1993). On a monotonicity problem in step-down multiple test procedures. *Journal of the American Statistical Association*, 88(423):920–923.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Frost, M., Doryab, A., Faurholt-Jepsen, M., Kessing, L. V., and Bardram, J. E. (2013). Supporting disease insight through data analysis: refinements of the MONARCA self-assessment system. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 133–142. ACM.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. CRC Press.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer.

Gama, J., Medas, P., and Rodrigues, P. (2005). Learning decision trees from dynamic data streams. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 573–577.

Gama, J., Rocha, R., and Medas, P. (2003). Accurate decision trees for mining high-speed data streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 523–528.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–37.

Gao, J., Fan, W., Jiang, J., and Han, J. (2008). Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 283–291.

García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15(6):617.

García-Martínez, C., Lozano, M., Herrera, F., Molina, D., and Sánchez, A. M. (2008). Global and local real-coded genetic algorithms based on parent-centric crossover operators. *European Journal of Operational Research*, 185(3):1088–1113.

Gartner (2018). Gartner says worldwide wearable device sales to grow 26 percent in 2019. `https://www.gartner.com/en/newsroom/press-releases/2018-11-29-gartner-says-worldwide-wearable-device-sales-to-grow-`. Accessed February 21, 2021.

Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S. (2003). Mining frequent patterns in data streams at multiple time granularities. *Next Generation Data Mining*, 212:191–212.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Godberg, D. E. (1989). Genetic algorithms in search. *Optimization, and Machine Learning*.

Golbeck, J. and Hansen, D. (2011). Computing political preference among twitter followers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1105–1108.

Gomes, J. B., Krishnaswamy, S., Gaber, M. M., Sousa, P. A., and Menasalvas, E. (2012). Mobile activity recognition using ubiquitous data stream mining. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 130–141. Springer.

Gonçalves, P., Araújo, M., Benevenuto, F., and Cha, M. (2013). Comparing and combining sentiment analysis methods. In *Proceedings of the First ACM Conference on Online Social Networks*, pages 27–38.

Goyal, A. and Daumé III, H. (2011). Approximate scalable bounded space sketch for large data NLP. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 250–261.

Goyal, P., Challa, J. S., Shrivastava, S., and Goyal, N. (2020). Anytime frequent itemset mining of transactional data streams. *Big Data Research*, 21:100146.

Gravenhorst, F., Muaremi, A., Bardram, J., Grünerbl, A., Mayora, O., Wurzer, G., Frost, M., Osmani, V., Arnrich, B., Lukowicz, P., et al. (2015). Mobile phones as medical devices in mental disorder treatment: an overview. *Personal and Ubiquitous Computing*, 19(2):335–353.

Guha, S., Mishra, N., Motwani, R., and o'Callaghan, L. (2000). Clustering data streams. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 359–366. IEEE.

Gunning, D. (2017). Explainable Artificial Intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA)*, 2(2).

Han, J., Pei, J., Yin, Y., and Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87.

Haque, A., Khan, L., and Baron, M. (2016). SAND: Semi-supervised adaptive novel class detection and classification over data stream. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 1652–1658.

He, Y. and Zhou, D. (2011). Self-training from labeled features for sentiment analysis. *Information Processing & Management*, 47(4):606–616.

HewaNadungodage, C., Xia, Y., Lee, J. J., and Tu, Y.-c. (2013). Hyper-structure mining of frequent patterns in uncertain data streams. *Knowledge and Information Systems*, 37(1):219–244.

Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer.

Hofman, E. (2018). senti-py. `https://github.com/aylliote/senti-py`. Accessed February 21, 2021.

Hofmann, M. and Chisholm, A. (2016). *Text mining and visualization: case studies using open-source tools*, volume 40. CRC Press.

Holmes, G., Kirkby, R., and Pfahringer, B. (2005). Stress-testing hoeffding trees. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 495–502. Springer.

Hong, T.-P., Kuo, C.-S., and Chi, S.-C. (1999). Mining association rules from quantitative data. *Intelligent Data Analysis*, 3(5):363–376.

