Pontifícia Universidade Católica
do Rio de Janeiro

Reinier Morejón Novales

A multi-agent approach to data mining processes: Applications to health care

Dissertação de Mestrado

Dissertation presented to the Programa de Pós-graduacão em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Carlos José Pereira de Lucena

Rio de Janeiro

April 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Reinier Morejón Novales

A multi-agent approach to data mining processes: Applications to health care

Dissertation presented to the Programa de Pós-graduacão em Informática of PUC–Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the undersigned Examination Committee.

Prof. Carlos José Pereira de Lucena
Advisor
Departamento de Informática – PUC-Rio

Profa. Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Dr. Marx Leles Viana
Pesquisador Autônomo

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Tecnico Cientifico – PUC-Rio

Rio de Janeiro, April 6th, 2018

Reinier Morejón Novales

Computer Science Engineer, University of Informatics Sciences 2009. Havana, Cuba. He is a researcher at Laboratory of Software Engineering – LES-PUC-Rio at Pontifical Catholic of Rio de Janeiro since 2016. His main studies are related to the area of software engineering and multi-agent systems.

Bibliographic data

Morejón Novales, Reinier

A multi-agent approach to data mining processes: Applications to health care. / Reinier Morejón Novales; advisor: Carlos José Pereira de Lucena. Rio de Janeiro: PUC-Rio, Departamento de Informática, 2018.

61 f. : il. (color); 30 cm

1. Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de informática.

Incluí bibliografia.

1. Informática – Teses. 2. Sistema Multiagente. 3. Engenharia de Software. 4 Mineração de dados. 5 Aprendizagem de Máquina I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my mother,
the best soul I met in my life,
who deserves all the good things in this world
and the best of me,
always.

Acknowledgements

I would first like to thank my advisor Prof. Carlos José Pereira de Lucena of the Departamento de Informática at PUC-Rio. The door to Prof. Lucena office was always open whenever I had a question about my research; even in times when he had a delicate health condition.

Second, I would like to thank my friend Dr. Marx Leles Viana, Autonomous Researcher. He always was available too whenever I needed to ask any question about my research and also about other matters of life beyond research, as a true friend does. He consistently steered me in the right direction whenever he thought I needed it.

I would also like to thank every person that one way or another helped me to reach this point today. Especially those friends and family members that believed in my ideas and provided me the financial aid and emotional support to left my country behind looking for a new start. They still support me today and I will always be grateful for that.

I must also express my very profound gratitude to my parents and my little brother for giving me the love and the strength I need every day to fight for a better future, even when I am more than 4000 miles away from them. Also, my very special gratitude to Monica, my partner, my confidant, my love; who always provides me with unfailing support and continuous encouragement and with the happiness of sharing with a very good family in the absence of my own. This accomplishment would not have been possible without them. Thank you.

Finally, I want to thank CAPES for funding this research.

Abstract

Morejón Novales, Reinier; Lucena, Carlos José Pereira de (Advisor). **A multi-agent approach to data mining processes: Applications to health care.** Rio de Janeiro, 2018. 61p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Data mining is a hot topic that attracts researchers from different areas, such as databases, machine learning, and multi-agent systems. As a consequence of the growth of data volume, there is a growing need to obtain knowledge from these large data sets that are very difficult to handle and process with traditional methods. Software agents can play a significant role performing data mining processes in ways that are more efficient. For instance, they can work to perform selection, extraction, preprocessing and integration of data as well as parallel, distributed, or multisource mining. This work proposes an approach (in the form of a framework) that uses software agents to manage data mining processes. In order to test its applicability, we use several data sets related to health care domain representing some usage scenarios (hypothyroidism, diabetes and arrhythmia).

Keywords:

Multi-agent Systems; Data Mining; Machine Learning.

Resumo

Morejón Novales, Reinier; Lucena, Carlos José Pereira de. **Uma abordagem multiagente para processos de mineração de dados: aplicações na área da saúde.** Rio de Janeiro, 2018. 61p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A mineração de dados é um tema em alta que atrai pesquisadores de diferentes áreas, como bancos de dados, aprendizado de máquina e sistemas multiagentes. Como consequência do crescimento do volume de dados, há uma necessidade crescente de obter conhecimento desses grandes conjuntos de dados que são muito difíceis de manipular e processar com os métodos tradicionais. Os agentes de software podem desempenhar um papel significativo ao executar processos de mineração de dados de maneira mais eficiente. Por exemplo, eles podem trabalhar para realizar seleção, extração, pré-processamento e integração de dados, bem como mineração paralela, distribuída ou de múltiplas fontes. Este trabalho propõe uma abordagem (na forma de um framework) que usa agentes de software para gerenciar processos de mineração de dados. Para testar sua aplicabilidade, utilizamos vários conjuntos de dados relacionados ao domínio de saúde, representando alguns cenários de uso (hipotireoidismo, diabetes e arritmia).

Palavras-chave:

Sistemas multiagentes; mineração de dados; aprendizado de máquina.

Table of contents

List of Figures

List of Tables

1 Introduction

Since its origins, computer science has always represented an accelerated path of constant evolution and transformation, at the same time implying a similar progression of information generation. Beyond the spectacular growth of traditional internet usage (Miniwatts, 2017), we saw strong evidences of such data generation rates in the equally outstanding rise of the Internet of Things (Columbus, 2016; Szewczyk, 2016) and the large volumes of data generated by key sectors such as public administration (Cavanillas et al., 2016), financial and banking services (Wang, 2003) and the health care industry (Raghupathi, 2016).

In this context, data mining plays a very important role, which is to provide tools and techniques to: (i) analyze such amounts of information to identify hidden patterns and (ii) to deliver meaningful knowledge in order to support decision making. Data mining is in use in several industries, such as retail, finance, manufacturing and health care. All of them are getting the most out of historical data because data mining helps analysts to recognize significant facts, relationships, trends, exceptions, and anomalies that might otherwise go unnoticed (Wang, 2003).

Even so, data creation is occurring at exponential rates (Villars et al., 2011; Jagadish et al., 2014), unveiling a scenario in which data mining solutions become less efficient (Christa et al., 2012) and it is imperative to enhance them on a regular basis. To help with this matter, software agents are a very important asset able to improve Knowledge Discovery Processes (KDD) (Cao et al., 2009), of which data mining is an essential part (Fayyad & Stolorz, 1997). Software agents are the main entities of the multi-agent systems paradigm (MAS) and they have qualities such as autonomy, reactivity, proactivity and social ability (Wooldridge, 2009). These features enable agents to manage and execute almost all steps of a KDD process, since they are capable of running without need of direct human intervention, interacting with other agents and reacting to changes in their environment.

The combination of data mining with the MAS paradigm can produce agent-driven data mining solutions in which software agents can manage data

selection, extraction, preprocessing, and integration (Cao et al., 2009). Furthermore, they are an excellent choice to manage parallel, distributed, or multisource mining (Cao et al., 2009). In this context, several authors have already proposed solutions that take advantage of this approach to work with specific domain problems. Beyond that, there is a shortage of architectural models or software framework solutions that offer a standard platform to build, for more than one domain, agent-driven data mining applications.

1.1 Motivation

Health care services and systems inspired our first motivation to develop this research, since they are an expanding source of large volumes of information about patient health and the care process itself. Today's modern health institutions automatically collect structured data relating to all aspects of care such as diagnosis, medication, test results and radiological imaging data (Jensen et al., 2012). Legacy systems, health monitoring devices, applications for disease management and fitness tracking (Szewczyk, 2016) and medical records data sets for research studies (Gao et al., 2005a) are sources that produce large volumes of data in regular basis.

The health care industry is just a glimpse of a broader context in which the accelerated growth of data generation is a ubiquitous fact in almost every area of modern human society (Columbus, 2016; Miniwatts, 2016; Szewczyk, 2016; Wang, 2003; Cavanillas et al., 2016; Zwolenski & Weatherill, 2014). Such amounts of data need to be mined to produce meaningful knowledge, since traditional ways of knowledge discovery are difficult to perform at those scales. Even the current data mining solutions become less efficient at those data volumes (Christa et al., 2012), being possible to improve their results by adding other techniques.

1.2 Problem Definition

Although there are already data mining solutions to deal with specific domains (Gao et al., 2005a; Palaniappan & Awang, 2008; Gorodetsky et al., 2003; Mitkas et al., 2003; Ralha & Sarmento, 2012; Gonzalez-Sanchez et al., 2011), these approaches do not explicitly present how to create a flexible architecture that take advantage of

software agents to manage and perform mining and classification of data for different domains. Software agent technology brings several benefits to data mining, such as autonomy, reactivity, collaboration, distributed processing, among others (Cao et al., 2009).

There are factors that reaffirm that problem in detail, they are:

- Some solutions for data mining and classification **do not provide mechanisms to be extended to other domains or at least evidence that they were actually extended to other domains** (Gao et al., 2005a; Palaniappan & Awang, 2003; Ralha & Sarmento, 2012; Gonzalez-Sanchez et al., 2011);
- Software agents allow the execution of parallel mining, which brings advantages such as the generation of several models at same time; nonetheless **the use of ensemble techniques to make profit of combining those models** (Di Stefano & Menzies, 2002; Woźniak et al., 2014; Alpaydin,1998) **to produce new ones is almost absent** (Gao et al., 2005a; Gorodetsky et al., 2003; Mitkas et al., 2003; Ralha & Sarmento, 2012; Gonzalez-Sanchez et al., 2011) **as is any comparison between models**;
- **It is not common** — beyond the knowledge generation purpose — **to offer the ability to predict unlabeled instances**. (Gao et al., 2005a; Gorodetsky et al., 2003; Mitkas et al., 2003; Ralha & Sarmento, 2012; Gonzalez-Sanchez et al., 2011).

In this context, emerges the need to develop a generic solution — either an architecture, a framework or both — that serves as platform to build MAS driven data mining applications for different domains able to perform (i) data preparation; (ii) parallel mining; (iii) combination and evaluation of models, and (iv) prediction of unlabeled instances.

1.3 Proposed Solution

In this work we aim at providing *a platform to build applications able to perform agent-driven data mining processes*. For this, we developed the Java Agent Framework for Data Mining (JAF4DM). There was a first version that provided support to build and operate several agents to run different classification-mining methods. With JAF4DM 1.0 (Morejón et al., 2017a), it was possible to build

MAS-based solutions able to work on one data set, to perform the training process (by running in parallel several classification algorithms) and the classification of new unlabeled instances.

The limited scope of version 1.0 made us realize that there was still much to improve. Therefore, we redesigned the process and the current version took shape. Then, we propose JAF4DM 2.0, *a software framework that leverages the MAS approach to manage a data mining process, providing support to build and simultaneously operate software agents that will interact and perform tasks such as: (i) data preprocessing; (ii) parallel mining; (iii) combination and evaluation of models, and (iv) prediction of unlabeled instances*.

In order to test the applicability of the framework, we considered the health care domain, which generates a large volume of data that feeds a constant need to obtain meaningful knowledge. The application of data mining algorithms in the health care industry plays "a significant role in prediction and diagnosis of diseases." (Durairaj & Ranjani, 2013), Therefore, we define three use scenarios related with health care to develop three application based on JAF4DM. These are Diabetes, Hypothyroidism and Arrhythmia conditions.

1.4 Contribution

The main contributions of this dissertation are described as follows:

- A multi-agent system architecture to provide guidance for the development of agent-driven data mining applications;
- JAF4DM 1.0, a software framework — based on such architecture — that provides support to build and operate MAS-based applications to run different classification-mining methods over data from different domains and to classify unlabeled instances based on the produced models. The outcomes of this work where published in two research papers (Morejón et al., 2017a; Morejón et al., 2017b);
- JAF4DM 2.0, an evolution from JAF4DM 1.0. that provides — beyond the original capabilities — a platform to build applications able to perform and manage by means of software agents a data mining process over different data domains. In this version, software agents work together to perform: (i) data preparation; (ii) parallel execution of several data mining algorithms;

(iii) combination and evaluation of models, and (iv) prediction of unlabeled instances;

- Three JAF4DM-based applications as evidence of its applicability.

1.5 Document Organization

The content of this dissertation is organized as follows:

Chapter 2: This chapter contains the theoretical background that supports this work, where we show several concepts that are useful as a starting point to understand the domain of this research and as guidance to perform it.

Chapter 3: This chapter presents the related work. We show its advantages and limitations and how they can be useful to this research.

Chapter 4: This chapter presents JAF4DM framework. We explain its architecture, process, implementation, improvement, etc.

Chapter 5: This chapter describes three use scenarios that show the applicability of JAF4DM. Each one of them depicts an application developed with it.

Chapter 6: This chapter describes the conclusion, limitations and future work.

2 Background

In this chapter we explain several concepts that together form the theoretical basis of this dissertation. Understanding them helped us to drive this research and also will help readers to comprehend it.

2.1 Software Framework

Since the beginning of software engineering as a field in late 60´s, engineers struggled with the problem of building large, reliable software systems in a controlled, cost-effective way (Krueger, 1992). Since then, *software reuse* has been a permanent resource to overcome that problem. It is the process of creating software systems from existing software rather than building software systems from scratch (Krueger, 1992). Building and designing software systems from scratch is still an expensive and error-prone process.

Software frameworks (hereafter 'frameworks') were developed with software reuse as its core idea. According to (Fayad & Schmidt, 1997), "*a framework is a reusable, semi-complete application that can be specialized to produce custom applications*"; they are "*application generators that are directly related to a specific domain*" (Markiewicz & Lucena, 2001). For the purpose of this research we rely on the concepts exposed in (Markiewicz & Lucena, 2001), according to which, "*frameworks must be able to generate applications for an entire domain*" and also, must have customizable flexibility points to suit each application that produces. Its code and design reuse capability enables higher productivity and shorter time-to-market development (Markiewicz & Lucena, 2001).

