UNIVERSIDADE FEDERAL FLUMINENSE

LEANDRO BOTELHO ALVES DE MIRANDA

TOWARDS MACHINE LEARNING-BASED ADAPTIVE HUMAN ACTIVITY RECOGNITION

NITERÓI

2022

UNIVERSIDADE FEDERAL FLUMINENSE

LEANDRO BOTELHO ALVES DE MIRANDA

TOWARDS MACHINE LEARNING-BASED ADAPTIVE HUMAN ACTIVITY RECOGNITION

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense, in partial fulfillment of the requirements for the Degree of Doctor in Computing. Area:. Computer Science

Advisor:
JOSÉ VITERBO

Co-Advisor:
FLÁVIA BERNARDINI

NITERÓI

2022

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

M672t Miranda, Leandro Botelho Alves de
TOWARDS MACHINE LEARNING-BASED ADAPTIVE HUMAN ACTIVITY RECOGNITION / Leandro Botelho Alves de Miranda ; José Viterbo Filho, orientador ; Flávia Cristina Bernardini, coorientadora. Niterói, 2022.
155 f. : il.

Tese (doutorado)-Universidade Federal Fluminense, Niterói, 2022.

DOI: http://dx.doi.org/10.22409/PGC.2022.d.07530492446

1. Aprendizado de Máquina. 2. Produção intelectual. I. Viterbo Filho, José, orientador. II. Bernardini, Flávia Cristina, coorientadora. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

LEANDRO BOTELHO ALVES DE MIRANDA

TOWARDS MACHINE LEARNING-BASED ADAPTIVE HUMAN ACTIVITY RECOGNITION

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense, in partial fulfillment of the requirements for the Degree of Doctor in Computing. Area: Computer Science

Approved in July, 2022.

THESIS COMMITTEE

Prof. JOSÉ VITERBO FILHO - Advisor, UFF

Prof. FLÁVIA CRISTINA BERNARDINI - Co-Advisor, UFF

Prof. ALINE MARINS PAES CARVALHO, UFF

Prof. FLÁVIA COIMBRA DELICATO, UFF

Prof. ANDRÉ CARLOS PONCE DE LEON FERREIRA DE CARVALHO, USP

Prof. EDUARDO SOARES OGASAWARA, CEFET-RJ

Niterói

2022

For God, Family and Friends.

Acknowledgements

Primeiramente, agradeço a Deus por todas as graças alcançadas ao longo dessa jornada, sendo primordial em minha vida.

A Lúcia, por todo carinho, amor e apoio ao longo destes anos, sendo uma pessoa que me incentivou bastante em minhas decisões pessoais. Não posso esquecer da minha cachorrinha Lua, que sempre trazia seu brinquedinho em momentos estressantes desta pesquisa.

Aos meus pais Leonildo e Lilian, que desde criança me ensinaram o caminho dos estudos e a perseverança em uma formação melhor, além de claro, todo carinho, amor e afeto recebido por eles. Ao meu irmão, Lucas Botelho, que teve participação importante em me apoiar em decisões ao longo do doutorado. E a todos os meus familiares que certa forma torceram e me apoiaram em minha jornada.

Aos meus orientadores, José Viterbo e Flávia Bernardini que são os pilares na minha formação como pesquisador. Agradeço constantemente a cada palavra de formação como pesquisador quanto cada dica para "viver esta nossa vida". Minhas virtudes acadêmicas positivas estão bem associados aos seus direcionamentos.

Aos membros da banca, pois obtive comentários essenciais e precisos em vários aspectos deste trabalho, permitindo correções essenciais e novos caminhos para esta pesquisa.

À Universidade Federal Fluminense, que forneceu seu espaço como ambiente de pesquisa. Juntamente com o corpo docente do Instituto de Computação, onde eu pude adquirir experiência e formação acadêmica. Em especial, gostaria de agradecer professores que fizeram parte desta caminhada como professores parceiros em projetos de pesquisas como Flávia Delicato, Débora Saade; além destes agradecer ao Professor Luís Kowada pelo período de coordenação na Tutoria-UFF e ao acompanhamento espiritual dado.

Às empresas que puderam me custear ao longo do Doutorado, como CAPES, UFF (durante a tutoria), Fundação Euclides da Cunha e ISABO. Todos puderam me ajudar financeiramente durante a minha formação. Também agradeço as instituições parceiras

deste projetos como a Seplag, Niterói e a Dell EMC, RJ.

Gostaria de agradecer a todos os amigos que me ajudaram na UFF, onde eu pude compartilhar experiências acadêmicas e conhecimentos de vida como Leonardo Pio, Heder Dorneles, Carlos Pantoja, Eduardo Oliveira, Raíssa Barcelos, Gelson Schneider, Jânio Carlos, Igor Garcia, Joris Guerin; os colegas em projetos e na Tutoria; e as todos os colegas não citados, mas que foram importantes em minha formação.

Agradecer aos meus amigos de Maceió, que puderam me apoiar em minhas decisões. Enfim são inúmeras pessoas para eu poder agradecer, mas quem ler essa seção de agradecimentos e de alguma forma, eu pude ajudar em sua vida, se sinta também agradecido.

Resumo

O Reconhecimento de Atividades Humanas (HAR, do inglês *Human Activity Recognition*) envolve o uso de informações de contexto para inferir as atividades que um usuário realiza em suas tarefas diárias. O HAR tem sido amplamente estudado usando diversos paradigmas, como diferentes abordagens de raciocínio, incluindo técnicas probabilísticas, baseadas em regras, estatísticas, raciocínio lógico ou aprendizado de máquina (ML, do inglês *Machine Learning*), ajustando modelos de inferência para reconhecer ou prever atividades do usuário. O ML para HAR permite que as atividades possam ser reconhecidas e até previstas através da análise de dados coletados de diferentes sensores, com maior precisão do que outros paradigmas. Em domínios de aplicativos HAR, como ambientes inteligentes, cidades inteligentes e domínios de e-saúde; a Computação Sensível ao Contexto (CAC, do inglês *Context-Aware Computing*) fornece uma infraestrutura para otimizar o projeto e a construção de soluções de software em cenários onde muitos sensores e dados são essenciais. A adaptabilidade é crucial para esses sistemas, pois um sistema usando um modelo de inferência obsoleto pode degradar a qualidade de reconhecimento da atividade. Em uma revisão sistemática da literatura (SLR, do inglês *Systematic Literature Review*), identificamos o estado da arte sobre o uso de ML para HAR em CACs. Observamos que ML oferece abordagens viáveis para construir modelos de inferência para HAR usando diferentes abordagens. Além disso, também podemos identificar três principais desafios ou lacunas, especialmente relacionados ao aprendizado de fluxo de dados e também para estratégias adaptativas. Primeiro, o uso de estratégias baseadas em lote pode elevar ao consumo excessivo de memória na implantação de modelos de aprendizado de máquina em sistemas para CACs. Em segundo lugar, o entendimento do uso de estratégias adaptativas em sistemas HAR e, consequentemente, explorar novas soluções para recomendar modelos adaptativos para novos usuários. Terceiro, há uma falta de abordagens para HAR e adaptativas em CACs. A partir do primeiro desafio, concluímos que o uso do aprendizado de fluxo de dados em um cenário com poucos recursos de memória é relevante. A partir do segundo desafio, concluímos que os melhores modelos para um usuário alvo são aqueles treinados com uma partição que apresenta características topológicas semelhantes à partição formada pelo usuário alvo. Finalmente, para abordar a terceira lacuna, são necessários novos algoritmos para lidar com HAR em sistemas para CACs. Este trabalho tem como objetivo propor soluções adaptativas robustas para o domínio HAR, utilizando algoritmos baseados em aprendizado de fluxo de dados (DSL, *Data Stream Learning*). Ao longo desta pesquisa, utilizamos um conjunto de algoritmos para DSL (de diferentes métodos) abrangendo métricas tradicionais de avaliação e validação presentes no estado da arte. Propusemos um novo algoritmo para desenvolver Reconhecimento de Atividades Humanas Complexas Sequenciais em sistemas para CACs, que concluímos que tem melhor poder preditivo em cenários com menos atividades atômicas presentes nesses sistemas.

Palavras-chave: Aprendizado de Fluxo de dados, Reconhecimento de Atividades Humanas, Computação Sensível ao Contexto, Adaptabilidade.

Abstract

Human Activity Recognition (HAR) involves the use of context information to infer the activities that a user performs in his/her daily tasks. HAR has been extensively studied using diverse paradigms, such as different reasoning approaches, including probabilistic, rule-based, statistical, logical reasoning, or Machine Learning (ML) techniques, fitting inference models to recognize or predict user activities. ML for HAR allows that activities can be recognized and even predicted through analyzing collected data from different sensors, with greater accuracy than other paradigms. In HAR application domains such as ambient intelligence, smart cities, and e-health domains, Context-Aware Computing (CAC) provides an infrastructure to streamline the design and construction of software solutions in scenarios where many sensors and data are essential. Adaptability is crucial for these systems, because a system using an obsolete inference model can degrade the activity's recognition quality. In a systematic literature review (SLR), we identified the state-of-the-art on the use ML for HAR in CACs. We observed that ML offers viable approaches to construct inference models for HAR using different ML approaches, including batch learning and adaptive learning. In addition, we could also identify three main challenges or gaps, especially related to data stream learning and adaptive strategies. First, the use of batch learning strategies may lead to excessive memory consumption in the deployment of machine learning models in CACs systems. The second is understanding the use of adaptive strategies in HAR systems, and consequently, it is worth exploring new solutions for recommending adaptive models for new users. Third, there is a lack of adaptive HAR approaches in CACs. From the first challenge, we concluded that the use of data stream learning in a scenario with low memory resources is relevant. From the second challenge, we concluded that the best models for a target user are those trained with a partition that presents topological characteristics similar to the partition formed by the target user. Finally, for tackling the third gap, new algorithms to deal with HAR in CACs systems are necessary. This work aims to propose robust adaptive solutions to the HAR domain, using algorithms based on data stream learning (DSL). Throughout this research, we used a set of DSL algorithms (from different methods) covering traditional evaluation and validation metrics present in the state-of-the-art. We proposed a novel algorithm to develop sequential complex HAR in CACs systems, which we concluded that has better predictive power in scenarios with fewer atomic activities present in these systems.

Keywords: Data Stream Learning, Human Activity Recognition, Context-Aware Computing, Adaptability.

List of Figures

List of Tables

List of Abbreviations and Acronyms

AAL	:	Ambient Assisted Living
AHAR	:	Atomic Human Activity Recognition
AC	:	Arbitrary Classifier
ADL	:	Ambient Daily Living
CAC	:	Context-aware Computing
CAM	:	Context-aware Middlewares
CHAR	:	Complex Human Activity Recognition
DSL	:	Data Stream Learning
LAC	:	Laboratory for Advanced Collaboration
LFCT	:	Learning from Concept Drift
f	:	Unknown Function
$\hat{f}$	:	Approximated Function of f Function
F	:	Source
h	:	Machine Learning Classifier;
h_o	:	Online Learning Classifier;
HAR	:	Human Activity Recognition
OM	:	Online Models
M-Hub	:	Mobile Hub
ML	:	Machine Learning
Mo_1	:	Model that uses a adaptive strategy
Mb_2	:	A model that does not use a adaptive strategy
MR-UDP	:	Mobile Reliable User Datagram Protocol
S	:	Set of Samples
SLR	:	Systematic Literature Review
SCHAR	:	Sequential Complex Human Activity Recognition
t	:	Time
u	:	Number of users
U	:	A respective User
UFF	:	Universidade Federal Fluminense

W	:	Window size in a data stream
x	:	A sample of S in a database
X_t	:	A sample of S in Data Stream
y	:	Classifier Target

Thesis Publications

This section describes the main publications throughout the Ph.D. process.

Related Works

- Miranda, L., Viterbo, J., & Bernardini, F. (2022). A survey on the use of machine learning methods in context-aware middlewares for human activity recognition. Artificial Intelligence Review, 55(4), 3369-3400.

- Miranda, L., Viterbo, J., & Bernardini, F. (2020, October). Impact of Memory Control on Batch Learning in Human Activity Recognition Scenario in Comparison to Data Stream Learning. In Mexican International Conference on Artificial Intelligence (pp. 145-157). Springer, Cham.

Other Publications in Journals and Conferences

This subsection describes other publications during the doctorate, fruits of the master's work, and partnerships with colleagues in the research group, which were important for his evolution as a researcher.

- Barcellos, R., Viterbo, J., Miranda, L., Bernardini, F., Maciel, C., & Trevisan, D. (2017, June). Transparency in practice: using visualization to enhance the interpretability of open data. In Proceedings of the 18th Annual International Conference on Digital Government Research (pp. 139-148).

- Faial, D., Bernardini, F., Miranda, L., & Viterbo, J. (2019, September). Anomaly detection in vehicle traffic data using batch and stream supervised learning. In EPIA Conference on Artificial Intelligence (pp. 675-684). Springer, Cham.

- Faial, D., Bernardini, F., Meza, E. M., Miranda, L., & Viterbo, J. (2020, July). A methodology for taxi demand prediction using stream learning. In 2020 International Conference on Systems, Signals and Image Processing (IWSSIP) (pp. 417-422). IEEE.

- Fernandes, L. F. D. O., Bernardini, F., Meza, E. M., Miranda, L., & Viterbo, J. (2020, July). Energy Consumption Prediction using Data Stream Learning for Commercial Buildings. In 2020 International Conference on Systems, Signals and Image Processing (IWSSIP) (pp. 441-446). IEEE.

- de Sousa Pinto, A., Bernardini, F., Miranda, L., Viterbo, J., & Meza, E. M. (2020, July). An exploratory study using stream learning algorithms to predict duration time of vehicle routes. In 2020 International Conference on Systems, Signals and Image Processing (IWSSIP) (pp. 299-304). IEEE.

- Miranda, L., Viterbo, J., & Bernardini, F. (2020). Towards the Use of Clustering Algorithms in Recommender Systems.

- Miranda, L., Viterbo, J., & Bernardini, F. (2022). A framework for spatial regionalization composed of novel clustering-based algorithms under spatial contiguity constraints. Transactions in GIS.

Summary

Chapter 1

Introduction

Human Activity Recognition (HAR), based on sensors data, has emerged as a key research area in ubiquitous computing. Activity recognition systems focus on inferring the current activities of users by leveraging the rich sensory environment [89, 30]. Human Activity Recognition can be used in smart health applications to alert stakeholders that a person of interest suffered an accident at home [38]; monitoring of the elderly people, identifying people that fall, or are not moving for a long time, and so on [33]; or detecting which type of activity a person is doing, like walking, eating and cooking [49]. Also, HAR can be used for surveillance and anti-crime securities (for instance, people walking much more faster, making others to fall, may indicate a robbery or a fugitive of a crime scene [111]), life logging [39], sports assistance of performance [74] and so on.

Essentially, HAR can use context information and sensors data for inferring the activities a user performs in his daily tasks. Over the years, several researchers have developed methods, systems and frameworks for HAR in low complexity scenarios [48, 110]. In scenarios with a great number of sensors and a complex context management infrastructure, HAR services may be provided and supported by context-aware middleware (CAM) solutions, thus avoiding scalability, data annotation, and robustness problems for the solutions developers [109, 142].

In addition, considering the Machine Learning (ML) paradigm, many methods and algorithms are appearing in literature as an excellent way for constructing inference models to many different application scenarios, presenting good results, such as *healthcare* [119, 120], *traffic monitoring* [46, 34], *sports assistance* [130, 74] and others. From the point of view of data availability [20], ML algorithms can be basically divided into two different approaches, which are (i) Batch Learning (BL), requiring that all collected data must be available for constructing the model and, in case of adapting the model, they suffer

from the catastrophic forgetting issue, being required that all previous and recent data be collected for constructing a new model; and (ii) Data Stream Learning (DSL), evolving the model when new data arrives from the stream, and can also consider concept drift in data. However, although traditional techniques like SVM and Hidden Markov models have been extensively used in HAR systems, prior studies have not targeted the different approaches of ML in Context-Aware Systems that provide HAR capabilities. All of these approaches are important to be considered for constructing inference models to HAR estimation and inference, or as the users can change the way they move over time, requiring model adaptation over time; or data may not be correctly or even labeled, requiring active labeling methods. The full integration of these ML approaches (and capabilities) directly within HAR can improve the engineering, (including design), of context-aware applications.

1.1 Contextualization

Traditional techniques for recognizing human activities in context-aware systems deal with data continuously transmitted by various sensors [142]. Sensory data is a data stream that contains unlimited data arriving at high speed. Therefore, it is increasingly difficult to assume scenarios where data is static over time in human activity recognition scenarios [4]. Dynamic changes in activities that reflect variations in user activities are expected and natural. Therefore, the use of online machine learning concepts (*Data Stream Learning*) in activity recognition becomes important for the quality and maintenance of the system's effectiveness.

In context-aware scenarios, within limited hardware resources (such as HAR in smartwatches, mobiles, and other wearable technologies), there is a high demand for cheaper algorithms for constructing inference models. They should be able to deal with concept drift. They should need a low volume of labeled data. Another example of a scenario is the real-time classification of complex activities. In this case, devices with limited resources need to be maintained by a solid infrastructure to lead data coming from heterogeneous sensors and support the machine learning model's storage. In these scenarios, in particular when dealing with large amounts of data that need to be processed locally and timely, robust infrastructures may also be needed, such as edge computing [133].

Furthermore, there is a limitation involving the model's adaptability in the level of knowledge acquisition. As evolve time, and the system collects data from new users,

machine learning models (usually in batch) struggle to extract new knowledge or improve the model's reasoning ability. For the most part, the designers of these systems generally discard the current model, and from new data, create a new model or feed this model with new knowledge from this new data [59, 21]. Moreover, since most systems use models based on data from several users (multi-user), it is important to have models that are more robust to the characteristics of the activities of a specific user.

In addition, there is a challenge in updating models that deal with Complex Human Activity Recognition (CHAR). A complex activity is composed of a set of activities (set of states), where these activities can be done by a set of users (evidence). The state-of-the-art proposes different approaches (described in Chapter 3). However, few works commonly used the Sequential Complex Activity Recognition (SCHAR). SCHAR uses parts of activities that involve great physical effort and are represented sequentially (such as falls, activities of bodybuilding). Therefore, it is necessary to study how many activities are necessary to represent a sequential complex activity, that adapts quickly and volatility.

1.2 Research Questions

Motivated by the contextualization of the problem involving the recognition of human activities, this thesis formulated some research questions:

- **Q1:** *Does the use of data stream learning models result better than batch learning models with limited infrastructure in implementing HAR techniques?* This question aims to analyze the need to use online machine learning approaches (data stream learning) compared to traditional learning approaches (batch learning), in scenario with limited memory consumption.
- **Q2:** *Are machine learning models able to adapt to similar users?* In this question, we seek to answer whether there is a possibility of maintaining the quality of machine learning models while increasing the data stream for a respective user. In essence, it is to propose a new approach to recommend a model similar to a respective user through the use of data stream learning.
- **Q3:** *Is it useful to create a model for complex sequential activities that can self-adapt to the data stream?* The objective is to propose a new algorithm for sequential complex human activities based on stream learning, presenting better performance than state-of-the-art algorithms.

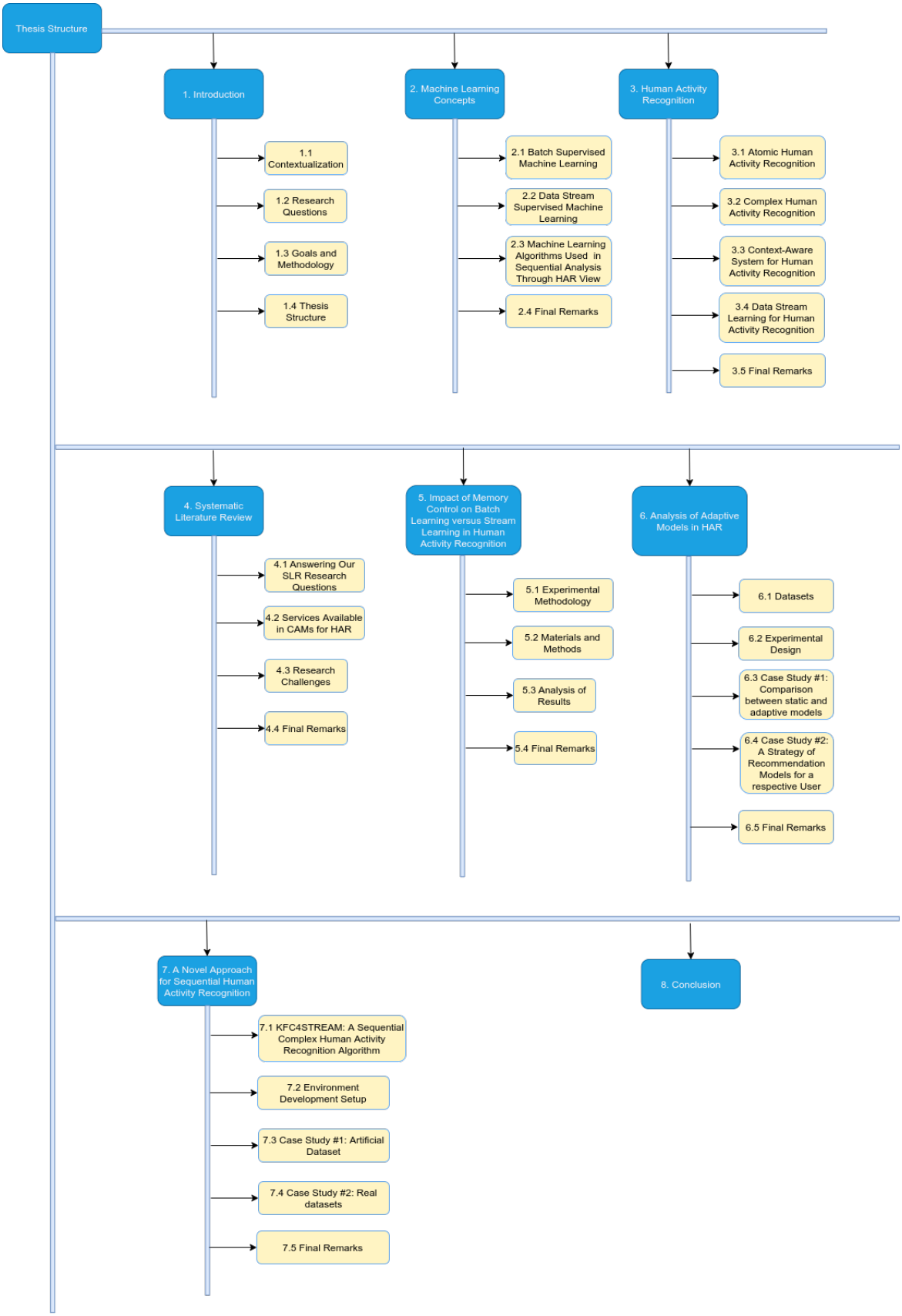

Fig. 1.1. Outline (organization) of this work.

1.3 Thesis Structure

Figure 1.1 presents the outline of this thesis, enrolling chapter by chapter through sequence workflow.

Chapter 2 presents two machine learning paradigms that will be useful for constructing the thesis. The first can fit stable (stationary) models that do not change over time, known as batch learning. Moreover, the second paradigm can fit non-stationary models, whose models undergo gradual changes over time. This learning is known as stream-learning.

Chapter 3 describes the problem of Human Activity Recognition in a vision of context-aware systems. This chapter presents the different types of Human Activities Recognition and research areas.

Chapter 4 describes how the entire process of selecting primary and secondary studies was carried out, with criteria for collecting and describing the motivation of the research questions proposed by this work. The chapter summarizes the primary services we found in Context-Aware Middlewares (CAM) for tackling HAR problems, considering the steps usually executed in the HAR solutions. Moreover, this chapter presents discussions on gaps and research challenges.

Chapter 5 compares the behavior of batch learning and data stream learning algorithm for constructing inference models in HAR scenarios with limited resources in ubiquitous environment. For this, we used in our methodology a *memory control module* for restricting memory consumption in batch learning.

Chapter 6 describes the importance of using adaptive strategies in HAR systems. In this chapter, we carried out two studies relevant to this case. First, we compared the use of adaptive and non-adaptive strategies. The second study proposes a model recommendation system based on user characteristics when performing their activities.

Chapter 7 describes a novel algorithm for stream learning based on Sequential Complex Human Activity Recognition. The experiment is based on simulated datasets on sequential complex human activity recognition. This dataset was created from concepts at the start-of-the-art and validated.

Chapter 8 presents the final remarks about this research. The chapter describes some limitations of this research and future works.

Chapter 2

Machine Learning Concepts

As previously mentioned, ML algorithms can be basically divided into two different approaches, which are (i) Batch Learning and (ii) Data Stream Learning. In batch learning approach, the entire training dataset is previously available for constructing the inference model. In this way, an algorithm learns a model after iteratively processing all the available data. Mainly, the examples are generated according to a stationary random distribution [51]. On the other hand, considering a set of sensors in which they send information in real-time to several devices using data based on transient data stream, batch learning may face a big challenge. Online learning, more recently called the Data Stream Learning approach, can modify the current model by adding new information [20]. This chapter briefly reviews those topics in different sections.

2.1 Batch Supervised Machine Learning

In supervised learning, the algorithms present a set S of input samples, where $S = \{x_1, x_2, \cdots, x_n\}$, where N is the number of samples, chosen from a domain $X = \{(x_1, y_1), \cdots (x_n, y_n)\}$ with unknown, random, fixed distribution, where $y = f(x)$, where f is an unknown function. We have that x_i are *M-dimensional* vectors characterized as follows $(x_{i1}, x_{i2}, \cdots, x_{iM})$, with values that can be discrete or continuous. The values y_i are usually associated with a discrete set of classes C_k, where k is the number of classes. This representation is for cases where we have a classification problem of single label.

Let S, a set of examples for a learning algorithm, a classifier h learns an approximate function $\hat{f}$ from the unknown function f. Given new examples x_i, h predicts a corresponding value y, using $\hat{f}$.

In batch machine learning, all training data sets are available in memory. An algorithm learns a decision model after iteratively processing all the data. Essentially the examples are generated according to a stationary distribution at random.

However, these approaches have no notion of adaptation or refinement of the already built model. Under realistic conditions, changing activities can occur over time. These changes include modifying the user's activity patterns, such as personalization (e.g., walking for one user can be represented as running for another); forgetting (activities that should be abandoned), and novel activities (to detect new activities). The latter even exists in approaches that use batch learning [97]. However, it requires model training on the available data representing the new concepts.

A solution to these existing problems in HAR is the use of stream learning techniques. A discussion of the using batch machine learning in HAR was given after the state-of-the-art review process described in Chapter 4. In the next section, the concept of stream learning is described and its application in scenarios for Human Activity Recognition.

2.2 Data Stream Supervised Machine Learning

On stream supervised machine learning scenarios, the studies cover continuous data stream [83]. A stream of data is a sequence of data instance (x_t, y_t) as described in Fig. 2.1, arriving one data at a time extracted from an unknown probability distribution $\rho_t(x_t, y_t)$; for time $t = \{1, 2, 3, ..., T\}$, where y is the corresponding class label. We assume that an online classifier h_o takes a new entry x_t in a time range t and then predicts its label. After some time, the current class label y_t is available and is used by h_o to evaluate predictive performance and provide additional information for training updates. This entire process will be repeated in the following steps. Most supervised learning algorithms based on data classification adopt this technique.

There are three ways [83, 127] of labeling in a continuous data stream: manual, automatic, and observational. Overall, the process of manually labeling each part of the data stream is costly and time-consuming. Automatic labeling is performed by configuring the application at the time of data collection. In the observational method, learning models are initially trained in automatic configurations; however, users can intervene by manually marking data streams in case of discrepancies.

However, DSL models present some constraints, as mentioned in Bifet et al. [18]: First (1), The amount of data that has arrived and will arrive in the future is substantial; in

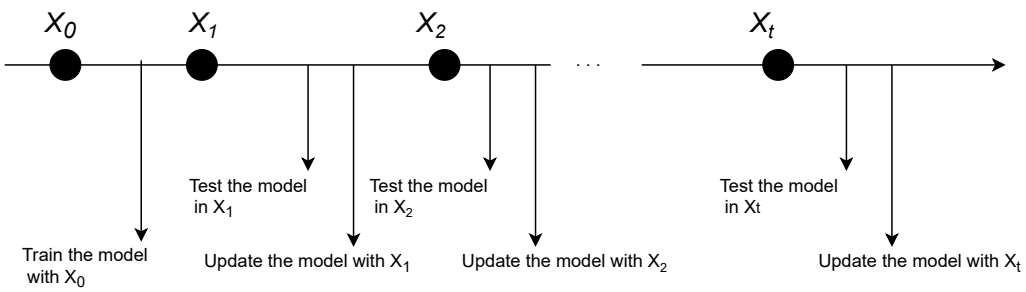

Fig. 2.1. Example of knowledge process from data stream models.

fact, the sequence is potentially infinite. Thus, it is impossible to store it all. Only a small **extract** can be computed and stored, and the rest of the information is not used. Even if the information could be stored, it would be unfeasible to go over it for further processing. Secondly (2), the speed of arrival is immense, so each particular element must be processed essentially in real-time and then discarded. Finally (3), the distribution generating the items can change over time. Thus, data from the past may become irrelevant to the current **extract**.

Constraints (1) and (2) limit the amount of memory and time-per-item the streaming algorithm can use. Intense research on the Data Stream algorithms has produced many techniques using few resources (memory and time-per-item), usually combined with the sliding-window technique (discussed after). Constraint (3), the importance to adapt to time changes. A typical approach for dealing is based on the use of *sliding windows*: The algorithm keeps a window of size W containing the last W data objects that have arrived (say, in the last W time steps). When a new object arrives, the oldest element in the window is deleted to make place for it. The **extract** of the Data Stream is at every time computed or rebuilt from the data in the window only. If W is of moderate size, this essentially takes care of the requirement to use low memory.

Over the years, researchers in machine learning have focused on the batch paradigm. However, the demand encourages mathematical models to present capacity processing for a large volume of information (that evolves over time according to non-stationary distribution). In this scenario, it is necessary to permanently maintain a robust decision model with the current state from newly generated data. Thus, incremental learning techniques are needed, modifying the current model by adding newly available information. In the presence of a non-stationary distribution, the system needs to incorporate knowledge elimination techniques, where some information or concepts are no longer needed, given the current state of the problem. Learning from a data stream requires incremental learning. This algorithms work with limited computational resources and can detect evolution in

concept drift [44].

Concept Drift [44] means that the process that generates the data evolves over time. The indication of changes in the concept reflects in some way in the training sets. Older observations, which characterize the behavior of data in the past, become obsolete for the current state under observation. Variations of changes can occur in attributes that are no longer observed or in the properties of observed attributes. In general, algorithms use adaptive strategies at regular intervals without considering whether there has been any change. Some works [54, 75] offer approaches that allow the visualization of important information from the analysis process (indicating points of change or time windows where change occurs) and quantifying the degree of change.

In pervasive scenarios, the literature indicates that concept drift must take different forms. Let F be a source. Assuming that the concept of a sequence of instances changes from F_i to F_j, we call it *sudden drift*, if at time t, F_i is suddenly replaced by F_j. Instead of an abrupt change (*sudden drift*), for example, a source is changed during a brief period of time after the change point t, the *gradual drift* refers to a change with a period where both sources, F_i and F_j, are active. Over time, the sampling probability of the source F_i decreases, the sampling probability of the source F_j increases until F_i is completely replaced by F_j. There is another type of gradual drift called *incremental* or *stepwise drift*. With more than two sources, the difference between sour is minimal, so *drift* is only noticed when looking at a longer period. *Recurring Concepts* means that the previously active concept reappears after some time. This concept is certainly not periodic. Therefore, there is no predictive information about when the source will reappear.

Figure 2.2 illustrates the main types of concept drift, assuming uni-dimensional data, where the average of the data characterizes a source. We only describe the data of a class.

A central feature of the data stream model is that streams evolve over time, and algorithms must react to the change (Concept drift). Change detection in data has a long tradition in statistics [20]. Neither all methods are apt for streaming because they require several passes over the data. We describe only a few streaming-friendly tests used in this thesis.

The **cumulative sum (CUSUM)** test [20] is designed to give a warning when the mean of the input data significantly deviates from its previous value. Given a sequence of observations x_t, define $z_t = (x_t - \mu)/\sigma$, where μ is the expected value of the x_t and σ is their standard deviation; if μ and σ are not known a priori, they are estimated from the sequence itself. Then CUSUM computes the indices and gives a warning. The

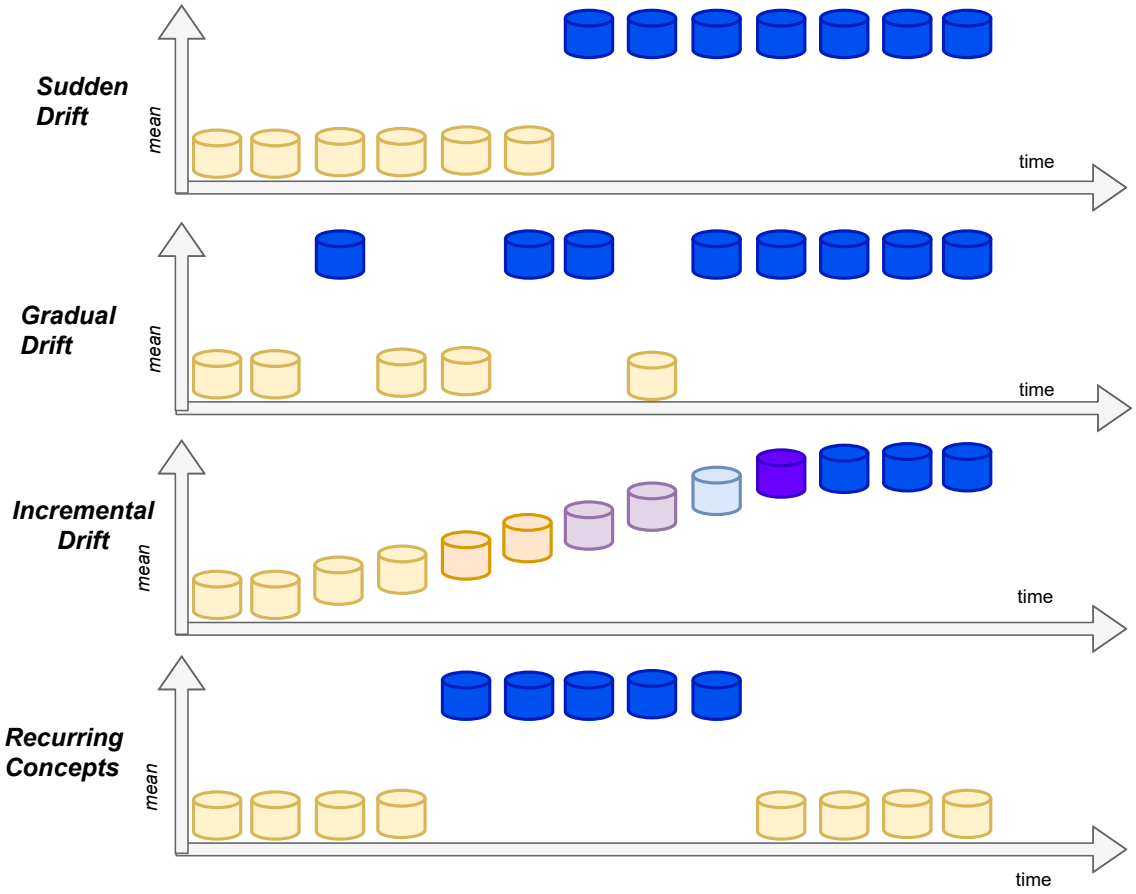

Fig. 2.2. Ilustration of four types of Concept Drift [83].

drift detection method (DDM) [20] is applicable in the context of predictive models. The method monitors the number of errors produced by a model learned on the previous stream items. Generally, the model's error should decrease or remain stable as more data is used, assuming that the learning method controls overfitting and that the data and label distribution are stationary. Instead, DDM observes that the prediction error increases; it takes this as evidence that change has occurred. The **geometrical moving average (GMA)** [15] presents the newest observations, greatest weight, and all previous observation weights decreasing in geometric progression from the most recent back to the earliest. The term "moving average"(without the adjective "geometric") implies an ordinary moving average in which k consecutive points are charged with equal weights of l/k. In the computation of a geometric moving average, the most recent point is assigned a weight of r, where $0 < r \leq 1$, and each point is assigned a fraction $(1 - r)$ of the weight of its immediate successor. Tests based on GMAs compare well with numerous run and moving average tests. In the statistical test of equal proportions **(STEPD)** [15], the accuracy of a chunk of recent samples is evaluated and correlated with the overall accuracy from the birth of the stream while employing a chi-square test to check for

deviations. The main idea of STEPD is to oversee the accuracy of a base learner over two windows: a recent window containing the latest examples and an older window with all the other examples observed by the base learner after the last detected drift.

At the level of algorithms, we describe only a few algorithms used in this thesis: Hoeffding Adaptive Tree, SamKNN, SGD Multiclass, and Naive Bayes Online. **The Hoeffding Adaptive Tree (HATT)** [20] is an adaptive extension to the Hoeffding Tree that uses ADWIN [19] as a change detector and error estimator. It has theoretical performance guarantees and requires no parameters related to change control. **SamKnn** [20] (The Self Adjusting Memory (SAM) model for the k Nearest Neighbor (kNN)) constitutes a proven classifier-based KNN within the streaming setting. SAMkNN can deal with heterogeneous concept drift, i.e., different drift types and rates, using biologically inspired memory models and coordination. **SGD Multiclass** [20] implements stochastic gradient descent for learning various linear models: binary class SVM, binary class logistic regression, and linear regression for multiples class. **Naive Bayes Online** [20] ins an incremental algorithm. It assumes independence of the attributes, and that might not be the case in many real data streams. It is well suited for the data stream setting. This classification algorithm is known for its low computational cost and simplicity. We chose these algorithms because each covers different learning methods and most algorithms leads with concept change (except Naive Bayes Online).

2.3 Machine Learning Algorithms Used in Sequential Analysis

Probabilistic Models (PM) have gained in sequential modeling. Typically, a sequence consists of smaller substructures with different functions, and different functional regions often display distinct statistical properties [141]. For example, a activity can consist of multiple domains, e.g, a fall covers some activities (Standing, standing-to-lying, lying). it would be interesting to predict the constituting domains (corresponding to one or more states in an probabilistic model) and their locations in the sequence (observations). This representation can be adapted in the same scenario in Human Activity Recognition, where we have activities (states) and a sequence of the activities (observations).

Overall, this problem needs a representation: We denote the observed symbol sequence as $x = x_1, x_2, \cdots, x_n$ and the underlying state sequence as $y = y_1, y_2, \cdots y_n$, where y_i is the underlying state of the $i - th$ observation x_i. Each symbol x_i takes on a finite number of

possible values from the set of observations $O = \{O_1, O_2, \cdots, O_N\}$ and each state y_i takes one of the values from the set of states S = $1, 2, ..., M$, where N and M denote the number of distinct observations and the number of distinct states in the model, respectively.

Two of the most algorithms used in this area are: *Hidden Markov Models* (HMM) and *K-Mer Filtered Classifier* [90]. In HMM, we assume that the hidden state sequence is a time-homogeneous first-order Markov chain. This implies that the probability of entering state j in the next time point depends only on the current state i, and that this probability does not change over time.

2.3.1 Hidden Markov Models

Hidden Markov Models (HMM) is a statistical model that can describe the evolution of observable events that depend on internal factors that are not directly observable. We call the observed event a 'symbol' and the invisible factor underlying the observation a 'state.' An HMM consists of two stochastic processes: an invisible process of hidden states and a visible process of observable symbols. The hidden states form a Markov chain, and the probability distribution of the observed symbol depends on the underlying state.

Firstly, let us define Hidden Markov Models (HMM) [1, 79]. We denote the observed symbol sequence as $X = \{x_1, x_2, \cdots, x_n\}$ and the underlying state sequence as $Y = \{y_1, y_2, \cdots, y_n\}$ where Y_i is the underlying state of the $i - th$ observation X_i. Each symbol x_i takes on a finite number of possible values from the set of observations $O = \{O_1, O_2, \cdots, O_N\}$, and each state Y_i takes one of the values from the set of states S= $\{1, 2, \cdots, M\}$, where N and M denote the number of distinct observations and the number of distinct states in the model, respectively. We assume that the hidden state sequence is a time homogeneous first-order Markov chain. This implies that the probability of entering state j in the next time point depends only on the current state i, and that this probability does not change over time. Therefore, we have

$$\begin{aligned} P\{y_{i+1} = a \mid y_i = b, y_{i-1} = i_{i-1}, \cdots, y_1 = b_1\} = \\ P\{y_{i+1} = a \mid y_i = 1\} = t(a, b) \end{aligned} \tag{2.1}$$

for all states a, b $\in$ S and for all i $\geq$ 1.

The fixed probability for making a transition from state a to state b is called the transition probability, and we denote it by t(a, b). For the initial state y_1, we denote the

initial state probability as $\pi(a) = P\{y_1 = a\} \forall a \in S$. The probability that the $i-th$ observation will be $x_i = x$ depends only on the underlying state y_i, hence

$$P\{x_i = x \mid y_i = b, y_{i-1}, x_{i-1}, \cdots\} = P\{x_i = x \mid y_i = b\} = e(x|b) \tag{2.2}$$

for all possible observations $i \in O$, all state $b \in S$, and all $i \geq 1$. the emission probability ($e(x \mid b)$.) of x at state b is represented by all possible observations x ∈ O, all state b ∈ S, and all i ≥ 1. The three probability measures $t(b, a)$, $\pi(b)$, and $e(x \mid b)$ completely specify an HMM. For pattern, we used the set of these parameters as Θ. Based on these parameters, we can now compute the probability that the HMM will generate the observation sequence x = $\{x_1 x_2 \cdots x_n\}$ with the underlying state sequence $y = y_1 y_2 \cdots y_n$. This joint probability $P\{x, y \mid \Theta\}$ can be computed by:

$$P\{x, y \mid \Theta\} = P\{x \mid y, \Theta\} P\{y \mid \Theta\} \tag{2.3}$$

where

$$P\{x \mid y, \Theta\} = e(x_1 \mid y_1) e(x_2 \mid y_2) e(x_3 \mid y_3) \cdots e(x_n \mid y_n) \tag{2.4}$$

$$P\{y \mid \Theta\} = \pi(y_1) t(y_1, y_2) t(y_2, y_3) \cdots t(y_{n-1}), y_n) \tag{2.5}$$

As we can see, computing the observation probability is straightforward when we know the underlying state sequence.

Figure 2.3 show a example of HMM model. The main line of the HMM contains a sequence of M states, which we call *Match States* , corresponding to positions that correspond to a complex activity (HAR) or a protein (biological computing). In Figure, $M = 6$. Each of these states can generate a sequence x. π means that each match states m_k, where $1 \leq m \leq 6$, have distinct distributions. Each m_k has a delete state d_k that does not produce any activity but is a state used to skip m_k. In Figure 2.3 we have match, delete and insert states described in squares, circles, and diamonds, respectively. From each state, there are three possibilities for transitions to other states. Transitions into a match or delete states always move forward in the model, whereas transitions into insert states do not. We can serve multiples insertions between match states can exist since the self-loop on the insert state allows a transition from the insert state to itself. The transition probability from state q to state r is, as mentioned before, $\pi(r|q)$.

In order to use HMMs in practical applications, we need to compute the observation

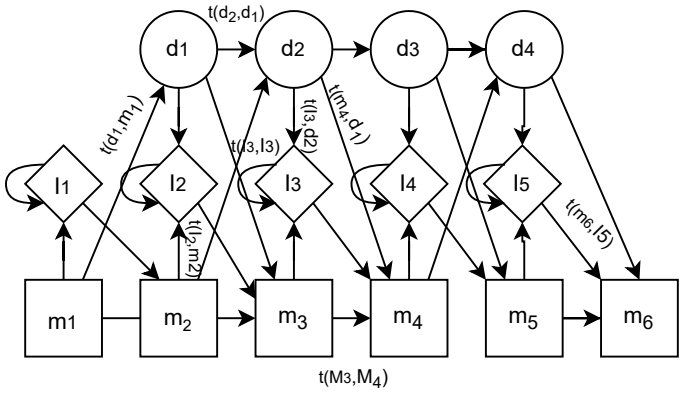

Fig. 2.3. A Illustration of a basic Hidden Markov Model.

probability $P\{x|\Theta\}$ based on a given HMM. This problem is sometimes called the scoring problem since computing the probability $P\{x|\Theta\}$ is a natural way of 'scoring' a new observation sequence $x = x_1, x_2, \cdots x_n$. One way to compute the observation probability is to consider all possible state sequences y for the given x and sum up the probabilities. However, this is computationally very expensive since there are n possible state sequences. Due to this, there is a dynamic programming algorithm, called the ***forward algorithm*** [79, 141], that can compute $P\{x|\Theta\}$ in an efficient manner.

Another important problem is to find the local optimal state sequence, in the HMM that maximizes the observation probability of the given symbol sequence x. Among all possible state sequences y, we want to find the state sequence that best explains the observed symbol sequence. Finding the optimal state sequence y by comparing all n possible state sequences is computationally infeasible. However, we can use another dynamic programming algorithm, well-known as the ***Viterbi algorithm*** [79, 141], to find the optimal path y efficiently. The Viterbi algorithm finds the local optimal sequence that can maximize the observation probability of the entire symbol sequence. The advantage of predicting the optimal states individually is that this approach will maximize the expected number of correctly predicted states. For this reason, the Viterbi algorithm is often preferred when we are interested in inferring the optimal state sequence for the entire observation x.

Both algorithms for scoring and optimal path are concerned with analyzing a new observation sequence x based on the given HMM. However, the solutions to these problems are meaningful only if the HMM can properly represent the sequences of our interest. Let us assume that we have a set of related observation sequences $x = \{x_1, x_2, \cdots, x_n\}$ that we want to represent by an HMM. For example, they may be different sequential of the same atomic activities (Chapter 7 describes a scenario) that belong to the same functional

complex activity; in addition, the most probable path itself can be found using the usual backtracking technique [79]. This is the method we use to obtain our multiple alignments: each sequence is aligned to the model by the Viterbi Algorithm, after which the mutual alignment of the sequences among themselves is then determined.

2.3.2 K-mer Filtering Classifier

In this section, we will describe one of the commonly used algorithms for the solution of many problems, particularly for bioinformatics, mainly by using k-mers as the main element in detecting sequential alignments.

First of all, we need to define *Feature Vector Mapping* and *K-mers*. Feature Vector Mapping is a mapping of classified objects to the feature space. This mapping is called a feature vector representation of subject area. Such feature vectors capture the compositional properties of the sequences. Feature mapping rule can be described as a function $M : \hat{S} \rightarrow F$, where $\hat{S} = (s_1, s_2, \cdots, s_N), s_i \in \{\Sigma\}^k$ is a sequence (for instance sequence of activities), Σ is the alphabet (for instance $\Sigma = \{A, C, D, T\}$) N is the length of a sequence, and $F = (f_1, f_2, \cdots, f_M), f_j \in \Re$ is a feature space, M is the length of feature vector.