Hong, T.-P., Kuo, C.-S., and Chi, S.-C. (2001). Trade-off between computation time and number of rules for fuzzy mining from quantitative data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(05):587–604.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

Hosseini, M. J., Gholipour, A., and Beigy, H. (2016). An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. *Knowledge and Information Systems*, 46(3):567–597.

Huang, D., Koh, Y. S., and Dobbie, G. (2012). Rare pattern mining on data streams. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 303–314. Springer.

Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106.

Iosifidis, V. and Ntoutsi, E. (2019). Sentiment analysis on big sparse data streams with limited labels. *Knowledge and Information Systems*, pages 1–40.

Ishibuchi, H., Kaisho, Y., and Nojima, Y. (2009). Complexity, interpretability and explanation capability of fuzzy rule-based classifiers. In *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, pages 1730–1735. IEEE.

Ishibuchi, H. and Yamamoto, T. (2004). Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems*, 141(1):59–88.

Jackowski, K. (2014). Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers. *Pattern Analysis and Applications*, 17(4):709–724.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Jang, J.-S. and Sun, C.-T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159.

Java, A., Song, X., Finin, T., and Tseng, B. (2007). Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65.

Jensen, M. J. and Anstead, N. (2013). Psephological investigations: Tweets, votes, and unknown unknowns in the republican nomination process. *Policy & Internet*, 5(2):161–182.

Jin, C., Qian, W., Sha, C., Yu, J. X., and Zhou, A. (2003). Dynamically maintaining frequent items over a data stream. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 287–294.

John, O. P., Srivastava, S., et al. (1999). The big five trait taxonomy: History, measurement, and theoretical perspectives. *Handbook of personality: Theory and research*, 2(1999):102–138.

Karp, R. M., Shenker, S., and Papadimitriou, C. H. (2003). A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55.

Kolter, J. Z. and Maloof, M. A. (2005). Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd International Conference on Machine learning*, pages 449–456.

Kolter, J. Z. and Maloof, M. A. (2007). Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790.

Konsolakis, K., Hermens, H., Villalonga, C., Vollenbroek-Hutten, M., and Banos, O. (2018). Human behaviour analysis through smartphones. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 2, page 1243.

Kosina, P. and Gama, J. (2015). Very fast decision rules for classification in data streams. *Data Mining and Knowledge Discovery*, 29(1):168–202.

Kotler, J. and Maloof, M. (2003). Dynamic weighted majority: A new ensemble method for tracking concept drift. In *IEEE International Conference on Data Mining*, pages 123–130.

Kouloumpis, E., Wilson, T., and Moore, J. (2011). Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5.

Krawczyk, B., Minku, L. L., Gama, J., Stefanowski, J., and Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156.

Kum, H.-C., Pei, J., Wang, W., and Duncan, D. (2003). Approxmap: Approximate mining of consensus sequential patterns. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 311–315. SIAM.

Kuncheva, L. I. (2004). Classifier ensembles for changing environments. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer.

Kusumakumari, V., Sherigar, D., Chandran, R., and Patil, N. (2017). Frequent pattern mining on stream data using hadoop cantree-gtree. *Procedia Computer Science*, 115:266–273.

Kwon, Y., Kang, K., and Bae, C. (2014). Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems with Applications*, 41(14):6067–6074.

Last, M., Maimon, O., and Minkov, E. (2002). Improving stability of decision trees. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(02):145–159.

Leung, C. K.-S., Carmichael, C. L., and Hao, B. (2007). Efficient mining of frequent patterns from uncertain data. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 489–494. IEEE.

Leung, C. K.-S. and Hao, B. (2009). Mining of frequent itemsets from streams of uncertain data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1663–1670. IEEE.

Leung, C. K.-S., Mateo, M. A. F., and Brajczuk, D. A. (2008). A tree-based approach for frequent pattern mining from uncertain data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 653–661. Springer.