2.1.1 Hot-spots and kernel

Frameworks are not executables (Markiewicz & Lucena, 2001), they have points of flexibility (hereafter '**hot-spots**') and a **kernel** (immutable and cannot be modified). Since they are abstract classes or methods, hot-spots allow developers

to instantiate the framework by implementing application specific code for each one of them. Only then an executable can be generated. Once the hot-spots are instantiated, the framework will use them using callback. In other words, “old code calls new code” (Markiewicz & Lucena, 2001).

Opposed to the hot-spots (on purpose and structure), is the kernel, which is constituted by those framework features that are immutable and cannot be modified; i.e., the frozen-spots. The kernel is formed by pieces of code already implemented within the framework, which will call the hot-spots provided in the implementation and always will be part of each instance of the framework (Markiewicz & Lucena, 2001). For a framework to be able to work, it is required to implement into an application the methods and classes (hot-spots) that will be used by the framework´s kernel. Figure 1 illustrates the framework concept by means of the engine´s metaphor: “The engine will not run until all plugs are connected” (Markiewicz & Lucena, 2001).

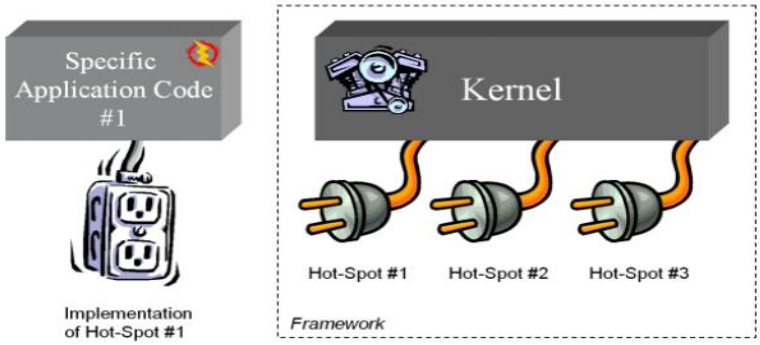

Figure 1. “The engine will not run until all plugs are connected”

2.1.2 Framework development stages

The three major stages of framework development are *domain analysis*, *framework design* and *framework instantiation* (Markiewicz & Lucena, 2001).

Domain analysis: Aims to discover the domain´s requirement and probable future requirements. A core task for it is the analysis of standards and related work — similar solutions or experiences — already published. In this stage, the hot-spots and the kernel are partially uncovered.

Framework design: Defines the framework´s abstractions. Hot-spots and kernel are modeled and the extensibility and flexibility proposed in the domain analysis are sketched. The use of design patterns is essential in this stage.

Instantiation: The framework hot-spots are implemented, generating a software system. Therefore, it is worth to note that several applications can be generated having in common the same kernel.

There are several differences between traditional object-oriented development (TOOD) and framework development (FD) (Markiewicz & Lucena, 2001). Beginning with the scope of the analysis phase, TOOD only studies the requirements of a single problem, unlike FD, which captures the requirements of an entire domain. In addition, TOOD´s outcome is an executable application, whereas FD can result in many applications from a framework. Figure 2 shows this comparison (Markiewicz & Lucena, 2001).

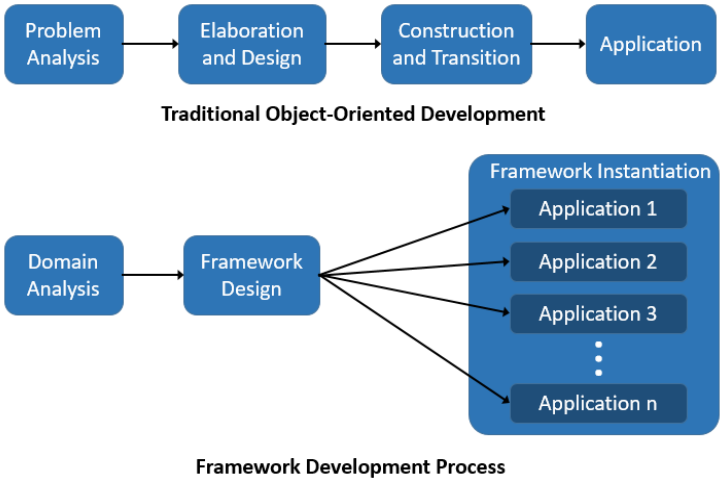

Figure 2. TOOD Vs FD Process

2.1.3 Framework Classification

Frameworks can be classified by the techniques used to extend them into two categories, as follows (Fayad & Schmidt, 1997):

White-box: Rely heavily on OO language features like inheritance and dynamic binding in order to achieve extensibility. Existing functionality is reused and extended by inheriting from framework base classes and overriding pre-defined hook methods. Application developers are required to have intimate knowledge of each framework's internal structure.

Black-box: Rely on defining interfaces for components that can be plugged into the framework via object composition. Existing functionality is reused by defining components that conform to a particular interface and integrating these components into the framework. They are structured using object composition and

delegation rather than inheritance. As a result, they are generally easier to use and extend than white-box frameworks but are more difficult to develop since they require framework developers to define interfaces and hooks that anticipate a wider range of potential use cases.

Figure 3 illustrates this classification of frameworks according to (Markiewicz & Lucena, 2001). The same authors describe as **gray-box**, those frameworks which contain both white-box and black-box characteristics.

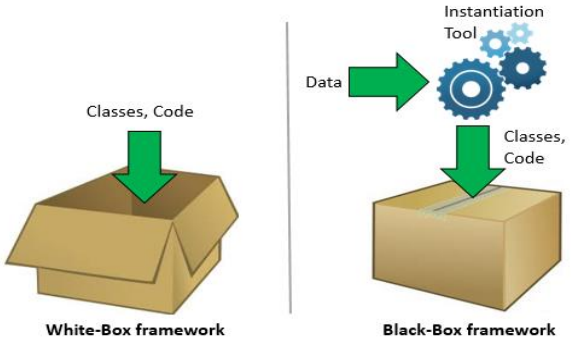

Figure 3. White-Box Vs Black-Box frameworks

2.2 Multi-agent systems

A software agent is a computer system situated in some environment, capable of autonomous action in order to meet its design objectives (Wooldridge, 2009). It can figure out by itself what needs to be done without having to receive an explicit order of what to do at any given moment.

A multi-agent system (MAS) is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. To interact successfully, these agents must be able to cooperate, coordinate, and negotiate with one another (Wooldridge, 2009).

2.2.1 Key Features

Software agents have four key features that define them (Wooldridge & Jennings, 1995):

Autonomy: They are able to operate without the direct intervention of humans, and have some kind of control over their actions and internal state.

Social ability: They are able to interact with other agents (and possibly humans) via some kind of agent-communication language.

Reactivity: They also perceive their environment and respond in a timely fashion to changes that occur in it.

Pro-activeness: Agents are able to exhibit goal-directed behaviour by taking the initiative, beyond just simply act in response to their environment.

In same course of action, there are **three primary attributes** which agents should exhibit (Nwana, 1996): **autonomy, learning** and **cooperation**. As Figure 4 shows, from these features it is possible to derive 4 main types of agents (Nwana, 1996): **collaborative agents**, **collaborative learning agents**, **interface agents** and **smart agents.**

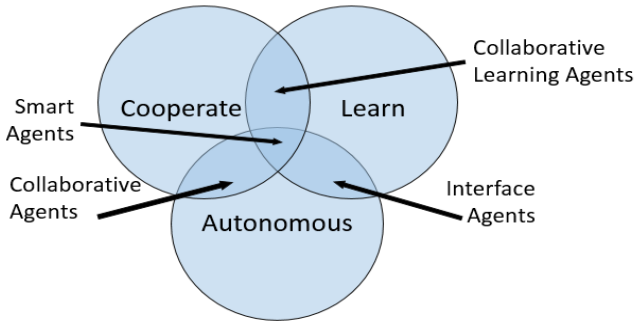

Figure 4. Four types of agents: Smart, Collaborative, Interface and Collaborative Learning

2.2.2 Suitability of an Agent-Based Solution

According to (Wooldridge, 2009; Jennings & Wooldridge, 1998), there are four important domain features that tells when an agent-based approach is suitable to address a problem. These are:

Open environments: In such environments (which can also be highly dynamic, complex or uncertain) the only appropriate solution is often the use of systems capable of flexible autonomous action

Natural Metaphor: The system is naturally considered as a society of autonomous components that cooperate with each other to solve complex problems or else compete with one another. In this context, the idea of an agent seems natural.

Distribution of Data, Control or Expertise: There are some situations in which a centralized solution is at best extremely difficult or at worst impossible. Scenarios that are defined by the distributed nature of data, control, expertise and

resources. In such situations, agents provide a natural way of modeling the problem, where each data source is a semi-autonomous component.

Legacy Systems: Many organizations rely on software systems that are technologically obsolete but functionally essential. These systems (called legacy systems) cannot generally be discarded and rewriting them is very expensive. So, in the long term they need to be able to cooperate and communicate with other (generally newer) software components. A solution for this can be the use of an 'agent wrapper' that enables the software to operate with other systems.

2.2.3 Application Domains of Multi-agent Solutions

MAS solutions are classified by the application domain (Jennings & Wooldridge, 1998) in order to illustrate the scope and diversity of agent applications. Those application domains were defined according to four areas of human activity (offering several examples), as follows:

- Industrial Applications: Manufacturing systems and critical systems like process control systems and air traffic control.
- Commercial Applications: Information management systems, business process management systems and electronic commerce systems.
- Medical Applications: Patient monitoring systems and health care systems that manages the patient care process.
- Entertainment: Computer games, theater and cinema interactive systems.

There is a more horizontal approach (Wooldridge, 2009) which defines the application domains of software agents into types of activities that can be performed over several areas of society, whether is health care, e-commerce or industrial applications. These are:

- Workflow and Business Process Management;
- Information Retrieval and Management;
- Human-Computer Interfaces;
- Electronic Commerce;
- Virtual Environments;
- Distributes Sensing;
- Social Simulation.

2.3 JADE

The Java Agent Development Framework (JADE) is a software framework to support the development of agent applications in compliance with the FIPA 2000 specifications for interoperable intelligent multi-agent systems (Bellifemine et al., 2001). Figure 5 shows a standard model of an agent platform, according to (Bellifemine et al., 2003).

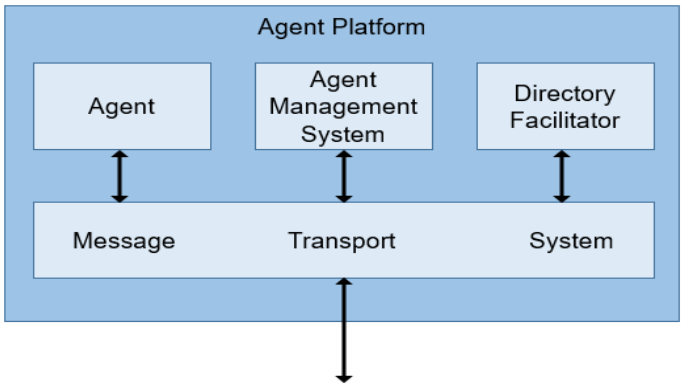

Figure 5. Reference architecture of a FIPA Agent Platform

According to Fig. 5, three elements are essential in a FIPA compliant agent platform. First, the Agent Management System (AMS) — only one per platform — which is the agent that exerts supervisory control over access to, and use of, the Agent Platform. Each agent must register with an AMS in order to get a valid agent identifier (AID) (Bellifemine et al., 2003). Second, the Directory Facilitator (DF), the agent that provides the default yellow page service in the platform. Finally, the Message Transport System (or Agent Communication Channel (ACC)), the software component controlling all the message exchanges within the platform, including messages to/from remote platforms.

There are two different points of view for describing the JADE system: first, JADE is a runtime system for FIPA compliant multi-agent systems, supporting application agents every time they need to exploit some FIPA covered feature such as the life-cycle management of the agent. Second, JADE is also a Java framework for developing FIPA compliant agent applications, making FIPA standard assets available to the programmer through object-oriented abstractions (Bellifemine et al., 2001).

JADE includes the libraries (i.e. the Java classes) required for developing application agents, and the runtime environment that provides the basic services.

This environment must be active on the device before the agent's execution. Each instance of the JADE run-time is called container. All containers taken as a whole form the platform, which provides a homogeneous layer that hides the complexity and the diversity of the underlying tiers (hardware, operating system, type of network, JVM) from agents and application developers (Bellifemine et al., 2008).

To perform communication between agents, JADE makes use of the FIPA Agent Communication Language (ACL) (Bellifemine et al., 1999). Agent communication is based on the exchange of messages, where agents communicate by formulating and sending individual messages to each other. The FIPA ACL specifies a standard message language by setting out the encoding, semantics and pragmatics of the messages. FIPA specifies that the messages transported should be encoded in a textual form. It is assumed that the agent has some means of transmitting this textual form (Bellifemine et al., 1999).

A software agent should be able to perform several tasks at same time in response to external events. Following this approach, the JADE execution model is based on the **Behaviour** abstraction (Bellifemine et al., 2001), which models agent tasks. In other words, every single JADE agent runs in its own Java thread, and all its tasks are modeled and implemented as Behaviours that are executed cooperatively.