We need define *K-mer*. Damaševicius [32] define K-mer as a K-base long sequence (k-tuple). Usually, this representation is for sequence (e.g DNA sequence in bioinformatics).

$$(a_1, a_2, \cdots, a_k), a_i \in \hat{S}, i = 1, 2, \cdots, k \tag{2.6}$$

K-mer filtering is a filter to convert a string attribute into numeric features represented as the frequency of its k-mer substrings.

where, $\hat{S}$ is K-mer feature vector (KMF). KMF is constructed constructed using a frequency (or probability) of each k-mer in a sequence.

$$\langle \hat{S} \rightarrow (\ p_j = \frac{n_j}{N - k + 1}) \ \rangle \hat{S} \in \Sigma^N, j = 1, \cdots, Z^k \tag{2.7}$$

where Σ is the alphabet and Z is number of letters in the alphabet. In probabilistic domain, alphabet is a set of states. N is the lenght of a sequence, k is a lenght of k-mer. Manekar and Sathe [88] detail according to best value of k. n_j is the number of j-th k-mer in a sequence.

For example, if you were to take the sequence TCGGTCA and split it into overlapping

4-mers you would obtain:

```
Sequence: TCGGTCA
k-mers:
    TCGG
     CGGT
      GGTC
       GTCA
```

K-mers are used to detect overlaps. If two sequences share a k-mer then they must share $\leq$ k bases and may overlap. For example (shared k-mer in red):

```
Read 1 : GGCATTG Read 2 : CCATTGC
k-mers : GGCA             CCAT
          GCAT             CATT
             CATT             ATTG
              ATTG             TTGC
```

It can be seen that both read 1 and read 2 share the 4-mer CATT, hence there is an overlap between them. K-mers can be used in taxonomic classification to detect overlaps between sequence reads and database entries.

The training set S is the correspondence between the k-mer frequencies of training sequences and their groups. The feature vector $Fv(s)$ for an input sequence s is constructed from the number of occurrences of all k possible k-mers (given Σ), divided by the total length of N. Next, we process the feature vectors for more efficient use by classifiers (**base classifier**). Due to computational limitations, the k for k-mers should be a small number, because the number of k-mers grows exponentially [57], in other words, the total number of different k-mers is $count(\Sigma)^k$, where count function is the number of distinct observation symbols.

Figure 2.4 shows an example of a workflow based on previous sequence. The first step is to input raw sequences for feature extraction and feature encoding using Kmer-processing and frequency-based techniques (Tokenization), respectively. Then the created sequence ingest to ML model for further analysis.

A major difference between HMM Classifier and traditional classifiers such as Naïve Bayes (NB), SVM and KNN is that HMM can be applied directly to sequence data while

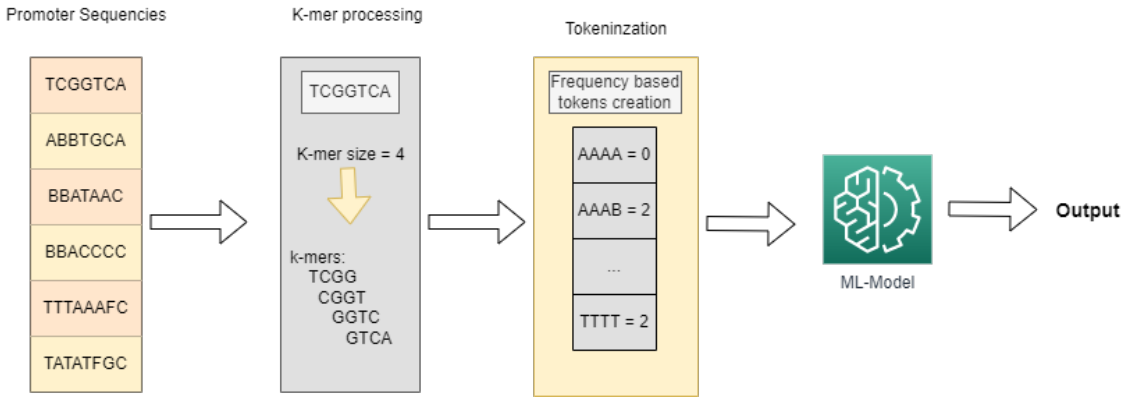

Fig. 2.4. Steps for classification of Kmer Filtered Classifier, adapted of [17].

traditional classifiers expect the data to be in the form of feature vectors extracted from the original sequence data. Here, K-mers filtering show how to simultaneously, extract features from sequence data and build/test a model. Thus, K-mer filter allows us to specify a Base Classifier (AC) and a filter to be applied on the fly before feeding the data to the predictor. Class for running an AC on data that has been passed through an K-mer filtering. Like the classifier, the structure of the filter is based exclusively on the training data and test instances will be processed by the filter without changing their structure. Thus, It is important to determine which AC would be the most effective for classifying sequences, using their respective k-mer frequencies as feature vectors (numerical representations).

2.4 Final Remarks

In this chapter, we described some concepts of supervised machine learning. Two learning paradigms were presented within this scenario: supervised batch machine learning and supervised online machine learning, specifically for classification problems. Unlike batch supervised machine learning, online supervised learning algorithms can handle continuous data streams that evolve over time and with changes of concept about the data, allowing to extract new knowledge without processing a huge volume of information.

In addition, this chapter presented some state-of-the-art algorithms that deal with sequential data analysis. These algorithms are important, mainly for one of the proposals of this work, which is described in Chapter 7.

Chapter 3

Human Activity Recognition

HAR intends to observe human-related actions in order to obtain an understanding of what kind of activities (or routines) individuals perform within a time span. As example, a system could detect when human is standing, lying down, falling, cooking, driving, among many other activities. HAR is divided into two areas [128]: *Atomic Human Activity Recognition* (AHAR) and *Complex Human Activity Recognition* (CHAR). Atomic activity recognition represents the repetition of a simple daily activity such as walking, jogging, sitting, cycling and so on. Complex activity recognition, involves recognizing activities with complex and repetitive patterns, represented with different sizes. For [128], Complex activity recognition can be achieved by recognizing simple activities. Examples of complex activities are cooking, washing the plants, washing the room, driving a car and so on. These two concepts are extremely important for conducting this research.

In general, data can be collected in two ways: Multi-user and mono-user. The system reaches user data around many different users. Mono-user reaches uses data based on a specific user. Data used for HAR include human postures (e.g sitting, standing, lying down) and movements (running, walking, jumping). Through these kind of data, HAR system can infer the executed activity. The most useful scenario for this case is the *Ambient Assisted Living (AAL)*, where the system analyzes the people behavior in an assisted environment, where, for example, a caregiver analyzes the patient's behavior. By increasing perspective, HAR systems may not be limited to studies with one user, but with multiple people interacting in an assist living. This problem is known as *Ambient Daily Living* (ADL) [142]. ADL is a way to describe a person's functional status and his/her interactions with others.

In recent years, there has been an explosion in the number of electronic sensors or

mobile devices with numerous features, such as accelerometer, proximity sensor, magnetometer, GPS, and so on. Wireless communication as portable and implantable bio sensors (as body area networks (BAN) or body sensor networks (BSN) [29]) are favoring the monitoring of human physiological activities and actions, such as exit status and movement patterns. The main advantages for these sensors are: *flexibility*, due to monitoring results can be sent to nearby devices such as smartphones, smartwatches and etc, depending on the application domain; *effectiveness and efficiency*, due to signals produced by body sensors can be processed effectively to obtain reliable and accurate physiological estimates; and *cost-benefit*, due to more sensors being produced in large scale at relatively low cost. Also, video and depth sensors became an important tool in this research area [126]. Frequently, investing in image identification problems by companies has developed efficient indexing, and storage schemes to improve the user experience. This requires learning methods from raw videos and video synthesizing based on its content. Some of the main commercially viable application scenarios are *Medical Informatics* [114], *Detection of Terrorist Attacks* [13] and *Sport Assistance* [130].

3.1 Atomic Human Activity Recognition

For an efficient inference process of sensor data activities, considering a high number of possible users, tasks and sensors, it is interesting to develop AHAR project with well-defined steps from data collection even final activity inference.

Figure 3.1, shows the steps of inference problem in AHAR in context middleware projects [142]. Firstly, sensor data is collected by a sliding window, with a specific time interval (size window) and an overlapping size value (when applied). It is important to carefully design the window size because it may provide more appropriate information for context classification. A shorter window may not acquire the context properly. However, wider windows would create latency in detection and adds a high computational cost. Preprocessing and feature extraction for HAR offer necessary modifications to correct missing and limitations in the data and extract information inside the data, respectively. After this step, we have diverse characteristics of feature vectors that enable us having training data for classification and regression algorithms to represent different context inferences. Classifiers' output sometimes cannot solve the consistent discrimination of the activity in a time sequence of adjacent context inferences. In this case, a basic smoothing technique takes a majority voting scheme with a sliding window of a specific history length of context inferences. Therefore, any inconsistency (i.e, false positives) can be

eliminated [142].

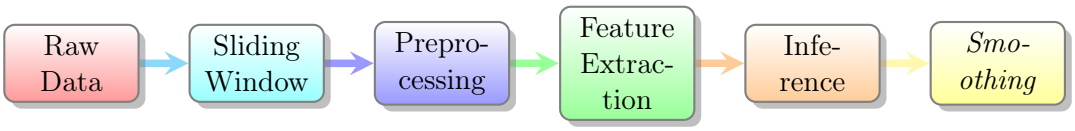

Fig. 3.1. Steps of AHAR inference process.

3.1.1 Sliding Window

The sliding window is a data collection strategy relevant to the description of a context in a time window. A metric to define the size of the window is reasonably necessary. It favors the generation of more relevant information for the classification of the context. A smaller window may not take advantage of information necessary to represent a context. However, a larger window may cause latency issues in context detection producing a higher computational cost. The value for overlapping is a critical parameter to detect any variation in context.

The literature presents us with three approaches to the sliding window. These approaches are discussed when data are collected in intelligent environments, whereby they can be adapted to HAR.

The first approach is **Explicit Segmentation** [78, 70]. This approach is divided into two steps. The data stream is segmented into blocks in the first step—each block corresponding to an activity. The classification of each of the blocks is performed in the second step. Afterward, the approach has to wait for the future data to make a decision about the past data, making it a somewhat non-streaming approach. Furthermore, due to their reliance on future data, significant temporal gaps in successive sensor activities that are real in daily routines will result in waiting for a long time to make a decision about past events.

The second approach is a **Time-based window** [78, 132] representing the division of the entire sequence of sensor events into equal-sized time intervals. This technique provides a more straightforward approach to learning activity models during the training phase on the Explicit Segmentation approach. It further reduces the computational complexity of the Explicit Segmentation process. A time-based window is a good approach when dealing with data streams obtained from sensors that operate continuously over time. Data for each time interval is always guaranteed. These systems commonly use accelerometers and gyroscopes data where is sampled at a constant rate.

The third approach is the **window based on sensor events**. This approach splits the sequence into windows containing an equal number of sensor events. During this process, this window may vary in duration. The advantage is during the performance of activities, where several sensors can be triggered. In contrast, during periods of silence, there will not be many sensor triggers. Sensor events that precede the last event in a window define the context of the last event. This method also has some inherent disadvantages. For example, The last sensor event matches the start sensor event in the next activity. There is a significant amount of time between this event and the previous sensor event. The relevance of all sensor events in this fragment to the last event may be minor if the interval is large. Treating all sensor events with equal importance is not a good strategy. This approach offers computational advantages over the Explicit Segmentation process. It does not require future sensor events to classify past sensor events.

3.1.2 Exploratory Analysis, Inference and Smoothing

Exploratory Analysis phase in AHAR explores the data processing, data fusion process and feature extraction. In general, it is challenging to analyze, fit a classification model, and estimate new activities from raw data. The data can consist of a large number of attributes, irrelevant information, and additive noise distortion. This phase extracts and investigates the spatial characteristics of sensor data in each window and assists in identifying different activities classes, allowing the prior separation of relevant information for HAR algorithms.

Features space are constructed using several signal processing primitives: with attributes based on time-space metrics such as mean, standard deviation, and attribute correlation based on frequency spectrum: entropy, fast Fourier transform coefficients (FFT), wavelet transforms. Table 3.1 summarizes the elements of a feature vector in the time and frequency domains.

Inference phase defines the training and estimation through machine learning algorithms in batch and stream learning (Chapter 2). Classifiers output sometimes cannot solve the consistent discrimination of the activity in a time sequence of adjacent context inferences. In this case, a basic smoothing technique takes a majority voting scheme with a sliding window of a specific history length of context inferences. Therefore, any inconsistency (i.e, false positives) can be eliminated [142].

Table 3.1. Elements of Feature space. Adapted table by Yurur [142].

Feature space	features
Time domain	mean, standard deviation, variance, magnitude, derivative, min-max, amplitude, histogram, interquartile range (IQR), mean absolute deviation, correlation between axes, peak count, zero crossing rate.
Frequency domain	Fourier transform (FT), discrete cosine transform (DCT), entropy, centroid, maximum frequency, energy FFT , FFT mean and FFT standard deviation.
Others	Autoregressive Coefficients wavelet transforms

3.2 Complex Human Activity Recognition

Complex Human Activity Recognition (CHAR) are not as repetitive as simple activities and may involve various hand gestures; for example, eating, drinking coffee, smoking and giving a talk. We also place using stairs in this category, because it is not easy to recognize such activities with a single accelerometer. Usually, additional sensors, such as a gyroscope [10] and a barometer [11], are gathered with the accelerometer to reliably recognize using stairs: walking upstairs and downstairs. Atomic activies are present as basis for CHAR. The thesis itemizes details of the types of CHAR.

- Sequential: Sequentially Complex Human Activity (SCHAR) is a a collection of simple activities occurring sequentially. For example, a user is stand and suddenly falls. The fall process evolves three atomic activities: Standing, transaction activity (stand-to-lye) and lying. Figure 3.2 illustrates three examples as falling (standing), falling (sitting) and upstairs fainting.

- Concurrent: Concurrent activity is a collection of atomic activities occurring simultaneously. For example, a user can clean the dishes while listening a music. Figure 3.3 illustrate a example. In this scenario, the recognition of the ”Brush the teeth” activity is displayed. This activity is composed of sub-activities such as standing, hand-to-face, and movement of the teeth. These activities take place simultaneously.

- Interleaved: Interleaved activity is a collection of atomic activities switching between

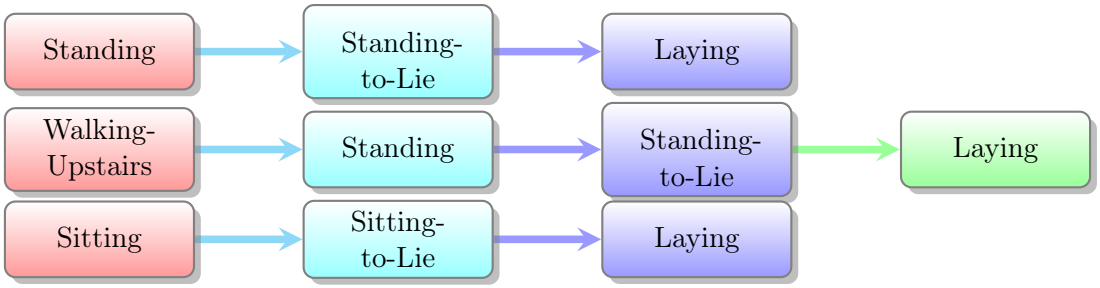

Fig. 3.2. Three examples of Complex Human Activity Recognition in Sequential Method.

their atomic activities. Figure 3.4 illustrates a example. While talking a phone call, a user may go to the kitchen to take off the cook and come back to resume the call.

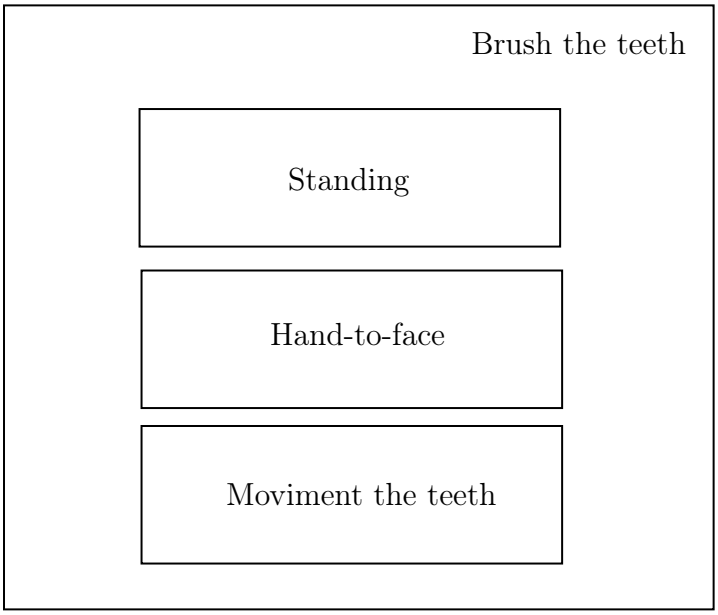

Fig. 3.3. Example of Complex Human Activity Recognition in Concurrent Method.

The thesis just covers the sequential complex method. This method is present in our daily activities, as resting, eating, drinking, using utensil, falling; identification of people with repetitive activities [47]; identification of musical movements where the player can present sequential of different musical notes [105]. Sequential approaches are the single-layered approaches that recognize human activities by analyzing sequences of features. The activity is composed as a sequence of observations (i.e., feature vectors), and induce that an activity has occurred if the system observes a particular sequence characterizing the activity. Sequential approaches first convert a sequence of atomic activities into a sequence of feature vectors by extracting features (e.g., degrees of joint angles) that describe the status of a person per timestamp. Once feature vectors have been extracted, sequential approaches analyze the sequence to measure how likely it is that the feature

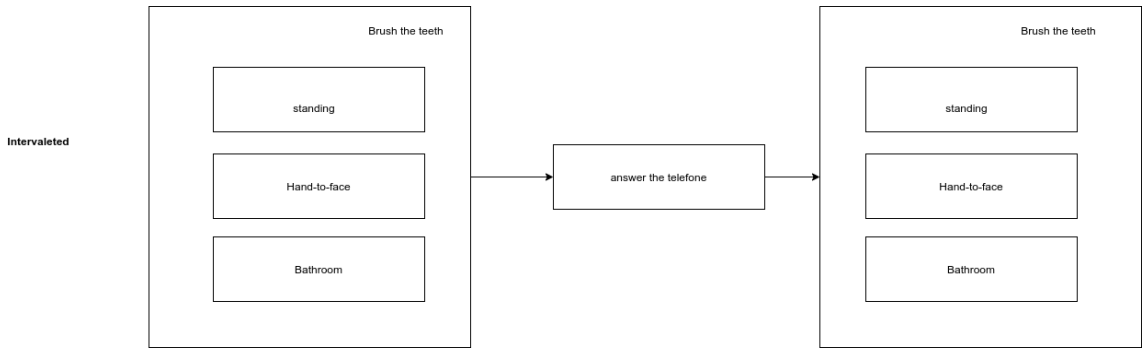

Fig. 3.4. Example of Complex Human Activity Recognition in interleaved method.

vectors were produced by single user performing the activity. If the likelihood between the sequence and the activity class (or the posterior probability of the sequence belonging to the activity class) is high enough, the system decides that the activity has occurred. Figure 3.5 shows a example of SCHAR.

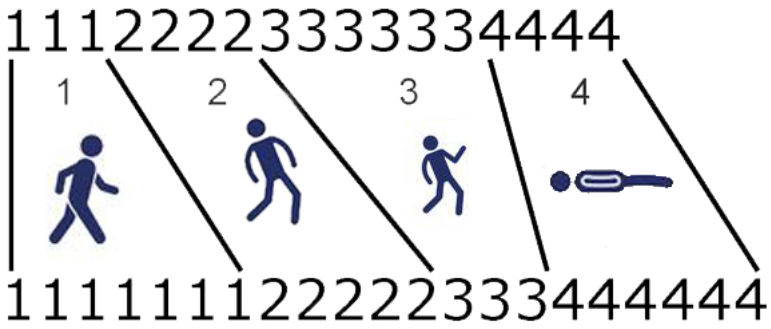

Fig. 3.5. An example matching between two "falling" sequences with different nonlinear execution rates. Each number represents a particular status (i.e., pose) of the person. Figure based on Aggarwal et al. [7].

The most approaches are State Model-Based and Exemplar-Based [7]. State model-based approaches are the sequential approaches that represent a CHAR as a model composed of a set of states. The model is statistically trained so that it corresponds to sequences of feature vectors belonging to its activity class. More specifically, the statistical model is designed to generate a sequence with a certain probability. Generally, one statistical model is constructed for each activity. For each model, the probability of the model generating an observed sequence of feature vectors is calculated to measure the likelihood between the action model and the input sequence. An atomic activity is assumed to be in one state at each time frame. Each state generates an observation (i.e., a feature vector)—the system transitions to another state in the next timestamp, considering the transition probability between states. Once transition and observation probabilities are trained for the models, activities are commonly recognized by solving the evaluation

problem. The evaluation problem is the problem of calculating the probability of a given sequence generated by a particular state model. If the calculated probability is high enough, the state model-based approaches can decide that the activity corresponding to the model occurred in the given input.

Exemplar-based approaches represent human activities by maintaining a template sequence or a set of sample sequences of action executions. Given a new input, exemplar-based approaches compare the sequence of feature vectors extracted from the input with the template sequence (or sample sequences). If their similarity is high enough, the system is able to deduce that the given input contains an execution of the CHAR. The purpose of this function is to include information about the sequence structure, i.e., to find overlapping sequences of primitives that occur together. As mentioned in previous chapter, k-mer is a string of length k associated to the number of times a k-mer Pattern appears as a substring of a sequence.

Finally, this information is closely associated with the concepts of HMM and K-mers mentioned in the previous chapter. Thus, we can implement SCHAR models from HMM and K-mer filtering algorithms.

3.3 Context-Aware System for Human Activity Recognition

Firstly, to discuss context-aware systems properly, it is necessary to define "Context". In the literature, the main definition of context is presented in Abowd [5]: "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.". There is a range of alternative definitions in the literature reflecting minor variations of this definition [102]. Specifically, Dey et al.[36] extended that definition to explain how context is used in a computational system or project: "A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.". This definition is commonly used in computing area, as it helps to identify that context-aware can be seen as a functionality in a computer system.

Context-awareness can be applied in two levels: *low-level* (hardware) and *high-level* (software). In the *low-level*, context recognition facilitates tasks such as efficient routing, modeling, reasoning, storage, and event detection, mainly considering energy consumption

and availability [102]. At this level, resources such as knowledge and data for decision-making are limited. So, complex data processing is not allowed in hardware level. A positive aspect is that simple features can be stored, such as energy consumption costs and data transport. In the *high-level*, knowledge is generated by extracting from data and reasoning. In other words, softwares in the *high-level* can be capable of modify raw data into knowledge. Systems that apply context-awareness in the *high-level* have more resources and flexibility to use, ensuring computational power for performing complex reasoning and the broad access and use of data and knowledge.

Context Aware Computing (CAC) should provide functionalities for acquiring context information from several sources (e.g. physical and virtual sensors). Moreover, raw-sensor data (*low-level* context) needs to be retrieved and automatically transformed into context data, structured according to a specific data model. This facilitates the inference of complex context data, which may be used to trigger different context-aware behaviors. This process may be divided into stages, composing a context life cycle [102]. The context life cycle represents how sensor data is collected, modeled and, processed and how complex knowledge can be extracted from sensor data to be later disseminated. The context life cycle is comprised by 4 (four) stages: Context Acquisition; Context Modelling; Context Reasoning; and Context Dissemination, as illustrated in Figure 3.6 . **Context acquisition stage**: data is collected from several real or virtual sensors; **Context modeling stage**: the collected data is stored according to a specific context model or representation. Thus, this stage is also responsible for defining context properties, relationships and details. **Context reasoning stage**: the reasoning operations occurs and they can be more efficiently performed if the previous stage is properly executed. **Context dissemination stage**: context is delivered to consumer applications [102].

Our focus on this study is the context reasoning stage. Usually, it involves (i) a deduction method (however ML is based on induction), which uses a knowledge based related to an *available context*; or (ii) a process defined by a higher-level context inference [22, 102]. The importance of reasoning arises from two inherent characteristics of the *low-level* data: *uncertainty* and *imperfection* in context reasoning, regarding to unknown, inaccurate or erroneous data source. The reasoning performance can be measured using efficiency, integrity and interoperability. This cycle is divided into 3 (three) steps: (a) context processing, (b) sensor data fusion and (c)context inference [102, 118]. During (a) context processing step, the data undergoes through a cleaning and enrichment process: relevant features are selected; inconsistent and redundant data are eliminated; missing values are corrected; outliers are removed; and context is validated via multiple

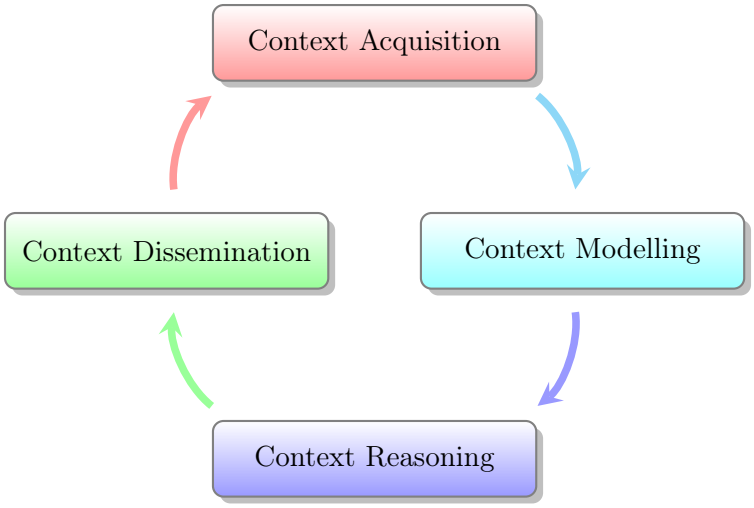

Fig. 3.6. The four stages of the context life cycle [102].

sources. In (b) sensor data fusion step, there is a scenario with multiple sensors, combined to ensure more accurate, reliable and complete data, which can not be generated from a single sensor. In (c) context inference step, the system generates *high-level* context information, based on lower level context.

There are many different context reasoning approaches that can be divided into four categories [102]:

1. *Context reasoning algorithms based on fuzzy logic* are very different from traditional logic. In traditional logic theory, acceptable truth values are 0 or 1. However, fuzzy logic allows a partial truth where confidence values represent degrees of pertinence membership of a value into a categories of a semantic variable. In ubiquitous computing scenarios, real world representations with fuzzy logic are more comfortable than traditional logic, especially when variables that need semantic interpretation are not easy. For instance, or very light, heavy and very heavy but the true threshold weight values for defining who is heavy or very light is not an easy task. Usually, fuzzy logic is used for this kind of knowledge representation, joined to a mechanism for fuzzy logic inference.

2. *Context reasoning based on ontology* uses a strategy dependent on description logic (another family of logic formalism for knowledge representation). An advantage of this reasoning is using integrated ontologies [102]. On the other hand, a disadvantage is the difficult for finding missing values or ambiguous information. Due to this aspect, it is more common the use of rules-based reasoning.

3. *Context reasoning based on rules* uses rules formed as propositional. Rules can also

be used with context reasoning based on ontology [67]. Context reasoning based on rules shows advantage in situations where it is necessary generate *high-level* context information using *low-level* context. Moreover, it can be used for defining events.

4. *Context reasoning based on ML* finds out relations among context variables from direct context samples. The literature covers studies using supervised, unsupervised, and reinforcement learning approaches [102]. In supervised learning, given a set of labeled training data, a function is constructed using the training data to label (predict) new instances. If results (predictions) are discrete values, we have a classification problem; if results are continuous values, we have a regression problem. For unsupervised learning, training samples do not have correct labels associated to them. Unsupervised learning tasks are useful for detecting patterns relying on data. Usually, its main task is ring for constructing data partitions. Reinforcement learning aims to take suitable action to maximize reward in a special situation.

3.4 Data Stream Learning for Human Activity Recognition

This section describes characteristics of the HAR inference process in scenarios that inference methods use data stream learning techniques. Some terms in activity recognition have their corresponding meanings in data stream learning, albeit in different settings and with different situations [4].

One is the Learning from concept drift - (LFCT). It mainly refers to the change in data distribution as a flow evolves. This change happens as described in Section 2.2. The main problem in Human Activity Recognition associated with LFCT is the *personalization problem*. *Personalization* is defined as the process of tuning an overall template to represent a user's customized way of performing different activities. An example of a scenario is when the user starts the training for a marathon. In initial workouts, the runner takes shorter strides with moderate speed. During practice, he changes his way of running to a longer stride with a more intense speed. The learning model must learn that there has been a "refinement"in the user's running mode. The significant limitation on personalization issues is that maybe the way a user A runs, not necessarily a user B, will run that way. Perhaps the way A performs the activity may represent just a cooper. In contrast, for the user B, it may represent a walk.

An outlier is an observation that deviates from the other observations that it arouses

suspicion that a different mechanism generated it. The outliers in continuous data stream learning correspond to abrupt changes in the data streams of the underlying concepts. Outliers can be referred to as noise, anomalies, and abnormalities. The term Outlier Detection in HAR refers to a sudden activity, such as fall detection. Detecting sudden activity is similar to detecting outliers in that both aim to find unusual events in the data. The "sudden activity" issue matches the "fall detection" problem widely discussed in HAR, especially for elderly people.

In stream learning approaches, *Concept Evolution* is the monitoring data stream to discover new concepts. The concept evolution term is analogous to detecting new activities in activity recognition.

In contrast to the *Concept Evolution*, some concepts have become outdated and no longer relevant to the target domain. These concepts require an adaptive mechanism to forget about abandoned concepts. The term *Concept Forgetting* [53] is present and relevant to Human Activity Recognition while users no longer perform activities. *Concept Forgetting* aims to continually update the model to reflect the latest changes to the data and remove outdated/abandoned activities.

The problems covered in HAR throughout this section are summarized in Table 3.2. The table shows a subset of problem terminology in stream learning and their corresponding related terms in activity recognition.

Table 3.2. Terminology between data stream learning and Human Activity Recognition (Based on Abdallah [4]).

Stream Learning Problems	**Meaning**	**Human Activity Recognition**	**Example**
Learning from concept drift	Data Stream change detection and response	Personalization	Model Tuning to suit a personal way of performing activities
Concept Evolution	Discovery the new concept in Data Stream	Detection of new activities	Discovery new activities
Outlier Detection	Anomaly Detection in data stream	Suddenly Activity Detection	Detection of suddenly changes of a activity
Concept forgetting	Decremented learning outdated learning	Concepting Forgetting	Decremental learning forgetting activities

3.5 Final Remarks

This chapter describes the definition of Recognition of Human Activities and the concepts of AHAR and CHAR concepts. In AHAR, the chapter describes the workflow of the activity from the collection process to the inference process. In CHAR, the chapter shows Different types of complex human activity recognition and detection strategies based on computational biology concepts.

This chapter also presents concepts of HAR applied in context-aware systems. It presents the definition of context, the life cycle of a context, and how these concepts are in the Human Activity Recognition domain. Finally, the chapter presents scenarios in recognition of human activities that use methods based on data stream learning.

Chapter 4

Systematic Literature Review

To achieve our goals in Chapter 1, the thesis includes a Systematic Literature Review (SLR) following the methodology proposed by [76]. The study intended to identify all relevant primary and secondary studies related to context reasoning in context-aware middlewares (CAMs). Three steps was executed: planning, collect and analysis. During the planning step, the study defined our research protocol, composed by research questions, search strategy and study selection (inclusion and exclusion) criteria. This study established four Research Questions (RQs): **RQ1**: What are the different context reasoning categories present on CAMs projects?; **RQ2**: Which are the machine learning algorithms (ML) used for generating models for context reasoning in CAMs?; **RQ3**: Which CAMs have tools to support data processing on real-time?; **RQ4**: What are the Human Activity Recogntion (HAR) scenarios usually tackled by the research works?

The search strategy had to embrace three areas of interest: CAMs, HAR, and ML, specially considering the DSL approaches. In this way, the study defined the search string as: *(("middleware") AND ("context-aware"OR "ambient intelligence") AND ("Machine Learning"OR "Decision Models") AND ("Human Activity Recognition"OR "activity recognition") AND ("data stream"OR "online reasoning"OR "real-time reasoning"))*. The real interest in this research was limited by the terms (*"ML"OR "Decision Models"*) in the context of CAMs. *"Decision Models"* term was chosen by [102], which generalizes the concept of inference techniques for context reasoning. The collection of terms (*"data stream"OR "online reasoning"OR "real-time reasoning"*) were chosen for understanding the state-of-the-art of using *data stream* in data stream learning domain, due to the potential of dealing with huge volume of data collection over time and possibility of presenting concept drift concept (see Section 2.2). The study selected the most commonly used di-

gital libraries to conduct our research, namely: *Ei compendex* [1], *ISI web of knowledge* [2], *Science Direct* [3], *Scopus* [4] and *springer Link* [5]. The study selected publications between 2013 and 2018.

Selection criteria was divided into inclusion and exclusion criteria. Only one inclusion criteria was (I1:) Publications that use of ML in CAM, specially in HAR Applications. Seven exclusion criteria were (E1:) Publications must have inference or reasoning methods on CAM; (E2:) Publications must have middlewares; (E3:) Publications must have HAR terms; or describe middlewares and ML interaction issues; (E4:) Publication must be full paper. Short papers, books, letters, notes, and patents are excluded; (E5:) Publications must be in English; (E6:) Publications must be unique. If a publication is repeated, other copies of that publication are excluded; and (E7:) Publications must be between 2013-2018. (There were scenarios where the publication database had returned results out-of-date.)

Figure 4.1 shows the PRISMA Flow Diagram [94]. The flow diagram describes the information flow through the SLR phases, mapping the number of identified, included and excluded papers. The first phase, consisted of applying our search string in each publication database, returned 234 publications. Second phase involves duplicity identification and screening the articles for applying inclusion and exclusion criteria. Firstly, the study removed eight duplicated publications; after the study selected 43 accepted publications for analysis. The main inclusion criteria was analyzing publications approaching ML as context reasoning in CAMs. The study decided to not include only HAR applications to understand if any CAM could be used for this kind of problem. Also, The study considered publications explaining different inference algorithms for context reasoning in CAMs on HAR scenarios.

4.1 Answering Our SLR Research Questions

4.1.1 Context Reasoning Categories in CAMs projects

Table 4.1 shows the different context reasoning approaches present in CAMs presented in our selected studies. The table was divided into five lines, each one corresponding to a

[1] https://www.engineeringvillage.com/home.url
[2] https://www.webofknowledge.com/
[3] https://www.sciencedirect.com/
[4] https://www.scopus.com
[5] https://link.springer.com/

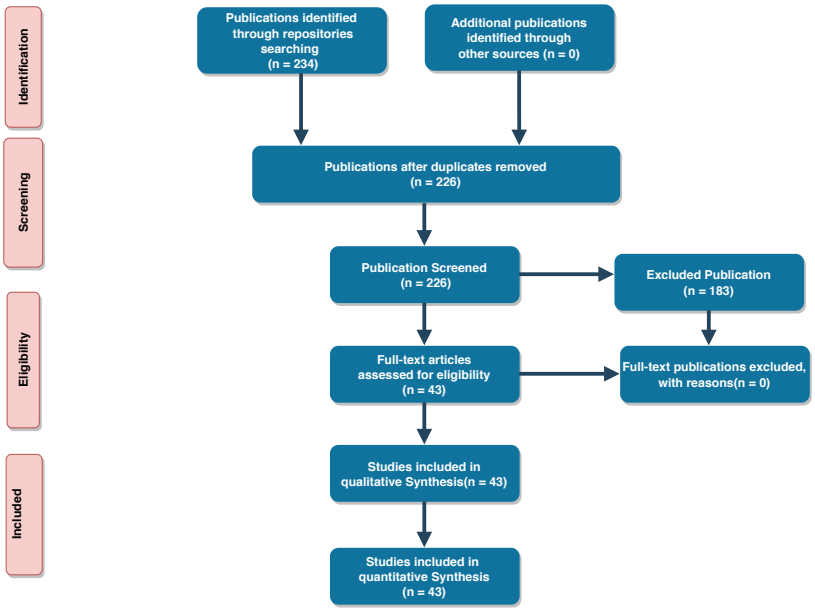

Fig. 4.1. Information flow through the steps of the review using PRISMA Flow Diagram (Adapted from Moher et al. [94]).

category of Context Reasoning, described in Section 3.3. The main observation is that all selected publications focus on the HAR problem, except for those containing machine learning as context reasoning technique. Some publications are present in more than one category of Context Reasoning. Most of them are secondary papers (surveys or literature reviews). These publications are marked in bold in our table.

4.1.2 ML Algorithms Used in CAMs

For answering the second research question (RQ2), we gathered studies that only uses machine learning algorithms for some context inference techniques in CAMs, *i.e.*, the 28 papers presented in the first line in Table 4.1. For this, the study considered not only papers that tackle HAR but also the ones that tackle other kinds of domains. Table 4.2 shows the learning algorithms in columns and the analyzed studies in lines. A cell marked with $\surd$ indicates that the study of the respective line uses the learning algorithm of the respective column. Supervised learning algorithms for regression and classification tasks were gathered: optimization-based (SVM and Artificial Neural Networks — ANN); distance-based (KNN); rules-based (Decision Tree — DT — and Random Forest — RF); probabilistic-based (Naive Bayes — NB — , Bayesian Networks — BN — Hidden Markov Models — HMM and, — probabilistic models adapted for Data Stream — PDS ; Fuzzy Logic-based; and Regression Models — RM. The results show the existence of a nearly uniform distribution in the number of papers using many classification techniques, except

Table 4.1. Answering RQ1: Studies presenting different CAMs with different context reasoning, grouped in diferent categories.

Cat. Context Reasoning	# Studies	Publications
Machine Learning	28	[40] [86] [13] [37] [95] [122] [52] [68] [142] [110] **[102]** **[118]** [134] [143] [8] [99] [117] [108] [98] [6] [27] [130] [12] [12] [106] [103] [23] [104] [87]
Fuzzy Logic	5	[58] **[102]** **[118]** [107] **[135]**
Rules	8	[114] [50] **[102]** **[118]** [123] [137] [136] [96]
Ontology based	7	[65] **[102]** **[118]** [66] [10] [139] [72]
Proabilistic Logic	4	[48] **[102]** **[118]** **[135]**

for Bayesian Networks, Fuzzy Logic, and probabilistic data stream models. We observe that *PDS* (Probabilistic Model for Data Stream) was only used by Rasanen et al. [108].

4.1.3 Processing Real Data in CAMs

For answering the third research question RQ3, the study collected publications that support real-time data processing. This data can be generated either in using a stream format, using some data model transmitted in a real-time stream, or manipulating this data from the collection up to the inference occurs in real-time. Table 4.3 shows the results. Publications assigned as 'No' are publications that neither work with real-time data nor papers that does not describe how data is transmitted (real-time or not). It is worth noticing that the majority of CAM publications assigned as 'No' use a strategy of contexts inference without ML algorithms.

Table 4.2. Aswering RQ2: 28 publications presenting the use of different ML algorithms associated to different Context Reasoning techniques. **Notes:** *Pub* = Publications; *FLB* = Fuzzy Logic-based *SVM* = Support Vector Machine; *KNN* = K-nearest Neighbors; *DT* = Decision Tree; *RF* = Random Forest; *BN* = Bayesian Network; *HMM* = Hidden Markov Model; *NB* = Naive Bayes; *ANN* = Artificial Neural Network; *FL* = Fuzzy Logic; *RM* = Regression Models; *PDS* = Probabilistic Model for Data Stream.

Pub.	**ML Algorithms**										
	optimization-based		Distance-based	Rules-based		Probabilistic-based				FLB	
	SVM	*ANN*	*KNN*	*DT*	*RF*	*BN*	*HMM*	*NB*	*PDS*	*FL*	*RM*
[40]	-	-	-	-	-	-	-	-	-	√	-
[86]	√	-	-	-	-	-	-	-	-	-	-
[98]	√	-	√	√	√	-	-	-	-	-	-
[13]	-	-	-	-	-	-	√	-	-	-	-
[37]	√	√	√	√	√	√	√	√	-	-	-
[142]	-	√	√	√	√	-	√	√	-	√	-
[95]	√	-	√	√	√	-	√	√	-	-	-
[52]	-	√	-	-	-	-	-	-	-	-	-
[68]	-	-	-	-	√	-	-	-	-	-	-
[87]	-	-	√	-	-	-	-	-	-	-	-
[110]	-	-	-	-	-	-	-	-	-	-	√
[122]	-	-	-	-	-	-	-	√	-	-	-
[102]	√	-	√	√	√	√	-	√	-	-	-
[118]	√	√	√	√	-	-	√	√	-	-	-
[134]	-	-	-	-	-	√	-	-	-	-	-
[143]	√	√	√	√	-	√	√	√	-	-	√
[8]	-	-	-	-	-	-	√	-	-	-	-
[99]	-	√	-	-	-	-	-	-	-	-	-
[23]	-	-	-	-	-	-	-	-	-	-	√
[117]	-	-	-	√	-	-	-	-	-	-	-
[108]	-	-	-	-	-	-	-	-	√	-	-
[106]	-	√	-	-	-	-	-	-	-	-	-
[6]	√	-	-	-	-	-	-	-	-	-	-
[27]	√	-	-	-	-	-	-	-	-	-	-
[130]	-	-	√	-	-	-	-	-	-	-	-
[12]	-	-	√	-	-	-	-	-	-	-	-
[104]	√	√	√	-	-	-	-	-	-	√	-
[103]	-	-	-	-	-	-	-	-	-	√	-

Table 4.3. Answering RQ3: CAM publications that support and do not support real-time data processing.

Support real-time data processing	# Pub.	Publications
Yes	26	[50] [13] [52] [95] [68] [58] [122, 98] [102],[142] [143] [99] [108] [118] [130] [134] [37] [12] [135] [10] [139] [136] [72] [103] [96] [110]
No	17	[114] [48] [86] [117] [87] [118] [65] [107] [8] [23] [66] [123] [106] [6] [104] [27] [137]

4.1.4 HAR Scenarios of Application Related to CAMs

For answering the fourth research question RQ4, this study gathered publications that tackle different possible scenarios of HAR application, shown in Table 4.4. These scenarios contain all algorithms for context reasoning under CAM for HAR. There is only 36 papers in this table because papers that do not tackle the HAR problem were not considered for answering this RQ4. It is worth noticing that Ambient Assisted Living and Daily Living Activities are the kinds of HAR with more associated papers.

4.1.5 Relation of Our Themes of Interest in the Analyzed Papers

The study conducted a broader overview based on selected publications. The author considered three main areas of interest: Context-Aware Middleware (CAM), Human Activity Recognition (HAR) and Machine Learning (ML) algorithms. The goal is to understand what are the relationship among them. In this way, the author questioned what are the studies that propose solutions in the following three intersections: ML and CAMs; HAR and CAMs; and HAR, ML and CAM. This study did not consider the intersection

Table 4.4. Answering RQ4: Different scenarios considering HAR.

Human Activity Recognition scenarios	# Pub	Publications
Ambient Assist Living	15	[40] [86] [48] [98] [107] [37] [102] [8] [117] [106] [137] [135] [10] [139] [136]
Medical Informatic	6	[114] [95] [107] [12] [72] [104]
Terrorist Attacks Detection	1	[13]
Activity Daily Living	13	[50] [52] [68] [87] [58] [102] [118] [99] [66] [123] [137] [6] [96]
Sport Assistance	2	[108] [130]

between HAR and ML because we are interested in situations where scalability is a main concern, such as ambient intelligence and smart cities scenarios, where a huge large of sensors could be used; Ambient Assisted Living (AAL) problems, where systems analyzes people behavior in an assisted environment; and so on.

For answering this question, the author constructed a Venn diagram, explicitly showing the intersection between the three interested areas. Figure 4.2 shows a visual representation of the constructed Venn diagram. Each circle in Venn diagram is relative to one of the three research areas: Context-Aware Middleware (CAM); Machine Learning (ML); and Human Activity Recognition (HAR). The set $\{(CAM \cap ML) - HAR\}$ (pink color in the figure) gather 5 publications that present context-aware middleware which context inference is made using ML-based models. The set $\{(CAM \cap HAR) - ML\}$ (lime or green color in the figure) gather 16 publications that present CAM project, description and context reasoning modeling (without ML algorithms) for tackling HAR domain. The set $\{(CAM \cap ML \cap HAR\}$ (blue color in the figure) gathers 22 publications, which present a CAM project with an inference method based on ML for HAR scenarios. As mentioned before, $\{(HAR \cap ML) - CAM\}$ does not present any publication because of the exclusion criteria of the systematic review. Table 4.5 shows all publications distributed in each area.

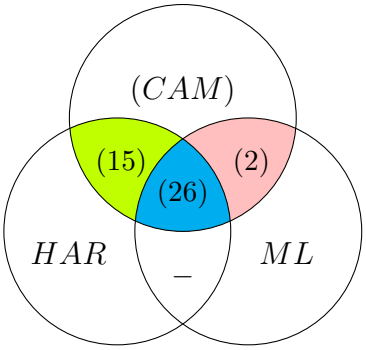

Fig. 4.2. Venn's Diagram for each research area. (N) represents the number of publications.

Table 4.5. Publications present in each (sub)set related to intersection areas of our Venn diagram shown in Figure 4.2.

Sets for intersections	Publications
$\{(CAM \cap ML) - HAR\}$ (2)	[110] [103]
$\{(CAM \cap HAR) - ML\}$ (15)	[114] [50] [48] [65] [58] [107] [136] [66] [123] [137] [135] [10] [139] [72] [96]
$\{CAM \cap ML \cap HAR\}$ (26)	[86] [142] [98] [13] [95] [40] [99] [52] [68] [87] [102] [122] [37] [134] [143] [8] [104] [23] [118] [117] [108] [6] [130] [12] [27] [106]

This document analyzed more carefully $\{CAM \cap ML \cap HAR\}$ subset to identify how ML algorithms act as a context reasoning technique in CAM projects in the HAR scenarios. this study observed that most systems use predefined models obtained using a training data collected through multiple resamples. However, the analysis is performed offline.

4.1.6 Papers in the Intersection of the Themes

In this section, we briefly describe all our analyzed papers. We firstly present them divided into the sets constructed using our Venn diagram. After, Table 4.6 shows the middleware proposed by each paper. In this table, papers marked with "survey" are secondary works and do not propose or present a new middleware; and papers marked with "no name" means that the authors did not give a name for their middleware.