Li, H., Zhang, N., Zhu, J., Wang, Y., and Cao, H. (2018). Probabilistic frequent itemset mining over uncertain data streams. *Expert Systems with Applications*, 112:274–287.

Li, P., Wu, X., and Hu, X. (2010). Mining recurring concept drifts with limited labeled streaming data. In *Proceedings of 2nd Asian Conference on Machine Learning*, pages 241–252. JMLR Workshop and Conference Proceedings.

Ling, R. F. (1974). Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69(348):859–866.

Littlestone, N. and Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, 108(2):212–261.

Liu, H., Lin, Y., and Han, J. (2011). Methods for mining frequent items in data streams: an overview. *Knowledge and Information Systems*, 26(1):1–30.

Liu, H., Zhou, K., Zhao, P., and Yao, S. (2018). Mining frequent itemsets over uncertain data streams. *International Journal of High Performance Computing and Networking*, 11(4):312–321.

Liu, M. and Qu, J. (2012). Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 55–64.

Loo, H. R. and Marsono, M. N. (2015). Online data stream classification with incremental semi-supervised learning. In *Proceedings of the Second ACM IKDD Conference on Data Sciences*, pages 132–133.

Losing, V., Hammer, B., and Wersing, H. (2016). KNN classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th International Conference on Data Mining (ICDM 2016)*, pages 291–300. IEEE.

Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. (2004). Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302.

Lughofer, E. (2011). *Evolving Fuzzy Systems-Methodologies, Advanced Concepts and Applications*, volume 53. Springer.

Manerikar, N. and Palpanas, T. (2009). Frequent items in streaming data: An experimental evaluation of the state-of-the-art. *Data & Knowledge Engineering*, 68(4):415–430.

Manku, G. S. and Motwani, R. (2002). Approximate frequency counts over data streams. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 346–357. Elsevier.

Marascu, A. and Masseglia, F. (2006). Mining sequential patterns from data streams: a centroid approach. *Journal of Intelligent Information Systems*, 27(3):291–307.

Martínez-Ballesteros, M., Martínez-Álvarez, F., Troncoso, A., and Riquelme, J. C. (2011). An evolutionary algorithm to discover quantitative association rules in multidimensional time series. *Soft Computing*, 15(10):2065.

Martínez Cámara, E., Martín Valdivia, M. T., Perea Ortega, J. M., and Ureña López, L. A. (2011). Técnicas de clasificación de opiniones aplicadas a un corpus en español. *Procesamiento del Lenguaje Natural*, 47:163–170.

Masseglia, F., Cathala, F., and Poncelet, P. (1998). The psp approach for mining sequential patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 176–184. Springer.

Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. (2008). A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *2008 Eighth IEEE International Conference on Data Mining*, pages 929–934. IEEE.

Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. (2010). Classification and novel class detection in data streams with active mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 311–324. Springer.

Masud, M. M., Woolam, C., Gao, J., Khan, L., Han, J., Hamlen, K. W., and Oza, N. C. (2012). Facing the reality of data stream classification: coping with scarcity of labeled data. *Knowledge and information systems*, 33(1):213–244.

Matuszyk, P. and Spiliopoulou, M. (2015). Semi-supervised learning for stream recommender systems. In *International Conference on Discovery Science*, pages 131–145. Springer.

Mavrodiev, H. (2020). London bike sharing dataset. `https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset`. Accessed February 21, 2021.

MeaningCloud (2018). meaningcloud-python. `https://github.com/MeaningCloud/meaningcloud-python`. Accessed February 21, 2021.

Memar, M., Sadreddini, M. H., Deypir, M., and Fakhrahmad, S. M. (2011). A block-based approach for frequent itemset mining over data streams. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2011)*, volume 3, pages 1647–1651. IEEE.

Metwally, A., Agrawal, D., and El Abbadi, A. (2005). Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*, pages 398–412. Springer.

Meulman, J. J. (1992). The integration of multidimensional scaling and multivariate analysis with optimal transformations. *Psychometrika*, 57(4):539–565.