Figure 6 illustrates the behaviours that are structured in JADE hierarchically (Bellifemine et al., 2007).

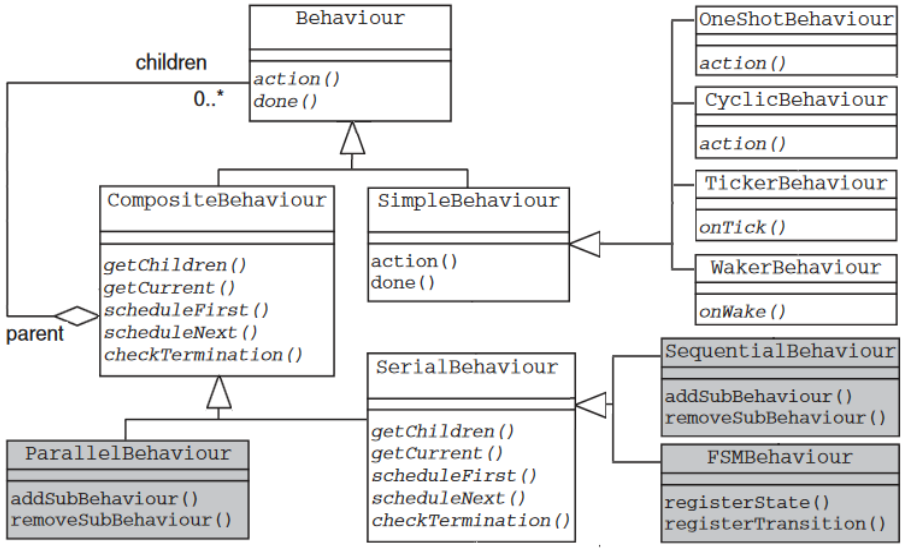

Figure 6. JADE Behaviours

Behaviour: Is an abstract class that provides a basic element to model agent tasks.

OneShotBehaviour: Is a SimpleBehaviour that models atomic behaviours that will be executed just one time and cannot be blocked.

CyclicBehaviour: Is a SimpleBehaviour that models atomic behaviours that will be executed permanently.

TickerBehaviour: Is a SimpleBehaviour that implements a cyclic task that must be executed periodically.

WakerBehaviour: Is a SimpleBehaviour that implements a task that must be executed just once after a period of time

CompositeBehaviour: Models behaviours that are composed of several other behaviors.

ParallelBehaviour: Performs its sub-behaviors in parallel and ends when a particular condition is found.

SequentialBehavior: Performs its sub-behaviors sequentially and ends when all of them have finished executing.

FSMBehaviour: Performs its sub-behaviors according to a Finite State Machine previously defined by the user.

2.4 Knowledge Discovery in Databases

As authors in (Fayyad & Stolorz, 1997; Fayyad et al., 1996), we also adopt the definition in which *Knowledge Discovery in Databases (KDD) is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data* (Piatetsky-Shapiro, 1996).

In other words, KDD refers to the overall process of discovering useful knowledge from data (Fayyad et al., 1996). Data mining refers to a particular step in this process. The other steps in the KDD process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data. It is a process that is interactive and iterative (with many decisions made by the user), involving numerous steps (Fayyad et al., 1996) as Figure 7 shows.

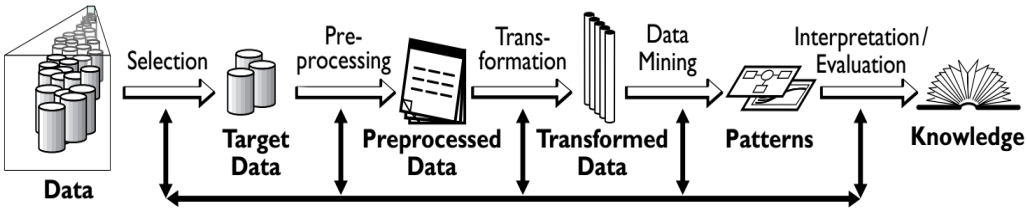

Figure 7. KDD process (Fayyad et al., 1996)

Steps of KDD:

The KDD stages are the following (Azevedo & Santos, 2008).

1. **Selection**: Aims to create a target data set, or to focus on a subset of variables or data samples, on which discovery is to be performed.
2. **Pre-processing**: The purpose here is to clean and pre-process the target data in order to make it more consistent.
3. **Transformation:** This stage consists on the transformation of the data using dimensionality reduction or transformation methods.
4. **Data mining:** This stage is about the search for patterns of interest in a particular representational form or a set of such representations depending on the data mining objective (usually, prediction).
5. **Interpretation/Evaluation:** This stage consists of the interpretation and evaluation of the mined patterns.

2.5 Data mining

Data mining is a particular step in the KDD process, which involves the application of specific algorithms (under acceptable computational efficiency limitations) to produce a particular set of patterns (or models) from the data (Fayyad & Stolorz, 1997). In (Witten et al., 2016), authors define data mining from the operational point of view as *the process of discovering patterns, automatically or semi automatically, in large quantities of data* –and the patterns must be useful.

It also allows computer-driven exploration of data, facilitating this kind of work for problems that are very difficult to explore by humans. For instance, it is very hard for humans to visualize and understand a large dataset, which can grow expressively in terms of the number of attributes and the number of cases (Fayyad

& Stolorz, 1997). The fact that traditional data analysis techniques cannot deal with such growth rates is what makes Data Mining a necessity.

As part of KDD process, data mining is a component that is concerned with the algorithmic means by which pattern are extracted and enumerated from data (Fayyad & Stolorz, 1997). As shown in Section 2.4, KDD includes the evaluation and possible interpretation of the mined "patterns" to determine which patterns may be considered new "knowledge".

Data mining techniques can be divided into five classes of methods (Fayyad & Stolorz, 1997):

1. Predictive modeling
2. Clustering
3. Data summarization
4. Dependency modeling
5. Change and deviation detection.

It is a good point to notice that *there is no universally best data mining method, choosing a particular algorithm for a particular application is something of an art* (Fayyad et al., 1996).

For the aims of this research, we pay special attention to Predictive Modeling, especially to **Classification**, since our solution´s final purpose is to predict new unlabeled instances. *In classification the basic goal is to predict the most likely state of a categorical variable (the class)* (Fayyad & Stolorz, 1997).

Decision trees and rules are popular model representations, as also linear and nonlinear models, among others. Model representation determines both the flexibility of the model in representing the data and the interpretability of the model in human terms (Fayyad et al., 1996). The same authors affirm that, typically, the more complex models may fit the data better but may also be more difficult to understand and to fit reliably. The same authors also mention successful applications in which the involved practitioners often relies in the use of simpler models due to their robustness and interpretability.

All work in the fields of classification and clustering in statistics, pattern recognition, neural networks, databases and machine learning would fit under the data mining steps (Fayyad & Stolorz, 1997).

2.6 Machine learning

Witten et al. (2016) also offers an operational definition for learning: *Things learn when they change their behaviour in a way that makes them perform better in the future*. This ties learning to performance rather than knowledge. Yet having a change of behaviour does not necessarily mean that there was some kind of learning. Applied to computers that definition has its own problems because whether artifacts can behave purposefully is unclear.

Machine learning (ML) addresses the question of *how to build computer programs that improve their performance at some task through experience*. Its application is especially useful in data mining problems (where large databases may contain valuable implicit regularities that can be discovered automatically), in poorly understood domains (where humans might not have the knowledge needed to develop effective algorithms) and in domains where the program must dynamically adapt to changing conditions (Mitchell, 1997).

ML draws on concepts and results from many fields, including statistics, artificial intelligence, philosophy, information theory, biology, cognitive science, computational complexity, and control theory (Mitchell, 1997) ML algorithms are being used routinely to discover valuable knowledge from large commercial databases containing equipment maintenance records, loan applications, financial transactions and medical records (Mitchell, 1997).

There are **several types of ML** (Ayodele, 2010):

Supervised learning: where the algorithm generates a function that maps inputs to desired outputs. One standard formulation of the supervised learning task is the classification problem: *the learner is required to learn (to approximate the behavior of) a function which maps a vector into one of several classes by looking at several input-output examples of the function.*

Unsupervised learning: which models a set of inputs: labeled examples are not available.

Semi-supervised learning: which combines both labeled and unlabeled examples to generate an appropriate function or classifier.

Reinforcement learning: where the algorithm learns a policy of how to act given an observation of the world. Every action has some impact in the

environment, and the environment provides feedback that guides the learning algorithm.

Transduction: similar to supervised learning, but does not explicitly construct a function: instead, tries to predict new outputs based on training inputs, training outputs, and new inputs.

Learning to learn: where the algorithm learns its own inductive bias based on previous experience.

2.7 WEKA

Waikato Environment for Knowledge Analysis (WEKA) is a collection of machine learning algorithms for data mining tasks. WEKA provides several implementations of learning algorithms to apply to data sets. Furthermore, it includes a diversity of tools for transforming data sets; it also allows preprocessing a data set, feeding it into a learning scheme, and analyzing the resulting classifier and its performance – all without writing any program code at all (Eibe et al., 2016). For a Java environment, it is possible to solve a learning problem without writing any machine learning code, just by accessing the available WEKA algorithms (Eibe et al., 2016). It is easy to extend thanks to simple API and plug-in mechanisms and facilities that automate the integration of new learning algorithms by means of its graphical user interface (Hall et al., 2009).

The workbench includes methods for the main data mining problems: regression, classification, clustering, association rule mining, and attribute selection. In addition, it provides many data visualization facilities and data preprocessing tools. All algorithms take their input in the form of a single relational table obtained from a file or generated by a database query (Eibe et al., 2016).

There are three ways of using WEKA: (i) to apply a learning method to a data set and analyze its output to learn more about the data; (ii) to use learned models to generate predictions on new instances, and (iii) to apply several different learners and compare their performance in order to choose one for prediction (Eibe et al., 2016). It also offers tools for data preprocessing, called filters. Like classifiers, it is possible to select filters from a menu and tailor them to any requirements.

3 Related Work

Some authors have proposed several approaches and solutions (Cao et al., 2009; Gao et al., 2005a; Gao et al., 2005b; Gorodetsky et al., 2003; Mitkas et al., 2003; Ralha & Sarmento, 2012; Gonzalez-Sanchez et al., 2011) to take advantage of applying the MAS paradigm to enhance and manage data mining processes. Almost all of them offer practical solutions for a specific problem or domain and with different scopes. All solutions referenced in this chapter have strengths and flaws, giving us the chance to reuse some ideas and to see what is lacking in order to produce a good solution at the end of this work. Each section in this chapter provides a summary of each reviewed work, beginning with the one that highlights the advantages of combining DM and software agents.

3.1 Agent Mining: The Synergy of Agents and Data Mining

Cao et al. (2009) aim to offer the definition of agent mining to gather any software solution that comprises DM and software agents working together. They offer a panoramic picture about how software agents and DM are related and how these research fields can help each other.

Multi-agent learning, adaptation, evolution, and behavior analysis are between typical problems in agents that could find satisfactory solutions in data mining. For instance, knowledge extracted through data mining can provide more stable, predictable, and controllable models or it can assist in the self-organization and evolution of multi-agent systems in acceptable directions. On the other hand, software agents can support and enhance the knowledge discovery process in many ways — in what is called as agent-driven data mining. For example, software agents can contribute to data selection, extraction, preprocessing, and integration and they are an excellent choice for parallel, distributed, or multisource mining. Agents are also a good match for interactive mining and human centered DM (Cao et al., 2009).

They analyze how that synergy between DM and software agents can occur, they also generate then a high-level research map of agent mining as a disciplinary

area. Some research components of such agent-mining disciplinary framework are agent-driven data processing, agent-driven knowledge discovery, agent-driven information processing and agent-mining performance evaluation. Figure 8 presents all the research components identified by the authors. Knowing about them, gave us perspective about the path followed in the development of this dissertation.

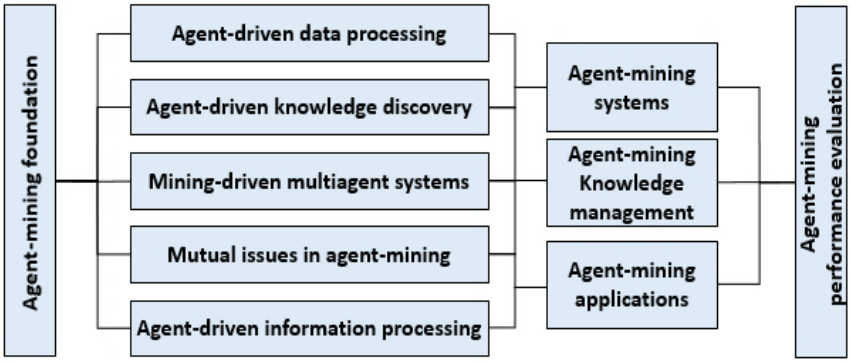

Figure 8. Agent-mining disciplinary framework

3.2 CoLe: a cooperative multi-agent data mining model.

The authors developed CoLe (Gao et al., 2005a), a model for cooperative agents to mine knowledge from heterogeneous data. They designed it, having in mind the use of various agents to perform a data mining process over a completely heterogeneous dataset by taking advantage of the cooperation between these agents to generate more significant and more useful structures of knowledge that no individual mining agent can produce alone (Gao et al., 2005a). This model works by making iterations of the mining process. For each iteration, discovered knowledge can be useful for the next one. It is worth to notice that the model proposed in this work served as an inspiration for us to design the process that defines how our proposed solution works.