- *Papers in* $\{(CAM \cap ML) - HAR\}$: These studies present CAM which context inference is made using ML-based models, but not tackle HAR problems:
  - ◇ **[110]** proposed an energy efficient framework for data gathering based on compression techniques, in particular as a component of a middleware for the smartphones.
  - ◇ **[103]** implemented a CAM to learn robot tasks.
- *Papers in* $\{(CAM \cap HAR) - ML\}$: These papers present CAM project, description and context reasoning modeling (without ML algorithms) for tackling HAR domain:
  - ◇ **[107]** proposed a context-aware service delivery under a middleware. This project aims to exploit the variety of interaction mode/capabilities according to the current situation, users, and the context in a distributed human-computer interface.
  - ◇ **[48]** developed a framework that supports intelligibility and user control of context-aware applications. It identifies and exposes the internal middleware models which influence adaptation decisions, and facilitates generations of explanations regarding evaluations of the models. These middleware models supports different context reasonings and context models developed using context modelling language.
  - ◇ **[66]** presented a CAM that provides context sharing in a cooperative way.
  - ◇ **[123]** proposed a CAM based on techniques and principles from semantic web and multi-agent systems domains. This middleware provides the necessary flexibility to deploy different kinds of context provisioning patterns to address different ambient intelligence applications.
  - ◇ **[10]** introduces a meta-level architecture for disseminating a *high-level* of context abstraction for heterogeneous profiles and service sources via a top-level ontology.

- ◇ **[114]** presented a initial study using CAM for medical workflow organization.
- ◇ **[50]** proposed a framework for collaborative-based body sensor networks. This tool can collaborate with each other to achieve a common goal. They allow the development of smart wearable systems for cyber-physical pervasive computing environments.
- ◇ **[65]** presented a CAM architecture for extracting contextual information from social networks in emergency scenarios considering Smart Cities environments.
- ◇ **[58]** proposed a context-aware system for modeling activity via Markov Logic Network. The system recognize atomic and complex (composite) activities.
- ◇ **[137]** presented a framework end-to-end web-based in-home monitoring system for convenient and efficient care delivery. This framework also supports an IoT (Internet of Things) middleware.
- ◇ **[136]** presented a prototype for Web-based IoT smart home systems. This project helps older people live in their own homes independently, longer, and safer. It exploits the heterogeneous contextual information (e.g., daily activities) captured and learned from IoT devices. So, it deliveries the context reasoner as a service and makes decision in an user-friendly way.
- ◇ **[135]** proposed a middleware property for detection of asynchronous context.
- ◇ **[139]** reviewed the application of the Semantic Web to pervasive and sensor-driven systems dedicate to stream data modeling and stream context reasoning.
- ◇ **[72]** presented sensors-based system to support clinicians' diagnosis for people suffering from Alzheimer disease and dementia.
- ◇ **[96]** proposed an approach to evaluate the risk of complex activities based on actions and the user performance in these activities.

Papers in $\{(CAM \cap ML \cap HAR\}$: These papers present a CAM project with an inference method based on ML for HAR scenarios:

- ◇ **[86]** described the construction of a system that uses information from mobile devices to detect person presence. For this, the paper uses a middleware in the cloud focused in Human-Computer Interaction.
- ◇ **[40]** proposed Ambient Assisted Living (AAL) solutions using robot ecology concepts with information processing algorithms, which include perception, learning and actuation.

- ◇ **[104]** surveyed the role of Wireless Body Area Network (WBAN) applications and network architecture dedicated for data collection, data transmission and data analysis in the realm of Internet of Things.
- ◇ **[98]** modeled user occupancy and activity patterns using ML approaches.
- ◇ **[13]** described a framework which supports a middleware for early identification and prediction of terrorist actions.
- ◇ **[52]** developed an incremental approach for recognizing, predicting, and tracking Activities of Daily Living (ADLs) within a smart home infrastructure. This work uses a middleware that focuses on accessible and inclusive user interfaces by allowing any device or service to be accessed and manipulated by any controller. Moreover, this tool provides a standardized architecture that supports the flexible integration and reuse of heterogeneous sources.
- ◇ **[134]** provided a middleware that supports a solution to the Bayesian network for uncertain inference and designing a mechanism to give the middleware access a powerful computational ability.
- ◇ **[143]** described the emerging concepts to applications of context-awareness in mobile platforms and research directions, pointing out difficulties and possible solutions.
- ◇ **[8]** proposed an ambient intelligent system of in-home psychiatric care service for emergency psychiatry (EM-psychiatry).
- ◇ **[99]** presented an activity recognition system that classifies in the near real-time a set of common daily activities. The data were sampled by sensors embedded in a smartphone carried out by the user from sensors. The authors used middleware capabilities in the form of service discovery and communication.
- ◇ **[87]** proposed a CAM for HAR, instead of using a simpler middleware, as the one proposed by [99].
- ◇ **[23]** proposed an intelligent middleware for mobile context-aware frameworks. It is able to learn sensor usage habits, inimizing energy consumption of the system.
- ◇ **[108]** presented an approach for modeling statistical dependencies in multivariate discrete sequences by using hyperdimensional random vectors in HAR domain.
- ◇ **[6]** presented a sensor system for enhanced context and activity recognition while requiring a minimal set of sensor nodes.

- ◇ **[130]** introduced a framework for professional volleyball training based on machine-learning techniques.
- ◇ **[12]** proposed an intelligent framework for the smart devices to enable continuous verification of users under different social networking applications.
- ◇ **[117]** proposed an immobility tracking system that can analyze physical activities and identify immobility behavior of the elderly on time. This system uses a message-oriented middleware.
- ◇ **[95]** presented a survey containing an overview of essential functions and services for monitoring and detecting human behavior, including its concepts, approaches and processing techniques. Moreover, they present an analysis and evaluation of the existing research approaches in the area of e-health systems, as well as some middlewares for e-health systems.
- ◇ **[68]** proposed an edge intelligence framework for building IoT applications based on a specific middleware. Authors also implemented a user activity recognition engine, and compared its performances on either an edge device or cloud servers.
- ◇ **[122]** proposed a context middleware that provides context information to applications and infers logical context.
- ◇ **[27]** presented a privacy-preserving middleware called PRECISE that implements context-aware services based on location. PRECISE provides users with custom context-aware recommendations.
- ◇ **[106]** introduces an annotation mechanism for an AAL platform that can recognize and provide alerts for generic activities and behaviors.
- ◇ **[102]** surveyed context awareness from an IoT perspective. He presented the necessary background by introducing the IoT paradigm and context-aware fundamentals. This publication assisted us as a basis for understanding definitions, terms, and concepts in context-aware systems.
- ◇ **[118]** reviewed, through a historical perspective ,ubiquitous and pervasive computing, ambient intelligence and wireless sensor networks. After, they described CAC researches.
- ◇ **[37]** surveyed any topics concerning to AAL. In particular, the authors propose future works related to ambient sensor data and data processing through ML algorithms for detecting and classifying activities.

◇ **[142]** surveyed many important aspects about CAMs in context-aware HAR systems. The authors discussed communication and support services and embedded infrastructure in these middlewares. Moreover, data mining process was present since the collect step up to the inference process.

Table 4.6. Different middlewares in the analyzed papers. Papers marked with "survey" are secondary works and do not propose or present a new middleware. Papers marked with "no name" means that the authors did not give a name for their middleware.

Publications	Middlewares	Links
[12]	no name	-
[99]	giraffPlus	http://www.giraffplus.eu/
[101]	survey	-
[87]	capm	-
[86]	no name	-
[98]	no name	-
[48]	PACE	https://tinyurl.com/u5kcfvd
[13]	no name	-
[8]	no name	-
[6]	no name	-
[123]	CONSERT	https://tinyurl.com/twcqzds
[130]	No name	-
[23]	no name	-
[66]	no name	-
[134]	(CARM)	-
[110]	no name	-
[107]	no name	-
[108]	no name	-
[142]	survey	-
[143]	survey	-
[52]	openURC	www.openurc.org
[10]	Kalimucho	-
[68]	WuKong	https://tinyurl.com/unaclwu
[95]	survey	-
[40]	peis	https://tinyurl.com/sjr82fr
[50]	no name	-
[137]	WITSCare	-
[72]	no name	-
[118]	survey	-
[58]	no name	-
[114]	no name	-
[65]	COLLEGA	-
[106]	no name	-
[37]	survey	-
[27]	precise	-
[117]	MOM	-
[122]	ConProVA	-
[104]	survey	-
[135]	MIPA	https://tinyurl.com/w4yrwl9
[139]	survey	-
[103]	no name	-
[96]	no name	-

4.2 Services Available in CAMs for HAR

In this step, this study analyzed publications that propose CAMs or services in CAMs for supporting the development of applications involving HAR problems, using or not ML algorithms to construct models to compose the context reasoner. Fig. 3.1 shows the steps usually followed by the solutions for tacling HAR problems, named: (i) Raw Data; (ii) Sliding Window; (iii) Preprocessing; (iv) Feature Extraction; (v) Inference; and (vi) Smoothing. In what follows, we summarize the main aspects and services presented in literature for each of the steps.

Related to (i) Raw data, the analyzed studies frequently use sensor technologies, including small sized sensors, *e.g.*, accelerometer, proximity sensor, magnetometer, GPS, and so on. In general, these resources are present in mobile devices nowadays, such as smartphones, smartwatches, and others. Overall, accelerometer is usually applied as either a tool to measure walking consumption or a monitor to recognize user physical activities such as postures, gestures and movements. Most of the measured features or events are related to human posture or movement, environment or human physiology signals [142, 143]. For instance, [117, 87, 23] use accelerometers and GPS, WiFi and cell towers as human posture or movement features; [37] use temperature and humidity sensors, microphones and cameras as environmental features; and [8] use attachment of external devices such as heart rate or electrocardiogram, finger pulse, and others as physiological signals.

Related to (ii) Sliding Window, (iii) Preprocessing and (iv) Feature Extraction, one of the key advantages when using CAMs for implementing HAR systems is allowing an abstraction level that deals with sensors processing, including sensors removal and insertion, data processing and others. A unified interface for heterogeneous sensors offered by CAMs brings flexibility to implement HAR systems using difforeat external data. In this way, different data sources can provide various kinds of contexts [102, 134], allowing many diverse applications to be constructed in an easier way than that not using CAM. Also, CAMs for implementing HAR systems are capable of wrapping, *i.e.*, controlling physical devices and interacting with them to receive data, analyzing, and delivering physical world information, through sensor networks, embedded systems, RFID or NFC tags, and so on, to the application services in a transparent way. This degree of transparency allows a clear separation between the application layer and the internal middleware layers. Therefore, the applications can receive the context but not know the source of it. Hence, a CAM creates a shielded interface by both enhancing the level of abstraction support

needed by the application, hiding lower layer operations between the physical layer (i.e., hardware and communications) at the bottom and the application layer at the top.

Related to (v) Inference and (vi) Smoothing, the analyzed studies usually maintain simple semantic structures for inference or context reasoning. In general, the used models appear to be simple constructions that guarantee legibility for humans. Thus, knowledge representation in the form of simple rules is more present in context reasoning modules of the analyzed CAMs. We understand that this occurs mainly due to expensive real-time operation in smart devices, scalability problem for labeling real data and bandwidth problem in exchanging large amounts of information when using ML algorithms. More specifically, collecting training data classes to fit statistical or other kinds of models using ML algorithms can be very challenging in some scenarios using CAMs. Therefore, sensors must be lightweight and unobtrusive. One solution is to take advantage of cloud computing technologies, enabling information and ensemble situational resources to be shared among co-located devices [82, 142, 95, 37].

4.3 Research Challenges

The thesis presented in Section 4.2 a discussion on the papers presenting CAMs that implement different context reasoners for HAR (Section 3.3) as well as which works presents CAMs (or services of CAMs) that use ML for constructing models to create their context reasoners. However, there are many limitations in these studies concerning to data stream management under different types of distribution, how to deal with the evolution of models that evolve over time and others.

Abdallah et al.[4] state that obtaining training data for ML into a supervised learning strategy is an expensive real-time operation for a smart device. There are difficulties related to data acquisition and analysis; resource management for training samples storage; data fusion when different data types are available; scalability in labeling data; and bandwidth on exchanging massive volumes of information. Therefore, using sensors should be computationally light, and ML algorithms should be not only using batch methods, which can be computationally costly in situations where many users may use the system. In this way, in what follows, we discuss the main research challenges for evolving CAMs for complex HAR systems.

One of the main advantages in CAMs is offering services for data fusion. [50] and [99] presented mechanisms for this end. However, only data from common sensors, such

as body sensors, were used. On the other hand, for HAR systems, investing in image identification problems by companies may lead to efficient indexing and storage schemes to improve user experience. This also requires learning methods from raw videos and video synthesizing based on its content. For this end, CAMs must evolve to include these types of services.

Traditional HAR approaches stand on the assumption that collected data distribution is stationary. This assumption is commonly violated when data provided by real-time sensors evolve over time. Data stream mining and stream learning approaches deal with this type of data, which brings constraints imposed by the stream data nature, such as Concept Drift [20]. Concept drift in HAR can be abrupt, for instance a change in walking pattern after an accident; gradual, for instance, an evolution of walking pattern for children; incremental, for instance, a change in walking pattern during healing from an injury; or recurrent, for instance, a sudden change in the walking pattern according to the situations [4]. In this way, there is a lack of services in CAMs that manage the distribution of non-stationary data in context-aware scenarios, as we could not find in literature many stream mining approaches used to address HAR in CAMs.

There are just a few works that tackle HAR with machine learning models considering the limitation of memory consumption. Therefore, cheaper learning algorithms and models, in the way that they consume a lower quantity of memory, may be present in these systems to assist the inference in these activities. Some studies have adopted strategies that deal with reduced memory [121, 124, 115, 77]. All of them use batch learning algorithms. However, none of them explicitly considers restriction in memory consumption and do not compare the two approaches (batch and online). This comparison is important, especially when consumption resources are limited.

Moreover, it is important to observe that, in HAR situations, obtaining online labeled data can be challenging. From the point of view of labeled data availability, there are two approaches [64]: (i) passive learning, where the data is previously labeled (in the case of batch learning) or arrives labeled in the stream labeled data; or (ii) active learning, where the data must be labeled over time but users. In this way, new proposals for obtaining online data in this scenario could be interesting, specially considering active labeling techniques, such as proposed by [93], whose propose a methodology for semi-supervised learning for labeling data for HAR. Other inspirations for this can be found in proposals for other problems where labeling is difficult [73].

Another open challenge is regarding to complex HAR. One way for modeling complex

HAR is representing a sequence of dependent atomic activities. Thus, an efficient HAR system may focus on dealing with the dependency among data for predicting executed activities. In the studies we analyzed, only [58] tackled composite activities. There are new approaches for tackling complex HAR, such as the ones proposed by [116] and [56], that should be considered for being added as services in CAMs.

Finally, there are many researchers discussing requirements for guaranteeing Trustworthy AI (TAI). According to [125], the fifth TAI principle is explicability, which seeks to produce (more) interpretable AI models whilst maintaining high levels of performance and accuracy. According to the guidelines of the European Union for TAI [43], explicability is composed by transparency, interpretability, reliability and controllability of AI. So, there is a need for investigating which of these aspects are important for HAR as well as including these features as services in CAMs as well, in order to improve the development of HAR solutions.

Resuming, there are some significant research directions for evolving CAMs to incorporate ML and data processing services for enhancing HAR systems development:

- In ubiquitous computing, resources managing (for instance memory consumption) is crucial. So, it is important the use of strategies that lead the memory comsumption in machine learning models deployed in this systems.
- CAMs must offer services to be able to develop complex HAR systems. This may lead to facilitate the development of applications in Ambient Intelligence in general, including Ambient Assisted Living applications, which are more and more necessary nowadays.
- CAMs must offer services for allowing adaptive models for HAR, considering the dynamics of both activities and context reasoning in streaming environment. For example, the context reasoner should perceive that the activity pattern is not static. For example, a person may walk differently from one day to another due to an accident.
- CAMs must offer services for online labeling the data, in order to allow stream learning algorithms to fit models for each user, considering their evolution in their activities.
- There are some papers presenting good results using deep learning algorithms for HAR, such as the method proposed by [144]. Offering deep learning framework in

CAM is interesting for faciliting by development of HAR systems. However, issues regarding to computational cost must be addressed in this scenario.

- Transfer learning aims to adapt a solution to a task or domain to help building models to other domains or tasks by exploiting prior knowledge [100]. Collecting labeled data for new users can be difficult. So, CAMs could also offer services for supporting transfer learning for HAR problems.

Despite the relevance and the extensive study in Context-Aware Middleware, our current research only covers Context-Aware Computing, dedicated to using a middleware in one of our studies.

4.4 Final Remarks

In this Section, we aimed to understand the state-of-the-art in the development of CAMs for aiding the construction of HAR applications, specially when using ML. To achieve this end, we conducted a systematic literature review, selecting publications related to CAM projects that either incorporate ML algorithms or in for HAR solutions. We were also particularly interested to understand how these middlewares deliver context inference (reasoning) service. Although the literature presented many different methods for context reasoning, this work emphasized cases involving ML. Our main interest in ML is due to has been presented good results in many different types of HAR problems. We answered our four research questions through analyzing the papers selected in our systematic review processes.

As result of our analysis, we could observe some limitations that deserve attention from context-aware researchers community. CAMs should evolve to offer services for (i) developing complex HAR systems; (ii) allowing adaptive models for HAR, including the management of non-stationary data by stream learning algorithms; (iii) allowing online labeling the data; (iv) incorporating deep learning frameworks; and (v) supporting transfer learning for HAR problems.

Chapter 5

Impact of Memory Control on Batch Learning versus Stream Learning in Human Activity Recognition

HAR systems need reasoning methods to infer high-level knowledge about the user and/or the environment. In general, HAR systems use several machine learning approaches to construct models for inference, such as distance-based [87, 130], probability-based [8, 14], optimization-based [104, 95, 86], and rules-based [117, 68]. Most of the state-of-the-art works propose the use of models based on batch machine learning. These models can lead to segmented data and extract the pattern of a (relatively) low volume of data. However, the disadvantages of batch learning algorithms are: (i) they can need high processing capacity and high memory consumption, in special; and (ii) they consider the data distribution is stationary [24, 138], *i.e.*, the data distribution does not evolve over time. Regarding the disadvantage (i), distance-based models are the ones that most consume memory among all the approaches, due to memorizing data. Probabilistic-based (more specifically the Naive Bayes family of algorithms) and rules-based (or tree-based) are most simple, due to the facility of calculating probabilities and defining rules, respectively. However, when retraining batch models (given some frequency-time) happens (due to the new data stream), the systems that use batch learning tend to consume much memory. The consumption happens due to having to store the entire stream. This consumption is necessary to reconstruct a new model when using batch learning algorithms [4].

Regarding the disadvantage (ii), Online Models (OM) [55] (constructed using Online or Data Stream Learning algorithms) were presented in the literature with more flexibility to lead non-stationary distribution data. Moreover, Online Models act on scenarios where data distribution changes are present. In this way, OM appears as a solution for systems

with low memory resources. The model incorporates new information of segmented data of activities, which evolves with high speed, detecting changes, and adapting the activities models to the most recent information. However, these models feed on recently labeled data. Furthermore, they require a minor computational cost. On the other hand, these models may require too much data to converge [20], as each instance is seen by the algorithm only once, which may lead to a minor potential for generalization when compared to batch algorithms, considering the same memory consumption.

An essential parameter is the point of data adaptability. As explained in previous chapters, the batch learning approach adapts periodically, reprocessing the most recent data. In contrast, the online approach does self-adaptation processes over time, as each sample is labeled.

Considering this panorama, analyzing the performance of batch models and online models focused on memory consumption is important. The objective is to verify the impact in batch models with similar memory resources to online models, considering the quality of the inference. To this end, memory consumption must be fixed for batch models, with memory size similar to the data stream learning models. In this way, our aim in this work is to analyze the robustness of batch models constructed considering limited memory resources, compared to online models. To this end, we created a *memory control module* that manages memory consumption when new data arrives to construct new batch models. Our focus is on verifying if the batch learning models on this setup can obtain the same (or best) results versus data stream learning algorithms.

5.1 Experimental Methodology

This study only consider the Atomic Human Activity Recognition (AHAR). As described in Section 3.1, AHAR process can be divided into various steps, as illustrated in Figure 3.1: *Raw data collection*, *Sliding window*, *Preprocessing and feature extraction*, and *Inference*. *Raw data collection* collects data from differents datasets described on Chapter 5.2.1. For *Preprocessing*; our study fuses data from different sensors. Each dataset describes its sensors (e.g accelerometer, gyroscope). The feature extraction is different for each dataset. Section 5.2.3 describes the features of construction for each dataset.

The key point in this work is the inference step. Our assumption is that batch learning, and online algorithms models can consume limited memory size in deployment environment. Thus, the author created a *memory control module* that helps to fit the

Algorithm 1 memory control module for (BM_{mc})

Input: $\mathbf{x}_i$, incoming segment stream; $\mathbf{y}_i$ predicted label for x_i, incoming stream; M: size of memory cap; S = security samples; Registry R
Output: A model BM_{mc} with size at most size M.

BM_{mc} is created under S samples
repeat
  $\mathbf{y}_i \leftarrow Classify(\mathbf{x}_i, BM_{mc})$;
  $R \leftarrow R \bigcup x_i$.
  $Retrain(BM_{mc}, x_i)$
  if $size(BM_{mc}) > M$ **then**
    $BM_{mc} \leftarrow \emptyset$
    repeat
      Take $x_i \in R$
      $Retrain(BM_{mc}, x_i)$
      $i \leftarrow i + 1$
    until There no are object in R **or** $size(BM_{mc}) > M$
  end if
until There are no stream x_i.

batch models (BM_{mc}), under a respective memory size of the model. Algorithm 1 describes the *memory control module* for fitting models (BM_{mc}). Let x_i be a new segment of the stream, and M be the memory size for constructing the model. For this value, we selected the memory size used by an online model within the same learning approach (probabilistic-based, rules-based, and so on). Let S (security samples) be a set of samples that offers initial information for (BM_{mc}). We used 7% of samples for the initial fit distributed uniformly per each class. Registry R is a subset of the dataset with recent x_i labeled instances. This registry is similar to a buffer for control the sample sequence. Its objective is maintaining the model updated with the new data arriving in the stream, considering a time-frequency. The algorithm retrain BM_{mc} model for each new segment x_i. This element is stored in R. If the memory of BM_{mc} is much larger than M, the algorithm uses the most recent labeled instances $x_i \in R$. The order in the sequence x_i and x_{i+1} is preserved.

Algorithm 2 Retrain function

Input: $\mathbf{x}_i$, incoming segment stream; A model BM_{mc} with size at most size M

the system gathered all object ($OBJs$) used on model BM_{mc}
BM_{mc} is trained as from $OBJs \bigcup x_i$

5.2 Materials and Methods

5.2.1 Datasets

We selected the datasets HAPT [1] (two parts), REALDISP [2]. We used the two datasets individually due to the different number of samples. Thus, it allows us to create one thin dataset and one big dataset. The choice of all datasets is due to the lack of studies involving datasets with labeled data concerning transition labels. This format of the dataset may allow support to develop complex human activities [112].

HAPT [112]: The dataset was carried out with a group of 30 volunteers aged 19-48 years. They performed a set of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs, and walking upstairs). The experiment also included postural transitions between the static postures: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. All participants wore a smartphone (Samsung Galaxy S II) on their waist during the experiment. The study captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz using the embedded accelerometer and gyroscope of the device. The experiments labeled the data manually. The obtained dataset was randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data. In our study, we separate two datasets: HAPT1 and HAPT2 means the training set and test set, respectively.

REALDISP [16]: The REALDISP (REAListic sensor DISPlacement) dataset was initially collected to investigate the effects of sensor displacement in the activity recognition process in real-world settings. The dataset includes a wide range of physical activities (warm-up, cool-down, and fitness exercises), sensor modalities (acceleration, rate of turn, magnetic field, and quaternions), and participants (17 subjects). The dataset contains information for three different scenarios depending on whether the sensors are positioned in predefined positions or placed: In the first scenario, the sensors are positioned by the instructor in predefined locations within each body part. In the second scenario, the user chose the position of the sensors on the body parts specified by the instructor, but without hinting where they need to be precisely placed. Finally, the instructor introduced an intentional mispositioning of sensors using rotations and translations concerning the ideal placement.

[1] https://bit.ly/3frazYN
[2] https://bit.ly/35N4ZM8

Table 5.1 describes the total of activities (number of classes); window size is the quantity of data under the window and information about overlapping. The total of segments is the number of samples for test and training models. Each segment is created by a sliding window, and lastly, the number of features extracted. More details about the execution process are described in the next subsection.

Table 5.1. Description of datasets used in Chapter 5.

Datasets	*HAPT*	*REALDISP*
Total of activities	12	33
Total of users	30	17
Window size (segments)	128 ($\approx$ 2.56s)	200 ($\approx$ 4s)
Total of segments (number of samples)	7767 (HAPT1) 3162 (HAPT2)	9213
Type of sliding window	Explicit Segmentation sliding windows (50% overlap)	Explicit Segmentation sliding windows (50% overlap)
Total of features	53	200

5.2.2 Deployment Environment

For the creation of a ubiquitous environment, the author decided to use in this experiment containers technology [26]. Containers provide a higher level of abstraction for the application management and contribute to development in many distributed application challenges [25], e.g. portability and performance overhead. This study used Docker technology. The use of Docker containers allows us high isolation and less resource consumption. Figure 5.1 describes the architecture of deployment environment.

In container, the author created several Images (img i), where $i \in [1, n]$. Each image process the AHAR steps in workflow (WF). Each workflow has a different deployed ML model. the setup of configuration is described in Table 5.2. We have a size of memory cap M (capsize) where the maximal of the memory where the model can reach. The λ means a "swap" that the model can not exploit. If $size(BM_{mc}) > M + \lambda$, the system will shut down.

Table 5.2. Information about deployment enviromment.

HAPT1			HAPT2			REALDISP		
Model	**Capsize**	**λ**	**Model**	**Capsize**	**λ**	**Model**	**Capsize**	**λ**
J48	76,9 KB	500 KB	J48	37,2 KB	500 KB	J48	89,3 KB	500 KB
KNN	35,3 MB	500 KB	KNN	14,5 MB	500 KB	KNN	15,3 MB	500 KB
NB	471 KB	500 KB	NB	471 KB	500 KB	NB	456,5 KB	500 KB
SMO	76,4 KB	500 KB	SMO	76,4 KB	500 KB	SMO	1,5 MB	500 KB

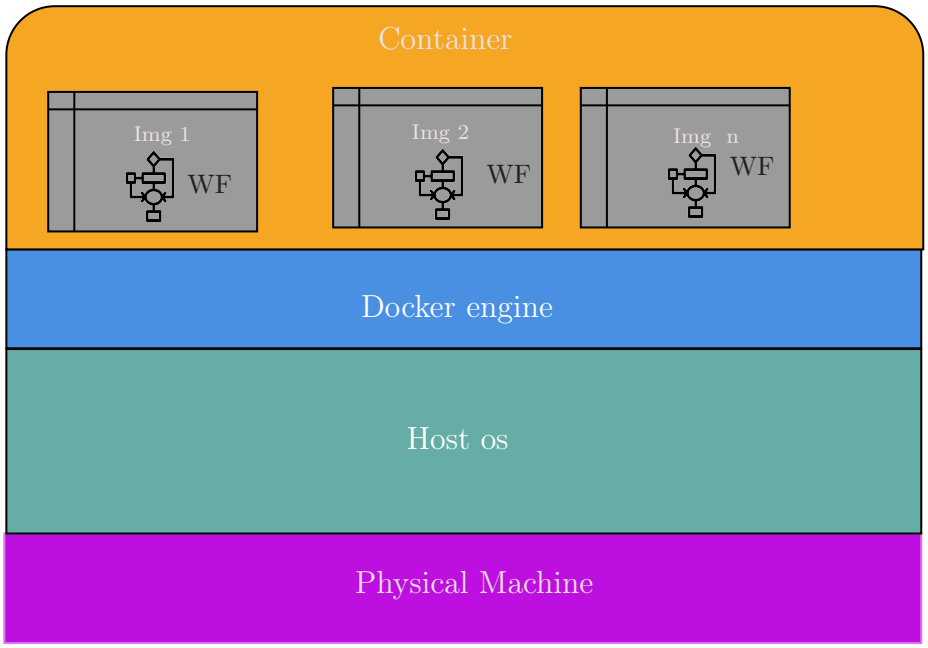

Fig. 5.1. Deployment enviroment.

5.2.3 Experimental Setup

Firstly, the data flow as a stream in raw data format. For data segmentation, the experiment used a fixed size of the sliding window. In this technique, a window is moved over the data stream to extract a data segment with a fixed size that is then used in subsequent processing steps. In this work, there is a 50% overlapping for all datasets.

After, the process of feature extraction is applied. For HAPT, the author used the extracted feature processing based by Anguita et al. [11]). We generated 53 features enrolling in this dataset. For REALDISP, we used 200 features for construction. Its calculation is based on mean, standard deviation, variance in x, y, and z. All of this, the author have a set of features (named by F), and created $\|F\|$ as a feature. We represented all features as a vector V. From V, we create a histogram in V_x,V_y,V_z. We

gathered all features, and generated A_i with 40 features. This dataset has five devices ($i = 5$) distributed on the body. So, we created A_i for each device and gathered all features. So, the author have 200 features.

We selected four batch learning algorithms and four data stream learning algorithms at the level of comparison between the two types of learning. To deal with different types of distribution and model construction methods, we chose for each approach of learning: an optimization-based algorithm, a distance-based algorithm, a probabilistic-based algorithm, and a rules-based (using trees). The batch algorithms are SMO, Ibk (KNN) ($k = 5$), Naive Bayes, and J48 (C4.5), considering all are implemented in WEKA [62]. The algorithms with an online approach are SGD, SAMkNN ($k = 5$), Naive Bayes (online version), and Hoeffding Adaptive Tree. All classifications algorithms are implemented on framework MOA [21]. The choice of these algorithms is due to their robustness and performance. The choice of the k value for the distance-based algorithms (SAMkNN and KNN) is due to satisfactory results compared to all k values.

During the stream processing, the batch models have a restricted memory consumption size M, according to the function *memory control module*. The M generated by the SGD is the value of M capsize for the SMO. The M of KNN value is the maximum consumption calculated by SAMkNN; the M value of Naive Bayes in the batch is equivalent to the memory consumption of the Naive Bayes online version.

Finally, the author performed a quantitative analysis between each algorithm, using accuracy and Kappa Statistic as an evaluation metric. As an evaluation strategy, we used the prequential [55]. It is the most commonly used in environments with data stream. Prequential allows the calculation of the error rate of algorithms (accumulated sum of loss functions between the prediction and observed values). We used it, especially in scenarios in non-stationary data distributions [55].

We performed these experiments using the Mac os High sierra 10.13 de 64 bits operating system and the JDK 1.8.0.201 64-bit Java runtime. The same machine was running both the sensor client nodes and the server. In addition to this, we have the following settings:

- Processor: 2.9 G2,3 GHz Intel Core i5
- Memory: 4 GB 1333 MHz DDR3

5.3 Analysis of Results

This section describes an analysis of the results of all experiments. Figures 5.2,5.3,5.4,5.4 and 5.5 show results about accuracy in HAPT (train and test) datasets. Figures 5.6,5.7 show results about same metric in REALDISP dataset. The author compared BM_{mc} models to online models with the same method. The author observed that distance-based algorithms (composed by KNN and SAMkNN) in these scenarios show oppositive results for HAPT datasets (dataset with the fewer segment data). Probabilistic-based methods (composed by Naive Bayes and Naive Bayes online) show middling and similar results. Both of them do not offer good accuracy for big datasets as REALDISP. Overall, optimization-based methods (composed by SMO and SGD) show a limited accuracy, except for SGD using REALDISP. While rule-based method as J48 as Hoeffing Adaptative Tree is a non-accentuated decrease in the learning curve, its convergence is faster than other models. In general, for all algorithms, the result shows that online models have the best results in all metrics; however, BM_{mc} models have a relation between the level of memory consumption and quality of accuracy. Increased consumption can improve the result of the accuracy metric. However, under big datasets, exhibited in Figures 5.6 and 5.7, batch models do not show good results to online methods, except for IBK.

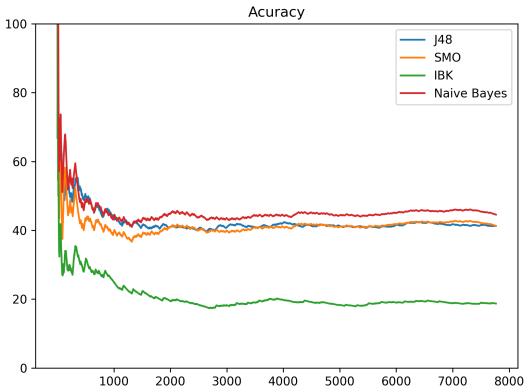

Fig. 5.2. Accuracy of BM_{mc} models for HAPT1. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

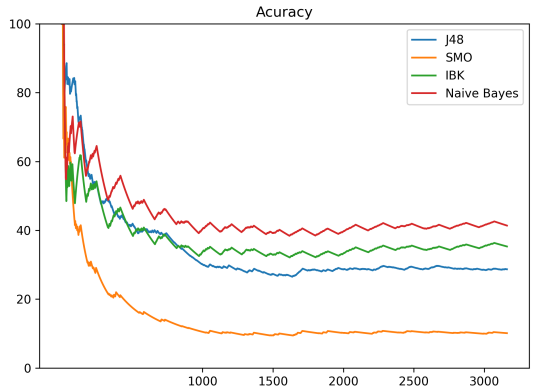

Fig. 5.3. Accuracy of BM_{mc} models for HAPT2. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

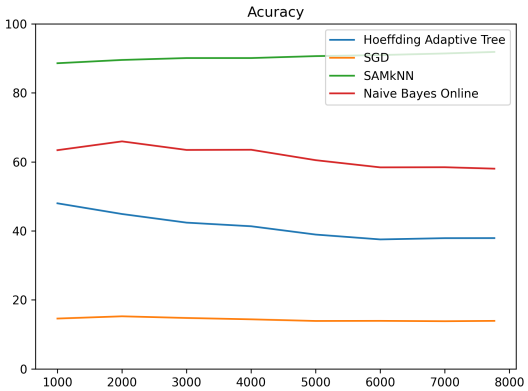

Fig. 5.4. Accuracy of online models for HAPT1.

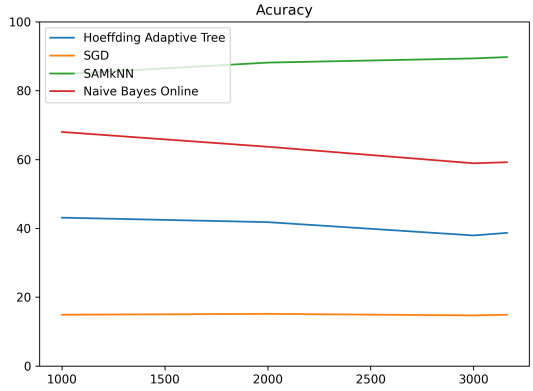

Fig. 5.5. Accuracy of online models for HAPT2. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

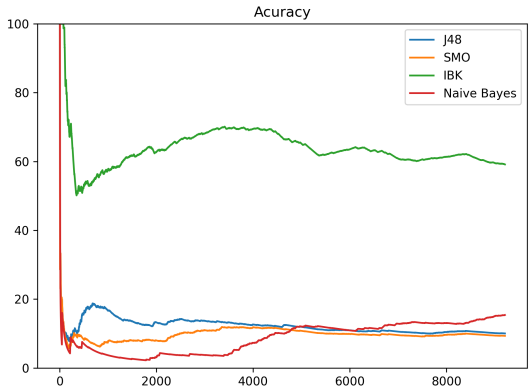

Fig. 5.6. Accuracy of BM_{mc} models for REALDISP. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

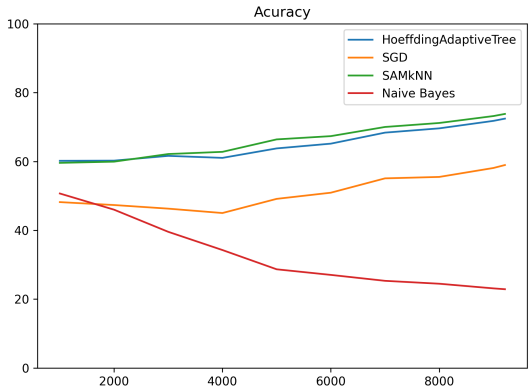

Fig. 5.7. Accuracy of online models for REALDISP. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

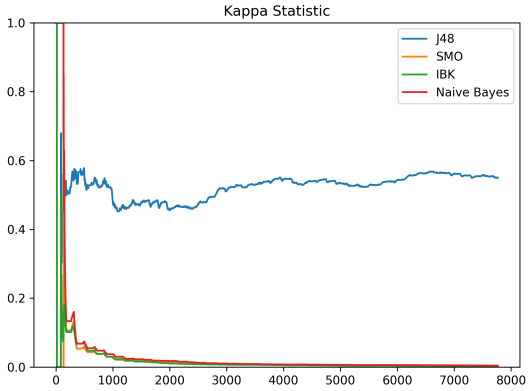

Fig. 5.8. Kappa statistic of BM_{mc} models for HAPT1. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

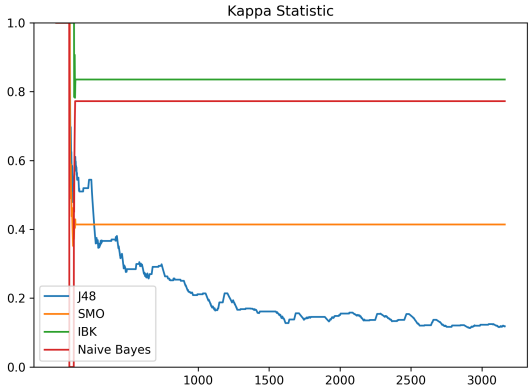

Fig. 5.9. Kappa statistic of BM_{mc} models for HAPT2. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

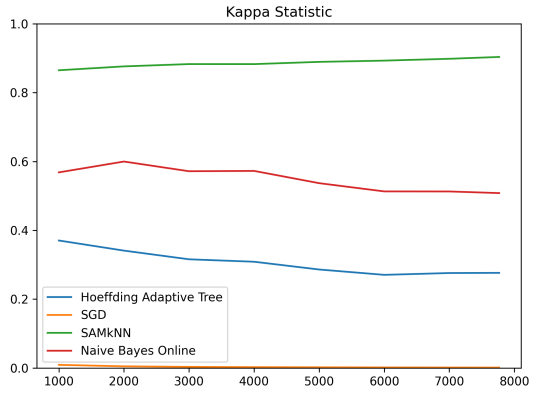

Fig. 5.10. Kappa statistic of online models for HAPT1. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

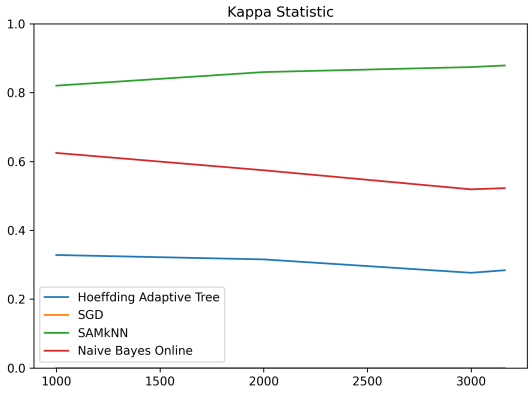

Fig. 5.11. Kappa statistic of online models for HAPT2. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

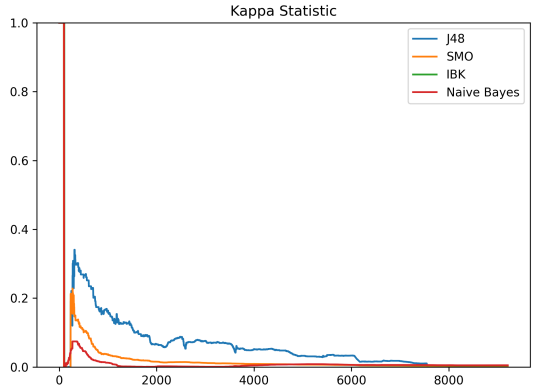

Fig. 5.12. Kappa statistic of BM_{mc} models for REALDISP. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

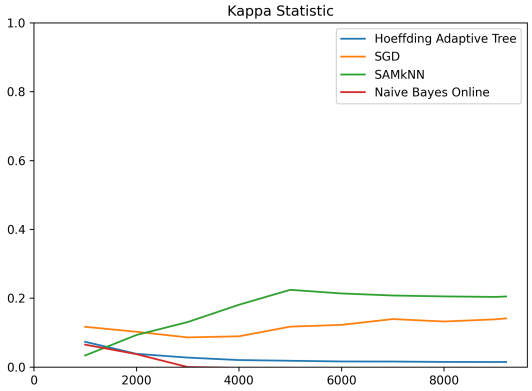

Fig. 5.13. Kappa statistic of online models for REALDISP. Axis-Y means the value of metric and Axis-X means learning evaluation instances.

Concerning to Kappa metric, Figures 5.8 to 5.13 show all results. For HAPT1, J48 executed a straightforward result; however, the other datasets, the performance was low. For data stream learning, SAMkNN has the best results. For HAPT2, two approaches obtained similar order for the evolution models. Distance-based and rules-based had substantial results. About the REALDISP dataset, neither batch models nor online models acquired good results. Overall, data stream learning obtains the best results, in particular, using distance-based models. The lack of new data (in quantity) shows the principal limitation for the batch model. Overall, strategies using data stream learning have better results than batch models with restricted memory using evaluation metrics.

Statistical Test: In order to verify if there is a statistical difference among the

algorithms according to accuracy and kappa metrics, The author executed the Wilcoxon Signed-Rank statistical test for multiples domains, one for each metric, as indicated by [69] and [35]. We aimed to compare the machine learning algorithms in batch and online in test-paired format to verify if there is a statistical difference among the algorithms according to accuracy and kappa statistic. The author compares algorithms by type of method: SGD x SMO, KNN x SAMkNN, J48 x Hoeffding Adaptive Tree, and Naive Bayes x Naive Bayes Online. The null hypothesis could not be rejected considering different critical values, indicating that the algorithms present similar results considering these metrics.

Thus, we can conclude that in a scenario with few sample data, there is no behavior significantly between the batch and online algorithms, according to the pairwise comparison between different approaches. Nevertheless, according to the graphics, there is a tendency for online algorithms to converge to a better result over time.

5.4 Final Remarks

Considering the scenario where computational resources are limited, effective strategies are welcome for HAR, in special for adaptability reasons. Systems that require agile decision making must fit into resource restrictions, such as memory size. In this way, this paper presents a comparative analysis between batch machine learning models considering restriction in memory consumption and data stream learning algorithms. The author used collected data from sensors available in benchmark datasets presented in the literature. The goal is to identify whether batch learning algorithms with limited memory resources can be robust compared to data stream learning algorithms.

In our study, the author compared four batch learning with limited memory size to four data stream learning algorithms. These algorithms follow different learning approaches: distance-based learning, rules-based learning, probability-based learning, and optimization-based learning. Our results show that batch models are more sensitive to low data volume and low memory size than online models. According to the reduced memory consumption, batch learning presented difficulty in generalizing the models. Online models converge fast and show good results, in particular to SAMkNN. Finally, we made a statistical test paired-test between algorithms that use same strategy. The null hypothesis could not be rejected indicating that the algorithms present similar results considering all applied metrics.

Chapter 6

Analysis of Adaptive Models in HAR

Considering ubiquitous scenarios, there is a huge difficulty in creating models based on multiple users; due to physical characteristics or different ways to execute the activities. For example, a sedentary person might perform walking one way, while that activity might be a jog for an athlete. Alternatively, the way a younger user sits in a chair may be different from how an older person sits.

One way to solve this problem would be to annotate data for a specific user. However, this hypothesis is invalid due to the lack of labeled data. Therefore, continuous learning approach for tailoring the model to best fit a specific user is crucial for improving recognition accuracy, as mentioned in Section 3.4. We define model personalisation as the process of tuning a general model to represent a user's personalised way of performing different activities. The vast majority of activity recognition research did not consider the personalisation issue.

Due to the intrinsic characteristics of each user, a more practical strategy is to identify a better general model that becomes specific to a specific user. Examples of works [61, 131, 84] address multi-user data to create models for HAR. However, the proposed approaches do not deal with data in a streaming environment. There is a challenge in dealing with the data distribution; while the stream evolves, the model can generate drifts from the data distribution. This scenario can fit into a problem of Learning from concept drift, when the data stream changes detection and response. A possible solution would be to identify user profiles with similar characteristics in performing activities. Thus, the evolution process occurs smoothly, and consequently, the model would adapt more and more to the end-user.

This chapter is separated into two points. The first point makes a comparison between

adaptive models over time about non-adaptive models. The analysis is broad and covers the impact of learning models adopting adaptive strategies over time. The objective is to identify how adaptive and non-adaptive models are based on training with multi-user data without distinguishing between user-profiles and the evolution of model learning over time. The process is as follows: u models were created, where u is the number of users present in the dataset. Given a respective user U, two models (Mo_1 and Mb_2) were trained with data $u-1$ users (except user data U), then we can call "leave one(user)-out". Mo_1 uses a model adaptation strategy based on the Data Stream Learning algorithms that evolves over time. Mb_2 uses a non-adaptive strategy, where the model does not update its model over time. Only the U data is tested in this scenario without any update process in Mb_2.

The second point (from the first point results), we realized that adaptive models are fundamental in an environmental stream and there is a possibility to recommend models to a specific user. This recommendation is based on relevant characteristics according to way that the users perform their activities. Thus, this second research proposes a technique for identifying similar users from generated training sets of different users. This technique integrates unsupervised algorithms, and applies hybrid similarity measures technique for recognizing activities.

6.1 Datasets

We selected the HAPT [1] and REALDISP datasets [2] (similar to last chapter). Moreover Forth-Trace [71], WISDM [80] and ExtraSensory datasets [129]. Different in previous study (Chapter 5), in HAPT, we gathered the training data and testing data in same dataset.

Forth-trace [71]: Forth-trace dataset is collected from 15 participants wearing 5 Shimmer wearable sensor nodes on the Left Wrist, Right Wrist, Torso, Right Thigh and Left Ankle. The participants performed a series of 16 activities (7 basic and 9 postural transitions).

WISDM [80]: WISDM dataset uses phone-based accelerometers to perform activity recognition, a task that involves identifying the physical activity a user is performing. Thus, the authors collected labeled accelerometer data from twenty-nine users who per-

[1] https://bit.ly/3frazYN
[2] https://bit.ly/35N4ZM8

formed daily activities such as walking, jogging, climbing stairs, sitting, and standing. They then aggregated this time series data into examples that summarize the user activity over 10-second intervals.

ExtraSensory [129]: ExtraSensory Dataset was collected from 60 participants who participated for approximately 7 days. The authors used a mobile app on their phones, and it was used to collect sensor measurements and labels (associated with a context-aware). The sensor measurements were recorded automatically for a window of 20-seconds every minute from an accelerometer, gyroscope, magnetometer, audio, location, and phone state from the person's phone, and accelerometer+compass provided by a smartwatch. In addition, the data was collected in the wild: participants used their phones in any way that was convenient to them, engaged in their typical behavior, and reported a combination of labels that fit their context.

The description of all parameters of the study is present in Table 6.1. This table describes: the total of the activities (number of classes); window size is the quantity of data under the window and information about overlapping. The total of segments is the number of samples for recognizing the activity.