Minku, F. L., Inoue, H., and Yao, X. (2009). Negative correlation in incremental learning. *Natural Computing*, 8(2):289–320.

Minku, L. L. and Yao, X. (2011). DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):619–633.

Mirsky, Y., Shabtai, A., Shapira, B., Elovici, Y., and Rokach, L. (2017). Anomaly detection for smartphone data streams. *Pervasive and Mobile Computing*, 35:83–107.

Misra, J. and Gries, D. (1982). Finding repeated elements. *Science of Computer Programming*, 2(2):143–152.

Mohammad, S. M., Sobhani, P., and Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):1–23.

Mønsted, B., Mollgaard, A., and Mathiesen, J. (2018). Phone-based metric as a predictor for basic personality traits. *Journal of Research in Personality*, 74:16–22.

Montiel, J., Read, J., Bifet, A., and Abdessalem, T. (2018). Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(1):2915–2914.

Moreno, A. and Redondo, T. (2016). Text analytics: the convergence of big data and artificial intelligence. *International Journal of Interactive Multimedia and Artificial Intelligence*, 3(6):57–64.

Moreno-Ortiz, A. and Hernández, C. P. (2013). Lexicon-based sentiment analysis of twitter messages in spanish. *Procesamiento del Lenguaje Natural*, 50:93–100.

Muaremi, A., Gravenhorst, F., Grünerbl, A., Arnrich, B., and Tröster, G. (2014). Assessing bipolar episodes using speech cues derived from phone calls. In *International Symposium on Pervasive Computing Paradigms for Mental Health*, pages 103–114. Springer.

Nishida, K. (2008). Learning and detecting concept drift. *Information Science and Technology*.

Nishida, K. and Yamauchi, K. (2007). Adaptive classifiers-ensemble system for tracking concept drift. In *2007 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3607–3612. IEEE.

Noorbehbahani, F., Fanian, A., Mousavi, R., and Hasannejad, H. (2017). An incremental intrusion detection system using a new semi-supervised stream classification method. *International Journal of Communication Systems*, 30(4):e3002.

Oliver, N. and Flores-Mangas, F. (2006). Healthgear: a real-time wearable system for monitoring and analyzing physiological signals. In *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*. IEEE.

Orriols-Puig, A. and Casillas, J. (2011). Fuzzy knowledge representation study for incremental learning in data streams and classification problems. *Soft Computing*, 15(12):2389–2414.

Orriols-Puig, A., Casillas, J., and Bernadó-Mansilla, E. (2008a). First approach toward on-line evolution of association rules with learning classifier systems. In *Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 2031–2038. ACM.

Orriols-Puig, A., Casillas, J., and Bernadó-Mansilla, E. (2008b). Fuzzy-UCS: a michigan-style learning fuzzy-classifier system for supervised learning. *IEEE Transactions on Evolutionary Computation*, 13(2):260–283.

Osmani, V., Maxhuni, A., Grünerbl, A., Lukowicz, P., Haring, C., and Mayora, O. (2013). Monitoring activity of patients with bipolar disorder using smart phones. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, page 85. ACM.

Ovi, J. A., Ahmed, C. F., Leung, C. K., and Pazdor, A. G. (2019). Mining weighted frequent patterns from uncertain data streams. In *International Conference on Ubiquitous Information Management and Communication*, pages 917–936. Springer.

Oza, N. C. (2005). Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345. IEEE.

Oza, N. C. and Russell, S. (2001). *Online Ensemble Learning*. University of California, Berkeley.

Pan, W., Aharony, N., and Pentland, A. (2011). Composite social network for predicting mobile apps installation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Papadimitriou, S., Brockwell, A., and Faloutsos, C. (2003). Adaptive, hands-off stream mining. In *Proceedings 2003 VLDB Conference*, pages 560–571. Elsevier.

Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *International Conference on Database Theory*, pages 398–416. Springer.

Patnaik, D., Laxman, S., Chandramouli, B., and Ramakrishnan, N. (2013). A general streaming algorithm for pattern discovery. *Knowledge and Information Systems*, 37(3):585–610.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Pedrycz, W. (2018). *Granular Computing: Analysis and Design of Intelligent Systems*. CRC press.