Based on the proposed model, the authors implemented and tested a multi-agent system for mining diabetes-related data (Gao et al., 2005a; Gao et al., 2005b) by having two agents running mining algorithms and another agent with methods to produce hybrid rules. Although they expressed intentions to develop other systems based on CoLe (Gao et al., 2005a) and also stated that CoLe is a multi-agent systems framework, there is no evidence in the literature of systems

based on CoLe developed beyond this diabetes scenario. So far it is just an ad hoc solution, not a software framework.

In additional, CoLe aims only at knowledge generation. It was not developed to perform prediction tasks over unlabeled instances with the acquired knowledge — which is represented in the form of "if … then …" rules.

3.3 A Multi-Agent Technology for Distributed Data Mining and Classification

The authors proposed a MAS technology for distributed data mining (DDM) and distributed classification (DC). They addressed as a key problem the fact that data sources are distributed, heterogeneous, and, as a rule, of large scale (Gorodetsky et al., 2003). In addition, the authors addressed the issue that the distributed data mining MAS design technology presumes collaborative activities of agent mediated distributed users (Gorodetsky et al., 2003). Therefore, they proposed an architecture of DDM and DC MASs and a technology for their design. They further proposed a number of well-developed protocols supporting agent-mediated design of applied DDM and DC MASs (Gorodetsky et al., 2003).

The authors make use of several software agents to execute several classification algorithms at different levels of the architecture at different moments of the process. There are also several agents that make internal decisions and other that manage the work of classification agents.

The developed solution was validated working with the dataset used for The Third International Knowledge Discovery and Data Mining Tools Competition from 1999 available at (Bache & Moshe, 2013). The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. Nevertheless, the results of such validation were not present in the published paper although a very specialized application for the same computer network security domain was later developed (Gorodetsky et al., 2004).

3.4 A Framework for Constructing Multi-Agent Applications and Training Intelligent Agents

Mitkas et al. (2003) present Agent Academy (AA), an open-source framework and an integrated development environment to create Software Agents and Multi-Agent Systems. It was implemented upon the JADE framework and using WEKA APIs. The framework provides embedded intelligence to agents by inserting decision models generated by a previous data mining process over a background data-specific application.

In this case, how data mining and agents are related responds to a different approach than previous cases — data mining-driven MAS instead of agent-driven data mining (Cao et al., 2009). Here, the use of data mining techniques aims to improve the software agent's performance and its reasoning capabilities. The authors implement a "training module" which embeds essential rule-based reasoning into agents. The framework is not designed to inherently produce software agents able to manage a whole data mining process since there is only one mining agent to produce the decision models.

3.5 A multi-agent data mining system for cartel detection in Brazilian government procurement

The authors of this work introduce AGMI, an agent-mining tool that integrates software agents into DM techniques, and applied it to the Brazilian government procurement domain (Ralha & Sarmento, 2012). It is built upon a Weka-based data mining framework and JADE framework. It has three architectural levels (strategic, tactical and operational) and operates with four different types of agents (coordinator, evaluator, local mining supervisor and mining).

AGMI was defined to apply different DM techniques, using a collaborative approach of interaction among agents to work over a distributed environment, with an integrated, intelligent perspective that primarily intends to improve the knowledge discovery process (Ralha & Sarmento, 2012). It uses agents to evaluate the knowledge generated in the KDD process (evaluator) and agents to run DM algorithms over the prepared datasets to extract useful knowledge from them. So far, we do not see any evidence that authors published any work that applies AGMI to another domain than the one described here.

3.6 Abe: An agent-based software architecture for a multimodal emotion recognition framework

Gonzalez-Sanchez et al. (2011) offer two solutions in one: an architecture (called ABE) based on components and software agents and a multimodal emotion recognition framework based on it. Its main purpose is to solve the lack of frameworks and models that can help developers to integrate emotion recognition into their software projects. Such approach included the use of hardware devices like brain-computer interfaces, eye tracking systems, face-based emotion recognition systems, and sensors for skin conductivity, posture, and finger pressure.

For each one of these hardware elements there is a Specialist Agent, which comprises a data source, a model, a controller and a communicator. This agent is responsible for collecting raw data (from the sensors), parsing it into sensed values and inferring beliefs, and for communicating their beliefs with Centre agent. The Centre agent is a "distinguished" agent to which Specialist agents are in permanent communication, contributing with data. It implements integration algorithms that convert, in real time, the beliefs reported by the Specialists into emotional states. It is interesting how authors state the idea of used already generated knowledge to generate new one. An idea we bring to our work although not in same way.

Authors claim that in experiments in a gaming environment scenario they were able to detect frustration in players. But, there is no explanation in the whole document of any data mining algorithm that was actually used or any explanation about how that knowledge was generated.

3.7 Discussion about the Related Work

These works have several useful elements for this dissertation; beginning with the first one (Cao et al., 2009), which provided us — from a theoretical perspective — with the confirmation of this dissertation´s field of action: agent-driven data mining. In this line of action, four of the reviewed works (Gao et al., 2005a; Gorodetsky et al., 2003; Ralha & Sarmento, 2012; Gonzalez-Sanchez et al., 2011) adopted the same approach, in contrast with the one that developed data mining-driven agents (Mitkas et al., 2003). Another element to take into account is that some solutions described here (Mitkas et al., 2003; Ralha & Sarmento, 2012) use WEKA for DM

and JADE as MAS framework since both are written in Java and are easy to integrate with each other. That speaks positively about the suitability of these tools to comply with the agent-mining paradigm.

Were proposed several generic type solutions such as models (Gao et al., 2005a), architectures (Gorodetsky et al., 2003; Gonzalez-Sanchez et al., 2011) and frameworks (Mitkas et al., 2003; Gonzalez-Sanchez et al., 2011). But there is no evidence that two of them (Gao et al., 2005a; Gonzalez-Sanchez et al., 2011) were successfully applied to more than one domain, leaving them as **ad hoc solutions in practice**, just as AGMI (Ralha & Sarmento, 2012). Furthermore, those two that were successfully applied were either very specialized (Gorodetsky et al., 2003) or — as we mentioned before — oriented to a different agent mining approach (Mitkas et al., 2003).

With one exception (Gao et al., 2005a), **none of these works performed any kind of combination or ensemble technique** to combine the produced models to create new knowledge — despite several authors defend such approach (Di Stefano & Menzies, 2002; Woźniak et al., 2014; Alpaydin,1998). Also, **none of them performed any comparison between models** to choose the best one. At the same time, **none of these solutions aims at explicitly offering the ability to classify new instances** based on the produced knowledge models.

Table 1 summarizes the discussion above by outlining the elements that are present in the aforementioned works — identified by the section number — that were useful for the development of this dissertation as well as the main shortcomings of those works. All deficiencies emphasized here, were elements addressed during the development of this dissertation and solved in our solution as a differential.

Table 1: Related Work

	Useful elements					*Shortcomings*		
Related work	**Agent-driven DM approach**	**JADE+ WEKA**	**Model**	**Architecture**	**Framework**	**Domain Specific**	**Ensemble Technique**	**Compare Models**
3.2	✓		✓			X		X
3.3	✓			✓		X	X	X
3.4		✓			✓		X	X
3.5	✓	✓				X	X	X
3.6	✓			✓	✓	X	X	X

4 JAF4DM Framework

JAF4DM is a software framework that went through two stages of development. Initially, there was a 1.0 version that gave us the experience that served as core element to develop the current 2.0 version. In this chapter we describe JAF4DM 1.0 at first, presenting its architecture, extension points, the process and how can be instantiated. Next, we describe JAF4DM 2.0 in more detail in order to show its features to allow readers to know the main differences between both versions. Although it is important to notice that there are three main features that remain unaltered.

First, JAF4DM is an extension of JADE (Bellifemine et al., 2003), a FIPA-compliant software framework (implemented in Java) that provides a simplified way to implement MAS. Second, it makes use of classification algorithms (provided by the WEKA API (Eibe et al., 2016)) to find patterns in data. And third, it allows to run in parallel several of these mining methods.

As we outlined in Section 1.3, applications based on our proposed solution will perform, as final task, the prediction of unlabeled new instances. Thus, there is a need to generate adequate models to perform those predictions. According to concepts described in Section 2.5 and because we aim at predicting a categorical field in a new instance, classification algorithms are the most suitable choice to produce such models. Also, the use of more than one data mining algorithm (classifiers in this context) is justified because it offers better results than just using one (Di Stefano & Menzies, 2002; Woźniak et al., 2014; Alpaydin,1998).

4.1 JAF4DM 1.0

The JAF4DM framework was originally set to process only clinical data about specific illnesses. This version comprises three main functions: (i) a data preparation task to make data ready for training; (ii) a training process, where a desired mining method (or several) is executed over the previously prepared data,

and (iii) the execution of the prediction tasks, performed by agents using the models generated by the training process over new unlabeled instances.

4.1.1 Architecture

JAF4DM 1.0 supports the creation of applications able to execute a data mining process and perform prediction tasks over a given set of unlabeled data. Initially our inspiration came from the model offered in (Gao et al., 2005a), but we later added features like the possibility to run more than two mining methods and to perform prediction tasks using the generated knowledge. Furthermore, JAF4DM 1.0 offers to users the chance to implement solutions based on it for several domain scenarios.

As explained earlier, this framework extends JADE, which allows the creation and execution of software agents. Also, it uses the WEKA API, which provides a set of data mining algorithms and methods to perform training and prediction tasks. Figure 9 depicts the JAF4DM 1.0 architecture, which stays simple beyond JADE and WEKA internal architectures. Likewise, the class diagram in Figure 10 offers a more detailed view of it.

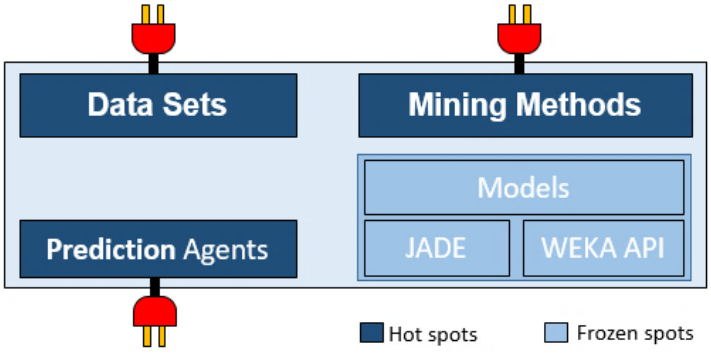

Figure 9. JAF4DM 1.0 Architecture

4.1.2 Hot-Spots

As pointed out in Section 2.1, frameworks have points of flexibility (hot-spots) and a kernel (i.e. frozen-spots) and JAF4DM 1.0 follows that standard. It inherits from JADE the same core elements and hot-spots – for instance the process used for the communication between software agents and the identifiers of agents. In addition, it also defines other specific hot-spots, which are described as follows:

Datasets (DataSetupAgent class). There are several ways to interact with a dataset, from the format of the source (csv file, arff file, etc.) to the way that data is prepared (divided by cross-validation split or by percentage split). Also, the id of each training agent that will work with the prepared data must be provided.

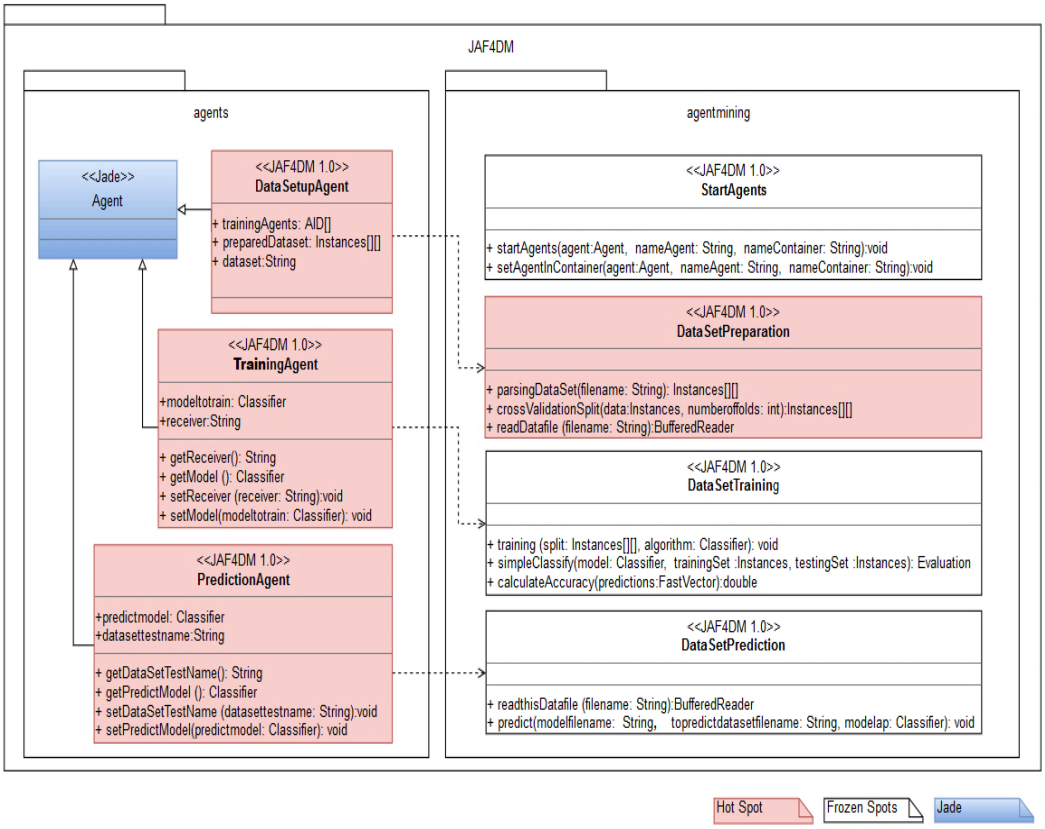

Figure 10. JAF4DM 1.0 Class Diagram

Mining methods (TrainingAgent class). There are many methods to carry out data mining - all different in some ways. An instance of JAF4DM 1.0 could be developed to perform the training process based on different algorithms. Therefore, training agents are going to train and produce several models by executing in parallel several mining methods provided by the WEKA API. For each algorithm to be used, an instance of this class must be implemented defining the algorithm itself and the id of the agent that will use the obtained model to perform prediction.