Table 6.1. Description of datasets used in Chapter 6.

Datasets	*HAPT*	*REALDISP*	*Forth-trace*	*Extrasensory*	*WISDM*
Total of activities	12	33	16	8	6
Total of users	30	17	15	60	36
Window size (segments)	128 (≈ 2.56s)	200 (≈ 4s)	10 (≈ 2s)	3(≈ 20s)	200 (≈ 1s)
Total of segments (number of samples)	10929 (HAPT)	9213	3900	237600	5424
Type of sliding window	Explicit Segmentation sliding windows (50% overlap)	Explicit Segmentation sliding windows (50% overlap)	Explicit Segmentation sliding windows (50% overlap)	Explicit Segmentation sliding windows (no overlap)	Explicit Segmentation sliding windows (no overlap)
Total of features	53	200	135	160	43

The central characteristic of the data stream models is that streams evolve over time, and algorithms must react to the change. One of the important characteristics of choosing the dataset is that the data changes over time. This feature allows us to validate the importance of using adaptive algorithms. We apply concept change detection algorithms to identify whether the data evolves over time. Figures 6.1, 6.2, 6.3, 6.4, 6.5 shows us

the results for the previously mentioned datasets. The axis-x means the data stream envolving in time. the axis-y is represented by data distribution in a x samples.

The DDM algorithm identified the concept drift of Forth-trace and HAPT. DDM observes that the prediction error of the model increases. It takes this evidence that change has occurred. CUSDM identified the concept drift in WISDM. Probably, it is identified when the mean of the input data deviates significantly from its previous value. The Geometrical Moving Average Detection Method identified concept drift in the Realdisp dataset. The method of assigning weights is based on geometric progression. The latest first observation is assigned with the greatest weight. The previous weights assigned to the observations decreased in geometric progression. The Extrasensory has concept drift from statistical analysis by STEPD. The STEPD means the accuracy of a classifier for recent W examples will equal the overall accuracy from the beginning of the learning if the target concept is stationary. A significant decrease in recent accuracy suggests that the concept is changing.

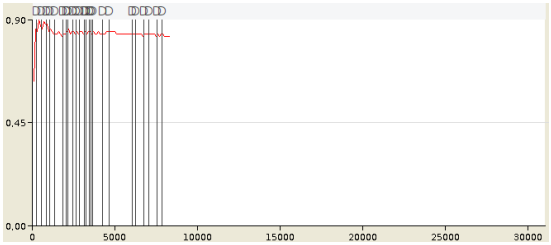

Fig. 6.1. Graphical Representation of concept drift identification in Forth-Trace dataset.

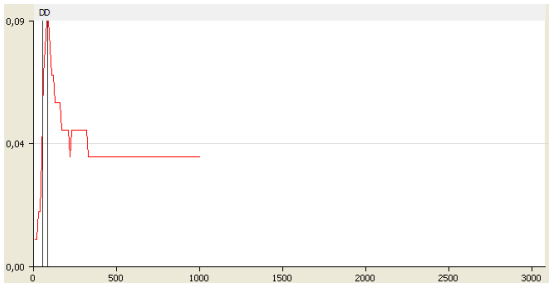

Fig. 6.2. Graphical Representation of concept drift identification in HAPT dataset.

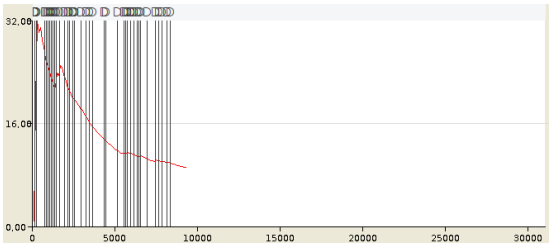

Fig. 6.3. Graphical Representation of concept drift identification in Realdisp dataset.

Fig. 6.4. Graphical Representation of concept drift identification in WISDM dataset.

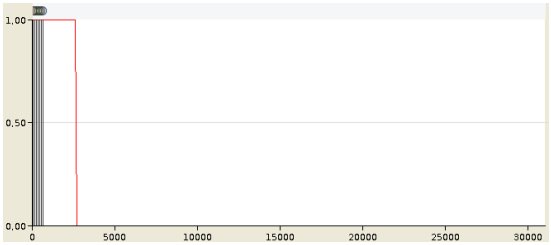

Fig. 6.5. Graphical Representation of concept drift identification in ExtraSensory dataset.

6.2 Experimental Design

Firstly, the data stream appears in raw data format. For data segmentation, the experiment used a fixed sliding window size. In this technique, a window is moved over the data stream to extract a data segment with a fixed size that is then used in subsequent processing steps. Table 6.2 shows all configuration of each dataset.

After, the process of feature extraction is applied. For HAPT, the author used the extracted feature processing based by Anguita et al. [11]). We generated 53 features enrolling in this dataset. For REALDISP, the author used 200 features for construction. Its calculation is based on mean, standard deviation, and variance in x, y, and z. All of this, we have a set of features (named by F) and create $\|F\|$ as a feature. We represented all features as a vector V and a histogram in V_x,V_y,V_z. We gathered all features, creating A_i with 40 features. As this dataset has five devices ($i = 5$) distributed on the body. Thus the author created A_i for each device and gathered all features. So, the author has 200 features. For FORTH-TRACE, we used similar to [71], we gathered: Mean, Median, Standard Deviation, Variance, Root Mean Square, Averaged Derivated, Skewness, Kurtosis, Interquartile Range, Zero Cross Rating, Mean Cross Rating, Pairwise Correlation, Spectral Entropy as feature in our system. For WISDM, we used similar to [80], we gathered the fraction of accelerometer samples. average x, y, and z values in a specific number of records, approximations of the dominant frequency, average absolute deviations from the mean value for each axis, standard deviation, the average of the square roots of the sum of the values of each axis squared; as features in our study. For ExtraSensory, we used similar to [129], we used Accelerometer and Gyroscope (26 features each); 9 statistics of the magnitude signal: mean, standard deviation, third moment, fourth moment, 25th percentile, 50th percentile, 75th percentile, value entropy and time-entropy; 2 autocorrelation features from the magnitude signal; Watch accelerometer (46 features); Location

(17 features); Audio (26 features): Phone State (34 features).

Experimental Setup: We selected four data stream learning algorithms. To deal with different distribution and model construction methods, this work choose the respective leearning algorithm: an optimization-based algorithm, a distance-based algorithm, a probabilistic-based algorithm, and a rules-based (using trees). The algorithms with an online approach are SGD, SAMkNN ($k = 5$), Naive Bayes (online version), and Hoeffding Adaptive Tree. All classification algorithms are implemented on framework MOA [21]. The choice of these algorithms is due to their robustness and performance. The choice of the k value for the distance-based algorithm (SAMkNN) is due to satisfactory results. Finally, the author performed a quantitative analysis of each algorithm, using the following performance evaluation measures: accuracy, Kappa Statistic and precision as evaluation metrics. As the metrics for Error Estimation, the most commonly used in environments with data stream was used prequential test-then-train [55]. Prequential allows the calculation of the error rate of algorithms (accumulated sum of loss functions between the prediction and observed values). This evaluation is best, especially in scenarios in non-stationary data distributions [55]. Processing time is measured in seconds, while memory usage is given in RAM-Hours, where 1 RAM-Hour equals 1 GB of RAM dispended per hour of processing (GB-Hour). We adopted a window size W = 100 to keep accordingly to the defaults provided in the Massive Online Analysis (MOA) framework. The other parameters follow the same case.

We performed these experiments using the Linux Ubuntu 20.04 64-bit operating system and the JDK 1.8.0.201 64-bit Java runtime. The same machine was running both the sensor client nodes and the server. In addition to this, we have the following settings:

- Processor: 2.9 GHz Intel® Xeon(R) CPU E3-1545M v5
- Memory: 16 GB

6.3 Case Study #1: Comparison between static and adaptive models

One of the significant challenges of context-aware HAR systems is to deal with the high heterogeneity of user-profiles (such as height, weight, gender) present in HAR models. This heterogeneity exists due to the lack of data labeled by users; consequentially, there is necessary to use different user data to generalize the models. However, most of these

models do not undergo updates over time, impacting adaptability and inference quality. This study compares models with adaptable strategies in relation to non-adaptive models (models that do not undergo adaptation).

Therefore, based on data stream learning algorithms, we use two strategies: In the first strategy, the classified data is tested, and later, the models are retrained. In the second strategy, data stream learning evolves over time, but the models do not undergo an update process. The model becomes static over time. The goal is to identify the predictive power of models with evolving data over time.

Figure 6.6 shows the design of this study. We separated the datasets by each user $(u_1, u_2, \cdots, u_N)$. All models are trained for all u_i, except u_x, where $u_x \neq u_i$. u_x is our test sample. This scenario is repeated for n users. Figures 6.7, 6.8, 6.9, 6.10, 6.11, 6.12 show the behavior involving all users in relation to adaptive and non-adaptive models. We adopted a window size $W = 500$ to keep track of in all experiments. Each line present in the graphic is a process of evolution of user data in the multi-user model. On the left side, we see the adaptive approach to the models, and on the right, the use of the models in a non-adaptive approach in three metrics. From these results, we can see an evolution curve in most users in adaptive approach. In other words, the adaptive strategy tends to increase the robustness of the algorithm inference process over time. Thus, we used these adaptive models as base classifiers in the next case study. The Evaluation time (cpu seconds) mean is present in Figure 6.13 e 6.14. Another relevant measure is the model cost (RAM-Hours) that in our scenarios is ≈ 0.0.

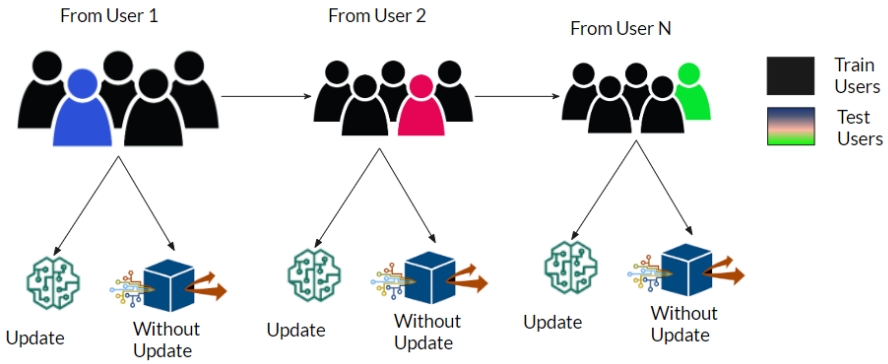

Fig. 6.6. Illustration shows us the process methodology. We have a test user with colorful color and train users with black color. The labelled model as "update"has an updating process evolving over time. The labelled model as "without update"does not have an updating process evolving over time.

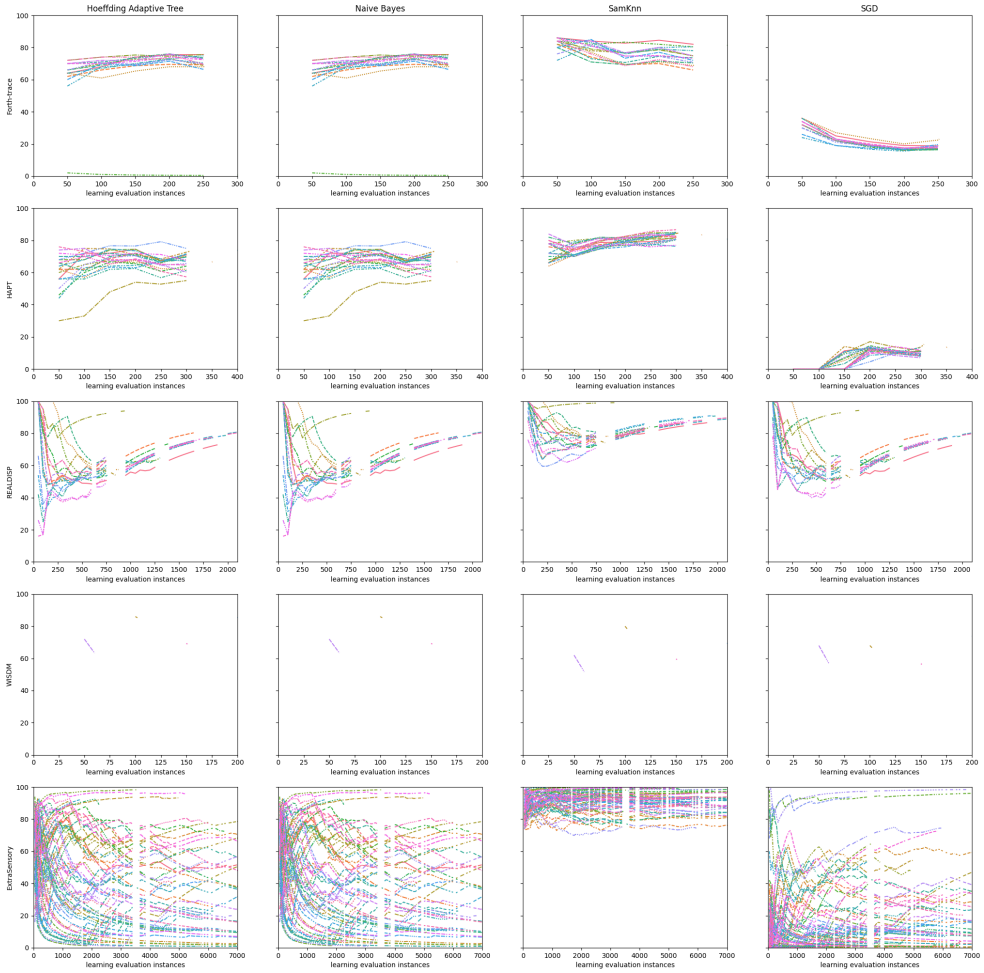

Fig. 6.7. Accuracy evolution involving adaptive models

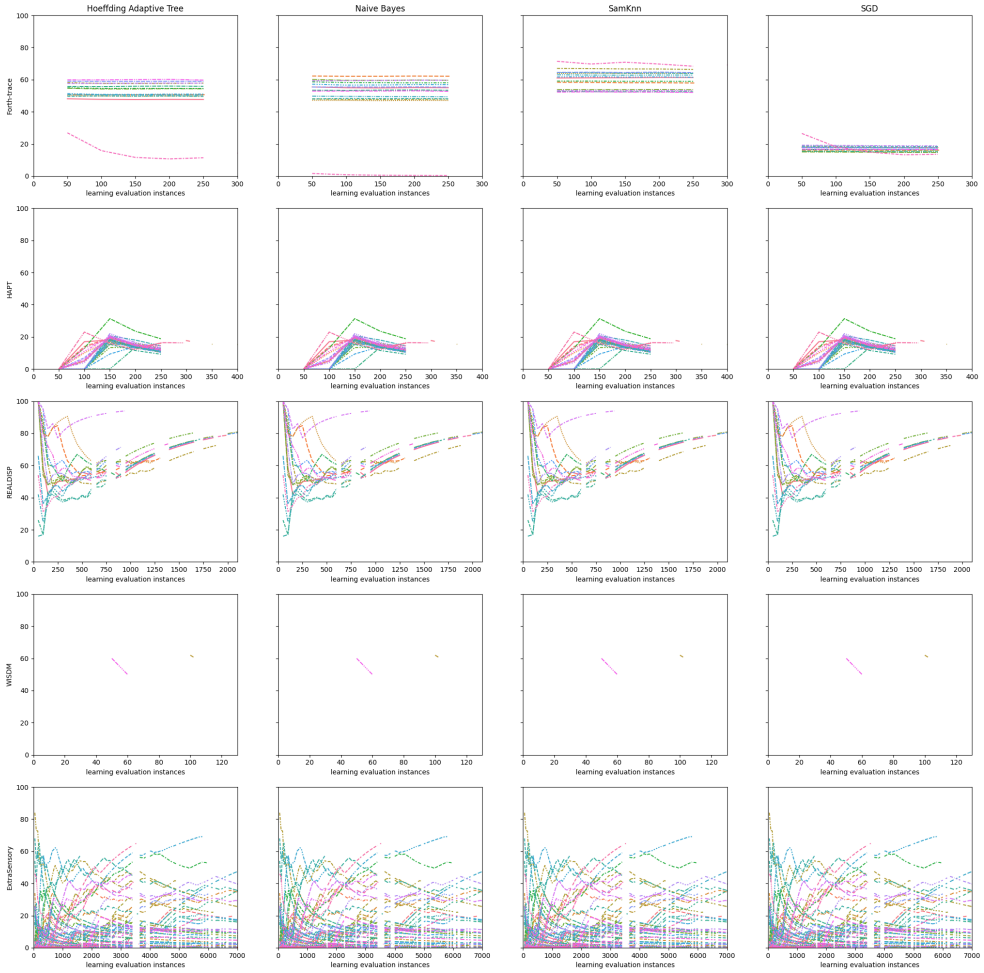

Fig. 6.8. Accuracy evolution involving non-adaptive models

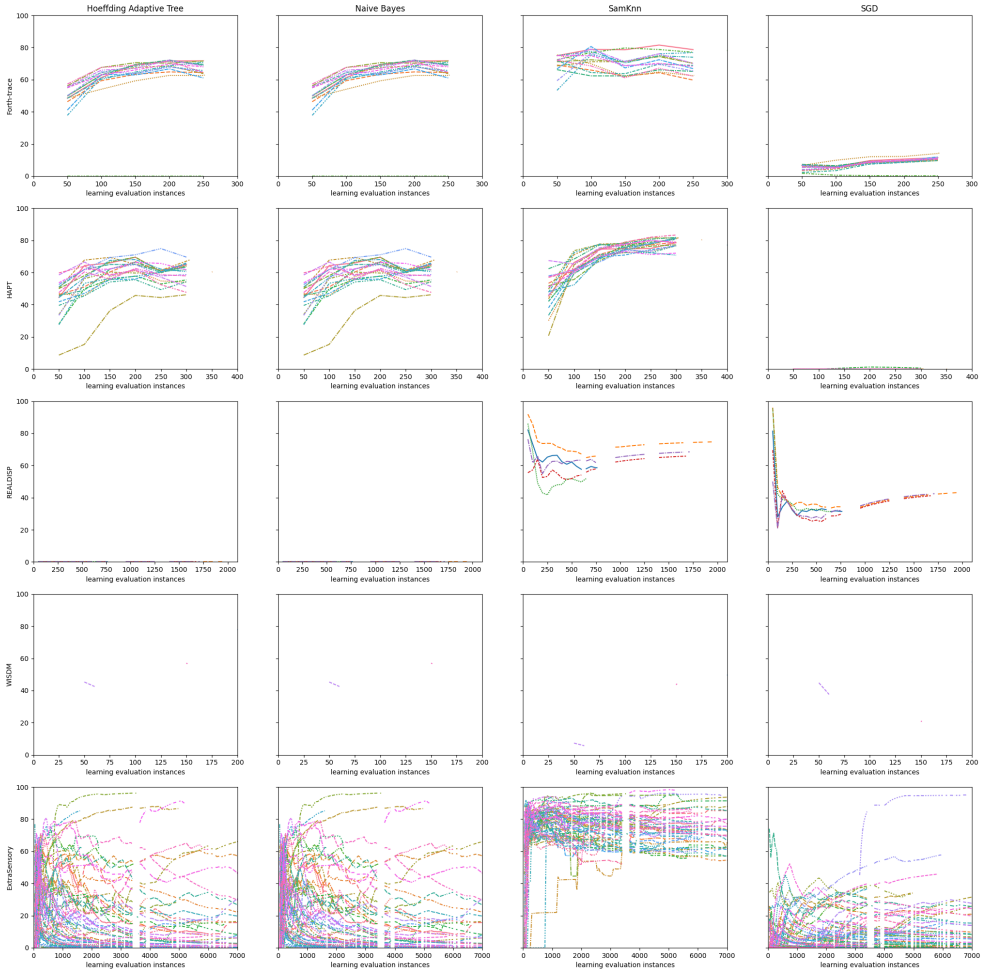

Fig. 6.9. Kappa metric evolution involving adaptive models

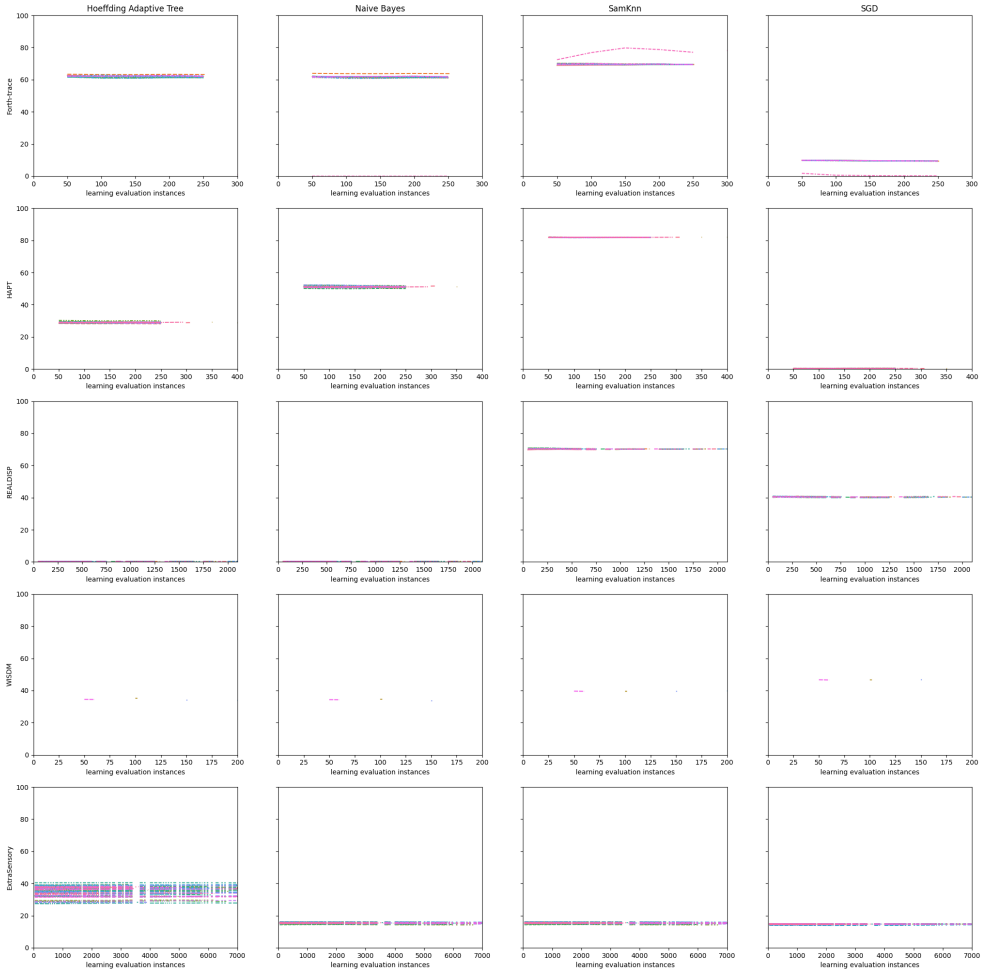

Fig. 6.10. Kappa metric evolution involving non-adaptive models

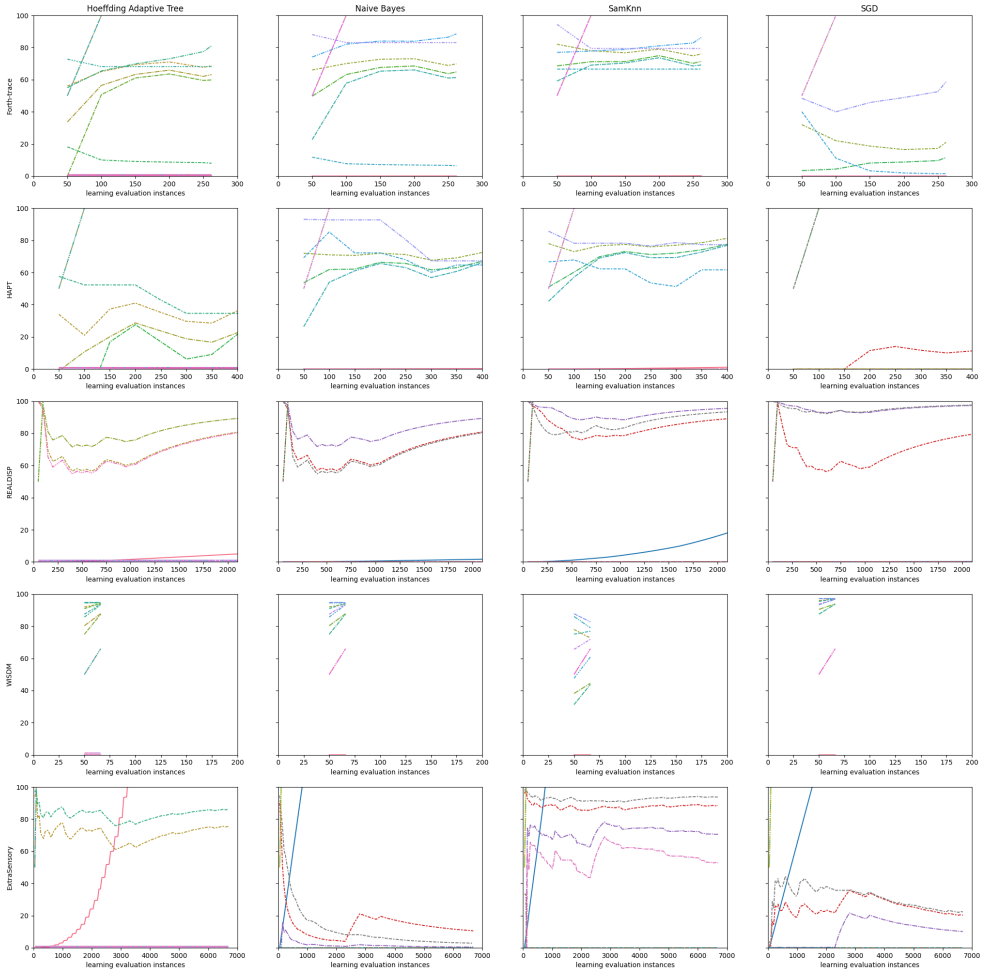

Fig. 6.11. Precision metric evolution involving adaptive models

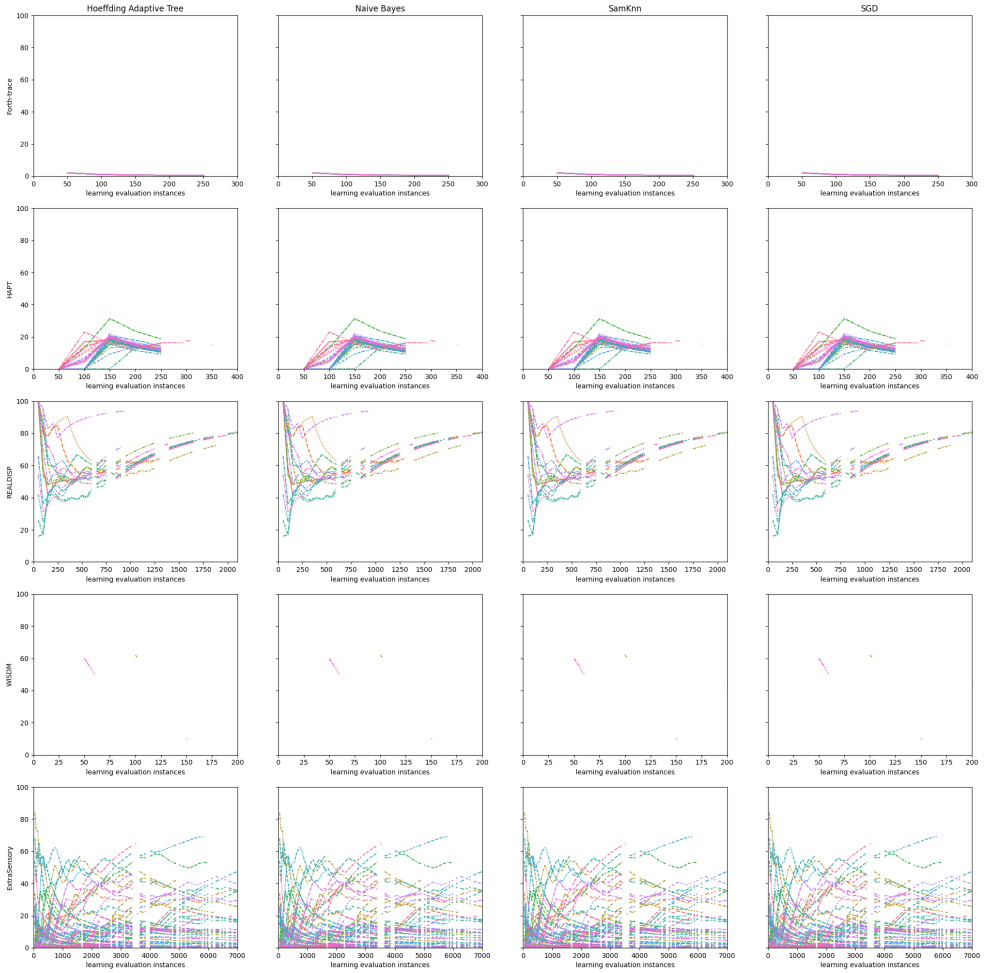

Fig. 6.12. Precision metric evolution involving non-adaptive models

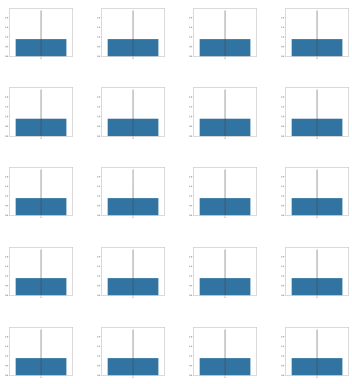

Fig. 6.13. Evaluation time (cpu seconds) mean involving adaptive models

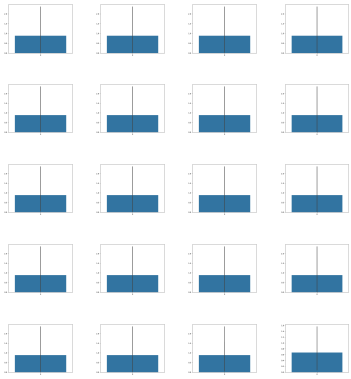

Fig. 6.14. Evaluation time (cpu seconds) mean involving non-adaptive models

Statistical Test: We executed the Friedman Signed-Rank statistical test for multi-

ples domains, one for each metric, as indicated by [69] and [35]. We aimed to compare four adaptive models and four models non-adaptive models. to verify if there is a statistical difference among the algorithms according to accuracy, kappa and recall.

Because of the null hypotheses were rejected on both metrics, we also executed the post-hoc Nemenyi test [69, 35], to verify how the algorithms are ranked and, also, where is the difference among them. Figures 6.15, 6.15 and 6.17 present the results obtained using, respectively, accuracy, kappa and recall metrics. As *CD* (Critical Difference) value is high, for both diagrams, we can conclude that the post-hoc test is powerful enough to detect any significant differences between the algorithms. So, the null hypothesis (they behave similarly) can be rejected with 0.05% significance.

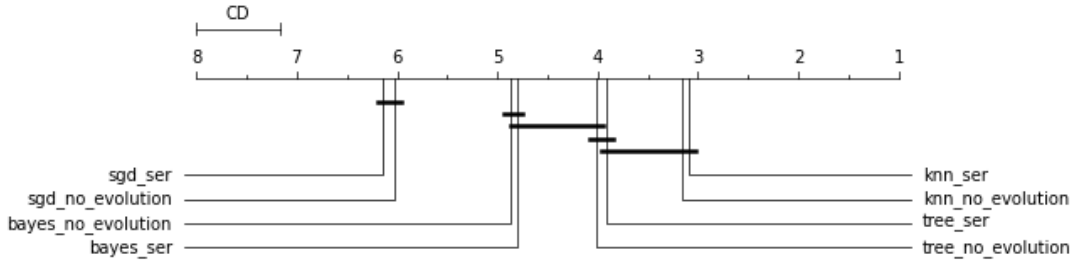

Fig. 6.15. Statistical test for accuracy metric. Models' name with suffix '*_ser*' are adaptive models. Models' name with suffix '*_no_evolution*' are non-adaptive models.

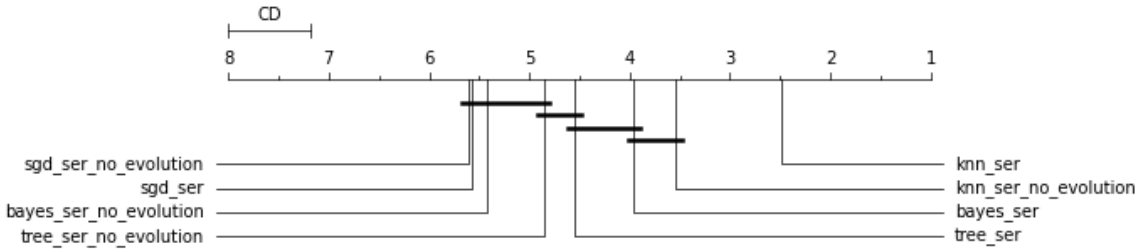

Fig. 6.16. Statistical test for Kappa metric. Models' name with suffix '*_ser*' are adaptive models. Models' name with suffix '*_no_evolution*' are non-adaptive models.

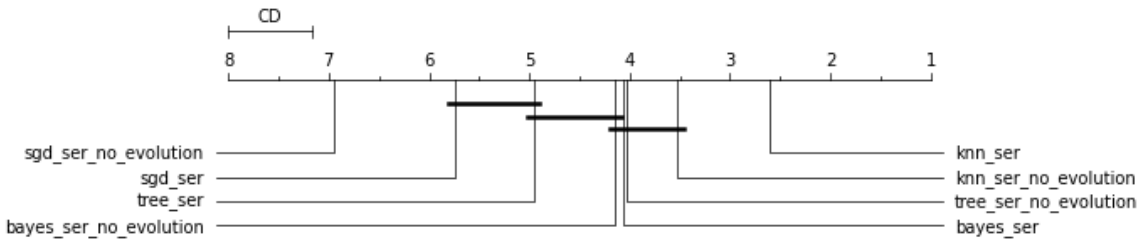

Fig. 6.17. Statistical test for Recall metric. Models' name with suffix '*_ser*' are adaptive models. Models' name with suffix '*_no_evolution*' are non-adaptive models.

6.4 Case Study #2: A Strategy of Recommendation Models for a Respective User

Throughout case study #1, we can consider the following scenario: Imagine a HAR system that, instead of offering a generalist model (which is the most traditional approach in state of the art) and using a strategy in which the implemented model fits a respective user. In other words, the system contains a deployed model where it will be adaptable to the user's characteristics. These systems would create fitted models with fewer data needed and, consequently, would adapt to the user robustly over time.

This second case study address this scenario, offering a model recommendation strategy initially in batch from initial data from the user group similar to the test user. After the recommendation, the model would use an adaptive strategy over time based on test user data. The strategy is divided into 2 phases: Phase 1: Build Learning Model and Phase 2: New Activity Recognition.

6.4.1 Phase 1: Build Learning Model

Initially, supervised learning is applied to labeled data to train and generate the learning model. Algorithm 3 shows Building Learning Model process. First of all, the generated model consists of a set of clusters for a unique user. Each cluster represents one of the labeled activities applied while training the model. The Model (M_o) creates a partition P composed by k clusters (in respect of k activities) in the training data. After, the algorithm analyze the partition characteristics. Partition characteristics are relative criteria that describe all clusters of a unique partition (User data). Characteristics of a partition include the cluster centroids, densities, within-cluster standard deviations, and boundaries (gravitational force). Thus, we used four measures that represent the features of training:

- Distance: In this metric, we adopt the use of distance in cluster centroids. A cluster centroid is a mean n-dimensional instance inside the cluster. One way of extracting information about all clusters is to use the information from the sum of the mean square error of all clusters.

- Density: Each cluster has unique density characteristics, which separate it from other clusters. Cluster density reflects the distribution of data points inside the cluster. It is described by the Formula 6.1, where (m) is the number of points in the cluster, (a) is an n-dimensional data point inside the cluster, and (c_i) is the

centroid cluster. Let S_z the size of the cluster, the average distance(Avg_{dist}) is the within-cluster sum of the distances between cluster's examples and respective cluster centroid divided by the number of examples within the cluster. This equation calculates the density of each cluster in relation to the other remaining clusters. In the end, the used formula in the study is the density of the partition, present in Equation 6.3. This equation calculates C^k_{dens} that sum of C_{dens} of all clusters.

$$C^k_{dens} = \frac{S_z}{Avg_{dist}} \tag{6.1}$$

$$Avg_{dist} = \frac{\sum_{i=1}^{m}(a - c_i)}{m} \tag{6.2}$$

$$C_{dens} = \sum_{i=1}^{k} C^i_{dens} \tag{6.3}$$

- Gravitational force: Based on state-of-the-art [1, 2, 3], the authors discuss that the gravitation force is the natural attraction force between any two objects in the universe. According to Newton's universal law of gravity, the strength of gravitation between two objects is in direct ratio to the product of the masses of the two objects, but in inverse ratio to the square of the distance between them. The law is described in Equation 3.4: Where F_gk is the gravitation between two objects (clusters); G is the constant of universal gravitation; m_1 is the mass of object 1 (size of cluster i); m_2 is the mass of object 2 (size of cluster j); r the distance between the two objects (Euclidean distance between clusters' centroids). According to Equation 6.5, each cluster generates its gravitational force created from its weight. We calculate the sum of the gravitational force of each cluster relative to all clusters in the same partition. The advantage of this strategy indicates that the bigger the candidate's weight, the stronger the gravitational force produced around it. Therefore, the probability of attracting more data objects increases.

$$F_{gk} = G\frac{m_1 \cdot m_2}{r^2} \tag{6.4}$$

$$F_g = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} G\frac{m_i \cdot m_j}{r^2} \tag{6.5}$$

- WICSD (Within Cluster Standard Deviation): This measure considers the cohesion inside each cluster. The standard deviation of n-dimensional points inside the cluster is calculated as the equation 6.6. Where (m) is the number of points in the cluster, a_i is an n-dimensional data point inside the cluster, and (c_k) is the cluster

centroid. Clusters with similar standard deviations are more likely to present the same activity/label. After, based on Equation 6.7, we apply the sum of WISCDM for different clusters. Thus, we created the global representation for each partition P.

$$WICSD_k = \sqrt{\sum_{i=1}^{m} \frac{(a_i - c_k)^2}{m}} \tag{6.6}$$

$$WICSD = \sum_{i=1}^{k} WICSD_i \tag{6.7}$$

Algorithm 3 Building Learning Model

Input: incoming segment stream of the user u_i; number a of activities
Output: A partition (P_f) with k=4 clusters with different number of users.

1: Let X the set of users u_i.
2: **repeat**
3: Set a in k-means (k = a), apply the algorithm in u_i, and create a partition P_i with a activities clusters.
4: Calculate Distance metric from all clusters.
5: Calculate Density metric from all clusters.
6: Calculate Gravitational Force metric from all clusters.
7: Calculate WICSD metric from all clusters.
8: Reserve all previous calculated metrics as features (V) to a instance u_i.
9: Let V associated to u_i and save it in dataset S_{users}.
10: **until** There are no listed user $u_i \in X$.
11: Apply K-means (k=4) in S_{users} and create P_f.

Thus, the feature set is represented by four features of training. This information represents unique information for each user (partition). In the end, we have a dataset with four features, where each row corresponds to a unique user. Finally, after the feature extraction process and feature set building, we applied the K-means algorithm with k=4 for separating groups of similar users (Partition P_f). We chose K-means for its simplicity and a commonly used algorithm in the literature. The choice of the value k is manual, and the value of 4 is due to different characteristics sensitive to the user's profile: age, weight, height, and sex.

Figure 6.18 shows this representation in the training step. We can see a group of users where each user passes individually for the respective workflow. After extracting features about all users, we applied a clustering algorithm to divide similarity groups. The centroids of the respective clusters are essential objects for the model recommendation step for new users.

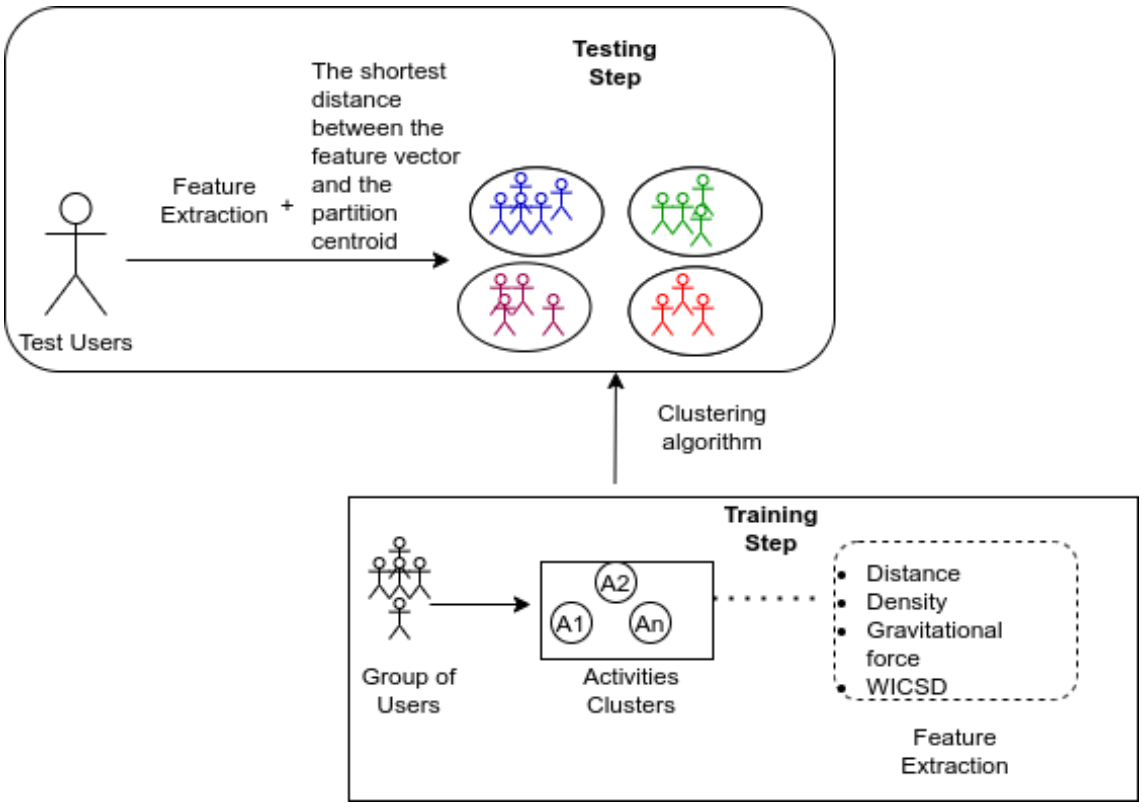

Fig. 6.18. Workflow for building a learning model and new model recommender.

6.4.2 Phase 2: New Model Recommender

As the stream evolves, we assess learning model partitions to predict new clusters' labels. This is handled with a distance between partitions. Partitions are similar if they match based on the distance of the centroid between the Partition candidate and partitions of all clusters. The algorithm checks how similar it is to other partitions in the learning model for each newly formed partition. Algorithm 4 describes and Figure 6.18 shows the process in Testing Step (Model Recommender). We can see the respective sample of the users test, where the data undergoes a feature extraction process. The feature extraction process is similar to the training phase: First, the groups of activities are generated from the clustering process, and values of the features are based on four functions that involve distance, density, gravitational force, and WICSD. From the vector (v) created with these features, the algorithm calculates the shortest distance between v and clusters' centroids ($c_{P_f w}$, where we have $w = [1 \cdots k]$ clusters) in Partition P_f. Clusters' centroids are a mean of the centroid values of each cluster present in the respective Partition P_f.

Algorithm 4 New Model Recommender

Input: incoming segment stream of the test user u_t; number a of activities, A partition composed by all training users (P_f)

Output: A recommendation of the best cluster (with users data for model building) to be addressed to u_t

1: Set a in k-means (k = a), apply the algorithm in u_t, and create a partition P_i with a activities clusters
2: Calculate Distance metric from all clusters.
3: Calculate Density metric from all clusters.
4: Calculate Gravitational Force metric from all clusters.
5: Calculate WICSD metric from all clusters.
6: Reserve all previous calculated metrics as features (V_u) to a instance u_t.
7: Calculate the distance (V_u) of all clusters' centroids ($c_{P_f w}$) that belongs to P_f. The shortest distance between c_u and any $c_{P_f w}$ (centroid of cluster $w \in P_f$) addresses that cluster x is the best cluster to be assigned to u_t.

6.4.3 Results

We present the experiments that we have performed to validate the robustness of the proposed method and discuss its improvements and limitations. Firstly, we randomly select a group of users for testing the experiment in two phases. The first one is updating a pre-trained model with groups of similarity, and the other is updating a multi-user model containing the data of all users in the dataset. The objective is to analyze the growth curve about the following metrics: Accuracy, precision, and recall. Table 6.2 shows us the characteristics of samples randomly selected from the partitions recommended for the respective user. The column *Number of users in Partition Cluster target* explains how many users were used to train the recommender model to respective user (RU), and the respective dataset used in the research.

Table 6.2. Description of selected users in case study #2.

Respective User (RU)	*Number of users in Partition Cluster target (G)*	Dataset
#1	6	Extrasensory
#2	7	Extrasensory
#3	5	HAPT
#4	12	HAPT
#5	5	FORTH-TRACE
#6	2	FORTH-TRACE
#7	5	REALDISP
#8	2	REALDISP
#9	7	WISDM
#10	12	WISDM

To check the robustness of the activities model, we described the trained activities from G individuals in Partition Cluster target. A improvement of the results were obtained

in the majority of the users, while using this strategy of recommendation models, as shown in the figures present in Appendix A. In this study, we did a pairwise comparison between the model trained with characteristics close to a RU ($metric\_user\_single$) and the model trained with data from all users $metric\_user\_multi$, evolving training over time with RU. These models are evaluated and validated based on Accuracy, Kappa, and recall metrics. We can see a significant improvement in most users about the three metrics. However, the main limitation is neither all users are trained for all activities available in the study, consequently, we have unbalanced classes. This situation compromised the quality of the inference.

Statistical Test: We executed the Wilcoxon Signed-Rank statistical test for multiples domains in pair of models, one for each metric, as indicated by [69] and [35]. We aimed to compare the performance of recommended model ($user\_single$) and model multi-user ($user\_multi$), to verify if there is a statistical difference among the algorithms according to accuracy, kappa and recall.

Because of the null hypotheses were rejected on both metrics, we also executed the post-hoc Nemenyi test [69, 35], to verify how the algorithms are ranked and, also, where is the difference among them. Figures 6.19, 6.20 and 6.21 present the results obtained using, respectively, accuracy, kappa and recall metrics. As CD (Critical Difference) value is high, for both diagrams, we can conclude that the post-hoc test is powerful enough to detect any significant differences between the algorithms. So, The null hypothesis (they behave similarly) can be rejected with 0.05% significance.

Fig. 6.19. Statistical test for accuracy metric. Models' name with suffix '$\_ser$' are adaptive models. Models' name with suffix '$\_no\_evolution$' are non-adaptive models.