Pedrycz, W. and Waletzky, J. (1997). Fuzzy clustering with partial supervision. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(5):787–795.

Polikar, R., Upda, L., Upda, S. S., and Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, part C (Applications and Reviews)*, 31(4):497–508.

Puiatti, A., Mudda, S., Giordano, S., and Mayora, O. (2011). Smartphone-centred wearable sensors network for monitoring patients with bipolar disorder. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3644–3647. IEEE.

Qin, K. and Wen, Y. (2018). Semi-supervised classification of concept drift data stream based on local component replacement. In *International CCF Conference on Artificial Intelligence*, pages 98–112. Springer.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R. (2014). *C4. 5: Programs for Machine Learning*. Elsevier.

Rai, P., Daumé III, H., and Venkatasubramanian, S. (2009). Streamed learning: one-pass SVMs. *arXiv preprint arXiv:0908.0572*.

Raïssi, C. and Poncelet, P. (2007). Sampling for sequential pattern mining: From static databases to data streams. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 631–636. IEEE.

Ramamurthy, S. and Bhatnagar, R. (2007). Tracking recurrent concept drift in streaming data using ensemble classifiers. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 404–409. IEEE.

Rangu, C., Chatterjee, S., and Valluru, S. R. (2017). Text mining approach for product quality enhancement:(improving product quality through machine learning). In *2017 IEEE 7th International Advance Computing Conference (IACC 2017)*, pages 456–460. IEEE.

Read, J., Bifet, A., Pfahringer, B., and Holmes, G. (2012). Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *International Symposium on Intelligent Data Analysis*, pages 313–323. Springer.

Rodríguez, F. J., Lozano, M., García-Martínez, C., and González-Barrera, J. D. (2013). An artificial bee colony algorithm for the maximally diverse grouping problem. *Information Sciences*, 230:183–196.

Rodríguez, J. J. and Kuncheva, L. I. (2008). Combining online classification approaches for changing environments. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 520–529. Springer.

Roesler, O. (2013). EEG eye state dataset. `https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State`. Accessed February 21, 2021.

Ronao, C. A. and Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235–244.

Roy, P., Khan, A., and Alonso, G. (2016). Augmented sketch: Faster and more accurate stream processing. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1449–1463.

Ruiz, E. and Casillas, J. (2018). Adaptive fuzzy partitions for evolving association rules in big data stream. *International Journal of Approximate Reasoning*, 93:463–486.

Rutkowski, L., Jaworski, M., Pietruczuk, L., and Duda, P. (2014). The cart decision tree for mining data streams. *Information Sciences*, 266:1–15.

Rutkowski, L., Pietruczuk, L., Duda, P., and Jaworski, M. (2012). Decision trees for mining data streams based on the mcdiarmid's bound. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1272–1279.

Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009). On-line random forests. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1393–1400. IEEE.

Sanchez-Valdes, D. and Trivino, G. (2015). Linguistic and emotional feedback for self-tracking physical activity. *Expert Systems with Applications*, 42(24):9574–9586.

Sancho-Asensio, A., Orriols-Puig, A., and Casillas, J. (2016). Evolving association streams. *Information Sciences*, 334:250–272.

Saponas, T., Lester, J., Froehlich, J., Fogarty, J., and Landay, J. (2008). iLearn on the iPhone: Real-time human activity classification on commodity mobile phones. *University of Washington CSE Tech Report UW-CSE-08-04-02*, 2008.

Sarker, I. H. and Salim, F. D. (2018). Mining user behavioral rules from smartphone data through association analysis. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 450–461. Springer.

Sayed-Mouchaweh, M. and Lughofer, E. (2012). *Learning in non-stationary environments: methods and applications*. Springer Science & Business Media.

Scholz, M. and Klinkenberg, R. (2007). Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28.

Shaker, A. and Hüllermeier, E. (2012). Iblstreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, 3(4):235–249.