Prediction agents (PredictionAgent class). They perform a prediction task over unlabeled instances, but the ways to achieve this can differ depending on the selected method.

In this context, the frozen-spots are composed of (i) the JADE framework; (ii) all classes that work directly with the mining algorithms; and (iii) a set of models that are produced to be used later on to perform predictions.

4.1.3 Agents

JAF4DM 1.0 has three types of agents:

Data Setup Agent: Prepares the data that has to be processed for training by selecting the source of the data, extracting the data, and getting it ready for the training process. This agent has a behaviour of type *OneShotBehaviour* (see Section 2.3) that is executed one time to let data ready for training and also to inform all training agents about it when it is done.

Training Agent: Over the previous prepared data, and with a selected classifier algorithm, this agent trains and saves a model, so that it can be used in future predictions. This agent has a behaviour of type *CyclicBehaviour* that is executed permanently waiting to receive the communication from the Data Setup Agent that data is ready. When it receives the data, it begins to perform the mining task. It also has a behaviour of type *OneShotBehaviour* (see Section 2.3) that is executed one time to inform prediction agents that models are ready to be used to perform prediction tasks.

Prediction Agent: This agent takes an unlabeled dataset and classifies it using an existing model. It comprises a behaviour of type *CyclicBehaviour* that is executed permanently waiting to receive from its corresponding Training Agent the communication that the model is ready. When it receives it, it begins to perform the prediction task.

It is important to highlight that, to implement and deploy one application, based on the JAF4DM 1.0 framework, it is necessary to implement at least one instance of each agent mentioned before in order to successfully execute the mining process. To build an application based on JAF4HDM, there must be at least one Data Setup Agent, one Training Agent, and one Prediction Agent for an effective execution of the mining process. And, for each Training Agent, a Prediction Agent must be implemented.

4.1.4 The process

Having knowledge of the framework and its elements, the next thing is to know how it works. Figure 11 depicts the process for which JAF4DM 1.0 was designed. Applications based on it perform their mining process as represented here. The process begins with one agent that prepares a data set for training. Then, several agents will train different mining methods over that data set, obtaining a trained model of knowledge for each one of them. Finally, by using these models, other agents will predict the classification of new instances in a provided unlabeled data set.

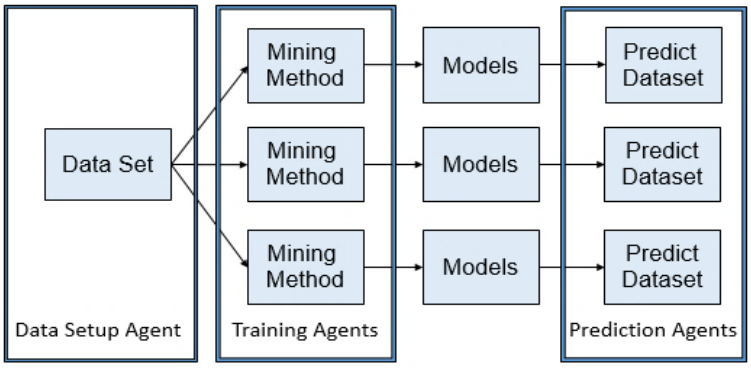

Figure 11. JAF4DM 1.0 Mining Process

4.2 JAF4DM 2.0

As mentioned at the beginning of this chapter, JAF4DM 2.0 (Hereafter, JAF4DM) evolved thanks to the experience acquired with version 1.0 (Morejón et al., 2017a; Morejón et al., 2017b). We realized that our vision was narrow at the beginning, since we had limited the applications of JAF4DM only to the health care area. New functionalities were also devised after version 1.0. For that reason, we redesigned the process and the elements of our first development. We then produced a software framework to build applications able to perform agent-driven data mining processes.

JAF4DM comprises four main tasks managed by software agents: (i) a data preparation task to make data ready for training; (ii) a training process, where several mining methods are executed over the previously prepared data; (iii) a combination process, that performs ensemble methods with the trained models and compares their accuracy against the models obtained individually, and (iv) the execution of a prediction task, over new unlabeled instances.

4.2.1 Architecture

JAF4DM maintains almost entirely its initial architecture but several changes were introduced in the behaviours of some elements. A new type of software agent was also added, with a set of specific functions. It follows the definitions proposed by (Markiewicz & Lucena, 2001) described in Section 2.1 to comply with the standard of how a software framework must be. Figure 12 describes JAF4DM architecture, which remains simple beyond JADE and WEKA internal architectures. In the following sections we describe the element of it in more detail. As well, the class diagram in Figure 13 offers a more detailed view of it.

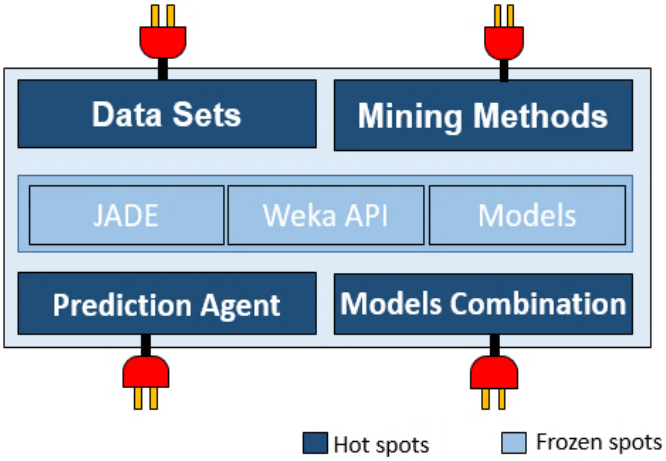

Figure 12. JAF4DM Architecture

4.2.2 Hot-Spots

As in version 1.0, JAF4DM follows the standards and concepts described in Section 2.1. It also remains with same frozen-spots and hot-spots. In addition, a new hot-spot is introduced. In general, the hot-spots (which are abstract classes) are described (in match with figures 12 and 13), as follows:

Datasets (DataSetupAgent class). Remains equal to version 1.0. Depending on the application scenario, datasets can receive different treatments regarding to the format, the source or the kind of preparation itself. This must be specified every time JAF4DM is instantiated, as well as the id of the training agents that will work with the prepared data.

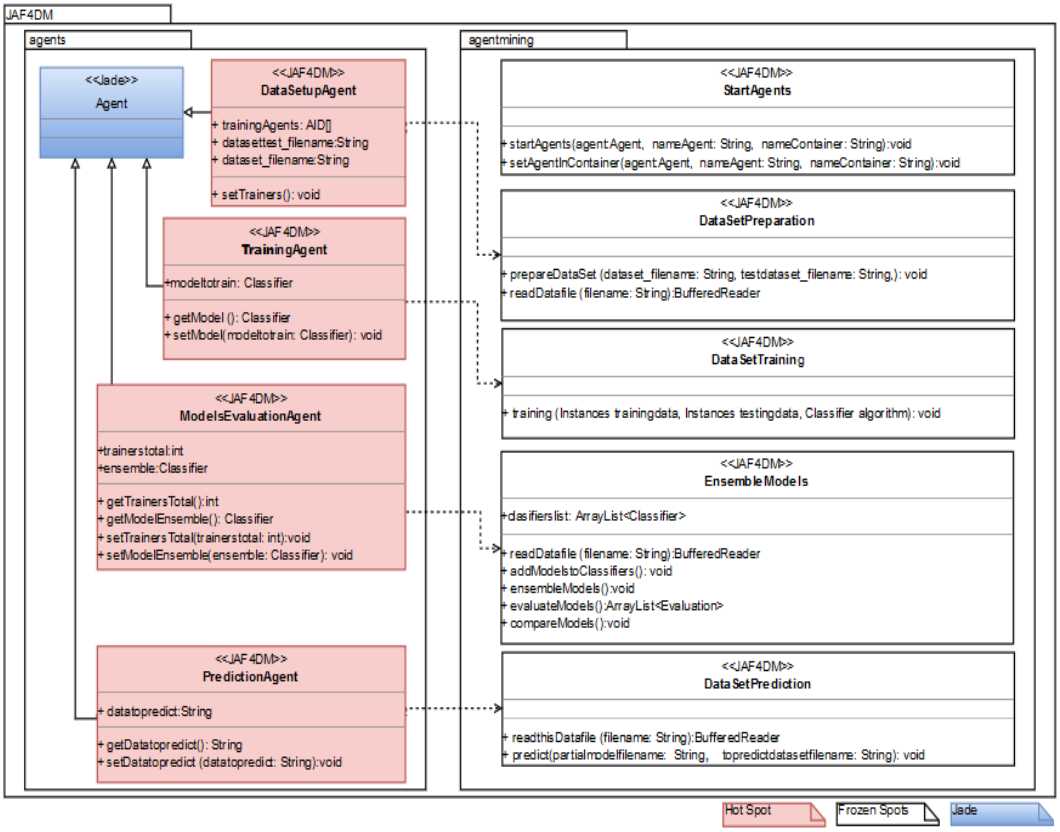

Figure 13. JAF4DM Class Diagram

Mining Methods (TrainingAgent class). Each application based on JAF4DM can use different algorithms to produce their models. Here, the way of functioning is the same from version 1.0. For one application scenario, users may have the need to produce several models based on different algorithms. To obtain the desired models, an instance of this class must be implemented for each one of them and the algorithm to run must be specified on it, as well the id of the agent that will use the produced models.

Models Combination (ModelsEvaluationAgent class). This hot-spot was devised after we finished the development of the first version. We realized that after applying several mining algorithms, the produced models could generate new models if combined using ensemble techniques. Several authors support and defends this practice (Di Stefano & Menzies, 2002; Woźniak et al., 2014; Alpaydin,1998). Therefore, the assembling methods must be defined according to user´s need in the specific scenario (all the models involved will be evaluated).

Prediction Agent (PredictionAgent class). Its purpose is to perform a prediction task to classify new unlabeled instances. The unlabeled data set to be predicted must be specified each time JAF4DM is instantiated.

4.2.3 Agents

JAF4DM involves four types of agents that will cooperate by exchanging messages between them and will run reactively depending on environment conditions at specific moments.

Data Setup Agent: This agent prepares the data that will be used as input for training, by selecting and extracting the data from its source and setting it ready for the training process. Here (as in version 1.0.) the behaviour remains as one of type OneShotBehaviour, which also is executed one time to let data ready for training and also to inform all training agents about it when it is done. Figure 14 displays an activity diagram that describes how a Data Setup Agent behaves.

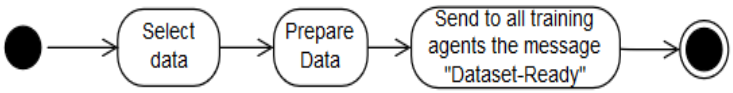

Figure 14. Data Setup Agent Activity Diagram

Training Agent: Over the previously prepared data, with a selected classifier algorithm, this agent will train and save a model, which will be used either to generate additional models by ensemble techniques or to perform prediction tasks. Here, the behaviour remains as CyclicBehaviour, designed for being executed repeatedly until to receive the warn from DataSetup Agent that data is ready. Then, it performs the mining task where the model is produced. It also has a OneShotBehaviour, but now is executed one time to inform the Models Evaluation Agent that models are ready to be used to perform assembling, evaluation and prediction tasks. Figure 15 illustrates how a Training Agent behaves.

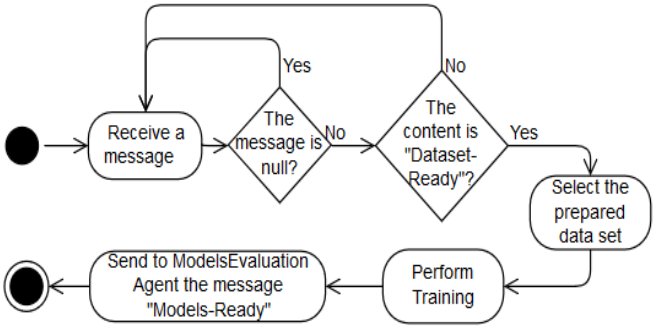

Figure 15. Training Agent Activity Diagram

Models Evaluation Agent: This agent is an exclusive feature of JAF4DM 2.0. It uses all the trained models to create new ones by using an assembly strategy.

Then, it evaluates and compares all of them (the first ones and those generated now) to offer users the more accurate one. It will put available the best result as input to Prediction Agent´s task. It also involves two behaviors. First, one *CyclicBehaviour*, which is executed several times waiting to receive notice from each Training Agent notifying that its model is ready. Once it receives all notifications from training agents, it proceeds to gather all models produced by training agents, performs the ensemble algorithm and then performs the evaluation of all models involved. Then, a OneShotBehavior is executed to notify the Prediction Agent that a model is ready to be used to perform its prediction task. Figure 16 illustrates how a Models Evaluation Agent works.