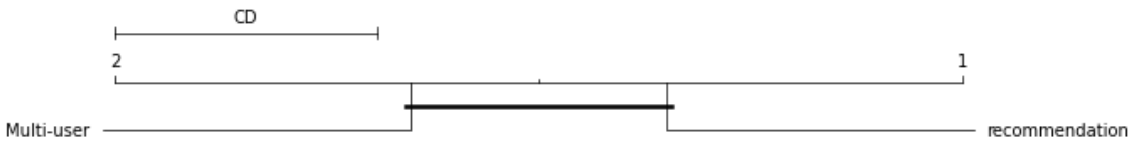

Fig. 6.20. Statistical test for Kappa metric. Models' name with suffix '$\_ser$' are adaptive models. Models' name with suffix '$\_no\_evolution$' are non-adaptive models.

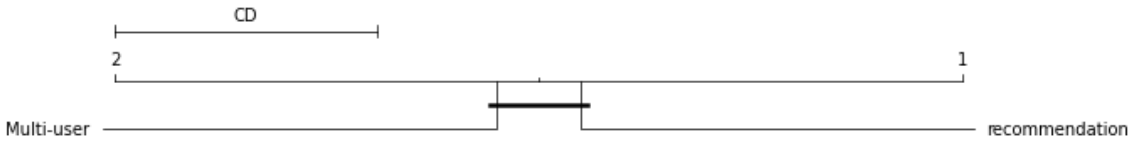

Fig. 6.21. Statistical test for Recall metric. Models' name with suffix '*_ser*' are adaptive models. Models' name with suffix '*_no_evolution*' are non-adaptive models.

6.5 Final Remarks

The need for HAR systems that are more cohesive with each user's profile leads machine learning developers to assemble more general strategies worrying about the diversity of information without actually worrying about the need for a specific user. In addition, updating these models is more expensive and expensive, as they happen in a manual, non-adaptive way.

Therefore, this study analyzed in two directions: 1) a comprehensive analysis of the importance and use of an adaptive strategy over time and 2) a proposal to recommend a model that best suits the profile of a target user. Regarding the first direction, the study shows that adaptive strategies tend to improve more the effectiveness of models than non-adaptive strategies, allowing deliveries of more robust models to users. Regarding the second study, this research proposes a strategy that covers several aspects in relation to the human activities features, according to groups of user-profiles globally. This proposal recommends HAR models through clustering algorithms and strategies based on relative criteria to define user groups' profiles. One of models can be more adjust to a respective user.

Chapter 7

A Novel Approach for Sequential Human Activity Recognition

As mentioned in previous Chapters, traditional HAR approaches stand on the assumption that collected data distribution is stationary. This assumption is commonly violated when data provided by real-time sensors evolve over time. The state-of-the-art proposes different approaches (described in chapter 3), mostly applied in Atomic Human Activity Recognition scenarios. As in AHAR scenarios, adaptability is also present in Complex Human Activity Recognition. The form a user performs a complex activity can be modified over time [140, 113], indicating the need for models to evolve. However, this chapter is focused on Sequential Complex Human Activity Recognition (SCHAR). In this way, there is a lack of algorithms to manage the distribution of non-stationary data with any stream mining approaches. We could not find a stream mining approache used to address SCHAR in the literature using sensor data (e.g accelerators, gyroscope, magnetometer).

The use of State Model-Based and Exemplar-Based SCHAR can be a challenge in a stream environment due to the factors such as temporal and action sequences. Considering these factors, this thesis proposes a new customized algorithm based on data stream mining for SCHAR, called KFC4Stream. KFC4Stream is based on a traditional computational biology algorithm for gene sequencing, known *k-mers filtering Classifier*. This algorithm adopts Exemplar-Based SCHAR as a strategy of knowledge. KFC4Stream algorithm is compared to two batch learning algorithms to SCHAR in the state-of-the-art: Hidden Markov Models (HMM) and K-mers filtering Classifier. We validated our proposal from two experimental analyses with artificial and real datasets. The first experimental scenario used artificially generated datasets to assess the performance of the algorithms considering a set of activities according to different sizes of representing the activities,

i.e., we evaluate the performance of these algorithms when varying the number of atomic activities to represent a SCHAR.

The second experimental scenario considered two real datasets: the Opportunity dataset [28] is a CHAR dataset where the levels between activities have a degree of dependency, e.g., the mid-level gesture annotations are generated automatically from the low-level hand actions. The second dataset is the Milan dataset: this dataset describes sequential ADL where a woman and dog lived for around three months. We used a group of quantitative metrics for evaluating the algorithms: Accuracy, kappa, precision, recall and F1 measures (traditional metrics in Stream Learning and MOA API [21])

7.1 KFC4STREAM: A Sequential Complex Human Activity Recognition Algorithm

This Section describes our proposal SCHAR algorithm within in a data stream learning context. We coined K-mer Filtering Classifier and adapted it to the stream-learning context.

7.1.1 Features of the KFC4STREAM

Most state-of-the-art algorithms could not be applied directly to SCHAR sequence data sets. The proposed algorithm implements all features in sequencial way. KFC4STREAM can integrate the SCHAR feature extraction step (mentioned in Section 2.3.2), allowing on-the-fly mapping of SCHAR sequences into feature vectors. This process simplifies the process of building machine learning-based models for different SCHAR tasks and facilitates the deployment of developed models [1].

The feature extraction is similar to that described in Section 2.3.2. This step has to be performed to map each SCHAR sequence into a feature vector. Figure 7.1 shows an example of workflow. Also, same figure describes the algorithm to calculate the sum of k-mer frequencies scores.

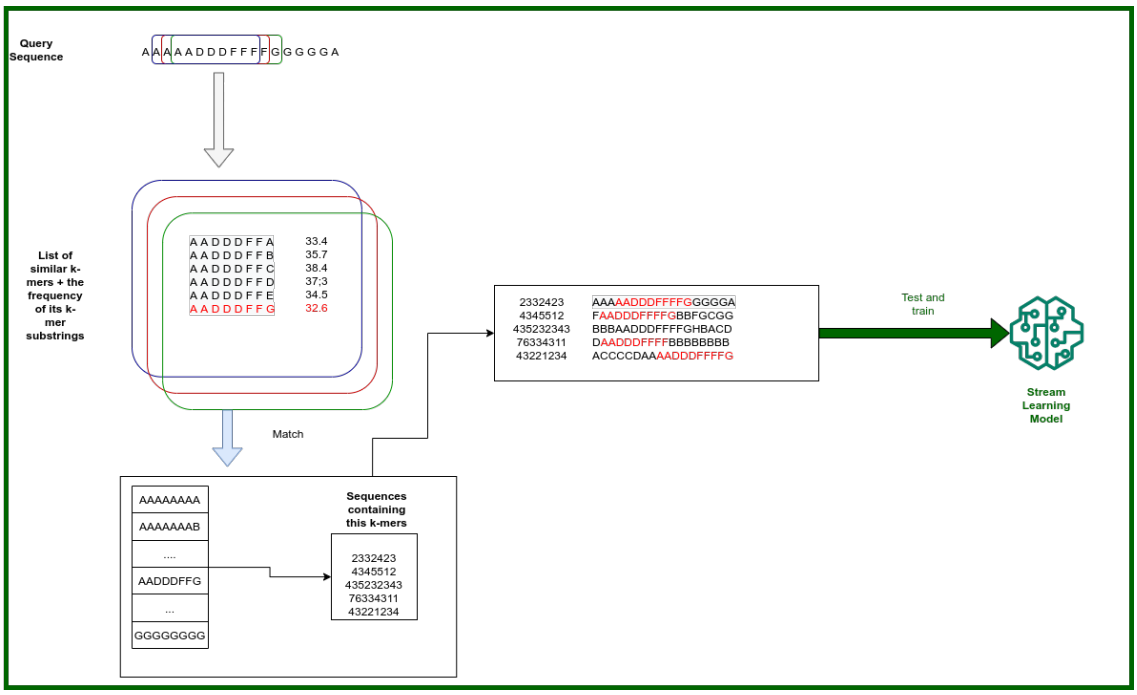

Fig. 7.1. For each k-mer in the query (k=6), a list of similar k-mers and their frequency scores is generated (green frame). For each such k-mer (red), a pointer to a list of representative sequences containing this k-mer is looked up in an array (index table)

.

7.1.2 Training Phase

The Training Phase is similar to described in Section 2.3.2. The training set S is the correspondence between the k-mer frequencies of training sequences (atomic activities) and their groups. The feature vector $Fv(s)$ for an input sequence s (atomic activities) was constructed from the number of occurrences of all k possible k-mers (given Σ), divided by the total length of N. Next, we processed the feature vectors as stream for more effective of the Stream Learning Classifiers (base classifier). Figure 7.1 shows final step of training. The table look-up and frequency of sequences constantly updates over time.

7.1.3 Inference

The process of inference starts in K-mer filtering step. This step shows how to simultaneously extract features from sequence data (atomic activities) and test a model. As KFC4STREAM is based on K-mer filtered classifier, the algorithm uses a stream learning algorithm and a filter to be applied on the fly before feeding the data to the predictor—class for running a stream learning algorithm (base classifier) on data passed through K-mer filtering. Like the classifier, the structure of the filter is based exclusively

on the training data, and the filter will process test instances without changing their structure. Thus, it is important to determine which stream learning classifier would be the most effective for classifying sequences, using their respective k-mer frequencies as feature vectors (numerical representations).

7.1.4 Implementation

K-mer representation is implemented as part of the Genome Annotation Toolkit (Gennotate) [81]. Gennotate is an extension of WEKA [63], a widely used machine learning workbench supporting many standard machine learning algorithms. Most of these algorithms could not be applied directly to SCHAR sequence data sets. Developers may pre-process their data for feature extraction and then apply MOA implemented algorithms to the data set in its numerical representation. Gennotate can integrate the SCHAR feature extraction step into WEKA and allow on-the-fly mapping of SCHAR sequences into feature vectors. This simplifies the process of building machine learning-based models for different SCHAR tasks and facilitates the deployment of developed models [1].

7.2 Environment Development Setup

This section guides through the basic description involved in the architecture. Our Environment used contextnet [41]. In its structure, there are a mobile node and stationary node of the SDDL core. This stationary node will play the role of a server processing node, capable of processing application messages from the logic node (LN) according to AHAR application (Context Life) specific logic, and sending messages back to the (LN).

Our sample application is composed of two components: a client node and a stationary node, as illustrated in Figure 7.2. The LN creates a stream that is sent to the SDDL core. Our processing node receives this stream and executes SCHAR inference, keeps track of the stream, and prints the classification output.

As a simplification, the entire application system, composed of the mobile client, gateway, and processing node, runs locally on a single machine. However, it can be easily modified to run on distributed machines (which will be a future step). The example illustrates the asynchronous send and receives primitives for both the client and core type of nodes.

We performed these experiments using the Linux Ubuntu 20.04 64-bit operating sys-

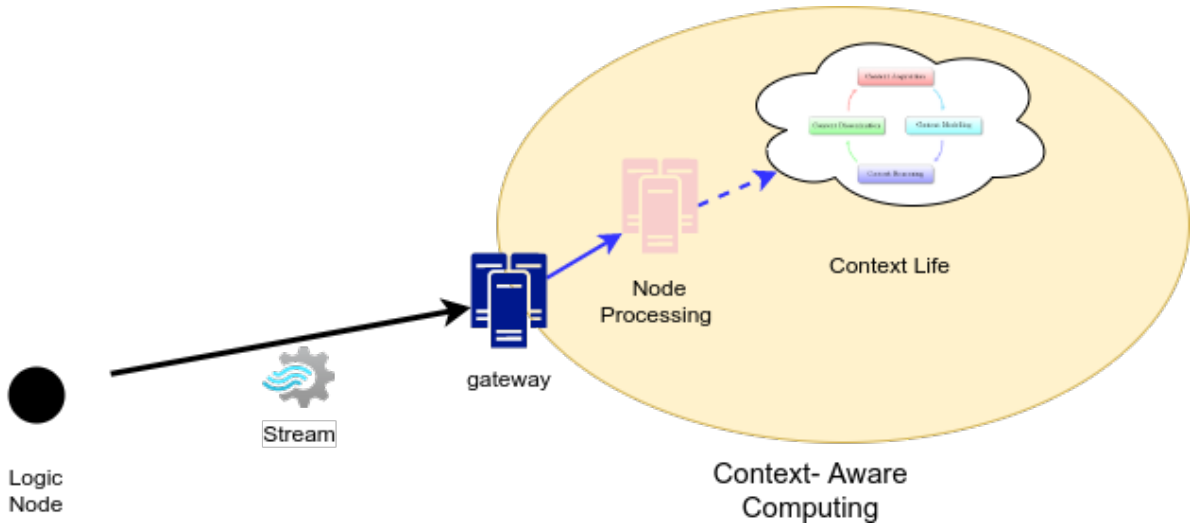

Fig. 7.2. Illustration about the used environment, based on Contextnet

tem and the JDK 1.8.0.201 64-bit Java runtime. The same machine was running both the sensor client nodes and the server. In addition to this, we have the following settings:

- Processor: 2.9 GHz Intel® Xeon(R) CPU E3-1545M v5
- Memory: 16 GB

7.3 Case Study #1: Artificial Dataset

This section presents the experiments to evaluate KFC4Stream performance compared to HMM, K-mer Filtered Classifier. We generated SCHAR activities from datasets mentioned before. Due to few SCHAR datasets, this thesis proposes three artificial datasets. We continue to use the *HAPT*, *REALDISP* and *Forth-trace* datasets for benchmark tests and creation of SCHAR datasets . We created HAPT_complex, Realdisp_complex, and Forth-trace_complex, which are SCHAR artificial complex datasets based on the mentioned datasets. Figure 7.3 shows the workflow to build the SCHAR datasets. The process is divided into three steps. The first step builds AHAR activities, which are base activities for building the SCHAR activity. The study gathers different stream learning classifiers and tests them to define which one should be a base classifier. The base classifier is the standard classifier for building essential AHAR activities in a sequential complex activity. The second step builds the activity buffer. According to the characteristics of segmentation and transition between two activities, the ordering of sequential complex activity and the number of activities (buffer) that represents a sequential activity are defined.

The third step is the final step, where recognizing sequential complex human activities is carried out.

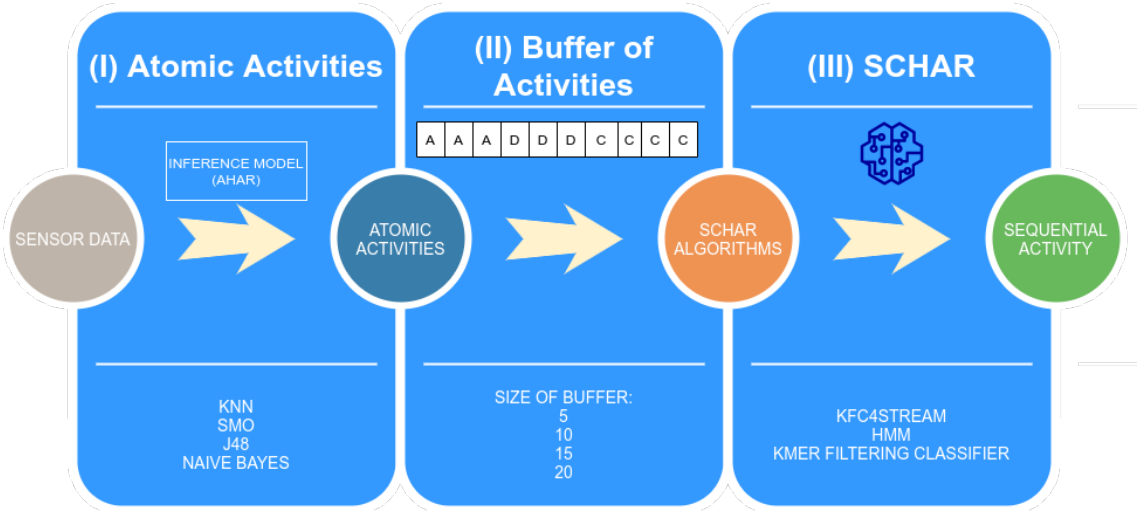

Fig. 7.3. A workflow of this case study. The author builds since sensor data step until the SCHAR inference.

7.3.1 Step I – Generating AHAR

In Step (I), we created a HAR, respecting the temporal order for each sample. We applied this data to the Atomic HAR process to generate atomic activities. Thus, we applied the process described in Section 3.1. We simulated a source that sends packages to the AHAR process (via ContextNet project). Moreover, we apply the SMOTE algorithm [42] to generate a larger population of samples. This SMOTE was applied to raw dataset samples, preserving order by activities. The algorithm was applied to the sliding windows, respecting the sequence of existing data. Table 7.1 shows all the characteristics of preprocessing and sliding window. Overall, 40% aleatory samples were used for training under all AHAR datasets. During this Step, we use the prequential evaluation method. The sampling frequency is 100 objects AHAR sliding window samples by stream window.

For classification, we used the used previous algorithms in stream learning. Each algorithm is based on different methods:

- SamKNN (lazy-based): k =3; limit of samples = 5000; minSTMSize = 50; Relative LTM Size = 0.4.
- Naive Bayes Online (probabilistic-based).

Table 7.1. Description of datasets used in Chapter 7.

Datasets	*HAPT*	*REALDISP*	*Forth-trace*
Total of activities	12	33	16
Total of users	30	17	15
Window size (segments)	128 (≈ 2.56s)	200 (≈ 4s)	10 (≈ 2s)
Total of segments (number of samples)	10929 (HAPT)	9213	3900
Type of sliding window	Explicit Segmentation sliding windows (50% overlap)	Explicit Segmentation sliding windows (50% overlap)	Explicit Segmentation sliding windows (50% overlap)
Total of features	53	200	135

- Hoeffding Adaptative Tree (rules-based): Grace Period = 200; Split Criteria = Info Gain.
- SGD Multi-Class (Optimization-based): Loss Function = Hinge; Lambda Regularization = 0.005; Learning Rate = 0.005.

Experimental Protocol: All metrics are computed accordingly to the Prequential test-then-train procedure. We adopted a window size $W = 500$ to keep track of in all experiments. All remaining parameters were set accordingly to the defaults provided in the Massive Online Analysis (MOA) framework.

Figures 7.4, 7.5, and 7.6 show the results established on different accuracy, precision, and kappa metrics (Tradicional metric in Stream Learning and MOA API [21]), respectively.

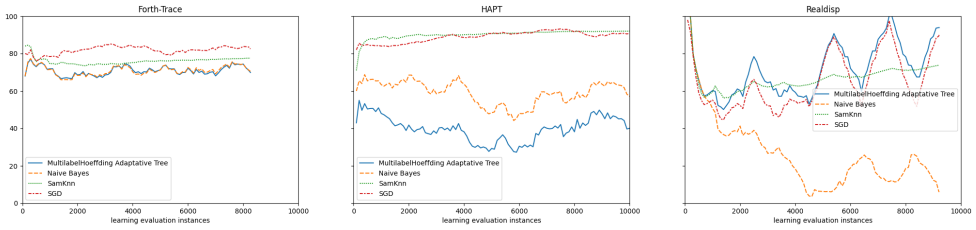

Fig. 7.4. Accuracy evolution involving adaptive models

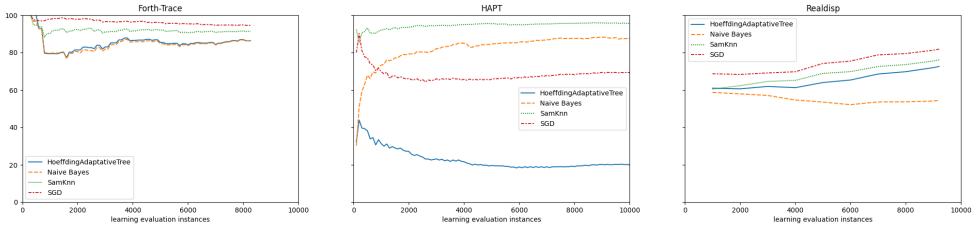

Fig. 7.5. Precision evolution involving non-adaptive models

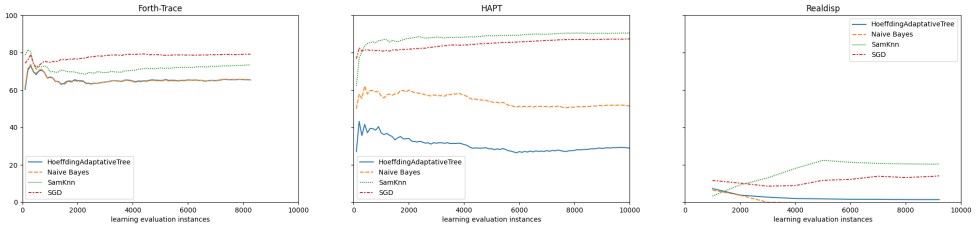

Fig. 7.6. Kappa metric evolution involving adaptive models

We could observe that SamKnn has the best predictive power among all algorithms in Hapt and Realdisp. SGD has the best result in the Forth-Trace dataset. Thus, we adopt SamKnn as the base classifier for AHAR to HAPT_Complex and Realdisp_Complex. SGD is the base classifier for Forth-Trace_complex.

7.3.2 Step II - Phase of Buffer of Activities

In this study, we have non-transition and transition atomic activities. For a close-to-real simulation, it is necessary to calculate how much information is needed from each activity to compose/form a sequential activity. We considered the approach published in [60] used the emerging pattern technique to perform the segmentation concurrently and recognize the SCHAR activities. In this approach, the segmentation is performed recursively. That is, by detecting an activity that has commenced at time t, with the average span of L, the duration from t to t + L is analyzed and considered a segment. Then, the next activity starts from time t + L.

Thus, we need to analyze the impact of more activities in the same sequential activity buffer. It means that each dataset has data in different buffer sizes. Figure 7.7 shows us examples of buffers of different sizes. Thus, HAPT_complex has 14 of size buffer, Forth-trace_complex has 12 of size buffer, and Realdisp has 22 of size buffer. This information allows for variability in how much information a buffer can store to represent an activity. The choice of this value is due to a range capable of carrying a comfortable amount of

SCHAR activities without compromising the classification quality. Because large and super small buffer values cannot capture the amount of information necessary for that activity, on the other hand, a high buffer value leaves the quality of the respective SCHAR down, as it may collect information that does not represent the respective activity. In Figure, The boxes with white colors represent the sequence. The boxes with orange colors represent other sequences over time.

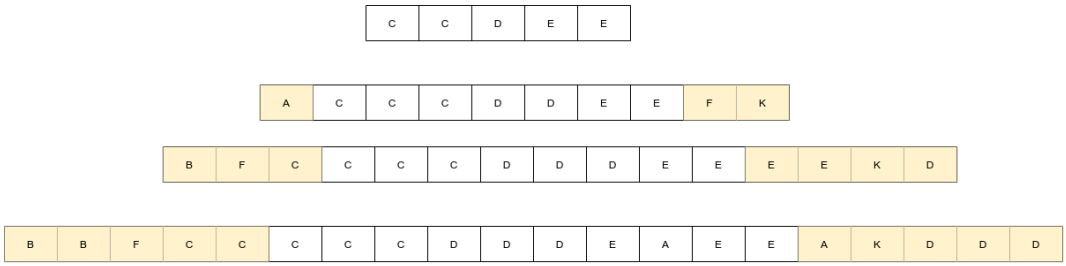

Fig. 7.7. Example of the Buffer of activities.

Table 7.2. Information about SCHAR datasets used in this study.

Dataset	length of a sequence (N)	Number of letters in alphabetic (Σ)
Forth-trace_complex	12	17
HAPT_complex	14	12
Realdisp_complex	22	33
Opportunity	89	31
Milan	50	32

7.3.3 Step III - Phase of SCHAR

In this step, the inference process is carried out to recognize sequential complex activities. For this study, we compare the evolution of KFC4STREAM versus Kmer Filtered Classifier and HMM. We adopt HMM because of its predictive potential in problems related to alignment and sequencing, as mentioned in Chapter 2.3. As HMM and Kmer Filtered Classifier are batch-based, we adopt the use of a meta classifier, known as WEKAClassifier (Annex A) that buffers a chunk of the stream and the batch classifier learns on this chunk.

Experimental Protocol: All metrics are computed accordingly to the Prequential test-then-train procedure. Processing time is measured in seconds, while memory usage is given in RAM-Hours, where 1 RAM-Hour equals 1 GB of RAM used per hour of processing (GB-Hour). We adopted a window size $W = 100$ to keep track of in all experiments. WekaClassifier has the same value for the size of first window ($= 100$) and W. For KFC4STREAM and K-mer Filtered Classifier, we use $k = 3$. Table 7.2 shows all

information related length of sequence and number of letter of alphabetic (Σ, number of atomic activities) of each dataset.

As parameters, K-mer Filtered Classifier uses Naive Bayes as a classifier base. Naive Bayes is commonly used in sequential analyses and alignment analyses [1, 9, 85].

Figures 7.8, 7.9, 7.10, 7.11 e 7.12 show the results based on different accuracy, precision, kappa, recall and f1 metrics, respectively. We have a Time taken to build model: For HAPT_complex: 0.92s in K-mer Filtered Classifier, 0.81s in KFC4STREAM, 0.02s in HMM. For Forth-trace_complex: 5.18s in K-mer Filtered Classifier, 5.28s in KFC4STREAM, 0.05s in HMM. In Realdisp_complex: 299.38s in K-mer Filtered Classifier, 299.78s in KFC4STREAM, 0.03s in HMM.

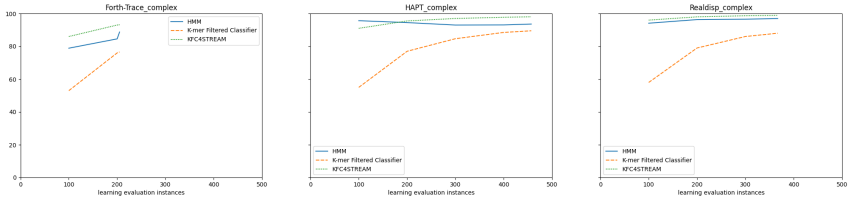

Fig. 7.8. Accuracy metric involving three algorithms in three datasets.

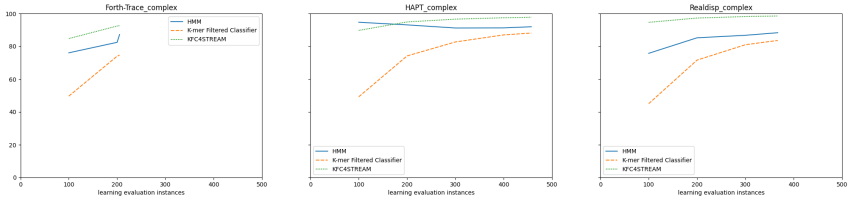

Fig. 7.9. Kappa metric involving three algorithms in three datasets.

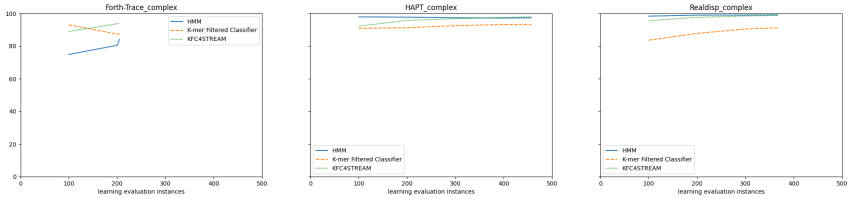

Fig. 7.10. Precision metric involving three algorithms in three datasets.

The results of the experiments are presented using several perspectives to observe the performance of the prequential variations from different situations related to the process of learning in data streams. We can see the most cases covering different metrics; KFCSTREAM shows better results and more stability in relation to all metrics over time. Initially, HMM developed good results; however, evolving the time, KFC4STREAM shows

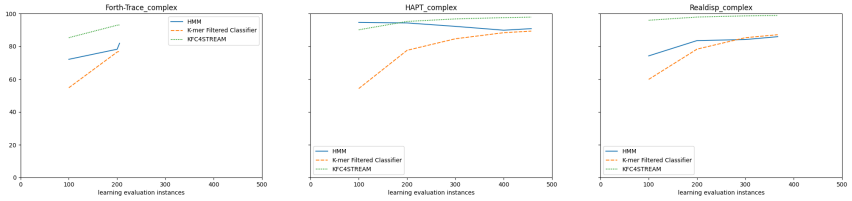

Fig. 7.11. Recall metric involving three algorithms in three datasets.

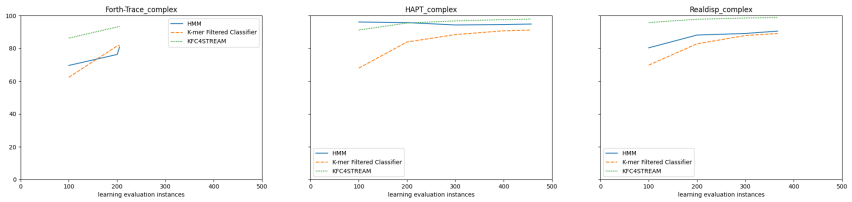

Fig. 7.12. F1 metric involving three algorithms in three datasets.

the best results. The most cases, K-mer Filtered Classifier shows the worst results from initial evaluation until the final process.

7.4 Case Study #2: Real Datasets

For this scenario, we show an exploratory study using two real datasets which use SCHAR. The respective datasets are:

- **Oportunity Dataset**: The OPPORTUNITY Dataset [28] for Human Activity Recognition from Wearable, Object, and Ambient Sensors is a dataset devised to benchmark human activity recognition algorithms.
- **Milan dataset**: This dataset describes sensor events collected in the WSU smart apartment testbed while two people execute their daily activities. Activities are represented by a sequence of events on some devices present in an apartment. [31].

7.4.1 Oportunity Dataset

This dataset [28] provides realistic scenarios of activities with a variety of sensing modalities. The number of instances for different activities is unbalanced. The data is collected by four different users performing daily morning activities. A total of 72 sensors in 10 modalities are deployed for data collection. The sensors are located on objects in the environment and the subject's body. Environmental sensors are attached to fixed places in a studio flat with a kitchen, deckchair, and outdoor access where subjects performed

Table 7.3. Time execution and Model cost in three artificially dataset.

Dataset	Algorithms	evaluation time (cpu seconds)	model cost (RAM-Hours)
Forth-Trace_complex	HMM	0.14	≈ 0.0
	K-mer Filtered Classifier	2.78231749	≈ 0.0
	KFC4STREAM	0.0913	≈ 0.0
HAPT_complex	HMM	0.12	≈ 0.0
	K-mer Filtered Classifier	1.59242633	≈ 0.0
	KFC4STREAM	0.217800914	≈ 0.0
Realdisp_complex	HMM	3.22	≈ 0.0
	K-mer Filtered Classifier	17.116238187	≈ 0.0
	KFC4STREAM	0.124645981	≈ 0.0

activities. The collected data consists of annotated complex, interleaved, and hierarchical naturalistic activities, with a vast number of atomic activities executions (mode of locomotion, postures of left and right arm and upper body posture) about 30,000.

Figure 7.13 shows the schematic example according to the dataset. This representation becomes important for how we represent our data. For example, Activity I and Activity II are represented in the article as middle activities. This activity is simultaneously composed of the atomic activities *Mode of locomotion*, *Right Arm Posture*, *Left Arm Posture*, *Upper Body Posture*. In our SCHAR scenario, the Activities $(I, II, \cdots, n)$ are atomic activities for the studied algorithms. The dataset also consists of data for the four high-level (complex) activities of "Relaxing", "Coffee time", "Early morning", and "Sandwich time" enrolled with 64 atomic Activities. The sampling rate for the OPPORTUNITY dataset is 20 Hz. We set the chunk size for all experiments to 50 instances (2.5 seconds) The representation is similar to Figure 7.7 where each box represents the atomic activities.

First, an analysis was made of the independence of the variables present in the dataset to define the atomic activities. This process was done exclusively in a combination of two features: middle level and locomotion activities. In our interpretation, they represent activities that make up a SCHAR dataset due to the description of the authors of the dataset [28], where the synchronization gesture between middle activities and locomotion are independent. Thus, we allow us to get the best power prediction in our test classifiers.

For the representation of the sequential dataset, we split the buffer of activities with the size of 50 activities in sequential form. This representation is similar to Figure 7.7. The atomic activities (Activity I, Activity II, $\cdots$, Activity n) is represented by the middle activities. The Complex activities means High level activities. We have the following activities:

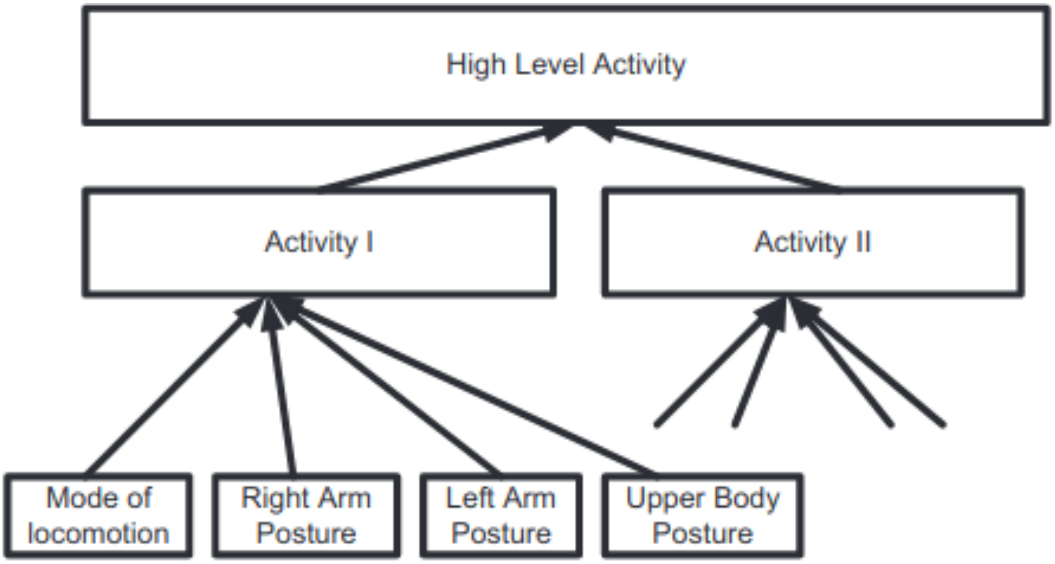

Fig. 7.13. Schematic Example of an activity hierarchy. Mode of locomotion, postures of left and right arm and upper body posture distinguish activity I. Activities I and II together form the high level activity. Description updated of [28].

- A pair P_a of two atomic Activities <middle activities, locomotion>. List of middle activities: Open Door 1; Open Door 2; Close Door 1; Close Door 2; Open Fridge; Close Fridge; Open Dishwasher; Close Dishwasher; Open Drawer 1; Close Drawer 1; Open Drawer 2; Close Drawer 2; Open Drawer 3; Close Drawer 3; Clean Table; Drink from Cup; Toggle Switch; List of locomotion activities: Stand, Walk, Sit and Lie. For two atomic activities.

- Complex Activities: Relaxing; Coffee time; Early morning; Cleanup; Sandwich time; Unlock

All activities P_a and complex activities have a neutral state where the person does nothing or something the model can not predict. We have 68 available atomic activities; however, we just used 31 because, with the more extended alphabet, the algorithm extrapolated the memory cost during model training.

For the first case, it is first necessary to find the initial atomic activities, representing the complex activity. All details present in this data are presented in Chavarriaga et al. [28]. As in case study #1, four data stream learning algorithms were used as a base classifier according to different methods. Each algorithm is based on different methods:

- SamKNN (lazy-based): k =3; limit of samples = 5000; minSTMSize = 50; Relative LTM Size = 0.4

- Naive Bayes Online (probabilistic-based)
- Hoeffding Adaptative Tree (rules-based): Grace Period = 200; Split Criteria = Info Gain
- SGD Multi-Class (Optimization-based): Loss Function = Hinge; Lambda Regularization = 0.005; Learning Rate = 0.005

Experimental Protocol: All metrics are computed accordingly to the Prequential test-then-train procedure. Processing time is measured in seconds, while memory usage is given in RAM-Hours, where 1 RAM-Hour equals 1 GB of RAM used per hour of processing (GB-Hour). We adopted a window size $W = 1000$ to keep track of in all experiments. WekaClassifier has the same value for the size of first window (= 1000) and W. We have a Time taken to build model: 96.56s in K-mer Filtered Classifier, 95.56s in KFC4STREAM, 0.29s in HMM. Table 7.2 shows all information related length of sequence and number of letter of alphabetic (Σ, number of atomic activities) of Oportunity dataset.

Figures 7.14, 7.15, 7.16, 7.17, 7.18 show the results. The SGD Algorithm showed better results in most of the metrics. Moreover, the evolution curve is more stable, where it represents a possible convergence. Given this, SGD served as a base classifier for AHAR.

For the recognition process of complex activities, we developed the same methodology as in the case study #1. We compared K-mer Filtered Classifier, HMM, and KFC4Stream. We chose k=3 to K-mer Filtered Classifier and KFC4Stream due to be a default value in all start-of-the-art mentioned before.

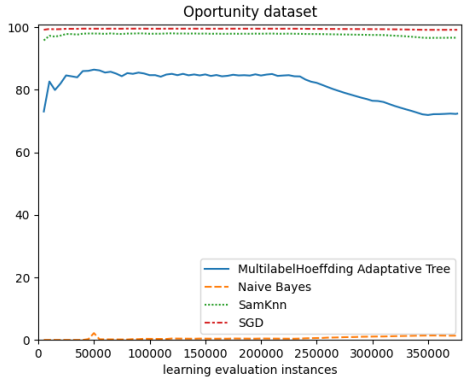

Fig. 7.14. Accuracy evolution involving AHAR Classifiers

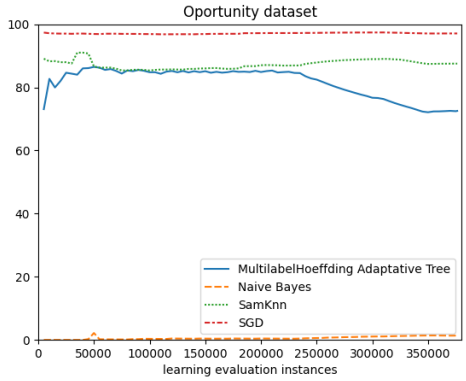

Fig. 7.15. Precision evolution involving AHAR Classifiers

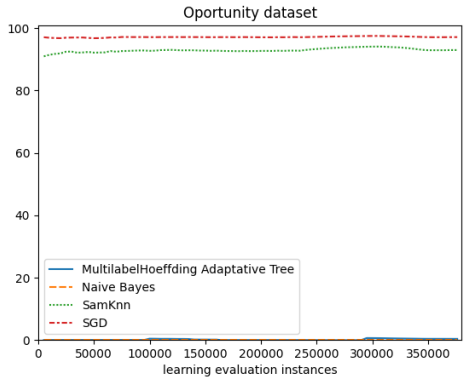

Fig. 7.16. Kappa evolution involving AHAR Classifiers

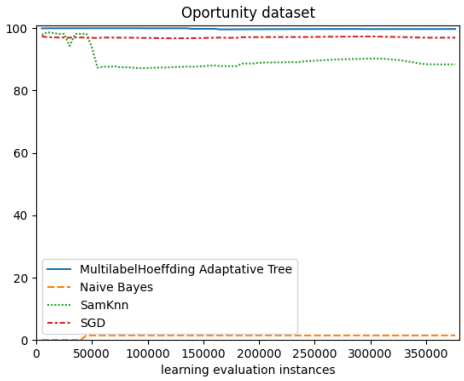

Fig. 7.17. Recall evolution involving AHAR Classifiers

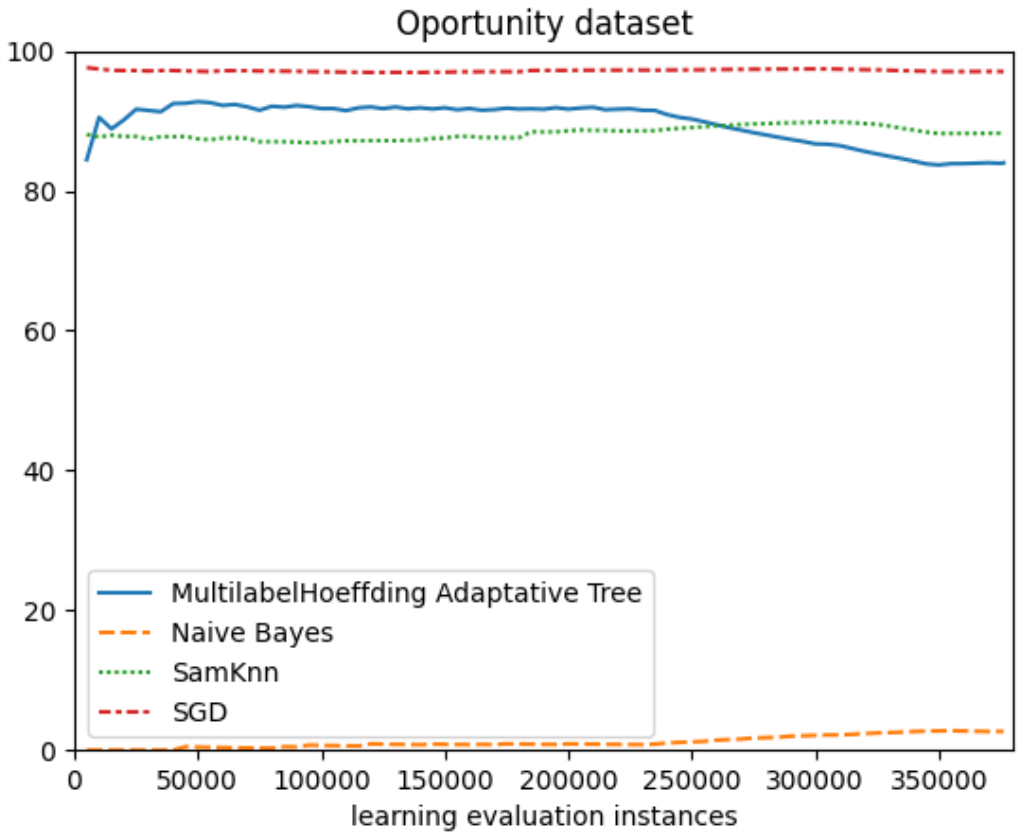

Fig. 7.18. F1 evolution involving AHAR Classifiers.

In SCHAR scenarios, we move forward to test KFC4STREAM on real life activity recognition data streams presented in this dataset. We used commonly metrics tested in previous study. We compare KFC4Stream versus HMM and K-mer Filtered Classifier. Figures 7.19,7.20 , 7.21, 7.22, 7.23 and Table 7.4 show the results.

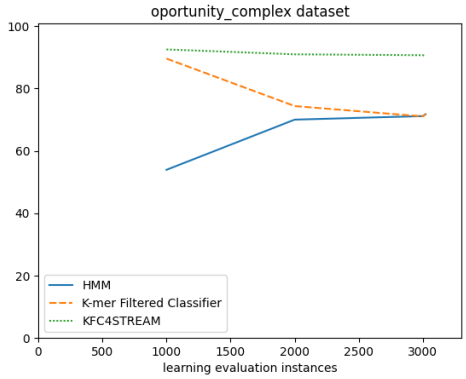

Fig. 7.19. Accuracy evolution involving SCHAR Classifiers

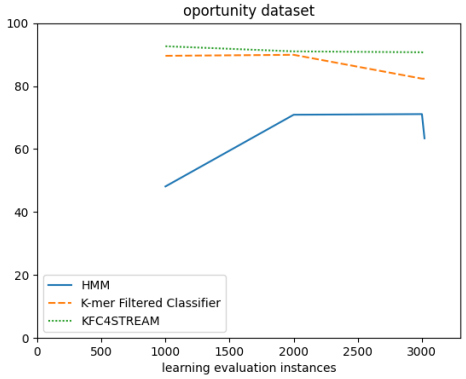

Fig. 7.20. Precision evolution involving SCHAR Classifiers

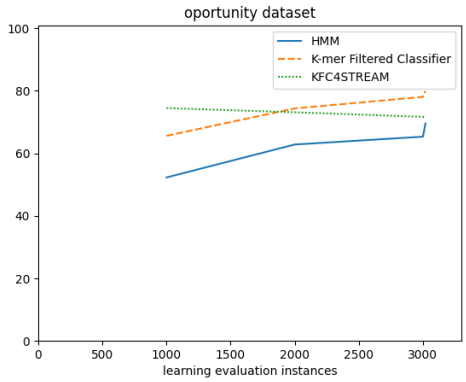

Fig. 7.21. Kappa evolution involving SCHAR Classifiers

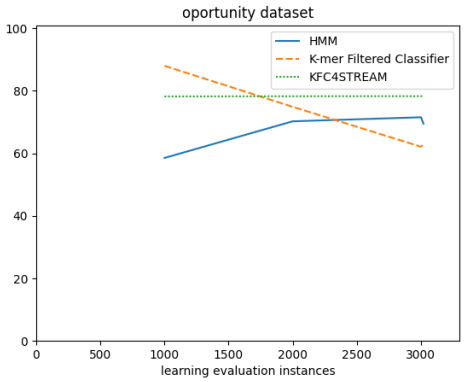

Fig. 7.22. Recall evolution involving SCHAR Classifiers

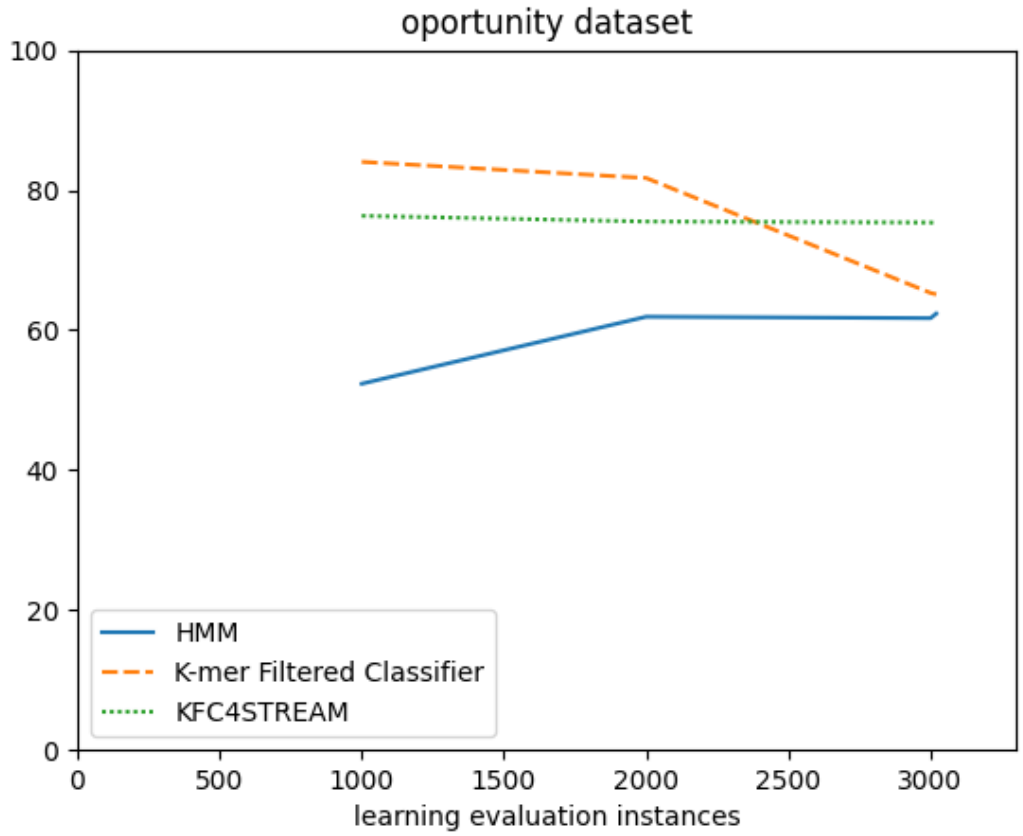

Fig. 7.23. F1 evolution involving SCHAR Classifiers.