Shao, J., Huang, C., Yang, Q., and Luo, G. (2016). Reliable semi-supervised learning. In *2016 IEEE 16th International Conference on Data Mining (ICDM 2016)*, pages 1197–1202. IEEE.

Sheskin, D. J. (2020). *Handbook of parametric and nonparametric statistical procedures*. CRC Press.

Shimojima, K., Fukuda, T., and Hasegawa, Y. (1995). RBF-fuzzy system with GA based unsupervised/supervised learning method. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems.*, volume 1, pages 253–258. IEEE.

Shmueli, E., Singh, V. K., Lepri, B., and Pentland, A. (2014). Sensing, understanding, and shaping social behavior. *IEEE Transactions on Computational Social Systems*, 1(1):22–34.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. d., and Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31.

Sobolewski, P. and Woźniak, M. (2017). SCR: simulated concept recurrence–a non-supervised tool for dealing with shifting concept. *Expert Systems*, 34(5):e12059.

Sogo, J. G. (2018). apicultur-python. `https://github.com/jgsogo/apicultur-python`. Accessed February 21, 2021.

Spark, A. (2021). Spark streaming. `http://spark.apache.org/streaming`. Accessed February 21, 2021.

Stefanowski, J. (2015). Adaptive ensembles for evolving data streams–combining block-based and online solutions. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 3–16. Springer.

Street, N. and Kim, Y. (2005). A streaming ensemble algorithm for large-scale classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,*, pages 377–382.

Sysoev, M., Kos, A., and Pogačnik, M. (2015). Noninvasive stress recognition considering the current activity. *Personal and Ubiquitous Computing*, 19(7):1045–1052.

Tan, J., Bu, Y., and Zhao, H. (2010). Incremental maintenance of association rules over data streams. In *2010 International Conference on Networking and Digital Society*, volume 2, pages 444–447. IEEE.

Tang, M., Nie, F., Pongpaichet, S., and Jain, R. (2017). Semi-supervised learning on large-scale geotagged photos for situation recognition. *Journal of Visual Communication and Image Representation*, 48:310–316.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Toscos, T., Faber, A., Connelly, K., and Upoma, A. M. (2008). Encouraging physical activity in teens can technology help reduce barriers to physical activity in adolescent girls? In *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*, pages 218–221. IEEE.

Trevisan, D. G., Sanchez-Pi, N., Marti, L., and Garcia, A. C. B. (2015). Big Data Visualization for Occupational Health and Security Problem in Oil and Gas Industry. In *Human Interface and the Management of Information. Information and Knowledge Design*, pages 46–54. Springer.

Troiano, R. P., Berrigan, D., Dodd, K. W., Masse, L. C., Tilert, T., McDowell, M., et al. (2008). Physical activity in the United States measured by accelerometer. *Medicine and Science in Sports and Exercise*, 40(1):181.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392.

Tu, Q., Lu, J.-f., Tang, J.-b., and Yang, J.-y. (2010). The FP-tree algorithm used for data stream. In *2010 Chinese Conference on Pattern Recognition (CCPR)*, pages 1–5. IEEE.

Tumasjan, A., Sprenger, T., Sandner, P., and Welpe, I. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 4.

Vaynman, S. and Gomez-Pinilla, F. (2006). Revenge of the "sit": how lifestyle impacts neuronal and cognitive health through molecular systems that interface energy metabolism with neuronal plasticity. *Journal of Neuroscience Research*, 84(4):699–715.

Veloso, B., Martins, C., Espanha, R., Azevedo, R., and Gama, J. (2020). Fraud detection using heavy hitters: a case study. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 482–489.

Ventruto, F., Pulimeno, M., Cafaro, M., and Epicoco, I. (2020). On frequency estimation and detection of heavy hitters in data streams. *Future Internet*, 12(9):158.

Vilares, D., Thelwall, M., and Alonso, M. A. (2015). The megaphone of the people? spanish sentistrength for real-time analysis of political tweets. *Journal of Information Science*, 41(6):799–813.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57.