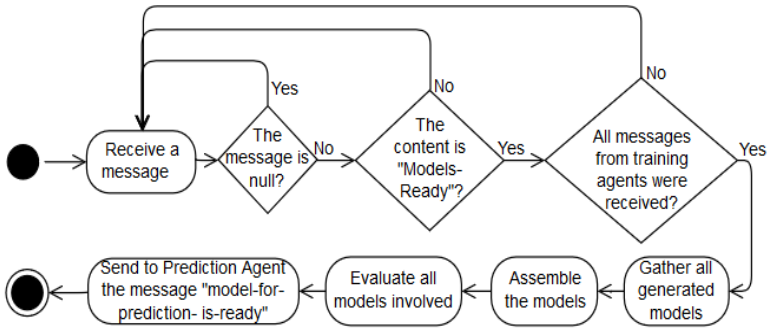

Figure 16. Models Evaluation Agent Activity Diagram

Prediction Agent: This agent evolved little from the first version. It takes new instances from an unlabeled dataset and, with the model provided after the Models Evaluation Agent´s result, will classify them. It contains a behaviour of type CyclicBehaviour that is continuously executed waiting this time to receive the communication from the Models Evaluation Agent that the model is ready. When it receives it, it begins to perform the prediction task which will produce a set of new classified instances. Figure 17 describes how a Prediction Agent works.

In this version, to implement and deploy an application based on JAF4DM 2.0, it is required to implement at least one instance of each agent described above in order to successfully operate, with the exception of training agents. For practical results, more than one training agent must be implemented to achieve a meaningful data mining process, otherwise combining the models will not make any sense.

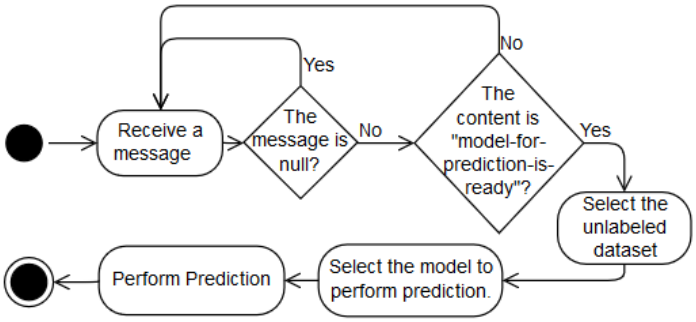

Figure 17. Prediction Agent Activity Diagram

4.2.4 The Process

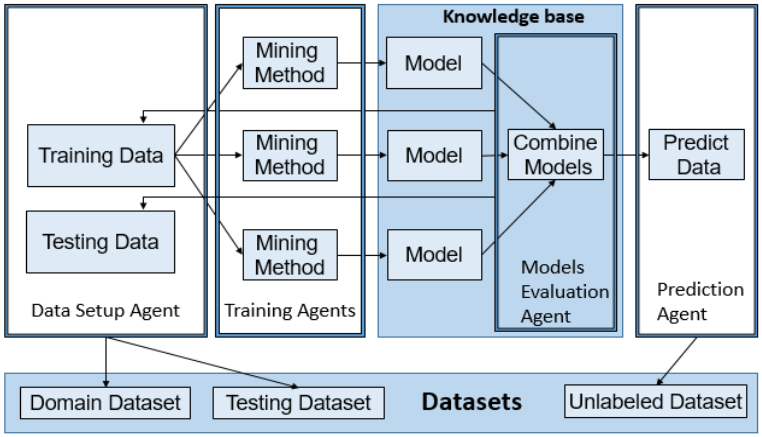

Figure 18. JAF4DM 2.0 process model

Given the fact that the agents involved were explained in detail in the previous section, it is simpler to understand how they cooperate as part of JAF4HDM´s work process as a whole. Figure 18 illustrates how the agents interact as part of that process, in which JAF4DM is described as an operational model. It represents a resource to make clear how the mining process work, where the data comes from and how it is used. It also outlines the stages in which knowledge (the models) is generated and used to generate more of it.

In harmony with the concepts described in Section 2.1, JAF4DM can be classified as a white box framework. In order to develop an application based on JAF4DM, a user needs to know at a basic level how to instantiate and import a JAVA class, the language on which the framework is based. Users also need to know about the mining algorithms that will be using. In the following chapter we describe several use scenarios in which specific applications are developed as JAF4DM instances.

5 Use Scenarios

In this chapter are described three use scenarios where JAF4DM was applied in order to test its applicability. As described at the beginning of this dissertation, our initial motivation came from seeing the health care field as a constant growing data generation source. Therefore, the scenarios in this section are in the health care domain. Each one of them is about diagnosis data related to a specific illness. Also, it is worth to notice that algorithms were picked randomly. To be a classification algorithm is the only criterion.

5.1 Hypothyroidism

This first use scenario is about Hypothyroidism diagnosis. The data correspond to a data set of thyroid disease records from 1987 and created at Garavan Institute and J. Ross Quinlan, New South Wales Institute, Sydney, Australia (Bache & Moshe, 2013). This dataset includes 3772 cases of thyroid disease diagnosis. Each of them is classified as compensated hypothyroid, primary hypothyroid, secondary hypothyroid or negative. In addition to the class, it counts with 29 attributes. The final output of this experiment is the prediction of new thyroid disease instances.

5.1.1 Instantiation

We developed an application based on JAF4DM that works with the aforementioned dataset. We call it JAF4HDM. It works with three classification algorithms to perform the training, PART (Eibe & Witten, 1998), Decision Table (Kohavi,1995) and OneR (Holte,1993), and will use the Stacking method (Wolpert, 1992) to produce a new model based on what resulted from training. Also, we provide the corresponding unlabeled dataset with new instances to be predicted. Figure 19 depicts the process that describes how JAF4HDM works in match with JAF4DM´s standard process defined in Section 4.2.

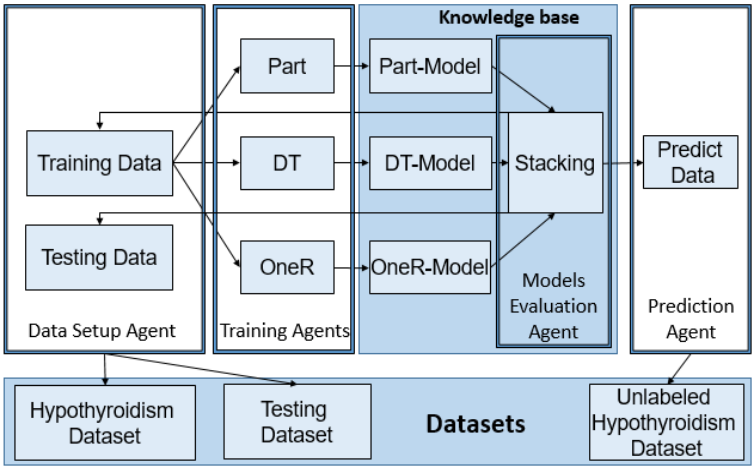

Figure 19. JAF4HDM Process

In this context, in order to implement an operational instance of JAFDM, six agents were instantiated as extensions from those defined in the framework (see Section 4.2): A data setup agent, three training agents (one for each classification algorithm), a models evaluation agent and a prediction agent. Figure 20 illustrates how the JAFDM hot-spots where extended in order to produce a new domain-specific application.

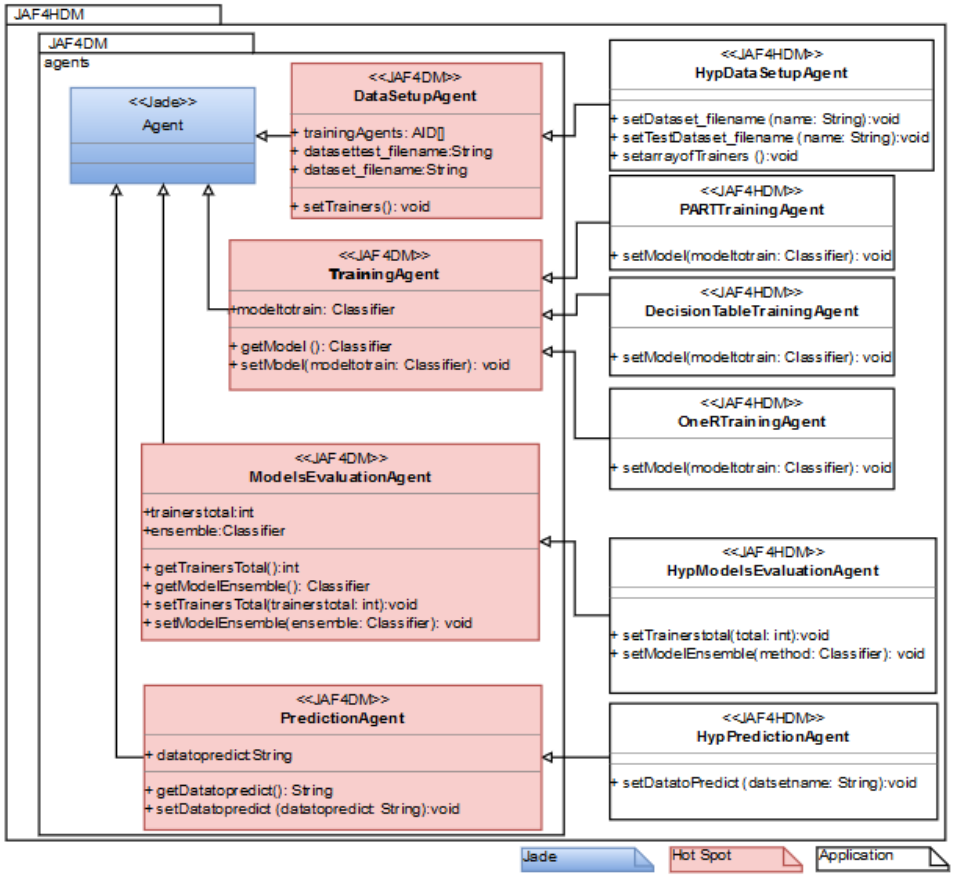

Figure 20. JAF4HDM, a JAF4DM instance

5.1.2 Results

The JAF4DM framework allows researchers to access the quality of all involved models according to different criteria. Users specify a metric at the moment of instantiating JAF4DM. Then, when the application is executed, the agent carries out the evaluation and comparison of all models. Next, the best model evaluated according to that metric is the one that serves as input to perform the prediction task.

In the scenarios presented in this dissertation, we considered four metrics to measure the quality of the obtained models:

Accuracy: the number of correctly classified instances, expressed as a proportion of all instances (Witten et al., 2016);

Precision: the ratio between the number of true instances predicted as true and the total of true and false values predicted as true (Fawcett, 2006);

Recall: the ratio between the number of true instances predicted as true and the total of true values predicted as true and false (Fawcett, 2006);

F-Measure: $2 * \frac{Recall * Precision}{(Recall + Precision)}$ (Witten et al., 2016).

Table 2 depicts the results of these metrics for each model produced in the process. We choose **Accuracy** to perform the comparison between the models. Therefore, a set of 10 unlabeled samples were classified looking for cases of thyroid disease based on the model generated by the PART algorithm.

Table 2: JAF4HDM metrics by model

Metrics \ Models	***Decision Table***	***OneR***	***PART***	***Stacking***
Accuracy	0.9832	0.9584	0.9946	0.9938
Precision	0.8510	0.5517	0.9148	0.9130
Recall	0.9302	0.7441	1.0	0.9767
fMeasure	0.8888	0.6336	0.9555	0.9438

5.2 Diabetes

In this scenario we produce one JAF4DM-based application to work with diabetes related data. To do so, we use the Pima Indians Diabetes Dataset, produced at the National Institute of Diabetes and Digestive and Kidney Diseases from U.S.A., published in 1990 and available at (Bache & Moshe, 2013). This data set its

composed of 768 instances of medical records classified as negative or positive for a diabetes mellitus diagnosis. All records are from female patients at least 21 years old of Pima Indian heritage. It has eight attributes plus the class. The final output of the data mining process in this application is to predict whether new instances have diabetes.

5.2.1 Instantiation

JAF4DDM was simpler to build since it uses just JRip (Cohen, 1995) and IBK (Aha et al., 1991) classification algorithms to perform the training tasks and Vote (Kittler et al., 1998) to perform the combination step. We also provide the correspondent set of new instances to be predicted. Figure 21 describes how this application works following the standard JAF4DM´s process defined in Section 4.2.

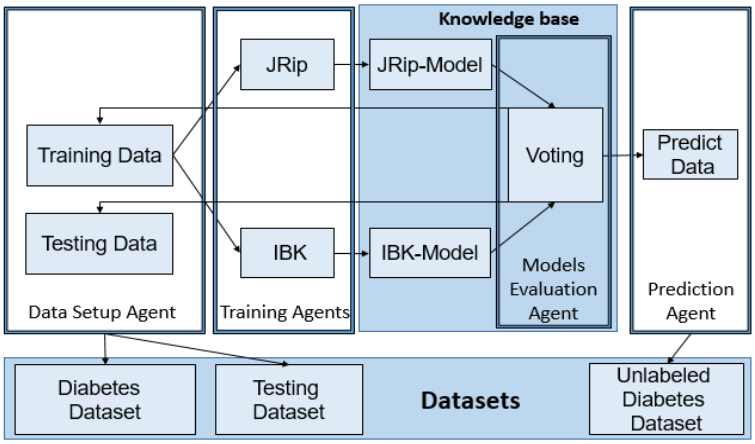

Figure 21. JAF4DDM Process

In this scenario, to implement JAF4DDM, five agents were instantiated as extensions from JAF4DM defined hot-spots (Section 4.2): A data setup agent, two training agents, a models evaluation agent and a prediction agent. Figure 22 depicts how these hot-spots where extended in order to produce JAF4DDM.