Table 7.4. Evaluation time and model cost to Opportunity dataset.

Algorithms	evaluation time (cpu seconds)	model cost (RAM-Hours)
HMM	0.04	≈ 0.0
K-mer Filtering Classifier	9.047	≈ 0.0
KFC4STREAM	3.904	≈ 0.0

In this SCHAR scenario, KFC4STREAM can handle better than other algorithms on most metrics, except for Kappa Statistic. In addition, we can observe more stability and fast convergence to the evolution curve of the algorithm. HMM, in most metrics, has an evolution in its curve. K-mer Filtered Classifier remains mostly a decreasing curve of metrics evolution. Kappa metric was an exception. Due to the Kappa metric being a more sensitive measure for quantifying the predictive performance, we assume the majority class has not a significant influence on the behavior of the models.

7.4.2 Milan Dataset

This dataset contains sensor data collected in the home of a volunteer adult. The residents in the home were a woman and a dog. The woman's children visited on several occasions. The following activities are annotated within the dataset: Bed-to-Toilet, Chores, Desk_Activity, Dining_Rm_Activity, Eve_Meds, Guest_Bathroom, Kitchen_Activity, Leave_Home, Master_Bathroom, Meditate, Watch_TV, Sleep, Read, Morning_Meds and, Master_Bedroom_Activity. Milan Dataset contains 2310 instances in all.

The sensor events are generated from motion, door closure, and temperature sensors. The layout of the sensors in the home is shown in Figure 7.24

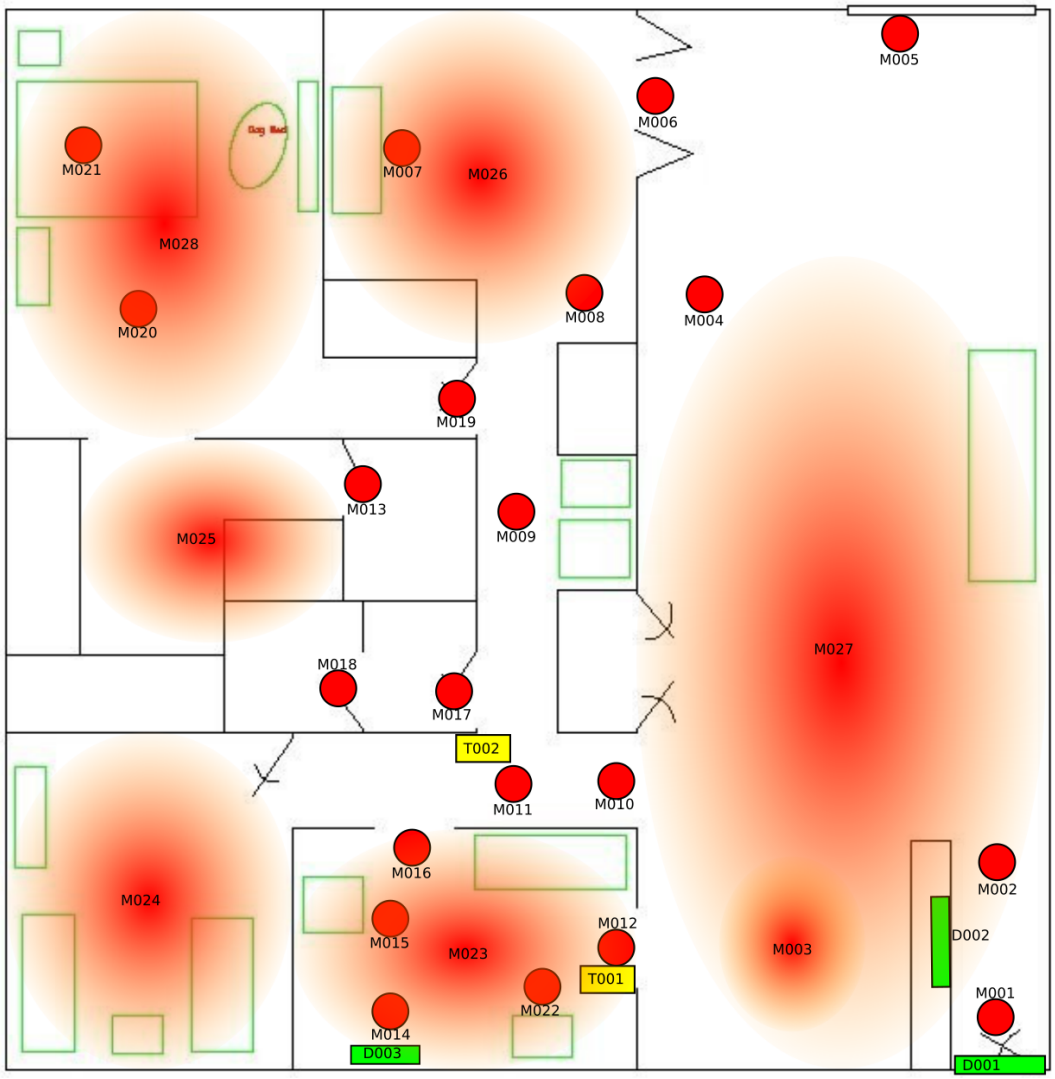

Fig. 7.24. Schematic Example of distribution of sensors in environment. Image extracted in [31]

Dataset is selected based on the same criteria by [45] on the challenges that include similar events in the instances of different activities, the addition of noise in the data-

```
Bed_to_Toilet begin
2009-10-16 03:55:58.000006 B OFF
2009-10-16 03:56:05.000085 A ON
2009-10-16 03:56:07.000039 A OFF
2009-10-16 03:56:08.000022 D ON
2009-10-16 03:56:09.000037 A ON
2009-10-16 03:56:11.000089 A OFF
2009-10-16 03:56:13.000002 A ON
2009-10-16 03:56:18.000024 D OFF
2009-10-16 03:56:20.000005 A OFF
2009-10-16 03:56:22.000019 E ON
2009-10-16 03:56:35.000079 E OFF
2009-10-16 03:56:38 J ON
2009-10-16 03:56:50.000008 J OFF
2009-10-16 03:58:14.000016 J ON
2009-10-16 03:58:26.000021 J OFF
2009-10-16 03:58:28.000002 E ON
Bed_to_Toilet end
```

Fig. 7.25. Example of the raw data in the dataset.

sets because of the presence of nonparticipating agents such as pets in the home, and the sensor errors affecting the inputs during an activity instance. Another factor is the number of instances per activity class since fewer instances make it challenging for the learning methods to be trained. In contrast, the large number of instances increases the computational cost. The data are represented as follows, as illustrated in Figure 7.25. We have a sensor activation sequence, where each sensor sets a binary value as open/close, on/off, and temperature. Each temperature has a binary value between more or less a ϵ. ϵ is a truncated mean value according to all values in a respective sensor. This case study extends the use of the proposed algorithm KFC4STREAM because, at this point, there are atomic activities in a sequential representation. The objective is to identify the predictive potential of the algorithms in sequential data.

The dataset has a structure as follow in Figure 7.25. The letters (A, B, C, D, E, F) mean an action in respective activities, and (OFF, ON) mean the state. The atomic activity is described by a sequence of states in sequential form. Other information means the timestamp of activity.

For our study of the case, A atomic activity is a par of activity + state. Based on previous example, we can assume e.g: (A,OFF) = A_1; (A,ON) = A_2; (B,OFF) = A_3; (D,ON) = A_4 (D,OFF) = A_5; (E,ON) = A_6;(E,OF) = A_7; (J,OFF) = A_8; (J,ON) = A_9.

Thus, our sequence for Bed_to_toilet is = $A_3A_1A_2A_4A_2A_1A_2A_5A_6A_7A_9A_8A_9A_8A_9$.

Data stream learning algorithms used according to different methods. Each algorithm is based on different methods:

- SamKNN (lazy-based): k =3; limit of samples = 5000; minSTMSize = 50; Relative LTM Size = 0.4
- Naive Bayes Online (probabilistic-based)
- Hoeffding Adaptative Tree (rules-based): Grace Period = 200; Split Criteria = Info Gain
- SGD Multi-Class (Optimization-based): Loss Function = Hinge; Lambda Regularization = 0.005; Learning Rate = 0.005

Experimental Protocol: All metrics are computed accordingly to the Prequential test-then-train procedure. Processing time is measured in seconds, while memory usage is given in RAM-Hours, where 1 RAM-Hour equals 1 GB of RAM used per hour of processing (GB-Hour). We adopted a window size $W = 50$ to keep track of in all experiments. WekaClassifier has the same value for the size of first window (= 50) and $W = 25$. the k value for KFC4STREAM and K-mer Filtered Classifier is k=3. We have a Time taken to build model: 2218.56s in K-mer Filtered Classifier, 2215.56s in KFC4STREAM, 0.03s in HMM. Table 7.2 shows all information related length of sequence and number of letter of alphabetic (Σ, number of atomic activities) of Milan dataset.

As in case study #1, we compared KFC4STREAM with other algorithms that deal with sequential data (K-mer Filtered Classifier and Hidden Markov models). We used the same metrics in case study #1 (Accuracy, Precision, Recall, and F-Measure). Figures 7.26 7.27, 7.28 ,7.29 and 7.30 show the results covering accuracy, precision, recall and F-measure, respectively. Table 7.5 shows the evaluation time and model cost.

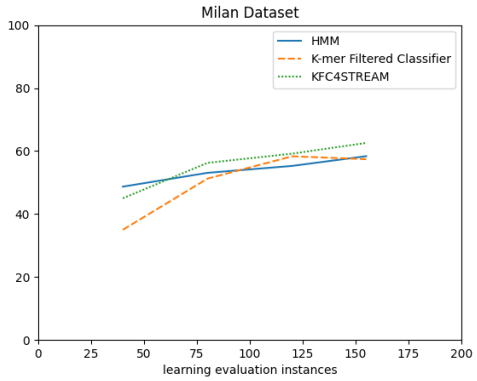

Fig. 7.26. Accuracy evolution involving HMM, K-mer Filtered Classifier, and KFC4STREAM.

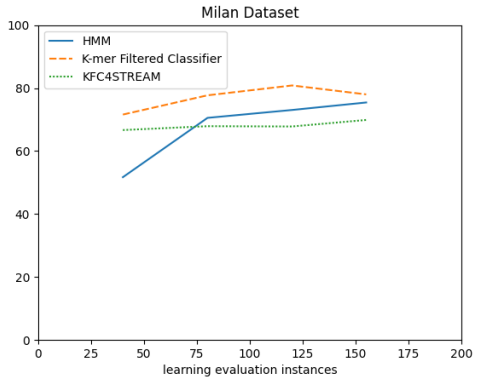

Fig. 7.27. Precision evolution involving HMM, K-mer Filtered Classifier, and KFC4STREAM.

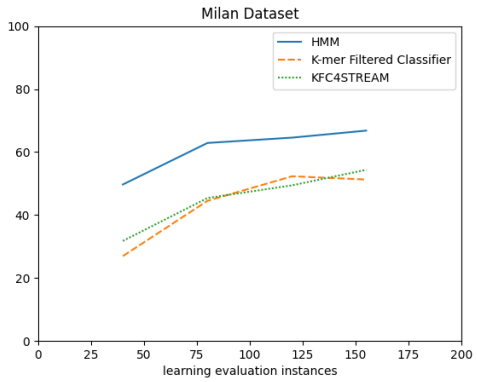

Fig. 7.28. Kappa evolution involving HMM, K-mer Filtered Classifier, and KFC4STREAM.

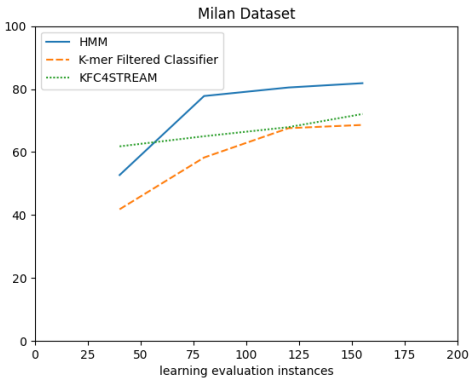

Fig. 7.29. Recall evolution involving HMM, K-mer Filtered Classifier, and KFC4STREAM.

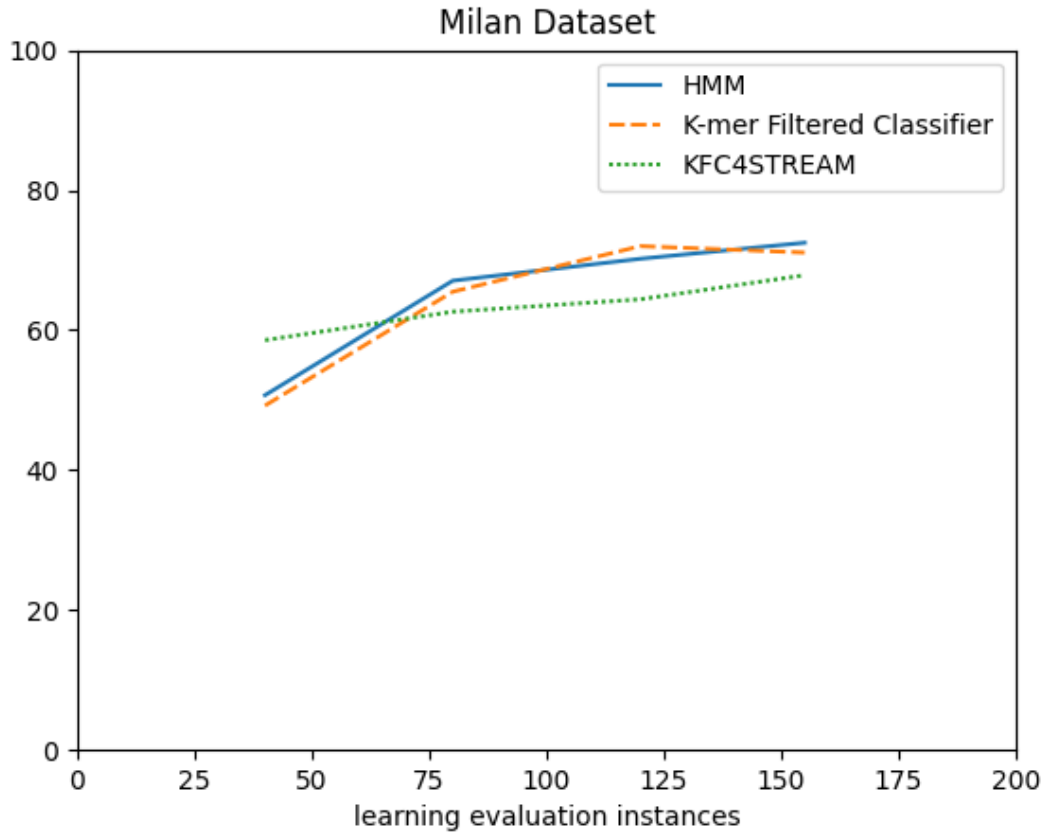

Fig. 7.30. F1 evolution involving HMM, K-mer Filtered Classifier, and KFC4STREAM.

Table 7.5. Evaluation time and model cost to Opportunity dataset.

Algorithms	Evaluation time (cpu seconds)	Model cost (RAM-Hours)
HMM	0.04	≈ 0.0
K-mer Filtering Classifier	2.817	≈ 0.0
KFC4STREAM	0.132	≈ 0.0

For this approach, the results show us that KFC4Stream is sensitive to poorly distributed classes (we can observe that it presents a fast growth curve of accuracy and a positive result to other algorithms. However, when looking at the other evolution metrics, its growth evolution has minimal growth. The growth curve is almost linear, reaching a value that converts more quickly. However, for this dataset scenario, where we have a reduced buffer with few samples, KFC4STREAM has major overfitting issues in specifically majority classes.

Statistical Test:

We executed the Friedman Signed-Rank statistical test for multiples domains, one for each metric, as indicated by [69] and [35]. We aimed to compare HMM, K-mer Filtered Classifier, and KFC4STREAM on three artificial dataset and two real datasets, to verify if there is a statistical difference among the algorithms according to accuracy, kappa metric, precision, recall and F1 measures. We used the final metrics evolved the data stream.

Because the null hypotheses were rejected on among metrics, we also executed the post-hoc Nemenyi test [69, 35], to verify how the algorithms are ranked and, also, where is the difference among them. Figures 7.31, 7.32, 7.33, 7.34 and 7.35 present the results obtained using, respectively, accuracy, kappa, precision, recall and F1 metrics. As *CD* (Critical Difference) value is high, for all diagrams, we can conclude that the post-hoc test is powerful enough to detect any significant differences between all algorithms. So, The null hypothesis (they behave similarly) can be rejected with 0.5% significance. We can conclude that KFC4STREAM has the best results in the overall form for different metrics.

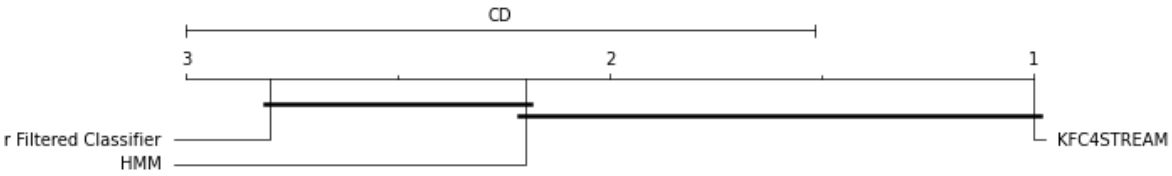

Fig. 7.31. Statistical test for accuracy metric on SCHAR algorithms.

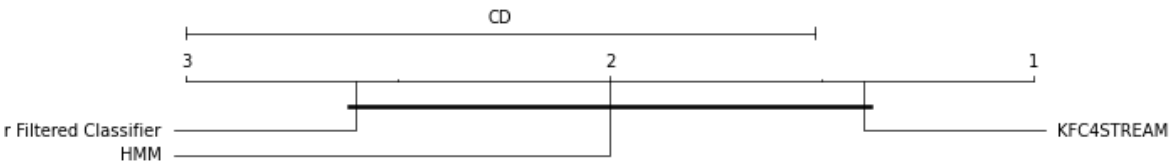

Fig. 7.32. Statistical test for kappa metric on SCHAR algorithms.

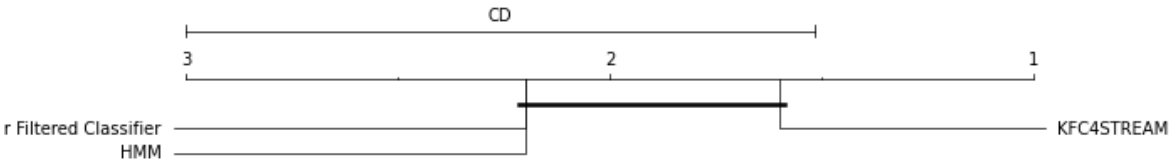

Fig. 7.33. Statistical test for precision metric on SCHAR algorithms.

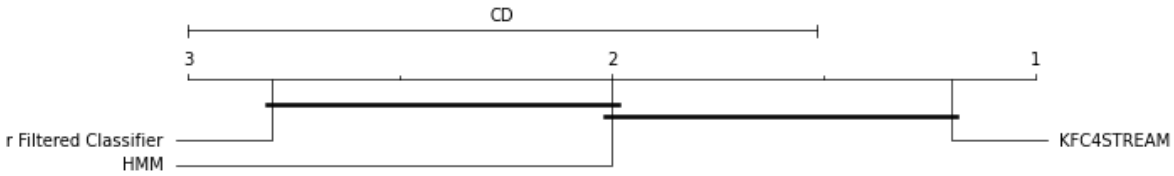

Fig. 7.34. Statistical test for recall metric on SCHAR algorithms.

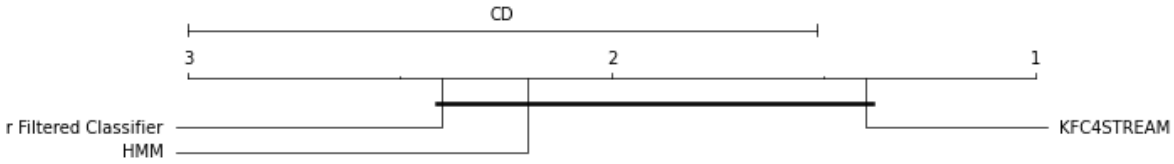

Fig. 7.35. Statistical test for f1 metric on SCHAR algorithms.

7.5 Final Remarks

This chapter covers a different type of HAR system from the study in previous chapters. We analyzed Complex Human Activity Recognition in a scenario where a temporal/sequential dependency exists between the atomic activities in a data stream called Sequential Complex Human Activity Recognition (SCHAR). Given the importance of SCHAR in previous chapters and the lack of work on data stream learning, we propose KFC4STREAM, an algorithm based on the K-mer Filtered Classifier, however, focused on data stream learning. Performance analyzes of our proposed algorithm were performed with traditional algorithms for sequential data analysis through artificial data (created by the author) and on original data available in the literature. We can observe that KFC4STREAM has high predictive power in reduced size compared to the number of atomic activities previously trained.

Chapter 8

Conclusion

Over the last decades, new technologies for collecting, storing, and processing data have arrived. These technologies are more and more accessible to users. Sensors have particularly invaded humans' life in the last years, being largely present in devices. These devices can communicate with each other, without user assistance. All those devices, the users that both generate and consume these data, and the computational systems that orchestrate all these entities form a software ecosystem. An advantage of these ecosystems is the generation of many opportunities for new applications that improve quality of life, such as the applications in Ambient Assisted Living, for instance, helping elders in their daily activities. Researchers have explored the Human Activity Recognition (HAR) area to propose techniques and tools to assist this type of activity with increasing scale, reliability, and safety. However, as more diverse are the activities, as larger the necessity for different types of sensors. In this way, larger is the necessity to properly collect and manage this kind of data, as well as manage the necessary knowledge for inferring HAR. This scenario leads to arise the necessity of CAMs to provide management solutions for helping HAR systems for diverse problems.

This study initially aimed to understand the state-of-the-art development of HAR applications in context-aware systems, especially when using ML. To achieve this end, we conducted a systematic literature review, selecting publications related to CAM projects that either incorporate ML algorithms or in for HAR solutions. All setups about this study was published in AI review Journal [92]. We were also particularly interested to understand how these middlewares deliver context inference (reasoning) service. Although the literature presented many different methods for context reasoning, this work emphasized cases involving ML. Our primary interest in ML is due to has been presented promising results in many different types of HAR problems. We answered our four

research questions by analyzing the papers selected in our systematic review processes. As a result of our analysis, we could observe some limitations that deserve attention from the context-aware researchers' community. CAMs should evolve to offer services for (i) impact the use of limited resources in data stream learning for context-aware computing; (ii) allowing adaptive models for HAR, including the management of non-stationary data by stream learning algorithms; (iii) (i) developing complex HAR systems in looking for the adaptability of the algorithms. Thus, we elaborate some research questions about these limitations:

RQ1: *Does the use of batch learning models with limited infrastructure result better than online models in implementing HAR techniques?*

Findings: Under the first limitation, considering the limited computational resources, effective strategies are welcome for HAR. Systems that require agile decision-making must fit into resource restrictions, such as memory size. We present a comparative analysis between batch machine learning models considering restrictions in memory consumption and data stream learning algorithms. All setups about this study was published in MICAI proceedings [91]. We used collected data from sensors available in benchmark datasets presented in the literature. The goal is to identify whether batch learning algorithms with limited memory resources can be robust compared to data stream learning algorithms. Our study compared four batch learning with limited memory sizes to four data stream learning algorithms. These algorithms follow different learning approaches: distance-based learning, rules-based learning, probability-based learning, and optimization-based learning. Our results show that batch models are more sensitive to low data volume and low memory size than online models. According to the reduced memory consumption, batch learning presented difficulty generalizing the models. Online models converge fast and show favorable results, in particular to SAMkNN, beside the statistical test means that most algorithms in similar methods does not have difference statistical significance. / **Future Works:** we will apply it in a distributed environment where we have heterogeneous data with high volume. Also, We intend to apply a study replacing memory model control for BM_{mc} to WekaClassifier. Moreover, we intend to apply this approach in edge computing and monitor the performance of models in edge nodes, for example in Federate Learning scenarios.

RQ2: *Are machine learning models able to adapt to similar users?*

Findings: For the second limitation, we observe that in context-aware HARs scenarios, there is a concern about how these HAR systems are best suited to a specific

user. We propose a strategy for recommending HAR models created from user data with characteristics similar to a respective target user. This similar user data refers to how training users perform the same activities as a target user. First, an analysis was made of the importance of implementing adaptive strategies over time, comparing adaptive algorithms with non-adaptive versions. We can observe in most datasets that non-adaptive models suffer a degradation in their quality compared to adaptive models. Thus, we propose a strategy for recommending adaptive models based on initial data from a target user. We apply a k-means clustering algorithm to these data and compare the structure of this partition with partitions previously trained by the system. These trained partitions gather the following characteristics: cluster centroids, densities, within-cluster standard deviations, and boundaries (gravitational force). The system recommends a robust model with characteristics in executing activities similar to the target user. / **Future Works:** A future step is implementing a recommender system for models of complex activities. Moreover, we intend apply this and proposed actual system as service in a ubiquitous context-aware middleware.

RQ3: *Is it possible to create a model for complex sequential activities that can self-adapt to the data stream?*

Findings: For the third limitation, we observed that after the systematic review process, there is a lack of works that propose some solution for data stream learning in a scenario of Recognition of complex activities in sequential form (SCHAR). Therefore, we propose the KFC4STREAM algorithm, a data stream learning algorithm based on the K-mer Filtered Classifier. We applied three experiments on an artificial dataset and two real datasets. The results show that KFC4STREAM has better predictive power in scenarios where the size of atomic activities is minimal, and the activity buffer is small. These results are compared to the main state-of-the-art algorithms in sequential data analysis, in the case of K-mer Filtered Classifier and Hidden Markov Models. In addition, the training of this model is sensitive to the number of atomic activities because the greater the number of activities, the longer the data training time. / **Future Works:** we will propose new algorithms based on stream learning that solve new problems in recognizing complex activities, such as Concurrent and Interveleated Complex Activities. Extend the study for different values of k, allowing us a broader view of the performance of the KFC4STREAM at the value of k. Other future step is to evaluate this scenario in a scalable production environment, with multiple nodes sending packets around the network with activity infos. The number of nodes must be in the thousands or millions. Another objective will be to apply KFC4STREAM in scenarios present in bioinformatics,

such as genome size estimation, Sequence Alignment, and other scenarios.

References

[1] Abbas, M. M., Mohie-Eldin, M. M., Yasser, E.-M. Assessing the effects of data selection and representation on the development of reliable e. coli sigma 70 promoter region predictors. *PloS one 10*, 3 (2015), e0119721.

[2] Abdallah, Z. S., Gaber, M. M. Kb-cb-n classification: Towards unsupervised approach for supervised learning. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)* (2011), IEEE, p. 283–290.

[3] Abdallah, Z. S., Gaber, M. M., Srinivasan, B., Krishnaswamy, S. Cbars: Cluster based classification for activity recognition systems. In *International Conference on Advanced Machine Learning Technologies and Applications* (2012), Springer, p. 82–91.

[4] Abdallah, Z. S., Gaber, M. M., Srinivasan, B., Krishnaswamy, S. Activity recognition with evolving data streams: A review. *ACM Comput. Surv. 51*, 4 (julho de 2018), 71:1–71:36.

[5] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., Steggles, P. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing* (1999), Springer, p. 304–307.

[6] Adelsberger, R., Tröster, G. Pimu: A wireless pressure-sensing imu. In *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on* (2013), IEEE, p. 271–276.

[7] Aggarwal, J., Ryoo, M. Human activity analysis: A review. *ACM Comput. Surv. 43*, 3 (apr 2011).

[8] Alam, M. G. R., Haw, R., Kim, S. S., Azad, M. A. K., Abedin, S. F., Hong, C. S. Em-psychiatry: an ambient intelligent system for psychiatric emergency. *IEEE Transactions on Industrial Informatics 12*, 6 (2016), 2321–2330.

[9] Alam, M. N. U., Chowdhury, U. F. Short k-mer abundance profiles yield robust machine learning features and accurate classifiers for rna viruses. *PloS one 15*, 9 (2020), e0239381.

[10] Alti, A., Laborie, S., Roose, P. Cloud semantic-based dynamic multimodal platform for building mhealth context-aware services. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (Oct 2015), p. 357–364.

[11] Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J. L. A public domain dataset for human activity recognition using smartphones. In *Esann* (2013).

[12] Anjomshoa, F., Aloqaily, M., Kantarci, B., Erol-Kantarci, M., Schuckers, S. Social behaviometrics for personalized devices in the internet of things era. *IEEE Access 5* (2017), 12199–12213.

[13] Archetti, F., Djordjevic, D., Giordani, I., Sormani, R., Tisato, F. A reasoning approach for modelling and predicting terroristic attacks in urban environments. In *Security Technology (ICCST), 2014 International Carnahan Conference on* (2014), IEEE, p. 1–6.

[14] Asghari, P., Soleimani, E., Nazerfard, E. Online human activity recognition employing hierarchical hidden markov models. *Journal of Ambient Intelligence and Humanized Computing 11*, 3 (2020), 1141–1152.

[15] Babüroğlu, E. S., Durmuşoğlu, A., Dereli, T. Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection. *Expert Systems with Applications 163* (2021), 113786.

[16] Banos, O., Toth, M. A., Damas, M., Pomares, H., Rojas, I. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors 14*, 6 (2014), 9995–10023.

[17] Bhandari, N., Khare, S., Walambe, R., Kotecha, K. Comparison of machine learning and deep learning techniques in promoter prediction across diverse species. *PeerJ Computer Science 7* (2021).

[18] Bifet, A. Adaptive learning and mining for data streams and frequent patterns. *ACM SIGKDD Explorations Newsletter 11*, 1 (2009), 55–56.

[19] Bifet, A., Gavalda, R. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining* (2007), SIAM, p. 443–448.

[20] Bifet, A., Gavalda, R., Holmes, G., Pfahringer, B. *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018.

[21] Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B. MOA: massive online analysis. *Journal of Machine Learning Research 11* (2010), 1601–1604.

[22] Bikakis, A., Patkos, T., Antoniou, G., Plexousakis, D. A survey of semantics-based approaches for context reasoning in ambient intelligence. In *European Conference on Ambient Intelligence* (2007), Springer, p. 14–23.

[23] Bobek, S., Porzycki, K., Nalepa, G. J. Learning sensors usage patterns in mobile context-aware systems. In *FedCSIS* (2013), p. 993–998.

[24] Cardoso, H., Moreira, J. Improving human activity classification through online semi-supervised learning. In *STREAMEVOLV@ECML-PKDD* (09 2016).

[25] Casalicchio, E. A study on performance measures for auto-scaling cpu-intensive containerized applications. *Cluster Computing 22*, 3 (2019), 995–1006.

[26] Casalicchio, E., Perciballi, V. Auto-scaling of containers: The impact of relative and absolute metrics. In *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)* (2017), IEEE, p. 207–214.

[27] Celdrán, A. H., Pérez, M. G., Clemente, F. G., Pérez, G. M. Precise: Privacy-aware recommender based on context information for cloud service environments. *IEEE Communications Magazine 52*, 8 (2014), 90–96.

[28] Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., Millán, J. d. R., Roggen, D. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters 34*, 15 (2013), 2033–2042.

[29] Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., Leung, V. C. Body area networks: A survey. *Mobile networks and applications 16*, 2 (2011), 171–193.

[30] Concone, F., Re, G. L., Morana, M. A fog-based application for human activity recognition using personal smart devices. *ACM Transactions on Internet Technology (TOIT) 19*, 2 (2019), 1–20.

[31] Cook, D. J., Schmitter-Edgecombe, M. Assessing the quality of activities in a smart environment. *Methods of information in medicine 48*, 05 (2009), 480–485.

[32] Damaševicius, R. Splice site recognition in dna sequences using k-mer frequency based mapping for support vector machine with power series kernel. In *2008 International Conference on Complex, Intelligent and Software Intensive Systems* (2008), IEEE, p. 687–692.

[33] de la Concepción, M. Á. Á., Morillo, L. M. S., García, J. A. Á., González-Abril, L. Mobile activity recognition and fall detection system for elderly people using ameva algorithm. *Pervasive and Mobile Computing 34* (2017), 3–13.

[34] de Sousa Pinto, A., Bernardini, F., Miranda, L., Viterbo, J., Meza, E. M. An exploratory study using stream learning algorithms to predict duration time of vehicle routes. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)* (2020), IEEE, p. 299–304.

[35] Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research 7*, Jan (2006), 1–30.

[36] Dey, A. K., Häkkilä, J. Context-awareness and mobile devices. In *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*. IGD-Global, 2008, cap. 13.

[37] Dimitrievski, A., Zdravevski, E., Lameski, P., Trajkovik, V. A survey of ambient assisted living systems: Challenges and opportunities. In *Intelligent Computer Communication and Processing (ICCP), 2016 IEEE 12th International Conference on* (2016), IEEE, p. 49–53.

[38] Dimitrov, T., Pauli, J., Naroska, E. A probabilistic reasoning framework for smart homes. In *Proceedings of the 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing: Held at the ACM/IFIP/USENIX 8th International Middleware Conference* (New York, NY, USA, 2007), MPAC '07, Association for Computing Machinery, p. 1–6.

[39] Doherty, A. R., Caprani, N., Conaire, C. Ó., Kalnikaite, V., Gurrin, C., Smeaton, A. F., O'Connor, N. E. Passively recognising human activities through lifelogging. *Computers in Human Behavior 27*, 5 (2011), 1948–1958.

[40] Dragone, M., Amato, G., Bacciu, D., Chessa, S., Coleman, S., Di Rocco, M., Gallicchio, C., Gennaro, C., Lozano, H., Maguire, L., others. A cognitive robotic ecology approach to self-configuring and evolving aal systems. *Engineering Applications of Artificial Intelligence 45* (2015), 269–280.

[41] Endler, M., Baptista, G., Silva, L., Vasconcelos, R., Malcher, M., Pantoja, V., Pinheiro, V., Viterbo, J. Contextnet: Context reasoning and sharing middleware for large-scale pervasive collaboration and social networking. In *Proceedings of the Workshop on Posters and Demos Track* (2011), ACM, p. 2.

[42] et. al., N. V. C. Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research 16* (2002), 321–357.

[43] European Comission. Ethics guidelines for trustworthy ai, 2019. Available at https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[44] Faceli, K. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. LTC, 2011.

[45] Fahad, L. G., Tahir, S. F. Activity recognition and anomaly detection in smart homes. *Neurocomputing 423* (2021), 362–372.

[46] Faial, D., Bernardini, F., Miranda, L., Viterbo, J. Anomaly detection in vehicle traffic data using batch and stream supervised learning. In *EPIA Conference on Artificial Intelligence* (2019), Springer, p. 675–684.

[47] Fan, Q., Bobbitt, R., Zhai, Y., Yanagawa, A., Pankanti, S., Hampapur, A. Recognition of repetitive sequential human activity. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)* (Los Alamitos, CA, USA, jun 2009), IEEE Computer Society, p. 943–950.

[48] Fong, J., Indulska, J., Robinson, R. A framework to support intelligibility in pervasive applications. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on* (2013), IEEE, p. 37–42.

[49] Forkan, A., Khalil, I., Tari, Z. Cocamaal: A cloud-oriented context-aware middleware in ambient assisted living. *Future Generation Computer Systems 35* (2014), 114–127. Special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.

[50] Fortino, G., Galzarano, S., Gravina, R., Li, W. A framework for collaborative computing and multi-sensor data fusion in body sensor networks. *Information Fusion 22* (2015), 50 – 70.

[51] Frank, E., Hall, M. A., Witten, I. H. *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. ed. Morgan Kaufmann, 2016.

[52] Frey, J. Adapt–a dynamic approach for activity prediction and tracking for ambient intelligence. In *Intelligent Environments (IE), 2013 9th International Conference on* (2013), IEEE, p. 254–257.

[53] Gama, J. *Knowledge discovery from data streams*. Chapman and Hall/CRC, 2010.

[54] Gama, J., Medas, P., Castillo, G., Rodrigues, P. Learning with drift detection. In *Brazilian symposium on artificial intelligence* (2004), Springer, p. 286–295.

[55] Gama, J., Sebastião, R., Rodrigues, P. P. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009), p. 329–338.

[56] Gani, M. O., Saha, A. K., Ahsan, G. M. T., Ahamed, S. I., Smith, R. O. A novel framework to recognize complex human activity. In *Proc. 41st IEEE Annual Computer Software and Applications Conference* (2017), p. 948–956.

[57] Garcia-Ceja, E., Brena, R. F., Carrasco-Jimenez, J. C., Garrido, L. Long-term activity recognition from wristwatch accelerometer data. *Sensors 14*, 12 (2014), 22500–22524.

[58] Gayathri, K., Elias, S., Shivashankar, S. Composite activity recognition in smart homes using markov logic network. In *Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015 IEEE 12th Intl Conf on* (2015), IEEE, p. 46–53.

[59] Gepperth, A., Hammer, B. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (esann)* (2016).

[60] Gu, T., Wu, Z., Tao, X., Pung, H. K., Lu, J. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *2009 IEEE International Conference on Pervasive Computing and Communications* (2009), IEEE, p. 1–9.

[61] Gu, T., Wu, Z., Wang, L., Tao, X., Lu, J. Mining emerging patterns for recognizing activities of multiple users in pervasive computing. In *2009 6th Annual International Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous* (2009), IEEE, p. 1–10.

[62] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. The WEKA data mining software: An update. *SIGKDD Explorations 11*, 1 (2009), 10–18.

[63] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. The WEKA data mining software: an update. *SIGKDD Explorations 11*, 1 (2009), 10–18.

[64] Hanneke, S. *Theory of Disagreement-Based Active Learning.* Now Foundations and Trends, 2014.

[65] Hasse, D., De Rolt, C. R. Collega semantic middleware for collaborative assistance in mobile social networks. In *AEIT International Annual Conference, 2017* (2017), IEEE, p. 1–6.

[66] Hoque, M. R., Kabir, M. H., Jang, J.-H., Yang, S.-H. Middleware aided context-aware service for smart home. In *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on* (2014), IEEE, p. 1–2.

[67] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., others. Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission 21* (2004), 79.

[68] Huang, Z., Lin, K.-J., Tsai, B.-L., Yan, S., Shih, C.-S. Building edge intelligence for online activity recognition in service-oriented iot systems. *Future Generation Computer Systems* (2018).

[69] Japkowicz, N., Shah, M. *Evaluating learning algorithms: a classification perspective.* Cambridge University Press, 2011.

[70] Junker, H., Amft, O., Lukowicz, P., Tröster, G. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition 41*, 6 (2008), 2010–2024.

[71] Karagiannaki, K., Panousopoulou, A., Tsakalides, P. A benchmark study on feature selection for human activity recognition. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct* (2016), p. 105–108.

[72] Karakostas, A., Lazarou, I., Meditskos, G., Stavropoulos, T. G., Kompatsiaris, I., Tsolaki, M. Intelligent user interfaces to support diagnosis and assessment of people with dementia: An expert evaluation. In *Pervasive Computing Paradigms for Mental Health* (Cham, 2016), S. Serino, A. Matic, D. Giakoumis, G. Lopez, and P. Cipresso, Eds., Springer International Publishing, p. 196–206.

[73] Kasarla, T., Nagendar, G., Hegde, G. M., Balasubramanian, V., Jawahar, C. Region-based active learning for efficient labeling in semantic segmentation. In *In Proc. 2019 IEEE Winter Conf. App. Computer Vision (WACV)* (2019).

[74] Keshtkar Langaroudi, M., Yamaghani, M. Sports result prediction based on machine learning and computational intelligence approaches: A survey. *Journal of Advances in Computer Engineering and Technology 5*, 1 (2019), 27–36.

[75] Kifer, D., Ben-David, S., Gehrke, J. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30* (2004), VLDB Endowment, p. 180–191.

[76] Kitchenham, B. Procedures for performing systematic reviews. *Keele, UK, Keele University 33*, 2004 (2004), 1–26.

[77] Kotthaus, H., Korb, I., Lang, M., Bischl, B., Rahnenführer, J., Marwedel, P. Runtime and memory consumption analyses for machine learning r programs. *Journal of Statistical Computation and Simulation 85*, 1 (2015), 14–29.

[78] Krishnan, N. C., Cook, D. J. Activity recognition on streaming sensor data. *Pervasive and mobile computing 10* (2014), 138–154.

[79] Krogh, A., Brown, M., Mian, I. S., Sjolander, K., Haussler, D. Hidden markov models in computational biology. applications to protein modeling. *Journal of molecular biology 235*, 5 (1994), 1501–1531.

[80] Kwapisz, J. R., Weiss, G. M., Moore, S. A. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter 12*, 2 (2011), 74–82.

[81] L-Manzalawy, Y., Bui, N., Sridharan, K., Brendel, V., Honavar, V. Gennotate: Genome annotation toolkit. http://ailab.ist.psu.edu/gennotate, 2015. Accessed: 2020-10-30.

[82] Lara, O. D., Labrador, M. A., others. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials 15*, 3 (2013), 1192–1209.

[83] Lemaire, V., Salperwyck, C., Bondu, A. A survey on supervised classification on data streams. *Lecture Notes in Business Information Processing* (03 2015).

[84] Li, Q., Gravina, R., Li, Y., Alsamhi, S. H., Sun, F., Fortino, G. Multi-user activity recognition: Challenges and opportunities. *Information Fusion 63* (2020), 121–135.

[85] Lorenzi, C., Barriere, S., Villemin, J.-P., Dejardin Bretones, L., Mancheron, A., Ritchie, W. imoka: k-mer based software to analyze large collections of sequencing data. *Genome biology 21*, 1 (2020), 1–19.

[86] Madeira, R., Nunes, L. A machine learning approach for indirect human presence detection using iot devices. In *2016 Eleventh International Conference on Digital Information Management (ICDIM)* (2016), IEEE, p. 145–150.

[87] Mane, Y. V., Surve, A. R. Capm: Context aware provisioning middleware for human activity recognition. In *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)* (May 2016), p. 661–665.

[88] Manekar, S. C., Sathe, S. R. A benchmark study of k-mer counting methods for high-throughput sequencing. *GigaScience 7*, 12 (2018), giy125.

[89] Mehrang, S., Pietilä, J., Korhonen, I. An activity recognition framework deploying the random forest classifier and a single optical heart rate monitoring and triaxial accelerometer wrist-band. *Sensors 18*, 2 (2018), 613.

[90] Melsted, P., Pritchard, J. K. Efficient counting of k-mers in dna sequences using a bloom filter. *BMC bioinformatics 12*, 1 (2011), 333.

[91] Miranda, L., Viterbo, J., Bernardini, F. Impact of memory control on batch learning in human activity recognition scenario in comparison to data stream learning. In *Mexican International Conference on Artificial Intelligence* (2020), Springer, p. 145–157.

[92] Miranda, L., Viterbo, J., Bernardini, F. A survey on the use of machine learning methods in context-aware middlewares for human activity recognition. *Artificial Intelligence Review 55*, 4 (2022), 3369–3400.

[93] Mohamad, S., Sayed-Mouchaweh, M., Bouchachia, A. Online active learning for human activity recognition from sensory data streams. *Neurocomputing* (2019). in press.

[94] Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., Group, P., others. Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. *PLoS medicine 6*, 7 (2009), e1000097.

[95] Mshali, H., Lemlouma, T., Moloney, M., Magoni, D. A survey on health monitoring systems for health smart homes. *International Journal of Industrial Ergonomics 66* (2018), 26–56.

[96] Neto, A. D. F., de Azevedo, B. R., Boufleuer, R., Lima, J. C. D., Augustin, I., Pasin, M. An approach based on activity theory and the srk model for risk and performance evaluation of human activities in a context-aware middleware. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia* (2014), p. 40–47.

[97] Nguyen, L. T., Zeng, M., Tague, P., Zhang, J. Recognizing new activities with limited training data. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers* (2015), ACM, p. 67–74.

[98] Ortega, J. G., Han, L., Whittacker, N., Bowring, N. A machine-learning based approach to model user occupancy and activity patterns for energy saving in buildings. In *2015 Science and Information Conference (SAI)* (2015), IEEE, p. 474–482.

[99] Palumbo, F., Gallicchio, C., Pucci, R., Micheli, A. Human activity recognition using multisensor data fusion based on reservoir computing. *Journal of Ambient Intelligence and Smart Environments 8*, 2 (2016), 87–107.

[100] Pan, S. J., Yang, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering 22*, 10 (2009), 1345–1359.

[101] Perera, C., Zaslavsky, A., Christen, P., Compton, M., Georgakopoulos, D. Context-aware sensor search, selection and ranking model for internet of things middleware. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on* (2013), vol. 1, IEEE, p. 314–322.

[102] Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials 16*, 1 (2014), 414–454.

[103] Phiri, C. C., Ju, Z., Kubota, N., Liu, H. Enhanced robot learning using fuzzy q-learning & context-aware middleware. In *Micro-NanoMechatronics and Human Science (MHS), 2016 International Symposium on* (2016), IEEE, p. 1–8.

[104] Punj, R., Kumar, R. Technological aspects of wbans for health monitoring: a comprehensive review. *Wireless Networks* (2018), 1–33.

[105] Radicioni, D. P., Esposito, R. Breve: an hmperceptron-based chord recognition system. In *Advances in Music Information Retrieval*. Springer, 2010, p. 143–164.

[106] Rafferty, J., Synnott, J., Nugent, C., Morrison, G., Tamburini, E. Nfc based dataset annotation within a behavioral alerting platform. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on* (2017), IEEE, p. 146–151.

[107] Ramírez, J. M. N., Roose, P., Dalmau, M. Distributed interfaces and context-oriented broadcast services in a smart-home environment. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2016 IEEE 12th International Conference on* (2016), IEEE, p. 1–8.

[108] Rasanen, O., Kakouros, S. Modeling dependencies in multiple parallel data streams with hyperdimensional computing. *IEEE Signal Processing Letters 21*, 7 (2014), 899–903.

[109] Raychoudhury, V., Cao, J., Kumar, M., Zhang, D. Middleware for pervasive computing: A survey. *Pervasive and Mobile Computing 9*, 2 (2013), 177–200.

[110] Razzaque, M. A., Clarke, S. Compression-based energy efficient sensor data gathering framework for smartphones. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)* (2016), IEEE, p. 126–132.