Voss, M. W., Vivar, C., Kramer, A. F., and van Praag, H. (2013). Bridging animal and human models of exercise-induced brain plasticity. *Trends in Cognitive Sciences*, 17(10):525–544.

Wang, E. T. and Chen, A. L. (2009). A novel hash-based approach for mining frequent itemsets over data streams requiring less memory space. *Data Mining and Knowledge Discovery*, 19(1):132–172.

Wang, E. T. and Chen, A. L. (2011). Mining frequent itemsets over distributed data streams by continuously maintaining a global synopsis. *Data Mining and Knowledge Discovery*, 23(2):252–299.

Wang, H., Can, D., Kazemzadeh, A., Bar, F., and Narayanan, S. (2012). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120.

Wang, H., Fan, W., Yu, P. S., and Han, J. (2003a). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235.

Wang, J., Han, J., and Pei, J. (2003b). Closet+ searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 236–245.

Wang, K., Tay, S. H. W., and Liu, B. (1998). Interestingness-based interval merger for numeric association rules. In *4th International Conference on Knowledge Discovery and Data Mining, AAAI Press*, volume 98, pages 121–128.

Wang, S., Minku, L. L., and Yao, X. (2014). Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368.

Wang, Y., Cang, S., and Yu, H. (2019). A survey on wearable sensor modality centred human activity recognition in health care. *Expert Systems with Applications*, 137:167–190.

Wang, Y. and Li, T. (2018). Improving semi-supervised co-forest algorithm in evolving data streams. *Applied Intelligence*, 48(10):3248–3262.

Welford, B. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420.

Wen, Y.-M. and Liu, S. (2020). Semi-supervised classification of data streams by birch ensemble and local structure mapping. *Journal of Computer Science and Technology*, 35:295–304.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in Statistics*, pages 196–202. Springer.

Williams, C. and Gulati, G. (2008). What is a social network worth? facebook and vote share in the 2008 presidential primaries. In *Proceedings of the 104th Annual Meeting of the American Political Science Association*, page 17. Annual Meeting of the American Political Science Association.

Wilson, S. (1998). Generalization in the XCS classifier system. *Proceedings of Genetic Programming 1998*.

Wong, R. C.-W. and Fu, A. W.-C. (2006). Mining top-k frequent itemsets from data streams. *Data Mining and Knowledge Discovery*, 13(2):193–217.

Woodruff, D. P. (2016). New algorithms for heavy hitters in data streams (invited talk). In *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15–18, 2016*, pages 4:1–4:12. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Woolam, C., Masud, M. M., and Khan, L. (2009). Lacking labels in the stream: classifying evolving stream data with few labels. In *International Symposium on Methodologies for Intelligent Systems*, pages 552–562. Springer.

World Health Organization (2011). mHealth: new horizons for health through mobile technologies.

Wu, S., Yang, C., and Zhou, J. (2006). Clustering-training for data stream mining. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW 2006)*, pages 653–656. IEEE.

Wu, X., Li, P., and Hu, X. (2012). Learning from concept drifting data streams with unlabeled data. *Neurocomputing*, 92:145–155.

Xiao, Q., Tang, Z., and Chen, S. (2020). Universal online sketch for tracking heavy hitters and estimating moments of data streams. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 974–983. IEEE.

Xie, M. and Tan, L. (2019). An efficient algorithm for frequent pattern mining over uncertain data stream. In *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, volume 1, pages 84–88. IEEE.

Xu, W.-h., Qin, Z., and Chang, Y. (2011). Clustering feature decision trees for semi-supervised classification from high-speed data streams. *Journal of Zhejiang University SCIENCE C*, 12(8):615.

Yamada, S., Funayama, T., and Yamamoto, Y. (2015). Visualization of relations of stores by using association rule mining. In *2015 13th International Conference on ICT and Knowledge Engineering (ICT & Knowledge Engineering 2015)*, pages 11–14. IEEE.