5.2.2 Results

For this scenario we considered the same metrics used in Subsection 5.1.2. This time, with the difference that we choose **Precision** metric to perform the comparison between the models.

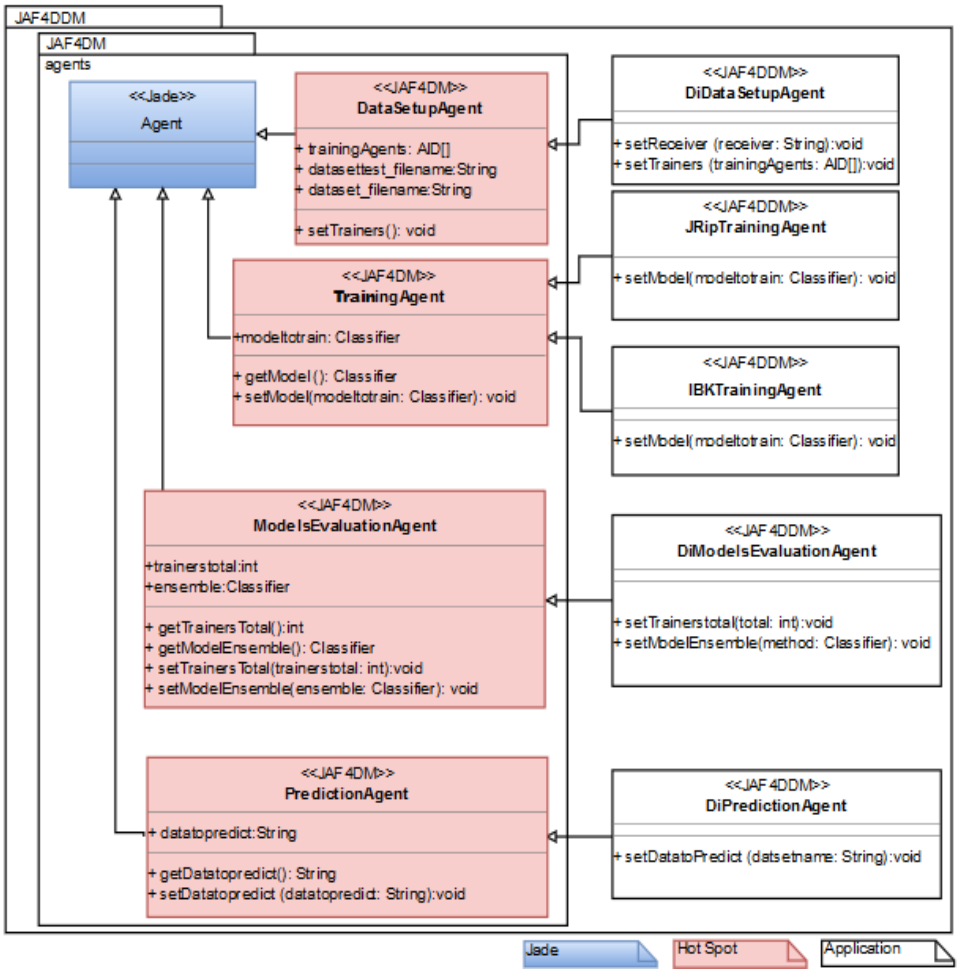

Figure 22. JAF4DDM, a JAF4DM instance

Table 3 depicts the performance of all models based on the same metrics. As can be appreciated, the model generated by the JRip algorithm presented the best **Precision** measure. Then, based on JRip, a set of five unlabeled cases were classified looking for Diabetes occurrences.

Table 3: JAF4DDM metrics by model

Metrics \ Models	***JRip***	***IBk***	***Vote***
Accuracy	0.7782	0.7043	70.0
Precision	0.675	0.5714	0.5657
Recall	0.6835	0.5569	0.5443
fMeasure	0.6792	0.5641	0.5548

5.3 Arrhythmia

As third scenario, we have a cardiac arrhythmia dataset - available at (Bache & Moshe, 2013), which contains 452 Electrocardiography (ECG) records from

Turkish patients. According to its creators, it was created to produce a knowledge base to determine the type of arrhythmia from ECG recordings (Guvenir et al., 1997). Each instance is classified as one of 16 diagnostics for cardiac arrhythmia. Class 01 refers to 'normal', ECG classes 02 to 15 refers to different classes of arrhythmia and class 16 refers to the rest of uncategorized ones. It contains 279 attributes plus the class. The final output of the data mining process in this JAF4DM-based application is to predict whether new ECG record instances have cardiac arrhythmia and from which type.

5.3.1 Instantiation

In this scenario, we use again the Stacking method (Wolpert, 1992) to perform the model´s combination. This time, the instance´s name is JAF4ADM, and it will perform four training process by executing LMT (Landwehr et al., 2005), JRip, J48 (Quinlan, 2014) and KStar (Cleary & Trigg, 1995) classification algorithms. Also, the involved data sets are also provided Figure 23 describes JAF4ADM´s operational flow according to the JAF4DM process.

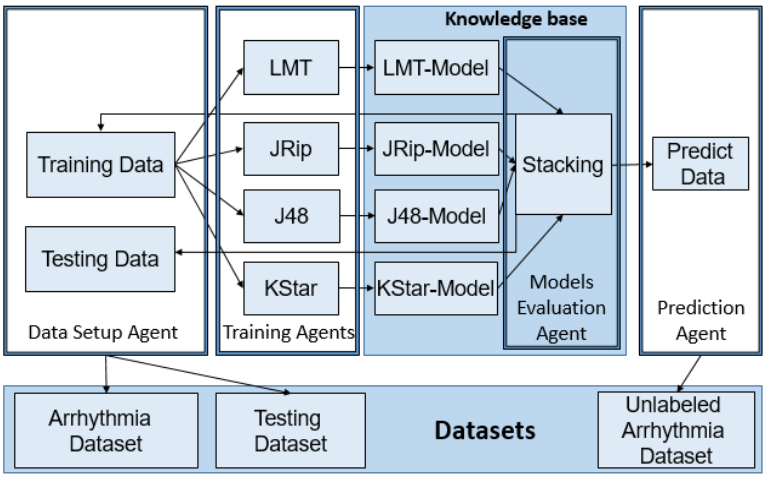

Figure 23. JAF4ADM Process

To implement JAF4ADM, seven agents were instantiated: One data setup agent, four training agents, a models evaluation agent and a prediction agent. Figure 24 depicts how these hot-spots where extended in order to produce JAF4ADM.

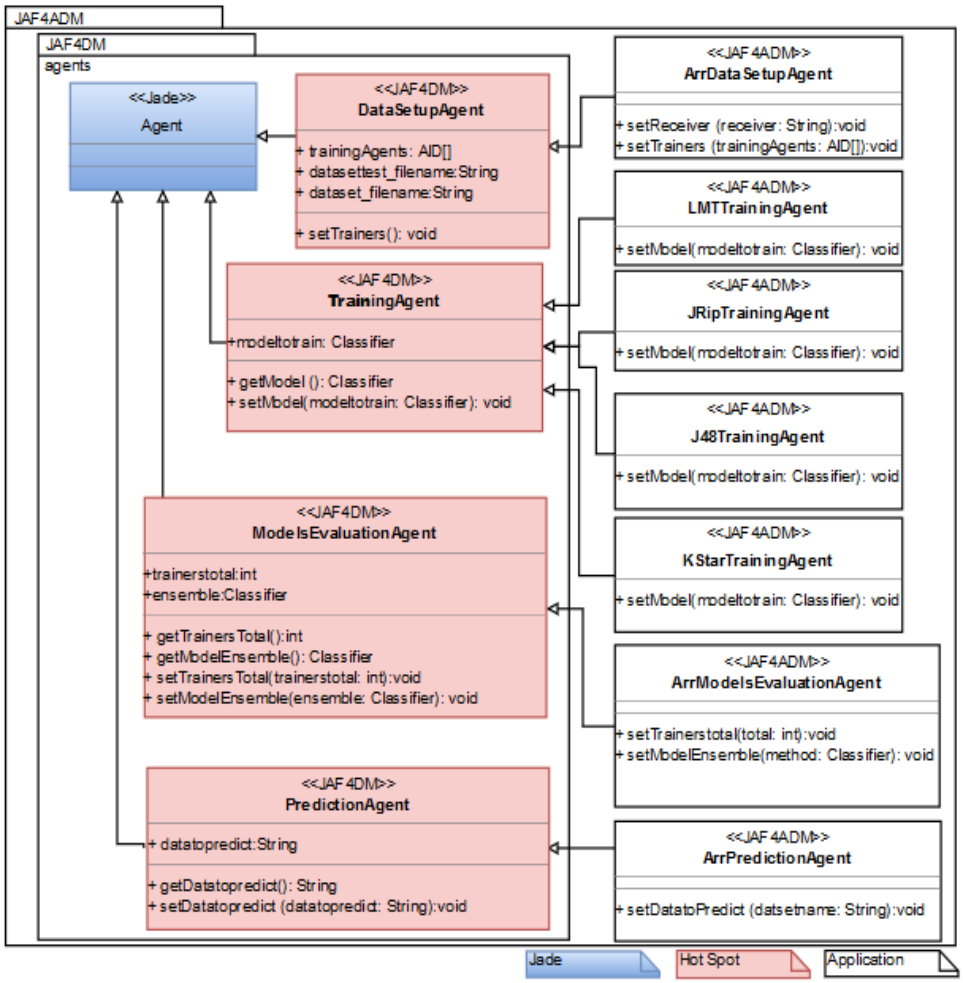

Figure 24. JAF4ADM, a JAF4DM instance

5.3.2 Results

In this scenario, we chose the same metrics used in Hypothyroidism and Diabetes scenarios and also the Accuracy metric to perform the comparison between the models. Table 4 depicts the performance of all the involved models. Based on JRip — the most accurate model — a set of 11 (eleven) unlabeled cases were classified as one the 16 groups of Arrhythmia.

Table 4: JAF4ADM metrics by model

Metrics	*J48*	*JRip*	*KStar*	*LMT*	*Stacking*
Accuracy	0.5220	0.6397	0.5220	0.6397	0.5294
Precision	0.1875	0.4545	0.5	0.5	0.3636
Recall	0.2142	0.3571	0.1428	0.5714	0.2857
fMeasure	0.1999	0.4	0.2222	0.5333	0.32

5.4 Discussion

In this chapter we described three use scenarios on which the applicability of our framework was successfully tested. For each one of them, we implemented a JAF4DM instance which works with diagnosis data related to a specific disease. For each scenario, the corresponding application worked well and the software agents also performed the mining process as expected. However, there are some elements that need to be discussed.

As we cleared at the very beginning of this chapter, we picked the data mining algorithms randomly, having as only criterion that they were classification algorithms. Beyond that, the selection of the algorithms is not relevant per se, because it depends highly on the human factor, on the expert's knowledge. At this point, it is good to remember the statement referenced in section 2.5: *there is no universally best data mining method, choosing a particular algorithm for a particular application is something of an art* (Fayyad et al., 1996). It is therefore advisable to have an area expert assisting the selection of such algorithms.

As already mentioned in the definition of this dissertation´s problem — as also in chapters 3 and 4 — several authors (Di Stefano & Menzies, 2002; Woźniak et al., 2014; Alpaydin,1998) defend the use of several data mining algorithms and ensemble techniques, arguing that it offers more reliable results than the use of a single algorithm may offer. Nevertheless, the execution of each JAF4DM-based application shows (as seen in Tables 2, 3, and 4) that ensemble algorithms do not offer the best result for any of the metrics considered to measure the quality of the obtained models. In this context, it is worth to notice that the objective of this dissertation is not to demonstrate that ensemble techniques offer the best result according to a certain metric. Our framework offers the possibility of evaluating all the models involved, regardless of whether they were generated by an ensemble technique or by individual data mining algorithms.

6 Conclusion and Future Work

This dissertation proposes JAF4DM, a software framework to develop applications which are able to perform agent-driven data mining processes. It provides support to build and operate software agents to interact and perform tasks such as: (i) data preparation; (ii) parallel mining; (iii) combination and evaluation of models, and (iv) classification of unlabeled instances. Accompanying this tool there is an architecture that serves as a guide to build applications based on it. Such a solution comes from the evolution of an initial idea, a first version whose development and results already produced two publications (Morejón et al., 2017a; Morejón et al., 2017b).

Our work is not oriented to a specific domain. Having said that, the motivation to develop this work comes from seeing how health care services and systems are an expanding source of large volumes of information about patient health and health care processes (Jensen et al., 2012; Szewczyk, 2016), given the importance and benefits of applying data mining algorithms in the health care industry (Durairaj & Ranjani, 2013). For these reasons, we considered three use scenarios related to health care domain to test JAF4DM´s applicability. Three applications based on JAF4DM were developed to deal with data related to Hypothyroidism, Diabetes and Arrhythmia conditions.

It is worth noticing that this work has some **limitations**:

i. It was designed only to perform predictive modeling — specifically classification;
ii. Since our work was more focused on Agent-oriented Software Engineering than on Data Mining. The JAF4DM framework was developed only using the WEKA API to provide the data mining component. There is no any integration with more popular tools across employed data scientists, such as Python and R (Kaggle, 2017);
iii. JAF4DM is a white box framework. Therefore, it is necessary for users to have basic programing skills to develop new instances;

iv. The proper selection of algorithms is largely dependent on the human factor. The framework itself does not guarantee the quality of the result, moreover, it offers the possibility of evaluating all the options defined.