[111] Rekha, S. B., Rao, M. V. Methodical activity recognition and monitoring of a person through smart phone and wireless sensors. In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)* (2017), IEEE, p. 1456–1459.

[112] Reyes-Ortiz, J.-L., Oneto, L., Samà, A., Parra, X., Anguita, D. Transition-aware human activity recognition using smartphones. *Neurocomputing 171* (2016), 754–767.

[113] Rezazadegan, F., Shirazi, S., Baktashmotlagh, M., Davis, L. S. On encoding temporal evolution for real-time action prediction. *arXiv preprint arXiv:1709.07894* (2017).

[114] Rodrigues, É., Viterbo, J., Conci, A., MacHenry, T. A context-aware middleware for medical image based reports. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)* (Nov 2015), p. 1–4.

[115] Saha, S., Duwe, H., Zambreno, J. An adaptive memory management strategy towards energy efficient machine inference in event-driven neuromorphic accelerators. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)* (2019), vol. 2160, IEEE, p. 197–205.

[116] Sakr, N., Soliman, H. Current trends in complex human activity recognition. *Journal of Theoretical and Applied Information Technology 96*, 14 (2018), 4564–4583.

[117] Sansrimahachai, W., Toahchoodee, M. Mobile-phone based immobility tracking system for elderly care. In *Region 10 Conference (TENCON), 2016 IEEE* (2016), IEEE, p. 3550–3553.

[118] Sezer, O. B., Dogdu, E., Ozbayoglu, A. M. Context-aware computing, learning, and big data in internet of things: a survey. *IEEE Internet of Things Journal 5*, 1 (2018), 1–27.

[119] Sharma, A. Ensembled machine learning framework for drug sensitivity prediction. *IET Systems Biology 14* (February 2020), 39–46(7).

[120] Sharma, A., Rani, R. Machine learning perspective in cancer research. In *Handbook of Research on Disease Prediction Through Data Analytics and Machine Learning*. IGI Global, 2021, p. 142–163.

[121] Si, S., Hsieh, C.-J., Dhillon, I. S. Memory efficient kernel approximation. *The Journal of Machine Learning Research 18*, 1 (2017), 682–713.

[122] Silva, F. A., Silva, T. R. M., Ruiz, L. B., Loureiro, A. A. Conprova: A smart context provisioning middleware for vanet applications. In *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th* (2013), IEEE, p. 1–5.

[123] Sorici, A., Picard, G., Florea, A. M. Multi-agent based context management in ami applications. In *Control Systems and Computer Science (CSCS), 2015 20th International Conference on* (2015), IEEE, p. 727–734.

[124] Starzyk, J. A., He, H. Spatio–temporal memories for machine learning: A long-term memory organization. *IEEE Transactions on Neural Networks 20*, 5 (2009), 768–780.

[125] Thiebes, S., Lins, S., Sunyaev, A. Trustworthy artificial intelligence. *Electron Markets* (2020).

[126] Turaga, P., Chellappa, R., Subrahmanian, V. S., Udrea, O. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video technology 18*, 11 (2008), 1473.

[127] ur Rehman, M. H., Liew, C. S., Wah, T. Y., Khan, M. K. Towards next-generation heterogeneous mobile data stream mining applications: Opportunities, challenges, and future research directions. *Journal of Network and Computer Applications 79* (2017), 1–24.

[128] V Vyas, V., Walse, K., Dharaskar, R. A survey on human activity recognition using smartphone. *International Journal 5* (03 2017).

[129] Vaizman, Y., Ellis, K., Lanckriet, G. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE pervasive computing 16*, 4 (2017), 62–74.

[130] Vales-Alonso, J., Chaves-Diéguez, D., López-Matencio, P., Alcaraz, J. J., Parrado-García, F. J., González-Castaño, F. J. Saeta: A smart coaching assistant for professional volleyball training. *IEEE Transactions on Systems, Man, and Cybernetics: Systems 45*, 8 (2015), 1138–1150.

[131] Wang, L., Gu, T., Tao, X., Lu, J. Sensor-based human activity recognition in a multi-user scenario. In *European Conference on Ambient Intelligence* (2009), Springer, p. 78–87.

[132] Wang, L., Gu, T., Tao, X., Lu, J. A hierarchical approach to real-time activity recognition in body sensor networks. *Pervasive and Mobile Computing 8*, 1 (2012), 115–130.

[133] Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications 37*, 6 (2019), 1205–1221.

[134] Wu, J., Li, C., Miao, Y., Song, S., Li, L., Ding, Q. Context-aware reasoning middle ware applied in the mobile environment. In *Machine Learning and Cybernetics (ICMLC), 2013 International Conference on* (2013), vol. 4, IEEE, p. 1829–1835.

[135] Yang, Y., Huang, Y., Cao, J., Ma, X., Lu, J. Formal specification and runtime detection of dynamic properties in asynchronous pervasive computing environments. *IEEE Transactions on Parallel and Distributed Systems 24*, 8 (2012), 1546–1555.

[136] Yao, L., Benatallah, B., Wang, X., Tran, N. K., Lu, Q. Context as a service: Realizing internet of things-aware processes for the independent living of the elderly. In *Service-Oriented Computing* (Cham, 2016), Q. Z. Sheng, E. Stroulia, S. Tata, and S. Bhiri, Eds., Springer International Publishing, p. 763–779.

[137] Yao, L., Sheng, Q. Z., Benatallah, B., Dustdar, S., Wang, X., Shemshadi, A., Kanhere, S. S. Wits: an iot-endowed computational framework for activity recognition in personalized smart homes. *Computing 100*, 4 (2018), 369–385.

[138] Yao, L., Sheng, Q. Z., Ruan, W., Li, X., Wang, S., Yang, Z. Unobtrusive posture recognition via online learning of multi-dimensional rfid received signal strength. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)* (2015), IEEE, p. 116–123.

[139] Ye, J., Dasiopoulou, S., Stevenson, G., Meditskos, G., Kontopoulos, E., Kompatsiaris, I., Dobson, S. Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing 23* (2015), 1–25.

[140] Yi, Y., Cheng, Y., Xu, C. Mining human movement evolution for complex action recognition. *Expert Systems with Applications 78* (2017), 259–272.

[141] Yoon, B.-J. Hidden markov models and their applications in biological sequence analysis. *Current genomics 10*, 6 (2009), 402–415.

[142] Yurur, O., Liu, C. H., Moreno, W. A survey of context-aware middleware designs for human activity recognition. *IEEE Communications Magazine 52*, 6 (June 2014), 24–31.

[143] Yürür, Ö., Liu, C. H., Sheng, Z., Leung, V. C., Moreno, W., Leung, K. K. Context-awareness for mobile sensing: A survey and future directions. *IEEE Communications Surveys & Tutorials 18*, 1 (2016), 68–93.

[144] Zdraveski, E., Lameski, P., Trajkovik, V., KJulakov, A., Chorbev, I., Goleva, R., Pombo, N., Garcia, N. Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering. *IEEE Access 5* (2007), 5262–5280.

APPENDIX A – Results involving models mono versus multi-user

This Appendix shows several views involving all users mentioned in Chapter 6. The results are based on metrics: Accuracy, Kappa, Recall. Visualizations were made from lineplot plots involving data from two types of users:

1. *acuracy_user_single*: Let the target user that was tested under the fitted model from groups of users with similar characteristics to the respective target user.
2. *acuracy_user_multi*: Let the target user in a fitted model from all data users (who participated in the training step).

A.1 Results for Each User

A.1.1 User #1

A.1.1.1 Accuracy

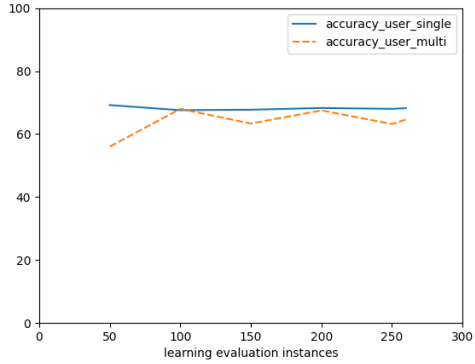

Fig. A.1. Accuracy of the Multi-label Adaptative Hoeffding Tree in User #1

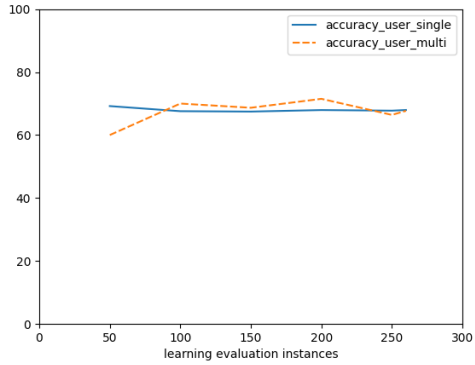

Fig. A.2. Accuracy of the Naive Bayes Online in User #1

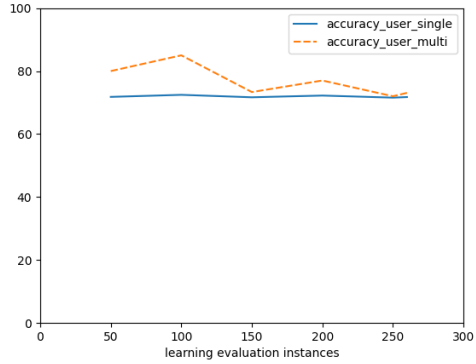

Fig. A.3. Accuracy of the SamKnn in User #1

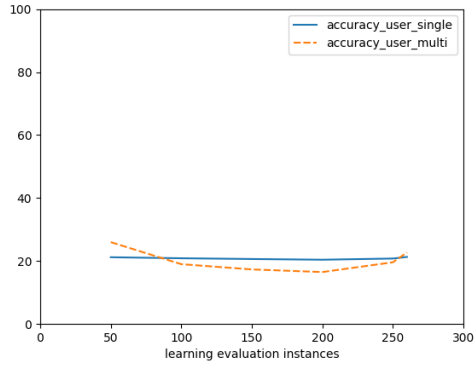

Fig. A.4. Accuracy of the Stochastic Gradient Descendant in User #1

A.1.1.2 Kappa

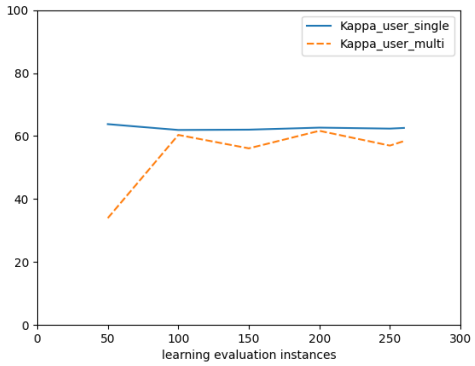

Fig. A.5. Kappa metric of the Multi-label Adaptative Hoeffding Tree in User #1

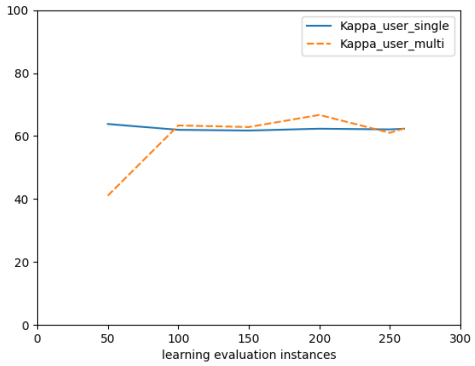

Fig. A.6. Kappa metric of the Naive Bayes Online in User #1

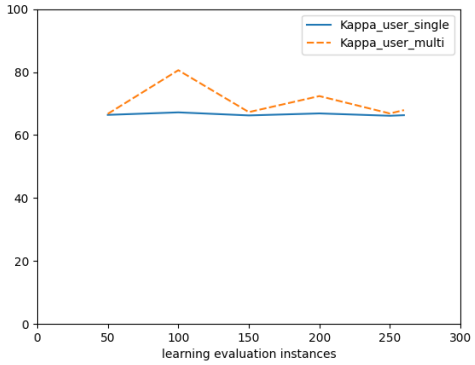

Fig. A.7. Kappa metric of the SamKnn in User #1

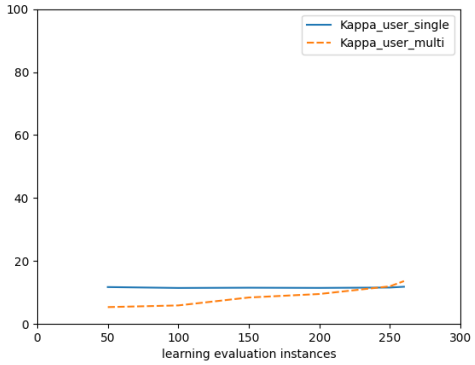

Fig. A.8. Kappa metric of the Stochastic Gradient Descendant in User #1

A.1.1.3 Recall

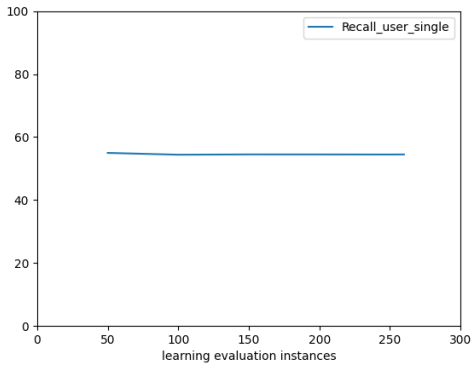

Fig. A.9. Recall of the Multi-label Adaptative Hoeffding Tree in User #1

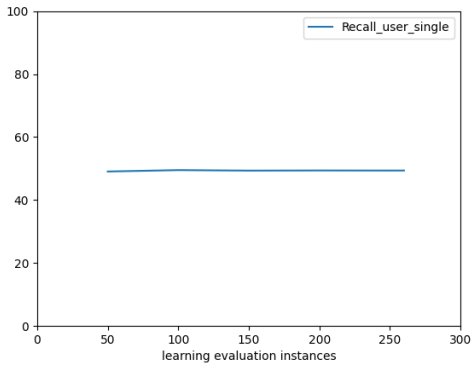

Fig. A.11. Recall of the SamKnn in User #1

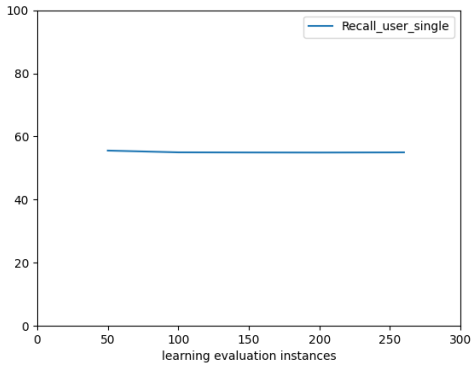

Fig. A.10. Recall of the Naive Bayes Online in User #1

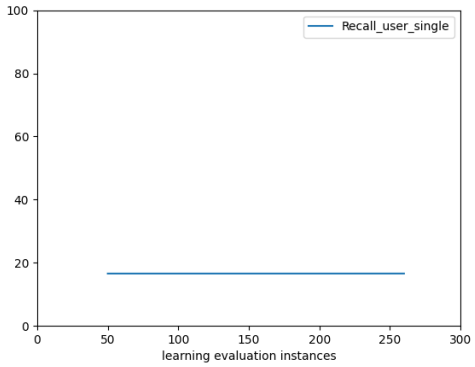

Fig. A.12. Recall of the Stochastic Gradient Descendant in User #1

A.1.1.4 Evaluation time (cpu seconds)

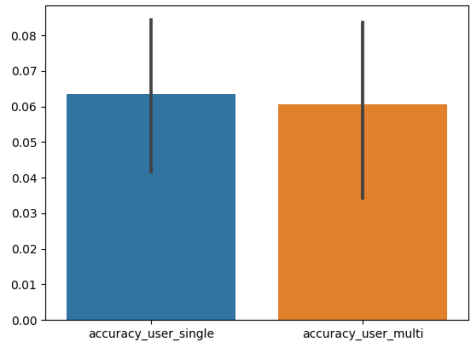

Fig. A.13. Evaluation time of the Multi-label Adaptative Hoeffding Tree in User #1

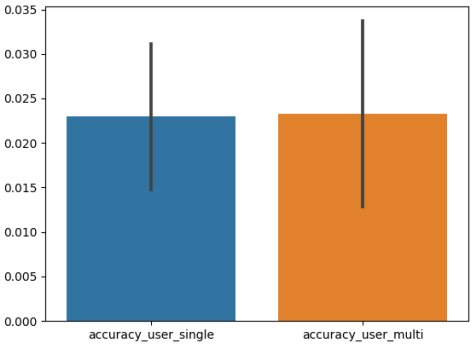

Fig. A.14. Evaluation time of the Naive Bayes Online in User #1

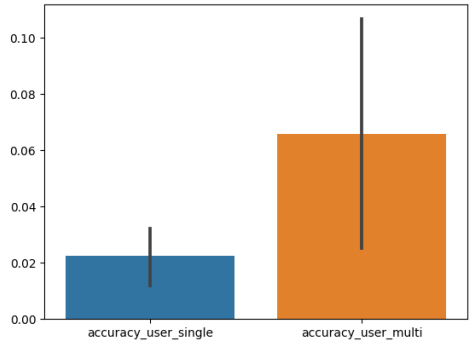

Fig. A.15. Evaluation time of the SamKnn in User #1

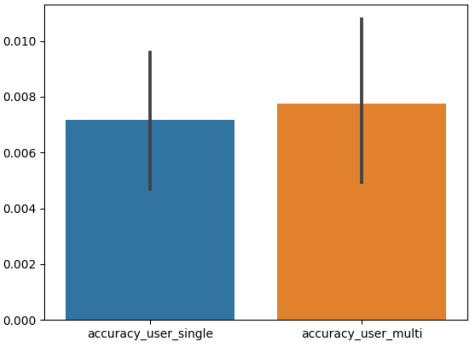

Fig. A.16. Evaluation time of the Stochastic Gradient Descendant in User #1

A.1.2 User # 2

A.1.2.1 Accuracy

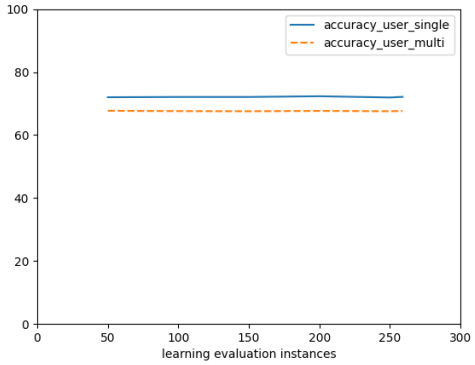

Fig. A.17. Accuracy of the Multi-label Adaptative Hoeffding Tree in the User # 2

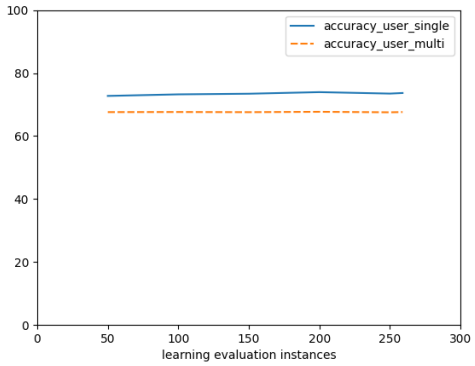

Fig. A.18. Accuracy of the Naive Bayes Online in the User # 2

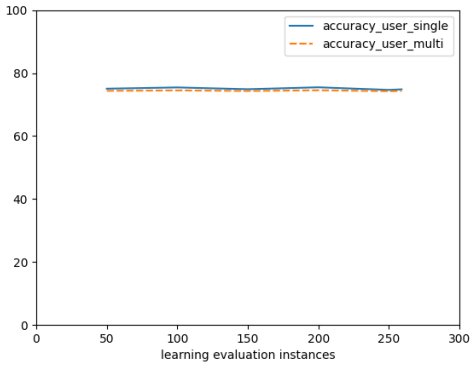

Fig. A.19. Accuracy of the SamKnn in the User #2

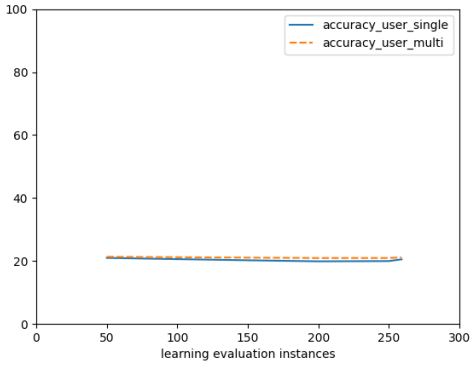

Fig. A.20. Accuracy of the Stochastic Gradient Descendant in the User #2

A.1.2.2 Kappa

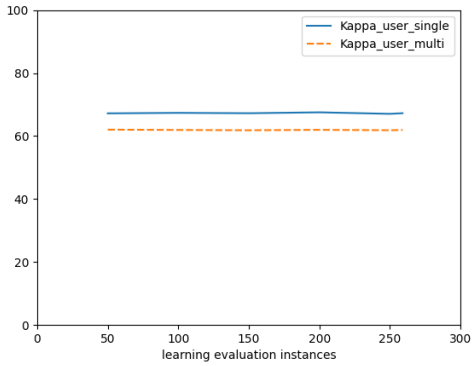

Fig. A.21. Kappa of the Multi-label Adaptative Hoeffding Tree in the User #2

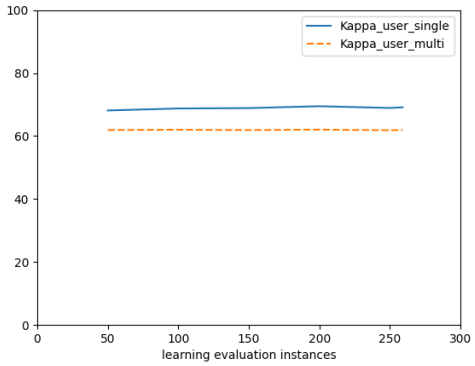

Fig. A.22. Kappa of the Naive Bayes Online in the User # 2

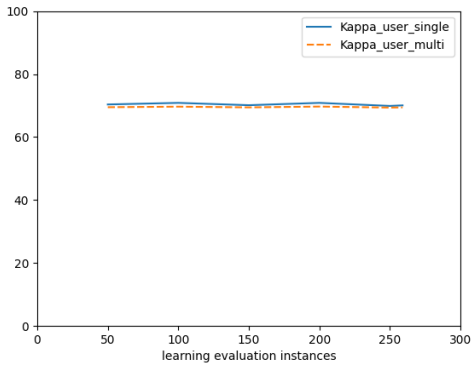

Fig. A.23. Kappa of the SamKnn in the User # 2

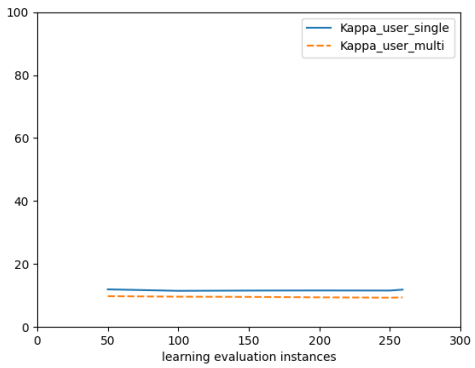

Fig. A.24. Kappa of the Stochastic Gradient Descendant in the User # 2

A.1.2.3 Recall

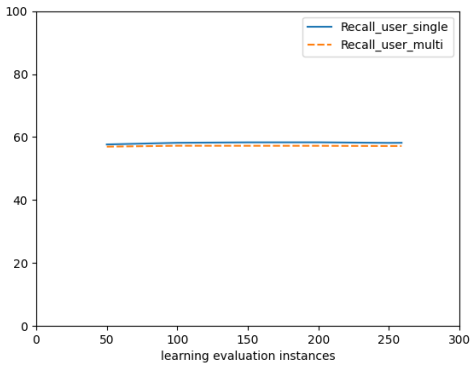

Fig. A.25. Recall of the Multi-label Adaptative Hoeffding Tree in the User # 2

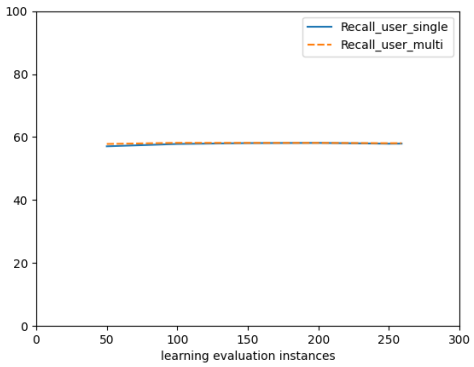

Fig. A.26. Recall of the Naive Bayes Online in the User # 2

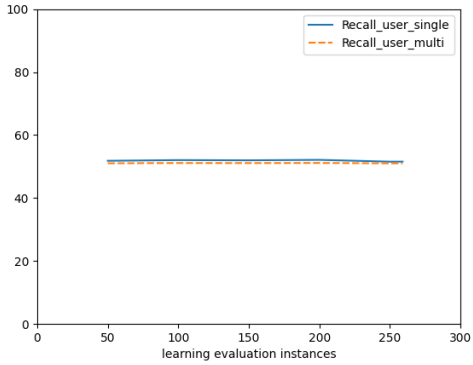

Fig. A.27. Recall of the SamKnn in the User # 2

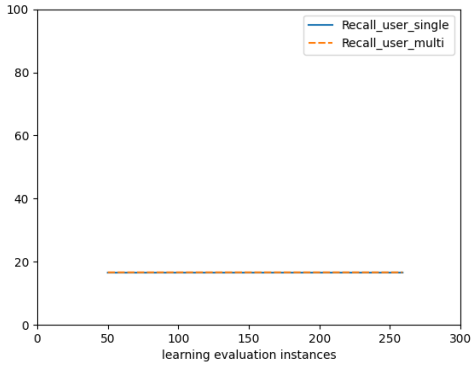

Fig. A.28. Recall of the Stochastic Gradient Descendant in the User # 2

A.1.2.4 Evaluation time (cpu seconds)

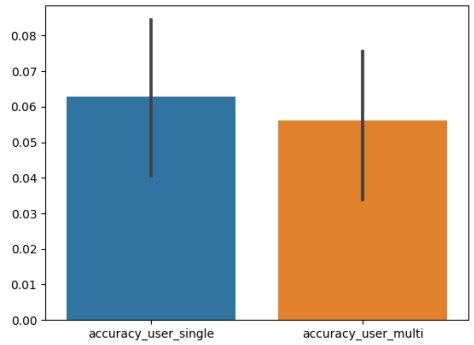

Fig. A.29. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User # 2

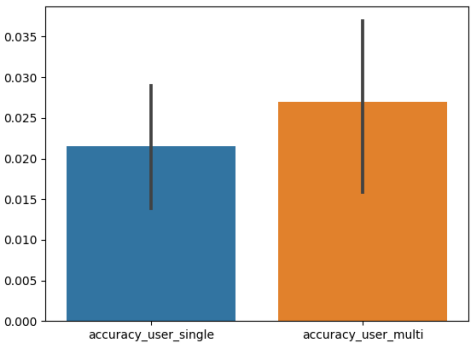

Fig. A.30. Evaluation time of the Naive Bayes Online in the User # 2

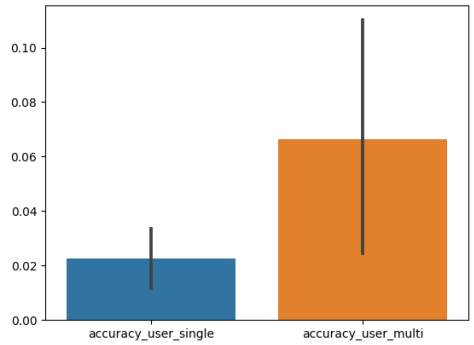

Fig. A.31. Evaluation time of the SamKnn in the User # 2

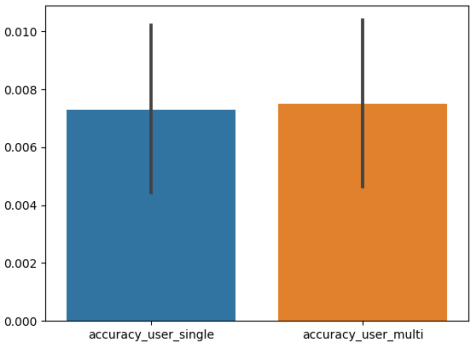

Fig. A.32. Evaluation time of the Stochastic Gradient Descendant in the User # 2

A.1.3 User # 3

A.1.3.1 Accuracy

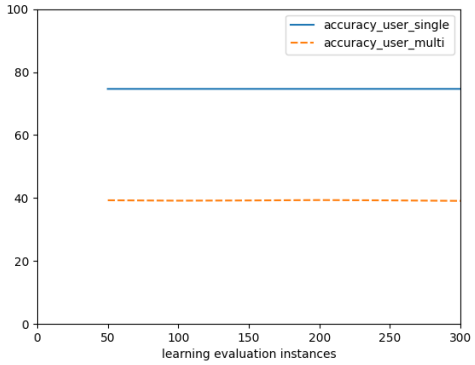

Fig. A.33. Accuracy of the Multi-label Adaptative Hoeffding Tree in the User # 3

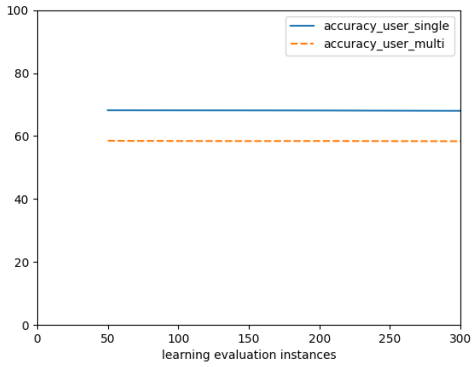

Fig. A.34. Naive Bayes Online in the User # 3

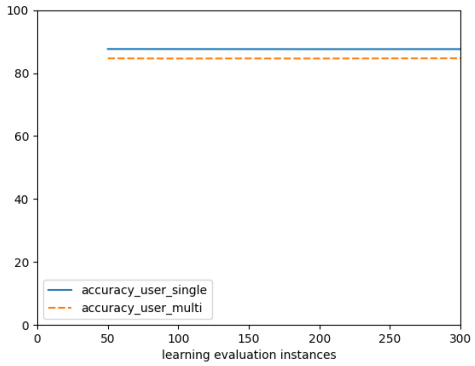

Fig. A.35. Accuracy of the SamKnn in the User # 3

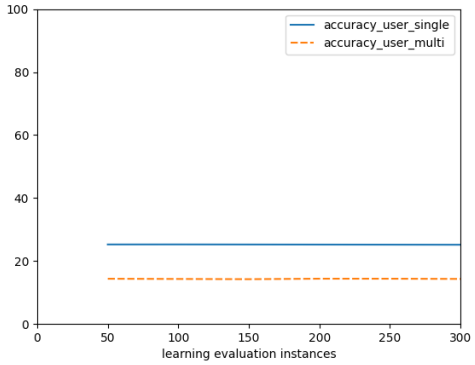

Fig. A.36. Accuracy of the Stochastic Gradient Descendant in the User # 3

A.1.3.2 Kappa

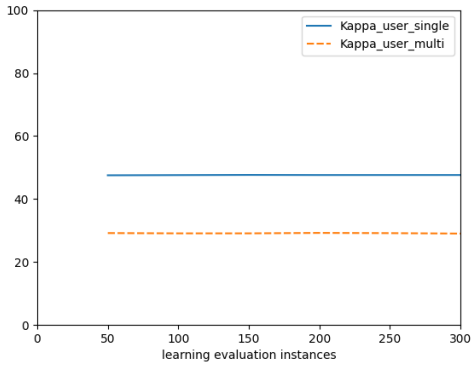

Fig. A.37. Kappa of the Multi-label Adaptative Hoeffding Tree in the User # 3

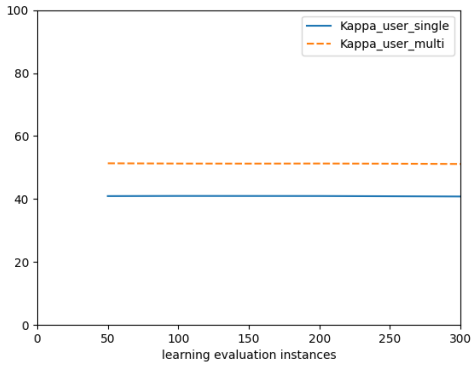

Fig. A.38. Kappa of the Naive Bayes Online in the User # 3

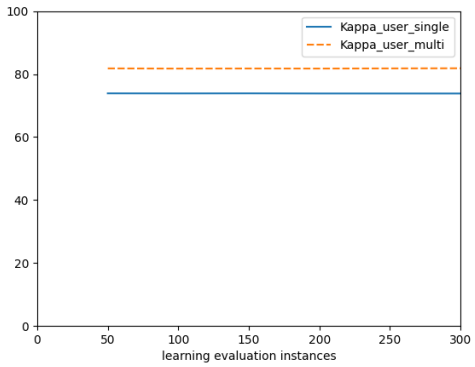

Fig. A.39. Kappa of the SamKnn in the User # 3

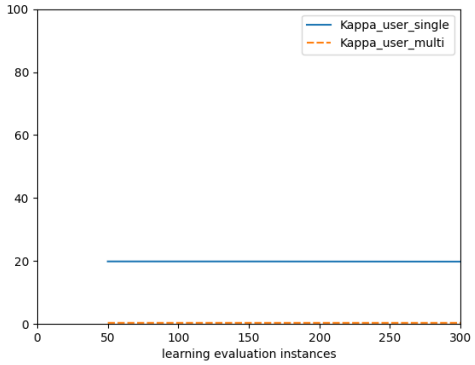

Fig. A.40. Kappa of the Stochastic Gradient Descendant in the User # 3

A.1.3.3 Recall

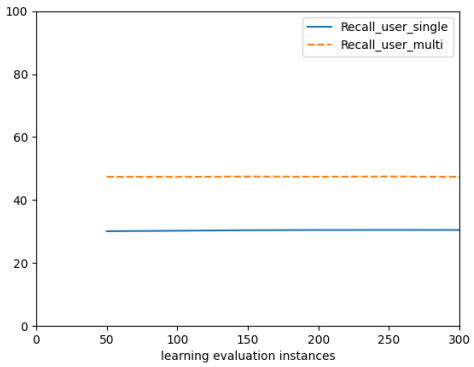

Fig. A.41. Recall of the Multi-label Adaptative Hoeffding Tree in the User # 3

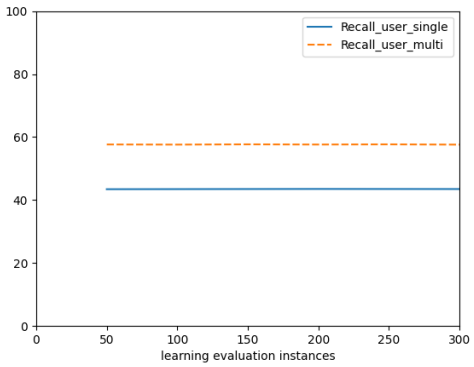

Fig. A.42. Recall of the Naive Bayes Online in the User # 3

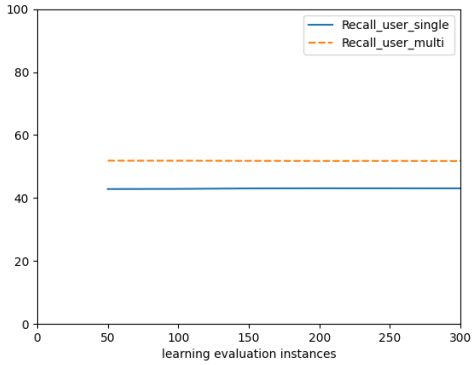

Fig. A.43. Recall of the SamKnn in the User # 3

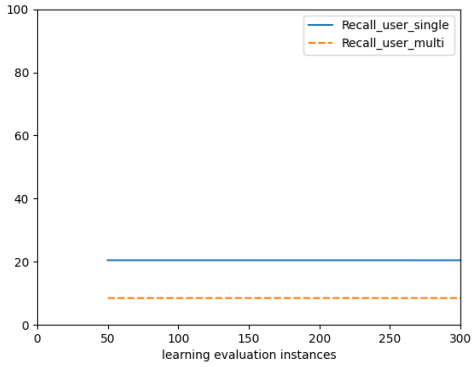

Fig. A.44. Recall of the Stochastic Gradient Descendant in the User # 3

A.1.3.4 Evaluation time (cpu seconds)

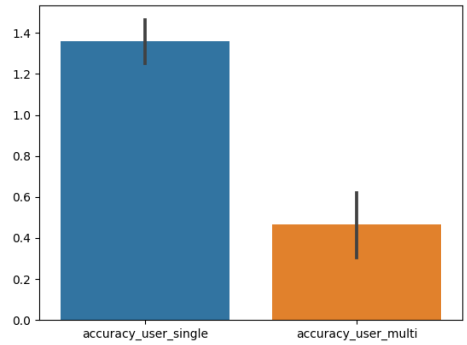

Fig. A.45. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User # 3

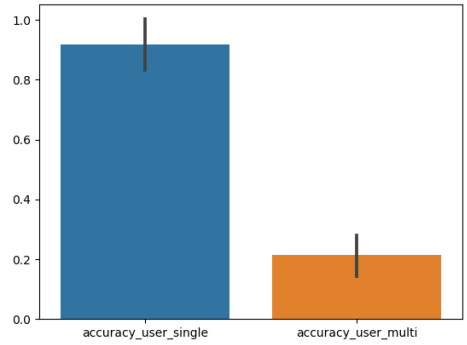

Fig. A.46. Evaluation time of the Naive Bayes Online in the User # 3

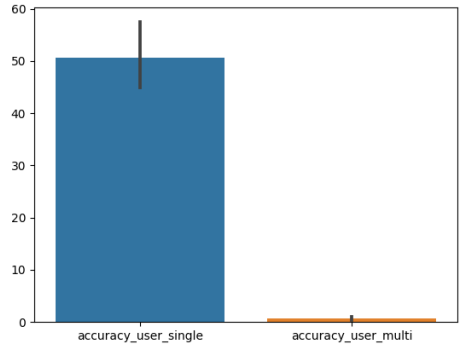

Fig. A.47. Evaluation time of the SamKnn in the User # 3

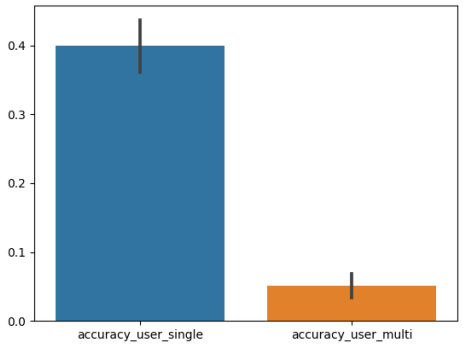

Fig. A.48. Evaluation time of the Stochastic Gradient Descendant in the User # 3

A.1.4 User # 4

A.1.4.1 Accuracy

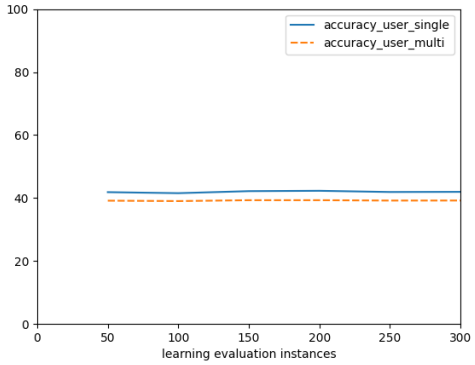

Fig. A.49. Accuracy of the Multi-label Adaptative Hoeffding Tree in the User # 4

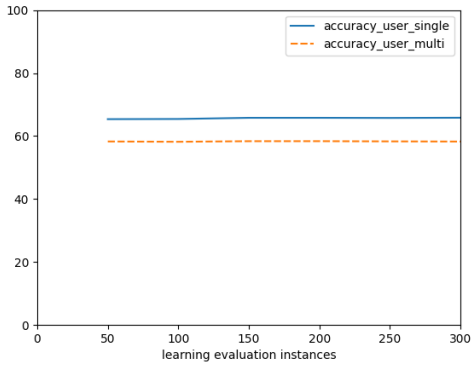

Fig. A.50. Accuracy of the Naive Bayes Online in the User # 4

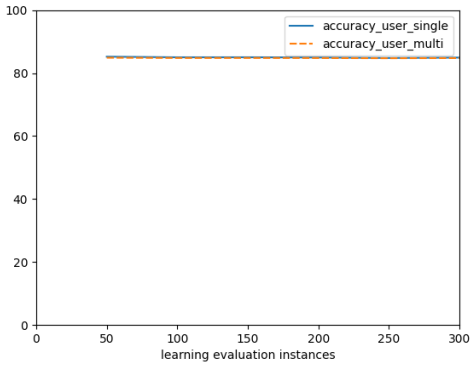

Fig. A.51. Accuracy of the SamKnn in the User # 4

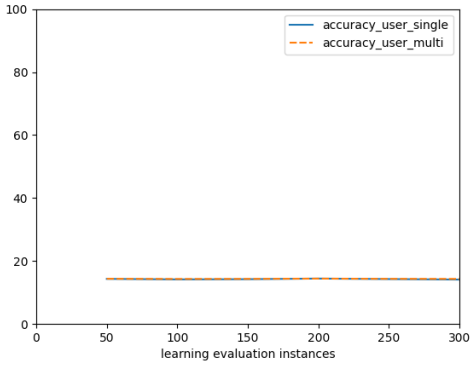

Fig. A.52. Accuracy of the Stochastic Gradient Descendant in the User # 4

A.1.4.2 Kappa

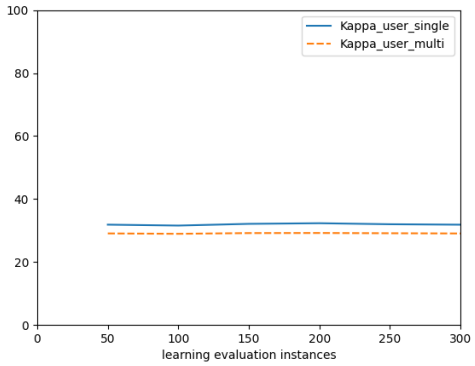

Fig. A.53. Kappa of the Multi-label Adaptative Hoeffding Tree in the User # 4

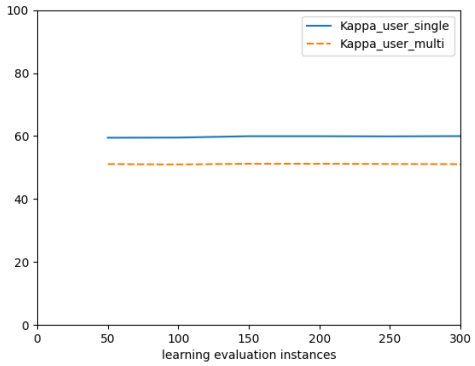

Fig. A.54. Kappa of the Online Naive Bayes in the User # 4

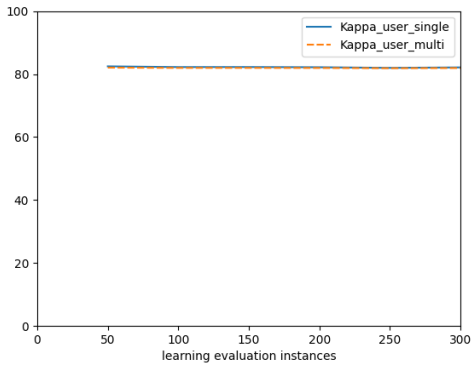

Fig. A.55. Kappa of the SamKnn in the User # 4

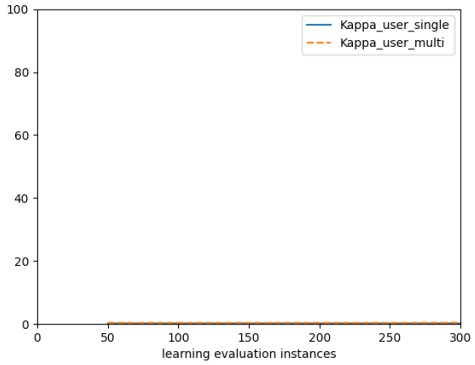

Fig. A.56. Kappa of the Stochastic Gradient Descendant in the User # 4

A.1.4.3 Recall

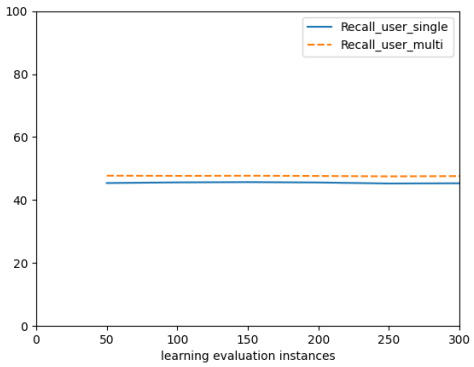

Fig. A.57. Recall of the Multi-label Adaptative Hoeffding Tree in the User # 4

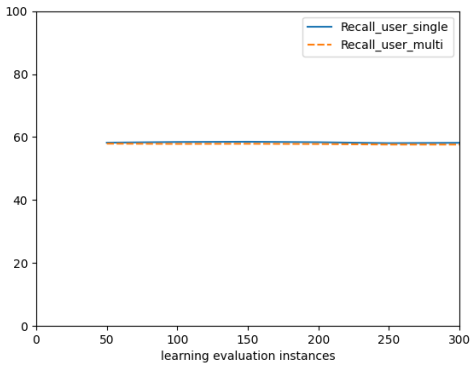

Fig. A.58. Recall of the Naive Bayes Online in the User # 4

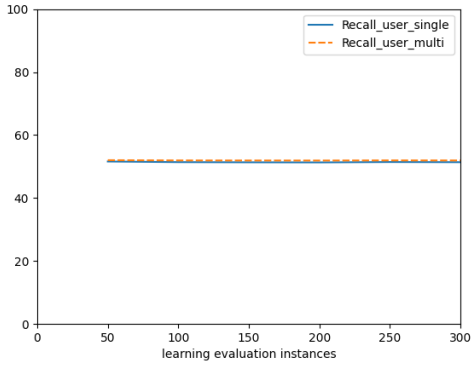

Fig. A.59. Recall of the SamKnn in the User # 4

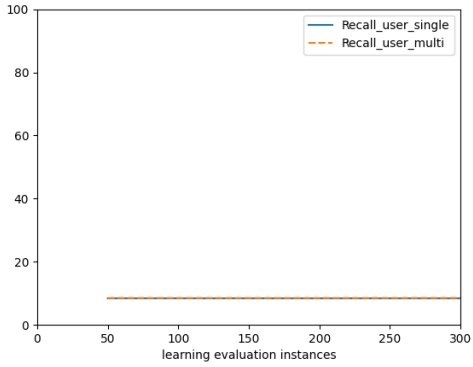

Fig. A.60. Recall of the Stochastic Gradient Descendant in the User # 4

A.1.4.4 Evaluation time (cpu seconds)