Yan, X. and Han, J. (2003). CloseGraph: mining closed frequent graph patterns. In *Proceedings of the Nineth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 286–295.

Yang, H., Liu, H., and He, J. (2007). DELAY: a lazy approach for mining frequent patterns over high speed data streams. In *International Conference on Advanced Data Mining and Applications*, pages 2–14. Springer.

Yang, R., Ye, D., et al. (2020). Hybrid time decay model and probability decay window model for data stream closed frequent pattern mining. *Journal of Applied Science and Engineering*, 23(4):611–618.

Yen, S.-J., Lee, Y.-S., Wu, C.-W., and Lin, C.-L. (2009). An efficient algorithm for maintaining frequent closed itemsets over data stream. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 767–776. Springer.

Yoshida, S.-i., Hatano, K., Takimoto, E., and Takeda, M. (2011). Adaptive online prediction using weighted windows. *IEICE Transactions on Information and Systems*, 94(10):1917–1923.

Yu, Y., Guo, S., Lan, S., and Ban, T. (2008). Anomaly intrusion detection for evolving data stream based on semi-supervised learning. In *International Conference on Neural Information Processing*, pages 571–578. Springer.

Yun, U., Kim, D., Yoon, E., and Fujita, H. (2018). Damped window based high average utility pattern mining over data streams. *Knowledge-Based Systems*, 144:188–205.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60.

Zaki, M. J. and Hsiao, C.-J. (2002). CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pages 457–473. SIAM.

Zaki, M. J., Parthasarathy, S., Ogihara, M., and Li, W. (1997). New algorithms for fast discovery of association rules. In Heckerman, D., Mannila, H., and Pregibon, D., editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, pages 283–286. AAAI Press.

Zhang, H. (2004). Ithe optimality of naive bayes,". In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004*, volume 1, pages 1–6.

Zhang, P., Gao, B. J., Liu, P., Shi, Y., and Guo, L. (2012). A framework for application-driven classification of data streams. *Neurocomputing*, 92:170–182.

Zhang, P., Gao, B. J., Zhu, X., and Guo, L. (2011). Enabling fast lazy learning for data streams. In *2011 IEEE 11th International Conference on Data Mining*, pages 932–941. IEEE.

Zhang, P., Zhu, X., Tan, J., and Guo, L. (2010). Classifier and cluster ensembles for mining concept drifting data streams. In *2010 IEEE International Conference on Data Mining*, pages 1175–1180. IEEE.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: an efficient data clustering method for very large databases. *ACM Sigmod Record*, 25(2):103–114.

Zhao, Q. L., Jiang, Y. H., and Xu, M. (2010). Incremental learning by heterogeneous bagging ensemble. In *International Conference on Advanced Data Mining and Applications*, pages 1–12. Springer.

Zhu, X. (2010). Stream data mining repository. `http://www.cse.fau.edu/~xqzhu/stream.html`. Accessed February 21, 2021.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130.

Zhu, X. and Jin, R. (2009). Multiple information sources cooperative learning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1369–1376.

Zihayat, M. and An, A. (2014). Mining top-k high utility patterns over data streams. *Information Sciences*, 285:138–161.

Zihayat, M., Chen, Y., and An, A. (2017). Memory-adaptive high utility sequential pattern mining over data streams. *Machine Learning*, 106(6):799–836.

Zimmerann, M., Ntoutsi, E., and Spiliopoulou, M. (2014). A semi-supervised self-adaptive classifier over opinionated streams. In *2014 IEEE International Conference on Data Mining Workshop*, pages 425–432. IEEE.

Žliobaitė, I. (2010). *Adaptive Training Set Formation*. PhD thesis, Vilnius University.

Žliobaitė, I., Bifet, A., Pfahringer, B., and Holmes, G. (2011). Active learning with evolving streaming data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 597–612. Springer.

Žliobaitė, I., Bifet, A., Pfahringer, B., and Holmes, G. (2013). Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):27–39.

Žliobaitė, I. and Gabrys, B. (2012). Adaptive preprocessing for streaming data. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):309–321.