For **future work** there are some elements that we have as goals to improve the scope and the applicability of this solution:

i. We aim at building more JAF4DM-based applications for other scenarios to increase evidences of the applicability of our framework;

ii. We also aim at developing a grey-box version of JAF4DM to make the instantiation process more user friendly;

iii. We are planning to develop the capability to perform other data mining tasks such as Regression and Clustering, among others;

iv. We offered WEKA API as a good integration to comply the agent-mining approach in this work. Nevertheless, it would be remarkable to build a version of this framework, based on the same proposed architecture, in which the data mining component were totally based on a data science tool such as Python, which is the most commonly used across employed data scientists (Kaggle, 2017).

References

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. **Machine Learning**, v. 6, n. 1, p. 37–66, 1 jan. 1991.

ALPAYDIN, E. **Techniques for Combining Multiple Learners**. Proceedings of Engineering of Intelligent Systems. ICSC Press, 1998

AYODELE, T. O. Types of machine learning algorithms. In: **New advances in machine learning.** [s.l.] InTech, 2010.

AZEVEDO, A. I. R. L.; SANTOS, M. F. KDD, SEMMA and CRISP-DM: a parallel overview. **IADS - DM**, 2008.

BACHE, K.; LICHMAN, M. **UCI machine learning repository**, 2013. Available at: <https://archive.ics.uci.edu/ml/index.php>

BELLIFEMINE, F. et al. **Jade programmer's guide**TILab, 2002. Available at: <http://ltodi.est.ips.pt/hgamboa/IAD2004/jadedocs/administratorsguide.pdf>

BELLIFEMINE, F. et al. **Jade administrator's guide.**TILab, 2003. Available at: <http://ltodi.est.ips.pt/hgamboa/IAD2004/jadedocs/administrators guide.pdf>

BELLIFEMINE, F. et al. JADE: A software framework for developing multi-agent applications. Lessons learned. **Information and Software Technology**, Special issue with two special sections. Section 1: Most-cited software engineering articles in 2001. Section 2: Requirement engineering: Foundation for software quality. v. 50, n. 1, p. 10–21, 1 jan. 2008.

BELLIFEMINE, F. L.; CAIRE, G.; GREENWOOD, D. **Developing Multi-Agent Systems with JADE**. [s.l.] John Wiley & Sons, 2007. v. 7

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. **JADE- A FIPA compliant agent framework**. Proceedings of PAAM. 1999

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. **JADE: A FIPA2000 Compliant Agent Development Environment**. Proceedings of the Fifth International Conference on Autonomous Agents. AGENTS '01. New York, NY, USA: ACM, 2001 Available at: <http://doi.acm.org/10.1145/ 375735.376120>

CAO, L.; GORODETSKY, V.; MITKAS, P. A. Agent Mining: The Synergy of Agents and Data Mining. **IEEE Intelligent Systems**, v. 24, n. 3, p. 64–72, maio 2009.

CAVANILLAS, J. M.; CURRY, E.; WAHLSTER, W. The Big Data Value Opportunity. In: **New Horizons for a Data-Driven Economy**. [s.l.] Springer, Cham, 2016. p. 3–11.

CHRISTA, S.; MADHURI, K. L.; SUMA, V. A Comparative Analysis of Data Mining Tools in Agent Based Systems. **arXiv:1210.1040 [cs]**, 3 out. 2012.

CLEARY, J. G.; TRIGG, L. E. K*: An Instance-based Learner Using an Entropic Distance Measure. In: PRIEDITIS, A.; RUSSELL, S. (Eds.) **Machine Learning Proceedings 1995**. San Francisco (CA): Morgan Kaufmann, 1995. p. 108–114.

COHEN, W. W. Fast Effective Rule Induction. In: PRIEDITIS, A.; RUSSELL, S. (Eds.). **Machine Learning Proceedings 1995**. San Francisco (CA): Morgan Kaufmann, 1995. p. 115–123.

COLUMBUS, L. Roundup of Internet of Things Forecasts and Market Estimates, 2016. **FORBES**, nov. 2016.

DI STEFANO, J. S.; MENZIES, T. **Machine learning for software engineering: case studies in software reuse**. 14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings. In: 14TH IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, 2002. (ICTAI 2002). PROCEEDINGS. 2002

DURAIRAJ, M.; RANJANI, V. Data Mining Applications in Healthcare Sector: A Study. **International Journal of Scientific & Technology Research**, v. 2, n. 10, p. 29–35, 25 out. 2013.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, ROC Analysis in Pattern Recognition. v. 27, n. 8, p. 861–874, 1 jun. 2006.

FAYAD, M.; SCHMIDT, D. C. Object-oriented Application Frameworks. **Commun. ACM**, v. 40, n. 10, p. 32–38, out. 1997.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. **Commun. ACM**, v. 39, n. 11, p. 27–34, nov. 1996.

FAYYAD, U.; STOLORZ, P. Data mining and KDD: Promise and challenges. **Future Generation Computer Systems**, Data Mining. v. 13, n. 2, p. 99–115, 1 nov. 1997.

FRANK, E. et al. The WEKA Data Mining Workbench. In: **Data Mining: Practical Machine Learning Tools and Techniques**. [s.l.] Morgan Kaufmann, 2016. p. 401–585.

FRANK, E.; WITTEN, I. H. **Generating accurate rule sets without global optimization**. [s.l.] University of Waikato, Department of Computer Science, jan. 1998. Available at: <https://researchcommons.waikato.ac.nz/handle/10289/1047>.

GAO, J.; DENZINGER, J.; JAMES, R. C. **A Cooperative Multi-agent Data Mining Model and Its Application to Medical Data on Diabetes**. Autonomous Intelligent Systems: Agents and Data Mining. Lecture Notes in Computer Science. In: INTERNATIONAL WORKSHOP ON AUTONOMOUS INTELLIGENT SYSTEMS: AGENTS AND DATA MINING. Springer, Berlin, Heidelberg, 6 jun. 2005a Available at: <https://link.springer.com/chapter/ 10.1007/11492870_8>. Acesso em: 1 abr. 2018

GAO, J.; DENZINGER, J.; JAMES, R. C. **CoLe: a cooperative data mining approach and its application to early diabetes detection**. Fifth IEEE International Conference on Data Mining (ICDM'05). In: FIFTH IEEE INTERNATIONAL CONFERENCE ON DATA MINING (ICDM'05). nov. 2005b

GONZALEZ-SANCHEZ, J. et al. **ABE: An Agent-Based Software Architecture for a Multimodal Emotion Recognition Framework**. 2011 Ninth Working IEEE/IFIP Conference on Software Architecture. In: 2011 NINTH WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE. jun. 2011

GORODETSKY, V. et al. **Multi-Agent Information Fusion: Methodology, Architecture and Software Tool for Learning of Object and Situation Assessment**. Proceedings of the Seventh International Conference on Information Fusion. In: FUSION 2004. Stockholm, Sweden: 2014

GORODETSKY, V.; KARSAEYV, O.; SAMOILOV, V. **Multi-agent technology for distributed data mining and classification**. IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003. In: IEEE/WIC INTERNATIONAL CONFERENCE ON INTELLIGENT AGENT TECHNOLOGY, 2003. IAT 2003. out. 2003

GUVENIR, H. A. et al. **A supervised machine learning algorithm for arrhythmia analysis**. Computers in Cardiology 1997. In: COMPUTERS IN CARDIOLOGY 1997. set. 1997

HALL, M. et al. The WEKA Data Mining Software: An Update. **SIGKDD Explor. Newsl.**, v. 11, n. 1, p. 10–18, nov. 2009.

HOLTE, R. C. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. **Machine Learning**, v. 11, n. 1, p. 63–90, 1 abr. 1993.

JAGADISH, H. V. et al. Big Data and Its Technical Challenges. **Commun. ACM**, v. 57, n. 7, p. 86–94, jul. 2014.

JENNINGS, N. R.; WOOLDRIDGE, M. Applications of Intelligent Agents. In: **Agent Technology**. [s.l.] Springer, Berlin, Heidelberg, 1998. p. 3–28.

JENSEN, P. B.; JENSEN, L. J.; BRUNAK, S. Mining electronic health records: towards better research applications and clinical care. **Nature Reviews Genetics**, v. 13, n. 6, p. 395–405, jun. 2012.

KAGGLE. **The State of Data Science & Machine Learning. 2017**. [s.l.] KAGGLE, [s.d.]. Available at: <https://www.kaggle.com/surveys/2017>. Acesso em: 25 mar. 2018.

KITTLER, J. et al. On combining classifiers. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 20, n. 3, p. 226–239, mar. 1998.

KOHAVI, R. **The power of decision tables**. Machine Learning: ECML-95. Lecture Notes in Computer Science. In: EUROPEAN CONFERENCE ON MACHINE LEARNING. Springer, Berlin, Heidelberg, 25 abr. 1995Available at: <https://link.springer.com/chapter/10.1007/3-540-59286-5_57>. Acesso em: 1 abr. 2018

KRUEGER, C. W. Software Reuse. **ACM Comput. Surv.**, v. 24, n. 2, p. 131–183, jun. 1992.

LANDWEHR, N.; HALL, M.; FRANK, E. Logistic Model Trees. **Machine Learning**, v. 59, n. 1–2, p. 161–205, 1 maio 2005.

MARKIEWICZ, M. E.; LUCENA, C. J. P. Object Oriented Framework Development. **Crossroads**, v. 7, n. 4, p. 3–9, jul. 2001.

MINIWATTS MARKETING GROUP. **Internet World Stats.** [s.l: s.n.]. Available at: <https://www.internetworldstats.com/stats.htm>.

MITCHELL, T. M. **Machine learning. WCB.** [s.l: s.n.].

MITKAS, P. A. et al. **A Framework for Constructing Multi-agent Applications and Training Intelligent Agents**. Agent-Oriented Software Engineering IV. Lecture Notes in Computer Science. In: INTERNATIONAL WORKSHOP ON AGENT-ORIENTED SOFTWARE ENGINEERING. Springer,

Berlin, Heidelberg, 15 jul. 2003 Available at: <https://link.springer.com /chapter/10.1007/978-3-540-24620-6_7>. Acesso em: 1 abr. 2018

MOREJÓN, R.; VIANA, M.; LUCENA, C. An Approach to Generate Software Agents for Health Data Mining. **International Journal of Software Engineering and Knowledge Engineering**, v. 27, n. 09n10, p. 1579–1589, 1 nov. 2017.

MOREJON, R.; VIANA, M.; LUCENA, C. J. **Generating Software Agents for Data Mining: An Example for the Health Data Area**. In: 29TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING & KNOWLEDGE ENGINEERING(SEKE'2017). PA, USA: SEKE/Knowledge Systems Institute, 5 jul. 2017Available at: <http://ksiresearchorg.ipage.com/seke/ seke17paper/seke17paper_86.pdf>

NWANA, H. S. Software agents: an overview. **The Knowledge Engineering Review**, v. 11, n. 3, p. 205–244, set. 1996.

PALANIAPPAN, S.; AWANG, R. **Intelligent heart disease prediction system using data mining techniques**. Computer Systems and Applications, 2008. AICCSA 2008. In: 2008 IEEE/ACS INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND APPLICATIONS. mar. 2008

PIATETSKY-SHAPIRO, G. **Advances in Knowledge Discovery and Data Mining - Semantic Scholar**. Menlo Park: AAAI press, 1996. v. 21

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. [s.l.] Elsevier, 2014.

RAGHUPATHI, W. Data mining in healthcare. In: **Healthcare Informatics: Improving Efficiency through Technology, Analytics, and Management**. [s.l.] CRC Press, 2016. p. 353–372.

RALHA, C. G.; SARMENTO SILVA, C. V. A multi-agent data mining system for cartel detection in Brazilian government procurement. **Expert Systems with Applications**, v. 39, n. 14, p. 11642–11656, 15 out. 2012.

SZEWCZYK, P. Impact of the Internet of Things on the economy and society. **Zeszyty Naukowe. Organizacja i Zarządzanie / Politechnika Śląska**, n. z. 93, p. 461--470, 2016.

VILLARS, R. L.; EASTWOOD, M.; OLOFSON, C. W. Big Data: What It Is and Why You Should Care. p. 14, 2011.

WANG, J. **Data Mining: Opportunities and Challenges**. [s.l.] IGI Global, 2003.

WANG, Y.; WITTEN, I. H. **Induction of model trees for predicting continuous classes**. [s.l: s.n.]. Available at: <https://researchcommons.waikato.ac.nz/handle/10289/1183>. Acesso em: 1 abr. 2018.

WITTEN, I. H. et al. **Data Mining: Practical Machine Learning Tools and Techniques**. [s.l.] Morgan Kaufmann, 2016.

WOLPERT, D. H. Stacked generalization. **Neural Networks**, v. 5, n. 2, p. 241–259, 1 jan. 1992.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: theory and practice. **The Knowledge Engineering Review**, v. 10, n. 2, p. 115–152, jun. 1995.

WOŹNIAK, M.; GRAÑA, M.; CORCHADO, E. A survey of multiple classifier systems as hybrid systems. **Information Fusion**, Special Issue on Information Fusion in Hybrid Intelligent Fusion Systems. v. 16, p. 3–17, 1 mar. 2014.

ZWOLENSKI, M.; WEATHERILL, L. The digital universe: Rich data and the increasing value of the internet of things. **Australian Journal of Telecommunications and the Digital Economy**, v. 2, n. 3, p. 47, set. 2014.