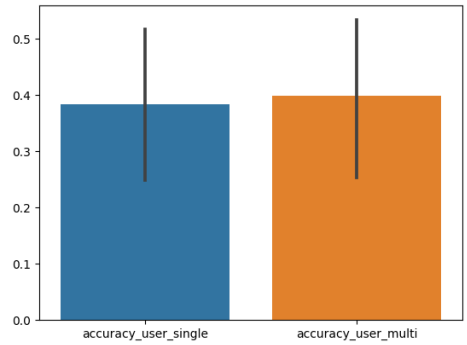

Fig. A.61. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User # 4

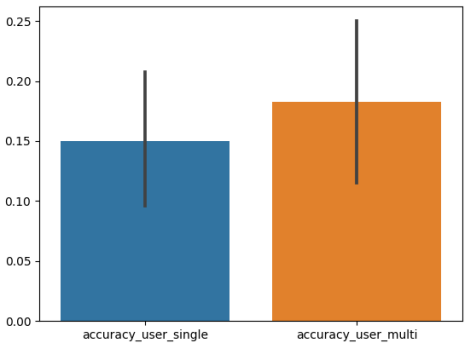

Fig. A.62. Evaluation time of the Naive Bayes Online in the User # 4

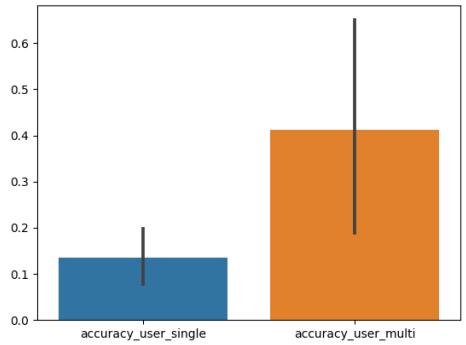

Fig. A.63. Evaluation time of the SamKnn in the User # 4

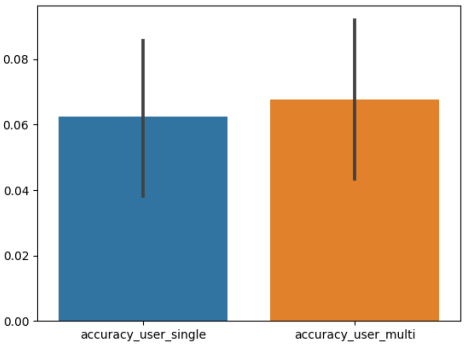

Fig. A.64. Evaluation time of the Stochastic Gradient Descendant in the User # 4

A.1.5 User # 5

A.1.5.1 Accuracy

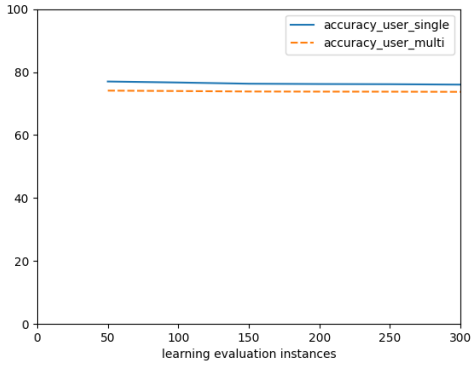

Fig. A.65. Multi-label Adaptative Hoeffding Tree in the User # 5

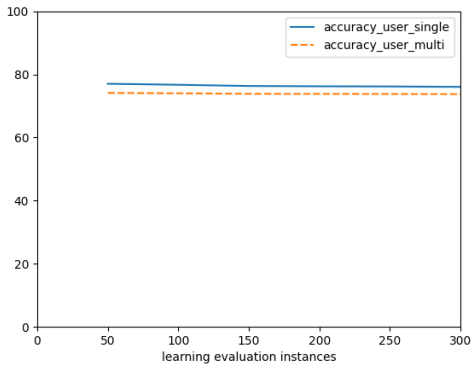

Fig. A.66. Naive Bayes Online in the User # 5

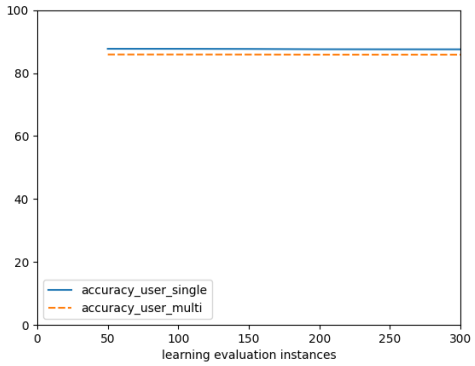

Fig. A.67. SamKnn in the User # 5

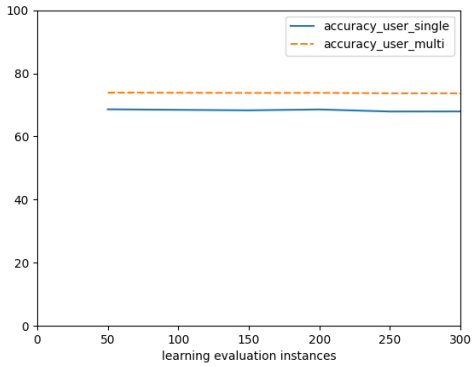

Fig. A.68. Stochastic Gradient Descendant in the User # 5

A.1.5.2 Kappa

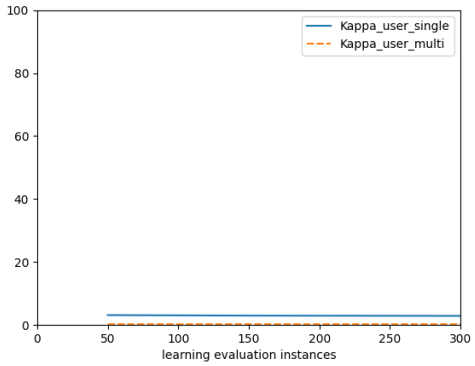

Fig. A.69. Kappa metric of the Multi-label Adaptative Hoeffding Tree in the User # 5

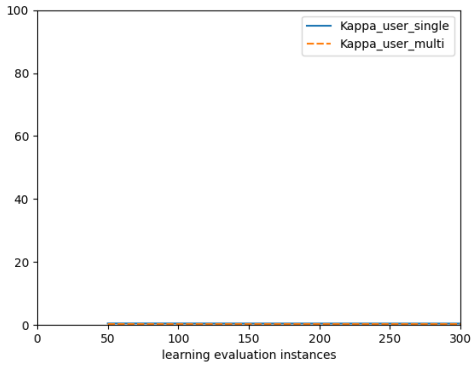

Fig. A.70. Kappa metric of the Naive Bayes Online in the User # 5

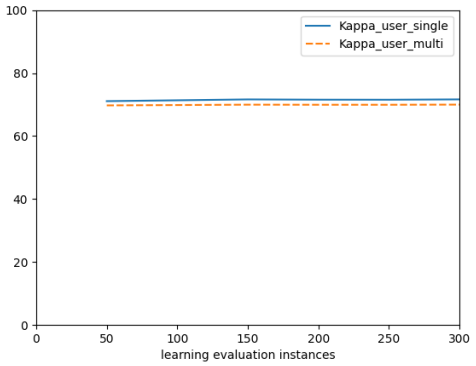

Fig. A.71. Kappa metric of the SamKNN in the User # 5

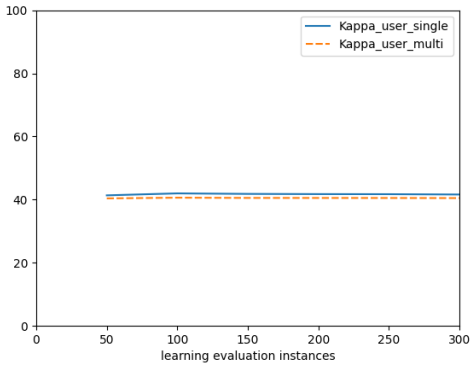

Fig. A.72. Kappa metric of the Stochastic Gradient Descendant in the User # 5

A.1.5.3 Recall

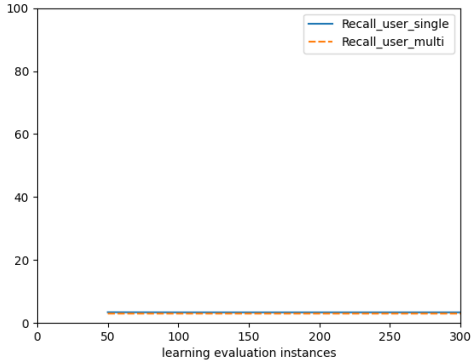

Fig. A.73. Recall of the Multi-label Adaptative Hoeffding Tree in the User # 5

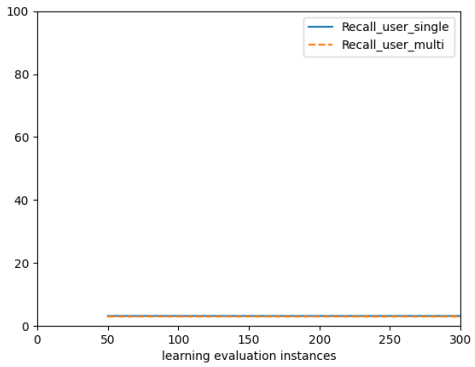

Fig. A.74. Recall of the Naive Bayes Online in the User # 5

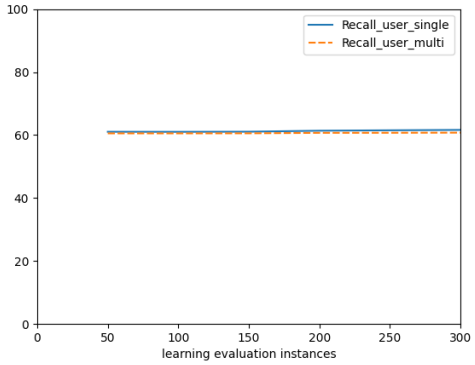

Fig. A.75. Recall of the SamKnn in the User # 5

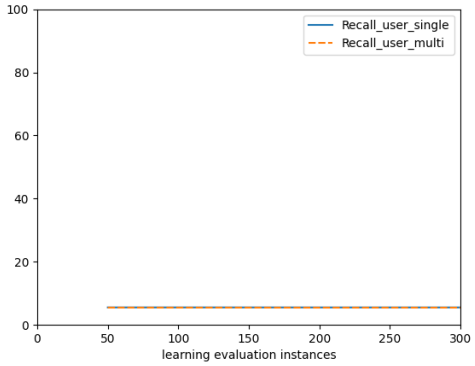

Fig. A.76. Recall of the Stochastic Gradient Descendant in the User # 5

A.1.5.4 Evaluation time (cpu seconds)

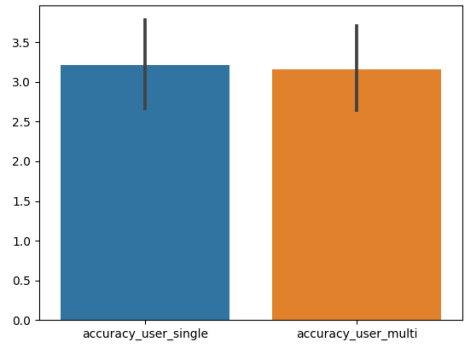

Fig. A.77. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User # 5

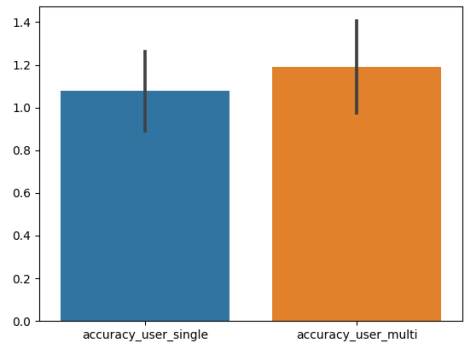

Fig. A.78. Evaluation time of the Naive Bayes Online in the User # 5

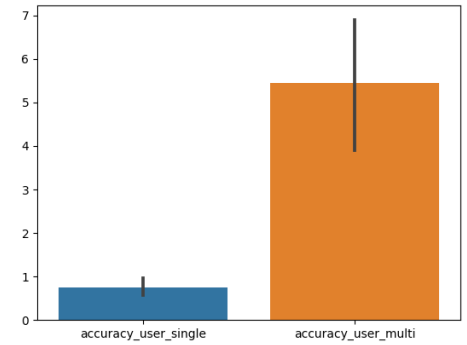

Fig. A.79. Evaluation time of the SamKnn in the User # 5

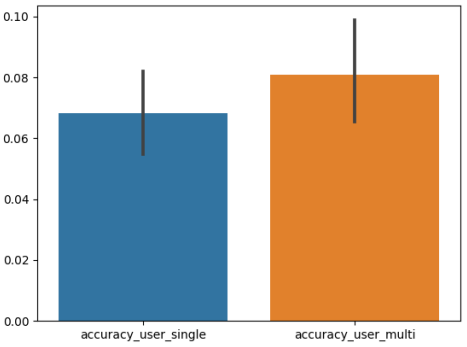

Fig. A.80. Evaluation time of the Stochastic Gradient Descendant in the User # 5

A.1.6 User # 6

A.1.6.1 Accuracy

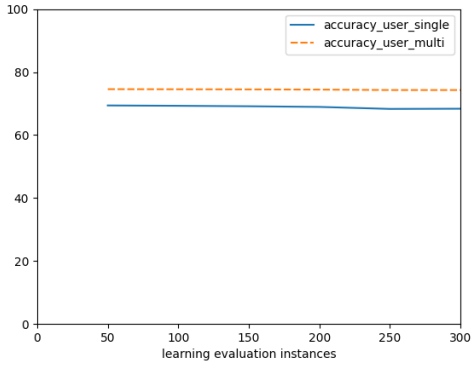

Fig. A.81. Accuracy of the Multi-label Adaptative Hoeffding Tree in the User # 6

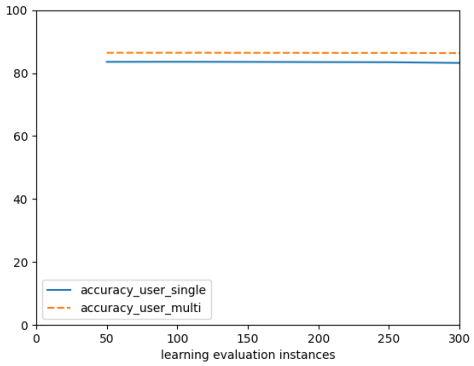

Fig. A.83. Accuracy of the SamKnn in the User # 6

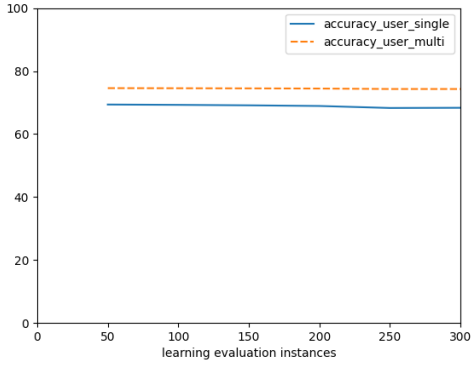

Fig. A.82. Accuracy of the Naive Bayes Online

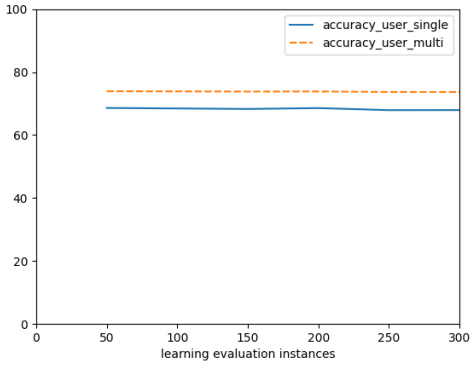

Fig. A.84. Accuracy of the Stochastic Gradient Descendant in the User # 6

A.1.6.2 Kappa

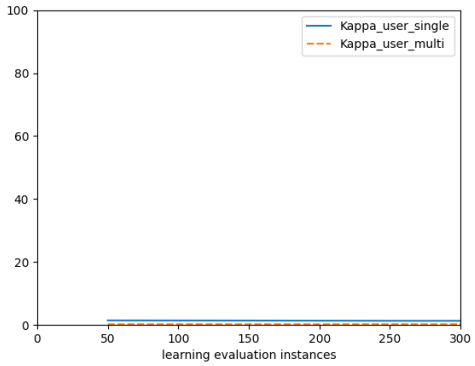

Fig. A.85. Kappa of the Multi-label Adaptative Hoeffding Tree in the User # 6

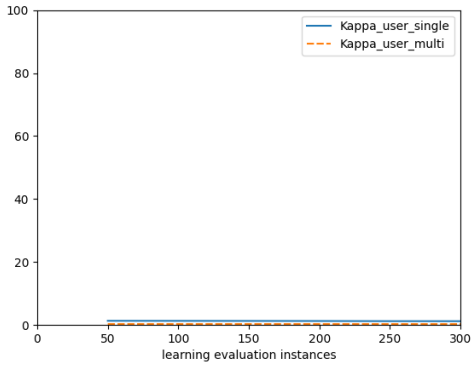

Fig. A.86. Kappa of the Naive Bayes Online in the User # 6

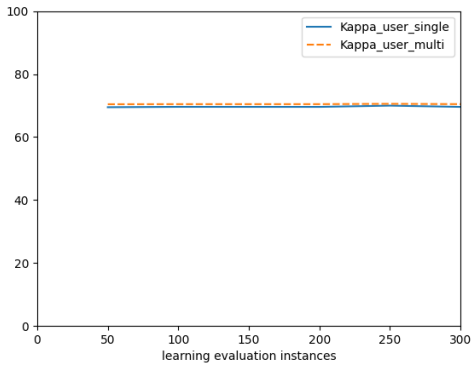

Fig. A.87. Kappa of the SamKnn in the User # 6

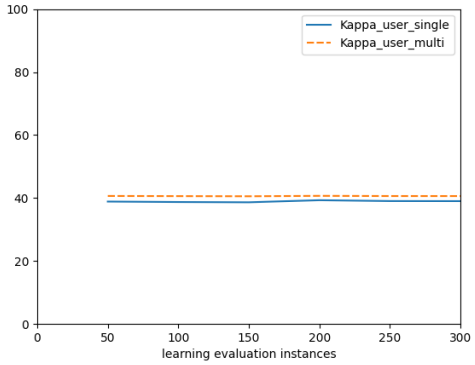

Fig. A.88. Kappa of the Stochastic Gradient Descendant in the User # 6

A.1.6.3 Recall

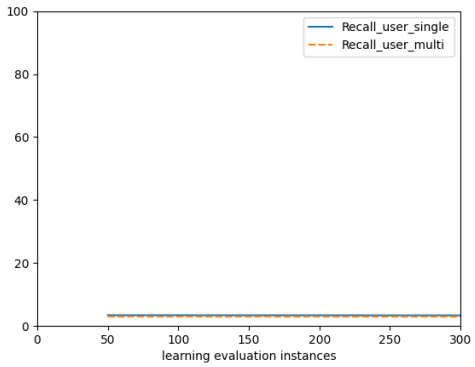

Fig. A.89. Recall of the Multi-label Adaptative Hoeffding Tree in the User # 6

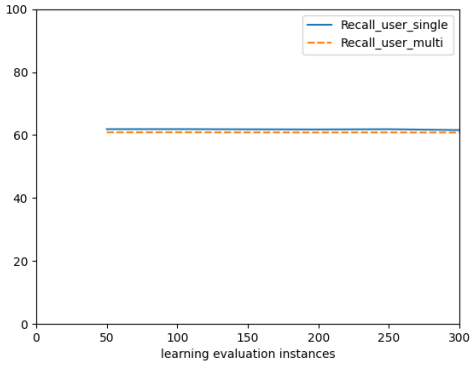

Fig. A.91. Recall of the SamKnn in the User # 6

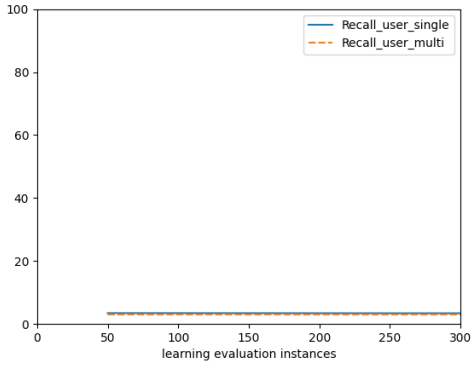

Fig. A.90. Recall of the Naive Bayes Online in the User # 6

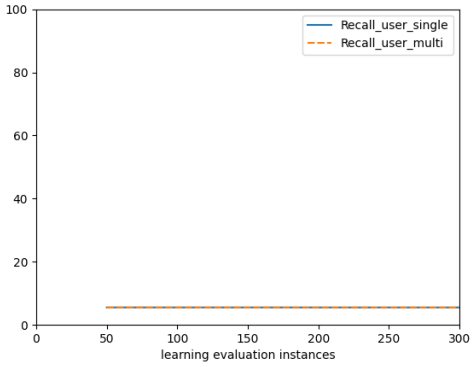

Fig. A.92. Recall of the Stochastic Gradient Descendant in the User # 6

A.1.6.4 Evaluation time (cpu seconds)

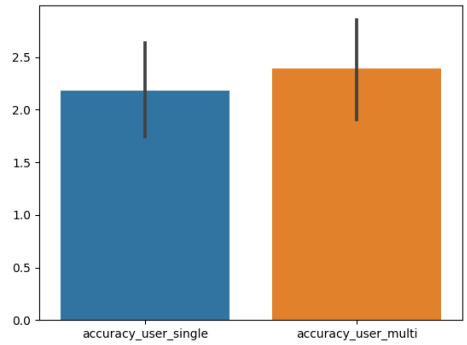

Fig. A.93. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User # 6

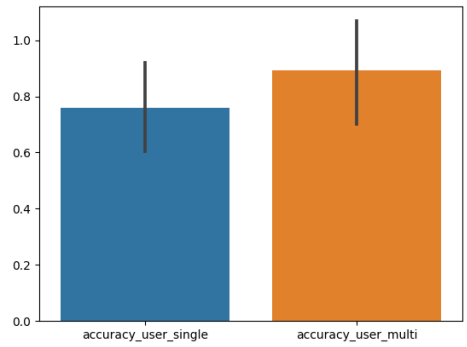

Fig. A.94. Evaluation time of the Naive Bayes Online in the User # 6

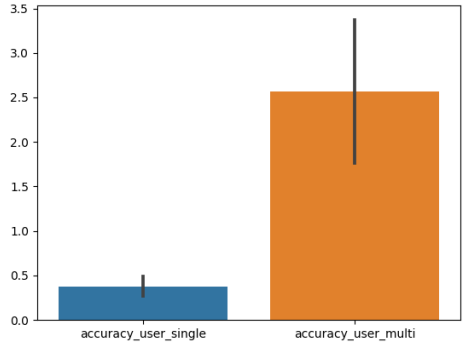

Fig. A.95. Evaluation time of the SamKnn in the User # 6

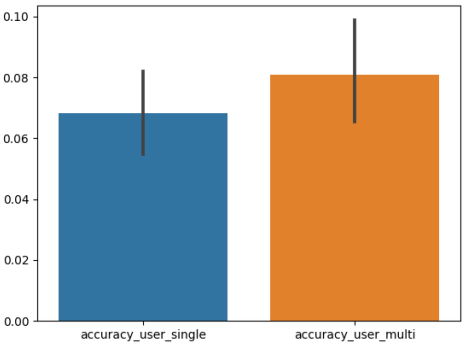

Fig. A.96. Evaluation time of the Stochastic Gradient Descendant in the User # 6

A.1.7 User # 7

A.1.7.1 Accuracy

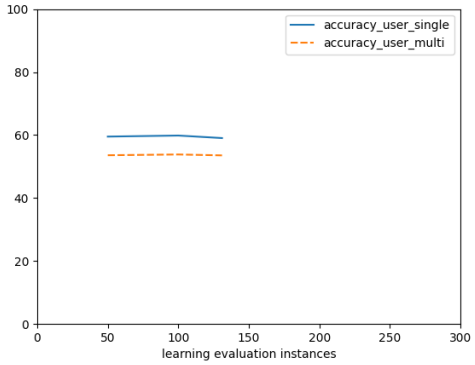

Fig. A.97. Accuracy of the Multi-label Adaptative Hoeffding Tree in the User # 7

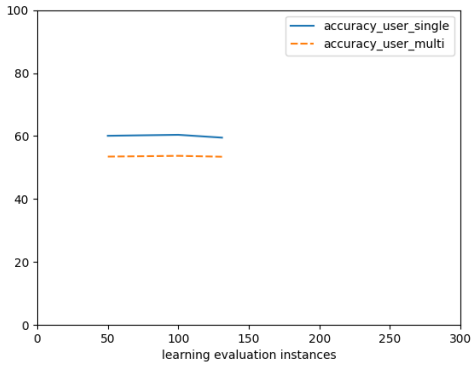

Fig. A.98. Accuracy of the Naive Bayes Online in the User # 7

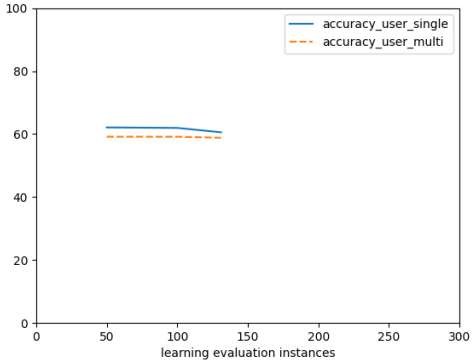

Fig. A.99. Accuracy of the SamKnn in the User #7

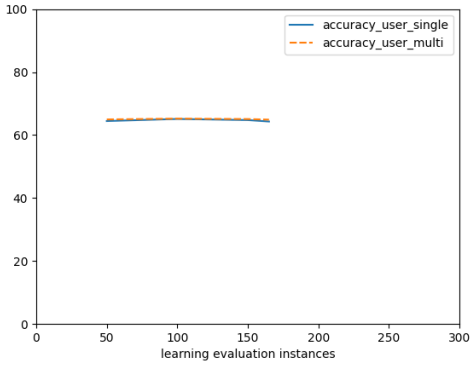

Fig. A.100. Accuracy of the Stochastic Gradient Descendant in the User #7

A.1.7.2 Kappa

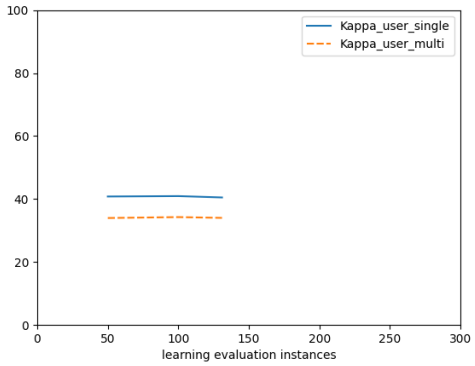

Fig. A.101. Kappa of the Multi-label Adaptative Hoeffding Tree in the User #7

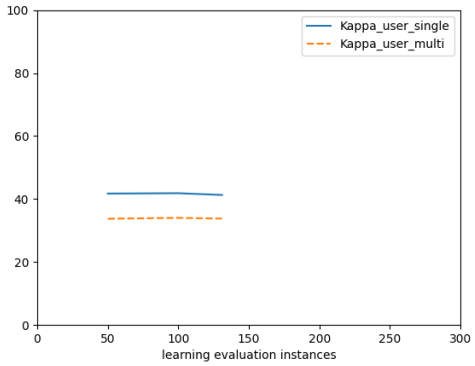

Fig. A.102. Kappa of the Naive Bayes Online in the User #7

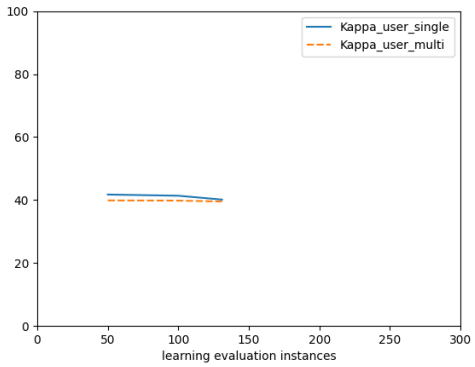

Fig. A.103. Kappa of the SamKnn in the User #7

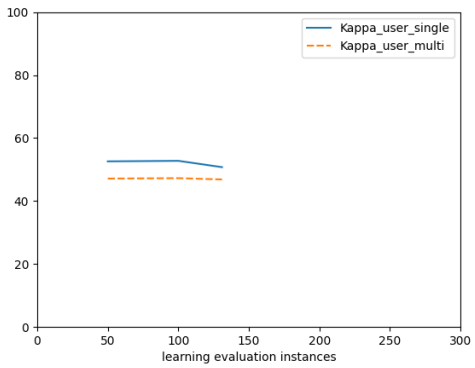

Fig. A.104. Kappa of the Stochastic Gradient Descendant in the User #7

A.1.7.3 Recall

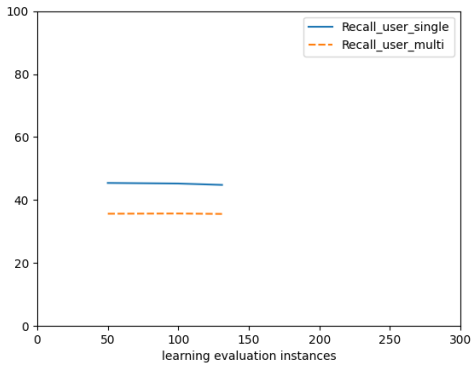

Fig. A.105. Recall of the Multi-label Adaptative Hoeffding Tree in the User #7

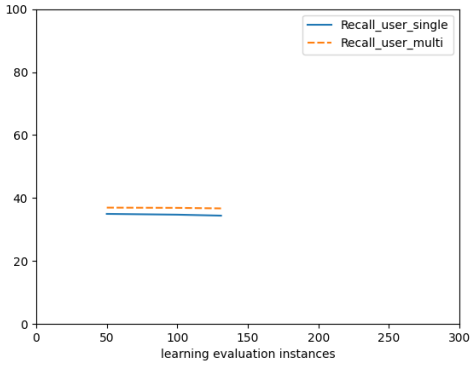

Fig. A.107. Recall of the SamKnn in the User #7

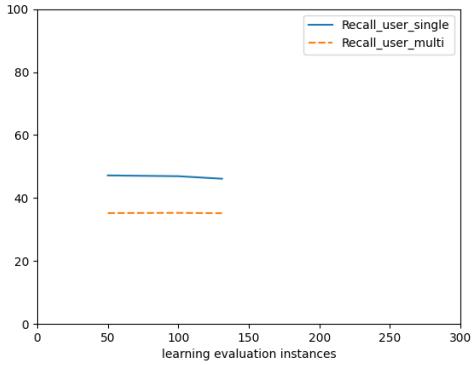

Fig. A.106. Recall of the Naive Bayes Online in the User #7

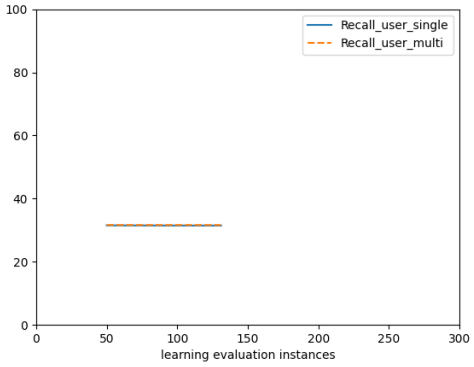

Fig. A.108. Recall of the Stochastic Gradient Descendant in the User #7

A.1.7.4 Evaluation time (cpu seconds)

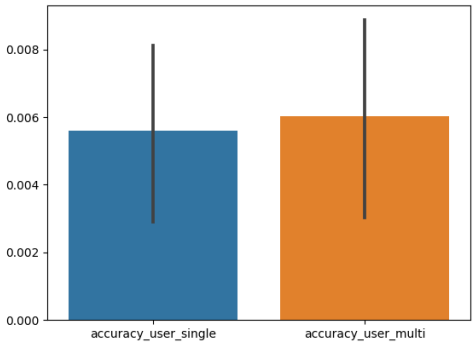

Fig. A.109. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User #7

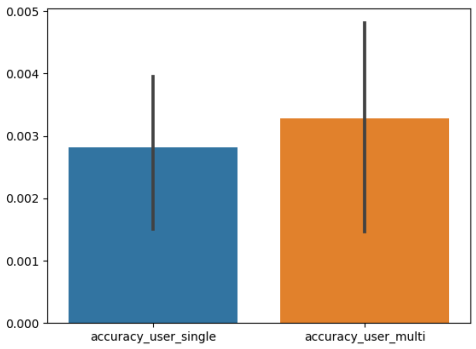

Fig. A.110. Evaluation time of the Naive Bayes Online in the User #7

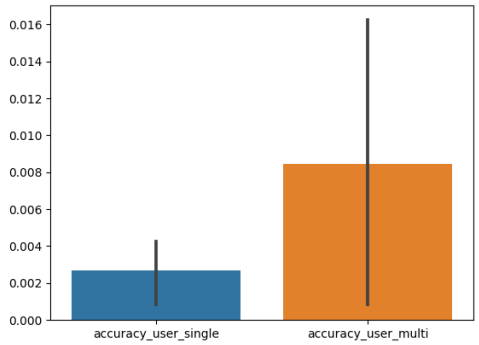

Fig. A.111. Evaluation time of the SamKnn in the User #7

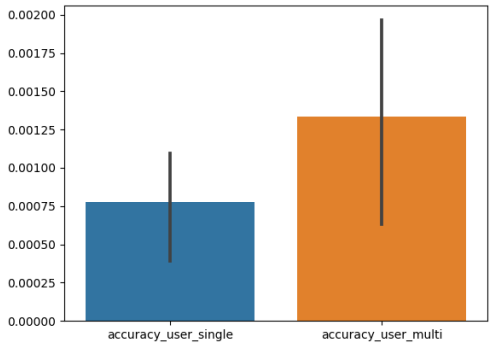

Fig. A.112. Evaluation time of the Stochastic Gradient Descendant in the User #7

A.1.8 User #8

A.1.8.1 Accuracy

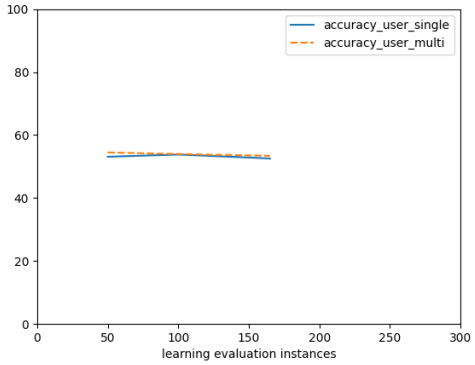

Fig. A.113. Accuracy of the Multi-label Adaptative Hoeffding Tree in the User #8

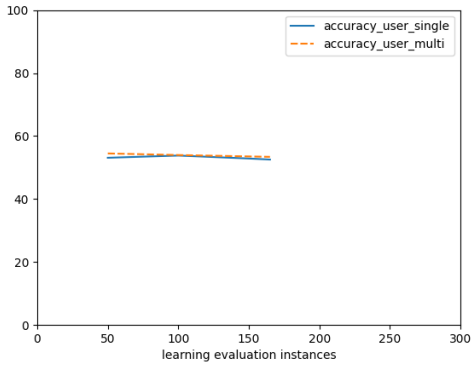

Fig. A.114. Accuracy of the Naive Bayes Online in the User #8

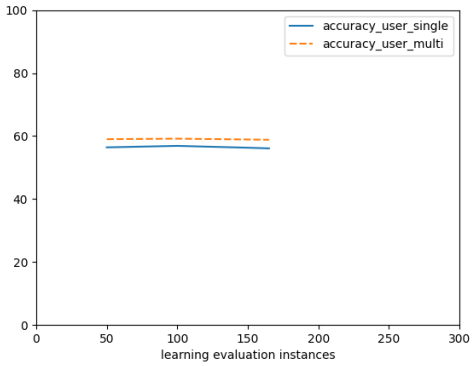

Fig. A.115. Accuracy of the SamKnn in the User #8

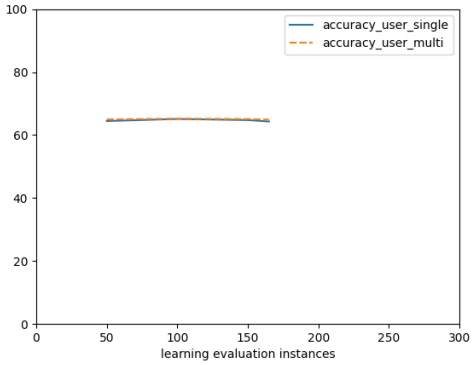

Fig. A.116. Accuracy of the Stochastic Gradient Descendant in the User #8

A.1.8.2 Kappa

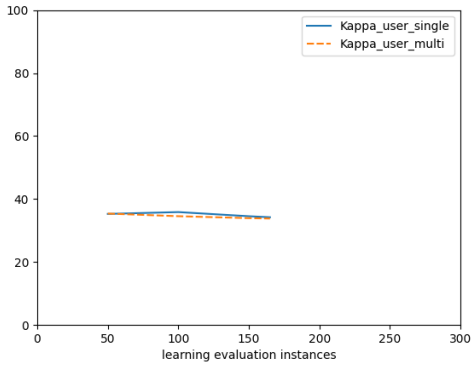

Fig. A.117. Kappa of the Multi-label Adaptative Hoeffding Tree in the User #8

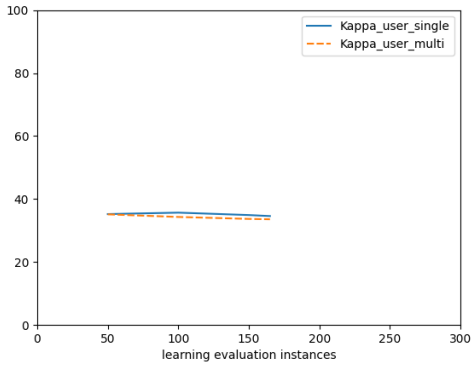

Fig. A.118. Kappa of the Naive Bayes Online in the User #8

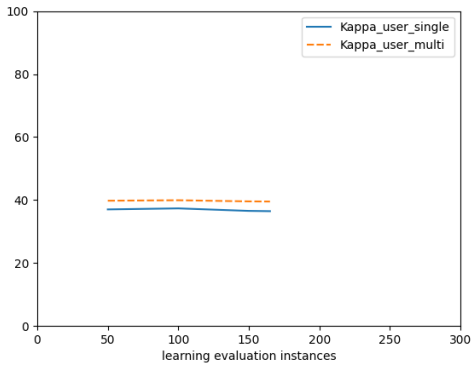

Fig. A.119. Kappa of the SamKnn in the User #8

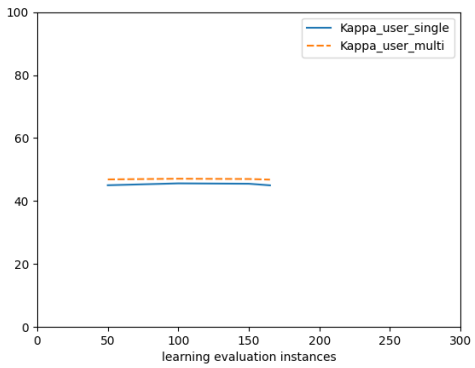

Fig. A.120. Kappa of the Stochastic Gradient Descendant in the User #8

A.1.8.3 Recall

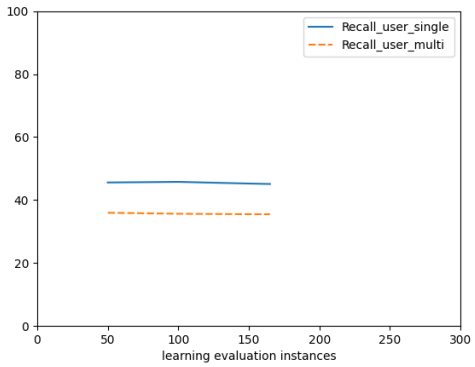

Fig. A.121. Recall of the Multi-label Adaptative Hoeffding Tree in the User #8

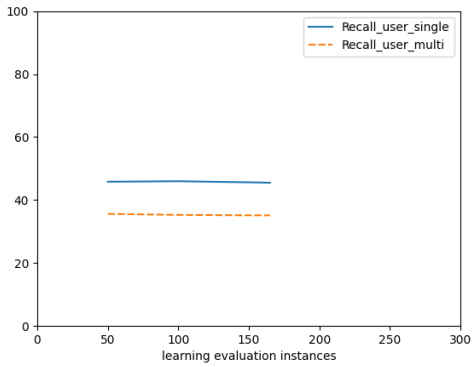

Fig. A.122. Recall of the Naive Bayes Online in the User #8

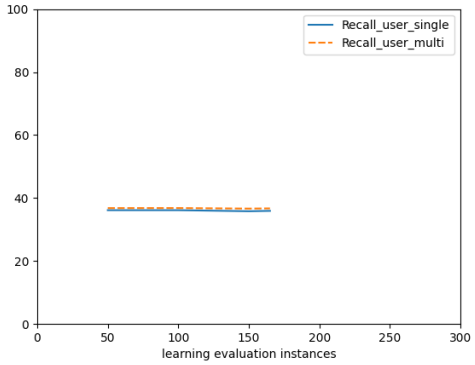

Fig. A.123. Recall of SamKnn in the User #8

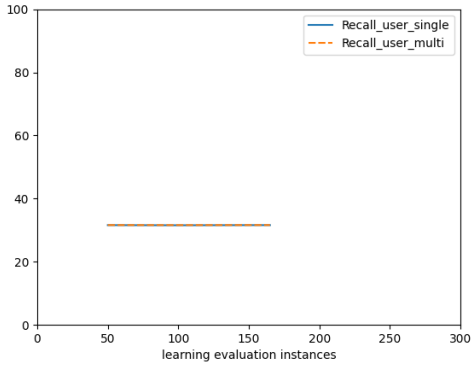

Fig. A.124. Recall of the Stochastic Gradient Descendant in the User #8

A.1.8.4 Evaluation time (cpu seconds)

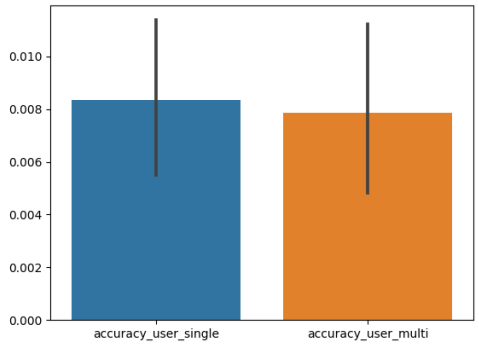

Fig. A.125. Evaluation time of the Multi-label Adaptative Hoeffding Tree in the User #8

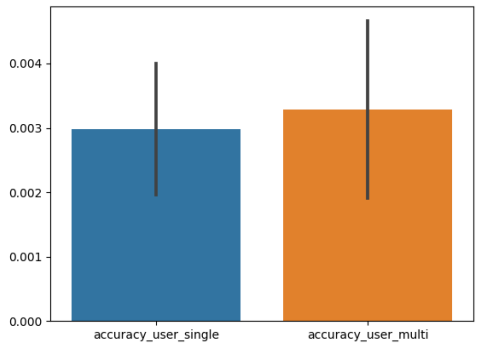

Fig. A.126. Evaluation time of the Naive Bayes Online in the User #8

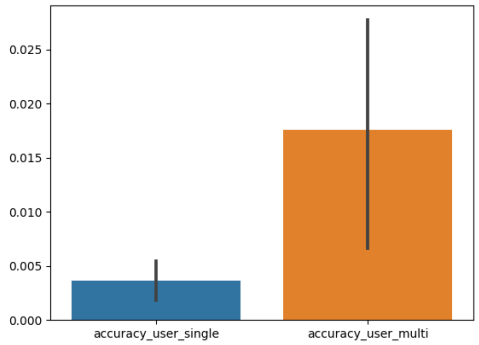

Fig. A.127. Evaluation time of SamKnn in the User #8

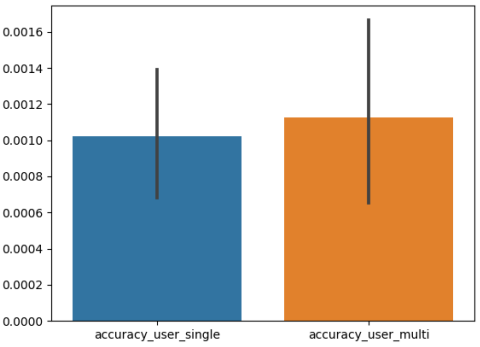

Fig. A.128. Evaluation time of the Stochastic Gradient Descendant in the User #8

ANNEX A – Wekaclassifier Package

Wekaclassifier is a meta-classifier to use classifiers from WEKA API. Figure A.1 shows a example of workflow of a model.

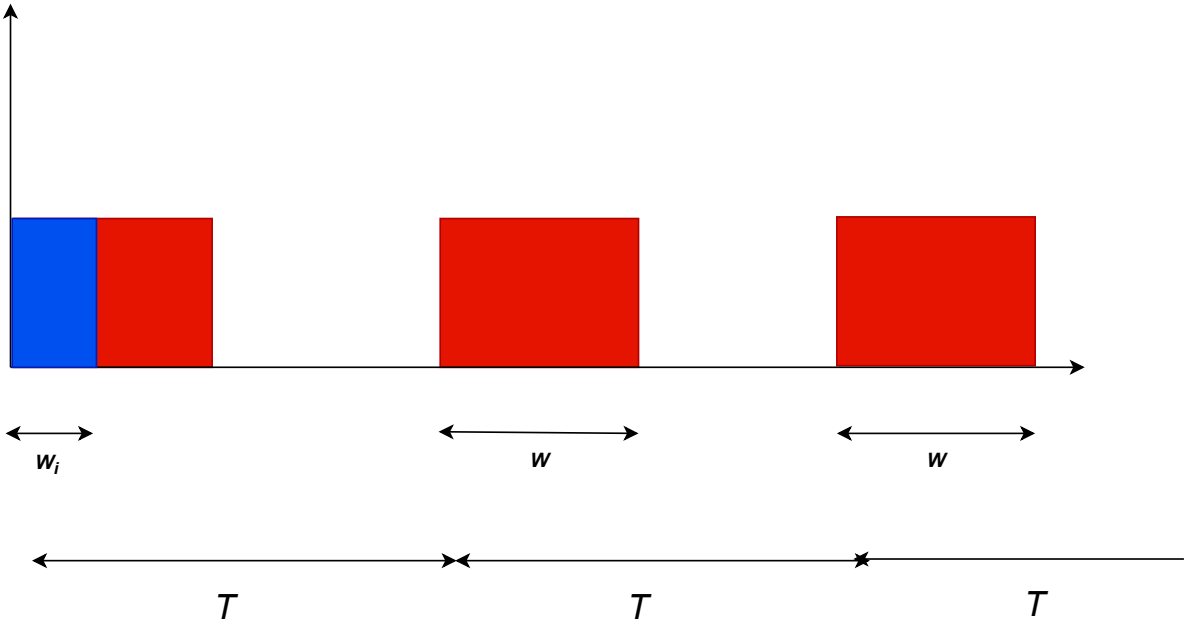

Fig. A.1. Workflow of a model built by WekaClassifier.

WEKAClassifier builds a model of W instances every T instances only for non incremental methods (In our study, all models are not incremental). For incremental methods, WEKAclassifier is trained for every instance. In W_i, the value i means the size of first Window for training learner. The implementations is available at GitHub[1].

[1] https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/classifiers/meta/WEKAClassifier.java